4-2024

Robust Machine Learning Under Vulnerable Cyberinfrastructure and Varying Data Quality

Sergei Chuprov
sc1723@rit.edu

Robust Machine Learning Under Vulnerable Cyberinfrastructure and Varying Data Quality

by

Sergei Chuprov

A dissertation submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in Computing and Information Sciences

B. Thomas Golisano College of Computing and
Information Sciences

Rochester Institute of Technology
Rochester, New York
April 2024

Robust Machine Learning Under Vulnerable Cyberinfrastructure and Varying Data Quality

by

Sergei Chuprov

Committee Approval:

We, the undersigned committee members, certify that we have advised and/or supervised the candidate on the work described in this dissertation. We further certify that we have reviewed the dissertation manuscript and approve it in partial fulfillment of the requirements of the degree of Doctor of Philosophy in Computing and Information Sciences.

Dr. Leon Reznik — Date
Dissertation Advisor

Dr. Igor Khokhlov — Date
Dissertation Committee Member

Dr. Stanislaw Radziszowski — Date
Dissertation Committee Member

Dr. Ivan De Oliveira Nunes — Date
Dissertation Committee Member

Dr. Sergey Lyshevski — Date
Dissertation Defense Chairperson

Certified by:

Dr. Pengcheng Shi — Date
Ph.D. Program Director, Computing and Information Sciences

Robust Machine Learning Under Vulnerable Cyberinfrastructure and Varying Data Quality

by

Sergei Chuprov

Submitted to the
B. Thomas Golisano College of Computing and Information Sciences Ph.D. Program in Computing and Information Sciences
in partial fulfillment of the requirements for the
Doctor of Philosophy Degree
at the Rochester Institute of Technology

Abstract

In our study, we investigate Machine Learning (ML) application robustness in **ML I**ntegrated with **N**etwork (**MLIN**) systems. We consider MLIN as an integration of three major components and interrelations between them: data sources, network facilities, and ML application. In contrast to the conventional approaches that focus on each separate component, we concentrate on their interrelationships, and consider them from the system integration perspective. We formulate our primary goal as to develop methods and tools aimed at assuring ML application performance towards Data Quality (DQ) variation in MLIN through improving ML application robustness. We examine the prior work on ML robustness definition, evaluation, and enhancement, and discuss existing challenges. As our first major contribution, we propose a novel approach to define and evaluate ML robustness towards DQ variations in MLIN that enables addressing these challenges. We develop ML robustness calculus based on the relationship between the quality of the input data and ML performance demonstrated over this input. As another major contribution, we examine and develop methods and tools to ensure ML performance through improving ML robustness in MLIN. With the integrated MLIN architecture in mind, we represent our third major contribution in which we develop a reactive MLIN feedback mechanism aimed at providing MLIN system restructuring recommendations in order to improve ML performance in the presence of DQ variations. In our fourth contribution, we expand the robustness from the ML execution to the ML training phase. We investigate the feasibility of proactive strategies, such as Transfer and Federated Learning, applied at the ML training phase in order to enhance ML performance to DQ variations during the ML execution, and the security of MLIN systems. We address security vulnerabilities posed by these strategies when applied in MLIN and develop Reputation and Trust-based techniques that allow to enhance the security and, in turn, improve ML robustness. We investigate multiple real-world use cases to verify the developed solutions in practice. Our practical studies embrace diverse

data modalities including images, sounds, voice recordings, videos, and the conventional qualitative and quantitative data represented in a table format. We examine various industrial open-source and commercial ML tools designated for processing data in such areas as computer vision, sound classification, voice recognition and transcription, and video object detection and classification.

Acknowledgments

This dissertation would not have been possible without the unwavering support and guidance of a remarkable group of individuals, who significantly contributed to shaping and maturing my research endeavor.

First and foremost, I extend my deepest gratitude to my advisor, Dr. Leon Reznik, Professor of Computer Science and Computing Security at the Rochester Institute of Technology (RIT). His exceptional knowledge, insightful advice, and unwavering dedication have been instrumental in shaping this research and propelling me forward throughout my doctoral journey.

I am incredibly grateful to my dissertation committee members. Dr. Stanislaw Radziszowski, Professor in the Department of Computer Science at RIT, provided valuable feedback and encouraged me to delve deeper into critical aspects of my research. Dr. Ivan De Oliveira Nunes, Assistant Professor of Cybersecurity at RIT, offered his expertise and steered me towards valuable research avenues. Dr. Igor Khokhlov, Assistant Professor of Cybersecurity at Sacred Heart University, CT, brought his fresh perspective and offered his prudent advice that strengthened the foundation of my work.

I want to thank the team of the University of Utah in partnership with Salt Lake City, and the Utah Education and Telehealth Network for their assistance and guidance on using the POWDER platform. The access to POWDER has allowed us to conduct our empirical study on real devices and the wireless network established over real terrain.

My sincere appreciation goes to Dr. Sergey Lyshevski, Professor at the Department of Electrical and Microelectronic Engineering at RIT. His guidance was valuable in many aspects and during the whole period of my doctoral journey.

I am especially grateful to Dr. Pengcheng Shi, Director, and Dr. Min-Hong Fu, Senior Staff Specialist at the Computing and Information Sciences Ph.D. Program at RIT, and to all the program staff for their unwavering support, navigation through the administrative processes, and assistance throughout my doctoral studies.

My heartfelt thanks go to Dr. Ernest Fokoue, Professor at the School of Mathematics and Statistics at RIT. His brilliance and unwavering support have been a constant source of inspiration and encouragement in both my academic and personal endeavors.

I am deeply grateful to Dr. Richard Zanibbi, Professor of Computer Science at RIT, for his guidance, feedback, eagerness to help, and encouragement throughout my research journey.

My sincere appreciation goes to Dr. Roman Yampolskiy, Associate Professor of Computer Science and Engineering at the University of Louisville, KY, for his valuable insights related to computing security and safety, and general support throughout my research.

My doctoral journey would not have been as enriching in its initial stage without the instrumental collaboration of Dr. Ilia Viksnin, and Dr. Igor Komarov. Their expertise and guidance have significantly contributed to the success of this research.

I extend my deepest gratitude to the incredible group of researchers, students, and individuals with whom I collaborated throughout my research endeavor: Dr. Garegin Grigoryan, Raman Zatsarenko, Dmitrii Korobeinikov, Anirudh Narayanan, Shivam Mahajan, Chirayu Anil Marathe, Moinuddin Memon, Kartavya Manojbhai Bhatt, Rahul Ganesh, Matthew Hyland, John Flory, Antoun Obeid, Akshaya Nandkishor Satam, Srujan Shetty, Ruslan Gataullin, Iuliia Kim, Egor Marinenkov, Pavel Belyaev, Evgenii Neverov, Nikita Tursukov, Eduard Lazarev, Timofey Melnikov, Maria Usova, Julia Lyakhovenko, Dr. Kseniya Salakhutdinova, and many others. Your dedication, hard work, and insightful contributions have been invaluable in achieving the research results presented in this dissertation.

To all of you, my deepest gratitude.

To my beloved wife Daria, your enduring support and constant faith have been the bedrock of this journey. To my parents and family, your everlasting encouragement fueled my perseverance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Machine Learning (ML) systems and information and communication technologies are evolving rapidly, enabling new applications and services that rely on data-driven solutions to optimize processes in various domains. For example, Intelligent Transportation Systems (ITS) [51, 52, 54, 56] integrate computer vision and other technologies to regulate traffic flows, reduce road congestion, and improve road users safety. These systems commonly employ cameras, sensors, and communication devices to collect and transmit data about the traffic conditions, and ML models to analyze the data and provide real-time feedback to the Road Side Units (RSUs) and traffic participants. The evolution of these technologies also poses new challenges for the design, development, and maintenance of integrated systems that consist of various components from different domains. These components may have distinct objectives, assumptions, and constraints, and may interact with each other in complex and unpredictable ways. As in practice these components are already integrated into a single architecture aimed at facilitating the ML application needs, in the design, implementation, and maintenance stages, separate components are commonly considered in isolation. This approach prevents from considering the interrelationships between various components, which disintegrates the whole system and decreases its entropy. Initial consideration of the components from the system integration perspective will allow more informed and effective system design, aimed at enhancing the performance of the targeted application. From here, the roots of our research's motivation stem.

In our work, we study complex ML application systems that integrate multiple components. We refer to these systems as ML with Integrated Network (MLIN) systems, as they involve the integration of data sources, cyberinfrastructure that incorporate network facilities, and ML applications.

MLIN systems are becoming more prevalent in various domains, such as smart cities, health care, e-commerce, and education. However, there is a lack of systematic methods and tools to design, develop, and maintain MLIN systems in robust and efficient ways. Existing methods and tools tend to focus on specific aspects or components of MLIN systems, such as Data Quality (DQ), ML performance, network reliability, or user satisfaction. However, these aspects or components are not independent from each other [46, 47]. Instead, in MLIN systems, they are interrelated and interdependent [43]. Therefore, we need to consider the integration between the separate components, the way how this integration affects each of these components, and how it affects the overall system performance.

We live in an era of data explosion, where massive amounts of data are generated every day from various sources, such as sensors and mobile devices. This data is employed to create more reliable and powerful ML-based systems to optimize various processes, e.g., planning better emergency routes during construction [228], improving the quality of terrain classification by autonomous vehicles [111], and others. According to some estimates, the global data volume will reach 180 zettabytes by 2025, a fourfold increase from 2018[1]. A wide diversity and heterogeneity of various data sources engender another challenge related to the quality and security of the produced data [20, 49, 99, 140, 230], and its fitness for the consumers. For example, the concept of smart factory, which also incorporates such technologies as ML-based systems and wireless networks, requires to maintain high DQ circulating in the system to successfully satisfy customers' needs and optimize manufacturing processes [135, 222, 223, 224]. Another instance are robotic and cyber-physical systems, which interact with real physical objects and humans, and are employed for various tasks, e.g., premises monitoring [146]. In these systems, maintaining the quality and security of data is crucial for correct and safe decision-making [242].

One of the key challenges in MLIN systems is to ensure that the DQ used by ML applications satisfies their requirements. DQ refers to the degree to which data satisfies the requirements of the application and the end user [107]. Poor DQ can result in ML performance decrease and erroneous decision-making, which, depending on the particular domain and application, can lead to unpleasant consequences or even jeopardize safety of the system's users [50, 62]. There are approaches that propose to autonomously adapt ML models to various objectives or data [156, 220], however, they require additional research before applying in industrial domains. For instance, in the healthcare domain, ML systems are widely employed to support diagnosis, prognosis, treatment, and prevention of diseases [47]. There are various data-driven ML application that can analyze medical images, such as X-rays or MRI scans, to detect anomalies. However, if the processed

[1] https://www.statista.com/statistics/871513/worldwide-data-created/

data input, or data used to train is of low quality, the ML models may fail to recognize the patterns of interest or produce false positives or negatives, which introduces risks for the patients' health and well-being [39]. Another example is ITS, which employs ML systems to optimize traffic management, road users safety, and road infrastructure efficiency. If the data used to train or test these ML systems is of low quality, the model may fail to provide accurate or reliable outputs [40]. This can have adverse effects on the traffic flow, congestion, and safety of the road users. DQ affects not only the technical aspects of ML systems but also the social and economic ones. DQ degradation leads to ML performance decrease, which can undermine the trustworthiness and credibility of ML systems and cause dissatisfaction or harm to the users and stakeholders [65]. Therefore, DQ management and assurance should be an integral part of ML system development and deployment. These real-world examples emphasize the importance of maintaining high DQ for ML systems in various domains and applications.

In MLIN systems, data may be affected by various components in each data life-cycle stage, such as data collection or retrieval, transmission, processing, and consumption. This poses significant challenges for contemporary ML applications to maintain robustness to these DQ variations. In ML, robustness usually refers to the ability of an ML model to maintain its performance when faced with uncertainties or adversarial conditions during processing the inputs. There are various approaches to define, measure, and improve robustness in ML, depending on the type and nature of the uncertainties or adversarial conditions. We discuss these approaches in detail in sec. 2.1. The most common approach to measure and quantify robustness is based on the distance between the original and the perturbed data sample, where the perturbed data is obtained by applying some noise, transformation, or adversarial attack against the original data [179]. In this case, the larger the distance that an ML model can tolerate without losing performance, the more robustness it possesses. However, this approach does not fully capture the DQ concept, as DQ is a complex characteristic that encompasses intrinsic data properties, contextual attributes, and the security and reliability of the data source [107].

In our research, we propose a novel approach to define, measure, and calculate ML robustness specifically for MLIN systems. Our approach is based on the relationship between the DQ of the data input and ML performance demonstrated by the ML application over this data input. Rather than focusing only on the separate MLIN components, we consider robustness as a property of the integral system. We also consider robustness during the execution phase of the ML model, when all the training and pre-training procedures are completed and the model processes unseen data. By taking into account the interrelationships between the MLIN components, we derive knowledge on how they affect each other and how they influence the overall ML application performance.

In our work, we introduce two types of methods to assure robustness in MLIN systems: reactive and proactive. Reactive methods are based on generating and applying feedback actions to restructure or reconfigure the MLIN system in order to satisfy the requirements established by the user and ML application. Proactive methods are based on applying preventive measures at the ML model training stage, targeted to enhance ML robustness during the execution phase. We verify the investigated methods on a real-world use cases to illustrate our approach feasibility and effectiveness. In the next section, we discuss our major research objective and the Research Questions (RQs) we approach in our work.

1.1 Research Goal and Major Research Questions

The **primary goal of our research is to develop methods and tools aimed at ensuring ML application performance towards DQ variation in MLIN through improving ML application robustness.** The proposed approach to ML application robustness evaluation is based on the relationship between the DQ of the input and the ML performance demonstrated over this input. That is being said, the ML application robustness directly depends on the performance level demonstrated by the application. As the method to ensure ML application performance, we introduce the MLIN feedback component, aimed at producing recommendations on adjusting MLIN structure in order to satisfy the requirements to ML performance specified by the user and ML application.

On the way to achieving the established goal, we have to investigate what impact the interrelationships between the integrated MLIN components have on the DQ, ML performance, and ML application robustness, and establish knowledge based on this investigation. Then, to verify the feasibility of our approach, we have to demonstrate how the knowledge on these interrelationships can be applied in practice to design the MLIN feedback component. To facilitate the achieving of our research goal, we setup multiple tasks, and concentrate each of them on answering the following major RQs in each chapter of our work.

1.1.1 RQ1. What methods and techniques should be employed to design the integrated MLIN system and formalize the interrelationships between its component's interactions and ML application robustness?

In chapter 2, we disclose our motivation accounting for the overall research, and conduct a feasibility study and classification of the existing approaches to define, measure, and quantify ML robustness. We define the scope of the considered system we work with, and develop its high-level architecture. We employ functional and architectural modeling methods to design the major components and their functions in MLIN, and to represent the major data life-cycles and information flows. With the designed integrated structure in mind, we employ system theory methods and techniques to define and develop the generic calculus for the DQ evaluation in MLIN. We determine how the data in MLIN might be affected by diverse technical, non-technical, and malicious factors while being processed by various components, and how this might influence the other MLIN components, and the performance of the ML application. We employ the developed interrelationships and indicators to provide our's ML application robustness definition, which is based on the input DQ and ML application performance demonstrated over this input. To further improve MLIN system security, we define and develop generic calculus for the Reputation and Trust techniques that incorporate DQ evaluation in order to detect untrustworthy and potentially compromised data sources. RQ1 can be subdivided into the following questions:

- What classification is feasible for the existing approaches to define, assess, and quantify ML application robustness, and what major benefits and disadvantages they posses?
- How effective are the existing approaches to ML robustness in addressing the DQ variation in MLIN systems?
- Which methods and techniques should be employed to design the integrated MLIN system architecture and incorporate the ML adjustment feedback component into this architecture?
- What calculus and metrics are feasible to measure and represent the DQ variations in the integrated MLIN system?
- How do the introduction and use of Reputation and Trust metrics contribute to increasing MLIN security?
- What calculus should be developed to effectively measure ML robustness that integrates the input DQ and ML performance demonstrated over this input?

1.1.2 RQ2. How to improve ML application robustness by data sources selection and adaptation?

In chapter 3, we develop an innovative approach to intelligent data source selection within the integrated MLIN system. We concentrate the major novelty of our approach on the integration of DQ with the security characteristics of the platform in which the data source is embedded, shifting the focus of DQ evaluation from quality alone to aggregating the security-related aspects, which proved to be critical in practical applications [107]. We tackle the challenges posed by data multimodality and diverse origins by developing a generic DQ calculus capable of evaluating data from multiple platforms. To enable real-time intelligent data source selection, we incorporate a Genetic Algorithms (GA)-based data sources selection technique with our integrated DQ and platform security calculus, making DQ a primary optimization parameter. We integrate the developed solutions into an Integration Framework for Data Source Selection, validated in a practical use case with the DQ characteristics derived from real-world diverse mobile devices, demonstrating superior performance in terms of computational efficiency and comparable DQ in contrast to the conventional brute-force search. We develop software methods and tools that realize our solution, implemented in multiple Android OS applications available for public use. In addition, we collect and make available the knowledge base on characteristics of real data sources and their platforms, making our contributions accessible to a wider community. Below we formulate the sub-questions answered in this chapter.

- What novel methodological framework should be developed for automatic data sources selection in MLIN that provides integration of platforms, data sources modalities, and diverse metrics?
- How to verify the developed novel framework in practice?
- How effective and efficient is the developed novel framework in comparison to the conventional approaches to data sources selection?
- How to assist and benefit the community in adopting the developed framework in practice?

1.1.3 RQ3. How to improve ML application robustness by the network infrastructure adaptation?

In chapter 4, we conduct an extensive investigation of multiple diverse real-world use cases to examine the impact of the MLIN cyberinfrastructure on input DQ, ML application performance, and ML application robustness. We focus on various ML domains, including image and sound classification, voice recognition and transcription, and video object detection and classification, all challenging for real-time ML applications. Using the POWDER platform [23], we examine real-world network disruptions to analyze how changing network conditions affect ML application performance. We investigate the interrelationships between MLIN components, input DQ, and ML application performance. Additionally, we demonstrate how knowledge on this interrelationships can be applied to develop a feedback adjustment system for MLIN aimed at ensuring ML application performance and robustness. We showcase how our ML robustness calculus can be implemented in practice by developing multiple types of ML robustness indicators and studying their feasibility for a particular application scenarios. Below we formulate multiple sub-questions answered in this chapter.

- How to realize the integrated MLIN architecture in practice?
- How to examine the interrelationships between various MLIN components and their impact on the input DQ and ML application performance demonstrated over this input?
- How to incorporate MLIN feedback adjustment component into the MLIN architecture in practice?
- What flexible and feasible calculus and indicators should be developed to realize the introduced ML application robustness generic calculus in practice?

1.1.4 RQ4. How to improve ML robustness by improving effectiveness, security, and privacy over the ML training phase?

In chapter 5, we focus on investigating strategies to enhance the ML application robustness in MLIN applied within the ML training phase. We concentrate on two major approaches, Transfer Learning (TL) and Federated Learning (FL), both demonstrating a great potential in mitigating the detrimental DQ variation effects on ML application performance. Through our empirical studies, conducted in industrial ML application contexts, we assess these strategies and analyze their

feasibility in practical scenarios. TL, initially designed for adapting an ML model from one application domain to another one, showcases its effectiveness in enhancing ML application robustness under shifting DQ conditions during the ML execution. On the other hand, FL, originally intended to reduce communications and enhance data privacy in distributed ML systems, also appears to be a prominent solution for improving ML application robustness when re-trained on high-quality data. To further enhance the security and robustness of FL, we strengthen it with our Reputation and Trust techniques, which enable detecting and excluding compromised local units during the ML training procedure. Our holistic approach to ML application robustness considers the integration of techniques applied during both at the ML training and execution stages, offering a comprehensive strategy that allows mitigating challenges posed by varying DQ in real ML applications. To thoroughly address the posed RQ4, in this chapter we formulate and answer the following sub-questions.

- Are TL and FL feasible to be employed over the ML training phase in order to enhance ML application robustness towards DQ variations within ML execution?
- How to evaluate if TL and FL are feasible in enhancing ML application robustness in practice?
- What conventional FL features make it vulnerable to malicious attacks against local units?
- What methods and techniques should be developed to further enhance the security and robustness of the FL?
- What reactive and preventive methods and techniques should be integrated to address the challenges posed by the DQ variation in practical ML applications?

We address and answer these RQs and their related sub-questions in the chapters further in our work. Below, we represent the major contributions we develop in our work and elucidate how they help us to find answers to the RQs mentioned above and to achieve the research goal.

1. We **develop and represent the integrated MLIN structure**, that incorporates multiple components and interrelationships between them. We represent the decomposition for each component, describe its functionality in MLIN, and demonstrate how the components and interrelationships between them can affect the DQ on various MLIN data life-cycle stages. The findings delivered by this contribution enable us to answer **RQ1**.
2. We **develop the integrated MLIN feedback approach** that enables monitoring the changes in DQ, ML performance, and their effect on ML application robustness, and apply

adjustment actions to restructure or reconfigure the system when the specified condition is met. In particular, we represent how the MLIN adjustment feedback component is incorporated into the integrated MLIN architecture, and showcase practical examples of how it can be implemented. This contribution allows us acquiring the answers to **RQ1**.

3. We **propose and develop a novel robustness definition and calculus** that, in contrast to other state-of-the-art approaches to ML robustness, captures the relationship between the input DQ and ML performance demonstrated by the ML application over this input in MLIN systems. To facilitate the ML application robustness calculus, we first formalize the DQ and provide means by which it can be quantified. The developed ML application robustness calculus considers the robustness during the ML execution phase, and allows to perform evaluations on the unseen data. The results delivered by this contribution accommodate the answer to **RQ1**.

4. We **develop the novel approach that integrates the DQ and data source's platform security** in order to optimize multi-modal and multi-platform data sources selection. The approach leverages GA to accommodate the selection in real time. We verify this approach on a set of real mobile devices embedded with heterogeneous data sources. The results delivered by this contribution facilitate the practical evaluation of the developed MLIN feedback component, and allows us to find answers to **RQ2**.

5. We **conduct an empirical study of multiple diverse real-world use cases** to investigate the interrelationships between the MLIN cyberinfrastructure conditions, DQ, and ML performance. We employ various real-world data collections and foundation ML models to demonstrate the impact of different technical and malicious factors, which result in DQ variation, on the robustness of ML applications. To enhance the practical value of our work, we conduct our experiments on real facilities, the access to which is kindly provided to us by the POWDER platform [23]. We **develop a practical realization of MLIN adjustment feedback component** that employs the combination of DQ, ML performance, and other MLIN network infrastructure metrics to adjust the network parameters in order to assure ML application robustness. We **validate our robustness calculus on a real example** of ML sound classification system use case. We demonstrate various ways to measure and quantify ML application robustness, which are suitable for different practical scenarios and objectives. By this contribution, we support our answer to **RQ3**.

6. We **evaluate the feasibility of TL as a proactive method employed in ML training in order to enhance ML application robustness** during the execution phase. We apply TL on a real ITS use case and show its effectiveness in enhancing the ML application robustness

in case of DQ variation. We **assess the feasibility of FL as another proactive method applied at the ML training stage in order to enhance ML application security and robustness** during the ML execution phase. We apply FL in an ITS use case and show its capabilities of improving the ML application security and robustness when faced with distributed and heterogeneous data sources with the data of varied quality. To further enhance the security and robustness of ML applications in MLIN, **we develop our novel Reputation and Trust techniques and integrate them with FL**. By equipping FL with our developed techniques, we mitigate the fundamental vulnerability in the conventional FL process, which enhances FL security, privacy, and ML application robustness. We verify our approach on a real industrial application of digital payment systems, and demonstrate our approach effectiveness and feasibility in practice. Employing our findings, we answer **RQ4**.

The research results, methods, techniques and solutions, developed in this work, have been presented on multiple conferences, workshops, and other research-oriented events, and published in peer-reviewed conference proceedings and journals as full-length articles. In view of the practical significance and commercial potential of the presented results and solutions, some of them have been transformed into inventions and two patent applications have been filed for the United States Patent and Trademark Office examination.

1.2 Products Developed in This Work

Patents:

1. **Chuprov, S.**, & Reznik, L. "Federated Learning with a Compromised Unit Exclusion from Receiving Global Model Updates". *Provisional patent application filed on January 13, 2023. Converted to non-provisional.*

2. **Chuprov, S.**, & Reznik, L. "Network Adjustment based on Machine Learning End System Performance Monitoring Feedback". *Provisional patent application filed on September 14, 2022. Converted to non-provisional.*

Journal publications:

3. **Chuprov, S.**, Belyaev, P., Gataullin, R., Reznik, L., Neverov, E., & Viksnin, I. (2023). "Robust Autonomous Vehicle Computer-Vision-Based Localization in Challenging Environmental

Conditions" in Applied Sciences. 2023, no. 13(9):5735. doi: 10.3390/app13095735.

4. Berezovskaya, O., **Chuprov, S.**, Neverov, E., & Sadreev, E. (2023). "Review and Comparison of Lightweight Modifications of the AES Cipher for a Network of Low-Power Devices" in Automatic Control and Computer Sciences. 2022, no. 56, pp. 994-1006. doi: 10.3103/S0146411622080028.

5. Melnikov, T., **Chuprov, S.**, Lazarev, E., Gataullin, R., & Viksnin, I. (2022). "Improving Reputation and Trust-Based Approach with Reliability Indicators for Autonomous Vehicles Intergroup Communication" in Tomsk State University Journal of Control and Computer Science. 2022, no. 61, pp. 108-117.

6. Marinenkov, E., **Chuprov, S.**, Tursukov, N., Kim, I., & Viksnin, I. (2022). "Study on Destructive Informational Impact in Unmanned Aerial Vehicles Intergroup Communication" in Symmetry. 2022. Vol. 14, no. 8, pp. 1-18.

7. Viksnin, I., Marinenkov, E., & **Chuprov, S.** (2022). "A Game Theory approach for communication security and safety assurance in cyber-physical systems with Reputation and Trust-based mechanisms" in Scientific and Technical Journal of Information Technologies, Mechanics and Optics, vol. 22, no. 1, pp. 47–59. doi: 10.17586/2226-1494-2022-22-1-47-59.

8. **Chuprov, S.**, Viksnin, I., Kim, I., Tursukov, N., & Nedosekin, G. (2020). Empirical Study on Discrete Modeling of Urban Intersection Management System. International Journal of Embedded and Real-Time Communication Systems (IJERTCS), vol. 11(2), pp. 16-38, doi: 10.4018/IJERTCS.2020040102.

9. Khokhlov, I., Reznik, L., & **Chuprov, S.** (2020). "Framework for Integral Data Quality and Security Evaluation in Smartphones" in IEEE Systems Journal, vol. 15, no. 2, pp. 2058-2065, doi: 10.1109/JSYST.2020.2985343.

10. **Chuprov, S.**, Viksnin, I., Kim, I., Marinenkov, E., Usova, M., Lazarev, E., Melnikov, T., & Zakoldaev, D. (2019). Reputation and Trust Approach for Security and Safety Assurance in Intersection Management System. Energies, 12(23), 4527, doi: 10.3390/en12234527.

Peer-reviewed conference proceedings:

11. Neverov, E., Viksnin, I., & **Chuprov, S.** (2023) "The Research of AutoML Methods in the Task of Wave Data Classification" in 2023 XXVI International Conference on Soft Computing and Measurements (SCM), 2023, pp. 156-158, doi: 10.1109/SCM58628.2023.10159058.

12. Tursukov N., Viksnin I., Neverov E., Sheinman E., & **Chuprov S.** (2023) "Evaluation of the Effectiveness of Neural Networks Based on the Criteria for Completing the Object Classification Task" in 2023 XXVI International Conference on Soft Computing and Measurements (SCM), 2023, pp. 120-122, doi: 10.1109/SCM58628.2023.10159070.

13. **Chuprov, S.**, Bhatt, K.M., & Reznik, L. (2023). "Federated Learning for Robust Computer Vision in Intelligent Transportation Systems" in 2023 IEEE Conference on Artificial Intelligence (CAI), Santa Clara, CA, USA, 2023, pp. 26-27, doi: 10.1109/CAI54212.2023.00019.

14. **Chuprov, S.**, Memon, M., & Reznik, L. (2023). "Federated Learning with Trust Evaluation for Industrial Applications" in 2023 IEEE Conference on Artificial Intelligence (CAI), Santa Clara, CA, USA, 2023, pp. 347-348, doi: 10.1109/CAI54212.2023.00153.

15. **Chuprov, S.**, Reznik, L., & Grigoryan, G. (2022). "Study on Network Importance for ML End Application Robustness" in 2023 IEEE International Conference on Communications. *Accepted for presentation and publication. Will be published after June 2023.*

16. **Chuprov, S.**, Satam, A. N., & Reznik, L. (2022). "Are ML Image Classifiers Robust to Medical Image Quality Degradation?" in 2022 IEEE Western New York Image and Signal Processing Workshop (WNYISPW), 2022, pp. 1-4, doi: 10.1109/WNYISPW57858.2022.9983488.

17. **Chuprov, S.**, Khokhlov, I., Reznik, L., & Manghi, K. (2022). "Multi-Modal Sensor Selection with Genetic Algorithms" in 2022 IEEE Sensors, 2022, pp. 1-4, doi: 10.1109/SENSORS52175.2022.9967296.

18. Khokhlov, I., **Chuprov, S.**, & Reznik, L. (2022). "Integrating Security with Accuracy Evaluation in Sensors Fusion" in 2022 IEEE Sensors, 2022, pp. 1-4, doi: 10.1109/SENSORS52175.2022.9967235.

19. **Chuprov, S.**, Khokhlov, I., Reznik, L., & Shetty, S. (2022). "Influence of Transfer Learning on Machine Learning Systems Robustness to Data Quality Degradation" in 2022 International Joint Conference on Neural Networks (IJCNN 2022), 2022, pp. 1-8, doi: 10.1109/IJCNN55064.2022.9892247.

20. **Chuprov, S.**, Reznik, L., Obeid, A., & Shetty, S. (2022). "How Degrading Network Conditions Influence Machine Learning End Systems Performance?" in IEEE INFOCOM 2022 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2022, pp. 1-6, doi: 10.1109/INFOCOMWKSHPS54753.2022.9798388.

21. **Chuprov, S.**, Gataullin, R., Neverov, E., Belyaev, P., Kim, I., & Viksnin, I. (2022). "Police Office Model Performance and Security Evaluation in a Simulated Group of Mobile Robots" in The 5th International Conference on Future Networks & Distributed Systems (ICFNDS 2021). Association for Computing Machinery, New York, NY, USA, pp. 606–615, doi: 10.1145/3508072.3508195.

22. **Chuprov, S.**, Viksnin, I., Kim, I., Melnikov, T., Reznik, L., & Khokhlov, I. (2021). "Improving Knowledge Based Detection of Soft Attacks Against Autonomous Vehicles with Reputation, Trust and Data Quality Service Models" in 2021 IEEE International Conference on Smart Data Services (SMDS), pp. 115-120.

23. Domnitsky, E., Mikhailov, V., Zoloedov, E., Alyukov, D., **Chuprov, S.**, Marinenkov, E., & Viksnin, I. (2021). "Software Module for Unmanned Autonomous Vehicle's On-board Camera Faults Detection and Correction" in CEUR Workshop Proceedings, Vol. 2893, pp. 1-10.

24. Lyakhovenko, Y., Viksnin, I., & **Chuprov, S.** (2021). "Integrating Smart Contracts into Smart Factory Elements' Informational Interaction Model" in CEUR Workshop Proceedings, Vol. 2893, pp. 1-6.

25. Usova, M., Viksnin, I., & **Chuprov, S.** (2021). "Informational Messages and Space Models Application in Smart Factory Concept" in CEUR Workshop Proceedings, Vol. 2893, pp. 1-8.

    Khanh, T.D., Komarov, I., Don, L.D., Iureva, R., & **Chuprov, S.** (2020). "TRA: Effective Authentication Mechanism For Swarms Of Unmanned Aerial Vehicles" in 2020 IEEE Symposium Series on Computational Intelligence, pp. 1852-1858, doi: 10.1109/SSCI47803.2020.9308140.

26. Melnikov, T., Lazarev, E., Berezovskaya, O., **Chuprov, S.**, & Viksnin, I. (2020). "Empirical Study on Premises Monitoring Algorithm Implementation in Mobile Robotic System" in The International Conference "Nonlinearity, Information and Robotics", pp. 1-6, doi: 10.1109/NIR50484.2020.9290188.

27. Usova, M., **Chuprov, S.**, & Viksnin, I. (2020). "Informational Space and Messages Interaction Models for Smart Factory Concept" in 2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT, pp. 617-621, doi: 10.1109/MetroInd4.0IoT48571.2020.9138292.

28. **Chuprov, S.**, Marinenkov, E., Viksnin, I., Reznik, L., & Khokhlov, I (2020). "Image Processing in Autonomous Vehicle Model Positioning and Movement Control" in IEEE 6th World Forum on Internet of Things (WF-IoT) Proceedings, pp. 1-6, doi: 10.1109/WF-IoT48130.2020.9221258.

29. **Chuprov, S.**, Viksnin, I., Kim, I., Reznik, L., & Khokhlov, I. (2020). "Reputation and Trust Models with Data Quality Metrics for Improving Autonomous Vehicles Traffic Security and Safety" in Proc. IEEE/NDIA/INCOSE Syst. Secur. Symp, pp. 1-8, doi: 10.1109/SSS47320.2020.9174269.

30. **Chuprov, S.**, Viksnin, I., & Kim, I. (2019) "Urban Intersection Management with Connected Infrastructure Objects and Autonomous Vehicles" in 2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE), pp. 1-4, doi: 10.1109/ICCVE45908.2019.8964917.

31. **Chuprov, S.**, Viksnin, I., Kim, I., & Nedosekin, G. (2019). "Optimization of Autonomous Vehicles Movement in Urban Intersection Management System" in 2019 24th Conference of Open Innovations Association (FRUCT), pp. 60-66, doi: 10.23919/FRUCT.2019.8711967.

32. **Chuprov, S.**, Viksnin, I., Kim, I., & Usova, M. (2019). "Intersection Management Tasks in Mobile Robotic System with Decentralized Control" in CEUR Workshop Proceedings, vol. 2344, pp. 1-12.

33. Usova, M., **Chuprov, S.**, Viksnin, I., Gataullin, R., Komarova, A., & Iuganson, A. (2019). "Model of Smart Manufacturing System" in International Symposium on Intelligent and Distributed Computing, pp. 356-362, doi: 10.1007/978-3-030-32258-8_42.

34. Marinenkov, E., **Chuprov, S.**, Viksnin, I., & Kim, I. (2019). "Empirical Study on Trust, Reputation, and Game Theory Approach to Secure Communication in a Group of Unmanned Vehicles" in CEUR Workshop Proceedings, vol. 2590, pp. 1-12.

35. Usova, M., **Chuprov, S.**, Viksnin, I., & Baranova, O. (2019). "Model of Secure Informational Messages for Ensuring Informational Interaction in Smart Factory" in CEUR Workshop Proceedings, vol. 2590, pp. 1-8.

36. Viksnin, I., Lyakhovenko, J., Tursukov, N., **Chuprov, S.**, & Sozinova, E. (2019). "Empirical Study on Modeling of People Behavior in Emergency" in CEUR Workshop Proceedings, vol. 2590, pp. 1-8.

37. Kim, I., Matos-Carvalho, J. P., Viksnin, I., Campos, L. M., Fonseca, J. M., Mora, A., & **Chuprov, S.** (2019). "Use of Particle Swarm Optimization in Terrain Classification based on UAV Downwash" in 2019 IEEE Congress on Evolutionary Computation (CEC), pp. 604-610, doi: 10.1109/CEC.2019.8790031.

38. Viksnin, I., **Chuprov, S.**, Usova, M., & Zakoldaev, D. (2019). "Police Office Model for Multi-agent Robotic Systems" in IOP Conference Series: Materials Science and Engineering, vol. 497, no. 1, p. 012036, doi: 10.1088/1757-899X/497/1/012036.

Other results (publications in Russian):

39. Belov, A., Belyaev, P., Viksnin, I., Kim, I., Radabolsky, V., Turushev, T., & **Chuprov, S.** (2023). "Software and Hardware Bundle for Controlling a Group of Unmanned Vehicles Based on Robust Computer Vision Algorithms" in LETI Transactions on Electrical Engineering & Computer Science. 2023, vol. 16, no. 6. pp. 52–69. – *in Russian.*

40. Buzina, E., **Chuprov, S.**, Tursukov, N., Belyaev, P., & Viksnin, I. (2022). "Algorithm for Uniform Coverage of the Monitoring Territory by Unmanned Aerial Vehicles" in LETI Transactions on Electrical Engineering & Computer Science. 2022. Vol. 15, no. 5/6. P. 41–50. doi: 10.32603/2071-8985-2022-15-5/6-41-50. – *in Russian.*

41. Berezovskaya, O., **Chuprov, S.**, Neverov, E., & Sadreev., E. (2022). "Review and Comparison of AES Lightweight Modifications for a Low-Power Devices Network" in Problems of Information Security. Computer Systems. no. 2, pp. 35-50. doi: 10.48612/jisp/gp9v-96dh-32 v1. – *in Russian.*

42. **Chuprov, S.**, Gataullin, R., & Viksnin, I. (2021). "Vulnerabilities Assesment in Mobile Robotic Control System" in Proceedings of the St. Petersburg State Electrotechnical University LETI, vol. 10, pp. 70-75. – *in Russian*

Chapter 2

System Integration Methods to Evaluate and Ensure Machine Learning Application Robustness in Machine Learning Integrated with Network Systems

The integration of data sources, network facilities, and cloud-based ML applications has become increasingly prevalent in today's real-world applications. However, existing approaches addressing the challenge of ML application robustness in the face of DQ variation tend to consider each component separately, discounting the crucial interrelationships between them. In our study, we investigate the significance of these interrelationships and propose a novel approach to ensure ML application robustness by considering the integrated structure of MLIN systems.

In this chapter, we present a comprehensive exploration of robustness in ML systems, followed by a review of existing approaches to improve ML robustness. We then introduce our novel approach, which leverages the interconnected components of MLIN systems to address the challenge of ML application robustness to DQ variations. We structure this chapter in the following way.

- In sec. 2.1 and 2.2, we present an overview of the existing approaches employed to define and evaluate robustness in ML systems. These section highlights the limitations of conven-

tional approaches that commonly do not pay enough attention to the integrated nature of MLIN systems and the interrelationships between data sources, network facilities, and ML applications.

- In sec. 2.3, we introduce our novel approach designed to ensure ML application robustness to DQ variations by considering the integrated structure of MLIN systems. We outline how our approach utilizes the interrelationships between MLIN components and aligns them with the ML application performance requirements. We incorporate our novel feedback mechanism into the MLIN that allows to adapt the MLIN structure to effectively meet these performance requirements in order to ensure ML application robustness.

- In sec. 2.4, we delineate how the concept of DQ is employed within the context of MLIN systems. We formally define the DQ metric and present our generic calculus, which enables the calculation of this metric's value. The DQ metric serves as a foundation for evaluating the quality of data, which might be affected during its life-cycle in MLIN.

- In sec. 2.5, we introduce the concepts of Reputation and Trust, which we leverage to select more reliable data sources and enhance the security of the MLIN system by detecting malicious and failed sources. We provide formal definitions for Reputation and Trust indicators and outline the calculus employed to determine the indicators.

- In sec. 2.6, we define the notion of ML application robustness as utilized in our work. We illustrate how our robustness indicator combines the previously introduced DQ metric with ML application performance results. As our major contribution, we present our calculus for ML application robustness and discuss the practical advantages of its application in practice.

2.1 Definitions and Approaches to ML Robustness

The concept of ML robustness has been used in different ways in the publications, depending on the context, ML application, and the perspective the authors pursue while working on the problem. However, the general "averaged" definition of ML robustness, derived from the literature reviewed in this work, is *the ability of model to maintain performance under various conditions, such as adversarial attacks, data perturbations, and shifts in data distribution.* The concept of ML robustness has evolved over time, as new challenges and applications have emerged in the field. At present, the growing body of literature recognizes that ML models should not only demonstrate high performance but also exhibit robust and reliable behavior in diverse and challenging scenarios

[78, 193]. This evolution reflects the increasing complexity of real-world ML applications, the continuing deep integration of ML systems into various routine activities, and the demand for models that can operate effectively and reliably in dynamic and uncertain environments. In order to provide the context for our work and support the ongoing discussion, we describe how ML robustness is defined, interpreted, formalized, and understood by other researchers in the field, and also review the approaches attempting to enhance ML robustness from various perspectives.

2.1.1 Robustness as an Ability to Explain and Interpret ML Decisions

Over the last decade, ML models had been deployed to automate decision-making process. Some examples are financial institutions, where ML models help to determine credit eligibility, or a hiring process, where ML is employed to process job applications. In some cases, for example job or credit applications processing, the particular reason, on which a ML decision is based, is important. By its nature, ML systems are not designed to explicitly present reasoning in their decision-making process. In other words, ML systems do not know why one or another input data-sample should receive a specific label. Some authors, who consider this problem, suggest interpretable ML models as a potential solution that is transparent on decision-making, and tend to claim a ML model as robust if its decision-making is traceable. However, the term "interpretability" itself is also ill-defined in the literature. Below we discuss some studies that look at ML robustness from the interpretability perspective.

Lipton [130] proposes to consider two components of interpretability. The first is transparency of the model's outcome, related to understanding of how the ML model works. Another consists of post-explanations and relates to getting additional information from the model. The former component can be treat as transparency or, in other words, the model decision-making process should be fully understandable by its user. The latter component relates to extracting a useful information from a trained ML model that can help to interpret the decision. For instance, a ML model classifies a traffic sign as a stop sign because the patterns in the image are similar to the images labeled as stop signs.

Lou *et al.* [133] study Generalized Additive Models (GAMs), and consider them as more interpretable by the users than complex classification and regression models. The reason is, in complex ML models understanding the contributions of a single feature to a classification result might be non-trivial. To provide user understanding on the individual feature contributions, the authors proposed to calculate each model's predictor impact. The authors provided extensive empirical study of different methods for GAMs learning, and showed that in some cases GAMs can be more

accurate than simple General Linear Models.

Ribeiro *et al.* [184] proposed method (SP-LIME) and algorithm (LIME) that help to explain the predictions of any classifier or regression by the interpretable model local approximation. The authors define interpretability as an ability to provide a qualitative relation between the input variable and the model's response, and employ it as one of the major factors to formulate decisions' explanation. The proposed method and algorithm were validated in a simulated user study, and in a study with human subjects. The studies included the interpreted decisions made by different classifiers, and assessing how these interpretations can help users to understand the classifier's prediction. The results demonstrated that both LIME and SP-LIME definitely outperformed the baseline greedy method in helping the non-expert in ML users to select better classifier by its prediction explanations.

There are a number of other studies on interpretability, which consider it as one of the major characteristic of robust ML model. Rudin [192] argues to use interpretable models for high-stakes decisions instead of black-box ones. Ross and Doshi-Velez [188] investigate the Deep Neural Networks (DNNs) Robustness and interpretability improvement by regularizing their inputs gradient. Poursabzi-Sangdeh *et al.* [174] conduct a comprehensive user-study to investigate how the model's interpretability helps to detect and correct models sizable prediction mistakes. Although an increasing number of publications on this topic [31,37,169] the iterpretability-related robustness definition lacks quantitative and unbiased evaluation, which is inapplicable for the approach we are pursuing.

2.1.2 Robustness as User's Trustworthiness towards ML

Another way to define ML robustness relates to evaluating of how trustworthy the model is to its users [226]. This manner shares the same direction as previously introduced interpretability-related definition. However, even if the way of ML model's decision-making is clearly transparent and can be explained to the user, it does not mean that the user trusts to this model or to the output it provides. A simple example is self-driving car accidents due to unusual situations for ML algorithms. Their occurrence can be treated by a user as a lack of ML system robustness and is not contributing to engendering trust toward them. The importance of maintaining users' trust towards the ML model as one of the major component of robust ML was mentioned long before wide ML systems application [57,185]. Trust is a highly social-related concept, which demonstrates the degree of how one entity can be confident that another entity will perform the expected action. In the context of users, in most cases it more relies on intuition and expectations than on formal quantitative metrics. Below we discuss some studies that implement such concept as a component

of robust ML.

The problem of ML models bias in making decisions and algorithmic intransparency does not contribute to users' trust. Training data with biased samples and prejudice in labels can shift ML decision-making results [24]. One of the recent proposals to increase users trust in ML models is AI Fairness 360 package[1], developed and maintained by IBM. The package provides a complex set of methods and documentation, which allows to test models and datasets for potential biases, and algorithms to mitigate them. AI Fairness 360 allows to process datasets for potential bias attributes (e.g., sex, age, income) and provides descriptive statistics based on which privileged and unprivileged groups are detected. After processing the dataset with bias mitigation algorithms, the package represents the comparison on how the bias changed.

Some studies include robustness as a component of ML trustworthiness. Liu [131] discusses two essential components of trustworthy ML: fairness and robustness. The former one relates to avoiding discrimination and bias in the ML decision-making process towards individuals or groups. The latter one "requires an ML system to be robust to the noisy perturbations of inputs and to be able to make secure decisions" [131]. Although the author proposed to maintain fairness and robustness to engender users' trustworthiness in ML, it is not clear how these terms are formally defined and calculated, are they represented quantitatively or qualitatively, and to which degree each of them contributes to the trustworthiness. It is important to note that even if the ML model or its decisions are trustworthy to a user, it does not mean that these decisions are fair or unbiased. Lakkaraju and Bastani [119] follow this insight and demonstrate an interesting example of how misleading explanations of black-box models can increase users' trust in these models. They conduct a systematic user study to demonstrate that malicious actors can convince users to trust the ML model that employs prohibited features in the decision-making process. The theoretical framework which formally defines users' trust towards a black box models and proposes theorems that specify misleading explanations was described in the paper. This framework utilizes features previously introduced in Model Understanding through Subspace Explanations framework [120], and expands it by supplementary limitations intended for generating high-fidelity explanations extraction that omit prohibited features and include desired ones. Their user-study results showed that the misleading explanations increased domain experts trust by 9.8 times.

There are a number of other studies that consider users trust as inevitable component of ML robustness. Hutchinson *et al.* [92] propose frameworks aimed at maintaining the process of collecting transparent and accountable datasets, which can help to increase users' trust in AI systems. Arnold

[1] http://aif360.mybluemix.net/

et al. [13] consider such components as fairness, explainability, security, and provenance as the vital ones that affect users' trust in AI services.

2.1.3 Robustness as the Ability of ML System to Perform Well on Low Quality Data

Some approaches rely on defining an ML model as robust if it is able to handle imperfect data inputs gracefully, ensuring that its outputs remain reliable and accurate. Rozsa *et al.* [190] investigate the correlation between ML image classifier's robustness and accuracy on a number of open-source well-known ML image classifiers, such as AlexNet [115], VGG16 and VGG19 [204], etc. To evaluate ML classifier's Robustness, a number of state-of-the-art adversarial attacks, such as the fast gradient sign (FGS) [75], and fast gradient value (FGV) and the hot/cold (HC) approach [191] are employed to generate adversarial testing images. To evaluate the difference between the original and adversarial images, the perceptual adversarial similarity score [191] is employed in addition to L_2 and L_∞ distance metrics. The robustness is evaluated as the ability of the ML classifier to perform well on adversarial images. The obtained empirical study results show that the ML classifiers that initially demonstrated better accuracy metric's value on the original images generally perform better on the adversarial ones. As a result, more perturbations is required to cause the performance drop for those ML classifiers. In addition, Rozsa *et al.* [190] evaluated cross-model generalization of the generated adversarial images. In other words, this evaluation shows which adversarial perturbation method results in higher performance drop over a number of tested ML classifiers. The findings demonstrate that FGS samples resulted in better generalization than FGV and HC ones, and Residual Networks [81] show higher robustness against all the types of tested adversarial examples.

The concept of open-world ML is a domain where ML end systems is considered to work in real-world applications and process data generated in real conditions. To provide appropriate performance, the inputs should possess similar characteristics as the train data distribution. Inputs that are not satisfy this criteria are known as out-of-distribution data (OOD). Despite there are an extensive literature on OOD [19, 82, 124], the potential of specifically generated adversarial OOD samples is under-researched [199]. OOD is the data that derived from the distribution the characteristics different from the data distribution the ML model was trained on. To maintain proper open-world ML classifier's operation, it is important to detect and discard OOD inputs. OOD detectors are usually implemented in open-world ML to distinguish the input data, which is not from the same distribution as the training data.

Sehwag *et al.* [199] define ML classifier's Robustness against adversarial OOD as an ability to correctly identify OOD inputs, even if they were generated in an adversarial manner. In their experiments, Sehwag *et al.* employ several well-known image datasets and use them to generate adversarial OOD inputs. For example, if the ML classifier is trained on CIFAR-10 [114] dataset, ImageNet [59] is used to generate adversarial OOD inputs. As ML classifier's models, several state-of-the-art architectures such as CNNs with various layer numbers (e.g., [30,137,241]), DenseNet [90], MobileNet [86] are utilized. The ML classifiers are employed alongside with the OOD detector. They conducted an empirical study and evaluated ML classifier's robustness against adversarial OOD on two use cases: with generated adversarial OOD inputs and in real-world attack conditions. As defense mechanisms in the first case, they tested OOD detectors and robust training approaches.

In the first case, the obtained results demonstrated that existing state-of-the-art OOD such as ODIN [126] and Confidence-calibrated classifier [124] are not robust to adversarial OOD inputs. Examined adversarial training approaches such as Convex polytope relaxation [241] and iterative adversarial training [137] indeed may be somewhat robust to in-distribution adversarial inputs, but lack robustness to OOD ones. As results evaluation metrics, classification accuracy, false negative rate, target success rate, and mean classification confidence are employed. For ML classifier's robustness evaluation, the authors used the following definitions: not robust, somewhat robust, and provably robust. However, these definitions are not strictly determined mathematically in the paper, and the reader might only infer potential thresholds from the provided results.

In the second case, Sehwag *et al.* use generated adversarial OOD samples to perform DoS attack on the Clarifai's content moderation system[2]. The results of their attacks demonstrate that they are able to fool the content's classifier with a high attack success rate, which creates intolerable delays for content reviewing moderators. The authors also provide several experimentally justified defensive mechanisms to improve ML classifier's robustness against adversarial OOD examples. They include: adding a small subset of OOD to the train data; using a single OOD dataset for training to achieve generalization on other OOD datasets; and combining multiple OOD datasets in a single training set. Another finding is that these robustness improvement methods deteriorate ML classifier's performance over in-distribution data only marginally, 1% for a single OOD dataset, and around 3.1% for the multiple ones.

Gu and Rigazio [76] propose to employ autoencoder (AE) to pre-process and denoise the ML model's input data. They found that AE trained to denoise one type of adversarial samples can generalize to denoise the other types. However, the classical AE's architecture allows the attacker to

[2] `https://www.clarifai.com/`

construct and employ new adversarial sample with smaller perturbation, which the already trained AE would not be able to detect. The authors leverage the ideas of denoising AE (DAE), contractive AE (CAE), and marginalalized DAE (mDAE) that can be used to train ML model robust to adversarial examples similar to ones proposed in [9, 34, 186]. Based on these ideas, they develop Deep Contractive Networks (DCNs) that allow to construct training samples with substantially higher perturbations for ML models robust training. The proposed method incorporates a layer-wise penalty mechanism, which bridges supervised and unsupervised learning and allows DCNs to minimize model's outputs' variance with respect to input perturbations.

Various ML systems testing approaches include ML systems performance evaluation on randomly picked or adversarial samples [75,157]. However, such testing approach usually lacks efficiency and demand an operator to analyze testing results and fine-tune the input samples. Pei *et al.* [167] propose DeepXplore intelligent technique designated for Deep Learning (DL) systems automated whitebox testing. They introduce the metric of neuron coverage, which allows to evaluate the efficiency of the performed ML system's testing. The metric evaluates the number of activated neurons by the provided testing samples. The solution also allows to determine incorrect corner case behaviors without manual operator's intervention. According to experimental results, DeepXplore is able to explore the testing DL system architecture and to carefully construct testing samples to maximize the neuron coverage and finding erroneous behaviors of the DL model. As a potential solutions to improve DL system accuracy, the authors propose to use DeepXplore to augment the dataset by diluting it with the generated samples. In addition, DeepXplore may be used to detect data pollution attacks, when one or more of the training samples are maliciously mislabeled. The developed solution has rich functionality and facilitate ML systems' researchers and developers with a valuable testing instrument. However, DeepXplore allows to work only on open-source DL systems, when the full system architecture is accessible to the tester.

Tian *et al.* [216] follow the neuron coverage idea proposed by Pei *et al.* [167], and develop a DeepTest automated testing framework for self-driving vehicles decision-making. In contrast to previous research [167], they expand the neuron coverage technique to Recurrent Neural Networks (RNNs). The main aim of the DeepTest framework is to generate such perturbed examples that induce autonomous driving ML models to demonstrate erroneous behaviors. Erroneous behavior can be characterized as ML model's decision that can potentially lead to fatalities (e. g., crash with another vehicle or pedestrian). The developed tool incorporates various image data augmentation and modification techniques (e. g, image translation, contrast alteration, adding weather effects, etc.), and is able to generate synthetic data for further ML model's training and testing. The framework

is evaluated over the Udacity self-driving challenge dataset[3] with various self-driving ML models: Rambo[4], Chauffeur[5], and Epoch[6]. The experimental results show that testing samples produced with DeepTest allows to substantially increase neuron activation ratio. The positive correlation between the neuron coverage and the diversity of input-output space justifies the idea of employing neuron coverage metric for ML models' robust testing. Experiments on erroneous behaviors revealed the ability of DeepTest to automatically generate the example, submit it to ML model for decision-making, and detect potentially unsafe decisions. However, some image augmentation techniques (e. g., rotation) may induce DeepTest to improperly mark safe ML decisions as unsafe, which can be classified as false-positive. Also, re-training of ML models with the samples generated by DeepTest allows to increase their accuracy over the diverse dataset of original and modified images almost by a half in the best case scenario.

Zhang *et al.* follow the ideas of self-driving ML model's testing frameworks proposed in [167] and [216], and develop DeepRoad. DeepRoad addresses the issues of the previous two frameworks and allows to synthesize testing inputs appearance of which is closer to the real-world ones. Besides of generating driving scenes with various weather conditions, the authors introduce the ability of adding heavy snowfall and hard rain effects to the synthesized testing inputs. The framework incorporates two major modules, designated for metamorphic testing and input validation. The former one is able to generate testing inputs, and to evaluate ML model's decisions based on these inputs in order to detect erroneous behaviours. The latter one is designated to evaluate if the provided input corresponds to the ML model training set's distributions. This validation allows to distinguish adversarial, modified, or OOD samples, which can deteriorate ML model's performance and lead to potentially unsafe decisions. Each of the framework's module is experimentally evaluated over Udacity self-driving challenge dataset[7]. To test the metamorphic testing module such ML self-driving models, as: Autumn[8], Chauffeur[9], and Rwightman[10] are employed. As the major metric, the number of detected ML model's improper decisions is evaluated. According to the experimental results, the metamorphic testing module is able to autonomously generate realistic testing examples, submit them to the ML model, and report if the ML model's decision over the

[3] https://github.com/udacity/self-driving-car/tree/master/challenges/challenge-2

[4] https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/rambo

[5] https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/chauffeur

[6] https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/cg23

[7] https://github.com/udacity/self-driving-car

[8] https://github.com/udacity/self-driving-car/tree/master/steering-models/evaluation

[9] https://github.com/udacity/self-driving-car/tree/master/steering-models/community-models/chauffeur

[10] https://github.com/udacity/self-driving-car/tree/master/steering-models/evaluation

augmented testing sample is inconsistent with the decision over the original sample. In the input validation experiments, the module is expected to detect if the sunny, snowy, and rainy images are not belong to the ML model's training distribution. The results revealed that the input validation module can successfully detect 100% of rainy, and 85% snowy images. However, the module is able to detect only 21% of sunny images, that can be justified the similarity of the sunny images to the original ones in terms of image patterns. The employment of the developed framework indeed may help to improve the robustness of ML model for self-driving, and may be generalized to other domains which exploit DL models.

Ma *et al.* [136] follow the approach of ML model's robustness testing from the software engineering perspective [167], and propose DeepGauge DNN testing system. Their idea incorporates and substantially extends neuron coverage concept, employed by Pei *et al.* [167] in their DeepXplore framework. In addition to simple neurons' coverage evaluation, Ma *et al.* introduce several sophisticated metrics, based on the lower and upper bounds of the values produced by a neuron. Also, they introduced layer-level metrics, which allow to evaluate the behavior of the neurons, that have the most influence on ML model's behavior on each layer. The proposed system is evaluated over MNIST [60] dataset with three LeNet family models (LeNet-1, LeNet-4, and LeNet-5) [122], and over ImageNet [59] with VGG-19 [204] and ResNet-50 [81] models. To produce adversarial testing examples, several state-of-the-art image adversarial attacks are implemented over the employed datasets: Fast Gradient Sign Method (FGSM) [75], Basic Iterative Method (BIM) [116], Jacobian Saliency Map Approach (JSMA) [165], and attacks proposed by Carlini and Wagner (CW) [30]. Empirical study results demonstrate that adversarial examples increase all the developed neurons coverage metrics in comparison to the original dataset examples for both employed data collections and all employed adversarial attacks. The extended neurons' coverage evaluation is indeed a helpful technique, which can be beneficial in investigating various unexpected outputs and behavior that ML models may demonstrate in extreme situations.

Huang *et al.* [91] consider the problem of robust ML image classification from the self-driving safety perspective in critical situations. They investigate how various images modifications (e. g, changes in size, lightning, and etc.) and adversarial perturbations affect the ML model's classification performance. They focus on evaluating ML models robustness against a single adversarial sample, which can be referred as measuring of a local adversarial robustness [97, 255]. Huang *et al.* develop a framework that is capable of automatic safety verification of self-driving operation decisions based on the correctness of image classification. The framework allows to verify feed-forward deep-neural networks and to guarantee that if there is a misclassification or falsification exists, it will be revealed. The verification analysis is performed layer by layer and is based on a searching of

manipulations in images that result in ML model's misclassifications. The developed framework is implemented as Deep Learning Verification tool[11] and tested over such state-of-the-art datasets as MNIST [60], CIFAR-10 [114], ImageNet [59], and the German Traffic Sign Recognition Benchmark (GTSRB) [205]. The experimental results demonstrated that the proposed tool can efficiently detect incorrect decisions made by ML models. However, the time of the detection depends on the input data dimensionality and grows exponentially with the number of features.

Papernot *et al.* [166] informally define DNN robustness against adversarial perturbations as its ability to resist this perturbations. Inspired by the idea proposed by Hinton *et al.* [84], they proposed a defense distillation technique against adversarial perturbations, which is based on decreasing of the ML model's sensitivity toward these perturbations. The problem of constructing appropriate adversarial sample can be formalized as the optimization problem of finding minimal perturbation that leads to ML model's misclassification. The developed approach is experimentally evaluated on MNIST [60] and CIFAR-10 [114] datasets, and two DNNs with 9 layer architecture. The obtained results demonstrated that the robust distillation can substantially decrease the adversarial attack success rate from $\approx$ 96% to 0.5% for MNIST, and from $\approx$ 88% to 5% for CIFAR-10. The significant decrease in ML model's sensitivity toward adversarial samples resulted in 1.37% drop in ML performance for both datasets. In addition, Papernot *et al.* [166] calculate ML model's robustness as the the average minimal perturbation required to produce an adversarial sample that results in improper decision. The implementation of defensive distillation technique allows to increase robustness by 790% in case of MNIST, and by 556% in case of CIFAR-10.

2.2 ML Robustness and Performance Improvement Approaches over Varied Input DQ

2.2.1 Data-Oriented Approaches

Xu *et al.* [248] propose feature squeezing technique that allows detecting adversarial examples from the ML model's input data. The technique is "non-invasive", which means that it does not alter the ML model's architecture and operate as a separate component. The technique is designated to work with image data, however, according to Xu *et al.*, it can generalize to other data types. The approach incorporates such input pre-processing methods as color dimensionality reduction, spatial smoothing, and others (e. g., the ideas of which proposed in [116,219]). The approach is investigated

[11] `https://github.com/VeriDeep/DLV`

on: MNIST [60] dataset with an ML model architecture provided on GitHub[12]; CIFAR-10 [114] with DenseNet [90]; and ImageNet [59] with MobileNet [86]. In overall, the approach is tested against 11 types of targeted and untargeted adversarial attacks. Targeted attacks aim at forcing the ML model to label the input sample to a targeted adversarial category. Untargeted ones aim to fool the ML model without any targeted category. The investigated untargeted adversarial attacks include FGSM [75], BIM [116], and DeepFool [150]. The examined targeted attacks are JSMA [165], and several variations of CW attacks [30]. The results demonstrate that inputs pre-processing with the feature squeezing technique allows to substantially improve ML model's performance for almost all common adversarial attacks.

In addition, Xu *et al.* [248] study the case of using the proposed feature squeezing technique as an adversarial image detector. The results reveal decent detection rate of $\approx 98\%$ on MNIST [60] dataset against all common adversarial attacks. However, on colored datasets, for some attacks such as FGSM and BIM, the feature squeezing technique does not work well. This can be related to larger perturbations that these attacks cause. They also considered various defensive techniques against adversarial adaptation to the employed squeezing methods. These methods may be based on the randomization of the feature squeezing process, which tangles the attacker's analysis of the ML model's outputs.

Zhang *et al.* [258] proposed the model to verify the object on multiple images taken from various illumination intensity or angle. The model is based on the light cone distribution physical properties, and is aimed to approximate the object on the images based on the distance between the perturbations caused by the lighting conditions. The proposed model allows to guarantee the robustness of non-convex objects with moving shadows boundaries verification under variable lighting. The experimental results reveal the developed model's effectiveness in object verification in 2D images and 3D models. However, for some distance and lightning conditions, substantial number of object's images is required for successful verification.

2.2.2 Network-Oriented Approaches

Network adjustment techniques can also be employed for improving the quality of data transmitted over the network in MLINs. By implementing mechanisms oriented at enhancing network Quality of Service (QoS), such as traffic prioritization [21] and bandwidth allocation [7], network adjustment ensures that ML applications related data is prioritized by the network services. In addition,

[12] `https://github.com/carlini/nnrobustattacks/`

optimizing network traffic through load balancing [74] and routing path selection [38] allows to mitigate congestion and improve data transmission efficiency. Proactive monitoring and network performance optimization commonly results in better transmission environment.

However, existing network adjustment and recommendation approaches commonly concentrate only on the network performance metrics and utilize them to apply the appropriate reactive measures to enhance the transmitted DQ and end users' Quality of Experience (QoE) [27, 88, 127, 245]. Cao *et al.* [27] focus on dynamic embedding and QoS-driven adjustment in cloud networks. The authors propose a framework that dynamically adjusts network embeddings based on desired QoS requirements. They rely on a reinforcement learning technique to optimize the allocation of virtual network functions and adaptively adjust network embeddings to enhance network performance and meet QoS requirements. The study specifically addresses resource allocation and QoS provisioning in cloud networks, which might not provide the direct effect on ML end performance. In contrast, we emphasize ML robustness in network adjustment recommendations, which allows us to aim the network adjustment actions to assuring ML robustness according to the provided specification.

Hu *et al.* [88] study path selection mechanisms for edge computing in software-defined networks (SDNs) considering both latency and packet loss metrics. A path selection algorithm that optimizes the trade-off between latency and packet loss to improve the QoS of edge computing applications is proposed in the paper. The best path for data transfers is dynamically selected using SDN-based approach that leverages network QoS metrics the path should satisfy. While the paper addresses path selection based on QoS metrics in edge computing within SDN, such an approach does not consider how the transmitted data affects the end application performance. Our proposed approach has its focus on ML robustness improvement in network adjustment recommendations.

Lin *et al.* [127] present a QoS-aware routing algorithm for SDN hybrid networks. The authors propose a routing approach that considers multiple QoS metrics, including delay, packet loss, and bandwidth, to improve the overall QoS in SDN hybrid networks. The algorithm dynamically adjusts the routing decisions based on real-time QoS metrics' evaluations. As the previous one, this paper approaches the routing path dynamic selection based on the QoS this path provides for transmitting the data. Prioritizing only the networking metrics without considering the interrelationship between them and the end application performance might not have the desired ML robustness assurance effect.

Xu *et al.* [245] address the challenge of low-latency update in SDN. The paper proposes a joint route selection and update scheduling approach that enables the trade-off optimization between updating latency and network resource utilization. The algorithm allows to dynamically select

routes and schedule updates based on the current network conditions to minimize latency and enhance the overall update efficiency. While the low-latency update in SDNs allows to enhance the data transmission performance itself, it does not provide any guarantees that the selected route positively affect the ML end performance and robustness to DQ variations.

In addition to QoS improvement, network adjustment can also focus on enhancing the end user's QoE [16]. These approaches go beyond traditional QoS metrics and consider the subjective perception of data by the end users and their requirements satisfaction. Some examples of QoE improvement approaches include dynamic adaptive streaming [239] that adjusts video quality based on the available bandwidth, latency reduction techniques for real-time applications [168], and intelligent network traffic management [155]. By considering end user's QoE, network adjustment approaches are focused on conveying a seamless and enjoyable network service, ultimately leading to improved satisfaction by the utilized networking applications and services or by the data transmitted through the network.

Laghari *et al.* [117] investigate the effect of packet loss and packet reordering on the quality of audio streaming. The paper analyzes the impact of different packet loss rates and reordering scenarios on the quality of transmitted audio. The authors provide insights into the packet loss/reordering and perceived audio quality relationship. McManus *et al.* [144] explore the effects of latency, bandwidth, and packet loss on QoE from cloud-based gaming services. The authors investigate how the network performance affects the online gaming experience, considering factors such as player actions, game state updates, and user perception of quality. Mrvelj and Matulin [153] research the impact of packet loss on the perceived quality of UDP-based multimedia streaming. On a number of use cases, the authors evaluated the users' QoE in real-life environments with varying packet loss rates. They analyzed the relationship between packet loss and quality perceived by the end user to gain insights into the impact on multimedia streaming applications.

The approaches aimed at enhancing the end user's QoE involve various network optimization methods and techniques, resource allocation, and QoS provisioning. However, in our work, we consider ML application instead of the user on the data receiving end. In this case, QoE improvement methods and techniques may not directly apply to enhancing the performance and robustness of ML end applications. Such end systems have unique requirements and challenges that differ from typical user-centric applications. ML models rely on accurate input data, whose distribution corresponds to the samples the model was trained on. While QoE-oriented methods and techniques primarily address user perception and satisfaction, ML end applications require further research to develop specific approaches considering the DQ and ML robustness interrelationship.

2.2.3 ML End System-Oriented Approaches

Another perspective on achieving higher ML robustness outcomes is through the careful manipulation of model parameters. This involves tuning hyperparameters, which govern the learning process, selecting the proper learning strategy, and adjusting weights, which influence the model's decision-making patterns. Techniques such as regularization can be employed to prevent overfitting, while methods like cross-validation help in assessing the model's generalizability. By systematically manipulating hyperparameters, one can guide the model towards a more robust state, where it not only performs well on training data but also demonstrate the required performance on unseen data.

In their comprehensive review on ML systems testing approaches, Zhang *et al.* [255] separate the definition of robustness from such testing properties as outputs' correctness, fairness, interpretability, and others. To define robustness they refer to the IEEE standard glossary [4], and specify robustness as "the resilience of an ML system's correctness in the presence of perturbations" [255]. The authors follow Katz *et al.* [97] idea to consider adversarial robustness as a sub-category, and also define local and global robustness. The former one evaluates ML system's robustness against a single adversarial sample, and the latter one against all inputs including non-adversarial ones. Local and global adversarial robustness are defined based on L_p distances, which represent the difference degree between the original sample and the adversarial one. As a p, various values may be used, e.g., 0, 2, or ∞.

Batsani *et al.* [17] emphasize the lack of objective ML robustness evaluation approach. They criticize the approach proposed in [75] to test the robustness of ML systems by comparing the performance demonstrated against adversarial inputs by the initial model and the model re-trained over adversarial samples. The major disadvantage of this approach is that ML model may simply have overfit to adversarial samples. To address this problem, Batsani *et al.* propose to employ two statistics of the robustness based on the L_∞ distance from the original sample to the nearest adversarial one. The former statistic allows to evaluate how often the adversarial sample occur. The latter one evaluates how severe this adversarial sample is. They also introduce the threshold parameter ϵ, which is based on the L_∞ distance and controls the decision boundary between the adversarial and the benign samples. This threshold ϵ is denoted as robustness of the ML model towards adversarial input. The experiments are conducted with MNIST [60] and CIFAR-10 [114] state-of-the-art datasets, and LeNet [122] and NiN neural network [129]. The approach proposed by Tabacof and Valle [207] is used as a baseline for the robustness evaluation. The results demonstrated that the approach based on two statistics allows to evaluate ML model's robustness with substantially better accuracy compared to the baseline.

Wong and Kotler [241] proposed the technique for training ML models provably robust to a norm-bounded perturbation. The approach is able to guarantee that the ML model can correctly classify an adversarial sample if its distance from the original sample is less than a specified threshold. They adapt the proposed approach to large ML models architectures (e.g., residual networks) by employing a nonlinear random projection technique, which scales linearly in the size of the hidden layers. The experimental results with the MNIST [60] and CIFAR-10 [114] datasets reveal the ability of the proposed approach to efficiently train dense and residual ML models to specified robustness level.

Shaham *et al.* [201] propose ML model's robust optimization training mechanism based on minimax procedure. They follow the idea of robust optimization given in [18], which defines it as an ability to obtain stable solutions under the data uncertainty conditions. They develop ML models training framework, the aim of which is to minimize the proposed loss function under the worst-case data perturbation conditions. The training is performed against the data with perturbations, and the ML model's hyperparameters is updated according to the worst-case examples close to the original data points. In their empirical study, Shaham *et al.* examined convolution network with ReLU units trained over MNIST [60], and VGG architecture from github[13] trained over CIFAR-10 [114]. For adversarial examples generation and robust training process, various distance metrics are examined: L_1, L_2, and L_∞. Based on these metrics, three versions of the robust trained ML models are produced for each dataset. According to the experimental results, the proposed framework allows to substantially improve ML models robustness against adversarial examples for both datasets. L_∞ demonstrates better results over other metrics with ≈80% accuracy demonstrated over the adversarial MNIST set, and ≈65% over the adversarial CIFAR-10 set. However, Shaham *et al.* do not investigate how the robust training may affect ML model's performance over the original dataset.

Madry *et al.* [137] represent the process of DL systems' adversarial robustness improvement as the optimization problem. They propose a saddle point optimization technique, which can be separated to inner maximization and outer minimization sub-problems. The first one refers to constructing such adversarial sample that maximizes the ML model's performance loss. Another one is aimed at selecting such network parameters that result in ML model's performance loss minimization caused by the constructed adversarial sample. They identify Robust ML model as the model which can achieve the specified saddle point, i.e., the saddle point is used to quantify the robustness measurement. The proposed approach is verified over MNIST [60] and CIFAR-10 [114] datasets. As DL models for robust training, simple convolution neural network for MNIST

[13] `https://github.com/torch/torch7`

and ResNet model [81] for CIFAR-10 are employed. The experimental results revealed that the classifiers trained with the proposed technique are robust against adversarial examples. However, the model trained on CIFAR-10 appeared to be less robust against adversarial examples comparing to the model trained over MNIST.

Metzen *et al.* [148] develop a binary detector, which can be embedded into ML model's architecture to distinguish adversarial inputs. The detector augments some layers of the ML model and is able to detect both static and dynamic adversary inputs. It is considered that the static adversary does not have access to the detector and can access only ML model and its gradients. The dynamic adversary presumably can access both detector and ML model, and can fine-tune the constructed adversarial samples on-the-fly. The proposed approach is tested on CIFAR-10 [114] dataset and ResNet [81] model with both static and dynamic adversaries. To examine the approach on the data of larger dimensionality, Metzen *et al.* tested it on a subset of ImageNet [59] with VGG16 [204] model. The experimental results for CIFAR-10 revealed that the detector trained to withstand static adversarial attacks demonstrates robust detectability only in the case of static adversary, and is not effective against dynamic attacks. The detector trained against dynamic adversarial attacks is substantially more robust, and can withstand in both static and dynamic adversarial scenarios with the detectability above 70%. The degree of the adversarial attack severity is determined according to the amount of perturbations the adversary may employ, and the accessibility of ML model and detector's outputs. The results also demonstrate generalizability of the detectors, which means that they are effective against similar or weaker adversaries. The study on a subset of ImageNet shows that such a detector also can be successfully employed on large datasets, however the optimizer can get stuck if the detector is not sensitive enough to sophisticated patterns.

2.3 MLIN Integrated Structure and Components Description

The integration of data sources, network facilities, and cloud-based ML applications within MLIN entails inherent interdependence of these components. The existing research has mainly focused on examining and developing these components in isolation, with limited exploration of their inter-relationships. In our work, we propose an innovative system approach that considers the mutual influence of these integrated components, and how this influence affects the quality of the data circulated in MLIN. Our MLIN design adopts a system integration perspective, aiming to enhance ML application robustness by establishing strong interrelations between MLIN components, and employing them to mitigate the effects of DQ variations on ML application robustness. We utilize the DQ metric as a medium between these components, which allows us to derive which MLIN

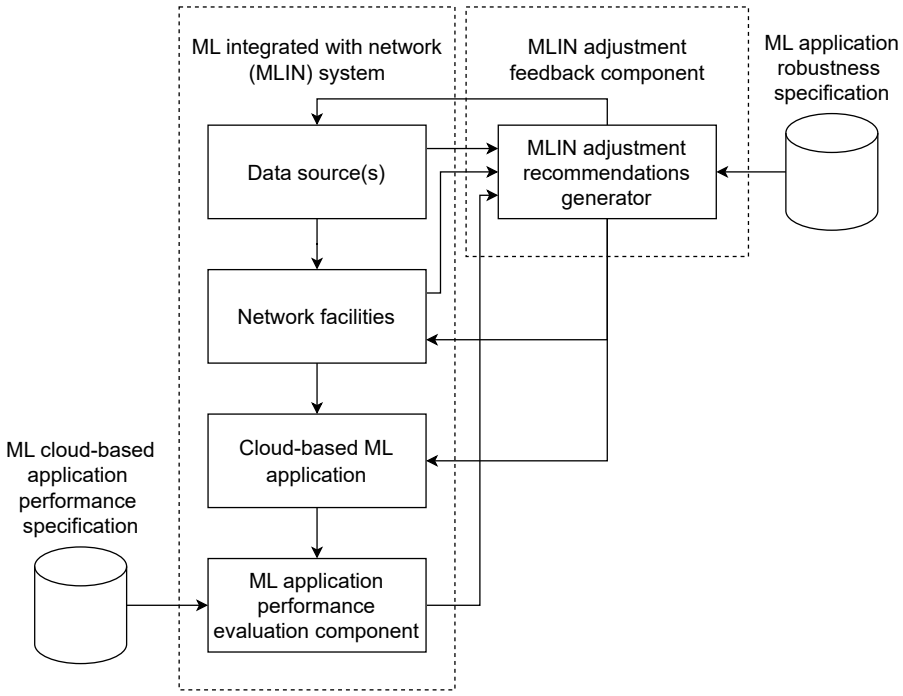


Figure 2.1: High-level representation of the overall MLIN architecture and its integration with MLIN adjustment feedback component

structure results in higher DQ and enables to ensure ML application robustness.

Our design addresses the demand of ensuring the robustness of ML applications through adjustments to MLIN components aimed at assuring the established ML application robustness requirements. These adjustments involve modifying the parameters of data sources or network facilities based on recommendations received from the feedback component specifically integrated by us into the MLIN architecture (as depicted in Figure 2.7). This feedback component generates recommendations for MLIN structural modifications. The recommendations are formulated by evaluating the ML application performance and comparing it to the established requirements. The primary MLIN components and how they are connected to the feedback component are illustrated in Figure 2.1. In the subsequent sections, we describe the MLIN architecture and its major components, and illustrate how our proposed feedback component is integrated into this architecture with the goal of assuring ML application robustness to DQ variations according to a specified requirements.

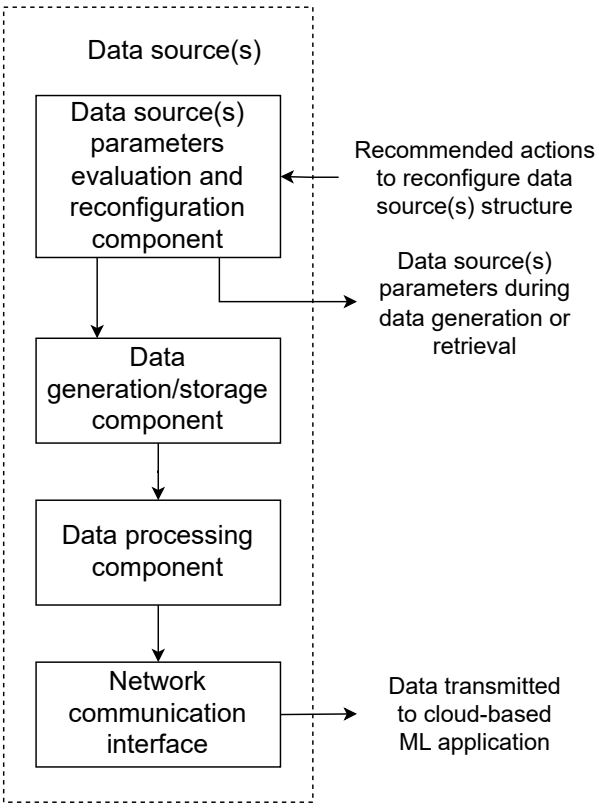


Figure 2.2: Decomposition of the data source(s) MLIN component

2.3.1 Data Source(s)

The data source(s) serves as the initial point where the data is stored or the location where physical information is initially transformed into digital format and then transmitted for further use. A practical example of the data source(s) might be a sensor device, such as a camera or accelerometer. The fundamental components of the data source(s) are illustrated in Figure 2.2. It is important to note that data source(s) can both be realized in a form of sensors for performing measurements from the surrounding environment, or they can also be presented in the form of databases from which the data is retrieved on demand. The data source(s) are equipped with a network communication interface, enabling the transmission of data over network facilities to a cloud-based ML application. They may also possess local memory for temporarily storing the collected data before transmission. The specifics of the measured or stored data, including the type, format, structure, and representation, alongside the components and architecture of the data source(s), are defined by ML application and end user requirements.

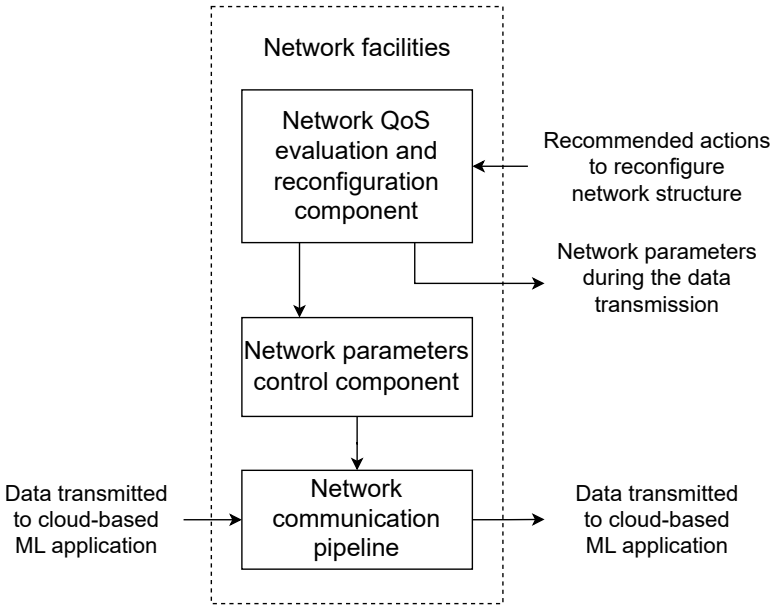


Figure 2.3: Decomposition of the network facilities MLIN component

2.3.2 Network Facilities

The network facilities component serves as a collective framework utilized by all nodes within a transmission system, encompassing network stations and the connecting transmission lines. The fundamental network facilities' units are depicted in Figure 2.3. The network facilities are responsible for conveying the data from the data source(s) to the cloud-based ML end system. In our MLIN architecture, it is essential that the network facilities enable the assessment of network QoS metrics at regular intervals during the data transmission. The concept of QoS encompasses various network characteristics that are relevant and aligned with the specified requirements delineated by the end user and ML application. Additionally, the network facilities should possess the capability of modifying their specific parameters (such as transport protocol or communication channel bandwidth) to meet the established requirements (in our case, aimed at ensuring ML application robustness). The architecture, components, particular network devices, and employed communication technologies within the network facilities are defined by MLIN specification and the user and ML application requirements.

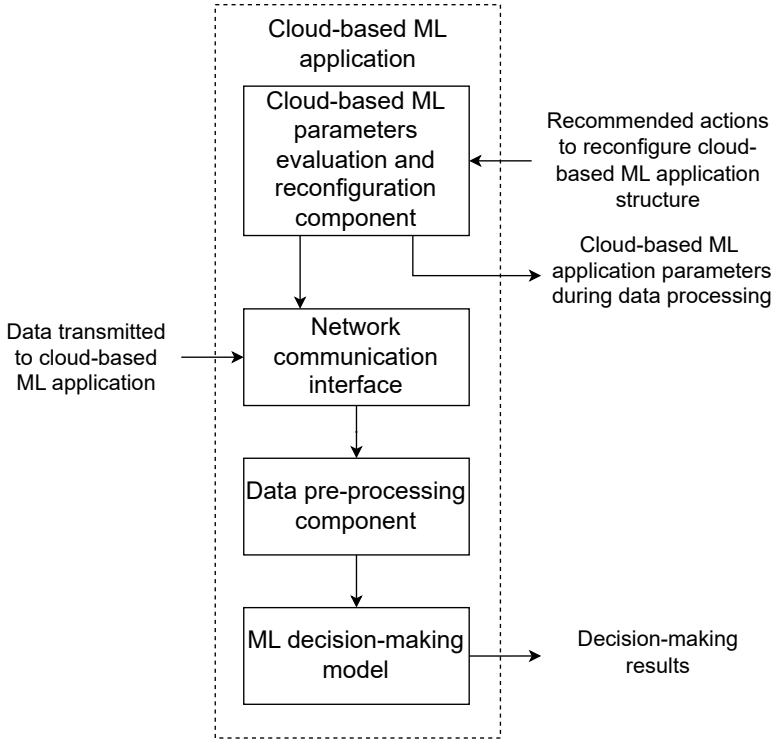


Figure 2.4: Decomposition of the cloud-based ML application MLIN component

2.3.3 Cloud-based ML Application

The cloud-based ML application is utilized to classify, detect, or recognize patterns in the data originating from the data source(s). The fundamental elements incorporated into the cloud-based ML application are depicted in Figure 2.4. The cloud-based ML application may be deployed on a remote server that is accessible over a network. The cloud-based ML application may encompass various software, hardware, and infrastructure components essential for the continuous operation of this application. The ML system itself can be based on diverse ML models, such as Recurrent or Convolutional Neural Networks, that are employed for pattern classification, detection, or recognition within the received data. The ML model employed may be pre-trained and subsequently re-trained using data of varying types, formats, and structures, as determined by the specific requirements established by the end user and ML application. In our design, we place particular emphasis on evaluating the operational performance of the ML application once all training and re-training procedures have been completed. In other words, we consider the ML application execution phase that requires processing and decision-making based on the unseen data.

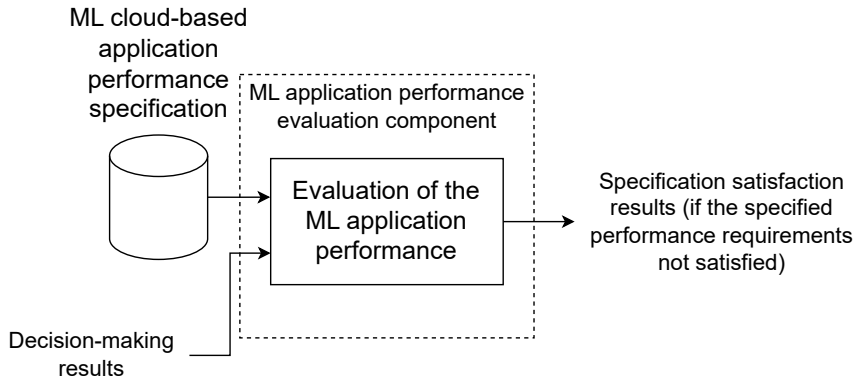


Figure 2.5: Decomposition of the ML application performance evaluation MLIN component

2.3.4 ML Application Performance Evaluation Component

The component responsible for assessing the operational performance of the cloud-based ML application is referred to as the ML application performance evaluation component. The generic structure of this component is illustrated in Figure 2.5. Performance evaluation may be conducted internally and be a part of the ML application itself, or may implemented as a distinct entity comprising software, hardware, or a combination thereof, alongside other infrastructure facilities required for its operation. Performance evaluation may utilize a designated set of metrics, such as accuracy, classification error, true positive rate, and confidence level, based on which ML application performance is evaluated. The particular metrics are specified by the requirements dictated by the end user, the employed ML model, and ML application itself. These established specifications delineate the desired performance level that the ML application must satisfy. As an example, if the confidence level metric is adopted to assess the decision-making performance, the specification may establish the required level not less than 95%. The acceptable performance levels are typically informed by industrial standards, policies, performance benchmarks set by existing industrial solutions, or performance levels recognized as satisfactory within the relevant field of expertise.

2.3.5 MLIN Adjustment Feedback Component

The MLIN adjustment feedback component produces recommendations on actions MLIN restructuring, which means adjusting the MLIN components' parameters according to a recommended ones. The component may consist of a software, a hardware, a combination thereof, and any other infrastructure required to maintain its operation. Figure 2.6 represents the basic elements of the

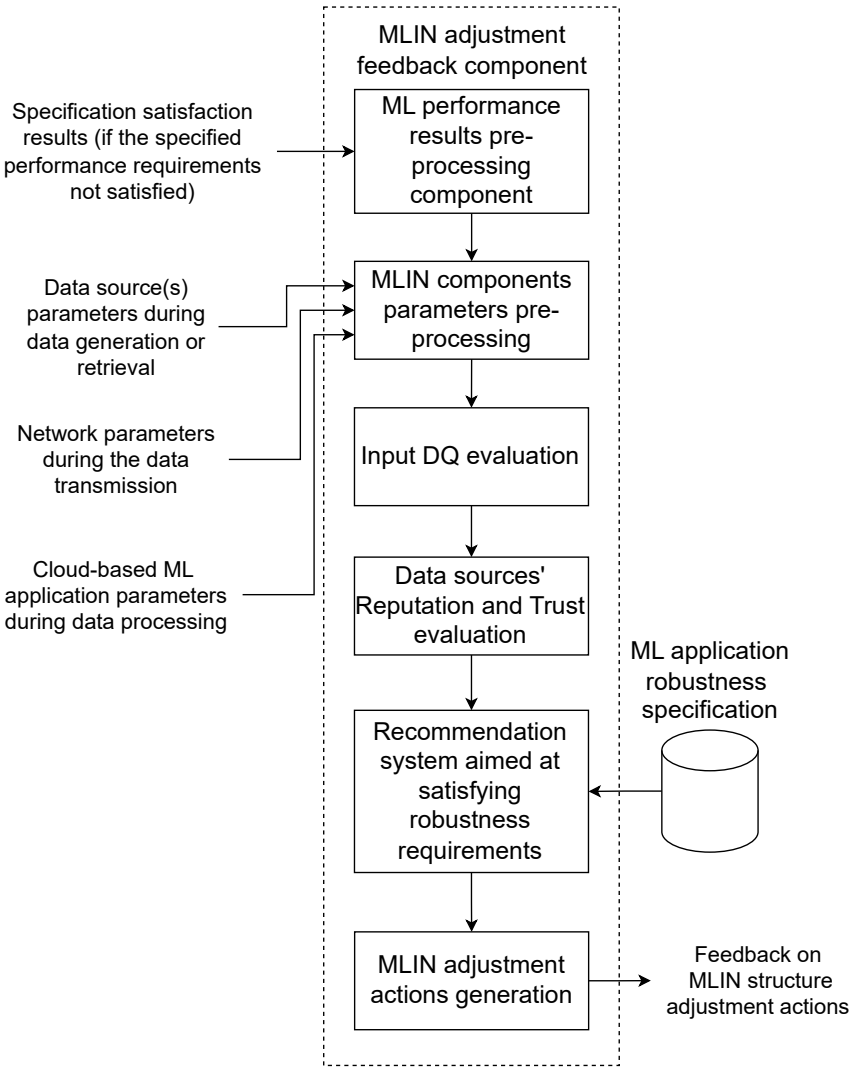


Figure 2.6: Decomposition of the MLIN adustment feedback component

component. The component possesses multiple inputs. One input is the ML system performance demonstrated over the processed data, provided by the performance evaluation component. Another input is the current MLIN components' parameters, used over the data transmission from the data source(s) to the cloud-based ML application. MLIN components' parameters may include, for example, the configuration of the data source(s) while producing the data, or network configuration and QoS demonstrated while the data transmission. These parameters are processed alongside the cloud-based ML application performance to calculate DQ (we describe the calculus for DQ in sec. 2.4), which is then used for ML application robustness evaluation.

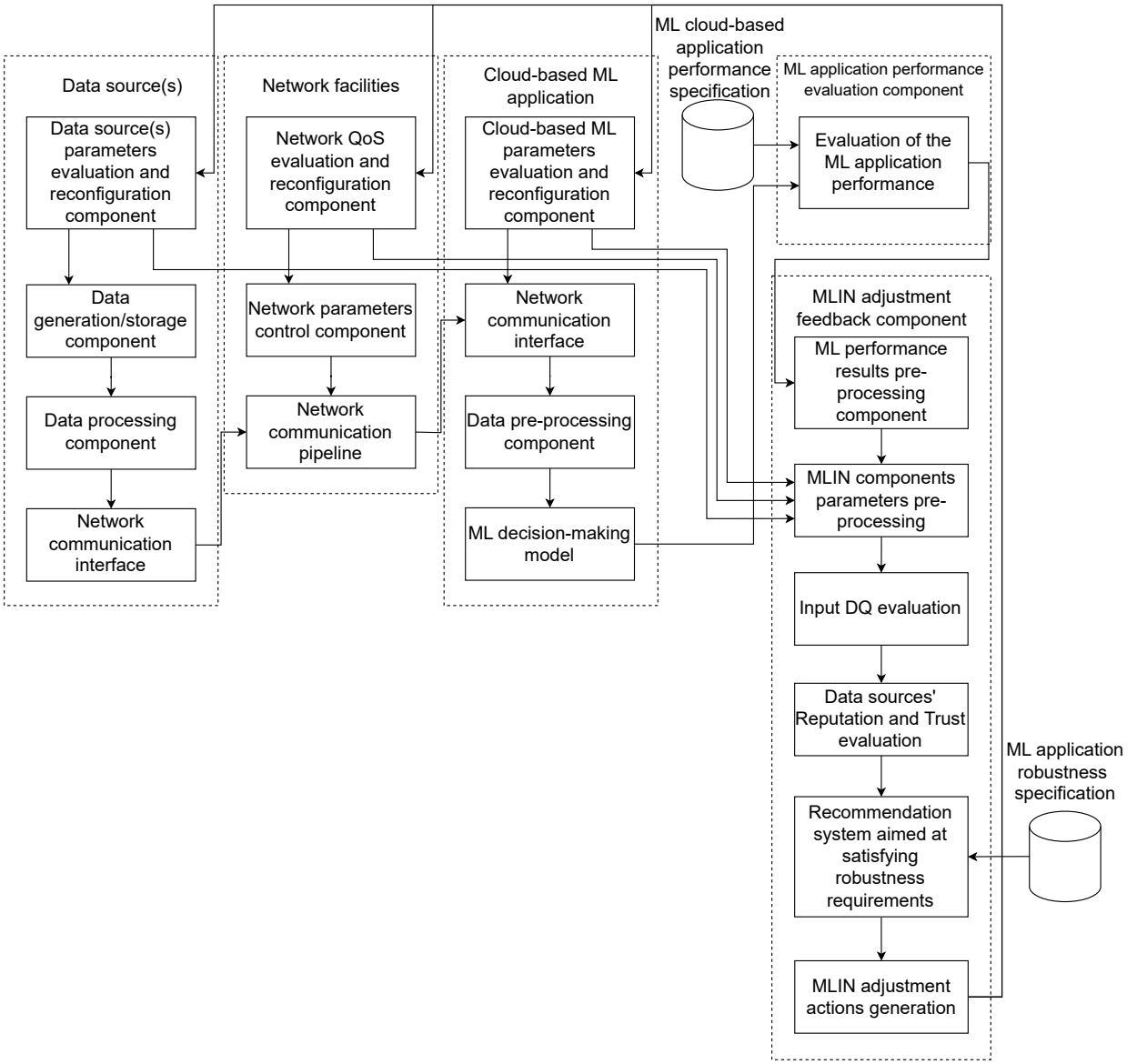


Figure 2.7: The overall high-level MLIN architecture and the interactions between its decomposed components

An output of MLIN adjustment feedback component may be produced in a form of recommendation on a certain action or a set of MLIN components' adjustment actions aimed to assure ML application robustness to DQ variations. The actions may be provided in an autonomous manner by the MLIN adjustment feedback component itself or, depending on the MLIN specification and requirements, may be delegated to another responsible MLIN component. The action or a set of actions is based on the result produced by the feedback component. The recommendations may depend, for instance,

on a particular data source(s) parameters (e.g., range, resolution, accuracy, etc.), network structure (e.g., topology, configuration, data transmission technology, etc.) and are determined by the user and application requirements. Examples of the recommendations on actions may be switching to another network transport protocol or switching to another available data source. Figure 2.7 represents the full MLIN integrated components' decomposition, and the interaction flows between these components.

The MLIN design approach presented in this section offers multiple advantages. First, the proposed MLIN architecture and feedback mechanism adopts a comprehensive system integration perspective we pursue in our research. By considering the interrelationships between distinct MLIN components, we proposed a holistic MLIN system design. This integration enables the employment of the interrelationships for more complex and accurate DQ evaluation, considering how DQ is affected by each MLIN components over the overall data life-cycle in MLIN. Additionally, the developed MLIN integrated structure allows to dynamically gauge system components' parameters and reconfigure them. Our design demonstrates how the distinct MLIN components interact on a system level, and enables combining their parameters into a single structure. Furthermore, our major contribution is the incorporation of the structure adjustment feedback component within the MLIN architecture. This feedback mechanism allows for dynamic adaptation and refinement of the MLIN structure based on the ML application robustness.

2.4 Data Quality Generic Calculus

DQ typically refers to the degree to how the specific data satisfies the user or application requirements in terms of its intended use [107]. Various metrics might be employed to measure DQ, including accuracy, completeness, consistency, timeliness, relevance, and reliability. High quality data is crucial for effective decision-making, analysis, and the successful operation of systems and applications that rely on data inputs. Depending on data modality and user requirements, DQ measurement and evaluation may involve various techniques and methodologies. Some examples are: data profiling, which examines the characteristics and statistics, and evaluates how a particular data satisfies them; conducting data validation and verification; and considering how data and its metadata change over their life-cycle.

The definition of DQ has evolved over the last few decades due to the evolution on the areas of information and communication technologies. Initially, DQ was often associated with traditional notions of accuracy and completeness, which are determined by sensor systems and devices that

produce this data. In other words, the concept of DQ was inherently related to the "quality" of the data source. However, the increase in data volumes and its complexity lead to the "Big Data" concept emergence, which expanded the scope of DQ to include other dimensions such as consistency, relevance, and timeliness. Additionally, with the recognition of data as a strategic asset and the active development of data-driven systems, such as ML- and AI-based ones, DQ became to be considered as a critical factor for developing reliable intelligent data-driven systems. ML systems heavily rely on high-quality training data to produce accurate and reliable models. Poor DQ can lead to inaccurate models and reduced ML performance. In ML domain, such training DQ issues, as missing values, inconsistencies, data integrity violations, and data biases, can significantly affect ML performance during the model's execution. As a result, there is an increased focus on ensuring DQ throughout the ML pipeline, from data collection and pre-processing to model training and evaluation. Moreover, the focus on DQ extends beyond technical aspects and encompasses broader societal considerations, emphasizing the need for transparency, accountability, interpretability, and ethical data practices in ML applications.

In this paper, we follow our approach developed in [107], and define DQ as the degree to which the data satisfies application requirements (in our present case, ML end application requirements at the execution stage). To satisfy ML application requirements, the quality of input data has to correspond to the modality, format, input size, training data distribution, and any other characteristics provided by the ML application's specification. Since in MLIN the components are integrated with each other, each of them might affect DQ at any point of data life-cycle. In generic case, DQ can be formalized according to (2.1).

$$DQ_i = F(DQ_{DS_i}, DQ_{NT_i}, DQ_{ML_i}), \tag{2.1}$$

where DQ_i is the integrated DQ value of the i-th data sample; DQ_{DS_i} is the DQ value of the i-th data sample determined by the DS – data source; DQ_{NT_i} is the DQ value of the i-th data sample determined by the NT – network transmission; DQ_{ML_i} is the DQ value of the i-th data sample determined by the ML – ML application; and $F(\cdot)$ is the function that is used for integrating all the DQ values into the final one.

Based on (2.1), MLIN systems introduce several aspects that can affect DQ throughout the data life-cycle.

2.4.1 Data Source

The quality of data generated by data sources effectively impacts the overall DQ in MLIN systems. Examples of DQ issues at the data source include sensor inaccuracies measurement errors, or data processing failures. For instance, in an Internet of Things (IoT) application monitoring environmental conditions, a malfunctioning sensor can produce poor quality data, that does not reflect the real current environmental characteristics (e.g., the sensor says that it is +30°C outside, when it is snowing and freezing). DQ_{DS} evaluation is a crucial aspect of measuring DQ based on the data source's characteristics that produces the data. It might involve assessing various technical characteristics, credibility, and overall quality of the data-producing entities and devices. In general case, the data source-related DQ is evaluated according to (2.2).

$$DQ_{DS_i} = f_{DS}(DS_{i_{m_1}}, DS_{i_{m_2}}, \ldots, DS_{i_{m_n}}), \tag{2.2}$$

where $DS_{i_{m_1}}$ is the 1-st metric, based on which data source DS that produced the data sample i is evaluated; n is the overall number of DS data source's metrics; and $f_{DS}(\cdot)$ is the function based on which DQ_{DS_i} is evaluated.

The choice of specific data source's DQ evaluation metrics depends on numerous factors, such as the type of the data source (e.g., sensor device or database) and modality of the produced data (e.g., visual graphics or temperature measurements). Some examples of the metrics' evaluation techniques are provided below.

- Sensor Device Measurements' Accuracy: evaluating the sensor device accuracy is essential to ensure reliable data production. This procedure involves examining the calibration procedures, accuracy specifications, and maintenance practices of the sensor devices. Guidance for such practices typically can be found in the sensor device specification or documentation.
- Sensor Data Validation and Quality Controls: assessing the data validation and quality control processes implemented in the sensor devices is crucial. This includes reviewing the algorithms, filters, and error detection mechanisms employed by the sensor devices. Quality control measures help ensuring generated data accuracy and consistency, and avoidance of excessive noise and outliers.
- Sensor Device Maintenance and Monitoring: maintenance and monitoring practices are necessary to ensure sensor device's ongoing performance and high DQ. This involves assessing the maintenance schedules, calibration frequency, and proactive sensor device monitoring.

Proper maintenance and monitoring practices contribute to the reliability and longevity of the sensor devices, and the quality of the data they produce.

- Sensor Device Documentation and Specifications: utilizing the documentation and specifications associated with the sensor devices is vital for understanding their technical characteristics and quality. This includes reviewing technical specifications, standards, user manuals, and calibration certificates provided by the sensor manufacturers. Well-documented and comprehensive information about the sensor devices enhances transparency and contributes to developing adequate DQ evaluation tools for the intended data collection purposes.
- Sensor Device Compliance and Certifications: assessing whether the sensor devices comply with standards for the employed application domain or hold relevant certifications is an important aspect of data source evaluation. This involves verifying if the sensor device adheres to specific regulations, standards, or certifications related to accuracy, precision, and performance. Compliance satisfaction can be used as the additional criteria that provides assurance of the sensor device's quality and reliability.

These data source's DQ evaluation metric and technique examples provide insights how the sensor device's intrinsic technical characteristics at the data production moment might be employed to evaluate the qulity of data they produce.

2.4.2 Network Transmission

Network facilities is another component involved in MLIN data life-cycle and is responsible for conveying the data from the data source to ML application. Network QoS is a set of technologies that enables prioritizing network-dependent applications and allocating the required network services for them, even in the conditions of limited communication capacity [159]. QoS allows the end user to specify the priority in which the specific network traffic should be processed, and the amount of bandwidth afforded to this traffic. Network QoS degradation refers to the deterioration of various services in the network due to different factors, such as congestion, packet loss, latency, jitter, distance between nodes, components' misconfiguration, or malicious attacks. As we demonstrate in sec. 4.2, network QoS degradation can affect the quality of transmitted data. DQ can be decreased in different ways, depending on the type and "sensitivity" of the data to the degradation factors. For example, real-time data traffic that puts high demands on the network services, such as voice over IP (VoIP), video streaming services, and Wireless Multimedia Sensor Networks (WMSNs) have high sensitivity to latency and jitter. In case of network QoS degradation, the data trans-

mitted over the network might appear corrupted at the receiving end, or might be lost completely. Because of the communication issues, the end network-related applications may experience delays, interruptions, or failures, which can severely impact their performance and the experience of the end users, whose requirements are expected to be satisfied.

Depending on the network QoS degradation conditions, the quality of transmitted data can be affected in various manners depending on the format and sensitivity of this data. As we demonstrate in sec. 4.2.2, network QoS degradation can result in various image quality issues. Increased latency and packet loss can lead to slow image files transmission, noise and artifacts in the image, or missing image parts. Insufficient receiving buffer socket size may also result in partial or complete missing of the image. In case of the video streaming, high latency may cause slow video buffering, interruptions, or delays in playback. Packet loss can result in missed video frames or visual artifacts and noise, affecting the smoothness, continuity, and quality of the video stream. Excessive jitter can lead to video stuttering or unsynchronized audio and video tracks, degrading the overall viewing experience. Insufficient bandwidth may result in reduced video quality due to excessive compression. Similar effects might be observed in voice recordings, as network QoS degradation may affect their clarity and intelligibility. In the case of sensor data, network disruptions may impact the integrity and accuracy of the transmitted measurements. Increased latency jeopardizes the timeliness of the sensor data, as at the measurements might be already outdated upon receiving. Packet loss may result in missing or corrupted measurements, compromising the accuracy and completeness of the collected data. Network QoS drop may affect the transmission of text data as well. Increased delay results in slower response times when retrieving text-based responses from remote applications. Packet losses lead to missing or incomplete text data, which may prevent the appropriate interpretations. Jitter can disrupt the order of text messages or data packets, affecting the text responses order and integrity.

As one can see, DQ might be affected in numerous ways during the network transmission. In general case, network-related DQ is calculated according to (2.3).

$$DQ_{NT_i} = f_{NT}(NT_{i_{m_1}}, NT_{i_{m_2}}, \ldots, NT_{i_{m_n}}), \tag{2.3}$$

where $NT_{i_{m_1}}$ is the 1-st metric, based on which NT network transmission DQ is evaluated for the transferred data sample i; n is the overall number of NT network transmission metrics; and $f_{NT}(\cdot)$ is the function based on which DQ_{NT_i} is evaluated.

Various tools and techniques might be employed to prevent or mitigate network QoS degradation, such as traffic classification, prioritization, queuing, shaping, policing, bandwidth management,

congestion avoidance, and load balancing. These tools and techniques help to optimize the network performance and ensure the delivery of high-quality data for different applications and traffic flows. However, these network performance-oriented tools and techniques are commonly not sufficient to effectively assure ML end application robustness in MLIN systems due to the interrelationships among the integrated components. ML application's robustness cannot be thoroughly addressed by solely considering network metrics or adjusting individual network parameters in isolation of the ML application performance.

In this work, we switch the traditional network optimization paradigm that mostly focuses on network QoS metrics. Instead, in our MLIN design, we employ ML application robustness as the primary metric for the network adjustment actions. This is facilitated by introducing the specific ML application requirements in the combination with the interrelationships between network facilities and ML application robustness. MLIN interrelatioships form internal knowledge about how one or another network parameter affects ML application performance. For example, as we show in sec. 4.2.2, 10% packet loss during the image file transfer over the network may decrease ML end image classifier performance by 5-10%. Knowledge about this interrelationsip between the network packet loss and ML image classifier performance might be employed for developing network adjustment rules for assuring ML end application robustness to DQ variations. By switching the focus to ML end application robustness, network adjustment actions can be tailored to address the specific needs of the ML application.

2.4.3 Cloud-Based ML Application

DQ can also be affected by the cloud-based ML end application itself. In MLIN, ML end application is based on the remote or cloud-based server available over the network. This means that ML end application operation depends on the cyberinfrastructure maintained by the application's owner or a third party which provides such a service. Hence, various cyberinfrastrucure failures might affect both the DQ and ML application's operation. From the practical perspective, if ML server experiences downtime or becomes unresponsive, data transmission to the server may be interrupted or failed. This can cause losses in separate samples and incomplete overall data collection, affecting the quality of the data employed by the ML end application. Software bugs or crashes in ML server's cyberinfrastructure can also affect DQ during its processing and storage. This may result in incorrect data handling or corruption, which decrease DQ.

Data pre-processing is commonly employed in ML applications to improve the quality of input data and make it acceptable for processing by the ML model. While the primary goal of data

pre-processing is often to improve ML application performance over the provided input, there are scenarios where it can inadvertently affect both input DQ and ML performance. Below we discuss practical examples of how DQ issues that can arise at pre-processing stage.

- Data incompleteness issues: missing data is a common challenge for data-driven systems, especially in the execution stage, when pre-trained ML model has to provide output for the unseen sample. Incomplete data can introduce biases in ML model's decision, which affects its performance. If the handling of missing data is performed improperly, it can lead to erroneous or substituted values that do not accurately reflect the original missing data, compromising DQ.
- Outliers in the input data: outliers are extreme values that deviate significantly from the rest of the data's distribution. Outliers' treatment aims at identifying and addressing these extreme samples to improve the overall DQ. However, in some cases, outliers might represent natural inherent characteristic of the measured object or phenomena, and represent the pattern of interest for the ML application. If outlier detection methods themselves are not reliable enough, or if outliers are mistakenly removed or modified, it may distort the distribution and characteristics of the input data, impacting DQ and ML application performance.
- Data encoding and transformation issues: encoding categorical variables (e.g., one-hot encoding) and various data transformations (e.g., affine transformation) are common pre-processing methods. However, if encoding methods are applied incorrectly or if data transformation techniques are not appropriately chosen or implemented, they may result in the loss of important features or excessively augment the input data. These issues may affect the integrity and accuracy of the input DQ and decrease the ML application performance.
- Feature scaling and normalization issues: scaling numerical features to a common range is often employed in ML systems to make one features comparable with the other ones, convert features into the format acceptable by ML model for proper decision-making, and prevent make the dataset more balanced. However, inappropriate scaling or normalization methods and techniques selection, or their incorrect implementation may erroneously alter the data distribution or relative features' importance, leading to a drop in DQ and ML application performance.
- Data sampling issues: in some cases, data sampling techniques like undersampling or oversampling (e.g., SMOTE [33]) are employed to address the problems of imbalance or insufficient samples in the data collection. However, improper sampling methods selection or insufficient sampling biases analysis may result in the loss of important relationships derived from

the data or the introduction of unnecessary artificial patterns, ultimately affecting the data collection representativeness, DQ, and ML application performance.

- Data integration issues: data integration involves fusing samples from multiple data sources or merging distinct data collections. There are various methods and tools employed for data fusion, such as Kalman Filtering and Semantic Integration. Careless data integration methods and techniques selection, or mismatches in data formats may introduce errors or inconsistencies into the integrated data, undermining DQ, affecting data analysis, and the overall ML application performance demonstrated over this data.

As one can see, data pre-processing procedures applied to the data on ML cloud-based server may both improve or deteriorate DQ. In relation to cloud-based ML application-related DQ evaluation, it is reasonable to employ DQ metrics that reflect the "usefulness" of the data for the ML application. The usefulness might be determined based on the ML application requirements, provided by the end user. As an example of the metrics, performance demonstrated by the ML application over the specific data sample, or by the relevance of the data to the task ML application is expected to solve might be employed. In general case, the cloud-based ML application-related DQ is determined according to (2.4).

$$DQ_{ML_i} = f_{ML}(ML_{i_{m_1}}, ML_{i_{m_2}}, \ldots, ML_{i_{m_n}}), \tag{2.4}$$

where $ML_{i_{m_1}}$ is the 1-st metric, based on which cloud-based ML application DQ is evaluated for the transferred data sample i; n is the overall number of ML cloud-based ML application DQ metrics; and $f_{ML}(\cdot)$ is the function based on which DQ_{ML_i} is evaluated.

2.5 Reputation and Trust Generic Calculus

Reputation and Trust concepts have a solid history in various disciplines, such as philosophy, sociology, and economics. They are related to the notions of collective evaluation and social cooperation. In computer systems, Reputation and Trust emerged as a way to deal with various challenges and opportunities of distributed networks, such as peer-to-peer networks, e-commerce applications, web services, etc. These applications commonly involve large-scale, heterogeneous, dynamic, and uncertain interactions between nodes that may not have prior information about each other or direct interaction. A practical example of such decentralized applications are sensor or IoT networks, where the devices might dynamically connect and drop from the network.

Reputation and Trust-based mechanisms are vital for ensuring the security and reliability of interactions in these decentralized networks, as they allow to mitigate attacks against which the conventional security methods and techniques are ineffective [48,147]. The employment of Reputation and Trust enables considering and evaluating the previous "behavior" of decentralized nodes in the system [55,141,229]. This allows accumulating historical knowledge about the actions performed by the particular node, and its interactions with other nodes in the system. This historical knowledge is then employed to evaluate the trustworthiness of the nodes in the systems, which represents to which degree the node might be trusted based on its previous actions. Enhancing Trust towards the particular node means building confidence in the fact that this node will perform the expected action (e.g., provides data that corresponds the established requirements).

As can be observed from sec. 2.4, multiple data sources might produce data of varied quality due to numerous factors. In addition, even a single data source might produce data of various quality in distinct time moments (e.g., due to hardware depreciation, outdated software, or malicious attack). Considering the need to select data sources providing DQ that is required to assure ML application robustness, Reputation and Trust mechanisms can be employed to benefit MLIN systems. Relying on Reputation and Trust indicators, the selection of data sources can be performed in a more intelligent, optimized, and secure way, as they allow to choose only those data sources that provide reliable data and best satisfy the ML application requirements. Moreover, as we demonstrated on the practical examples of autonomous vehicles [48,52,53] and FL [42], Reputation and Trust-based mechanisms effectively enhance the security in distributed systems. Reputation and Trust indicators enable identifying data sources that provide incorrect or erroneous data, and discarding them from the further communication. Following [48,53,55], we define the employed indicators below and describe how they used in MLIN.

In our approach, we employed three basic models: $Truth$, *Reputation* (R), and $Trust$.

- $Truth$ is an indicator characterizing how the DQ of the data sample i satisfies the established ML application requirements. This value can be formalized according to (2.5).

$$Truth_t = f_{Tr_t}(DQ_i), \tag{2.5}$$

  where $Truth_t$ is a value representing how the DQ of the i-th data sample satisfies the established requirements in the time moment t, and $f_{Tr_t}(\cdot)$ is the function employed to evaluate $Truth$ in the time moment t.

- Reputation (R) is an indicator which reflects the data source's historical "behavior" since the

system operation beginning. R is an indicator that accumulates history and is based on the combination of the current and previous $Truth$ values. R value is formalized according to (2.6).

$$R_t = f_{R_t}(Truth_t) = f_{R_t}(f_{Tr_t}(DQ_i)), \tag{2.6}$$

where R_t is the data source's reputation value at t time moment, and $f_{R_t}(\cdot)$ is the function employed to calculate R value at t time moment.

Based on the above indicators, the historical knowledge on the DQ provided by data sources can be gathered. Moreover, this knowledge allows to track the temporal changes in the provided DQ and, based on these changes, analyze and diagnose the system for the potential failures [53]. However, using only the R value to decide if the data source is trusted or not limits the flexibility of the system, makes it too sensitive to outliers, and might introduce excessive False Positive and False Negative errors. To mitigate these issues, we introduce $Trust$ indicator in addition to the previous ones, which allows regulating the sensitivity to the changes in DQ values and making the Trust evaluation more robust.

- $Trust$ is an indicator calculated based on the combination of R value at the preceding time moment $t-1$ and current $Truth$ value. $Trust$ can be formalized according to (2.7).

$$Trust_t = f_{Trust_t}(R_{t-1}, Truth_t) = f_{Trust_t}(f_{R_{t-1}}(f_{Tr_{t-1}}(DQ_i)), f_{Tr_t}(DQ_i)), \tag{2.7}$$

where $Trust_t$ is the $Trust$ value calculated at t time moment, and $f_{Trust_t}(\cdot)$ is the function employed to calculate $Trust$ at t time moment.

Each of the presented indicators are based on the functions, which are derived from the established ML application requirements. However, based on our empirical evaluations [42, 48, 53, 55], the indicators provide best results if normalized in the range between 0 and 1. For example, the closer the $Trust$ value to 1, the highest trust towards the data source is established. Based on this, we introduce the following assumptions for the employed indecators' values:

- $Truth \in [0, 1]$;
- $R \in [0, 1]$;
- $Trust \in [0, 1]$.

Alternatively, the process of Reputation and Trust indicators evaluation can be performed by the decentralized data sources. If required by the ML application and end user architecture (e.g., peer-to-peer or ad-hoc network topology), data sources can communicate with each other directly and perform the evaluation. Below, we describe how specifically the Reputation and Trust indicators are calculated by data sources themselves in a decentralized manner.

Based on the previously introduced definitions, data sources evaluation requires calculating the $Truth$, R and $Trust$ indicators produced by each data source. Suppose that $e \in E, E$ is a set of data sources, that can communicate with each other through the network facilities. Then:

- $\overline{Truth_e} = \begin{pmatrix} \dots \\ Truth_{e_i} \\ \dots \end{pmatrix}$, where $Truth_{e_i}$ is the Truth value of $e, e_i \in E, e_i \neq e, i = 1 \dots |E|$;
- $\overline{R_e} = \begin{pmatrix} \dots \\ R_{e_i} \\ \dots \end{pmatrix}$, where R_{e_i} is the Reputation of the data source e_i, calculated by data source $e, e_i \in E, e_i \neq e, i = 1 \dots |E|$; and
- $\overline{Trust_e} = \begin{pmatrix} \dots \\ Trust_{e_i} \\ \dots \end{pmatrix}$, where $Trust_{e_i}$ is the Trust value to data source e_i calculated by the data source $e, e_i \in E, e_i \neq e, i = 1 \dots |E|$.

Below, one can find further explanation of how these indicators are calculated by the data sources in a decentralized manner.

Truth

$Truth$ assessment of the data source is based on the knowledge about this data source collected from the other data sources. If the data, based on which the data source is evaluated, is represented in the form of several blocks of information, for example, the measurements performed by the data source are given as the set of values, then the computation of the indicator is limited to averaging the $Truth$ value over all the blocks. In a formalized form, we present the calculation of this indicator for the data source e_i by the data source e, according to (2.8).

$$\overline{Truth_e^s} = \begin{pmatrix} Truth_{e_i}^{s_0} \\ \dots \\ Truth_{e_i}^{s_{bl}} \end{pmatrix}, \tag{2.8}$$

where bl is the number of data blocks, based on which the $Truth$ indicator is evaluated. In such a case, the vector of Truth's estimates for all agents can be represented as:

$$\overline{Truth_e^s} = \begin{pmatrix} \dots \\ \frac{\sum_{j=1}^{bl} Truth_{e_i}^{s_j}}{bl} \\ \dots \end{pmatrix}, \tag{2.9}$$

where $Truth_{e_i}^{s_j}$ is the estimated $Truth$ value for the data source e_i by the data block s_j.

However, when the data source does not have the ability to evaluate the data source based on the data received from another data source, then the $Truth$ value is estimated based on the average values of the indicators received from other data sources that conducted the evaluation:

$$Truth_{e_{e_i}} = \frac{\sum Truth_{e_{j_{e_i}}}}{n_{truth}}, \tag{2.10}$$

where $e \in E$ and $e_i \in E, n_{truth}$ is the number of data sources having an estimate of the $Truth$ of the e_i-th data source. If there are no such data sources, the $Truth$ value is estimated as 0.5, i.e., the average value at which the data are not assessed as correct or incorrect.

Reputation

Calculation of R value can be conducted as:

$$R_{e_{e_{i_t}}}^S = \begin{cases} R_{e_{i_{t_0}}} + \sum_{i=1}^{t} Truth_i, Truth_t \geq \alpha \\ R_{e_{i_{t_0}}} + \sum_{i=1}^{t} Truth_i - (R_{t-1} - e^{-(1-Truth_t)t}), Truth_t < \alpha \end{cases} \tag{2.11}$$

where $Truth_{e_{e_{i_t}}}$ is the Truth value, received from data source e_i by the data source e at the current time moment t, and α is the threshold for a positive or negative decision on data source's reputation value.

We consider $R^S_{e_{e_{i_t}}}$ as an intermediate step in calculating the reputation value. To calculate the final R value, we need to normalize this value $R^S_{e_{e_{i_t}}}$ over the system operation time period. At the initial time of the data sources' operation, the Reputation value can be taken equal to 0.5, i.e., with $t = 0, R^S_{e_{e_{i_t}}} = 0.5$. The value of α, at which the $Truth$ evaluated as "positive" behavior, is chosen empirically. In general, the α value can be set to 0.5. The R value can be calculated according to (2.12).

$$R_{e_{e_{i_t}}} = \frac{\sum_{t=1}^{|E|-1} R_{e_{j_{e_{i_t}}}}}{|E|}, \tag{2.12}$$

where $e_j \in E, e_j \neq e_i$. In this case, the reputation is calculated based on the other data sources' evaluations. The introduced calculus forces the R value to increase linearly and decrease exponentially. It means that the failed or malicious data sources are unable to immediately gain high reputation level.

Trust

As mentioned above, the function of assessing the $Trust$ value is a function of two parameters – the value of R for the preceding time moments and the $Truth$ value at the current time moment – and is calculated according to (2.13)).

$$Trust_{e_{e_{i_t}}} = f(R_{e_{e_{i_{t-1}}}}, Truth_{e_{e_{i_t}}}) \tag{2.13}$$

The overall data source's $Trust$ evaluation boils down to the comparison of the $Trust$ value against the given threshold, and is defined according to (2.14).

$$Trust_{e_{e_{i_t}}} \geq \alpha_{trust} \tag{2.14}$$

If the condition in equation (2.14) is met, the "behavior" of the data source is assessed as acceptable. The function of calculating the Reputation value can be represented as a function built on weights. In this case, the values of $Truth$ and R are taken into account when calculating the $Trust$ value with some coefficients characterizing the effect on the calculated value of each indicator. In a generalized form, this function can be represented according to (2.15).

$$Trust_{e_{e_{i_t}}} = \gamma Truth_{e_{e_{i_t}}} + (1-\gamma) R_{e_{e_{i_{t-1}}}}, \gamma \in [0,1], \tag{2.15}$$

where γ determines the system reactivity coefficient, which is provided by the end application and user requirements.

In general, the Reputation of the data source is formulated as a function that depends on this data source's Reputation values calculated at previous time moments. With the fully linear Reputation function, when it increases and decreases linearly, the data source's "behavior" can be estimated incorrectly, as, during a long observation period, the Reputation might not change quickly enough with a sharp change in the data source's actions. To address this challenge, we have introduced the exponential R decrease and the $Trust$ indicator that allows to regulate how "sensitive" is the Trust evaluation towards the changes in the R value.

2.6 Robustness Calculus that Integrates DQ and ML performance

DQ is a crucial factor for the success of any ML application that relies on data as its input [183]. However, DQ is not a static property of the data, but rather dynamic in terms of the content and context, and is influenced by various factors such as the data source, the data fusion processes, the ML model, and the end user requirements. Therefore, DQ evaluation in MLIN is a challenging task that requires a comprehensive and adaptive approach, which we introduced in sec. 2.4. Reputation and Trust, which we introduced in sec. 2.5, are two important concepts that contribute to evaluating and selecting data sources that provide high quality data for ML applications. Although Reputation and Trust is calculated based on the DQ provided by the data source, in some scenarios data from multiple sources might be fused before being processed by the ML end application. In this case, multiple data sources likely have various Reputation and Trust indicators values, which might be used for selecting data sources that provide the required DQ for the data fusion operation. After the data fusion, the resulting input data is submitted to processing by ML application, and is used for calculating ML robustness. Below we discuss this approach's advantages.

Conventional approaches to ML robustness evaluation and improvement, reviewed in sec. 2.1 and 2.2, often consider it in relation to only one system component, such as the ML model or data pre-processing techniques. Despite these approaches have provided valuable insights into enhancing the specific aspects of ML robustness, they commonly do not consider the interrelationships between system components in MLINs and their influence on ML application performance and robustness. In MLINs, the interrelationships between system components, such as the data source, network

facilities, and ML application, play a crucial role in determining the overall ML robustness. The quality of the data in MLINs, for example, can be influenced not only by the data sources, but also by the network conditions (see sec. 4.2). Neglecting MLIN interrelationships may result in inaccurate and ill-defined ML robustness definition, which will not reflect the inherent aspects of how the MLIN integrated components affect ML application robustness. We see the following advantages in our novel approach that utilizes the integrated DQ indicator influenced by each MLIN component, and the performance of the ML application to calculate ML application robustness:

- Taking into account the influence of data sources on the DQ value: by incorporating optimal data sources selection mechanisms, our approach enables choosing data sources that consistently provide DQ that satisfies ML application requirements. The selection process ensures that the ML application is fed with the data that satisfies ML application input format and specification, ensuring the performance of the resulting ML model's outputs and its robustness towards DQ variations. By considering the DQ provided by the data sources, we can prioritize those ones who provide better DQ, reducing the risk of utilizing failed or compromised sources.

- Adaptability to dynamic environments: in dynamic network environments, the data sources' availability and DQ provided by them may vary. By incorporating the input DQ into the ML robustness evaluation calculus, our approach allows MLIN system adjustments based on specific ML application and end user requirements. This adaptability ensures that ML models maintain performance even in the face of dynamic cyberinfrastructure conditions, which makes them robust to DQ variations.

- ML application robustness assurance: integrating data sources selection based on DQ assessment and ML robustness evaluation contributes to improving the overall MLIN system performance. Adaptive data sources selection based on the quality of data provided benefits the performance demonstrated by an ML application, which results in more reliable outputs and robustness to the cases when DQ goes down.

Since DQ is affected by each of the MLIN components, evaluating and ensuring ML application robustness relying on the integration between these components is more advantageous than considering each component in isolation. By taking into account the interrelationships between the components, we can better understand how they affect each other and how they influence the overall system's performance. For example, we can identify and eliminate unreliable sources that consistently produce low quality data (e.g., using the Reputation and Trust indicators introduced

in sec. 2.5), we can optimize the data fusion process to enhance the DQ (e.g., selecting sensors based on their DQ with Genetic Algorithms, described in sec. 3.4), we can employ re-training or adversarial training techniques to adapt to the changes in DQ (e.g., using TL or FL), described in sec. 5.2). By doing this, we can increase the ML application robustness of the ML application and ensure its reliability and usability in different scenarios when DQ may vary.

In our work, we define the ML application robustness through the relationship between the input DQ and the ML performance based on this input. Below, we review and analyze major metrics affecting ML application robustness and integrate them into the overall robustness calculation. We base the robustness metric calculus on the combination of the input DQ and the performance of the ML application demonstrated based on this input data. It is possible to divide this step as follows. In general, ML application robustness ($\mathcal{RB}$) is defined according to (2.16) as a function of two variables.

$$\mathcal{RB} = \frac{\Delta\mathcal{DQ}}{\Delta\mathcal{PRF}}, \ \Delta\mathcal{PRF} \neq 0, \tag{2.16}$$

where $\mathcal{DQ}$ represents the DQ values over a specified data inputs set: $DQ_i, \ldots, DQ_n \in \mathcal{DQ}$, and $PRF_i, \ldots, PRF_n \in \mathcal{PRF}$ depicts the set of ML application performance values demonstrated over this inputs' set.

According to the DQ calculus (2.1) introduced in sec. 2.4, the quality of data input depends on multiple MLIN components. This means that the way how the separate MLIN components are integrated into a single *structure* directly affects the input DQ and ML application robustness. The structure here represents the specific set of MLIN parameters, related to each of the MLIN components, that can be modified and adjusted during the MLIN execution. Some practical examples of these parameters are: the protocol employed by the network facilities to convey the data from the data source to the ML application; the particular data sources employed for the data collection; or the data pre-processing methods employed by the ML application. Our major goal is to assure ML application robustness to satisfy specified requirements, which means that we need to find the appropriate MLIN system structure that contributes to satisfying those requirements. For this, we introduce the formal definition of MLIN structure, which is defined according to (2.17).

$$STR_j = f_{STR}(pr_1, pr_2, \ldots, pr_m), STR_j \in \mathcal{STR}, \tag{2.17}$$

where STR_j is the j-th MLIN structure, pr_1 is the 1-st MLIN parameter that determines STR_j, m is the overall number of parameters that determines STR_j, and $\mathcal{STR}$ is the set of all possible

MLIN structures available based on the parameters that can be modified or adjusted.

As one can see, MLIN structure is determined by the set of parameters and their values. These parameters are defined by MLIN components and cyberinfrastructure characteristics. In other words, STR defines the MLIN configuration which results in a particular ML application robustness level. As we discussed in sec. 2.3, the input DQ value is affected by all MLIN components and cyberinfrastructure, likewise the robustness value that incorporates DQ (see (2.16)). Therefore, the major goal of our work is to find such MLIN structure that satisfies the established ML application requirements. Formally, to assure ML application robustness at the specified level β, we have to find the corresponding structure STR_j, which can be formalized as (2.18).

$$\text{find } STR_j \Rightarrow \mathcal{RB}_j \geq \beta, \tag{2.18}$$

where $\mathcal{RB}_j$ is the ML application robustness value demonstrated over the STR_j MLIN structure, and β is ML application robustness required value, specified by a user or application requirements.

2.7 Conclusion

In this chapter, we introduced our novel approach to define, calculate, and assure ML application robustness to DQ variations. We reviewed the most recent advances and perspectives on ML robustness presented in the publications, classified them into various categories, and discussed their advantages and flaws. Unfortunately, the majority of the reviewed state-of-the-art ML robustness perspectives rather limited in scope, quantitative measures, and the definition itself that might be fuzzy or ill-defined. In contrast to other conventional approaches, we considered MLIN from the system integration perspective, which allowed us to leverage its integrated structure and the interrelationships between MLIN components in order to generate MLIN adjustment feedback aimed at ML application robustness assurance. Below we list the major contributions we developed in this chapter.

- In sec. 2.3, we developed and presented the **novel MLIN integrated architecture**, which incorporates the **ML adjustment feedback component**. The architecture delineates the composition of each component, the interaction between the components, and the data lifecycle in MLIN. In addition, on the high-level it presents how the feedback is generated and how the recommendations on MLIN adjustment actions are conveyed to the components.

- In sec. 2.4, we **defined DQ** in terms of our research, and **developed and presented the DQ generic calculus**, which is based on calculating the DQ value related to each of the MLIN components. We leveraged MLIN integrated structure to integrate DQ metrics pertaining to various components into a single final value.
- In sec. 2.5, we **defined the Reputation and Trust indicators**, and described how they can be employed to select reliable data sources that consistently provide high quality data. We **developed the generic Reputation and Trust calculus**, and outlined how the history of DQ provided by the data sources is accumulated and employed in the calculation of Reputation and Trust indicators.
- In sec. 2.6, we **developed and presented our ML application robustness definition**, and how we employed it in the context of MLIN. Our **ML application robustness generic calculus** that integrates the DQ and ML application performance, which allows to **(1)** capture the interrelationships between MLIN components; **(2)** evaluate the ML application robustness in real time as it does not require ground truth; and **(3)** quantify the ML application robustness measure. We **demonstrated how we can ensure ML application robustness to DQ variations** according to a specified requirements by searching and selecting the **MLIN structure** that satisfies these requirements.

Our approach has several advantages for assuring ML applications robustness from DQ variations in the MLIN architecture. First, using the DQ and Reputation and Trust indicators, it allows selecting data sources that provide high quality data consistently and reliably during the whole system operation period. Second, using the DQ generic properties, it allows fusing data from multiple sources, which means that we can improve the overall input DQ by aggregating the data based on some criteria. In sec. 3.4.1, we demonstrate how this data sources selection can be optimized and realized in practice. Third, our ML application robustness definition and calculus allows to take into account the impact of DQ on the ML performance, which means that we can monitor and adjust the MLIN structure according to the changes in DQ over time. Fourth, our DQ, Reputation and Trust, and ML application robustness generic calculi are flexible enough to adapt our approach to different domains, contexts, and user requirements, which means that the end user can customize them according to the specific needs of their application.

Chapter 3

Intelligent Data Sources Selection based on Data Quality and Security Integration

In this chapter, we develop and present our innovative approach to intelligent data sources selection, which is intended to be employed as an integral part for MLIN restructuring to satisfy the ML application requirements. In our approach, we integrate DQ provided by data sources with their platforms' security. The paramount challenges we tackle encompass the multi-modality of data and its diverse origin, demanding a sophisticated data fusion approach. By integrating DQ and platform security metrics, we empower the selection process by making security an integral component of the data sources selection. We introduce our multi-modal and multi-platform data sources selection framework, which leverages our DQ calculus and Genetic Algorithms to optimize the data sources selection process and provide results in real time. We validate our approach through a practical use case, which relies on our extensive knowledge base derived from real-world data source-incorporating mobile devices. Below we overview the content presented in each section of this chapter.

- In sec. 3.1, we discuss the significant challenges inherent in real-world DQ evaluation. We describe how we are going to evaluate DQ in our practical use case, and disclose our motivation for the selected practical use case.
- In sec. 3.2, we present our integration framework architecture, engineered to handle the multi-modal and multi-platform data sources selection. We present the architecture as a

multiple levels, with the principal integration level orchestrating the fusion of multi-modal data originating from diverse devices. Auxiliary sub-levels are responsible for the computation of DQ metrics, platform security evaluations, and the integration of existing knowledge and requirements for further data fusion.

- In 3.4, we offer an in-depth exploration of a practical use case based on developing a mobile application for the medical research purposes. This case emphasizes the variability in DQ originating from diverse data sources, in particular produced by personal devices such as smartphones employed for the data collection.

- Sec. 3.4.1 and 3.4.2 are dedicated to formalizing the data sources selection task and describing our novel DQ calculus. This multi-layered calculus integrates DQ and platform security metrics, aggregating diverse measurements into an overall DQ value, which we employ as a major indicator after the data fusion. Through concrete examples within the context of mobile devices, we demonstrate illustrative practical examples of calculating various metrics.

- In 3.5, we verify our GA-based sensor selection approach in a practical use case employing real-world mobile devices. We empirically evaluate its performance and computational efficiency compared to the conventional brute force-based search.

- In 3.6, we emphasize the practical implementation of the presented approach by detailing the mobile software applications, which implement the methods and tools developed in this chapter. These applications embed our calculus and GA-based data sources selection technique. Additionally, we elaborate on our collected sensor devices' characteristics knowledge base, made publicly available as well.

As the major chapter's contribution, we develop and present an innovative approach to autonomous and optimized data sources selection in modern data sourcing landscapes, which we integrate into MLIN. This integration allows adjusting MLIN structure in order to satisfy the ML application performance requirements and improving ML application robustness. This chapter offers an intelligent framework that allows to address the complexities of multi-modality and platform diversity in the data fusion process through the fusion of DQ and platform security, and allows to select data sources that satisfy DQ and ML application performance requirements.

3.1 Data Quality Evaluation in Practice: Challenges and Our Approach

As we discussed in sec. 2.4, in the generic case, metrics for gauging DQ are determined by the user needs and application requirements. To move the generic calculus into practice, we need to consider the specific characteristics and challenges of the data domain and the data sources that we are dealing with. In this chapter, we focus on the mobile devices as data sources, which include smartphones, tablets, smart bands, and other wearable or portable devices, as they: (1) have become ubiquitous in the real world; (2) are responsible for generating vast amount of data used in contemporary industrial applications; (3) serve as a valuable practical example since in our research we work with real devices; and (4) pose their unique challenges in DQ evaluation, as they might possess various technical and security characteristics. Below we describe our motivation of pursuing the domain of smartphones and mobile devices for our practical DQ calculating example.

According to Statista[1], in 2025 the number of mobile users worldwide is projected to reach 7.49 billion, and the number of mobile devices is expected to reach 18.22 billion by 2025[2]. The number of smartphone mobile network subscriptions worldwide reached almost 6.6 billion in 2022 and is foretasted to exceed 7.8 billion by 2028[3]. Modern smartphones and other mobile devices, such as tablets and smart watches, intensively generate various kinds of data, such as:

- Media data: this includes various types of images, videos, sound and audio recordings, and other forms of media created by the users on their mobile devices.
- Communication data: this type of data includes all the traffic related to text messages, voice calls, video calls, emails, social media posts, and other forms of communication that users engage in on their mobile devices.
- Application data: this is data generated by various mobile applications that users install and use on their mobile devices, such as games, productivity tools, health trackers, navigation systems, etc.
- Sensor data: this type incorporates data collected by various sensors that are embedded into mobile devices, such as cameras, microphones, GPS sensors, accelerometers, gyroscopes, and biometric sensors.

[1] https://www.statista.com/statistics/218984/number-of-global-mobile-users-since-2010/
[2] https://www.statista.com/statistics/245501/multiple-mobile-device-ownership-worldwide/
[3] https://www.statista.com/statistics/330695/number-of-smartphone-users-worldwide/

A wide range of sensor devices embedded into diverse mobile sensor platforms and the readings obtained using these sensors attract the attention of data scientists, researchers, and engineers [102]. Data acquired from these data sources can be combined in external applications. For example, a step-counter might employ an accelerometer and a gyroscope, while a weather app might utilize a built-in GPS sensor. The growing number of crowdsourcing apps [72, 164], mobile crowdsensing tasks [71, 105, 106, 109, 231, 257], and the Internet of Things generate huge amount of sensor readings. Sometimes, data obtained from multiple sensor devices is aggregated together to enhance the performance and functionality of mobile applications. This process is known as data fusion and is realized by employing the aggregation approaches aimed at balancing the strengths and weaknesses of various sensors [172,200]. An example of data fusion in smartphones is improving positioning by aggregating the data obtained from gyroscope, accelerometer, magnetometer, and other sensors [194].

However, not all data sources are homogeneous, and the DQ they generate can vary significantly, which may substantially impact the produced data processing outcomes. Unfortunately, in the process of optimizing data fusion techniques, sensor system designers and developers often neglect the importance of security aspects. Their major focus is aimed at enhancing the measurements accuracy, leaving security concerns to be addressed later by security experts. On the other hand, security professionals possess expertise in designing secure sensor systems, however they might lack knowledge on important data fusion aspects related to optimization and accuracy. This disconnect in the design approach hinders the overall optimization and efficiency of sensor systems in practical applications. Hence, while data fusion techniques can enhance attributes like accuracy, they may inadvertently compromise data and data source's platform security.

In this chapter, we integrate data sources selection methods and tools into a unified framework that allows for the automation of picking up data providers that result in satisfying the required level of ML application performance. As we consider cloud-based ML application robustness in the execution stage, the quality of measurements produced by data sources may vary due to changing operational conditions. Our solution is aimed to work at both MLIN design and operation stages that allows dynamic restructuring of the system in near real-time in order to adjust to the current conditions. This operation poses additional requirements on the optimization techniques that requires the application of intelligent methods.

In our research [44, 101], we collected data, developed knowledge, and implemented in practice several methods and tools that we are now integrating into our framework described in this chapter. In [101], we developed a DQ integration calculus, which incorporated accuracy, security, and other

metrics in order to evaluate a smartphone sensor system. In [44], we expanded the developed DQ calculus from smartphones to other mobile devices and included measurements of various modalities. In this chapter, we employ our collected knowledge base on sensor-embedded mobile device characteristics to develop real-time data sources selection methods and tools based on Genetic Algorithms (GA) techniques.

Generalizing from our previous research referenced above, here we concentrate on the integration of the developed data sources selection methods and tools into a unified framework, which itself can be optimized in real-time in order to satisfy ML application performance requirements. Our framework subsumes several integration levels that are described in sec. 3.2. The novelty and major contributions of the approach proposed in this chapter include:

- an integration of the developed methods and tools into a unified data sources selection framework that can be employed in both MLIN design and operation stages;
- multi-level measurements and the data sources integration procedures (see Fig. 3.1), which incorporate:
  - DQ evaluation with both accuracy and security,
  - multi-modality data fusion,
  - multi-platform system realization,
  - and knowledge utilization;
- use-cases demonstrating how the developed methods and tools can be realized and integrated in order to select data sources embedded in various platforms;
- expansion of previously developed DQ calculus and integration into a unified framework. In our previous research [101], the DQ calculus was developed specifically for mobile platforms. Here we present a novel theoretical framework, described by our generalized DQ calculus, which can be used on any instrumentation platform equipped with real-time sensing instruments;
- demonstration of practical applications of the developed theoretical framework. In sec. 3.6, we provide a comprehensive overview of our practical contributions to the field. Based on the methods and tools we describe in this chapter, multiple Android applications and an instrument-selection knowledge-base aimed at automating the novel DQ-informed instrument selection process on Android platforms were developed.

3.2 Integration Framework for Data Sources Selection

We develop a novel data sources quality and their platform security integration framework that aims at optimizing multi-modal and multi-platform data sources' composition. The major goal of this optimization is to ensure the performance of the ML application and to improve its robustness to DQ variations. Being an inherent part of MLIN cyberinfrastructure, data sources directly influence the quality of the input data received and processed by the ML application. As we defined in sec. 2.6, ML application robustness is dependent on the combination of ML application performance and input DQ (see equation (2.16)), and ML application performance directly depends on input DQ [46]. Considering that ML application end user is interested in system robust to input DQ variation, the requirements towards ML application performance have to rely on a specific level of input DQ achieved by selecting those data sources that can satisfy this specific level. Our intelligent framework enables dynamical optimization of data sources selection and switching to those who satisfy the actual DQ requirements.

Our framework adopts multi-dimensional DQ assessment procedures, integrating various metrics that span from data source accuracy to data source platform security. Typically, data source security receives insufficient attention during the design stage [67]. Nevertheless, security violations may result in the deterioration of DQ collected from data sources, particularly when data from multiple platforms is fused, leading to overall malfunctions [44]. To address these challenges, we incorporate data source platform security as an essential DQ component, considered alongside data accuracy and other quality metrics. In our framework, security characteristics depend not only on the data source itself but also on the platform into which the data source is embedded. This approach enables a comprehensive security evaluation that accounts for the practical implementation of the data source platform. As a result, our framework can be tailored to each unique data source platform, facilitating the evaluation of its distinct security attributes.

Data aggregation from diverse data sources can effectively enhance data accuracy [238]. Currently, users often base their data source selection solely on the accuracy it provides. However, incorporating security metrics into the data source selection process could improve the overall DQ and complimentary enhance system's security. The suggested comprehensive data source selection framework integrates various methods, techniques, and knowledge to provide ML applications with data that best meets the requirements for ensuring performance and improving robustness to input DQ variations.

In Figure 5.8, we depict a schematic representation of the framework structure, illustrating how

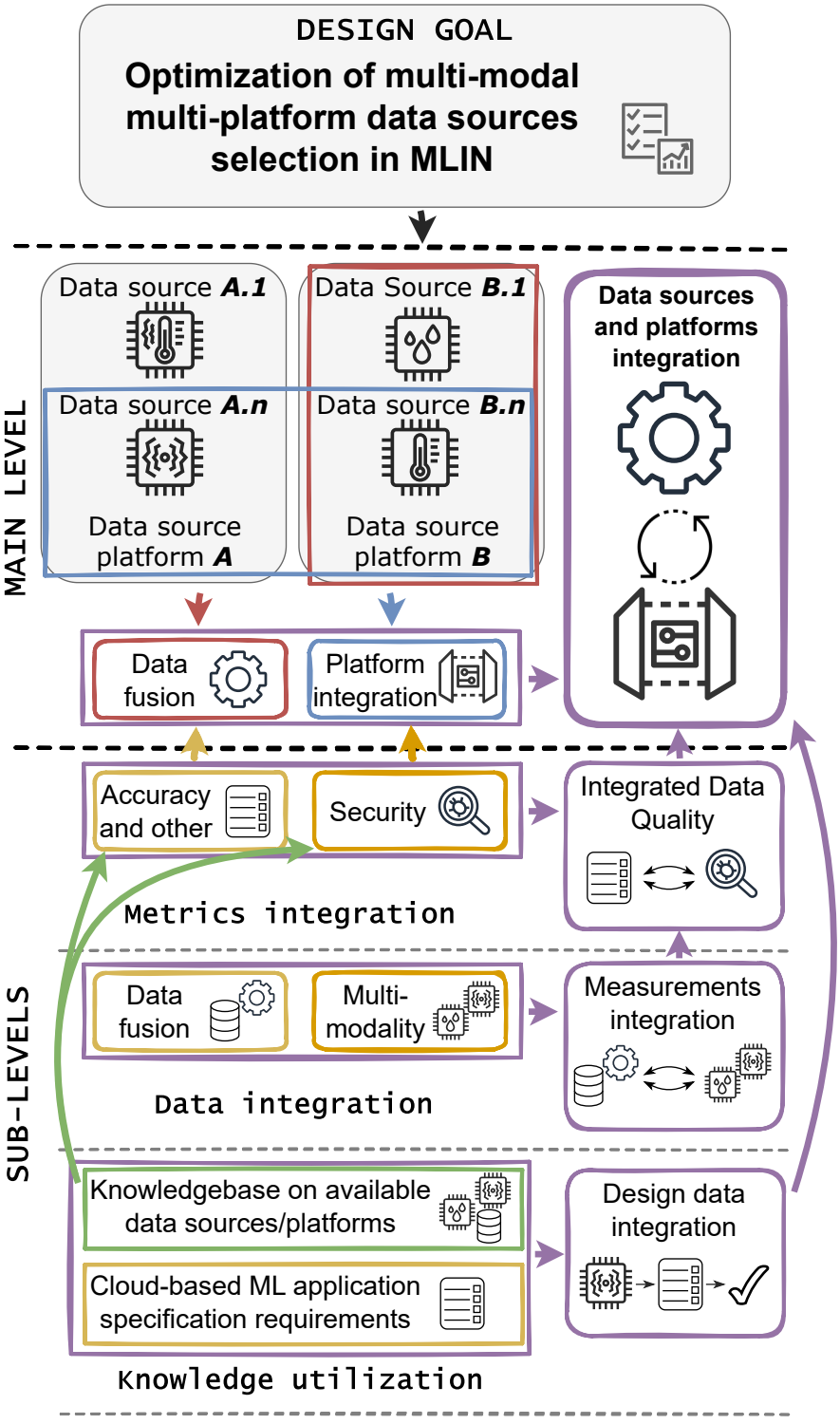


Figure 3.1: Multi-level integration procedures in framework design and operation. Violet color represents the major integration operations incorporated into the framework; red color refers to multi-modal data sources fusion; blue color refers to multi-platform data source platforms fusion; other colors are used to represent elements integrated on each sub-level

multi-layer integration is accomplished. The primary objective of our framework is to accommodate the optimization of multi-modal and multi-platform data source systems. Below the primary design goal, we introduce the main level of integration proposed by the framework. This level showcases the key operations outlined by the framework's design: data source fusion and data source platform integration, both working towards achieving the optimization goal.

Beneath the main level, we portray three sub-levels that facilitate the operations conducted at the main level. The first sub-level, metrics integration, involves the aggregation of various data characteristics, such as data source accuracy (dependent on the data source itself) and data source platform security. This integration produces a comprehensive DQ characteristic.

One layer down, the subsequent data integration sub-level encompasses the integration of sensor measurements from multiple sensor modalities and platforms. The outcome of this stage is the combined data acquired from various data sources and their platforms, followed by the aggregation of metrics to calculate the overall DQ for this data.

The bottom sub-level is knowledge utilization, where available information on data source and its platform characteristics is leveraged to manage the data source/platform integration at the main level. This sub-level also incorporates robustness specifications for ML applications, which are employed to select data sources that align with these requirements.

3.3 Data Fusion Effect on Accuracy and Security in Practice

3.3.1 Security Metrics Integration into DQ Calculus

Integrating data from diverse data sources is a well-established strategy employed by designers to enhance the accuracy and reliability of measurement data. Numerous fusion techniques have been implemented in sensor systems to achieve this goal. For example, fusing measurements from local sensors, such as cameras, LiDAR, and radar, with global sensors like global navigation satellite systems, has been shown to improve the real-time quality and robustness of the acquired data [177]. Additionally, combining inertial and vision sensors has led to advancements in data accuracy and trustworthiness [66, 85].

However, alongside the efforts to enhance data accuracy and trustworthiness, insufficient attention is dedicated to data security characteristics considering the threats to which data sources are exposed to due to various malicious attacks. For instance, Cao *et al.* [29] demonstrated a novel adversarial

attack against a multi-sensor fusion system, impacting both 3D LiDAR point cloud and camera pixels representation. Neglecting data and sensor platform security may compromise the overall effectiveness of data fusion and the resulting DQ, despite the benefits it offers.

In this chapter, we integrate data source's platform security with other characteristics into a unified DQ indicator. We consider data source's platform security as a DQ calculus inseparable component, which allows evaluating of how security conditions affect the overall DQ. We develop calculus tools, which are described in more detail in section 3.6 and demonstrate their utilization in a real-world scenario. Below we describe the practical use case of how data from various data sources' platforms can be fused and how this fusion affects the resulting DQ. In our use case, we employ smartphones and mobile devices as data sources' platforms.

3.3.2 Measurements Fusion Practical Use Case

The research organization MedResML (a fictional name) is conducting medical investigations into the feasibility of diagnosing movement system impairment syndrome based on individual motion pattern analysis. To accommodate their research, MedResML gathers measurement data from the patients' smartphones, which is then processed to extract diagnostic patterns. The data is collected using various sensors, including a gyroscope, accelerometer, pedometer, and magnetometer. MedResML chooses to leverage smartphones and other personal mobile devices (e.g., smart watches) as data collection instruments with the specially developed Android application. Since the smartphones and other mobile devices are highly affordable, commonly equipped with the necessary sensors, and can be carried during the physical activities, there is no need to equip the patients with the additional devices or sensors. After the data collection, it is being processed by the ML application, which is responsible for predicting the potential diagnosis.

To improve the measurements' quality and diagnosis predictions made based on these measurements, the company leverages the aggregation of data collected from various users and multiple smartphones and mobile devices. Since smartphones and other mobile devices are equipped with a variety of embedded multi-modal sensors with diverse technical characteristics, the fusion of data obtained from these sensors can influence the overall DQ in both positive and negative ways. An illustration of data fusion is shown in Figure 3.2, where data from "Platform A" and "Platform B" are combined, resulting in a higher DQ score. As depicted in Table 3.1, prioritizing accuracy over security can lead to higher accuracy scores but may compromise the overall DQ due to lower security levels. The majority of conventional multi-modal and multi-platform data fusion methods are commonly focus on accuracy metrics, disregarding the security aspects of the data sources'

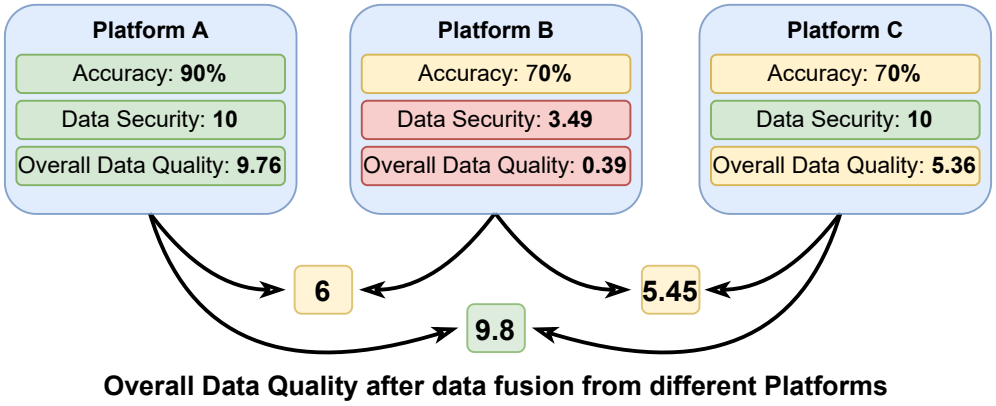


Figure 3.2: An example of fusing the data from various data sources' platforms with various accuracy and security characteristics

platforms. This accuracy-centric approach may result in favoring low-security platforms for data fusion, posing risks to research subject privacy and the trustworthiness of predictions made by the ML end system.

To assure the ML predictions reliability and mitigate the effects of security vulnerabilities to which patients' smartphones and mobile devices are exposed to, the company incorporates the data sources' platform security evaluation into their Android application. Table 3.1 showcases the example of data sources' platform overall DQ evaluation on the example of four Android smartphones. In Table 3.1, we demonstrate some examples of the characteristics, based on which the data source's security is evaluated, such as the rate of blacklisted apps and the root access status. In our example, we classified the security characteristics based on the smartphone's component they pertain to: application security, which relates to the analysis of software installed on the smartphone; device features, which are based on smartphone system characteristics, such as current Android OS version and the version of the installed security patches and updates; sensor security, which are represented by the characteristics reflecting the device's features, for instance, the current status of screen lock and developer's menu; and cloud security characteristics, which include the tracking how the measured smartphone's security characteristics changed since the previous measurements, and the comparison of the particular smartphone with other smartphones of similar brand and model. Figure 3.2 also demonstrates the example of how the measurements' aggregation obtained from distinct data sources with various accuracy and security characteristics affects the "Overall DQ Score".

Table 3.1: Data sources' accuracy and their related platforms' security evaluation on the example of Android OS smartphones

Data source and platform characteristics		Platform A	Platform B	Platform C	Platform D
App security	blacklisted apps	0%	20%	0%	0%
	potentiallyDangerous	0%	10%	0%	0%
	unknown sources	0%	50%	0%	20%
	app permission	1%	60%	1%	1%
	Overall	**10.00**	**4.78**	**10.00**	**6.80**
Device feature	OS version	26 [API 26]	24 [API 24]	26 [API 26]	26 [API 26]
	security patches	2 [1-Jun-18]	8 [1-Dec-17]	2 [1-Jun-18]	2 [1-Jun-18]
	device model	5.00	9.00	5.00	5.00
	Overall	**10.00**	**5.00**	**10.00**	**5.43**
Sensor security	bootLoader	locked	unlocked	locked	unlocked
	rootAccess	disabled	enabled	disabled	disabled
	developer's menu	disabled	enabled	disabled	enabled
	device lock	locked	unlocked	locked	locked
	Overall	**10.00**	**0.00**	**10.00**	**0.00**
Cloud security	historic trend	1 [increasing]	(-)0.5 [decreasing]	1 [increasing]	0.1 [increasing]
	same device comparison	0.95 [top 5%]	0.2 [bottom 20%]	0.95 [top 5%]	0.75[top 25%]
	Overall	**9.81**	**2.74**	**9.81**	**5.16**
Device security		**10.00**	**3.49**	**10.00**	**5.44**
Sensor accuracy	accelerometer	90%	5%	40%	70%
	gyroscope	90%	10%	10%	40%
	proximity sensor	90%	12%	60%	50%
Total Sensor Accuracy		**90%**	**9.47%**	**42.03%**	**61.64%**
Overall DQ Score		**9.76**	**0.39**	**5.36**	**5.21**

3.3.3 Accuracy and Security Evaluation Pipeline

Below we present the steps incorporated into the data accuracy and security evaluation procedure in our practical use case.

1. **Measurement Quality Evaluation.** In this step, the measurements quality (e.g., accuracy, range, etc.) provided by all data sources (e.g., accelerometer, gyroscope, etc.) embedded into a data source's platform is evaluated. To facilitate this evaluation in our practical use case, we implement this evaluation in an Android application that we describe in detail in sec. 3.6.2. We employ this application to collect the measurements provided by the data source alongside with their technical characteristics, and employ them to calculate the measurements quality score. Further, we collect all the measurements quality scores calculated for various employed smartphones and mobile devices into a comprehensive knowledge base, described in sec. 3.6.4.

2. **Assessment of Measurements Fusion Accuracy.** In this phase, the accuracy of the measurements fusion process is evaluated after aggregating the data from various data sources embedded into a single platform. The methodology employed for measurements fusion can vary depending on the specific data source platform implementation and its intended application. This evaluation is focused on assessing the degree to which the data fusion technique enhances the overall accuracy of the measurements. The process involves comparing the fused data against the individual data sources' measurements and the DQ specification in order to gauge the level of accuracy improvement achieved through fusion. To accommodate the effective data fusion, it is essential to consider the unique characteristics and complexities associated with each data sources' platform and its applications. The calculus utilized for the measurements fusion needs to be adaptable and well-suited to address the distinct requirements and constraints of each platform realization. By evaluating the accuracy of the measurements fusion, we gain valuable insights into the data integration effectiveness, thus selecting those data sources that better contribute to satisfying the specified DQ requirements.

3. **Data Sources' Platform Security Evaluation.** This stage involves evaluating inherent security characteristics pertaining to the data sources' platforms. As in the previous step, the approach for security evaluation may vary upon the data source's platform implementation and the end application. The platform's security evaluation is primarily focused on deriving and assessing the appropriate security characteristics within each unique data sources' platform. By analyzing the security characteristics, we identify the potential weaknesses in

Table 3.2: Example of how the fusion of data from multiple platforms may affect accuracy, security, and the overall DQ score

	Platform A+B	**Platform A+C**	**Platform B+C**
Security	3.49	10	3.49
Accuracy	95%	95%	80%
Overall DQ Score	6	9.8	5.45

the data source's platform. Then, we employ our security evaluation calculus to reflect the severity of these weaknesses for the particular end application. To facilitate the data sources' platform security evaluation in our practical case, we implement the security evaluation methods and tools in a specified Android OS application, which we describe in detail in section 3.6.1. This application employs our developed systematic methodology to examine each data source's platform, providing a comprehensive assessment of its security characteristics.

4. **Evaluation of Multi-Platform Fusion Security.** This step encompasses the fusion of data obtained from multiple data sources' platforms, followed by an assessment of the resulting multi-platform security score. During the multi-platform fusion process, data collected from diverse sources is integrated to improve the overall DQ. As the data from multiple platforms is fused, the security evaluation calculus considers the individual security characteristics of each platform involved in the fusion. Further, these individual security characteristics are aggregated into the resulting multi-platform security score.

5. **Optimal Sensor Selection.** This stage employs the process of selecting the most suitable combination of data sources that satisfies the established DQ requirements, determined by the ML end application specification. The data sources selection process may involve various techniques, tailored to the specific needs of the end application. In our practical example, we implement our GA-based data sources selection technique in an Android application, presented in detail in sec. 3.6.3. The application is capable of selecting a collection of data sources that provides DQ according to the specified requirements. The process of data sources selection is instrumental in optimizing the data fusion process, as it contributes to satisfying the ML end specification requirements by improving the overall DQ.

To integrate both the accuracy and security of data sources' platforms into the overall DQ score, we implemented a fuzzy rule-based expert system. For the detailed description of the employed fuzzy

rules, please, refer to [107]. This expert system combines security and accuracy metrics, thereby providing a comprehensive assessment of the data collected from various sources and platforms. In our illustrative example, we assume that the measurements fusion results in an improved accuracy when compared to data acquired from distinct sensors. The fuzzy rule-based expert system plays a key role in quantifying the data source's platform security score. Alongside the security characteristics, the expert system integrates the accuracy scores obtained from individual data sources. This comprehensive approach results in calculating the overall DQ score that accounts how the fused data satisfies the established requirements.

3.3.4 Security Role in Measurements Fusion: Practical Example

In the process of data fusion design, the security and privacy aspects of the resulting data should be treated seriously and evaluated from various perspectives. For example, some data sources may possess private or confidential data. The integration of data with various clearance levels can jeopardize its confidentiality level, posing a significant risk to the overall system's security. Let us consider the example of integrating measurements from two data sources' platforms with various security characteristics. As can be seen from the findings presented in Table 3.2 and depicted in Figure 3.2, the fusion of measurements obtained from "Platform A" with "Platform B" results in a higher accuracy. However, on the other hand, such aggregation leads to a lower security score. In this case, the security characteristics pertain to "Platform B" contribute to the diminished overall DQ score when integrating A and B data sources' platforms. This observation reinforces the vital role of platform security in influencing the overall DQ obtained after the data fusion.

In some instances, conflicts may arise when employing various access control policies within a single data source's platform. In case of employing Bell-LaPadula model for access control management [25], the integration of data of various confidentiality levels is prohibited as it violates the security principles established by the model. Such security conflicts emphasize the requirements for employing robust and adaptable security measures within the data fusion process. Resolving these conflicts requires careful consideration of the security policies and mechanisms employed by each data source's platform.

To optimize data fusion design, it becomes vital to maintain a trade-off between accuracy improvement through the data integration, and preserving the employed security and privacy measures' effectiveness. In case when the measurements are integrated from the data sources' platforms possessing various clearance levels, the clearance levels of all integrated platforms are decreased to the lowest one. This may lead to accessing the integrated data by the low-privileged entities, who

originally did not have such an access.

The integration of data obtained from a data sources possessing similar clearance level does not lead to the overall level of clearance reduction. While this approach may seem advantageous for preserving data security and privacy, it still may affect the accuracy and integrity of the integrated data. Moreover, such a strategy might not align with integrity-enforcing access control models, such as the Biba access control model [149]. The Biba model is particularly sensitive to data fusion scenarios involving multiple platforms. Employing this model in such cases can lead to inappropriate DQ calculation due to the fusion of data with varying veracity levels. Integrating appropriate security measures into the data fusion process is crucial to ensure the preservation of data security and integrity.

In our example from Table 3.1, the fusion of measurements from "Platform A" and "Platform B" indeed enhances the overall data accuracy. However, it is essential to note that this fusion also leads to a reduction in the overall data security level, primarily influenced by the "Platform B" security characteristics. We can observe similar effect when integrating measurements from "Platform B" and "Platform C". In both cases, data fusion results in a data accuracy improvement but a decline in the data security level, which is attributed to data sources' platforms with lower security scores participating in the aggregation. On the other hand, the fusion of data obtained from "Platform A" and "Platform C" appears to be highly beneficial for the overall DQ score. The reason behind this is that both "Platform A" and "Platform C" possess high security scores, which eventually enhances the overall DQ score.

The demonstrated practical use case emphasize the significance of considering data sources' platform security alongside the measurements' accuracy and other intrinsic characteristics. By carefully evaluating the security of each platform involved into the fusion process, we can provide more informed overall DQ calculation, which allows to consider ML application requirements towards both data accuracy and security. The selection of data sources' platforms that satisfies not only the accuracy but also the security requirements ensures the preservation of data integrity and confidentiality, ultimately contributing to a higher overall DQ score.

To achieve the required overall DQ level through multi-platform data fusion, we conduct a comprehensive analysis of the data sources' platforms security characteristics and data accuracy provided by the data sources embedded in these platforms. We employ these security and accuracy characteristics and integrate them into our overall DQ calculus, which we describe in sec. 3.4.2. By effectively selecting the combination of data sources that provides DQ satisfying the established requirements, our approach allows to supply the ML end application with data that contributes to

better performance, demonstrated over this data. Below we describe the example of our intelligent data sources selection in practice.

3.4 Intelligent Data Sources Selection Use Case

GA belong to a category of evolutionary algorithms that have found extensive application in optimizing search and selection processes across diverse domains. Notably, GA have been successfully employed in optimizing various applications, including routing optimization in the IoT [250] and enhancing network QoS [151]. Compared to other search techniques, GA introduce unique characteristics that make them well-suited for handling high-dimensional problems in real-time scenarios. GA allow to find a solution that satisfies the requirements in a restricted time. GA employ the natural selection concept that outperform random search algorithms via using historical data to take the search to the best performing region within the solution space [68]. This exceptional feature played a key role in motivating our choice of implementing GA in our data sources selection use case. Given the multi-dimensional nature of the data sources selection problem and the real-time demands of the ML end application we consider, we found GA to be an ideal fit for this task.

In our research [103,182], we employed a GA fitness function that utilized the integral DQ indicator that integrates data accuracy and data source's platform security metrics. However, those studies faced limitations concerning the population size and data sources' diversity. In this chapter, we aim to overcome these restrictions by significantly expanding the range of metrics used in the DQ calculation and diversifying our sensor device population. To achieve this, we first gather extensive knowledge on the characteristics of multi-modal data sources embedded in various platforms, enabling us to create a diverse population for our empirical study. Additionally, we dismiss the constraint on the number of data sources considered in each GA generation, which allows more profound search for the optimal sensor combination.

To handle data of various modalities effectively, we extend the DQ calculus previously introduced in [103]. In this work, we extend our DQ calculus by introducing multiple hierarchical levels that encompass manifold metrics calculated for various data sources and their platforms. Following the concept presented in sec. 3.2, these metrics are then integrated at a higher level to derive the overall DQ score. Our novel approach addresses the challenge of handling diverse data sources and enables us to select the combinations of data sources that not only offer high accuracy but also meet the security requirements and satisfy the overall DQ specification.

To implement the developed calculus in practice, we incorporated it into the Android application [212], which is extensively described in sec. 3.6.2. Through this application, we evaluate the effectiveness of our GA-based data sources' selection optimization approach in real-world scenarios. We conduct an empirical study, comparing the performance of GA with brute force data sources' selection in terms of elapsed wall-time and the achieved overall DQ score. In section 3.5, we deliberate on the results and analysis, and showcase the efficiency and performance achieved by our GA-based approach in selecting the most suitable data sources. By expanding the DQ metrics, diversifying the data sources population, and dismissing previous quantity limitations, our research contributes to a more comprehensive and effective data sources' selection process, ensuring the delivery of high-quality data that satisfies the ML end application requirements.

3.4.1 Formalization of Data Sources Selection Problem for Multi-Modal Data Fusion

In data sources selection, we evaluate the two major data generation components: data sources themselves and platforms into which these data sources are incorporated. In the practical example demonstrated in this chapter, we integrate data sources embedded into various Android OS-based mobile devices. We use various types of data sources integrated into a single platform employed for data collection. Below we formalize the data sources selection and data fusion problems.

Given:

- a set of N data source platforms, which include the data sources P_i, $i \in \{1, \ldots, N\}$;
- a set of quality indicators PQ_{iq}, $q \in \{1, \ldots, M\}$, where M represents the number of quality indicators defined by each data sources' platform technical characteristics;
- each platform is composed of K data sources S_{ij}, $j \in \{1, \ldots, K\}$;
- each data source's quality indicator can also be defined with PS_{ijr}, $r \in \{1, \ldots, L\}$, where L is a number of quality indicators determined for the particular data source.

Goal: to find such data source combination $\overline{S}$ that will provide the required level of the overall DQ indicator.

Selecting the combination of data sources incorporates: the (1) integration of data obtained from multiple sources; and (2) the integration of their respective platforms to achieve the required DQ

level. The process of combining the collected data is denoted as $D = FZ(data_{ij})$, with $FZ(\cdot)$ representing the fusion operation applied to data derived from the combination of S_{ij} data sources. Following this, a comprehensive integration of diverse DQ metrics is achieved through employing the FZQ operator, which can be fine-tuned to accommodate multi-modal data characteristics effectively. The A operator is then employed to aggregate the metrics into the overall DQ indicator.

Hence, the overall DQ resulting from the fusion of multi-modal and multi-platform data can be expressed as $DQ = A(FZQ(PQ_{iq}))$. Our primary goal is to optimize the DQ by maximizing the $FZQ(PS_{ij})$, or alternatively, to achieve a minimum DQ threshold γ specified by the end user and application.

3.4.2 Data Quality and Security Evaluation Calculus

In this section, we introduce a practical illustration of the DQ evaluation, specifically tailored for the selected data sources discussed in detail in sec. 3.4.3. As data sources and their respective platforms possess diverse characteristics, we designed a flexible DQ calculus that is adapted to accommodate these variations. In our development, we follow up our previous research [103], which primarily focused on data fusion from a single modality sources embedded into a single platform. To improve our DQ calculus, we have significantly expanded it to encompass multi-modal data fusion and multi-platform integration functions. Below, we present a detailed description of our DQ calculus, tailored to suit the unique attributes of the employed data sources and their platforms within our practical example.

Multiple data sources may be embedded into a platform S. These data sources are denoted as $s_{i_j} \in S,\ j \in \{1, \ldots, m\},\ m \geq 1$. These data sources are further categorized into groups $t_i \in T,\ i \in \{1, \ldots, n\},\ t \geq 1$ based on the type of data they store or generate. For the employed set of data sources, below we introduce multiple metrics designed to assess the DQ score.

Data Source's Precision (SP) and Total Data Source's Precision (TSP)

The metric denoted as SP represents precision and is determined by the resolution properties of the data source. The resolution of a data source refers to its ability to detect and measure small changes in the quantity of the object or phenomena it is designed to measure. In the accelerometer example, its resolution is determined based on the minimal acceleration it can measure over the all axes. On the other hand, TSP is a metric computed by aggregating SP values over all selected data source combinations utilized for the data fusion. The SP and TSP metrics can be calculated

according to (3.1) and (3.2) respectively.

$$SP = 1 - \frac{resolution(s_{i_j})}{max(resolution)} \tag{3.1}$$

$$TSP = \sqrt{\frac{\sum_{i=1}^{m} SP^2}{m}} \tag{3.2}$$

Data Source's Latency (SL)

This evaluation metric is derived from the average of the minimum (min) and maximum (max) time delays exhibited by the data source between its measurements or responses when providing data. In the practical example with accelerometer, the latency is determined based on the time difference between the real change in the acceleration and its measured result provided by the accelerometer. Given that data source delays can be influenced by various factors, such as shifts in environmental conditions or electromagnetic interference, we normalize the delay values to a unified scale via dividing them by max for each data source. Consequently, the normalized delay value for each data source falls within the $\{0, 1\}$ range. To ensure that higher delays lead to lower SL values, we introduce the latency metric l, which can be calculated according to (3.3). Subsequently, the smallest l value across all data sources within the combination is employed to define the overall SL, as defined by (3.4).

$$l_{i_j} = 1 - \frac{delay(s_{i_j})}{max(delay(t_i))} \tag{3.3}$$

$$SL = min(l_{i_j}) \tag{3.4}$$

Data Source's Platform Power Consumption (PPC)

This evaluation metric is derived from the power consumed by the data source's platform during its operation in both measuring and idling states. A practical example of PPC might be the value of 5 mA, consumed by the platform during the accelerometer's measurement activity per a time unit. The power consumption characteristics can be obtained from the data source's specifications, publicly available documentation, or determined empirically using the appropriate measurement instruments. In our practical example, we consider that all data sources operate simultaneously during measurements, allowing us to formalize the PPC as shown in (3.5).

$$PPC = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} power(s_{i_j})}{n \times m} \tag{3.5}$$

Table 3.3: System's parameters that are gathered for the initial security evaluation

Metric	Symbol	Values
Screen lock	M_{SL}	1 - Pattern, PIN or password; 0 - otherwise
Android OS version	M_V	2 - The latest version, 1 - previous version; 0 - otherwise
Unknown sources	M_{US}	1 - Unknown sources disabled; 0 - otherwise
Potentially harmful applications	M_{PH}	0 - Installed at least one potentially harmful application; 1 - otherwise
Developer's menu	M_{DO}	1 - Developer option menu disabled; 0 - otherwise
Basic integrity test	M_{BI}	1 - System passed basic integrity test; 0 - otherwise
Android compatibility test	M_{CT}	1 - System passed Android compatibility test; 0 - otherwise

Data Source's Platform Security (PS)

This metric evaluates the security characteristics of the data source's platform. Practical examples of the evaluated security metrics in our Android OS mobile devices use case are the current status of the screen lock (type, complexity, enabled or not) and the result of Android OS basic integrity test. To determine the overall PS score for a specific platform, we calculate the metrics for each data source involved in the data fusion separately, and then take the minimum value across all the data sources, which serves as the initial PS value. The computation of the overall PS metric for a particular platform is defined as shown in (3.6).

$$PS_s = sl(s) + dm(s) + bit(s) + act(s) + \sqrt{(us_{max} - us(s)) \times |us(s)|} + (pha_{max} - pha(s)) \times |pha(s)|, \tag{3.6}$$

where PS_s is the data source platform's s security evaluation; $sl(s)$ is a value based on the platform's screen lock parameter; $dm(s)$ is a value based on the platform's s developer menu parameter; $bit(s)$ is a value based on the platform's s basic integrity test parameter; $act(s)$ is a value based on the platform's s Android OS compatibility test parameter; $us(s)$ corresponds to the platform's s unknown source value; us_{max} is a maximum value over all the evaluated platforms; $|us(s)|$ corresponds to the number of unknown applications installed on the instrumentation platform s; $pha(s)$ is a value for potentially harmful applications installed on the platform s; pha_{max} corresponds to the pha's maximum value; and $|pha(s)|$ is a number of potentially harmful applications installed on the platform S. The overall PS value can be calculated as (3.7).

$$PS = min(PS_s) \tag{3.7}$$

In Table 3.3, we represent the metrics according to which data source's platform security is evaluated in our use case.

Data Source's Overall DQ

In this section, we provide a detailed example of the DQ calculus implementation tailored to the set of diverse mobile devices we employed to facilitate our practical example. However, the presented calculus is highly adaptable and can be customized to meet the end user's specific needs. To accommodate this customization, we introduce the concept of the overall DQ fitness function at the higher DQ integration level, which is based on the adjustable weights incorporation. These weights may be modified and fine-tuned based on the particular needs. An illustrative example of this overall DQ function, incorporating our established metrics, is demonstrated in equation (3.8). By manipulating the weights, our calculus can be adjusted to various data fusion scenarios and provide optimized outcomes based on the specific goals of the end application.

$$DQ = \frac{w_1 TSP + w_2 SL + w_3 \frac{1}{PPC} + w_4 PS}{\sum_{i=1}^{W} w_i}, \tag{3.8}$$

where w_1, w_2, w_3, w_4 represent the weight coefficients, and W is the number of weights incorporated in the DQ calculation. In our practical example in sec. 3.5, we calculate the DQ value with equal weights.

3.4.3 Data Sources and Platforms Characteristics with their DQ Evaluation Knowledge Base

To validate our developed intelligent data sources selection framework, we leverage our extensive data collection on the characteristics of real-world diverse mobile devices and data sources embedded into them. In particular, we focus our investigation on a trio of the most widely utilized data source types: accelerometer, gyroscope, and proximity sensor [142]. We apply our framework to these data sources' types, which are integrated into the platforms within our collection. In general, our knowledge base encompasses details concerning 52 distinct accelerometers' vendors, 20 proximity sensors' vendors, and 14 diverse gyroscope vendors. In Table 3.7, we demonstrate the attributes contained within our knowledge base that pertain to the considered data sources' types. Leveraging our data collection extracted from real-world devices, we emphasize the practical applicability of our approach across a diverse range of various platforms available on the market.

Below we provide some descriptive statistics of the sensors and their features employed in our experiments. Table 3.4 represents descriptive statistics for the accelerometer sensor type based on the data accumulated in our developed knowledge base of sensors, while Tables 3.5 and 3.6 shows

Table 3.4: Descriptive Stats for Accelerometer sensor type

Statistical metric	Sensitivity	Non-linearity	Noise Density
Min	16	0.10	75
Max	17039	2.00	800
Mean	6033.91	0.61	308.08
SD	7434.01	0.39	183.23

Table 3.5: Descriptive Stats for Gyroscope sensor type

Statistical metric	Sensitivity	Noise-Density	Cross-axis sensitivity	Non-linearity
Min	33.8	0.0038	1.0	0.10
Max	131.2	0.030	2.0	0.20
Mean	114.55	0.0117	1.66	0.142
SD	24.62	0.008345	0.32025	0.036

descriptive statistics for the gyroscope and proximity sensor types respectively.

3.5 Intelligent Data Sources Selection Framework Evaluation

We verify our intelligent data source selection framework on a real-world use case by gauging its effectiveness in enhancing the overall DQ using real data from our knowledge base. Since the data source selection process have to satisfy the requirements to operate in real time, our evaluation incorporates the measurement of computational performance. For this purpose, we track the wall-time elapsed from the initiation of the data sources' selection procedure to finding the data sources' combination providing the highest DQ value. To facilitate our empirical evaluation, we develop the software application that realizes our GA-based data source selection procedures. This enables us to perform a comparative analysis, in which we evaluate the performance of our approach against the conventional resource-intensive brute-force-based selection. This evaluation not only validates the practical applicability of our framework but also showcases its efficiency and effectiveness in real-world scenarios. We describe the details of our realized GA-based solution below.

Table 3.6: Descriptive Stats for Proximity sensor type

Statistical metric	Resolution	Range	Absolute Response
Min	8.00	50.00	100.00
Max	20.00	100.00	165.00
Mean	12.91	93.75	131.42
SD	3.43	11.023	17.54

GA is a type of optimization algorithm that finds either minimum or maximum value for a fitness function. The value of the fitness function is known a fitness value. As defined previously, we can formulate the goal for our GA as $FZQ(DQ) \rightarrow max$, where the DQ for an individual platform is defined by equation (3.8), and FZQ corresponds to the DQ integration operator. Below, we provide definitions of the classic terminology used in GA in the context of our sensor selection problem.

- **Population** is a group or list of solutions, each of which can solve the problem at hand. This would be represented by a list of all *data source objects*.
- **Chromosome** is a single value in the population, i.e. a single solution to the problem. In our case, it is a combination of data sources represented by a *data source object*, as previously described in sec. 3.4.1.
- **Gene** is a single element in the solution/chromosome, i.e. a single data source.
- **Fitness function** is a measure of the solution's optimality. The fitness evaluation of a particular solution is mathematically presented by equation (3.8).

First, the algorithm encodes information about the data source within a *data source object*. Then it randomly generates a list of *data source objects*. The number of sensors within each *data source object* is defined by the *chromosome length* parameter. The number of *data source objects* in the population is defined by the *population limit* parameter. To **initialize the population**, we need an initial list of possible solutions that can be improved in the next steps. Hence, we randomly generated the initial population with a size equals to *population limit* which is a hyper parameter. This parameter was set as a rounded value of 1/10th of all available platforms with embedded data sources. To create a population, we go through every platform and randomly decide whether to

select it or not. After selecting the platforms, we go through data sources associated with these platforms and again randomly decide whether to select them. In this way we form one *data source object* and continue in this manner till we have as many *data source objects* as the population limit value. Once the initial population is produced, the evolution process starts:

1. The fitness value, as defined by equation (3.8), for each *data source object* in the population is calculated;

2. The population of *data source objects* is then sorted in descending order of their fitness values, such that the data source with the best fitness value appears first;

3. Based on the parameter *retention limit*, a percentage of *data source objects* are selected, and a list of the best data source objects in this population (*rank-based selection*) is generated. To avoid sorting altogether, the *roulette-based selection* is used, wherein a selection probability based on the relative fitness of the data source object is assigned. A data source object with a higher fitness value has a higher chance of being retained;

4. A list of *data source objects* that are ready for mutation and crossover is generated;

5. To ensure that the data sources selection process is not stuck in a local maximum, based on the parameter mutation probability, each of the *data source objects* is altered, wherein one of the randomly chosen data sources within the *data source objects* is replaced by another data source from the list of all available ones. Then, two *data source objects* are randomly chosen for the crossover operation, where a new child *data source object* is generated by combining half of the data sources from each of the selected parent *data source object*. As such, during the crossover operation, the data source's (**genes**), along with their characteristics, that belong to a particular *data source object* are being crossed over.

6. A newly generated *data source object* is added to the new population. This crossover process repeats until the population limit is reached;

7. The average fitness value of the new population is evaluated;

8. Once there is no sufficient change in the average fitness value, the selection process stops and returns the data sources contained within the best population. These data sources are expected to provide the highest DQ if data from them is fused.

3.5.1 Brute Force Algorithm Analysis

To provide a baseline for the evaluation of our proposed GA, we developed a brute force algorithm that exhaustively selects devices with the best DQ score while generating all possible combinations of data sources for each type. Consider a list of available data sources n as well as a number of data source types t for a particular platform. We can represent a selection of data sources for a particular type as a binary string. As an example, consider $n = 3$ and $t = 1$, which means that we have 3 data sources (s_1, s_2, and s_3) of a single type to select from. We can represent each possible selection as a binary string, e.g. 111 for a selection of $[s_1, s_2, s_3]$, or 101 for a selection of $[s_1, s_3]$. For a single platform, the number of possible selections to generate can be evaluated as $2^{n \cdot t}$. Given d platforms, the number of possible platforms to select from can also be represented as a binary string equal to at most 2^d when all platforms are selected. As a result, a brute force algorithm would have to go through all the possible data source selections of each type for each platform, with the number of possible selections equal to $2^{n \cdot t} \cdot 2^d$, which results in the overall time complexity of the algorithm being exponential: $O(2^{n \cdot t + d})$.

3.5.2 Genetic Algorithm Analysis

The time complexity of a generic GA can be defined in terms of the population size N, number of generations G, fitness evaluation time $T_{fitness}$, and the complexity of *selection, crossover,* and *mutation*:

$$O(G \cdot (N \cdot T_{fitness} + N \cdot O(Selection) + N \cdot O(Crossover) + N \cdot O(Mutation))) \tag{3.9}$$

In our implementation, the crossover operation takes $O(1)$ time as we are simply recombining the data encoded in the parent *data source objects*. The mutation operation is linear in terms of time complexity, $O(d)$, where d is the number of all available platforms. The evaluation time of a fitness of a particular solution $T_{fitness}$ takes $O(1)$ time as it is a numerical computation. If the *rank-based* selection gets used, equation (3.9) is dominated by $O(Selection)$ and the overall complexity of the algorithm largely depends on the sorting method used. Assuming a sorting algorithm similar in time complexity to merge sort is used during selection, and also assuming the population size $N = d$, where d is the number of platforms, equation (3.9) can be simplified to $O(G \cdot d \log d)$. If the *roulette-based* selection method is used, the selection step takes $O(d)$ and equation (3.9) boils down to $O(G \cdot d)$. In our use case, we run the *roulette-based* algorithm until the desired level of

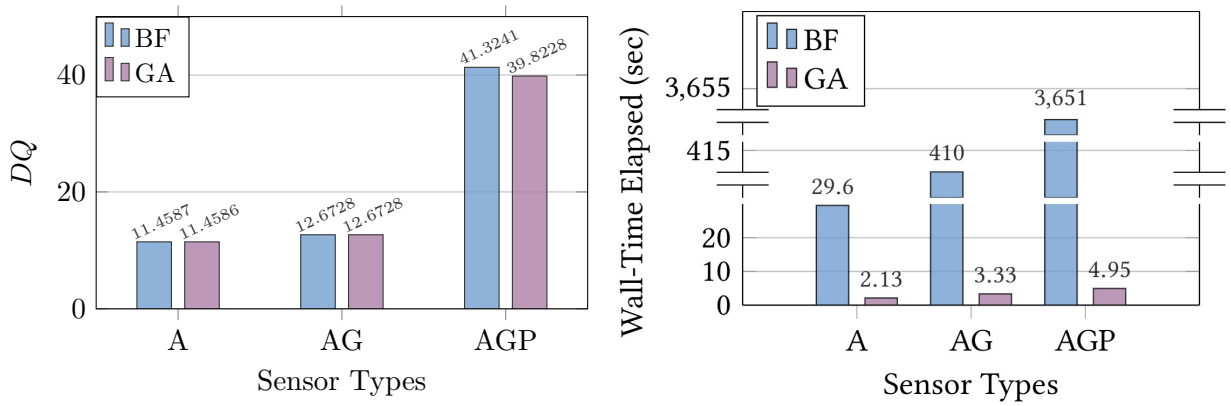


Figure 3.3: Evaluation of GA vs brute force search: (a) – in terms of DQS; (b) – in terms of computational performance; BF refers to brute force; GA to Genetic Algorithms, A means accelerometer, AG means accelerometer and gyroscope (search across two sensor types), and AGP means accelerometer, gyroscope, and proximity sensor (search across three sensor types). In (b) we represent the case with the maximum wall-time taken by the brute force to return the result for each sensor type over all experiments

Table 3.7: Employed data sources and their characteristics

Data Source Type	**Characteristics**
Accelerometer (ACC)	Sensitivity, Non-linearity, Noise Density
Gyroscope (GYR)	Sensitivity, Noise Density, Cross-axis Sensitivity, Non-linearity
Proximity (PRX)	Resolution, Range, Absolute Response

DQ is reached or a user-imposed time limit is exceeded instead of running the algorithm for G generations.

Evaluation Results

In Figure 3.3, we present the comparative analysis of the evaluated data sources' selection techniques. Figure 3.3(a) showcases the overall DQ value achieved by the data sources' combination selected with our GA-based tool, compared against the results obtained with the conventional brute force approach. In this evaluation case, we ensure that both techniques have sufficient time to converge without manually stopping the search. The GA-based tool demonstrates performance

Table 3.8: Results for DQ evaluation for the data sources of multiple types

Data Source Type	*TSP*	*SL*	*PPC*	*PS*	*DQ*
ACC	84.918	0.9831	9.304	3	22.2521
ACC, GYR	82.9613	0.9831	52.1095	4	21.9909
ACC, GYR, PRX	85.9504	0.9830	47.9222	4	22.7385

similar to the brute force method with only marginal deviations. However, the brute force technique slightly outperforms the GA-based approach in case of the proximity data source.

Figure 3.3(b) demonstrates the results in terms of computational efficiency demonstrated by both techniques during the selection. The comparison reveals that our GA-based tool significantly reduces the time required to find the best data sources combination while maintaining comparable performance across all data sources' types. In addition, the brute force-based technique struggles to converge within the established time constraint when selecting data sources in the proximity category.

After identifying the data sources combination that provides the best DQ, we proceed to assessing the resulting measurements' fused DQ. Leveraging our developed calculus, we compute the metrics elaborated in sec. 3.4.2 and aggregate them into an integrated DQ indicator using equation (3.8). In Table 3.8, we present the computed metrics and overall DQ values for three different combinations of the considered data sources. This example demonstrates the potential impact of fusing multimodal data from diverse platforms on the overall DQ value. The minimal deviations in the DQ value across the diverse data sources' combinations is a result of the equal weights employed in equation (3.8) for our empirical example. However, to enhance sensitivity, the DQ calculation can be adjusted by fine-tuning the coefficients within equation (3.8) according to the individual's needs.

3.6 Prototypes Implementation

In order to enhance the practical applicability, we integrated our approach into multiple Android OS applications, employed in our research, and publicly accessible through the Google Play Store. Moreover, we make available our manually collected comprehensive knowledge base on the char-

acteristics of various data sources embedded into diverse mobile devices for public use. These resources can be employed by the broader community for expanded research and development purposes. Table 3.9 showcases both our contributions, presented in this chapter, and relates them to the actual practical applications, incorporating these contributions. Moreover, we also demonstrate which integration aspects are covered by each of our contribution and their practical realizations. These aspects show how our developed techniques and tools facilitate the data sources selection process and demonstrate their real-world implementation.

3.6.1 Data Source Platform Security Evaluation Tool

This application [213] is capable of comprehensively assessing the security aspects of a data source's platform (in this application – Android OS smartphone). This evaluation employs calculating the security score, which is based on the integration of multiple platform parameters. These parameters include, for example, the status of screen lock on the mobile device (if applicable), the Android OS version and the date of the latest patches, the presence of applications sourced from unverified repositories, the potential existence of detrimental software on the device, the status of the developer's mode, the results of both the Android OS "Basic integrity" and "Android compatibility test". Alongside the platform security evaluation metrics, the application also leverages Google SafetyNet library [5], which is a valuable resource encompassing diverse parameters related to the evaluated Android device, such as the state of the bootloader (whether locked or unlocked).

3.6.2 Data Sources Quality Assessment Tool

The application [214] allows to assess diverse of data sources embedded into Android OS-based mobile platforms, such as smartphones, tablets, and various wearable devices. The application allows not only to find the appropriate data sources combination embedded into the device, but also employs our knowledge base [211] to extract comprehensive information regarding the quality and characteristics of each data source. Subsequently, the tool provides user with the data sources' quality categorical analysis by classifying them as either "good" "bad" or "average" data providers. Moreover, the capabilities of the application extend beyond the data sources quality evaluation. It also incorporates educational features by providing its users with the knowledge regarding the

[4] https://play.google.com/store/apps/details?id=com.dataqualitylab.sensorquality&hl=en&gl=US
[5] https://play.google.com/store/apps/details?id=com.igorkh.trustcheck.securitycheck&hl=en&gl=US
[6] https://play.google.com/store/apps/details?id=edu.rit.dataqualitylab.sensorselector&hl=en&gl=US
[7] http://www.dataqualitylabs.com/dataView

Table 3.9: Data sources selection framework integration aspects, and developed methods and tools

Multi-modal and multi-platform data fusion	**Our contributions**	**Developed prototypes and products**
Metrics integration	Data Quality and Security Evaluation calculus	Data Sources Quality Assessment Android OS application[4]; Data Source Platform Security Evaluation Android OS application[5]
Data fusion and multi-platform integration	GA-based instrument selection technique	Data Sources' Selector Android OS application[6]
Knowledge utilization	Autonomous instrument selection tools implementation on the collected database	Collected knowledge base[7]

diverse data sources embedded into their mobile devices. This knowledge includes the list of the available data sources, their technical characteristics, insights into their potential use cases as well as the limitations associated with each data source.

3.6.3 Data Sources Selector Tool

This application [212] allows the selection of a diverse data sources embedded into various mobile Android OS-based platforms according to the established DQ specifications. The application leverages our developed GA-based data source selection techniques presented in sec. 3.4.1. This enables the tool to dynamically select the data source combination, most suitable for the further data fusion. The application employs our calculus, developed and presented in sec. 3.4.2, to aggregate data accuracy, platform security, and other metrics to calculate the overall DQ value. The application benefits users with the ability to manually pick or discard specific metrics from the selection process. Moreover, the users are able to establish the required values for certain metrics, to which the selected data sources should correspond. The employed GA-based selection technique allows to optimize the data sources selection process and provides users with the results in real-time.

To address users' security and privacy concerns about providing the access to their personal mobile devices, the publicly available version of the application employs only pre-uploaded knowledge base of data sources and their platform characteristics, collected in [104, 108], and made available through [211].

3.6.4 Knowledge Base on Data Sources and Platforms Quality Characteristics

In the prior works [104,108], an extensive data regarding the attributes of numerous mobile devices that incorporate various data sources was gathered. We employed and extended this knowledge to calculate the DQ provided by these data sources. Our database encompasses such diverse attributes as measurement types, dimensions, resolution, camera specifications, and their intrinsic hardware characteristics. Currently, our knowledge base incorporates 9443 data sources-embedded platforms, which span from smartphones operating under Android OS and iOS, to other mobile devices, such as tablets, smart watches, and beyond. This repository incorporates data on 58 diverse characteristics pertaining to data sources and their platforms, manufactured by over 114 brands. The roster of data sources types presented within our knowledge base incorporates 19 distinct types [211], including but not limited to barometers, pedometers, gyroscopes, accelerometers, and other data sources actively used in contemporary devices. Below we outline the structure and key components comprising our knowledge base.

Generic Information on Data Sources-incorporating Devices

This segment represents various technical attributes pertaining to data sources' platforms. The platforms' manufacturers are presented by such well-known vendors, as Samsung, OnePlus, Xiaomi, Motorola, and others. Some examples of the platform aspects represented in our knowledge base are device form factors, device dimensions, and camera characteristics.

Data Sources Characteristics

Here, the information regarding various data sources available within the selected platforms and their association with their host platforms is presented. Our collection is represented by a diverse array of data sources' attributes, including but not limited to latency, resolution, range, power consumption, and a large spectrum of others.

Data Source Platform Security

In this segment, we concentrate on the security attributes pertaining to the selected data sources' platforms. Here, some examples of the presented attributes are screen lock activation status, the results of Android OS "Basic integrity test", the number of potentially harmful applications installed on the device, and the presence of applications installed from unverified sources.

Our comprehensive knowledge base empowers not only our developed applications but also benefits regular mobile device consumers to employ this knowledge with their intent. Additionally, our open-access repository contributes to the broader research and development community as a valuable resource for further exploration and advancement in the area.

3.7 Conclusion

In this chapter, we developed our innovative approach to intelligent data sources selection, which is one of the critical components in the MLIN restructuring. The **major novelty** of our approach is **the integration of DQ with platform security characteristics of the platform into which this data source is embedded**. This enabled us to switch the focus of the conventional DQ evaluation from merely quality characteristics to the security incorporated ones, which, as our investigation showed, affect the DQ critically in practical application. Alongside this novelty, we **addressed the challenges posed by data multi-modality and its diverse origin**. We **developed our generic DQ calculus** that **allowed to evaluate DQ of various modalities fused from multiple diverse platforms**. To facilitate the intelligent data sources selection in real-time, we **employed GA-based data sources selection technique** and **incorporated it with our developed DQ calculus**, which enabled to utilize **DQ as the major optimization parameter**. We integrated all the developed solutions into an **Integration Framework for Data Sources Selection**, which is aimed to find the best data sources combination that satisfies end user and application requirements. We **verified our developed Framework in a practical use case incorporating real-world diverse mobile devices**, served as platforms, with the embedded sensors, served as data sources in our empirical study. Below we list our contributions developed in each chapter's section.

- In sec. 3.1, **we formulated the major challenges in practical DQ evaluation** and **elaborated our motivation behind the selected practical use case**. We discussed the complexities faced when dealing with data sources of varying quality, data types, and

technical characteristics, especially when utilizing mobile devices like smartphones as data sources.

- In sec. 3.2, we **developed the architecture of our Integration Framework for Data Sources Selection**, aimed at finding the data sources combination that satisfies the established requirements. The architecture includes a main integration level, responsible for aggregating multi-modal data from diverse devices, and several auxiliary sub-levels, focused on integrating DQ with platform security characteristics and leveraging available knowledge and requirements.

- In sec. 3.3.2, we **introduced and described our practical use case, illustrating how DQ evaluation can be realized in real-world applications**, particularly in the context of medical research with the employment of ML end system. We **emphasized the challenges of data variability and security in further data fusion**, when a variety of personal mobile devices are employed as data sources.

- In sec. 3.4.1 and 3.4.2, we **developed our formalization for the data sources selection task**. We **developed our innovative calculus that integrates DQ metrics with platform security characteristics**. The calculus incorporates multiple hierarchical layers that enable aggregating diverse metrics into a single DQ value, acting as a major quality indicator for our data sources selection. We also **demonstrated practical example of how various metrics can be calculated within the context of mobile devices**.

- In sec. 3.3 and 3.5, we **verified our approach in practice**. We **developed our GA-based sensor selection tool**, into which we incorporated our DQ calculus. We **evaluated the performance of our developed tool on a real set of mobile devices** in terms both effectiveness and efficiency. We compared the obtained performance results with the conventional brute force-based selection method. The results demonstrated that **our approach substantially outperformed the conventional search method in terms of computational efficiency while providing similar DQ**.

- In 3.6, we **elaborated on the practical tools we developed, which are implemented in Android OS applications, to further facilitate the use of our tools in practice**. We described in detail multiple Android OS applications that incorporate our DQ evaluation methods and realize our GA-based data sources selection. Moreover, **we provided additional details on the knowledge base on characteristics of thousands real data sources and their platforms, also making it public** to further facilitate and benefit other researchers and developers in the area.

This chapter introduces solutions that facilitate intelligent data sources selection in MLIN, addressing the challenges of data multi-modality and platform diversity while integrating DQ and platform security. The developed solutions extend the outlined contributions beyond theory to practical applications, empowering MLIN users to make informed decisions about their data sources and promoting optimized and secure data integration within the MLIN architecture.

Chapter 4

Cyberinfrastructure Integrated with Machine Learning Applications

In this chapter, we perform a comprehensive investigation of multiple real-world use cases that demonstrate how MLIN cyberinfrastructure affects the input DQ, and ML application performance and robustness. In particular, we study how the network facilities component and other MLIN cyberinfrastructure problems impact the quality of the transmitted data. In our research, we employ data of various types and modalities, utilized in diverse ML applications. We cover the domains of ML image and sound classification, ML voice recognition and transcription, and ML video object detection and classification that are highly challenging for the contemporary ML applications, especially when they have to operate in real time. In our examination, we employ POWDER platform [23] that allow us establishing real-world wireless network and employing it for obtaining data affected by real network disruptions, which enhances the practical value of our work. Using POWDER, we investigate how the changing network conditions affect the performance of various ML applications and how robust they are to DQ variations. We leverage the obtained knowledge on the interrelationships between the input DQ and ML application performance to develop the practical example of MLIN adjustment feedback component and demonstrate how it can be integrated into the MLIN structure. We then move our developed ML robustness generic calculus into practice and show how it can be applied in the considered practical example. We divide this chapter into multiple sections, whose content we briefly describe below:

- In sec. 4.1, we discuss how MLIN cyberinfrastructure components can be realized in practice. We elaborate on how they impact the quality of the data they operate with, and concentrate

our discussion on the network facilities practical example.

- In sec. 4.2.1, we describe POWDER platform's general characteristics and disclose our motivation on employing this platform in our research. We describe how the platform is employed for our research by disclosing the details on the established wireless network characteristics and topology. In particular, we concentrate on varying two network characteristics to recreate the network QoS variations: packet loss and size of the receiving buffer socket.

- In sec. 4.2.2, we elaborate on the traffic signs classification real-world use case. We provide the details on the employed data collection, ML models' architectures, experimental design, and analyze how the images affected by the varying network conditions and other problems, such as failures and errors in the MLIN cyberinfrastructure hardware and software, impact the input DQ and the ML image classification performance.

- In 4.2.3, we elaborate on another image classification task, however employed in another domain. In particular, we concentrate on the medical images classification real-world use case. We provide details on the employed medical X-ray images dataset, ML models' architectures, empirical study design, and analyze how the images affected by the varying network conditions impact the ML medical image classifiers' performance.

- In sec. 4.2.4, we elaborate on the sound classification real-world use case. In this case, we follow the renowned approach of utilizing ML image classifiers for processing spectrogram images extracted from the sound recordings. We provide details on the data extraction and preparation, ML model architectures and their re-training, experimental design, and analyze how the sound recordings affected by varying network conditions impact the ML sound classification performance.

- In sec. 4.2.5, we describe the voice recognition and transcription real-world use case. In this case, we employ real voice recordings and transmit them over a network with the varying network conditions to be recognized and transcribed by the state-of-the-art ML application. We provide details on the data preparation, including obtaining the corrupted input data, processing of this data by the ML application, and analyzing the obtained results on the revealed interrelationships.

- In sec. 4.2.6, we concentrate on the video object detection and classification real-world use case. Here, we employ real-world car dash cam video recordings, transmit them over a network with varying network conditions, and process them with the commercial Rekognition video analysis service provided by Amazon. In comparison to the previous use cases, in this one we are not able to pre-train the employed ML model and even do not have access to the ML

model architecture. We analyze the performance of the employed ML application and discuss the obtained results on the revealed interrelationships.

- In sec. 4.3, we leverage the revealed knowledge on the interrelationships between the input DQ and ML application performance and utilize them to design the practical example of MLIN adjustment feedback component, aimed at providing recommendations to adjust network facilities' parameters. We show how it can be realized in practice, how it can be integrated into the MLIN architecture, and develop examples of rule-based feedback employed to ensure the ML application performance and robustness to DQ variations.
- In sec. 4.4, we move our ML robustness calculus into practice and demonstrate how the ML robustness can be measured based on the data obtained in one of our real-world use cases. We formalize two types of the ML robustness indicators and showcase how they can be employed for the particular use case. We discuss how ML robustness is interrelated with the input DQ and ML application performance, and how this interrelationship can be employed to ensure the overall ML robustness.

4.1 Cyberinfrastructure in MLIN Systems and its Influence on DQ

We define cyberinfrastructure in the context of MLIN as the interconnected technological framework that supports the operation, management, and coordination of the various MLIN interconnected components, mentioned in sec. 2.3. It provides the necessary computational, data storage, networking, hardware and software resources to enable the efficient and effective MLIN operation. Cyberinfrastructure plays a critical role, as it should be reliable enough to facilitate data exchange, processing, analysis, and decision-making across distributed and interconnected devices, networks, and computational resources. Depending on the particular practical application and requirements, cyberinfrastructure might include various components, some examples of which are described below.

- Data storage and management systems: these components are responsible for managing the storage and retrieval of data from various sources within the MLIN system. Practical examples include databases, data warehouses, and distributed file systems. In MLIN, data may be produced or stored by various data sources and is conveyed from them to the ML application for further processing and use.

- Computational resources: these components include the combination of hardware and software facilities that provide the computational power required for data processing, processing ML algorithms, data gathering and fusion, real-time analysis, and other processes related to MLIN operation. Some practical examples of these components include servers, clusters, GPUs, and cloud computing resources.

- Network infrastructure: these network components facilitate the communication between devices, data sources, and processing units within the MLIN system. Here, routers, switches, access points, and network protocols ensure data transfer while maintaining the required network parameters.

- Security and privacy mechanisms: as we demonstrated in chapter 2, measuring and ensuring the security of data plays an important role in maintaining high DQ. Here, components such as firewalls, encryption mechanisms, identity and access management systems, and intrusion detection systems might be employed in practice. These measures allow enhancing the security and confidentiality of sensitive data collected from various data sources embedded into diverse platforms.

- Data fusion and integration tools: these components handle the integration of data from diverse sources, performing data preprocessing and fusion. In chapter 2, we demonstrated an example of such tool practical realization based on the GA-based sensor selection. Such tools ensure that data from different sensors, devices, and platforms is combined and transformed into a required format for further use.

- Monitoring and management tools: these components are responsible for real-time monitoring, management, and control of the MLIN system's performance, resource utilization, and how it satisfies the established requirements. One of the practical example is our MLIN feedback system, which we described in sec. 2.3, is responsible for monitoring the ML application robustness and providing recommendations to adjust MLIN structure. Other examples include monitoring dashboards, logging systems, resource allocation tools, and automated management scripts.

In this chapter, we concentrate on the network facilities, as a vital part of MLIN cyberinfrastructure, and its interrelationship with the DQ of transmitted data and with ML application robustness. In the practical MLIN implementation, the network facilities infrastructure is a crucial element that enables conveying of the produced data from its origin to the ML application. At the same time, as we mentioned in sec. 2.4.2, the configuration and operational characteristics of network facilities

indeed influence the quality of transferred data. These network configurations and characteristics depend on various network devices, technologies, and topology employed to facilitate data exchange among the interconnected MLIN components.

In the case of network transmission, DQ degradation refers to the deterioration of various DQ attributes due to multiple factors that can occur during the network communication. We provide some practical examples of these factors below.

- Network congestion: this factor refers to the scenario when the network is overloaded with more traffic than it can process normally, and results in delays, packet loss, and higher packets retransmission rate. Network congestion can affect in both wired and wireless networks, especially when there are multiple users or devices competing for the same bandwidth or channel. This is an actual problem for IoT and sensor networks, which leads to reduced throughput and higher devices' battery consumption due to the need to frequently retransmit the packets not reached the target destination. Increased energy consumption lead to discharged devices, which diminishes the overall system efficiency and reliability, especially in some low-power wireless networks (e.g., LPWAN or LPWA).

- Network errors: the errors may occur during the data transmission or reception due to some physical or logical failures in the network. Network errors result in data corruption, distortion, or loss, and depend on the quality and condition of the network facilities' components and connections. For instance, if MLIN network facilities employ a fiber optic cable to transmit data, it may experience DQ degradation due to fiber breaks, bends, or splices that affect the signal quality [161].

- Network attacks: refer to malicious attempts by adversaries to compromise, disrupt, intercept, or damage the network or the transmitted data. Network attacks may employ various methods and techniques to exploit vulnerabilities or weaknesses in the network security. For example, if MLIN network facilities employ a TCP/IP protocol to transmit data, it may experience DQ degradation due to data losses caused by Denial-of-Service (DoS), man-in-the-middle (MITM), or IP spoofing attacks that aim to overload, intercept, or alter the data packets.

- Networks interference: in wireless networks, the access point signals might be interfering with each other due to various factors, such as network misconfiguration or initial design issues. For example, the normal overlap between the access points is 15% to 20% coverage in case of real-time voice data transmission, and 10% in case of regular data[1]. However, if the coverage

[1] `https://learningnetwork.cisco.com/s/question/0D53i00000KsqAmCAJ/10-15-cell-overlap`

overlap becomes too large, the access points may compete for the same channel and cause the unwanted interference. Otherwise, if the overlap is too small, the access points may employ distinct channels and cause adjacent-channel interference. These both types of interference can degrade the DQ and performance of the network. In addition, network interference can be caused by other devices that share similar transmission bandwidth.

- Environmental factors: MLIN cyberinfrastructure operates in physical environment, which can also affect the wireless signal propagation and reception. For instance, some materials can block, reflect, or absorb the wireless signals, resulting in signal attenuation or multipath fading [210]. Some examples of materials that can cause interference are metal, concrete, brick, marble, and glass[2].

As one can see, there are numerous challenges causing DQ deterioration while the network transmission. Some practical examples of DQ degradation aspects include:

- Data loss: happens when some data packets are lost, dropped, or discarded during the transmission over the network due to congestion, errors, or attacks. Data loss can result in incomplete or missing information that affects the MLIN system's and leads to ML application performance and robustness decrease. For example, if MLIN network facilities employ UDP network protocol for the data transmission, some packets may be lost due to network congestion or a DoS malicious attack.

- Data modification: happens when some data packets are altered or corrupted during the network transmission due to errors or malicious attacks. Data modification can result in incorrect or inconsistent information that also affects the ML application performance. For example, if receiving images from a remote camera, data modification might happen due to file transfer errors, compression artifacts, or malicious tampering.

- Data obsolescence: reflects how up-to-date the transmitted data is with respect to the real-world object or phenomena that it represents (applicable for real-time systems). Data timeliness is important for various ML industrial applications, where the predictive ML models have to provide outputs over a real-time data inputs. If the data is obsolete, it may not reflect the current or relevant state of the object or phenomena, and thus leads to ML application performance degradation. A practical example of factor causing data obsolescence is a network latency that prevents the ML application from receiving the data in a timely manner.

[2] `https://www.signalboosters.com/blog/materials-that-block-wifi-signals/`

One of the ways to prevent or mitigate DQ drop in MLIN is to ensure the required network QoS established for the ML application. QoS is a set of technologies that works on a network to control traffic and ensure the performance of critical applications with the limited network capacity. QoS-aware systems enable means to adjust their overall network traffic by prioritizing specific high-performance applications [127]. QoS is typically applied to networks that operate with traffic for resource-intensive and real-time systems such as multimedia streaming content, online games, video conferences, and voice transmission (e.g., VoIP). The employment of QoS in networking allows to establish and maintain network requirements demanded by the end application. It allows to monitor and adjust a multitude of network parameters, such as delay, jitter, and packet loss rate during the data transmission.

The conventional network adjustment approach commonly relies on various QoS metrics, such as bandwidth, latency, packet loss, jitter, throughput, availability, and others. For instance, the traditional network adjustment approach can manage the available bandwidth to allocate the network resources and balance the network traffic. It can also employ the latency metric to minimize the network delay and enhance the network responsiveness. In our MLIN integral approach, we are focused on ensuring the requirements towards application performance and robustness by finding MLIN structure that satisfies these requirements (see sec. 2.6). In this chapter, we concentrate on describing the practical example of adjusting the MLIN network parameters to assure ML application performance to the required level in order to improve its robustness to DQ variations. In contrast to the conventional network adjustment approach that considers various QoS metrics as primary indicators, we switch our focus to ML application performance as a major indicator. We make this possible by leveraging our integral MLIN architecture we introduced in sec. 2.3, which allows us to employ interrelationships between various MLIN components. In this particular example, we employ the interrelationship between MLIN network facilities and ML application performance, which allows us to determine: (1) how the values of specific network parameters, employed while the data transmission, affect the ML application performance; and (2) which network parameters and structure can be used to satisfy the ML application performance requirements.

We dedicate this chapter to a number of real-world use cases we investigate, which enable us to demonstrate how the dynamic network nature and changing QoS parameters may lead to DQ variation and ML application performance deterioration. To further expand our investigation and make it closer to real ML industrial applications, we leverage real-world network facilities, kindly made available for us by the POWDER network research platform [23], which we will describe in detail in sec. 4.2.1. We employ POWDER to establish a real-world wireless communication channel between multiple network nodes, and we investigate the real aspects of network QoS degradation

while the data transmission. By these means, we move our theoretical MLIN architecture into practice, and establish a segment of MLIN, which incorporates data source, network facilities, and ML application, in real conditions. In our real-world use cases, we investigate multiple ML industrial applications that leverage data of various modalities, sizes, dimensions, and contexts, commonly employed in real-time scenarios. In particular, our use cases incorporate such ML domains, as: traffic signs images detection and classification; medical images classification; sound classification; voice recognition and transcription, and object detection and classification in videos. The considered ML domains encompass a variety of today's ML practical applications, especially those operating in real-time conditions, which emphasizes our research's practical importance and value.

As our major contributions, we (1) demonstrate how our novel MLIN network adjustment based on ML application performance monitoring feedback, introduced in 2.3, may be moved into practice. For this, we first conduct an empirical study on determining which network parameters enable improving the network transmission in case of the network QoS degradation. Specifically, we evaluate the communication characteristics of various network protocols in the QoS degradation scenario. Then, we leverage the obtained results with the investigated interrelationships between the MLIN network parameters and ML application robustness, from our real-world use cases, to formulate examples of network adjustment rules. As another contribution, we (2) present how our novel ML application robustness calculus, presented in sec. 2.6, can be moved into practice by developing two types of indicators, local and global robustness. We showcase how the ML application robustness can be measured in one of our real-world use case, and how the end user can benefit from these measurements in various scenarios.

In the following sections, we provide the detailed description of our real-world use cases we investigate in this study. As the experimental setup related to the communication facilities configuration is similar for all the considered use cases, we first elaborate on this aspect, and then consequently move to the setup for each particular use case.

4.2 Practical Investigation on how MLIN Network Facilities Affect DQ

4.2.1 Communication Facilities Configuration in POWDER

What is POWDER and Why we Employed it in our Research?

The POWDER platform serves as a testing ground to explore the potential of wireless networking within a city-scale and provides access to real-world devices and networks. This initiative aims to experiment with the future possibilities of wireless communication[3]. POWDER stands for Platform for Open Wireless Data-driven Experimental Research, is organized by the National Science Foundation (NSF), and led by the University of Utah in collaboration with Salt Lake City, Utah Education and Telehealth Network, Rice University, and others. The POWDER platform offers the following features and capabilities for network researchers[4]:

- Programmable radio devices, connected to a network that can be configured by the user, and to a wide variety of auxiliary resources and services, such as computational, storage, and cloud platforms.
- A massive multiple-input multiple-output (mMIMO) base-station with 64 radios and a growing complement of open source software stacks.
- There are eight rooftop base-stations, each equipped with a maximum of four versatile software-defined radios (SDRs) linked to either broadband or frequency-specific antennas.
- Seven static endpoints, each comprising a dual set of versatile SDRs linked to broadband antennas.
- POWDER allows to regulate and specify the demanded radio frequency (RF) setting with a multitude of SDRs and off-the-shelf devices.
- An operational framework with a user-friendly graphical interface that facilitates remote accessibility along with advanced experimental workflow control and management tools.
- An incorporated profile mechanism that facilitates users to precisely define the hardware and software settings required for their research.

[3] `https://powderwireless.net/`

[4] `https://www.flux.utah.edu/project/POWDER`

(a) (b)

Figure 4.1: Example of POWDER mobile endpoint, deployed in University of Utah's campus shuttles: (a) – the location of the mobile endpoint inside the campus shuttle; (b) – a closer look to the employed hardware

- An extensive variety of template profiles for various practical use cases, which enables users to spend less time on creating and instantiating POWDER profiles from scratch.

The POWDER platform provides a comprehensive environment for network research, including but not limited to 5G, and any other technologies that can be accommodated on SDRs. Moreover, the POWDER platform facilitates the practical deployment and evaluation of wireless applications and services in a dynamic setting by employing real-world devices and networks. For instance, the platform allows to connect to the campus shuttles equipped with communication transmitters and receivers, operating within the University of Utah campus, which contributes to a realistic experimental ground. Some photos of the mobile endpoint deployed in the campus shuttles can be seen in Figure 4.1 (the images are taken from the POWDER website[5] on August 14, 2023).

POWDER is available for free use by academic researchers, industry collaborators, governmental entities, and other interested individuals or groups. Users can initiate the access to the POWDER portal by submitting an account request, thereby gaining entry to a variety of resources and tools available for network research. Furthermore, POWDER provides access to the documentation, on the portal and network research facilities, including details about the platform's functionalities, potential applications, tutorials, research publications that employed POWDER, and other related information. In Figure 4.2, one can see a screenshot of the interactive map that shows the location of the POWDER base-stations and other network facilities allocated over the University of Utah's campus territory.

[5] `https://www.powderwireless.net/news\#news-may3-2021`

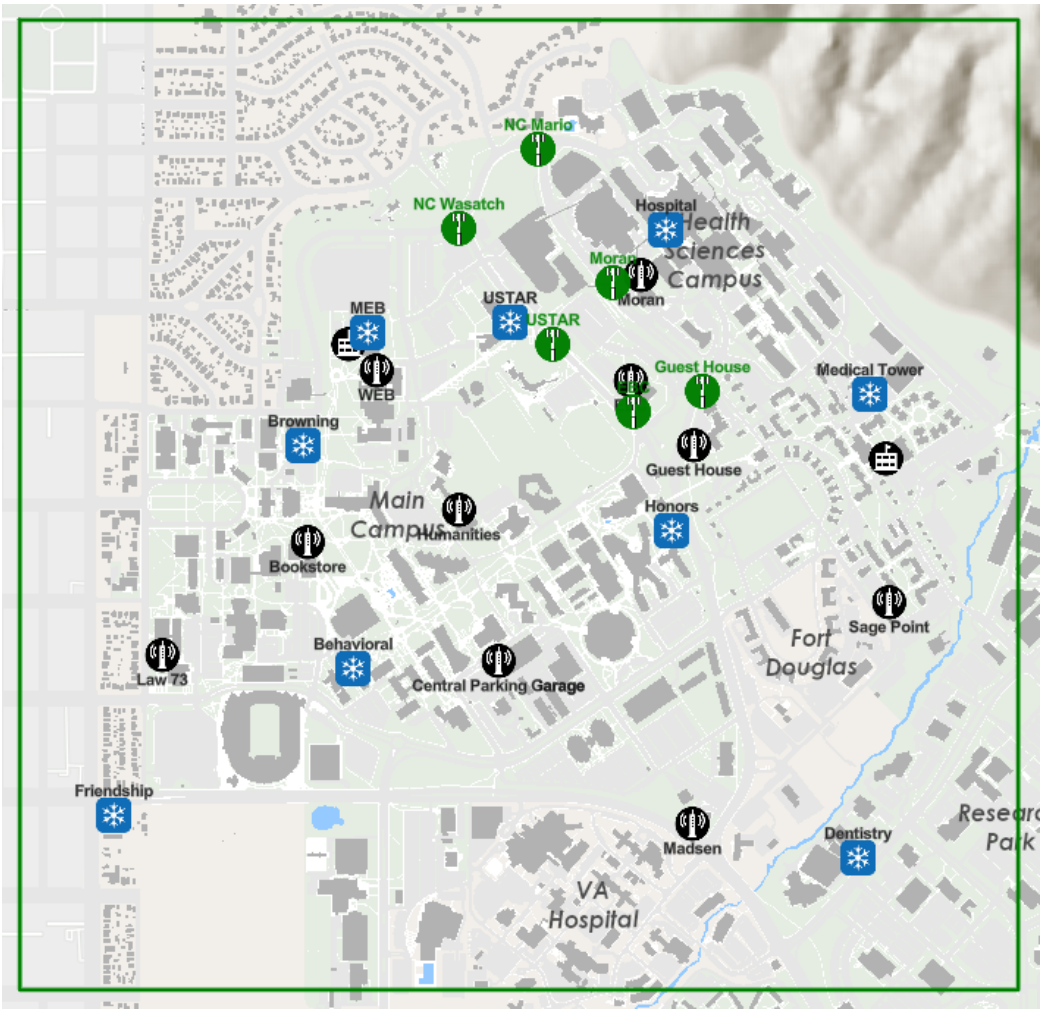


Figure 4.2: The map of the University of Utah campus (Utah, USA), with the location of POWDER base-stations and other network facilities on a real terrain. A screenshot is taken on August 14, 2023, from https://www.powderwireless.net/map

In our real-world use cases, we are primarily focused on investigating the impact of DQ variation due to changing network conditions on the performance and robustness of the considered ML applications. Specifically, in our practical examples, we studied such network QoS parameters, as packet loss and the available buffer resources on the data receiver. By leveraging the POWDER capabilities, we are able to manipulate the network parameters to create real-world dynamic network conditions we are looking for. Hence, our intention to utilize the POWDER platform is emphasized by the following reasons that greatly enhance the value of our research.

1. The POWDER platform provides a fully controlled network environment to navigate each step of our empirical study. This enables us to study how various network parameters and configurations affect the transmitted DQ. Further, we will use this knowledge to design our example of network adjustment rules we describe in sec. 4.3.

2. The fact that POWDER's facilitates operate in the real world enables us to take into account the complexities of real industrial networking scenarios. This feature is particularly advantageous as it bridges the gap between our theoretical investigations and the real challenges that ML applications face when deployed in dynamic network environments.

The employment of the POWDER for our real-world use cases significantly contributes to investigating the interrelationships between the DQ variations due to changing network conditions and their subsequent effects on ML application performance. By leveraging the platform's features, we are able to analyze real-world network challenges, providing us with a foundation to design our MLIN network adjustment feedback system practical example.

POWDER Experimental Setup for our Real-World Use Cases

For each of our real-world use cases, we establish a similar POWDER topology to facilitate wireless data transmission. Depending on our use cases, we employ various ML applications and transmit diverse types of related data. Our study involves transferring various media-files, such as images (traffic signs and medical images), sound recordings, voice recordings, and videos. We transfer them using the UDP protocol between two network nodes over a wireless LTE connection. Since the UDP allows for a packet loss, this transmission may result in the files corruption while processing the transmitted packets at the receiving end. To recreate poor network QoS conditions, we manipulate multiple network parameters, such as packet loss rate and buffer size of the receiving node. In each

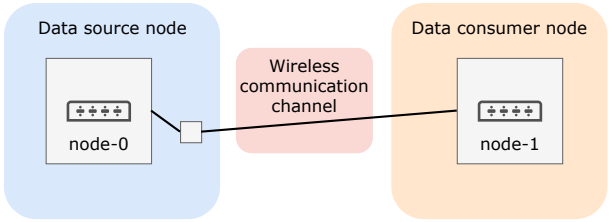


Figure 4.3: Established wireless network topology for the data transmission channel in POWDER

use case, we conduct multiple data transmission series that allow us to obtain data with a varied quality affected by the changing network conditions.

We instantiate a POWDER profile that allows creating an end-to-end LTE network in a controlled RF environment. Basically, we establish two network nodes: data source, on which we initially upload our data for the further transmission; and cloud-based ML application (data consumer node), which receives the data and processes it with the installed ML model. In Figure 4.3, we schematically represent the topology, which is basically a wireless connection between two POWDER nodes. In our use cases, we employ ML models pre-trained on the data related to the particular ML application, we will describe the employed training collection and ML models for each case individually. Cloud-based ML application is responsible for processing the received files one by one and providing the ML performance results. For each use case, depending on the data type, ML model, and ML application, we employ various performance metrics appropriate for the considered application.

Since our investigation encompasses various ML applications, for each of our use cases we first conduct a preliminary study to determine the QoS conditions leading to DQ variation during the data transmission. Diverse applications might have various tolerance to packet loss, for example, packet loss rate below than 1% is required to ensure clear and smooth voice quality in VoIP; video conferencing and streaming applications also commonly require packet loss rate lower than 1% to ensure high video quality and synchronization; and online gaming applications commonly provide the acceptable performance when there are no more than 0.5% packets are lost. However, the network QoS might drop due to multiple factors, described in sec. 4.1, which results in higher packet losses need to be handled by the end applications. There are a number of papers that study how network quality degradation affects the performance of the end data-driven systems [93, 237], however, they usually consider the problem on rather narrow specific applications, which does not allow to generalize the obtained results. In our research, we determine the network QoS conditions that result in DQ variation enough to impact the ML application performance. To maintain the

experiments feasibility, we avoid extreme cases when the transmitted data is too corrupted to be processed by the ML applications or non-readable on the receiving edge. For each use case, we describe the packet loss rates we experiment with, and also specify other manipulated network parameters if applicable.

For configuring the network, we utilize the *geni-lib* toolkit, an effective tool that enables to create *RSpec* specification files using Python. The code Listing 4.1 illustrates how we generate a multisite topology and the connections between the nodes in POWDER. We used this topology to transmit data in each of our real-world use cases.

Listing 4.1: Source code for the established LTE network topology in POWDER

```
#!/usr/bin/env python
import geni.portal as portal
import geni.rspec.pg as rspec
import geni.rspec.igext as IG
import geni.rspec.emulab.pnext as PN
import geni.urn as URN

class GLOBALS(object):
    NUC_HWTYPE = "nuc5300"
    COTS_UE_HWTYPE = "nexus5"
    UBUNTU_1804_IMG = "urn:publicid:IDN+emulab.net+image+emulab-ops
----//UBUNTU18-64-STD"
    SRSLTE_IMG = "urn:publicid:IDN+emulab.net+image+
----PowderProfiles:U18LL-SRSLTE:1"
    COTS_UE_IMG = URN.Image(PN.PNDEFS.PNET_AM, "
----PhantomNet:ANDROID444-STD")
    ADB_IMG = URN.Image(PN.PNDEFS.PNET_AM, "
----PhantomNet:UBUNTU14-64-PNTOOLS")

pc = portal.Context()
pc.defineParameter("ue_type", "UE Type", portal.
    ParameterType.STRING, "nexus5", [("srsue", "
----srsLTE UE (B210)"), ("nexus5", "COTS UE (Nexus
----5)")], longDescription="Type of UE to deploy.")

pc.defineParameter("enb_node",
"eNodeB Node ID", portal.ParameterType.STRING, "",
advanced=True,
longDescription="Specific eNodeB node to bind to.")

pc.defineParameter("ue_node", "UE Node ID", portal.
    ParameterType.STRING, "", advanced=True,
    longDescription="Specific UE node to bind to")

params = pc.bindParameters()
pc.verifyParameters()
request = pc.makeRequestRSpec()

# Add a NUC eNB node
enb1 = request.RawPC("enb1")
enb1.component_id = params.enb_node
enb1.hardware_type = GLOBALS.NUC_HWTYPE
enb1.disk_image = GLOBALS.SRSLTE_IMG
enb1.Desire("rf-controlled", 1)
enb1_rue1_rf = enb1.addInterface("rue1_rf")
enb1.addService(rspec.Execute(shell="bash", command=
"/local/repository/bin/update-config-files.sh"))
enb1.addService(rspec.Execute(shell="bash", command=
"/local/repository/bin/tune-cpu.sh"))
```

```
enb1.addService(rspec.Execute(shell="bash",
command="/local/repository/bin/add-nat-and-ip-forwarding.sh"))

# Add a UE node
if params.ue_type == "nexus5":
    adbnode = request.RawPC("adbnode")
    adbnode.disk_image = GLOBALS.ADB_IMG
    rue1 = request.UE("rue1")
    rue1.hardware_type = GLOBALS.COTS_UE_HWTYPE
    rue1.disk_image = GLOBALS.COTS_UE_IMG
    rue1.adb_target = "adbnode"
elif params.ue_type == "srsue":
    rue1 = request.RawPC("rue1")
    rue1.hardware_type = GLOBALS.NUC_HWTYPE
    rue1.disk_image = GLOBALS.SRSLTE_IMG
    rue1.addService(rspec.Execute(shell="bash",
    command="/local/repository/bin/update-config-files.sh"))
    rue1.addService(rspec.Execute(shell="bash",
    command="/local/repository/tune-cpu.sh"))

rue1.component_id = params.ue_node
rue1.Desire("rf-controlled", 1)
rue1_enb1_rf = rue1.addInterface("enb1_rf")

# Create the RF link between the UE and eNodeB
rflink = request.RFLink("rflink")
rflink.addInterface(enb1_rue1_rf)
rflink.addInterface(rue1_enb1_rf)

tour = IG.Tour()
tour.Description(IG.Tour.MARKDOWN, tourDescription)
tour.Instructions(IG.Tour.MARKDOWN,
tourInstructions)

request.addTour(tour)

pc.printRequestRSpec(request)
```

In Figure 4.4, we represent what roles and responsibilities the components of the established POWDER topology posses in our empirical studies. In the sections below we describe the empirical study setup and results for each individual real-world use case. In Table 4.1, we represent additional high-level details and aspects of the investigated ML application real-world use cases, such as the employed data collection, employed ML models and systems, and the particular investigated DQ variation aspects.

4.2.2 Traffic Signs Images Use Case

Data losses caused by network QoS degradation can result in images classification patterns modification. Modifications in those patterns' pixels representation caused by various distortions and artifacts create additional challenges for proper classification. These challenges require more sophisticated pre-processing or decision making techniques. To study how network disruptions affect ML image classification performance, we employ a subset of traffic sign images taken from the

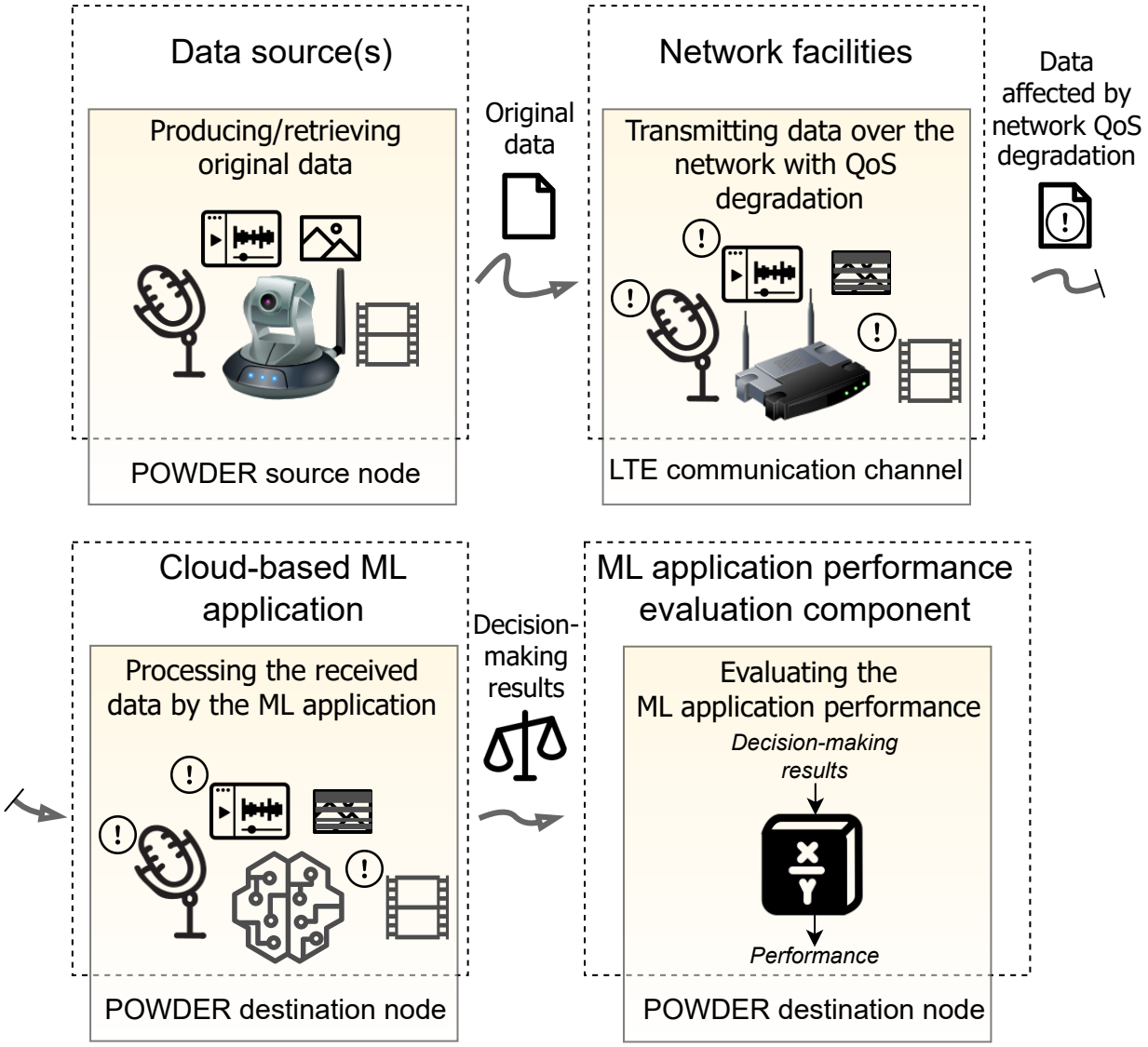


Figure 4.4: Experimental setup for all investigated real-world use cases and its relation to the established POWDER topology

Table 4.1: High-level details of the investigated ML application real-world use cases

Investigated ML application use case	**Employed data collection and its modality**	**Employed ML system(s)**	**Investigated DQ variation conditions**
Traffic sign images classification (sec. 4.2.2)	Subset of traffic and stop sign images from the Open Image V6 dataset [2]	VGG16, InceptionV3, EfficientNet, YOLO, Faster-RCNN	**Network QoS variations:** packet loss, buffer size; **other cyberinfrastructure problems:** noise, grayscale, contrast variation
Medical images classification (sec. 4.2.3)	Chest X-ray images dataset [98]	ResNet50, InceptionV3, VGG16	**Network QoS variations:** packet loss
Sound classification (sec. 4.2.4)	Firework [1] and gunshot [3] sound recordings datasets	VGG16, VGG19, ResNet50, InceptionV3	**Network QoS variations:** packet loss
Voice recognition and transcription (sec. 4.2.5)	English language subset of Mozilla Common Voice dataset [12]	DeepSpeech [79]	**Network QoS variations:** packet loss
Object detection and classification in videos (sec. 4.2.6)	Subset of The Berkeley Deep Drive (BDD110K) Dataset[6]	AWS Rekognition[7]	**Network QoS variations:** packet loss, buffer size

Open Images V6 dataset [2]. The dataset contains images labeled for classification, object detection, and semantic segmentation. The first step included transferring a set of images using UDP between two network nodes. Since UDP is an unreliable protocol, the transferred images appeared to be corrupted on the receiving endpoint. To recreate network service degradation conditions, two parameters are investigated during the image transmission: the packet loss and the receiver's buffer size. In Figure 4.5, we represent the steps of our empirical study related to investigating how the changing network conditions affect the ML application performance. In addition, in this use case we also consider the DQ variation due to other cyberinfrastructure conditions, such as sensor failures or data processing errors, which might result in noise or color changes. In Figure 4.6, we show the steps of our investigation.

Buffer Size

The buffer size of a UDP socket is defined as the amount of data transferred in one go from the sender side. It is important to note that if we naively transfer the image packets as we read them from a file, we could end up causing buffer overflow and have a part of the image being transferred in full, and a part of the image being completely missing due to the router or the software rejecting all other packets after the overflow.

Packet Loss

The packet loss in a network represents the ratio of packets that are dropped during communication, and can be calculated according to (4.1).

$$PLR = \frac{N^{tx} - N^{rx}}{N^{tx}} \times 100\%, \tag{4.1}$$

where N^{tx} and N^{rx} are the total number of transmitted and received packets [162]. Since in the POWDER platform data are transmitted over a real terrain, there is some inherent and underlying packet loss in each transfer. However, before starting each image transmission, we use built-in tools like *ping* and *iperf* in Linux to confirm that the inherent packet loss is minimal. These tools send out ICMP packets and return the percentage of packets that were dropped. We found that in almost all cases of our images transmission, 100% of the packets were successfully transmitted, except for one case, where we found that 99% of the packets were successfully transmitted.

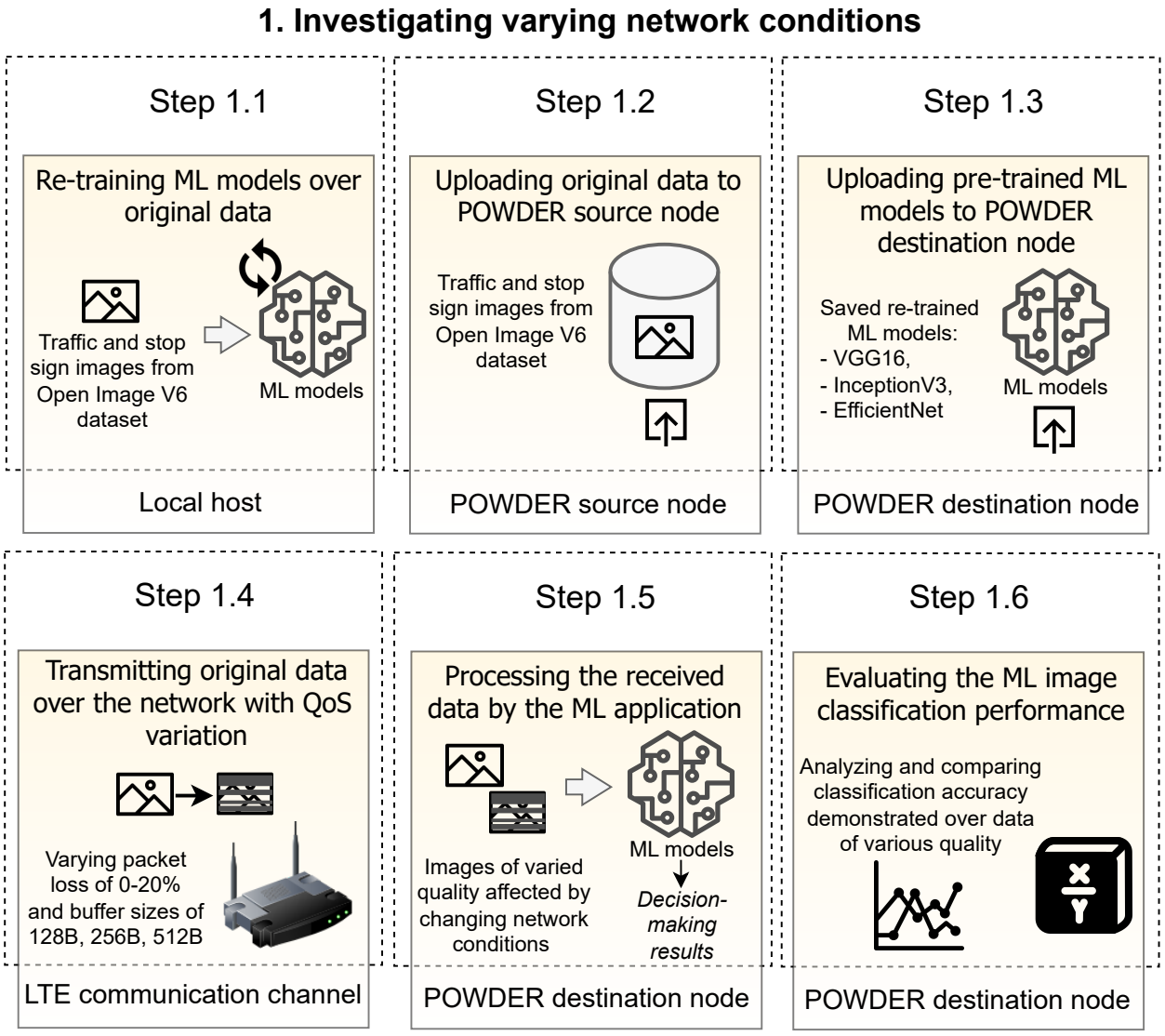


Figure 4.5: Schematical representation of the steps employed in the investigated traffic signs image classification real-world use case: varied DQ due to changing network conditions

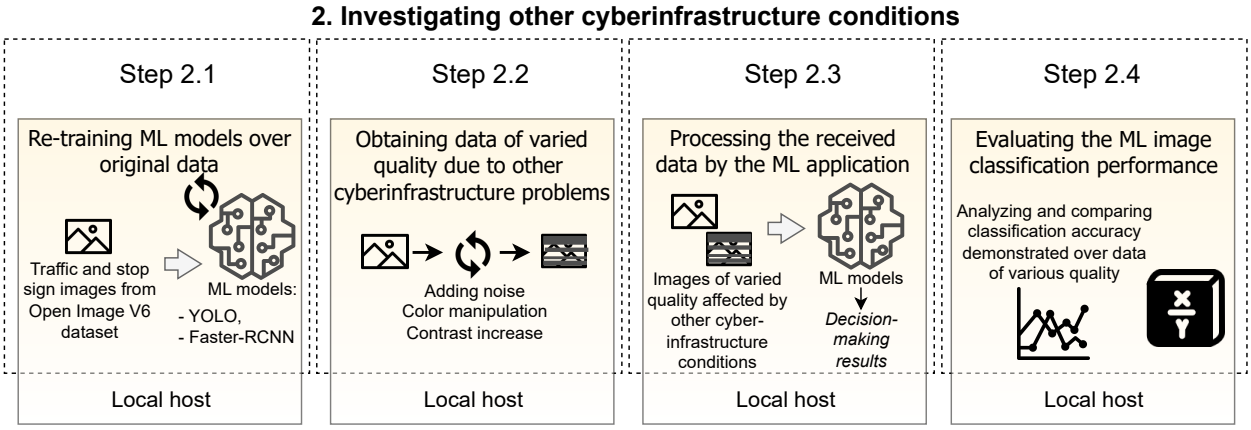


Figure 4.6: Schematical representation of the steps employed in the investigated traffic signs image classification real-world use case: varied DQ due to other cyberinfrastructure conditions

To recreate the packet loss, we employ the Linux network utility tools like *iptables* and *nftables*. *iptables* allows the user to set network parameters for dropping packets on statistical probability based matching rules. Using these utilities, we set rules for dropping packets on a random basis from 1 to 20%. Higher packet loss percentages are possible, but they result in too excessive packet losses (which creates unreasonably slow data transfers and makes images non-readable) and can interfere with the SSH connection itself (if the *-p* argument is set to "*all*"). Examples of the images transmitted with various buffer size and packet loss rates can be seen in Figure 4.7.

Other cyberinfrastructure problems

To assess how ML image classification applications behave being affected by such common technological factors as noise or contrast increase, we manually recreate these effects on the employed images. The first distortion involve adding various levels of noise to the input images. Noise is added by generating a random tensor the same size as the input image, multiplying it by a scaling factor, adding it to the input image, and clipping the result to be within the accepted range for an RGB image, which is 0 to 255. The employed noise addition function is represented by the equation (4.2). We first evaluate the performance of YOLOv3 and Faster-RCNN demonstrated over various noise levels to see how much noise is needed to distort the images enough to have an effect on the object detectors, and to see how much various noise levels affect the visual appearance of the images. Examples of images affected by various noise levels can be seen in Figures 4.8(b), 4.8(c), and 4.8(d).

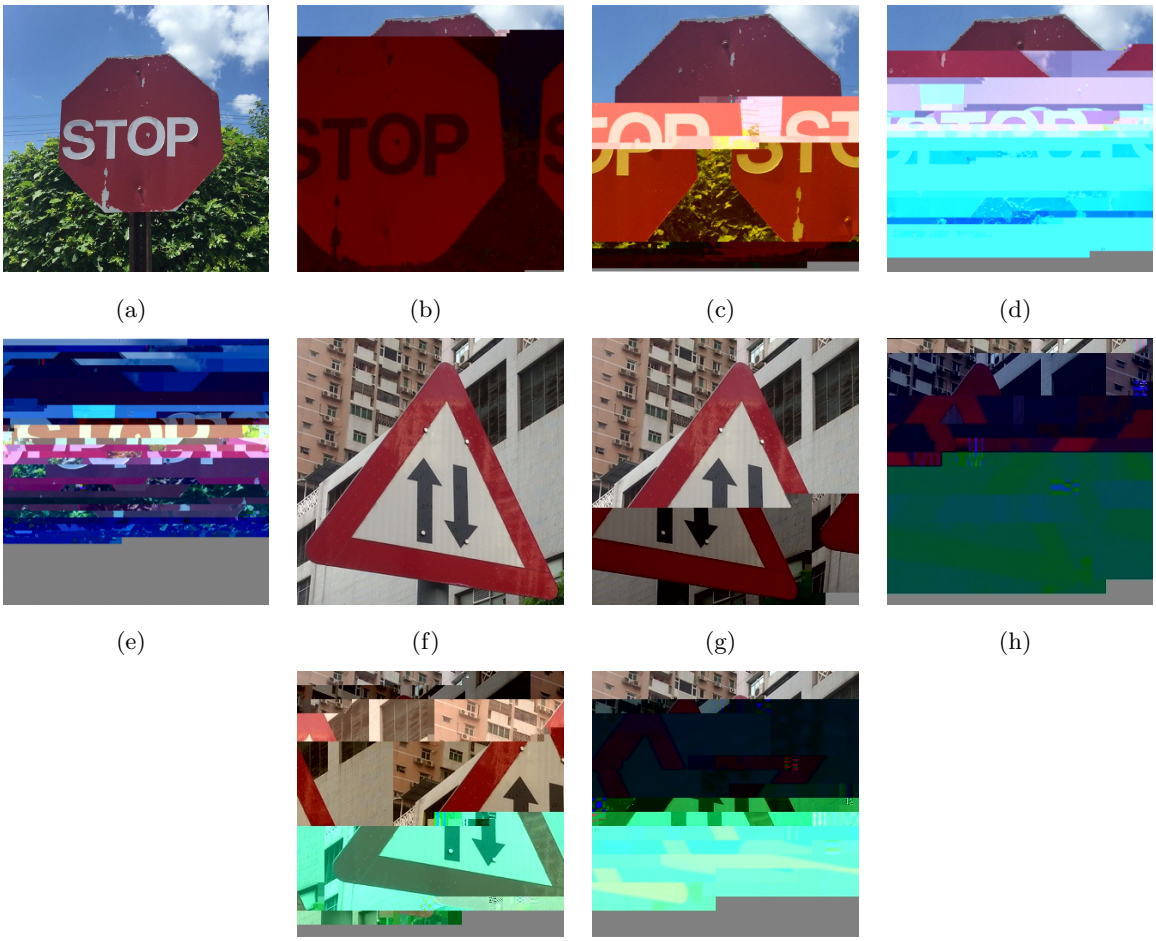


Figure 4.7: Stop and traffic sign after transmission them over a network with a 512B buffer size and various packet loss percentages: (a), (f) – original images; (b), (g) – 1% packet loss; (c), (h) – 5% packet loss; (d), (i) – 10% packet loss; (e), (j) – 20% packet loss

$$\hat{x} = clip(x + C \times rand(0, 1), 0, 255), \tag{4.2}$$

where x is the original image to which the noise is being added; $\hat{x}$ is the resulting image after the noise addition; *clip* is the clipping operation; C is the scaling factor, which acts as a noise intensity parameter; and *rand* is a function that generates a random number in the specified interval.

We also investigate another technical issue factor which leads to image's pixels color modification through contrast increase. We modify the original contrast by using "contrast enhancer" method from the Python's library Pillow[8] with a value of 0.01. An example of an image with increased contrast can be seen in Figure 4.8(e).

Another technique we use to recreate images affected by technical issues is converting the color spectrum into grayscale. This is done by averaging the brightness value of each pixel in the image according to its RGB components. An example of a grayscale image produced using this technique is represented in Figure 4.8(f).

Employed ML Image Object Detectors and Classifiers

In our network QoS variation experiments, we employ VGG16, Inception, and EfficientNet, which are well-known state-of-the-art ML image classifiers. These models are established to provide reliable classification results, as they are pre-trained over well-known huge ImageNet dataset [59] that includes around 15 million miscellaneous images spread among 1000 categories. Below we provide a brief description of each of these ML models.

VGG16 VGG16 (Very Deep Convolutional Networks for Large-Scale Image Recognition) model [204] is widely regarded as a robust and reliable model in the ML community. VGG16 has a sequential operational flow. It includes multiple layers consisting of 3×3 shaped filters, designated to decrease the input image features dimension. At each hidden layer the ReLU activation function is used. The final parameters number is 138 million. VGG16 was the runner-up at the 2014 ILSVRC conference competition, and was outperformed by the Google's GoogLeNet model, which is currently avowed as the Inception model.

[8] `https://pillow.readthedocs.io/en/stable/`

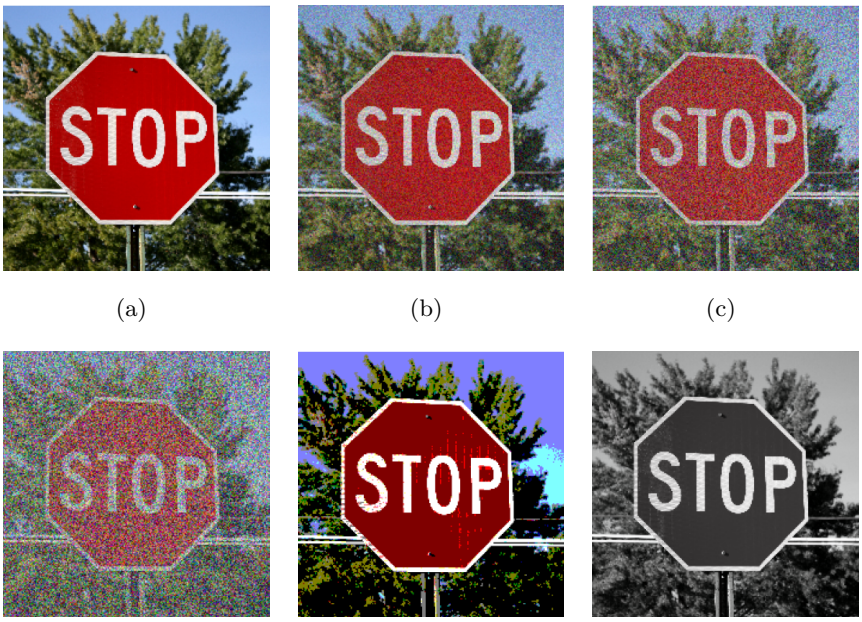


Figure 4.8: Images affected by other cyberinfrastructure failures or errors: (a) - original image; (b) - noise = 100; (c) - noise = 200; (d) - noise = 500; (e) - contrast change; (f) - image converted into grayscale

Inception The earlier version of Inception had only 7 million parameters, which was far fewer as compared to the well-known models such as VGG16 and AlexNet. It demonstrated much lower error rate that was considered as a design breakthrough. The model realization included a novel Inception module [206]. The Inception module performs the convolution process with multiple filter sizes (1×1, 3×3, and 5×5), follows it up with Max Pooling, and appends the result to the following layer. The employment of 1×1 shaped filter for convolution operation allows to drastically reduce the data dimension. The major enhancement was introduced in the second version of Inception, which made the architecture less complicated, and elevated the classification performance. In the same paper [206], the authors described the modifications released in the third version of Inception. The major enhancements were: batch normalization employment, RMSProp optimizer, and greater factorization.

EfficientNet On the same vein with the Inception Model, the EfficientNet model was developed by Google. As the major novelty, the EfficientNet developers proposed the compound scaling strategy [209], that directs the model to adjust its parameters depending on the input image size and resolution. All the previous models utilized the traditional scaling approach, which made the

models' architecture deeper by adding up more layers in advance. On the contrary, to achieve better flexibility and optimize the classification performance, the EfficientNet developers proposed to scale the image dimensions by a compound coefficient simultaneously and consistently.

Below we describe ML models employed in other cyberinfrastructure failures and errors scenarios.

YOLO The YOLO object detection system is a one-stage detection algorithm that applies a single neural network to the full image. This network divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities. The authors claim that YOLO has several advantages over classifier-based systems. YOLO looks at the whole image at test time so its predictions are informed by global context in the image. It also makes predictions with a single network evaluation unlike RCNN systems which require thousands for a single image. This makes YOLO much faster than RCNN systems [180]. This makes YOLO a highly popular open-source solution widely employed both in research and industry.

Faster-RCNN Faster-RCNN (region based convolutional neural network) is the fastest object detector in the family of RCNN detectors. Faster-RCNN is a two-stage object detector, and is, therefore, more computationally expensive than YOLO. The architecture of Faster RCNN consists of two modules: RPN (region proposal network) for generating region proposals and Fast-RCNN for detecting objects in the proposed regions. The code developed by the authors of [181] is open-source and, again, can be used in academia or commercially.

Empirical Results on how DQ Variation Affects ML Image Classification and Object Detection Performance. Network QoS Variation Scenario

In our experiments we started the ML performance evaluation with original images. The classification train accuracy amounted to 92.24% for VGG16, 89.27% for Inception, and 87.92% for EfficientNet; the transfer accuracy demonstrated by the same models was 93.24%, 93.92%, and 95.96%. The train accuracy evaluates the classifier performance on images submitted to the classifier models, which have been pre-trained on the generic training set. On the other hand, the transfer accuracy evaluates the classifier performance after it has been further trained on images similar to submitted ones.

Figure 4.9 and Table 4.2 show the drop of image classification models accuracy on the test image

set with various packet loss percentages and receiver's socket buffer sizes sent over the POWDER testbed. From the results one can see that for smaller buffer sizes, increasing packet loss causes a higher image file corruption, and as a result, image classification models may loose their ability to assign a label to these images at all. Retention column in Table 4.2 lists the ratio of images that the model was able to process and classify. Relative Accuracy in this Table takes into account this ratio in re-calculating the classifier's accuracy, so this column shows a further accuracy drop as some images were too corrupted and became non-readable.

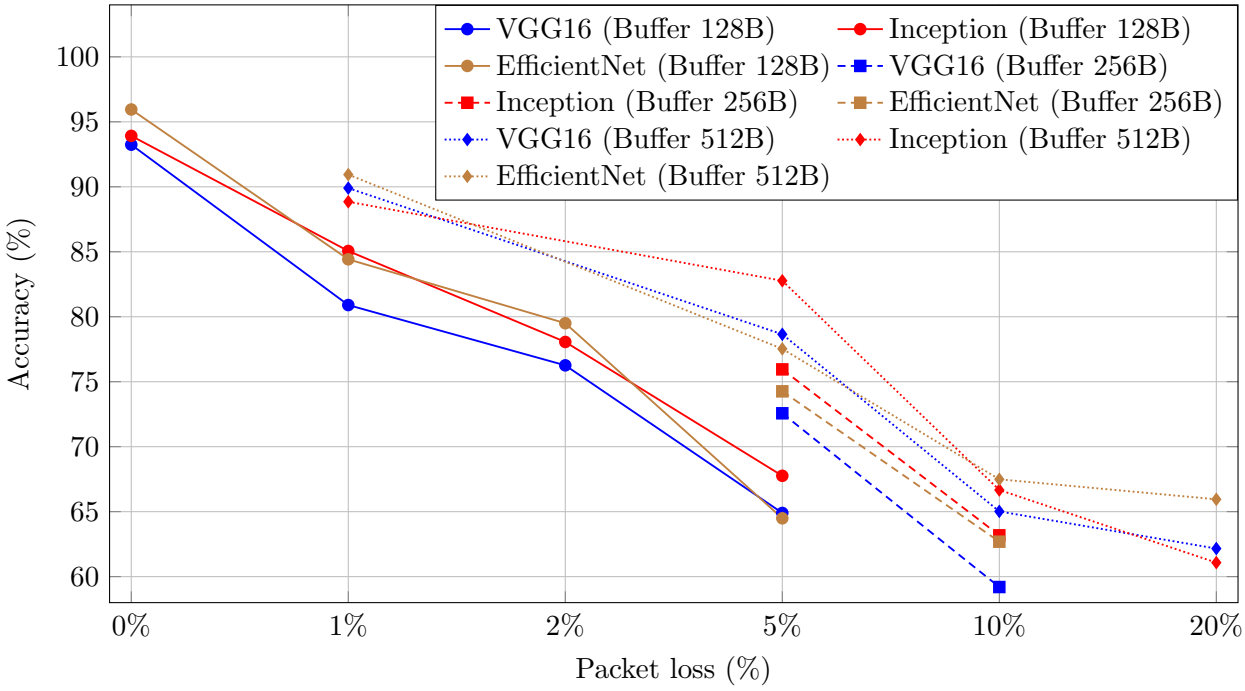


Figure 4.9: Packet loss and buffer size influence on the image classification accuracy demonstrated by the ML models

As can see from Table 4.2, in the case of buffer scarcity (128B), even 1% packet loss may result in more than 10% drop of accuracy of ML classifiers such as VGG16 and EfficientNet. In the meantime, with a larger buffer size (512B) the accuracy of ML classifiers decrease by at most 5%. For 5% packet loss, we can see that the accuracy is still better than 75% for all the models. However, 10% packet loss decreases the accuracy classifiers by more than 30% for almost all models, which makes the classifier not acceptable in most applications.

Although some of the considered ML models appear to be more robust to the network QoS degradation than others, in all the considered cases even a small packet loss resulted in intolerable ML application performance. Such factors as increase in packet loss rate and receiver node's buffer

Table 4.2: Change in accuracy of image classification models depending on network conditions and resources

Image type	**Retention (%)**	**Models**	**Accuracy**	**Relative Accuracy (%)**
Original	100	VGG16	93.24	-
		Inception	93.92	-
		EfficientNet	95.95	-
Buffer 128B, Packet loss 1%	95	VGG16	80.9	76.85
		Inception	85.07	80.82
		EfficientNet	84.42	79.25
Buffer 128B, Packet loss 2%	85	VGG16	76.26	64.82
		Inception	78.06	76.35
		EfficientNet	79.5	67.57
Buffer 128B, Packet loss 5%	75	VGG16	64.9	48.67
		Inception	67.76	50.82
		EfficientNet	64.49	48.36
Buffer 256B, Packet loss 5%	80	VGG16	72.57	58.05
		Inception	75.95	60.76
		EfficientNet	74.26	59.4
Buffer 256B, Packet loss 10%	75	VGG16	59.2	44.4
		Inception	63.18	47.38
		EfficientNet	62.69	47.01
Buffer 512B, Packet loss 1%	95	VGG16	89.9	85.4
		Inception	88.85	84.4
		EfficientNet	90.94	86.39
Buffer 512B, Packet loss 5%	85	VGG16	78.65	74.71
		Inception	82.77	78.63
		EfficientNet	77.53	73.65
Buffer 512B, Packet loss 10%	80	VGG16	65.02	52.01
		Inception	66.67	53.33
		EfficientNet	67.49	53.99
Buffer 512B, Packet loss 20%	75	VGG16	62.16	46.62
		Inception	61.08	45.81
		EfficientNet	65.95	49.46

Table 4.3: Accuracy for YOLOv3 and Faster-RCNN on the images affected by various cyberinfrastructure failures and errors

Cyberinfrastructure factor	**Accuracy**	
	YOLOv3	**Faster-RCNN**
Original image	0.99987	0.99987
Noise ($C = 100$)	0.99987	0.99993
Noise ($C = 200$)	0.99968	0.99610
Noise ($C = 500$)	0.99985	0
Grayscale	0.99986	0.99868
Contrast increase	0.99984	0.99859

resource limitations may lead to the transmitted data corruption, and to the incorrect decisions made by ML end applications based on this data. Real-time ML applications that are actively employed in such areas as, for example, transportation or healthcare could be especially sensitive to performance decline.

Empirical Results on how DQ Variation Affects ML Image Classification and Object Detection Performance. Cyberinfrastructure Failures and Errors Scenario

To investigate how various technical issues resulting in DQ variations can affect the performance of ML image classifiers, we processed a number of images disrupted by techniques described in sec. 4.2.2. In total, we evaluate the performance of 10 images of each category and reported the average performance in Table 4.3. According to our results, none of the considered image distortions had a significant impact on the pre-trained foundation image classification models, with the exception of adding noise. The performance only dropped when a substantial amount of noise was added into the image so that it began to have an impact on the visual appearance of the image. When the noise scaling factor was set to 500, Faster-RCNN was unable to detect the stop sign in the images at all. A summary of these results is presented in Table 4.3. It is clear that the considered distortion techniques did not have enough of an effect on ML models performance in order to work as adversarial examples. More advanced techniques must be attempted to be able to create images which are visually similar to their originals while still being able to fool pre-trained foundation image classification models actively employed in the ML community.

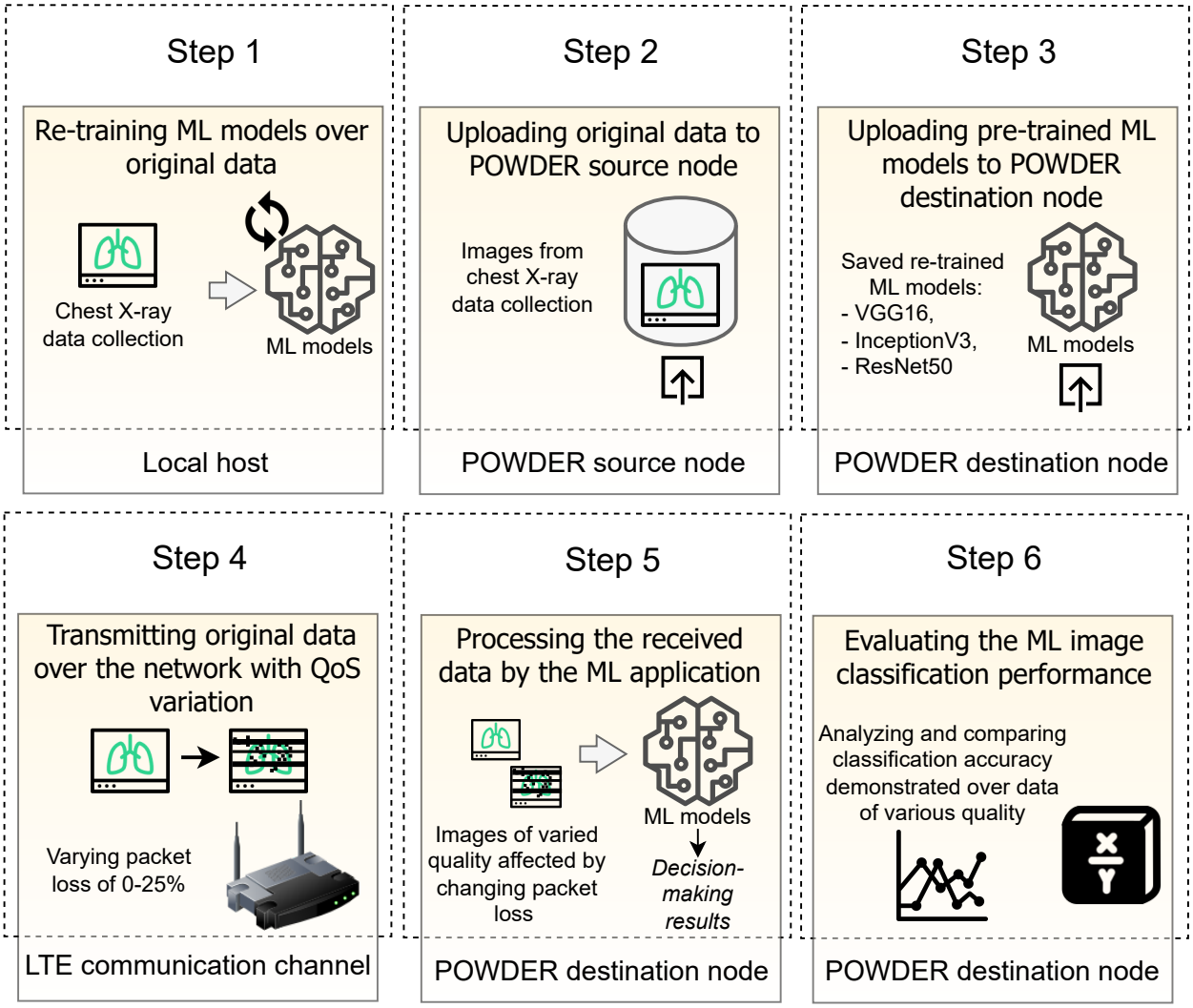


Figure 4.10: Schematical representation of the steps employed in the investigated medical image classification real-world use case involving varied DQ due to changing network packet loss

4.2.3 Medical Images Use Case

Input DQ is extremely essential in the medical-related ML applications, as there is the risk of improper and harmful decisions regarding the patient's treatment. In this use case, we investigate the interrelationships between DQ degradation due to network QoS deterioration and ML application performance on the example of ML medical images classification system. For our experiments, we employ real-world X-ray scan images labeled into two categories: normal and abnormal one. We utilize a part of the dataset to re-train the ML image classification models we employ. We study

the performance of three well-known industrial ML image classification systems: VGG16 [204], ResNet50 [81], and InceptionV3 [206] pre-trained over ImageNet dataset [59]. To investigate how the medical image data transfer over a network may affect the performance of ML image classification, we employ POWDER platform. We leverage the network topology, described in sec. 4.2.1, to transmit images over a network. To recreate network QoS degradation conditions during the data transfer, we vary packet loss percentage. Then, after the data transfer, on the transferred medical images are submitted to ML image classifiers. We evaluate the performance demonstrated by the ML image classifiers on each of the medical image group transmitted with various packet loss percentage. Then, we compare and analyze the performance achieved by each ML image classifier over various packet loss percentages. Below we provide additional details related to our experiments. In Figure 4.10, we represent the steps of our investigation and the facilities used throughout the study.

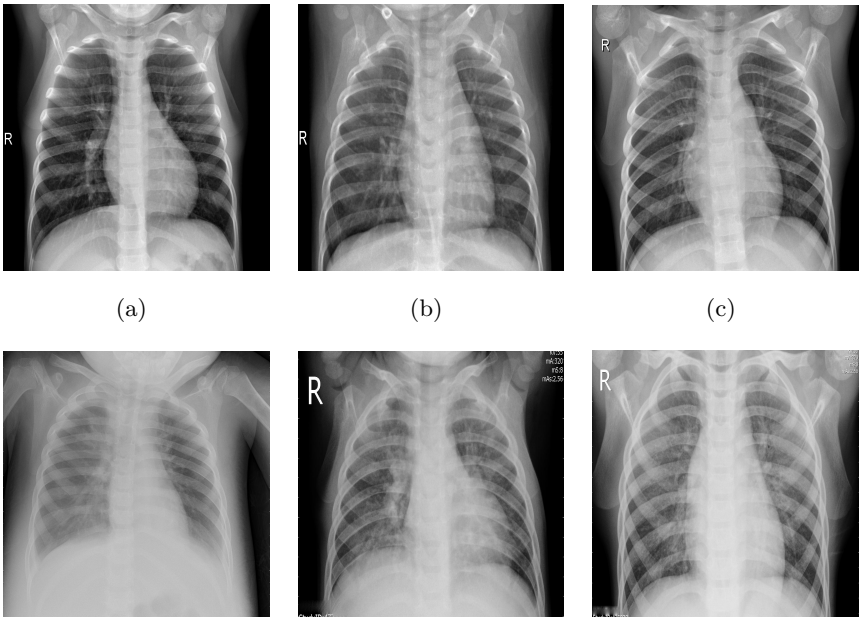
Employed medical images collection

A chest X-ray dataset [98] is used to identify the positive cases of pneumonia. The images are categorized as normal (no lung opacity) and opaque (showing lung opacity). Examples of the employed data samples are demonstrated in Figure 4.11. The employed dataset contains 5,856 total observations, which are partitioned as follows: training set: 4,192 (1,082 normal, 3,110 lung opacity); validation set: 1,040 (267 normal, 773 lung opacity); testing set: 624 (234 normal, 390 lung opacity). To decrease the risk of ML models overfitting to the training data, we perform training data augmentation procedures. In particular, we apply such augmentation techniques as images re-scaling, shear transformation, zooming, and horizontal flipping.

Employed ML Image Classifiers

In our study, we leverage such foundation image classification models, as ResNet50, InceptionV3, and VGG16. We import the weights from the models pre-trained over the ImageNet dataset, and perform TL to adapt the models for our target domain. In sec. 4.2.2 and 4.2.2 we provided general description for the InceptionV3 and VGG16 ML models respectively. Below we describe the employed ResNet50 ML architecture.

ResNet50 It is a residual Artificial Neural Network (ANN) model's variation, which consists of 48 Convolution layers, 1 Max Pooling layer, and 1 Average Pooling layer. ResNet50 has substantially



(a) (b) (c)

(d) (e) (f)

Figure 4.11: Examples of the medical images from the employed dataset: (a) – (c) are the X-Ray scans of normal lungs without opacity; (d) – (f) are the X-Ray scans of abnormal lungs with opacity

deeper structure than VGG16 and Inception v3, with a total of 50 layers. A residual neural network is a type of ANN the architecture of which is based on stacking residual blocks on top of each other. This allows to establish shortcut connections between network's layers, which helps to train the model over thousands layers without substantial performance decrease. Residual networks, on the other hand, have fewer filters and are less extensive than VGG16. ResNet is based on micro-architecture modules, which can be dscribed as network-in-network architecture. Input images for ResNet50 should be cropped to satisfy the height and width dimension requirements: 224×224 pixels.

Employed Network Parameters while the Data Transmission

For our empirical study, we employed similar LTE network topology with two nodes we established in sec. 4.2.1. The former node is used as image dataset storage, it transmits X-ray images over the network toward the second node. The latter ML application node receives the transmitted images and classifies them into two categories: normal and abnormal (with opacity in lungs). Before the image transmission, we set the network packet loss rate, varied from 0 up to 25%, as higher rates

Table 4.4: Medical image classification performance metrics, demonstrated by the employed ML image classifiers over original X-ray scans

Characteristic\Model	**VGG16**	**ResNet50**	**InceptionV3**
Input image size	224×224	224×224	299×299
Weights size	>500 MB	≈100 MB	≈100 MB
Parameters number	138,357,544	23,512,130	21,789,666
Test Accuracy	0.952	0.916	0.886

result in too corrupted images which cannot be opened on the receiver. We experiment with the following percentages: 1, 2, 5, 10, and 25%.

Empirical Results on how DQ Variation Affects ML Medical Image Classification Performance

First, we re-train the employed ML image classifiers on the original X-ray images. For this, we rely on TL procedure, which allows to re-use the previously trained model's weights except the last layer that is re-trained according to a required classification task. We re-train our models on the set of 624 X-ray images (234 normal, 390 lung opacity). Then, we test the employed ML classifiers on the original non-distorted image set and evaluate their performance benchmark. We also extract the ML models hyperparameters and analyze them. Table 4.4 represents the results on ML classifiers performance and hyperparameters after the re-training stage. As one can see from Table 4.4, VGG16 demonstrated the hishest performance of 0.95, and has the largest weight size and parameters number, which is five times higher than ResNet50 and InceptionV3 parameters. Such a difference in weight size and parameters number is justified by the VGG16 model convolutional structure that is substantially deeper than ResNet50 and InceptionV3. ResNet50 demonstrated slightly better performance than InceptionV3 on the original images testing set: 0.916 versus 0.886.

After the re-training and ML performance benchmark establishment step, we test the classifiers on the images corrupted by varied packet loss percentages in the process of their network transmission. We evaluate the classification performance demonstrated by each employed ML image classifier, and compare it with the benchmark established on the original non-corrupted images. Based on our comparison, we analyze how the network packet loss increase affects the ML image classification performance. The classification performance results on corrupted images is demonstrated in Figure 4.12.

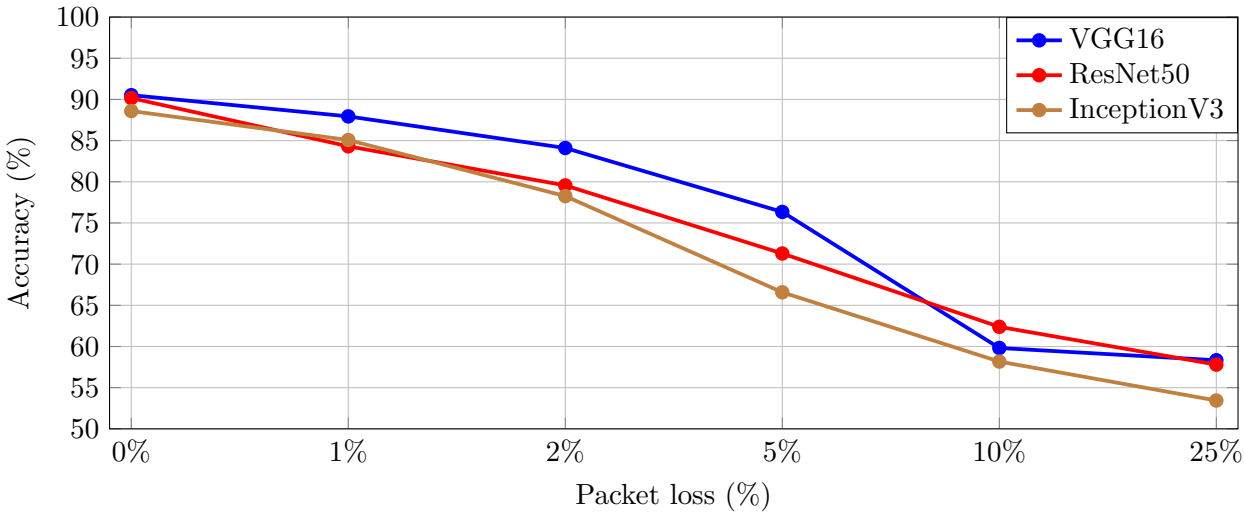


Figure 4.12: The interrelatioship between network packet loss and ML classification performance

From Figure 4.12 one can see, that with the network packet loss increase ML classification performance gradually decline. Indeed, network packet loss affects the transmitted data integrity and quality, which was also investigated in our preliminary studies on traffic sign images [46]. Depending on the packet loss percentage, the images visual representation may vary. For example, low packet loss rates might slightly corrupt insignificant part of image's pixels, so the major classification patterns can still be captured and utilized by the ML classifier to assign the image to a proper category in most cases. In case of higher packet loss percentage (typically, higher than 5%) the image's pixels representation on the receiving end might be significantly modified: pixels might be misplaced; pixel's color might be changed; various artifacts and noises might appear; some parts of the image might be completely lost. The reason of those changes is the image file encoding errors emerging due to partial data losses in the process of its network transmission.

From the obtained results one can see, that with the network packet loss increase all the studied ML image classifiers loose their capacity to properly assign labels the X-ray medical images they are tested on. VGG16 demonstrates the leading ML classification performance up to 5% packet loss rate. ResNet50 and InceptionV3 perform quite similar, but both has lower performance than VGG16 from the very beginning, and follow the same trend until the packet losses rate reaches 2%. After 2%, InceptionV3 demonstrates the worst performance up to the highest packet loss rate tested. On 10%, VGG16's performance surprisingly drops to 59.82, which is below ResNet50 with 62.39. However, after 10% VGG16 drop is smoothed out, and on 25% VGG16 again shows the best performance of 58.33, which is marginally better than ResNet50 with the result of 57.8. Another

interesting observation, is that the performance of all the studied ML image classifiers follows similar decline trend until 10%, but then the decline becomes substantially smoother. Indeed, on the interval from 0 to 10% packet losses the performance of all the studied image classifiers decline by around a third (33.03%) on average. However, the same decline on the interval from 0 to 25% is 37.03%. This can be explained by smaller differences how the data losses of 10% and 25% affect the visual representation of the medical image file on the receiving end.

In summary, comparing the general behavior of all three studied foundation image classification models with respect to various packet loss ratios, one can see that VGG16 consistently performs better than both InceptionV3 and ResNet50. However, 10% packet loss reduces accuracy by more than 25-30% for all the studied ML classifiers, which cannot be tolerated in the medical image classification area. According to our results, even a small packet loss rate can result in a considerable decline in ML medical image classification performance. We found that depending on a medical image classification system, only the packet loss of less than 1% could be reliably tolerated. All considered foundation image classification models fail when packet loss reaches 2-5%.

We can recommend employing VGG16 classifier that demonstrated more robust performance on corrupted images against ResNet50 and InceptionV3. Also, in order to reduce the image quality degradation, when packet loss reaches 2%, we recommend replacing UDP file transfer with more reliable protocols for data transmission, as studied in [43]. Medical image classification and recognition systems are required to provide robust performance results to assist medical workers in accurate patient diagnosis. Lack of robustness to DQ variations might have critical health-related consequences for the patients in case wrong diagnosis and improper treatment, which need to be addressed by the medical and ML communities.

4.2.4 Sound Classification Use Case

The transmission of sound in sensor and IoT networks is common in many modern ML applications. In such applications, data sources might be equipped with sound sensors (e.g., microphones), and are able to transmit the captured sounds to a remote server or other devices. In some cases, the processing of the captured sound can take place on the IoT device, however, it usually requires sufficient computational resources, which may be too prodigal for small IoT devices. An example of such an application is ShotSpotter system [63], that is installed in many US cities. This system employs data from acoustic sensors to detect the gun shootings and determine their approximate location in order to dispatch law enforcement.

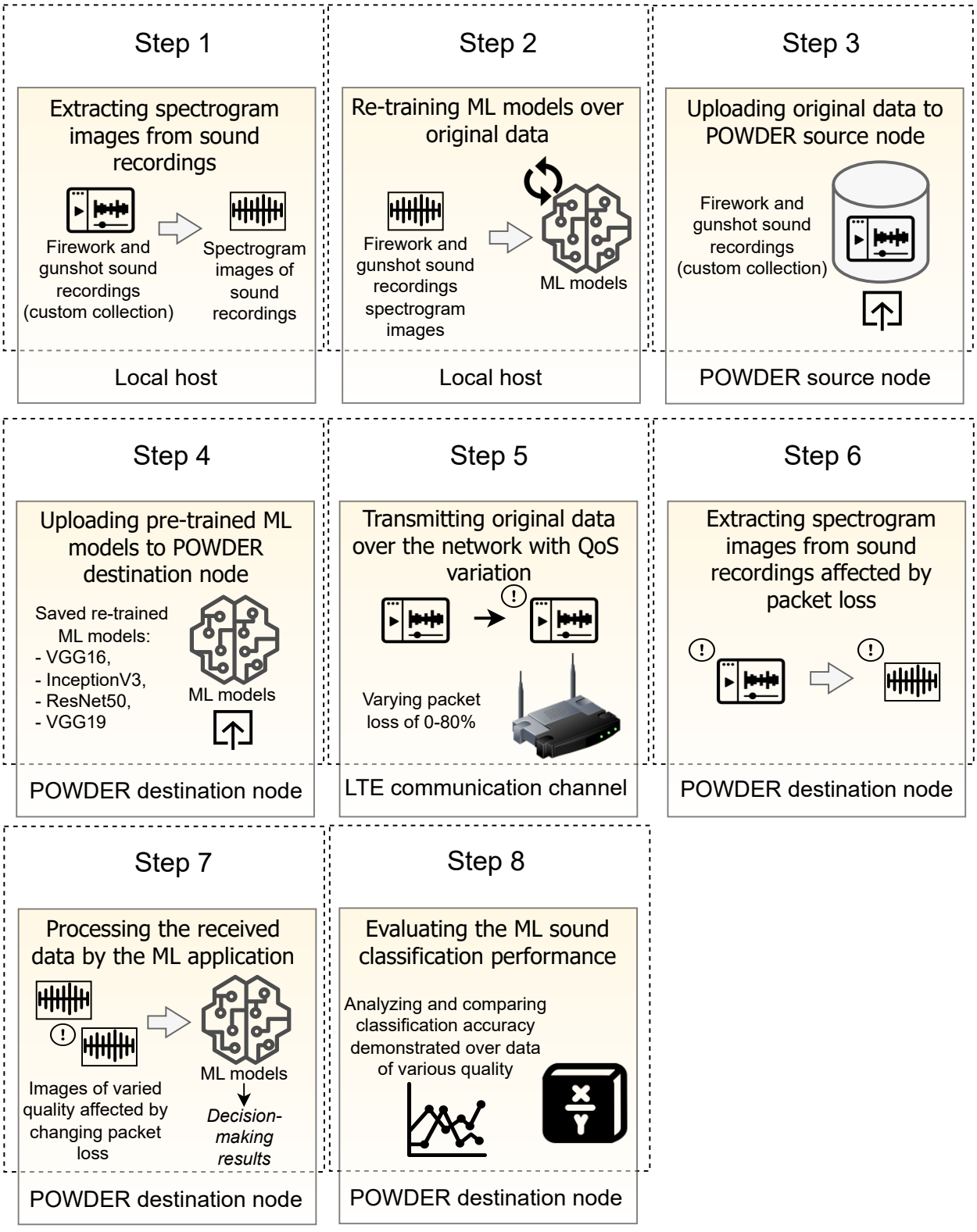


Figure 4.13: Schematical representation of the steps employed in the investigated sound recordings classification real-world use case involving varied DQ due to changing network packet loss

Numerous network protocols are used to broadcast media files over Wireless Multimedia Sensor Networks (WMSNs) that are employed to deliver multimedia data, including images, videos, and audio, between various sensor and IoT devices [196]. Some of them utilize UDP at the transport level (e.g., CoAP, MQTT-SN), while others are based on TCP (e.g., MQTT, SMQTT). As IoT devices are usually limited in computational and energy resources, the employment of UDP allows longer device uptime and reduces resource consumption. While not reliable, UDP provides a substantially higher data transmission rate than TCP, which makes UDP indispensable for real-time network-related ML applications, which we study in our use cases.

In some IoT network configurations, it might be necessary to broadcast data from one IoT device to many others. In this case, either UDP or TCP transport protocols can be used. However, the use of TCP is more resource constraint, as it requires to establish the connection and then to verify that the transmitted data has been received. This increases the resource consumption that may not be feasible for low-power IoT devices, and can lead to their fails or network latency increase. In addition, the use of TCP decreases the data transmission rate in comparison to UDP, which may not be tolerable in real-time systems.

In this real-world use case, we employ our custom sound dataset synthesized from the two YouTube videos [1,3], which we transmit over the network with varying QoS conditions using a UDP protocol. The dataset contains sound recordings of gunshot and firework sounds. As in the previous use cases, we transmit these sounds over the established POWDER wireless network topology (see sec. 4.2.1). On the cloud-based ML application, the spectrogram images are first extracted from the transmitted sound recordings. These spectrogram images are then classified by the ML image classifier, which has to differentiate the spectrograms between the gunshot or firework categories. Based on the provided classification results, we evaluate the ML application performance. We examine the interrelations between various QoS levels and the ML end performance. In Figure 4.13, we showcase our investigation steps alongside the data, tools, and facilities employed in these steps.

Employed Sound Recordings Collection

We extract audio files from the YouTube videos incorporating gunshot and firework sounds. Gunshot sounds dataset [3] consists of 3 hours of audio, and firework sounds set [1] of 1.14 hours of audio recordings. Out of both files, 66.75 minutes of audio are extracted and split into 15 sec *.wav* files to compose a balanced dataset consisting of 267 files for the gunshot class and 267 files for the firework class files. To provide consistent data to the pre-trained classifiers, all *.wav* files are set

at a frame rate of 16 kHz. The audio *.wav* files are then converted into spectrogram images to be given as an input to the image classifiers.

Employed ML Classifiers

To investigate another ML application, we employ ML image classifiers for a sound classification task, which is a common practice in ML sound classification [83]. To make sound recordings acceptable to be processed by the image classifiers, we represent these recordings as their spectrogram images. In this case, we employ such foundation image classification models, as VGG16, ResNet50, InceptionV3, and VGG19. We provided general description of VGG16, ResNet50, and InceptionV3 in sec. 4.2.2 and 4.2.3 respectively. VGG19 differs from VGG16 only in the number of the convolutional layers employed in the ML architecture. After the training stage, we process our spectrogram images, extracted from the audio files transmitted with various packet loss percentages. Then, we evaluate the performance ML application demonstrated over the transmitted data samples.

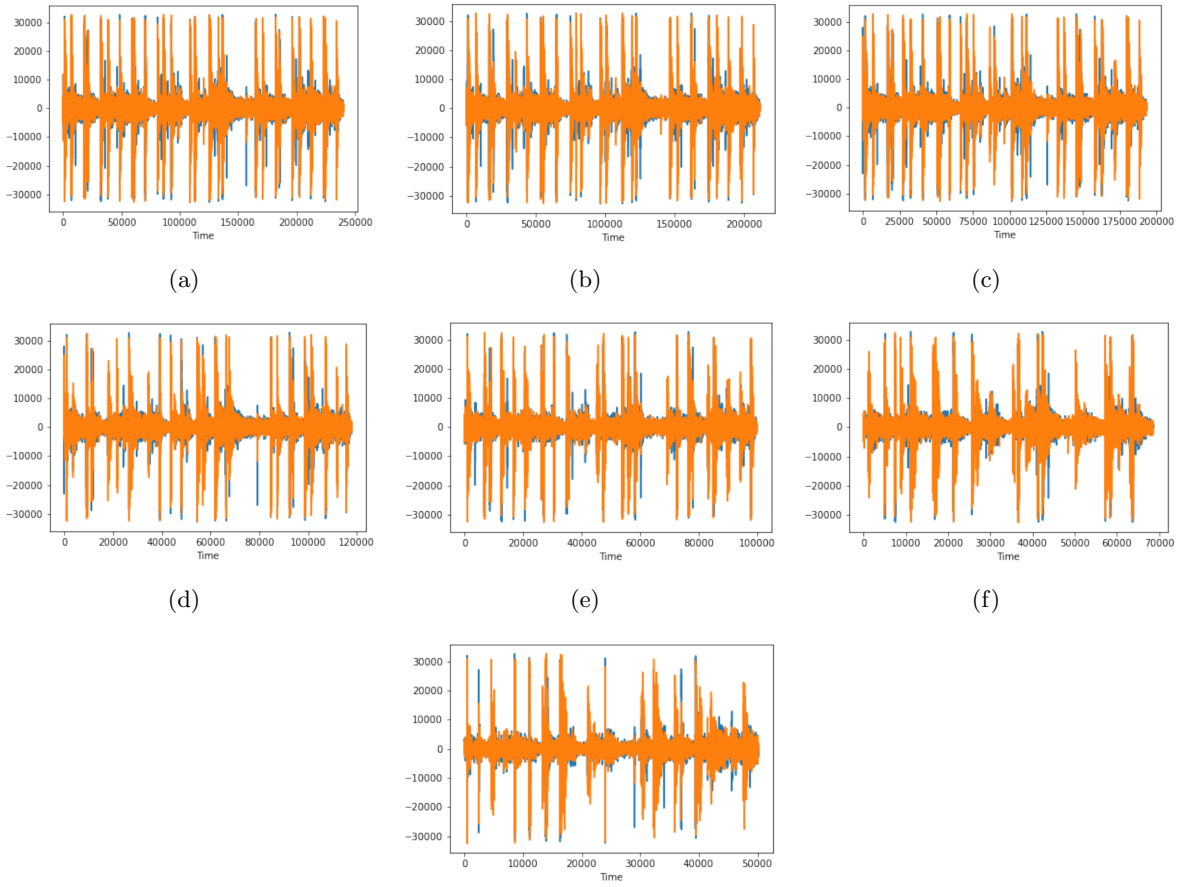
Employed Network Parameters while the Data Transmission

As in the previous use case, original audio *.wav* files are communicated through the POWDER platform network with the UDP protocol. Since audio files have been found to be more robust to packet loss than images, we investigate network conditions causing higher loss ratios between 10-80% with a 10% step. In some cases, packets lost the valuable *.wav* header information, which is replaced by the default *.wav* header, so the system can still read them properly. The audio *.wav* files are then converted into spectrogram images. In addition to 534 original files, 2670 new files with various losses are included into the dataset bringing the total dataset size to 3204 samples. Some examples of the employed spectrograms are represented in Figure 4.14

Empirical Results on how DQ Variation Affects ML Sound Classification Performance

To train the employed classifiers, the original dataset without any packet loss corruptions is split into 60:40 train and test sets that resulted in 160 spectrogram files of both classes. To test these models, the test set containing 107 files is used over the saved models to produce testing accuracy. The testing accuracy after the training step resulted in:

- VGG16 test accuracy: 95.75%;



(a) (b) (c)

(d) (e) (f)

(g)

Figure 4.14: Examples of firework sounds spectrograms affected by various packet losses: (a) – no packet losses; (b) – 10%; (c) – 20%; (d) – 50%; (e) – 60%; (f) – 70%; (g) – 80%

- VGG19 test accuracy: 96.23%;
- ResNet50 test accuracy: 94.81%;
- Inception V3 test accuracy: 91.04%.

Based on the accuracy achieved by all the models, we can state that these foundation image classification models can be used to classify the technical images but only after the initial retraining on the particular image types. We employ the models to process a set of spectrogram images extracted from the sound files transmitted with the packet loss of 10, 20, 50, 60, 70, and 80%. Each corrupted image set contains 107 file samples. These test files are used as input to the pre-trained models described above.

Testing accuracy achieved over the files corrupted by different packet loss ratio is presented in Table 4.5. While VGG16, VGG19, and InceptionV3 demonstrate high robustness to low packet loss until the ratio reaches 20%, ResNet50 performance experiences the fastest drop. However, when the packet loss passes over 60%, VGG16's, VGG19's, and ResNet50's performance become commensurate too. With 80% loss, ReSNet50 shows better outputs than VGG16 and VGG19, in which detection performance drops by half from the initial value. InceptionV3 demonstrates the lowest accuracy on the original image recognition while increasing packet loss does not cause any substantial detection performance decline until around 80% of transferred data are missed. Based on these results, we can conclude that pre-trained VGG16 and VGG19 should be employed in well-serviced trusted networks, in which loosing more than a half of transmitted data is highly unlikely. On the other hand, InceptionV3 classifier could be recommended if the network QoS could go low.

We also analyze the files' size decrease in the process of their transmission over the network with various packet loss ratios. The results for both firework and gunshot sound sets can be observed in Table 4.5. High tolerance of ML-based classifiers to the significant packet loss in original sound files can be explained by a substantial redundancy in that results in producing similar spectrograms over the conversion of sound files into image files to be classified. Notwithstanding the packet loss is decreasing the information content in the original file, the conversion process lower the modifications of the image files due to the same packet loss that allows image classifiers to perform much better on audio spectrogram images than on real corrupted images.

Table 4.5: Results on ML performance for sound classification

Packet loss	Avg. file size (kB)		Model Performance			
	Firework	Gunshot	VGG16	VGG19	ResNet50	InceptionV3
0%	938	938	95.75%	96.23%	94.81%	91.04%
10%	826	850	96.23%	96.23%	87.74%	92.92%
20%	756	754	95.75%	91.04%	77.36%	90.57%
50%	460	483	95.75%	91.04%	77.36%	90.57%
60%	390	384	81.13%	74.06%	76.42%	90.09%
70%	268	280	64.34%	64.62%	68.40%	86.32%
80%	196	188	54.25%	51.89%	67.45%	81.60%

4.2.5 Voice Recognition and Transcription Practical Use Case

To transmit a voice over a network in real-time, a VoIP application has become prevalent. VoIP usually employs UDP as a transport protocol, which allows faster data transmission in contrast to TCP. TCP traffic needs to undergo some connection establishing procedures, as synchronization and acknowledgement (known as SYN, SYN-ACK, ACK). If there are a large number of nodes to communicate in the network, the transmission of voice over TCP protocol on the transport level becomes inefficient in terms of resources and can create intolerable latency. Using UDP for real-time voice transmission is significantly more efficient in terms of required network resources and data transmission rate. However, UDP is unreliable and cannot handle network packet losses that can deteriorate the end user's QoE. In other words, when the packet loss ratio exceeds some specified threshold, the communication network QoS becomes unacceptable to provide users with high quality calls.

In recent years, many companies and organizations started employing various real-time voice assistants to address growing consumer demands and provide better services [175]. These tools are commonly designated to assist users in performing the specific actions during the phone calls (e.g., creating a maintenance request or obtaining on-demand information). Usually, these assistants are based on ML models, that are able to recognize and interpret users voice, and automatically perform specified actions based on these interpretations. The loss of network packets in the process of voice communication can decrease the assistant's performance, and decrease the user's QoE.

The effect of network QoS degradation on the human end users experience has been actively exam-

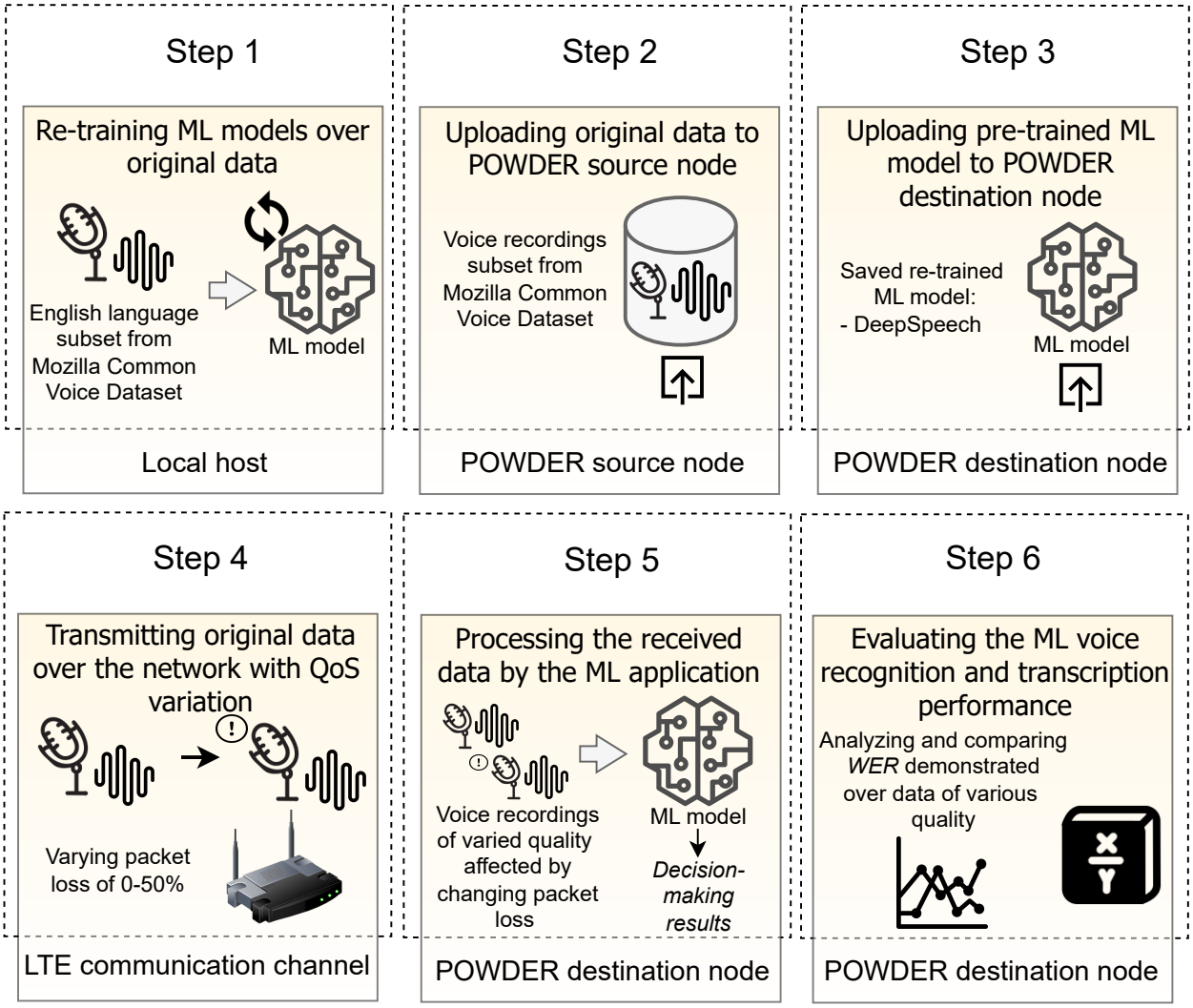


Figure 4.15: Schematical representation of the steps employed in the investigated voice recognition and transcription real-world use case involving varied DQ due to changing network packet loss

ined in various applications [118, 144, 153]. However, its effect on artificial ML end system remains under-researched. In this real-world use case, we investigate how the performance of ML voice recognition and transcription application is affected by the network QoS degradation. We employ real Mozilla Common Voice Dataset [12], and transmit voice recordings via established wireless network in the conditions of various packet loss percentage to an end node with the installed DeepSpeech ML system [80]. Using DeepSpeech, we perform recognition and transcription of the transmitted voice and evaluate the ML performance. We empirically study the interrelationship between the employed ML application performance and the network QoS variation. Figure 4.15 details the steps of our study, the employed data and tools, and the POWDER facilities that accommodates the established MLIN cyberinfrastructure.

Employed Voice Recordings Collection

As data samples, we employ a subset of real voice recordings, extracted from the open-source Mozilla Common Voice Dataset [12]. This public dataset contains short voice recordings in various languages and can be employed to train speech-enabled applications. For the purpose of our empirical study, we utilize a subset of 3,995 English language voice recordings, which labels are stored in *.CSV* file. Then, we upload our voice recordings into the data source node, and transmit them in the conditions of the network QoS degradation to the cloud-based ML application. On the receiving node, the transmitted voice is recognized and transcribed by the DeepSpeech model.

Employed ML voice recognition and transcription system

For voice recordings recognition and transcription, we employ DeepSpeech [79], which is a well-known open-source ML-based model designed to transform voice inputs into text outputs. DeepSpeech employs a Recurrent Neural Network (RNN) to perform to perform the ML model training on audio data samples. The learning features are extracted from voice recording spectrograms, into which the recorded speech is converted. The advantages of DeepSpeech are the ability to be trained quickly and efficiently in a parallel structure, and high robustness to a background noise in audio recordings. In our study, we select DeepSpeech to recognize and transcribe voice recordings as it is commonly acknowledged as robust to low quality data. We train the DeepSpeech model on a part of the employed dataset, and test it over the Switchboard Hub5'00 dataset [215]. To evaluate the recognition and transcription performance, we use the Word Error Rate (WER) metric, which is actively employed to evaluate speech-enabled applications' performance. After the training stage,

Table 4.6: Voice recording files average size after transfer with various packet loss rates

Packet Loss Rate (%)	Avg. File Size (kB)
0	146
1	144
5	134
10	123
25	93
50	52

DeepSpeech is able to achieve a 16% WER on the full set.

Employed Network Parameters while the Data Transmission

In this use case, we leverage the established POWDER topology, described in sec. 4.2.1. After the packet loss rule is set, and the speech recordings uploaded to the data source node, the data transfer can commence. This is done using sockets in a Python script, which iterates through all the files in a directory and sends them out one by one. On the receiving end, the node is listening at a specified port for the incoming data, and writes the data into its local memory. Once the transfer is complete for all files, they can be submitted to DeepSpeech for recognition and transcription. After the processing, DeepSpeech provides the performance of the recognition and transcription procedures. This process is iterated with packet loss rates of 1, 5, 10, 25, and 50%. As expected, the size of the dataset decreased with a higher packet loss, which can be attributed to the datagram packets not arriving to the intended destination. Table 4.6 displays the size of the average audio sample size with each tested packet loss rate.

Empirical Results on how DQ Variation Affects ML Voice Recognition and Transcription Performance

As a performance indicator for this scenario, we selected WER indicator as the most common metric used to evaluate speech recognition and transcription performance. This complex indicator takes into account the number of substitutions, deletions, insertions, and correct words that allows for a comprehensive performance evaluation. Equation (4.3) demonstrates how WER is calculated [11].

$$WER = \frac{S + D + I}{S + D + C}, \tag{4.3}$$

where S is the number of substitutions, D is the number of deletions, I is the number of insertions, and C is the number of correct words. This equation provides an error rate metric that can measure how often the ML application incorrectly transcribes a word.

The original subset from the Mozilla Common Voice Dataset was transcribed and the WER was evaluated using the Python library JiWER[9]. The error rate was found to be 24.18%, which is a significant increase from the 16% value presented by Hannun *et al.* [79]. This change could be a result of the Common Voice Dataset being less clean compared to the Switchboard Hub 5'00 dataset. The calculated WER was also normalized to ensure that sentences carried their proportional weight, meaning that the WER for longer sentences carried more weight than the one for shorter sentences. This is done by multiplying the WER of a given audio sample by the number of words in the sample, then dividing by the total number of words that make up the 3,995 file dataset used. This value is then added to the total WER variable, which represents the WER of the entire dataset after all the transcriptions are analyzed. For reference, the total number of words in this dataset adds up to 37,853 words, which constitutes 9.48 words per audio sample on average.

The DeepSpeech documentation recommends using audio samples of around 15 seconds. Many of the audio samples in the Common Voice Dataset were shorter than this value. However, an additional test was done, where the performance of DeepSpeech was tested only on the longest sentences, and the cumulative WER was not differ significantly. We can therefore conclude that sample length was not the reason for the higher WER compared to the study by Hannun *et al.* [79].

The empirical study results can be seen in Figure 4.16, which illustrates the calculated accuracy at each packet loss rate imposed, which was calculated as $100 - WER$. As one can see from Figure 4.16, the accuracy has a non-linear relationship with packet loss rate. In fact, throughout the range of packet loss rates, it can be seen that an increase in packet loss causes a disproportional increase in the WER, indicating that small rates of loss could result in large errors being made by the classifier. Also, the obtained results indicate that packet loss could cause a significant detriment to speech transcription accuracy, with a 5% loss rate causing an approximate 21% decrease in performance that should make its application not acceptable in most domains.

[9] `https://pypi.org/project/jiwer/`

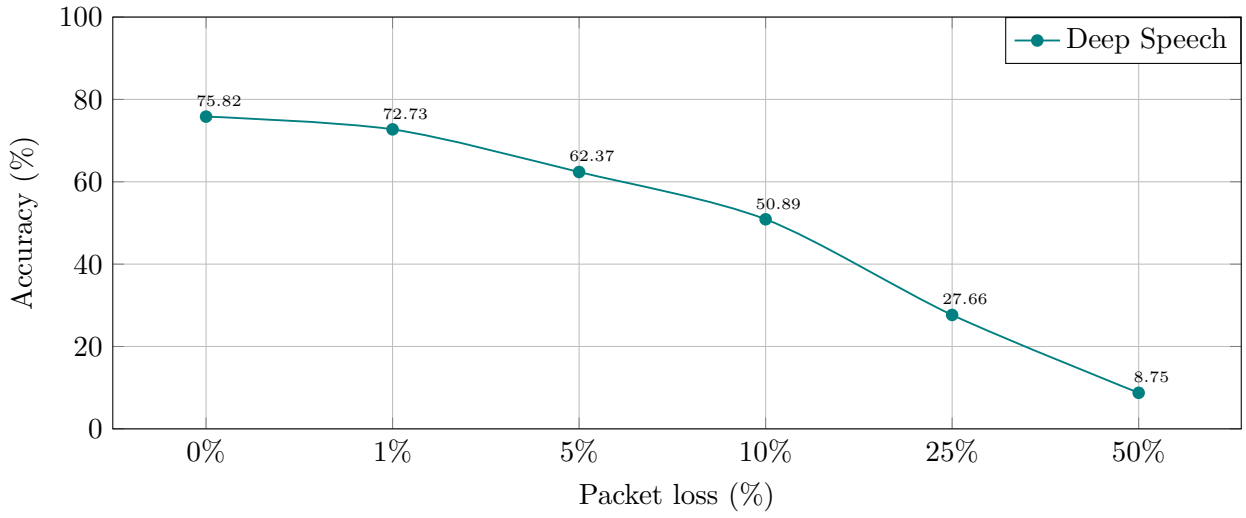


Figure 4.16: Packet loss influence on DeepSpeech accuracy (with the buffer size of 1024B)

4.2.6 Object Detection and Classification in Videos Practical Use Case

In this section, we investigate the impact of network disruptions on the performance of video object detection and classification. In our previous real-world use cases, our examination was limited to only open-source and publicly available ML systems. In this use case, we investigate the performance of a commercial Amazon Web Service (AWS) Rekognition[10] platform, which is a popular image and video analysis tool. For this analysis, we transfer a collection of video files between the two nodes using the real wireless network. We investigate various network degradation conditions while the data transmission by varying network packet loss rates and receiver buffer size. We then upload the obtained video files with the varied DQ to AWS Rekognition to evaluate how the varied network conditions affect the video object detection and classification performance. We analyze the obtained results and derive knowledge on the observed interrelationships. Figure 4.17 represents our investigation steps, and discloses data, tools, and facilities employed to study the interrelationships between the input DQ and ML video object detection and classification performance.

[10] `https://aws.amazon.com/rekognition/`

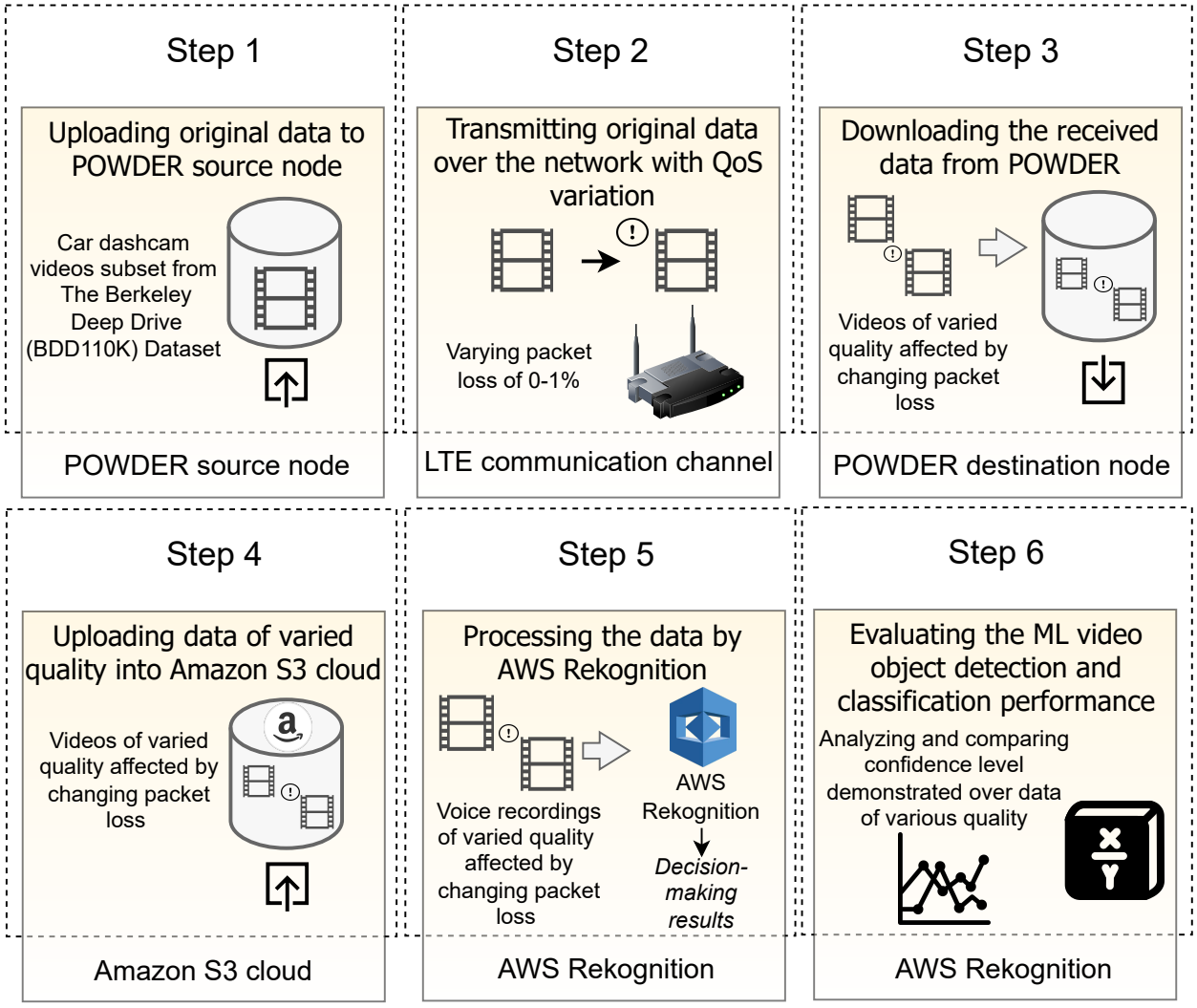


Figure 4.17: Schematical representation of the steps employed in the investigated object detection and classification in videos real-world use case involving varied DQ due to changing network packet loss

Employed Video Data Collection

To investigate the effects of network degradation on video object detection and classification performance, we employ the The Berkeley Deep Drive (BDD110K) Dataset[11]. This is a large, diverse, crowd-sourced video data collection containing over 100,000 videos featuring various scene types such as city streets, residential areas, and highways in varying weather conditions recorded at different times of the day. To facilitate our examination, we manually selected a set of 35 high-quality videos from the employed collection, in which all the patterns of interest (traffic lights and road signs) are clearly visible for a reasonable amount of time. As each video contained a traffic light or a road sign interesting patterns, we divided the selected files into two categories: 25 videos with the visible traffic lights pattern; and 10 videos with the visible road sign pattern. We also made sure that AWS Rekognition can clearly recognize the objects of interest, hence we uploaded the selected videos to the Rekognition for the preliminary testing first. For each of the selected videos, Rekognition demonstrated a confidence score of greater than 90% for each detected pattern of interest. All the selected videos are 8-10 seconds in length and converted to the H.264 file format upon the network transmission to enable the reconstruction and execution of the corrupted video files.

Employed ML Video Object Detection and Classification System

AWS Rekognition allows the use of pre-trained black-box ML models for various ML tasks, such as celebrity facial recognition, text detection, and generic object detection. As the service is provided to the user on the commercial basis, AWS Rekognition does not provide any specifics on which ML model architectures are employed for the object detection and classification services. Specifically for the object detection service, Rekognition provides "Detect Labels" API endpoint. This service returns a list of objects that have been detected in an uploaded image. The response contains confidence scores for each of the objects detected in the submitted data. If the confidence score is high enough, AWS also provides a bounding box with the detected object location in the image. AWS Rekognition is employed by various business organizations operating in such diverse areas, as e-commerce, education, finance, security, transportation, and others[12].

[11] `https://bdd-data.berkeley.edu/`

[12] `https://aws.amazon.com/rekognition/customers/`

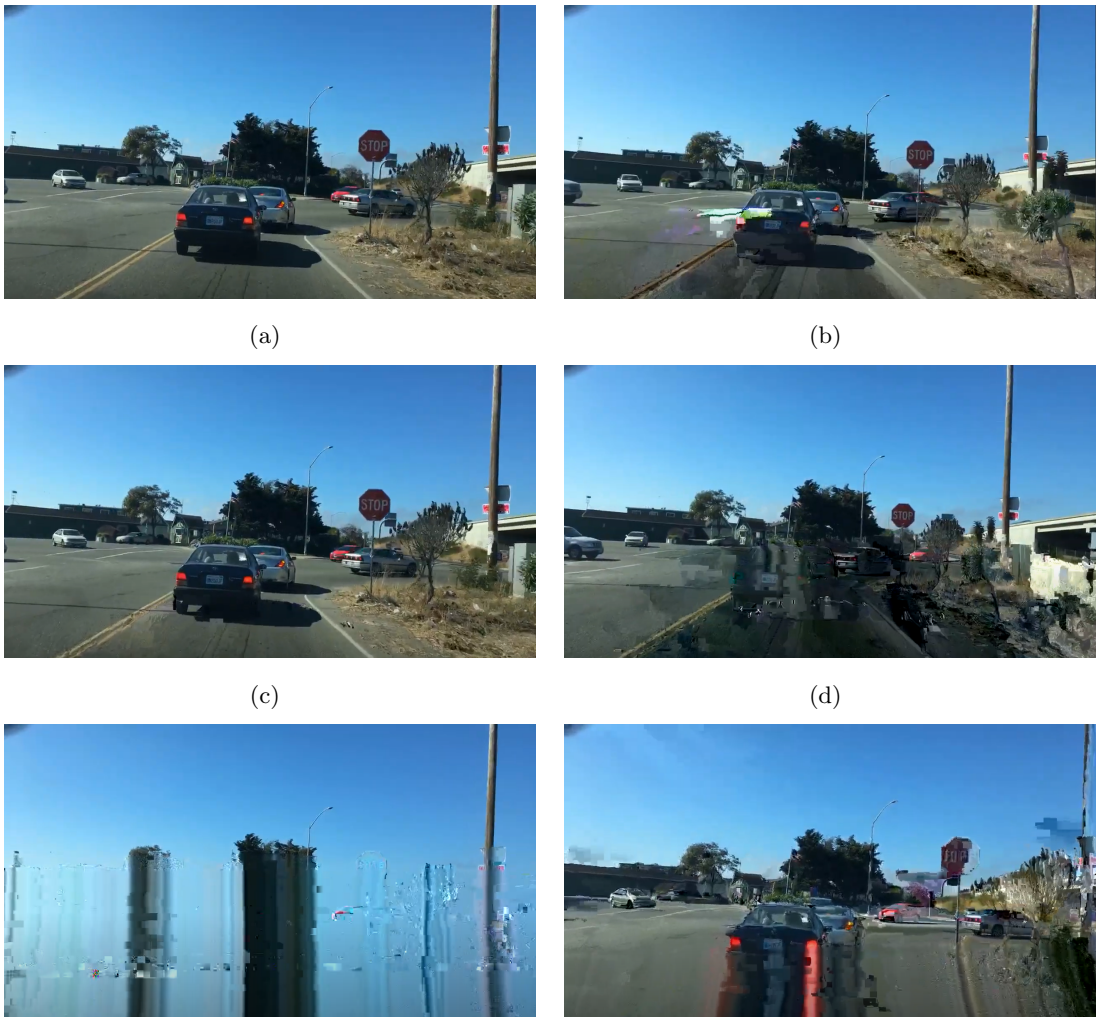
Employed Network Parameters while the Data Transmission

In this real-world use case, we also employ the POWDER network topology we described in sec. 4.2.1. In contrast to the previous investigated cases, in the present one we observe that even 1% packet loss affect the video quality significantly and results in too severe corruptions. Some screenshots from the corrupted videos are represented in Figures 4.18 and 4.19 compared with their original version. As one can see, the patterns of interest are hardly visible in the case of 1% and higher packet loss rates. Hence, in this use case, we concentrated on varying the packet loss rate in the range from 0.1 to 1%. As another network QoS factors, we also investigated the two sizes of the receiver's node buffer: 512B and 1024B. Figures 4.18 and 4.19 represent some examples of the frames from the road sign category video, transmitted over the network in the varied QoS conditions.

Empirical Results on how DQ Variation Affects ML Video Object Detection and Classification

Upon uploading the video to AWS Rekognition, it analyses each frame and uses various proprietary ML models and techniques to detect and identify the objects in the video frame. It assigns labels for the objects detected in the video and outputs a confidence score for each object. This confidence score represents the degree of certainty with which the detected object is present in the frame. In our empirical study, we recorded the minimum, maximum, average and range of the confidence values provided by Rekognition for each of the uploaded video.

Table 4.7 represents the AWS Rekognition performance for the videos with the traffic light pattern. As can be seen from the results, higher packet losses result in the ML performance drops. Additionally, one can observe that, in the majority of cases, larger buffer slightly compensated the ML performance decrease, as less packets are dropped due to the receiver can use more resources to store the incoming packets. Table 4.8 represents the Rekognition performance demonstrated over the videos with the road sign pattern. As one can see, in this case the results demonstrate similar general trend of decreasing ML performance with the growing amount of packets lost. Similarly to the videos with traffic light pattern, the increase in buffer size also helps to mitigate higher ML performance degradation in the majority of experiments. The obtained results clearly demonstrate that even minimal packet loss of 0.1%, deemed acceptable for the most applications, introduced significant degradation in the quality of the transmitted videos, and resulted in substantial ML application performance decrease.



(a) (b)

(c) (d)

(e) (f)

Figure 4.18: Examples of the frames from the road sign category video transmitted over a wireless network with 512B receiver buffer size and with packet loss rate of: (a) – 0%; (b) – 0.15%; (c) – 0.25%; (d) – 0.5%; (e) – 0.75%; (f) – 1%

Figure 4.19: Examples of the frames from the road sign category video transmitted over a wireless network with 1024B receiver buffer size and with packet loss rate of: (a) – 0%; (b) – 0.15%; (c) – 0.25%; (d) – 0.5%; (e) – 0.75%; (f) – 1%

Table 4.7: Variation in the average of the confidence score statistics of the Traffic Light Label for multiple packet loss values and buffer sizes

Packet Loss (%)	**Confidence Score (%)**					
	512 B			1024 B		
	Min	Max	Average	Min	Max	Average
0	56.136	97.952	85.18	56.136	97.952	85.18
0.1	56.404	95.752	82.196	55.096	96.936	80.888
0.15	54.02	89.88	77	52.16	92.296	77.536
0.2	53.916	90.568	77.068	53.796	92.756	78.916
0.25	54.228	91.3	76.252	57.42	94.288	79.872
0.3	53.964	86.88	74.336	56.588	93.608	79.528
0.5	56.432	89.6	76.436	50.244	80.28	67.748
0.75	51.448	71.368	64.068	54.496	84.5	74.476
1	45.144	64.016	56.088	51.06	81.164	68.388

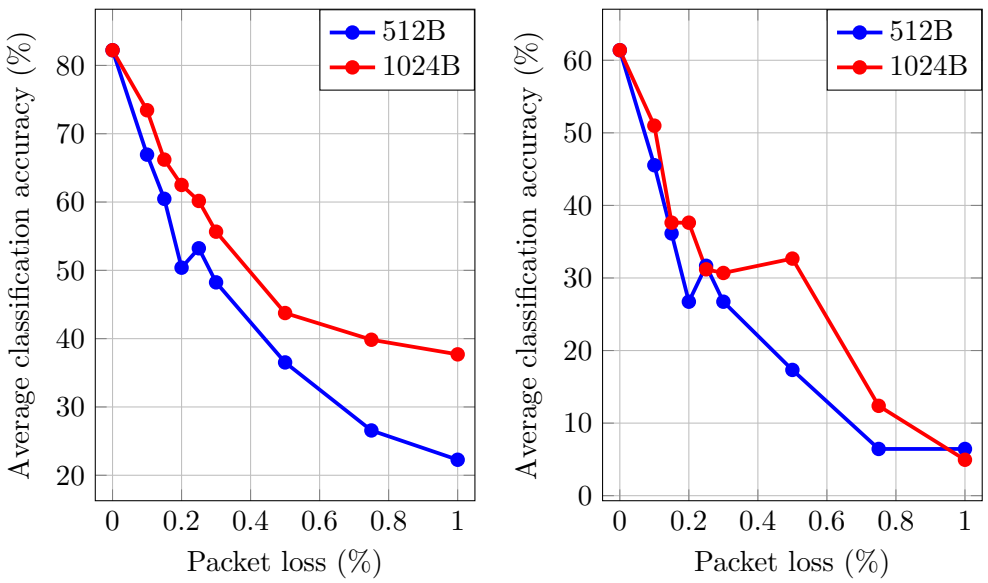
Due to the inherently random nature of packet loss, the visual quality of videos affected by the same packet loss rate may vary. This means that even if we transmit the same video with a similar packet loss rate, we might result in various number of frames affected by the corruption in different locations and with divergent intensity. This results in various confidence scores, demonstrated by the Rekognition. Therefore, we also obtained the average classification accuracy demonstrated by Rekognition. We calculated this accuracy using the number of frames, in which Rekognition correctly detected a pattern of interest over the total number of frames for every video in each pattern category. For both pattern categories, a decline in the classification accuracy (or an increase in the number of missed classifications) is clearly visible with an increase in packet loss for both buffer sizes, as displayed in Figure 4.20(a) and Figure 4.20(b) for the traffic light and road sign categories respectively.

According to the results demonstrated by AWS Rekognition, a slight increase in packet loss leads to a massive drop in the confidence score and classification accuracy. The ML application's tolerance to network QoS variations generally increases when more resources are employed at the receiver. In our practical use case, a packet loss of 0.5% dropped ML application performance almost by a half, despite that in many other applications such packet loss rate is considered as negligible[13]. Packet loss higher than 1% made the transmitted videos unusable for the analysis by AWS Rekognition,

[13] `https://www.techtarget.com/searchnetworking/definition/packet-loss`

Table 4.8: Influence of packet loss and buffer size on AWS Rekognition Confidence Score for Traffic Light Label

Packet Loss (%)	**Confidence Score (%)**					
	512 B			1024 B		
	Min	Max	Average	Min	Max	Average
0	56.54	93.34	79.08	56.54	93.34	79.08
0.1	53.61	84.65	71.11	54.52	88.81	75.14
0.15	50.5	75.05	64.54	52.31	79.64	68
0.2	46.6	70.71	59.15	51.66	78.31	68.24
0.25	50.45	74.55	64.57	38.11	61.54	52.37
0.3	56.27	80.3	68.72	43.17	65.1	56.71
0.5	43.05	56.18	49.9	44.43	69.25	59.42
0.75	34.99	44.53	40.66	38.24	47.5	43.7
1	39.37	44.2	41.97	29.26	33.35	31.87



(a) (b)

Figure 4.20: Classification accuracy demonstrated by AWS Rekognition over the videos with various pattern of interest: (a) – traffic lights; (b) – road signs

as they became too corrupted and no patterns of interest left visible. Hence, we can recommend to stop using UDP in the event of even small packet loss of 0.1%, and switch to a more reliable protocol, such as HLS or RTMP, if the latency, jitter, and other network requirements are satisfied.

4.3 Feedback from the ML Application to the MLIN Components. Practical Example

In this section, we demonstrate a practical illustration of the MLIN adjustment feedback component, previously introduced in sec. 2.3, using the example of rules for changing network parameters while the data transmission. In the previous sections, we investigated the interrelationships between the input DQ and ML application performance across diverse real-world use cases. Our investigations encompassed a wide spectrum of ML tasks, including classification of various image types (traffic signs and medical images), sound classification, voice recognition and transcription, as well as video object detection and classification. We evaluated how varying MLIN cyberinfrastructure conditions, such as network QoS variation, and sensor failures and errors can influence the performance of ML applications. The practical results we obtained allowed us to derive knowledge on the integral structure and features of MLIN that affect the DQ on various data life-cycle steps.

Throughout our practical investigations, we studied data loss, cyberinfrastructure resource insufficiency, and failures and errors that result in DQ degradation as the major DQ variation aspects. In all examination use cases, we observed that the decline in input DQ had a direct impact on the ML application performance, and unequivocally resulted in its decrease. This empirical evidence emphasizes a direct connection between the input DQ and the performance of ML applications. As one could observe, the more degradation was experienced by the input DQ, the larger impact it had on the ML application performance drop, the extent of which varied across the considered applications.

Elaborating on the obtained practical results, we can bridge the derived knowledge on the investigated interrelationships with our MLIN adjustment feedback approach to enhance ML application robustness, developed and presented in sec. 2.3.5. In sec. 2.6, we defined ML robustness as the relationship between the input DQ and the performance of ML application demonstrated over this input. As we showed in practice, the drop in input DQ results in ML application performance degradation, and according to (2.16), also leads to ML robustness decrease. Hence, to improve ML robustness to input DQ variations, we need to prevent the ML application performance decrease due to these variations.

The particular MLIN feedback actions depend on the interrelationships between MLIN components within the particular practical scenarios. According to sec. 2.3, the feedback mechanism itself may incorporate diverse mechanisms and methods, such as rule-based systems or sophisticated intelligent mechanisms offering recommendations for appropriate adjustment actions. In our illustrative instance, we focus on switching network protocols in the case of network disruptions as the primary measure to prevent DQ variation while transmitting the data over the network İnitially, we conduct an empirical study to analyze which network protocols provide better service in various network conditions. Then, based on the knowledge derived on the interrelationships between the input DQ and the ML application performance, and the network protocols practical evaluation results, we develop our rules for MLIN adjustment feedback actions. Through this example, we represent how our MLIN adjustment feedback component may be designed in practice, and show how it is integrated into the MLIN integral structure.

4.3.1 Network Adjustment for ML Image Classification Application Use Case

Based on our practical use cases examination, we propose a novel generic approach to improve MLIN robustness against network QoS degradation. Existing methods propose to adjust network based on the network conditions monitoring mechanisms (e.g., [27, 88, 128]). In contrast to other approaches, we employ knowledge on the interrelations between network QoS and ML end performance to recommend network parameters modification. The generic structure of our dynamic network adjustment system is given in Figure 4.21. Below we describe examples of our approach implementation.

Suggested Improvements to Network QoS

In our real-world use cases, we showed that network disruptions such as packet loss lead to the DQ variations and ML application performance degradation. Since many modern smart technologies rely on ML systems running on cloud or remote servers, it is important to improve network reliability in the cases of packet loss spikes. The outcome of packet losses depends on the transport protocol used for the network communication. Since UDP protocol is unreliable by its design, packet loss leads to irreversible data loss as it is shown in our use cases. TCP protocol preserves DQ over transmission, however, it significantly increases data transfer time. One solution to the problem of DQ degradation prevention during network disruptions might be switching to transport protocols alternative to TCP or UDP.

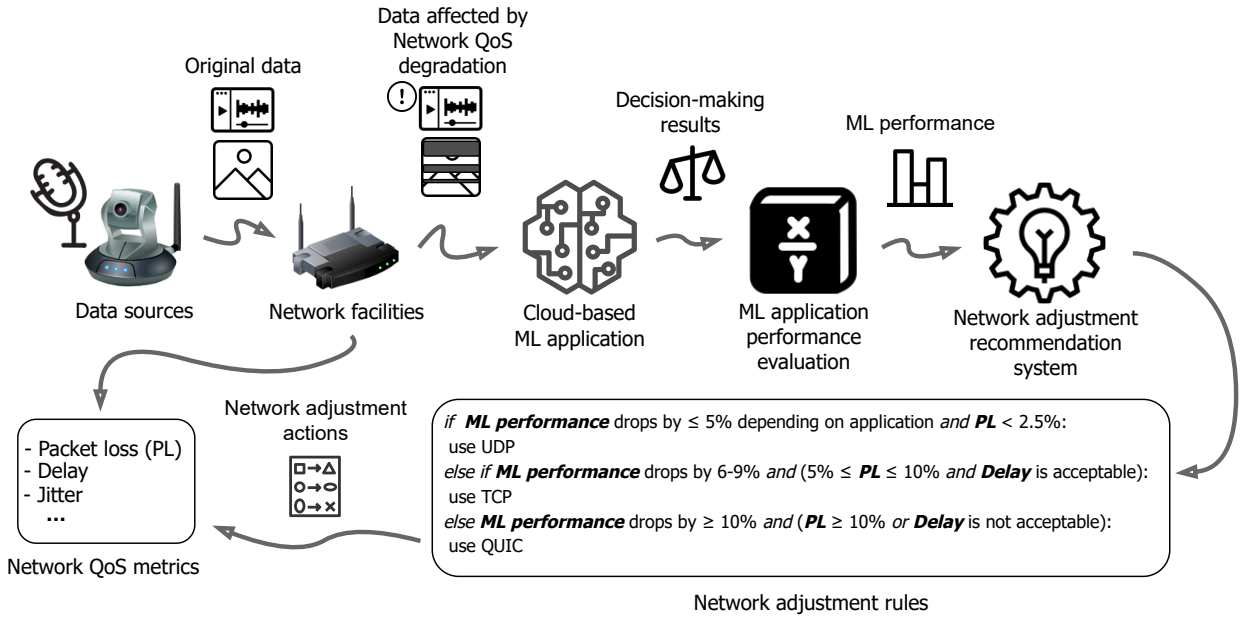
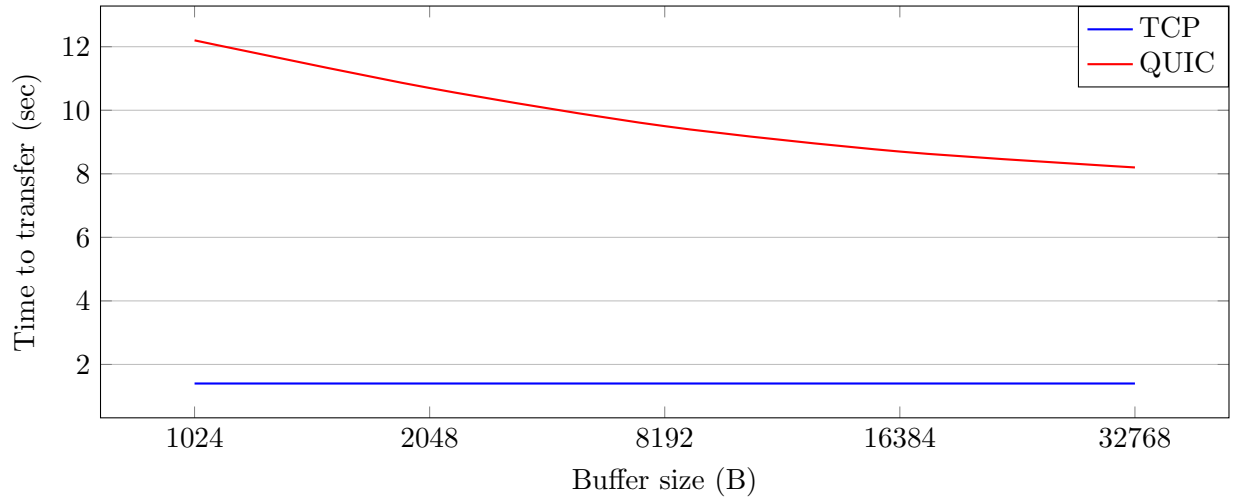


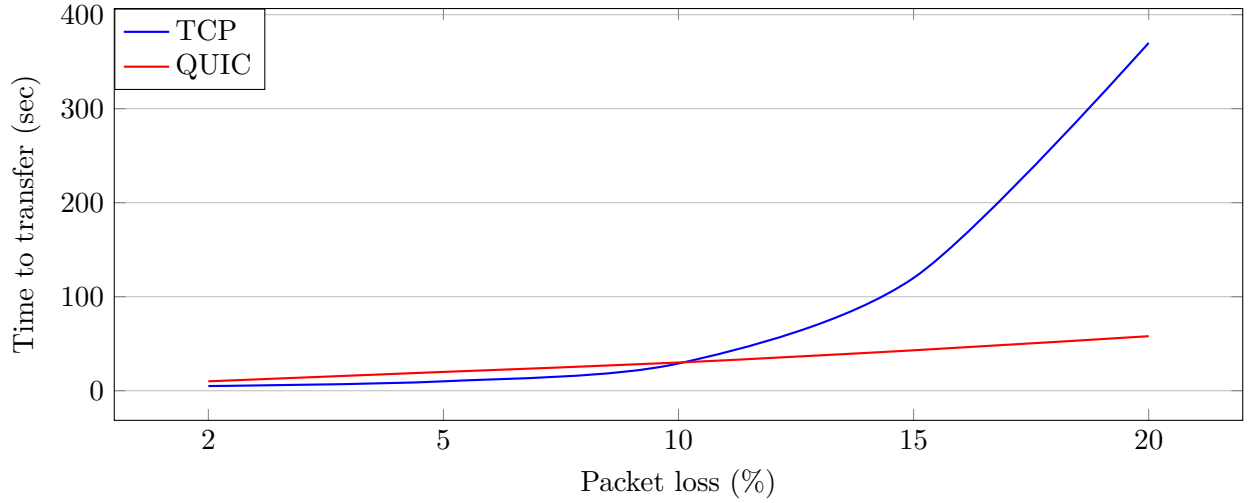
Figure 4.21: An example of the dynamic network adjustment system design. The feedback on network adjustment actions is provided based the user and application requirements, which consider ML application performance as the major indicator

Specifically, we investigate QUIC protocol [121] designed by Google to reduce latency compared to TCP, while providing reliable data delivery service. To compare how TCP and QUIC react on varying network conditions, we perform wireless data transfer when using each of these protocols leveraging the experimental wireless POWDER infrastructure [23]. In our major experiments, we transmit image files with varying either the packet loss rate or the buffer size due to different QoS conditions (see sec. 4.2). Below we present the results of our evaluation.

Varying Buffer Size with the Minimum Packet Loss By keeping the network packet loss to the minimum, we vary the buffer size to get a sense of how data transfer speed is affected by a packet size. Figure 4.22(a) shows the time taken to transfer for TCP and QUIC while the buffer size is varied from 1024 up to 32768B. In these experiments, TCP outperforms QUIC and maintains a steady transfer time regardless of buffer size. On the other hand, QUIC shows a dramatic improvement as the buffer size is increased – a drop of 12.3 seconds to 7.7 seconds indicates an approximate performance increase of 40%, suggesting that QUIC is more sensitive to buffer size changes than TCP. However, this case investigates the small packet loss rate, and under such conditions we recommend to use UDP since it outperforms both QUIC and TCP in terms of the



(a)



(b)

Figure 4.22: TCP and QUIC experimental results comparison: Transfer time change with (a) - varying buffer size for TCP and QUIC with minimum packet loss; (b) - varying packet loss for TCP and QUIC with the maximum buffer size of 32768B

data delivery time. We can confirm that if the UDP application does not result in a notable packet loss, UDP protocol should be employed as usual for media services.

Varying Packet Loss with the Constant Buffer Size In these experiments, we compare the transfer time of the transmissions using TCP and QUIC protocols under the same technological conditions with the fixed buffer size (the maximum size of 32768B is chosen, so that buffer size is not the bottleneck in the experiments). Figure 4.22(b) shows the time taken to transfer for two protocols when the packet loss rate changes in the range from 2 to 20%. The results demonstrate that TCP outperforms QUIC under near perfect network conditions when the packet loss rate is relatively low. However, as the packet loss rate increases above 10%, the time taken to transfer data for TCP increases significantly, while the transfer time with QUIC scales almost linearly. As we show, this difference enables QUIC to outperform TCP under network QoS conditions with packet loss rates above 10%.

To summarize, under network conditions with high packet loss ratio ($> 10\%$) and QoS degradation, utilizing QUIC as the transport protocol may improve QoS and deliver data to the destination faster and with better quality that in turn results in improving the performance of the end ML-based classifiers that utilize the delivered data. However, under good network QoS conditions, when the packet loss rate is below 10%, we recommend using UDP as a transport protocol for the remote ML classification. Leveraging the results of our protocols' evaluation, and the knowledge obtained in the traffic sign images classification use case, presented in sec. 4.2.2, we developed an example of the rule-based system presented in Fig. 4.21, which can effectively mitigate the drops in ML performance caused by DQ variations due to changes in network QoS. This eventually allows to improve ML application robustness to these variations.

4.4 Demonstrating how MLIN Cyberinfrastructure Affects ML Robustness

Earlier, in sec. 2.6, we introduced the definition and calculus of ML robustness. In contrast to other approaches to ML robustness, we designed it to represent the relationship between the input DQ and the ML performance demonstrated over this input. This approach allows obtaining a quantitative measure that represents how the variation in the input DQ impacts the performance of ML applications. However, as we discussed in sec. 3.1, considering the existing wide variety of data modalities, structures, and representations, DQ metrics used in practice highly depend

on the user and application requirements, and may be measured in different ways. In this case, the ML robustness evaluation may also vary and include additional steps or indicators added up to its generic calculus to satisfy the user requirements. For example, robustness for the ML application can be calculated for each separate data input (e.g., local robustness [35]) or can represent an aggregated value based on all processed data inputs over the operation time (e.g., global robustness [35]).

In this section, we adapt our generic ML robustness calculus for the practical implementation and demonstrate how ML robustness can be calculated in the specific real-world scenario. In particular, we concentrate on the ML sound classification application. We leverage our results on ML sound classification performance over the data affected by network disruptions, obtained in sec. 4.2.4, and employ them to design the ML robustness calculation function. In this practical example, we calculate the DQ value based on the percentage of packets lost while transmitting the input data sample over the network. Below we represent the details of calculating ML robustness in our practical use case.

Practical ML Robustness Calculation Example

According to equation (2.16), we basically need two parameters to calculate ML robustness ($\mathcal{RB}$): DQ of the data input ($\mathcal{DQ}$), and the performance demonstrated by the ML application over this data input ($\mathcal{PRF}$). In this practical use case, we leverage the experimental results on the ML sound classification performance, represented in Table 4.5. This Table showcases the performance demonstrated by the ML application over the data samples of various quality, corrupted by the network packet loss. In our case, we first need to define and calculate the $\mathcal{DQ}$ value, which we describe below.

$\mathcal{DQ}$ Calculation for our Practical Use Case In this practical use case, we employ the packet loss rate as the major DQ characteristic that represent the percentage of data lost during the network transmission.

Let us define the $\mathcal{DQ}$ in the interval between 0 and 1, where 0 corresponds to the lowest possible quality value, and 1 to the highest possible one.

Let us assume that in our practical use case, $\mathcal{DQ}$ is determined by the amount of data lost while the network transmission. Hence, a single data sample transmitted over the network without any

packet losses corresponds to $\mathcal{DQ}$ of 1. Following the definitions introduced, 5% packet loss while transmitting a single sample over the network results in $\mathcal{DQ} = 0.95$ for this sample, 20% packet loss results in $\mathcal{DQ} = 0.8$, and etc.

In sec. 4.2.4, we investigated packet loss rates in the interval between 0 and 80%: 0, 10, 20, 50, 60, 70, and 80%. In Table 4.9, we represent the $\mathcal{DQ}$ values calculated for each packet loss rate, represented in this Table.

$\mathcal{RB}$ Calculation for our Practical Use Case Let us consider 7 distinct time moments $t_i \in T, i \in \{1, \ldots, 7\}$ of the considered MLIN system operation, where t_1 is the initial step, and t_7 is the final step. In this practical example, we assume that in each consequent time moment t, ML application receives and processes a new data sample, transmitted over a network. In our case, in each time moment t packet loss varies, and hence, $\mathcal{DQ}$ of the transmitted data varies as well. We assume that the ML performance evaluation is conducted by the ML application and the result is provided by the system for each consequent time moment.

As we mentioned above, the $\mathcal{RB}$ indicator can be calculated locally, for each distinct time moment t, or globally, which means that it integrates all $\mathcal{RB}$ values demonstrated by ML application during the MLIN operation. In our example, we demonstrate the calculation of both local and global $\mathcal{RB}$ indicators. We denote them as $\mathcal{RB}_L$ and $\mathcal{RB}_G$, and show how they are calculated for time moment t_i in equations (4.4) and (4.5) respectively.

$$\mathcal{RB}_{L_{t_i}} = \begin{cases} \dfrac{\Delta\mathcal{DQ}_{t_i}}{\Delta\mathcal{PRF}_{t_i}}, & \text{if } \Delta\mathcal{PRF}_{t_i} \neq 0, \\ \mathcal{RB}_{L_{t_i}} \to max, & \text{if } \Delta\mathcal{PRF}_{t_i} = 0,\ \Delta\mathcal{DQ}_{t_i} < 0, \\ \mathcal{RB}_{L_{t_i}} \to min, & \text{if } \Delta\mathcal{PRF}_{t_i} = 0,\ \Delta\mathcal{DQ}_{t_i} > 0, \end{cases} \tag{4.4}$$

where $\Delta\mathcal{DQ}_{t_i}$ and $\Delta\mathcal{PRF}_{t_i}$ are changes in $\mathcal{DQ}$ and $\mathcal{PRF}$ values between t_{i-1} and t_i time moments.

$$\mathcal{RB}_{G_{t_i}} = \frac{\sum_{i=0}^{|T|}(\mathcal{DQ}_{t_i} - \overline{\mathcal{DQ}})(\mathcal{PRF}_{t_i} - \overline{\mathcal{PRF}})}{\sum_{i=0}^{|T|}(\mathcal{DQ}_{t_i} - \overline{\mathcal{DQ}})}, \tag{4.5}$$

where $|T|$ is a number of the considered time moments t; $\overline{\mathcal{DQ}}$ and $\overline{\mathcal{PRF}}$ are mean values for $\mathcal{DQ}$ and $\mathcal{PRF}$ demonstrated by the system over the all considered time moments t.

For our calculation example, we employ ML application performance values demonstrated by

ResNet50 model, and represented in Table 4.5. We convert these values to represent them in the interval between 0 and 1 for the $\mathcal{RB}$ calculation convenience, as values for $\mathcal{DQ}$ has a similar range.

In Tables 4.9, 4.10, and 4.11, we demonstrate our calculation example for the introduced local and global ML robustness notations for the considered ML sound classification use case. We denote t_1 as the first time moment after the system's initialization, when ML application is expected to receive and process the first input data sample. After the initialization, we consider the scenarios of gradual packet loss increase, decrease, and deviation.

As one can see from the results, both global and local RB indicators reflect the interrelationship between $\mathcal{DQ}$ and $\mathcal{PRF}$. However, $\mathcal{RB}_L$ and $\mathcal{RB}_G$ enable to analyze this interrelationship in various manners. $\mathcal{RB}_L$ encompasses only the local changes in $\mathcal{DQ}$ and $\mathcal{PRF}$ between the current and previous time moments. This indicator is very sensitive to the changes in $\mathcal{DQ}$ and $\mathcal{PRF}$, which can be observed in t_4 and t_5 time moments in Table 4.9. In t_4, the $\mathcal{DQ}$ value change was -0.3, and the relative $\mathcal{PRF}$ value change was only -0.0052 in comparison to the previous time moment. In this case, a significant change in $\mathcal{DQ}$ from 0.8 to 0.5 resulted in a relatively small $\mathcal{PRF}$ degradation, which meant that ML application demonstrated high $\mathcal{RB}_L$ of 57.69 to the $\mathcal{DQ}$ variation. In the next time moment t_5, the $\mathcal{RB}_L$ value dropped to 9.71, as a relatively small change in $\mathcal{DQ}$ of -0.01 resulted in a $\mathcal{PRF}$ decrease of -0.0103. Figure 4.23(b) represents how the $\mathcal{RB}_L$ and $\mathcal{RB}_G$ values change throughout the system operation.

In comparison to the $\mathcal{RB}_L$ indicator, $\mathcal{RB}_G$ allows taking into account the dynamic of how the interrelationship between $\mathcal{DQ}$ and $\mathcal{PRF}$ changes over the MLIN operation time. As one can see from Table 4.9, $\mathcal{RB}_G$ in essence integrates all the changes in $\mathcal{DQ}$ and $\mathcal{PRF}$ into a single indicator that represents the trend demonstrated by the MLIN system from its initialization to the current time moment. As one can see, the $\mathcal{RB}_G$ indicator is much less sensitive to the changes in $\mathcal{DQ}$ and $\mathcal{PRF}$ than $\mathcal{RB}_L$, and can be employed for the more comprehensive time series analysis.

Table 4.10 and Figure 4.24(b) represent the $\mathcal{RB}_L$ and $\mathcal{RB}_G$ indicators' values calculated for the increasing $\mathcal{DQ}$ case. As one can see, the values for the $\mathcal{DQ}$ and $\mathcal{PRF}$ are just sorted in a different order to recreate the consequent $\mathcal{DQ}$ increase case. As in the previous case, $\mathcal{RB}_G$ experience less deviations to the changing relationship between $\mathcal{DQ}$ and $\mathcal{PRF}$, and allows to better represent it on the global time-scale. The difference between $\mathcal{RB}_L$ and $\mathcal{RB}_G$ can be clearly observed in the moment of transition from time moment t_4 to t_5, as the indicators' values behave in a completely different way. $\mathcal{RB}_L$ shows a rapid growth, as the $\mathcal{DQ}$ also experiences increase while maintaining the $\mathcal{PRF}$ slowly growing trend. In contrast, $\mathcal{RB}_G$ decreases on the same interval, as rapid growth

Table 4.9: ML application robustness calculation practical example for DQ decreasing case

Time moments	**Packet loss**	$\mathcal{DQ}$	PRF	$\Delta\mathcal{DQ}$	$\Delta\mathcal{PRF}$	$\mathcal{RB}_L$	$\mathcal{RB}_G$
t_1	0%	1	0.95	1	0.95	–	–
t_2	10%	0.9	0.88	-0.1	-0.07	1.43	0.7
t_3	20%	0.8	0.7755	-0.1	-0.1045	0.96	0.87
t_4	50%	0.5	0.7703	-0.3	-0.0052	57.69	0.34
t_5	60%	0.4	0.76	-0.1	-0.0103	9.71	0.27
t_6	70%	0.3	0.68	-0.1	-0.08	1.25	0.30
t_7	80%	0.2	0.67	-0.1	-0.01	10	0.29

Table 4.10: ML application robustness calculation practical example for DQ increasing case

Time moments	**Packet loss**	$\mathcal{DQ}$	PRF	$\Delta\mathcal{DQ}$	$\Delta\mathcal{PRF}$	$\mathcal{RB}_L$	$\mathcal{RB}_G$
t_1	80%	0.2	0.67	–	–	–	–
t_2	70%	0.3	0.68	0.1	0.01	10	0.1
t_3	60%	0.4	0.76	0.1	0.08	1.25	0.45
t_4	50%	0.5	0.7703	0.1	0.0103	9.71	0.38
t_5	20%	0.8	0.7755	0.3	0.0052	57.69	0.18
t_6	10%	0.9	0.88	0.1	0.1045	0.96	0.25
t_7	0%	1	0.95	0.1	0.07	1.43	0.3

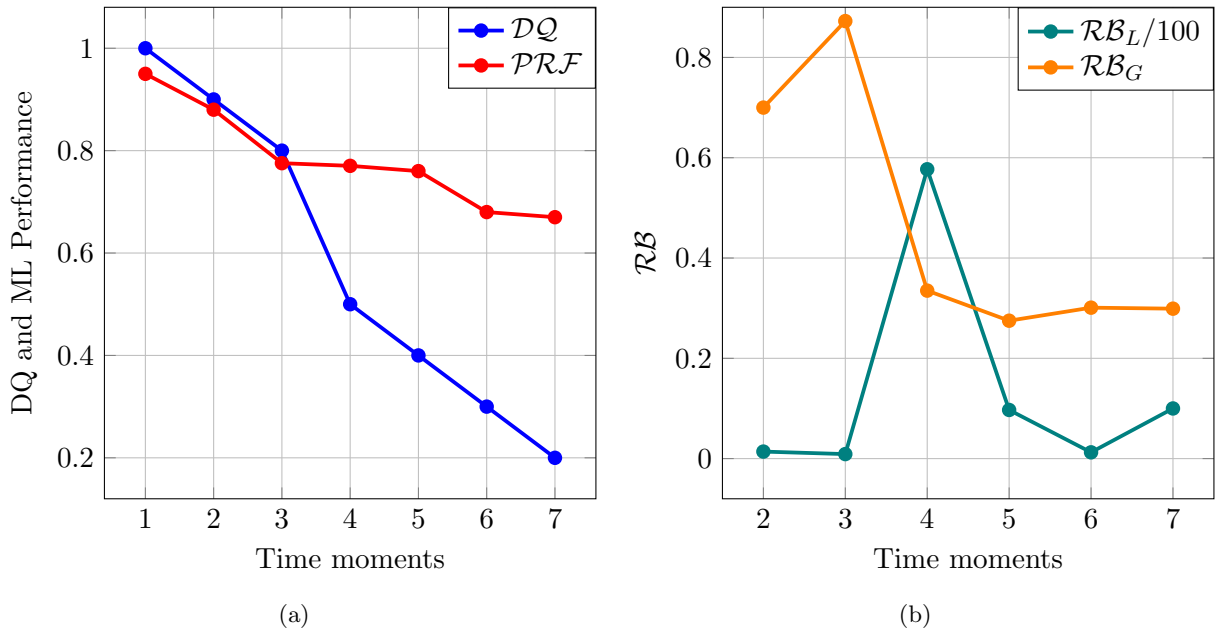
of $\mathcal{DQ}$ does not invoke the similar $\mathcal{PRF}$ trend.

Table 4.11 and Figure 4.25(b) refer to the case with $\mathcal{DQ}$ dynamically increasing and decreasing throughout the system operation. In this case, from Figure 4.25(a), one can see that despite the larger deviations in $\mathcal{DQ}$, the changes in $\mathcal{PRF}$ are less severe. Since we can see sudden $\mathcal{DQ}$ drop and recovery, and similar but less severe changes in $\mathcal{PRF}$, both $\mathcal{RB}_L$ and $\mathcal{RB}_G$ demonstrate stable behavior during the whole system operation period. This case demonstrates that when the system is able to maintain stable robustness over the operation period, it means that the performance is not subject to change a lot under the conditions of significant DQ variations.

In this ML robustness practical calculation example, we demonstrated how ML robustness can be affected by the dynamic changes in MLIN cyberinfrastructure, and developed calculus that enables quantifying this impact. We showed how the $\mathcal{RB}$ indicator can be defined and calculated based on

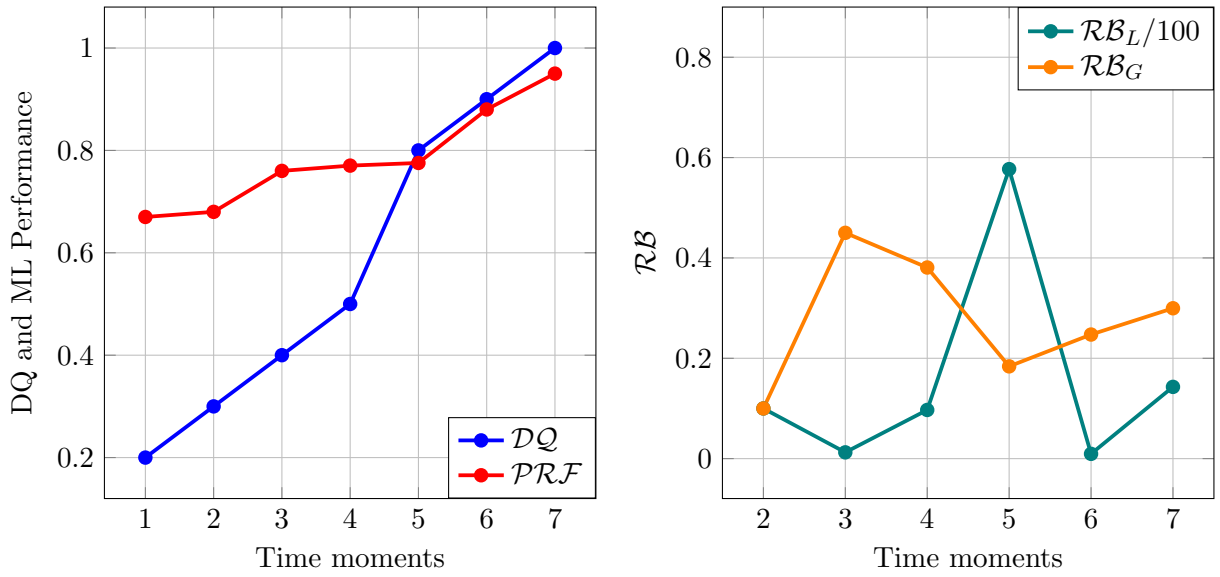
Table 4.11: ML application robustness calculation practical example for varying DQ case

Time moments	**Packet loss**	$\mathcal{DQ}$	PRF	$\Delta\mathcal{DQ}$	$\Delta\mathcal{PRF}$	$\mathcal{RB}_L$	$\mathcal{RB}_G$
t_1	20%	0.8	0.7755	–	–	–	–
t_2	80%	0.2	0.67	-0.6	-0.11	5.69	0.17
t_3	50%	0.5	0.7770	0.3	0.10	2.99	0.17
t_4	10%	0.9	0.88	0.4	0.11	3.64	0.24
t_5	0%	1	0.95	0.1	0.07	1.43	0.30
t_6	60%	0.4	0.76	-0.6	-0.19	3.16	0.29
t_7	70%	0.3	0.68	-0.1	-0.08	1.25	0.3



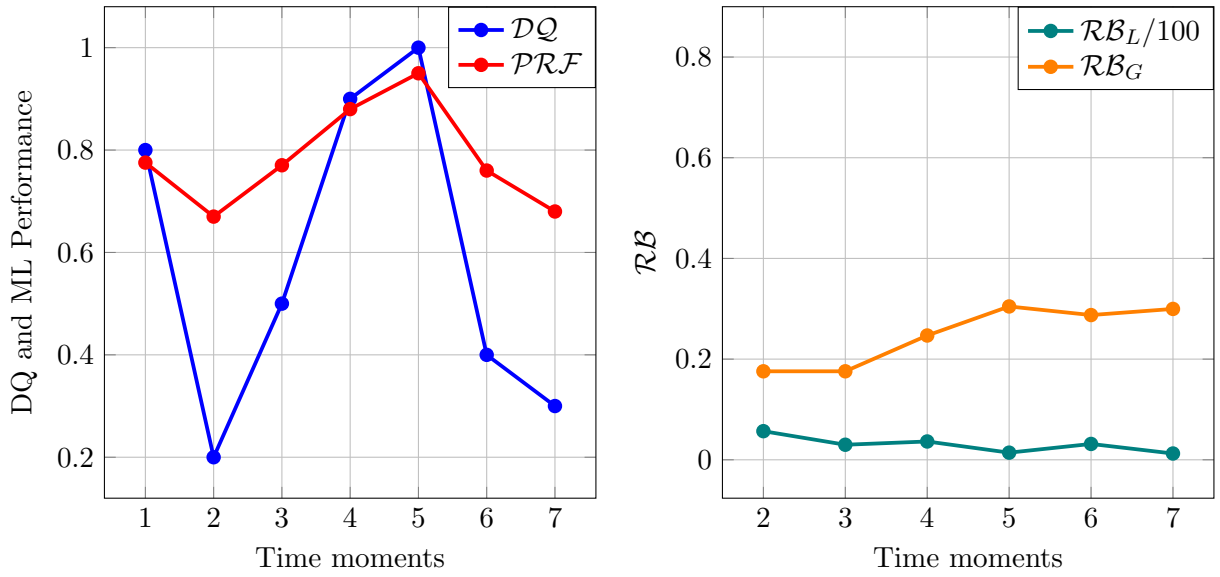
(a) (b)

Figure 4.23: ML application robustness calculation practical example for the DQ decreasing case: (a) – the relationship between DQ and ML Performance; (b) – calculation results for local and global robustness over the various tome moments



(a) (b)

Figure 4.24: ML application robustness calculation practical example for the DQ increasing case: (a) – the relationship between DQ and ML Performance; (b) – calculation results for local and global robustness over the various tome moments



(a) (b)

Figure 4.25: ML application robustness calculation practical example for the varying DQ case: (a) – the relationship between DQ and ML Performance; (b) – calculation results for local and global robustness over the various tome moments

DQ, obtained in a real-world use case described in sec. 4.2.4. We developed and represented two types of $\mathcal{RB}$ indicators that can be utilized in various manners:

1. $\mathcal{RB}_L$, which enables calculating ML robustness for a single time moment of the MLIN operation, in relation to the previous time moment. This indicator is in essence sensitive to the changes and may be employed for determining responsive tactical measures aimed at enhancing ML robustness in the next time moment. The sensitivity feature enables evaluating how effective were the employed tactical ML robustness assurance measures.

2. $\mathcal{RB}_G$, which shows ML robustness trend since the MLIN initialization. This indicator may be employed for a long-term time series analysis targeted at determining ML robustness improvement strategy, and evaluating the effectiveness of this strategy.

In comparison to the existing distance-based robustness measurement, quantification, and calculation approaches, commonly aimed at calculating how the ML model can tolerate perturbations of various intensity in the input data samples, ours is concentrated on the input DQ, and allows determining how this DQ affects the performance of the ML application. This provides the advantage of **(1)** employing intrinsic and contextual DQ measures [100], including security metrics that depend on the particular MLIN characteristics, and user and application requirements. This makes our approach highly flexible, personalized, and applicable to a wide range of ML applications. As additional advantages, our approach: **(2)** considers the integrity of MLIN components; **(3)** considers the interrelationships between them that affect the DQ; and, as we showed in this practical example, **(4)** reflects how these interrelationships impact the overall ML application robustness to the DQ variation.

4.5 Conclusion

In this chapter, on a wide variety of real-world use cases, we demonstrated how the MLIN cyberinfrastructure affects the quality of the input data processed by various MLIN components. We further showed how the variations in DQ impact diverse practical ML applications. In our comprehensive empirical investigations, we studied such ML applications, as image and sound classifiers, voice recognition and transcription systems, and video object detection and classification tools. Moreover, using POWDER, we employed a real-world network facilities to obtain data with real network corruptions. Below we list the major contributors developed in this chapter:

1. We **showed how the MLIN architecture, developed in 4.2.1, can be realized in the real-world** using POWDER platform facilities.

2. In sec. 4.2, on a number of practical examples, **we demonstrated the interrelationships between MLIN cyberinfrastructure, input DQ, and ML application performance. We developed practical knowledge** on how the MLIN cyberinfrastructure conditions affected the input DQ, and the performance of studied ML applications.

3. In sec. 4.3, **we represented how the MLIN feedback adjustment system can be realized in practice. We showed how the practical knowledge, obtained in our real-world use cases, can be leveraged to develop the rule-based network adjustment system**, employed to assure ML application performance in MLIN.

4. In sec. 4.4, **we showed how our generic ML robustness calculus, introduced in sec. 2.6, can be moved into practice** by developing two distinct ML robustness indicators that enable analyzing the ML robustness in various ways and employ it for different tasks and applications.

5. By demonstrating our ML robustness calculus practical example, we emphasize its flexibility and applicability to a wide variety of data types and modalities, and diverse ML applications.

Chapter 5

Enhancing Machine Learning Robustness from the Learning Perspective

In this chapter, we consider the approach to enhance ML robustness during the MLIN execution by employing various methods and techniques in the ML model training process. In comparison to the proposed MLIN adjustment feedback component, presented in 2.3, which is employed during the ML application operation, these methods and techniques are designated for the MLIN design and ML model learning stages. In particular, we focused on two techniques not originally proposed to enhance ML robustness but appeared to be effective in this matter: TL and FL. To verify the potential of these approaches in practice, we evaluate them on a multiple industrial applications, including transportation and digital payment systems domains. We first discuss and compare the effectiveness of both TL and FL in training ML models robust to DQ variations, and analyze what practical scenarios are more applicable for each of them. Next, we propose to further enhance the security and privacy of FL by integrating it with our Reputation and Trust techniques, proposed in 2.5. We assess the ability of our approach to detect the compromised local units, and to enhance the ML application security by excluding the compromised local units from the training procedure. Our results indeed demonstrate that the employed techniques are able to enhance the ML application robustness to DQ variation. Being combined with the approaches employed during the execution stage, MLIN highly benefits from this synergy, as this allows to address more aspects that impact ML robustness and decrease risk of its degradation, which makes the combination approach more comprehensive. Below we overview the content for each section we present in this chapter.

- In sec. 5.1, we discuss a variety of strategies aimed at ensuring ML application performance and enhancing its robustness to DQ variations. We explore their nuances, strengths, weaknesses, and unique features. Our focus narrows to two major approaches: TL and FL, both chosen for our empirical verification.
- In sec. 5.2, we investigate TL and FL within an industrial context. We employ these techniques to train ML traffic signs image classifier using data of varying quality. We compare these approaches and analyze their effectiveness in handling input data of diverse quality during the execution.
- In sec. 5.3, we propose to further enhance the security of the conventional FL by integrating it with our Reputation and Trust techniques we developed in sec. 2.5. We evaluate our approach over a real-world financial dataset, and demonstrate the effectiveness of our approach in enhancing ML performance and the overall FL security.
- In 5.4, reconsidering the results obtained in our industrial cases, we discuss both TL and FL with a focus on enhancing ML application robustness during the execution phase.
- In 5.5, we elaborate on the synergy between techniques applied during the learning stage and those implemented in the ML execution. We emphasize the benefits of combining these strategies to further enhance ML application robustness and generalizability, offering a comprehensive solution to the challenges of ML robustness improvement.

5.1 Approaches to Enhance ML Robustness through Learning

As we demonstrated in our real-world use cases in sec. 4.2, assuring the required ML robustness is an important aspect of practical ML applications. Previously, we considered ensuring ML application performance and improving robustness only in the ML model execution stage, when its already pretrained and provides inference results over the unseen data samples. In sec. 2.1 and 2.2, we discussed various approaches to define, quantify, and enhance ML robustness, and in sec. 2.6 we proposed our novel approach to ML robustness based on the combination of the relationship between the input DQ and the ML application performance. To improve ML robustness, we introduced our MLIN adjustment feedback component, aimed at selecting MLIN structure that satisfies the established requirements. As this component provides feedback based on the triggered event (e.g., the drop in the ML application performance), the feedback approach is reactive and operates during the ML application execution stage.

There are numerous other approaches proposed to increase DQ of the input samples [8,14,123] and enhance ML robustness against DQ variations [64] during the execution stage. Data pre-processing approach is aimed at improving the quality of data samples to make them more usable for the ML model and effective in terms of ML performance demonstrated on these samples [77]. The pre-processing is performed over the unseen data in order to prepare it for submitting to the ML model. The rationale here is to adjust the input samples to the format and structure acceptable by the trained ML model, which allows effective processing of these samples [176]. The methods and techniques in this stage typically involve data normalization, scaling, or any other transformations to adjust the samples to the range and format relevant to the training data. The pre-processing in the inference stage ensures that the input data is compatible with the ML model's expectations to produce accurate and meaningful decision-making.

ML model ensembles have the goal to process the input data by the multiple ML architectures in order to produce a number of decisions that are integrated to make the final one [254]. The major idea here is to compose the ensemble of diverse ML architectures trained on diverse data in order to make the ML application more generalizable and reduce the effect of outliers and noise in the input samples on ML application performance. Here, such methods as averaging (e.g., Bootstrap Aggregating), boosting (e.g., AdaBoost, XGBoost), and stacking (e.g., with Cross-Validation) might be mentioned as the examples.

Another perspective is to leverage special preventive methods and techniques during the ML model's learning stage targeted at enhancing ML application robustness against the input DQ variation during its execution. These methods try to prevent or reduce the impact of DQ variations on the ML models a-priori. A well-known and widely used example of preventive methods is training data augmentation. This technique allows to substantially enhance and diversify the training data by applying various transformations to the existing data samples. Another common instance is regularization, which is employed to make the ML model less susceptible to the input DQ variations and adversarial attacks [208, 235]. The regularization is aimed at reducing the risk of ML model excessive reliance on the specific patterns and features in the training data. This allows to better control the model's complexity and to prevent its high sensitivity to the non-relevant information in the inputs. Instances of the regularization techniques are L1 and L2 regularizations, dropout, data augmentation, early stopping, and others.

Adversarial training is a technique initially aimed at enhancing ML model's robustness to the data modifications introduced by malicious attackers [217]. Adversarial attacks are intentional attempts to modify the ML model outputs in order to decrease the ML overall performance or to

force the model to produce targeted malicious results on a specific samples. These attacks exploit the vulnerabilities and weaknesses in ML models design such as insufficient model's complexity and a lack of contextual verification mechanisms. Some examples of adversarial attacks are data poisoning, model inversion, and composing adversarial examples [32].

There are several approaches to mitigate the adversarial attacks with adversarial training techniques [253]. The training collection might be initially populated with the adversarial examples in order to enhance ML model's robustness to them in the inference stage. Another approach is to dynamically generate adversarial examples after each training iteration based on the ML model's parameters. Iterative methods employ small data modifications and perturbations that enable the ML model to learn a wide diversity of adversarial examples, which is commonly more effective than initially populating the training set with a limited number of adversarial samples. Other approaches are ensemble adversarial training, which employs the aggregation of outputs provided by ML models separately trained on benign and adversarial examples; and defensive distillation that incorporates teacher and student models in the learning process.

Alongside the mentioned preventive methods, recently appeared FL technique also found its successful application in enhancing ML robustness towards DQ variations. Originally designed to enhance privacy and reduce communication and computation resource consumption, FL aggregation mechanism was found fairly effective in improving ML models generalizability. In this chapter, we demonstrate how the ML robustness to DQ variations can be further improved via techniques implemented in the training stage. In particular, we first concentrate on two major preventive techniques: TL and conventional FL, and study and discuss their effect on various industrial ML applications. Then, we focus on further enhancing the security and privacy of the conventional FL by empowering it with our novel Reputation and Trust-based techniques, proposed in sec. 2.5. Below we describe TL and FL techniques, applied in the model's learning stage, and discuss their contribution to enhancing ML application robustness towards the input DQ variation during the ML execution stage.

5.1.1 Transfer Learning

TL is a highly powerful technique in ML that leverages knowledge from one domain to enhance ML model's learning performance in another, commonly referred as a target domain [41]. It has gained significant attention due to its ability to address the challenges of data availability, impractical costs to gather or generate this data, or costs to train an ML model from scratch appropriate in terms of performance. By utilizing the knowledge on a source domain, TL enables the transfer of this

knowledge and relationships to a target domain, even when these two domains differ in their data distribution, feature spaces, or even classification task. The major objective of TL can be defined as achieving the required ML performance on a target domain with limited unlabeled observations by utilizing a pre-trained model on a labeled source domain [256].

Weiss *et al.* [240] classify TL approaches as homogeneous or heterogeneous, based on the similarity or disparity between the source and target domains' feature spaces. Heterogeneous feature spaces present additional challenges in adapting the distribution and feature spaces, requiring sophisticated and intelligent approaches [58]. Zhuang *et al.* [260] categorize TL approaches based on the elements employed for knowledge transfer in the target domain. Data-based approaches focus on modifying and adapting the input data. These strategies include instance weighting and feature transformation. Instance weighting aims at reducing the difference in the observations' marginal distribution between the source and target domains by adjusting their weights. Feature transformation strategies, on the other hand, involve identifying common features and transforming them to align with the target domain.

One of the well-known TL approaches is domain adaptation, which focuses on adapting the knowledge from the source domain to the target domain by minimizing the distribution between them. Various methods have been proposed, including distribution matching or adversarial training. Distribution matching aims at minimizing the divergence between the source and target domain distributions, such as the Maximum Mean Discrepancy [69] or the Wasserstein distance [236]. Another method is Metric TL, which leverages various metrics encoding in TL [249]. Instance weights are learned and utilized to aggregate the distributions of different domains, while a distance metric is learned simultaneously to maximize the distances within a specific category and minimize the cross-class distances for the target domain [249].

Adversarial training incorporates the process of domain discriminator training, which is responsible for distinguishing between the source and target domain while the feature extractor aims at confusing the discriminator by generating domain-invariant representations [61]. This approach enables the feature extractor to learn the domain-invariant relationships that can be applied even to the target domain.

Another approach to transfer the model from one domain to another is ML model fine-tuning. Fine-tuning involves adapting a pre-trained ML model to a source domain in order to enhance its performance on a target domain [232]. The process commonly relies on adjusting the ML model's hyperparameters using a labeled data from the target domain. This enables the ML model to learn the target domain-specific characteristics and enhances its performance on a target samples.

However, excessive adaptation may lead to ML model's overfitting to the target dataset. Techniques such as gradual unfreezing [87], where the pre-trained ML model's layers are unfrozen and adjusted dynamically, can help in mitigating these issues and improving the re-trained model adaptation to the target domain.

There are other TL approaches designed and developed for the specific cases. Multi-task learning can be combined with TL and is aimed at leveraging the common features between various models designed for different classification tasks in order to improve the ML performance on the target domain [145]. The employment of TL highly benefits various industrial applications and domains, including computer vision, natural language processing, and audio processing and analysis. For example, by utilizing ML models pre-trained on well-known state-of-the-art image datasets like ImageNet [59], TL enables the efficient re-training on smaller and task-specific sample collections, as we demonstrate in this chapter.

5.1.2 Federated Learning

FL has gained prominence due to its ability to address the evolving privacy and communication resource consumption needs for contemporary ML applications [143]. FL offers a privacy-preserving learning process by preserving the local training data on individual physically or logically distributed units, thereby reducing the requirement for the centralized data accumulation [10]. Initially introduced by Google to improve smartphone keyboard query suggestions quality [251], FL has become a well-known ML technique in various applications, especially in user privacy-oriented ones [152]. The major FL aim consists in training local ML models on the distributed local data samples and exchanging the ML model's parameters (e.g., the weights and biases) with the aggregation unit to generate a global model further shared by the local units. The conventional FL architecture typically involves three major components: an aggregation unit, responsible for gathering, aggregating, and distributing the model updates; a set of local units, responsible for iterative local training, submitting the models for aggregation and applying the global updates; and a communication channel, employed for transmitting the model updates between the local and aggregation units.

The FL process can be divided into several stages. First, ML models are trained locally on the available data within each participating unit. Next, the trained model parameters are transmitted to the aggregation unit. In the subsequent stage, the received local models are aggregated to create a global model [160]. Finally, the obtained global model is communicated back to the local units for the further local training. This iterative steps compose the aggregation round, and these rounds are repeated until the target ML model meets user and application requirements.

The fundamental FL feature is preserving the local data privacy by communicating only the model updates instead of raw data to the aggregation unit. However, addressing various challenges in FL is crucial to ensure its effectiveness. One of the key challenges is how to aggregate the updates from local units in a way that allows to balance the generalizability, robustness to the local data variations, and performance of the resulting model. Various aggregation functions have been proposed, which may have manifold effects on the FL performance, robustness, and privacy.

One of the most common aggregation strategies is FedAvg [143], which incorporates averaging of all the local updates submitted for aggregation. This strategy is advantageous in terms of computational efficiency and implementation simplicity, however, it is subject to the aggregated model performance degradation due to a lack of robustness to the outliers in the local data and due to malicious updates. Other aggregation strategies examples are weighted averaging, where the aggregation unit assigns various weights to the local units based on some criteria; using Geometric Median [171] instead of simple averaging, which is more robust to the outliers in the data; Krum [70]; trimmed mean [252]; and FedMGDA [89]. These strategies aim at addressing some of the limitations or challenges in the conventional FL aggregation.

Not independent and identically distributed training data (non-iid data) residing on the local units possesses a significant challenge for the FL process, as it may introduce shifts between the local datasets and result in degraded global model performance. This problem is especially actual if we consider real-world industrial applications, where DQ may highly vary due to numerous factors. In addition, the analysis of non-iid data may also be employed by an adversary to detect to which particular local unit this data pertains, and extract sensitive or confidential information about this local unit.

The non-iid local data problem in FL has been addressed from various perspectives. Some studies leverage the robust aggregation strategies to train ML model more robust to heterogeneous or shifted data distributions. Examples of such solutions are FedProx [125] and FedNova [234] aggregation schemes. Other works focus on reducing the communication burdens during the FL process. Some examples of the solutions here are SCAFFOLD [96] and FedMA [233].

Industrial applications pose several challenges to achieve ML application robustness to the DQ variations [134]. These challenges include limited access to the data on the target domain, dynamically generated data which quality depends on various factors, complex and dynamic ML application operation environment, operation costs and resource constraints, as well as others. In this chapter, we investigate the two approaches not initially designed to enhance ML robustness against the DQ variations, but practically appeared to demonstrate effectiveness in this manner: TL and FL.

However achieving the similar goal, these approaches drastically vary in the means to achieve this goal. TL makes use of adjusting the last layers of the ML model's architecture and re-training it on the new training collection relevant to the target domain. Since TL re-uses the majority of ML model's parameters trained on the previous data, re-training on the varied DQ enhances the model's robustness and generalizability, but commonly decreases the performance [227]. In contrast, FL utilizes the mechanism of ML local models' aggregation, which employs the specific aggregation function to produce the global model based on the majority of submitted local updates. Hence the local ML models are trained by numerous data sources on various samples that might be of manifold quality and distributions, the resulting global model is inherently able to distinguish diverse features carried by these local models. In our research, we investigate how effective these two approaches when employed to enhance ML application robustness against the DQ variations in the industrial use cases. In sec. 5.2.2, we analyze and compare the effects of various setups and derive the best methodology to train more robust ML models using each of the approaches.

5.2 Employing Transfer Learning and Federated Learning to Enhance Industrial Machine Learning Robustness to Data Quality Variation

As we showed in sec. 4.2, DQ plays a vital role in the performance of data-driven systems, including ML applications [107, 183]. In industrial applications, DQ commonly varies depending on multiple diverse factors, including technological conditions, operational environment, and malicious attacks. Technological factors encompass issues in data gathering, storage, and transmission, such as sensor device failures or degradation in network QoS during the data transmission. Operational factors involve environmental conditions that affect the data gathering process, leading to increased noise. Malicious actions pose a significant threat to DQ, including attacks like data poisoning or introducing excessive noise to degrade ML model performance. These attacks can be directly targeted at the data or at the ML cyberinfrastructure itself, compromising data integrity [107]. In sec. 3.1, we discuss these factors in detail and provide other relevant practical examples.

Many industrial applications employ ML industrial systems designated for a particular application domain, e.g., Google Vision AI[1] and AWS Rekognition[2] for image classification and object detection. Unfortunately, those systems explore models, which have been typically pre-trained on good

[1] `https://cloud.google.com/vision/`

[2] `https://aws.amazon.com/rekognition/`

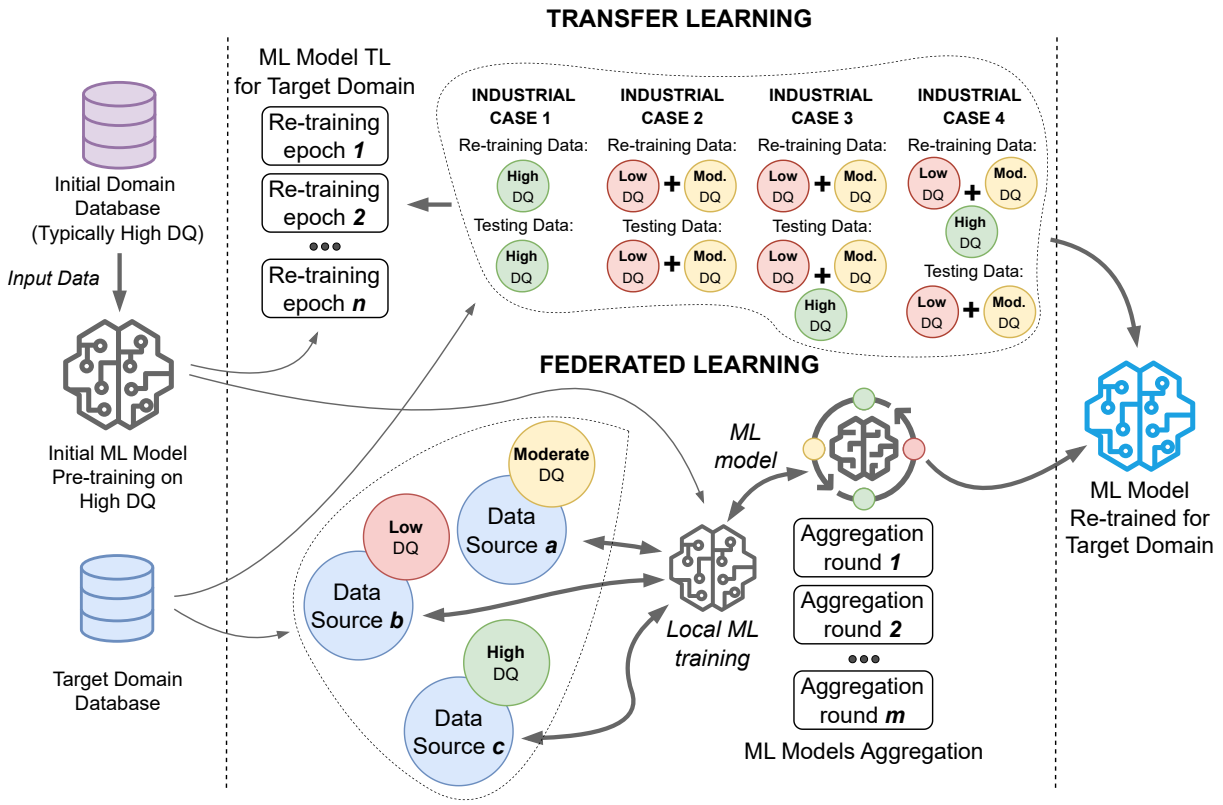


Figure 5.1: Schematic representation of the conceptual differences and similarities between the TL and FL empirical study setups, employed in this chapter. The upper part of the Figure depicts the studied TL cases on using the various DQ combinations for ML model re-training. The bottom part portrays how the training progress is organized in the FL setup. Both TL and FL techniques employ the pre-trained ML model, and then this model is re-trained in order to achieve higher ML model robustness to DQ variations in the industrial application scenario

quality images only. Their performance while executed in real life industrial cases with decreased DQ may demonstrate a significant decline. To improve the robustness of ML applications against DQ variations, it is necessary to handle and adapt to these effects while minimizing their impact on learning and inference performances. In order to enhance learning efficiency and ML generalizability, the approach to adapt already pre-trained models to a target knowledge domain have become favorable.

Generalizable and robust ML models are crucial for industrial applications as they can adapt well to data distribution shifts and effectively handle unseen samples. Moreover, robust ML models increase trust and confidence in their outputs, making them more reliable for critical domains where accuracy is of utmost importance [244]. The problem of DQ variations in ML applications has been approached through various methods. Data-oriented techniques, such as data cleaning [123], denoising [8], or wrangling [14], aim to improve DQ but do not actually contribute to the robustness of ML models against DQ variations during their execution. The reason for this is ML models are inherently dependent on the data they were trained on [227]. Hence, these data-oriented methods may improve ML performance on high-quality testing samples but limit the model's generalizability and performance on the data of lower quality.

An investigation of various ML model's robustness in industrial application design does deserve further research. In this section, we examine, verify, and analyze the capabilities of TL and FL to address DQ variations in real life data. Our goal in this section is to demonstrate that despite TL and FL distinct natures and original purposes, they both are suitable for improving ML application robustness against DQ variations but their selection depends on the available resources and other conditions. We focus on an industrial real-world use case with the ML computer vision application for the ITS to detect traffic signs, specifically dealing with varied DQ due to network QoS decrease. Figure 5.1 demonstrates similarities and differences between TL and FL in industrial ITS application settings. The quality of data in ML industrial applications is commonly dynamic and might temporally vary due to numerous factors, described in sec. 3.1. As suggested by Vela *et al.* [227], the most known effective solution to address this problem is ML model periodical re-training in order to correspond its knowledge representations and performance to the actual data distribution.

Considering that in real-world industrial applications data might temporally change due to numerous random factors, we study various DQ re-training and testing set combinations and analyze how they affect the performance after the ML model re-training. In both TL and FL cases, we employ industrial ML models pre-trained on state-of-the-art dataset composed of high quality data. As our target domain data, we employ the traffic sign images of varied quality corrupted by the unstable

network conditions (see sec. 4.2.2). We examine and analyze the robustness of ML models, trained in TL and FL manners, during the ML application execution. Through empirical results, we discuss the applicability of both approaches in enhancing ML robustness in industrial applications.

5.2.1 Industrial Use Case Description

To facilitate our industrial use case, we design a prototype of ITS, which is responsible for transmitting the images from the data sources over a wireless network to a cloud-based ML application and classifying them into stop sign or non-stop sign categories. In our experiments, we employ a similar subset of traffic signs images from the Open Images V6 Dataset [2], used in sec. 4.2.2. We employ the corrupted images, obtained by transmitting them through the POWDER wireless network with QoS variations (see sec. 4.2.1), in both TL and FL setups to train and test several ML image classifiers on various DQ. By evaluating the performance of ML image classifiers re-trained in TL and FL manners on DQ of various quality, we investigate and derive the interrelationships between the ML robustness, re-training and testing DQ variations, and the selected learning architectures. Moreover, we focus on understanding how to train more robust ML models for the studied ITS use case, and we provide our generic recommendations. Below we describe the experimental setup in more technical details.

Transfer Learning Setup

In this use case, we employ VGG16 image classifier [204], described in sec. 4.2.2. We first transmit our data over a real network in the unstable QoS conditions, and obtain the images of varied quality. We then submit these corrupted data to the employed classifier, pre-trained on the ImageNet [59] dataset and re-trained on the good quality images from our target domain. We measure the ML baseline performance and find that the DQ variations reduce the ML model's ability to correctly classify the provided samples due to the lack of robustness to the low quality input data. To investigate ML robustness enhancement ways, we further re-train our baseline model on the images of varied quality. We study various re-training and testing data compositions and evaluate how these methods contribute to the performance and robustness of the produced ML model. We re-train the model for 20 epochs and 74 steps per epoch in each experimental scenario with the Learning Ratio (LR) of 0.001. We present the obtained results for each of our experiments in sec. 5.2.2.

Federated Learning Setup

In this study, we follow the approach developed by Manias and Shami [139], who suggested using Road Side Units (RSUs) as FL units to train the local ML models for ITS applications. We aim to improve the reliability of FL-based traffic sign classification system in the face of input data corruption caused by the vulnerable conditions of ITS cyberinfrastructure. We envision a scenario where RSUs obtain data from mobile and static nodes over the network and leverage this data to train a local ML model, which is then transmitted for aggregation with the updates from other RSUs. The data obtained by RSUs originates from various sensor devices (e.g., embedded into vehicles or road infrastructure), and is conveyed over a wireless communication channel, which might induce the DQ variations. This data of diverse quality is then used to train the local models on each RSU. In our prior work, we measured the performance of centralized ML models over the data influenced by various network QoS conditions [45]. In this paper, we investigate how to train ML models robust to DQ variations, and we study FL as one of such robustness enhancement techniques. In our experiments, we employ the images of various quality to re-train the ML model in the FL manner, and then cross-evaluate the re-trained model performance on various DQ data cohorts. We analyze the obtained results and compare them with the ones obtained in the TL scenario. Furthermore, we propose our recommendations on boosting the ML computer vision classification system robustness for the considered ITS industrial use case.

To facilitate our empirical study, we develop the FL framework in Python using PyTorch. As the ML image classification model, we selected ResNet50 architecture [81], described in 4.2.3, and pre-trained over the ImageNet [59] data. As a data collection, we utilize the original (high quality) and corrupted (low quality) labeled traffic sign images, described in sec. 4.2.2. The corrupted images are represented by five cohorts corresponding to the network packet loss ratio during their communication: 1, 2, 5, 10, and 20%. In this case, we did not change the buffer size and employed images transmitted with 512B buffer. The employed ML model is supposed to classify images into stop and non-stop sign categories.

We perform several experiments with the models trained on a single image cohort influenced by a certain packet loss percentage, and tested on all other image cohorts. For instance, we trained the FL model over the high quality data distributed over 10 clients, and then we evaluated the resulting model on image sets of diverse quality (influenced by the packet loss of 1, 2, 5%, etc.). The image corpus assigned to each unit is split into 66% of training and 33% of testing data in order to perform a local training iteration.

In our experiments, we perform 10 successive FL training rounds with 10 local training epochs for each client. During the training, we set LR to a constant of 0.001. After the local training, the acquired models are transmitted to the aggregation unit, where the aggregation procedure is performed. We contrast two FL aggregation strategies: Federated Average (FedAvg) and Geometric Median (GM). FedAvg employs averaging of all the model parameters received from each unit [143], and is calculated according to (5.1), where w_{t+1} is the global model parameter at round $t+1$, n is the number of local units participating in the aggregation, and $w_{t,i}$ is the local model parameter of unit i at the aggregation round t. GM is claimed to be more robust to outliers and deviations in the models' parameters [171], and is calculated according to (5.2), where w is the point that minimizes the sum of distances to all the local model parameters of round t. After the aggregation procedure, the global model is produced and its updates are transmitted back to the local units. The local units apply the received global model to perform the next FL training round.

$$w_{t+1} = \frac{1}{n} \sum_{i=1}^{n} w_{t,i} \tag{5.1}$$

$$w_{t+1} = \underset{w}{\operatorname{argmin}} \sum_{i=1}^{n} \|w - w_{t,i}\| \tag{5.2}$$

5.2.2 Industrial Use Case Results

Transfer Learning Re-training Results

Baseline Model To adapt the pre-trained VGG16 to our specific knowledge domain, we re-train it on the set of high quality images. Then, we test this re-trained model on samples corrupted by the unstable QoS while transferred over a network. For the model evaluation, we employ images transmitted with various buffer sizes and packet loss percentages: buffer size of 128B and packet loss of 5%; buffer size of 256B and packet loss of 5%; buffer size of 512B and packet loss of 20%. The results of the baseline model testing classification accuracy are shown in Figure 5.2, which provides mean values of the classification accuracy demonstrated by the model over 20 re-training epochs, and Standard Deviation (SD) of these values. According to the results obtained, the ML model re-trained on original images demonstrates on average the lowest performance in comparison to other data re-training cases. Hence, ML models pre-trained on the high quality data could not be suitable for classifying the samples of varied quality in the selected industrial use case, and need to be further trained.

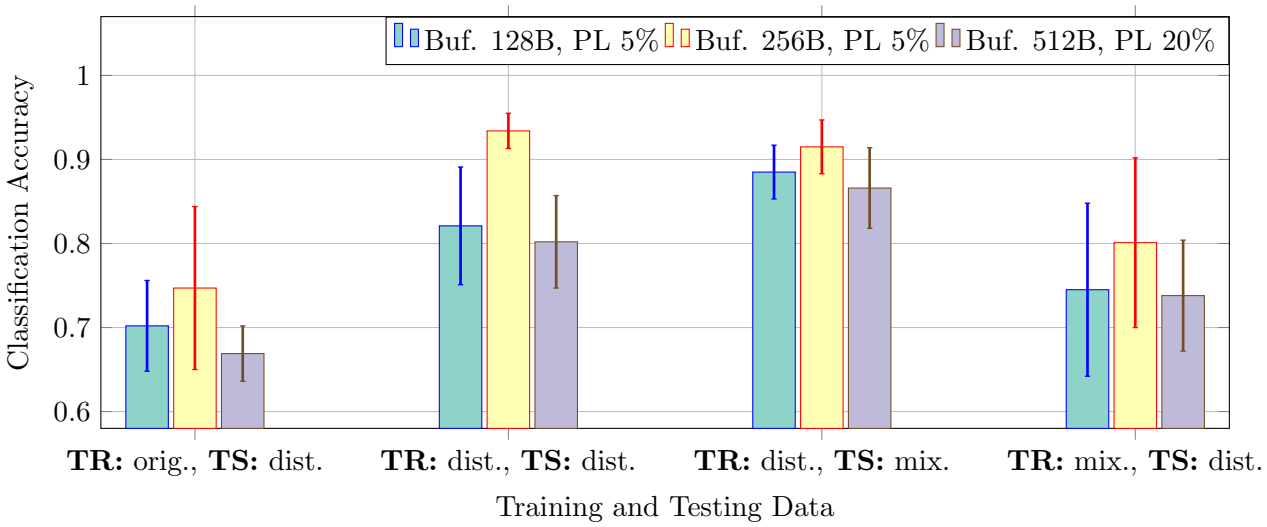


Figure 5.2: Average classification accuracy and its SD over 20 training epochs, demonstrated by the pre-trained ML model after the further TL procedure. TR refers to the data, on which the ML model was re-trained; TS refers to the data, on which the ML model was tested after re-training; orig. refers to the original set of images; dist. refers to the set of images distorted by the placket loss during the network communication

Re-training and Testing on Distorted Samples To explore possible ways of enhancing ML model's robustness to the varied DQ, we continued to further re-training the baseline model on corrupted images. TL is performed separately for various DQ variations categories, such as buffer size and packet loss percentage, e.g., first we re-trained the baseline model on images distorted by communication through the channel with the 128B buffer size and 5% packet loss, and tested this model on images distorted in the same way; then we repeated this procedure on other buffer sizes and packet loss percentages. According to Figure 5.2, in comparison to the baseline model training results, the classification accuracy did not improve enough over the training process and was still not sufficient for the industry-level systems. This means that it is much more difficult for the model to learn the knowledge representations over the varied quality data. Also, the higher the image corruption degree we trained the model on (e.g., an increased packet loss or reduced buffer size), the lower classification accuracy we obtained on the testing set.

Re-training on Distorted Data and Testing on a Mix of Distorted and Original Data In this case, we examined further training the baseline ML model only on the distorted samples and testing it on a combined collection of high and low quality samples. As in the previous scenarios,

we conducted experiments for all corrupted images categories. The mean and SD values for the classification accuracy for each data cohort are shown in Figure 5.2. One can see that the model re-trained on low quality data only and tested on a mix of distorted and original images demonstrated on average higher performance in the majority of cases.

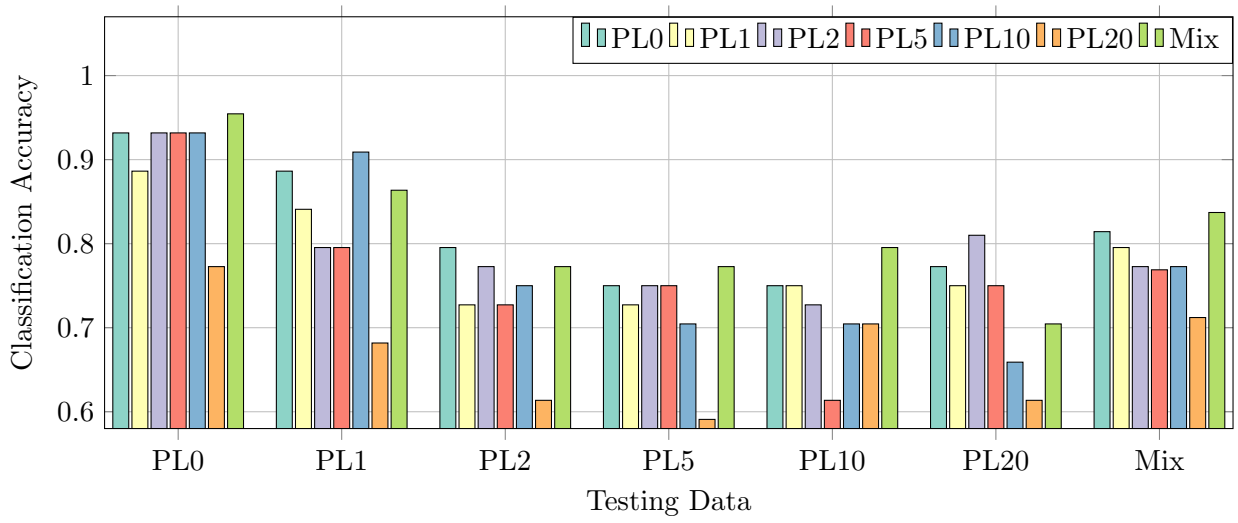
Re-training on a Mix of Original and Corrupted Data and Testing on Corrupted Data In this scenario, we continued to re-train the baseline model on a mixed set of high and low quality images, and evaluated its performance on the low quality samples only. As in the previous case, the model was re-trained and tested separately on each DQ image cohort. According to Figure 5.2, combining high and low quality samples into a single training set helped to slightly improve the ML performance in comparison to the baseline model. However, in this case training and testing sets differed more than previously, and the ML model trained on the mixed set demonstrated lower performance over the low quality data than the model re-trained on this low DQ only.

Federated Learning Re-training Results

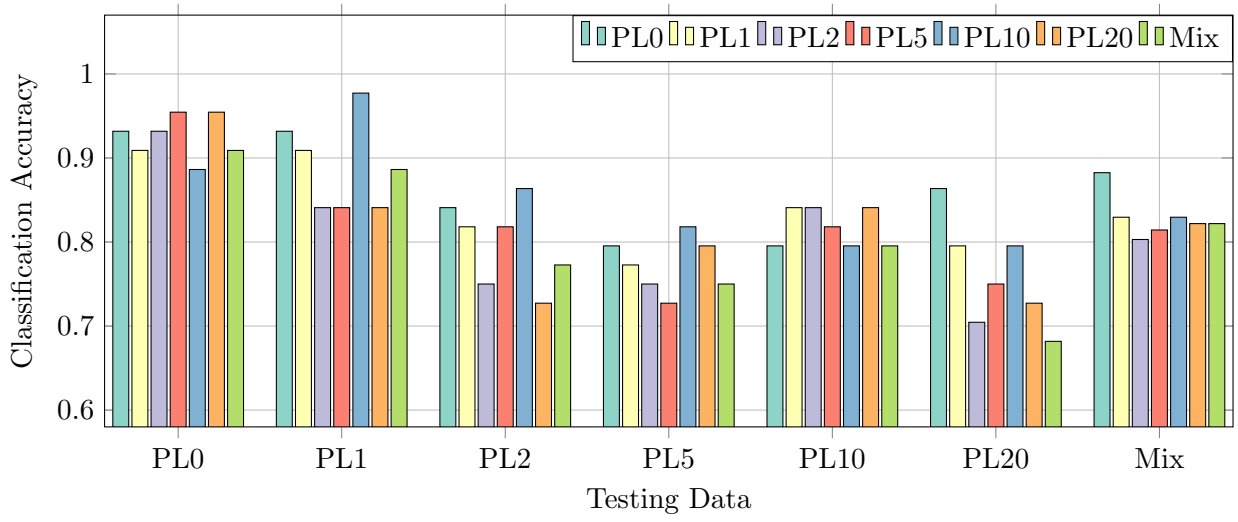
Figure 5.3 illustrates the results for the image classification accuracy attained by the model re-trained in a FL manner over the image testing sets of diverse quality. In each experiment, we evaluated the ML models trained on various DQ and measured their performance against the images corrupted by real unstable network conditions. To examine the case when the data produced by a single local unit might be of varied quality, we incorporated the cohort that contained the combination of all employed DQ, i.e., the combination of original images and images affected by 1-20% packet losses. This image testing set is marked as "Mix" in Figure 5.3.

Figure 5.3(a) reveals the ML performance results obtained with the application of FedAvg as the FL aggregation strategy. The models re-trained on the mixed DQ set showed on average better performance on the original images. Interestingly, despite the re-training on the low DQ categories, the model exhibited the highest performance on the original images in all cases. The model re-trained on the "Mix" image cohort allowed to achieve more stable results in terms of classification performance. It surpassed models re-trained on other DQ categories in four testing cases, while the models re-trained on other cohorts excelled only once: re-trained on "PL10" and tested on PL1", re-trained on "PL0" and tested on PL2", and re-trained on "PL2" and tested on "PL20".

Figure 5.3(b) demonstrates our FL model image classification accuracy with the GM aggregation strategy. As one can see from the results, the models re-trained with this FL aggregation strategy



(a)



(b)

Figure 5.3: FL model's performance after 10 consequent FL re-training rounds demonstrated over various DQ using two aggregation strategies: (a) – FedAvg; and (b) – Geometric Median (GM). Various colors represent the employed re-training data, and the labels on the horizontal axis correspond to the data the models were tested on. PL is a packet loss, and the number after PL corresponds to the packet loss percentage, for example, PL0 corresponds to 0% packet loss percentage during the image transmission over the network

attained on average better performance than with FedAvg in almost all the considered cases. Like the FedAvg case, the models classified the original images on average better than the other categories. However, the model re-trained on "Mix" cohort did not follow the same trend and did not even dominate in any experiment. The model trained over "PL10" showed better performance in "PL1", "PL2'', and "PL5" cases, and surprisingly, the model trained over the original data demonstrated the higher classification accuracy in "PL20" and "Mix" testing categories.

Resuls Analysis

Transfer Learning Case Results Analysis As we learned in our investigation, pre-trained open-source ML models, even the state-of-the-art ones, may not perform sufficiently well in industrial applications, where DQ may vary. ML models are usually trained only on good quality images (e.g., state-of-the-art datasets such as ImageNet [59]), but in practice, images can be distorted by various factors, such as vulnerable cyberinfrastructure. Further re-training is needed to demonstrate better performance on both high and low quality samples from the target domain. If the DQ is not expected to vary much during the ML application operation, a short re-training with some additional samples can rapidly help in ML performance improvement. However, this may not work for more complicated cases, where the data undergoes multiple stages and is processed by various MLIN cyberinfrastructure components before being classified. In this case, the DQ might be affected by noise, interference, data loss or malicious attacks. Our investigation showed that re-training on good quality images only does not improve the ML performance on the varied DQ.

In this chapter, we studied TL from a source domain (high DQ) to a target one (varied DQ) as one of the ways to enhance ML robustness against DQ variations. TL can involve re-training on the varied DQ only or on the mixed set of varied quality images. Our results demonstrated that TL allowed improving the ML testing performance on both high and low quality images. With re-training on the low quality data, the ML performance on both DQ types was commensurable. Re-training the ML model on the mixed set resulted in less efficient training process, as it required more time for the ML performance to converge because of the dynamic and inconsistent patterns in the training data. Based on our investigation, we can recommend to perform further ML model re-training to the target domain data in order to enhance its robustness to the varied DQ, and to improve the performance on both high and varied quality data.

Based on our results, one can see that current open-source ML classifiers are needed to be re-trained on lower quality data samples to achieve acceptable performance in industrial applications. The classifier's performance robustness to possible DQ variations can be improved with TL by further

re-training on bad quality images. TL on low DQ only appeared to be more effective than extending the training base by combining high and low DQ.

Federated Learning Case Results Analysis Based on our investigation results, we can offer the following suggestions on how to increase the robustness to the DQ variations of ML computer vision ITS systems re-trained in a FL manner. One suggestion is to use ML models initially pre-trained on comprehensive datasets (e.g., ImageNet [59]) and re-train them for the target domain rather than training a model from scratch only on the local data. This pre-training allows improving the model's generalizability on new data. Another suggestion is to consider the dynamic ITS environment and changing image DQ used for training, which may affect the ML training performance. If the operational conditions are stable, only the local data may be used for the training. Otherwise, the better strategy is to mix the data received from various sources on RSU for the local training in order to improve the robustness of the model. Moreover, in almost all the investigated cases, the FL models trained with the GM-based aggregation strategy showed higher robustness against the DQ variations than FedAvg.

Discussion on Transfer Learning and Federated Learning Capabilities to Enhance ML Robustness to Data Quality Variations The performance of ML industrial applications is significantly influenced by the quality of the training data. When ML models, trained on original data, are tested on the lower DQ, their performance tends to decrease. This observation emphasizes the importance of maintaining high input DQ for achieving the required ML performance in the operational stage. However, in industrial applications, the quality of input data may dynamically vary due to multiple reasons. As a way to address this problem and to enhance ML robustness to DQ variations, we investigated two approaches that were not directly designed for this task: TL and FL.

As our empirical results demonstrated, TL strategies might be successfully employed to address the DQ variations. In cases when DQ variations were a concern, training the model solely on the low DQ, without including the high quality data into the re-training set, allowed to achieve higher performance. This can be attributed to the initial pre-training of the model on the original data, which establishes the necessary knowledge for recognizing patterns and capturing relevant features. By re-training the ML model only on the data of degraded quality, it better adapts to the characteristics and challenges posed by varied DQ, resulting in improved performance in the presence of DQ variations.

When dealing with DQ variations, FL with the GM as the aggregation strategy was found to be more robust compared to FL with FedAvg, provided the necessary computational resources are available. By employing GM, the FL aggregator is better equipped to produce more effective global model under the adverse local data conditions, making FL with GM a preferable approach when DQ is a concern.

When considering training or re-training on the high quality data, FL demonstrated greater robustness to DQ variations compared to TL. FL's inherent ability to leverage distributed data sources and aggregate models from multiple participants enabled it to handle variations in data distributions more effectively. FL proved to be a robust approach for mitigating the impact of DQ variations, surpassing TL in terms of overall performance and adaptability in the case of employing high DQ for the training.

Despite the advantages of FL, TL can still achieve comparable results when trained or re-trained on the varied DQ. In the scenarios examined, FL and TL demonstrated similar performance levels. This suggests that both FL and TL are viable strategies for addressing DQ variations, with each approach offering different advantages and trade-offs.

Our findings underscore the crucial role of DQ in training ML models that satisfy the industrial application requirements. ML models specifically trained to address DQ variations tend to exhibit lower performance not only on the varied DQ but also on the high quality data compared to models solely trained on high quality data. This observation highlights the impact of DQ on testing the model over both low and high quality data samples, emphasizing the need to address and improve the DQ throughout the entire MLIN data life-cycle.

5.3 Improving Communication and Security in Federated Learning with Trust and Data Quality Evaluation for Distributed Industrial Applications

Previously in this chapter, we considered the conventional FL process [112] (FedAvg aggregation case), which does not incorporate any additional security and privacy protection mechanisms. In the conventional FL, privacy protection is accomplished mainly by communicating only the local models instead of the original data to the aggregation units. Other techniques of further security and privacy enhancement have been proposed and incorporated into FL process, such as: employing more robust aggregation functions, e.g. Geometric Median [170], considered in

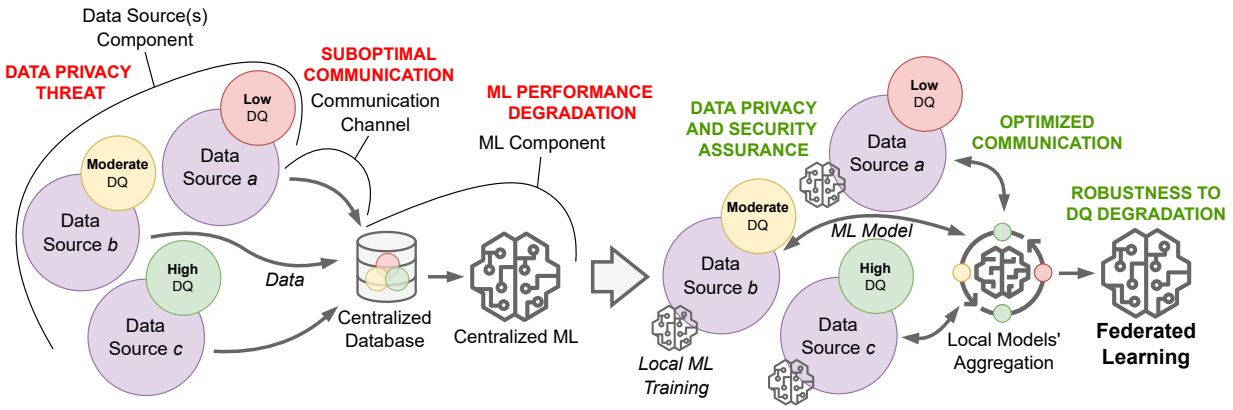


Figure 5.4: Distributed FL reduces communication burdens and enhances security and robustness in comparison to centralized ML, whose performance in practical applications may dramatically decrease with DQ degradation caused by communication problems or attacks against communication channels and data sources

the previous section, or Trimmed Mean [252] instead of FedAvg [143]; FL Based on Dynamic Regularization [6]; Personalized FL by Pruning [225]; employing root dataset to verify local models [28]; selecting a global model from the set of local models based on the square-distance score [22].

In Figure 5.4, we demonstrate the high level architectures of both centralized ML and conventional FL, show major weaknesses of the former architecture, and depict the inherent advantages of the latter one. However, despite the benefits introduced by the conventional FL, it still possesses certain vulnerabilities. The conventional FL process [112] includes the action of sending back the global model updates from the aggregation unit to local units in each aggregation round without considering the possibility that these local units might be compromised by an adversary that may result in a major privacy violation. In this case, the adversary might gain access to these global updates and apply data inference attacks to derive sensitive information. The leak of this information may violate the privacy of local units participating in the FL system. These attacks might be employed to infer the data stored on the local unit, which poses obvious privacy concerns [154]. Various studies have addressed malicious update detection and mitigation [22, 26, 203]. However, the conventional FL operation flow [112] does not envisage the possibility of excluding compromised local units from the global updates distribution. This challenge needs to be addressed in order to improve the trust to the FL process and the prospects of its applications in industry.

So far, FL models have been commonly trained and evaluated against popular datasets such as MNIST [60], CIFAR-10 [113], and others. Unfortunately, these models are not able to demonstrate

the state-of-the-art specification performance on real-world data [246] while employed in the industry. This happens due to the complexity of the real-world data structure, the inherent noise that decreases the quality of the data, and the larger diversity of the patterns that are not presented in those training collections. Popular data collections heavily cited in the literature are carefully curated and have specific characteristics that might not represent diverse real-world patterns that the models might occur in the operation stage. Trained on the set of specific samples, which do not represent the manifold of real industrial cases, the model may overfit to them, which makes them less generalizable and perform poorly on unseen real-world samples. In addition, in real operation conditions, the data can exhibit a wide range of corruptions due to various natural, technological, and malicious factors. The lack of exposure to these real-world challenges results in reduced performance when models encounter them in practice.

In this part of the chapter, we evaluate our approach in the FL environment against the industrial dataset containing various tangled interrelationships. Unlike other approaches, we leverage novel knowledge-based methods, which incorporate detecting and excluding compromised local units from the global model distribution. Based on the extracted knowledge on local updates, and accumulated knowledge on their characteristics and historical behavior, the local units are categorized into trusted and compromised ones. If a local unit is classified as compromised, it is not communicated the actual global model update. In this way, we optimize the communication between the local and aggregation units and further reduce communication load, which in turn allows us to optimize ML training. Further actions on managing compromised local units may depend on the user and application requirements. For example, they can be completely discarded from the aggregation and blocked from accessing the FL system, or they can be supplied with specially constructed fake global updates with the integrated backdoor that allows tracking the activity of these units and infer the adversary capabilities and strategy.

Detecting compromised local units based on their local updates might be performed with various statistical and intelligent techniques. In this section, we develop and evaluate FL system's model and methods for security and privacy enhancement. The employed methods can be categorized as following:

1. Methods for local model's pre-processing. These methods allow comparing and clustering the parameters of all the received local updates parameters. Here, various statistical and/or intelligent techniques might be employed, depending on the employed data and user and application requirements. In the present scenario, we employ K-means clustering technique to cluster the local models transmitted to the aggregation unit.

2. Knowledge-based methods, which allow evaluating trust towards FL local units and excluding potentially malicious units from the global model distribution. In our approach, we employ: **(1)** DQ and data source's security evaluation methods [44, 101, 107], which enable the integration of the characteristics pertaining to the data source into the trust calculation process; and **(2)** Reputation and Trust-based mechanisms [53, 55], introduced in sec. 2.5, that involve identifying trusted and potentially compromised local units based on the model updates they send for the aggregation. The integration of these two knowledge-based approaches allows more informed and accurate trust estimation, and contributes to enhancing FL **(1)** communication, **(2)** security, **(3)** robustness, and **(4)** privacy.

We verify our approach in two distinct scenarios: in regular conditions and under malicious attacks. Initially, we investigate the original collection, without employing intentional data augmentation, quality degradation, or data poisoning attacks. We evaluate how the Reputation and Trust-based mechanisms contribute to enhancing the effectiveness of the resulting model by discarding the local units that provide updates lying out of the major distribution. In addition to this case, we investigate the scenario with malicious modifications to the local training data performed by an adversary. In particular, we study two distinct scenarios of label flipping attacks. In the former one, the adversary maliciously modifies the labels on one of the units by inverting them, i.e., by changing the anomalous label to a benign one, and otherwise. In the latter case, the adversary changes all the labels on one of the units to the anomalous ones. Through our empirical evaluations, we demonstrate how our approach helps to improve both: the FL performance, by discarding from the aggregation the local units providing low quality models; and FL security and robustness, by effectively detecting local units with the compromised training data.

Our solution does not introduce any additional communication burdens on the FL system, as it does not require to transfer any auxiliary data to accommodate the Reputation and Trust calculations. Instead, by excluding the untrusted local units, we optimize the amount of data transmitted over the network and the number of communicated units.

5.3.1 Trust-Incorporating Approaches in Federated Learning

In the recent years, Trust-driven approaches gained attention in the research oriented at enhancing security, reliability, and robustness of FL. The concept of Trust in decentralized computing is not novel, as there exist manifold solutions that leverage Trust and Reputation mechanisms, e.g., to improve security and safety in Intelligent Transportation Systems [55]. FL with limited

communication between distributed and aggregation units and data privacy preservation attracted an investigation of Reputation and Trust incorporation into the process. However, the Trust and Trustworthiness definitions in the literature may vary, and there is no consensus on how to better approach, gauge, and decide if the FL unit is trusted. Below, we review works that approach Trust in FL from various perspectives and discuss our DQ-oriented Reputation and Trust evaluation.

Approaches that employ the concept of Trust to optimize communication in FL are mainly focused on establishing and maintaining trustworthy interactions between the participating network nodes in a decentralized setting. These approaches recognize the importance of secure and efficient communication while preserving data privacy and integrity. For example, Gholami *et al.* [73] leverage the concept of Trust to evaluate the trustworthiness of the network nodes in a decentralized FL setting. The proposed algorithm incorporates trust as a relation between different network nodes, established and updated based on evidence generated during nodes' collaboration. The trust is evaluated based on such metrics as access control, resource allocation, node participation in FL, and others. Trust estimates are used to compute and aggregate trust within the network, ensuring that collaboration is contributive towards achieving specific goals. The results show that the proposed algorithm consistently outperforms the unprotected system, even with varying degrees of malicious attacks severity.

To address the problem of adversaries sending malicious updates in mobile network environments, Kang *et al.* [95] propose to use Reputation as a fair indicator to select network units participating in the aggregation. The scheme assumes the presence of multiple task publishers (i.e., aggregation units), which are responsible for calculating the Reputation for each local unit they contacted according to a Subjective Logic Model [94], and then for combining this value with the Reputation calculated by other task publishers. The local units Reputation is calculated based on positive and negative interactions with the task publishers, i.e., the more positive interaction are committed, the higher Reputation the local unit will achieve. However, to mitigate the local data manipulation attacks, the paper suggests to employ other state-of-the-art techniques, such as Reject on Negative Influence [202] and FoolsGold [70]. To ensure reliable Reputation calculation and management, a consortium blockchain is designed as a trusted and decentralized ledger to record and manage the data owners' Reputation. The solution is evaluated over the MNIST dataset [60] and demonstrated the effectiveness in selecting trusted local units in the scenarios with one and two attackers.

Another communication-oriented FL approach is presented by Roy *et al.* [189], who propose BrainTorrent, a novel server-less peer-to-peer approach, where local units directly communicate among themselves. This decentralized environment eliminates the need for a central server, making it

resilient to failures and eliminating the requirement of having a universally trusted authority. The high frequency of interaction and updates per client in BrainTorrent leads to faster model convergence and accuracy comparable to models trained with pooled data from all clients. BrainTorrent demonstrates its superior performance in whole-brain segmentation tasks, surpassing traditional server-based FL, especially in scenarios with varying data distributions across centers.

Cao *et al.* [28] propose FLTrust method, which uses trust bootstrapping technique to defend against Byzantine attacks in the FL environment. Trust bootstrapping works by first training a "trusted" model on a small number of samples, which is stored on the aggregation unit and called root dataset. The root dataset is a collection of non-malicious samples that are manually gathered by the service provider. The "trusted" model is then compared with each local model transmitted for the aggregation, and the trust score towards each local unit is calculated using the cosine similarity measure with the clipped ReLU, i.e., the closer the local client's model is to the trusted model, the higher trustworthiness it possesses. FLTrust is tested on various popular datasets including MNIST-0.1, Fashion-MNIST [243], CIFAR-10 [113], etc., and is able to achieve comparable accuracy to non-robust FL aggregation strategies, while providing significantly better security against both conventional and adaptive Byzantine attacks.

Rjoub *et al.* [187] propose DDQN-Trust, a trust-based double deep Q-network reinforcement learning algorithm for IoT devices, which involves monitoring their CPU and RAM consumption in order to optimize the local units selection in FL. The goal is to identify devices that exhibit abnormal computation and communication resource utilization behavior in the FL process, including those with excessively high consumption and those whose consumption falls below the normal minimal habitual levels, indicating potential failures. By using a modified Z-score statistical technique, which incorporates the median instead of the mean, the method provides robustness to outliers and calculates the difference of a certain score from the median. This approach helps detecting devices that may not dedicate sufficient resources for FL tasks and identifies potential local units' malicious behavior. The algorithm is evaluated over CIFAR-10 [113] and Udacity Self Driving Car Dataset [221] collections and compared to DQN [158] and random scheduling utilizing various FL aggregation strategies: FedAvg, FedProx, FedShare, and FedSGD. The proposed solution outperforms the baseline algorithms in terms of produced FL model accuracy and contributes to accurate IoT devices selection for participating in FL aggregation.

Blockchain is one of the technologies deemed as a perspective solution to address security, reliability, traceability, and other challenges possessed by FL systems [178, 195, 259]. By leveraging blockchain, the exchange of local model updates among participants can be securely recorded in an

immutable and distributed ledger that might be verified by each FL participant. This ensures the integrity of the updates and prevents malicious tampering. Additionally, blockchain enables consensus mechanisms and smart contracts, which allow for trustless coordination and verification of participants' contributions, promoting fair collaboration. The blockchain's transparent nature also increases accountability and auditability of the actions that enables FL traceability [138]. Bao *et al.* [15] propose FLChain, which is a blockchain-based system to enhance FL reliability. FLChain uses blockchain to store the training data, model updates, and other information about the FL process. This exacerbates tampering with the transmitted model updates or stored local data for the adversary. FLChain employs a reward and punishment system to incentivize local units to provide benign and accurate updates, and to discourage them from misbehaving. The misbehavior detection system uses cryptographic hash functions and public blockchain records to identify units who are sending incorrect updates or who are failing to participate in FL, and punishes them.

Kim *et al.* [110] propose Blockchained FL (BlockFL), which also leverages blockchain technology to enable secure and rewarding exchanges of local model updates between devices. To enhance conventional FL security and reliability, BlockFL replaces the central server with a blockchain network that facilitates the exchange of local model updates between units while ensuring verification and providing corresponding rewards. BlockFL addresses the issue of a single point of failure and extends the federation to untrustworthy devices in a public network through a validation process of local training results. Furthermore, BlockFL incentivizes the participation of devices with larger training sample sizes by offering rewards proportional to their contribution. The decentralized nature of the blockchain network reduces the risk of inaccurate global model updates and ensures the integrity of the training process. However, the paper does not provide empirical evaluations or comparisons with other existing FL secure aggregation methods in terms of ML performance.

Another way to address security challenges in FL is to leverage Trusted Execution Environment (TEE) concept. TEE is a secure and isolated environment within a computing system where sensitive computations can be performed securely and protected from external threats. TEE ensures that the computations and data within it are isolated from the rest of the system (e.g., by the means of Intel SGX [163]), reducing the risk of affecting the data or training process by malicious attacks. TEE permits FL participants to only execute their local learning algorithms within the TEE in order to ensure the privacy and integrity of their training data and computations. Chen *et al.* [36] propose a privacy-preserving FL scheme that guarantees the integrity of the deep learning process, which is based on leveraging TEE by the units. The proposed scheme ensures data privacy and introduces a training-integrity protocol, which allows detecting and mitigating dishonest actions, such as tampering with locally trained models or delaying the local training. The solution's

performance is evaluated over two well-known datasets: MNIST [60] and Foursquare Bangkok location dataset collected by the authors. The experimental results demonstrate the training integrity and practicality of the scheme, however, it is effective only against model manipulations and still vulnerable to local data poisoning or tempering attacks in FL.

There are other studies that leverage the concept of trust in a combination with Differential Privacy, Secure Multiparty Computation, and Homomorphic Encryption. Truex *et al.* [218] define trust as the minimum number of non-colluding parties in the scenario of multiple malicious FL units. Xu *et al.* [247] employs the concept of trusted third party, which is responsible for cryptographic keys generation, storage, and distribution between the FL units. Liu *et al.* [132] also introduces trusted Key Generation Center entity, which is acknowledged by all FL units and is responsible for public and private key management.

As one can see, various approaches and methods mentioned above addressed lacking of the inherent robustness and security assurance instruments in the conventional FL techniques. However, the reviewed solutions commonly possess their own flaws and challenges, which we discuss below.

- One of the major FL challenges is to deal with the heterogeneity and diversity of the data sources, which may have different data distributions, sizes, qualities, and availability. However, in the vast majority of publications, the approaches are verified over the popular state-of-the-art datasets, such as MNIST [60], CIFAR-10 [113], and others, which are well-curated, balanced, and standardized. These data collections do not reflect the realistic scenarios in industry, where the data may be noisy, incomplete, shifted, or changing over time.
- The trust evaluation and quantification in trust-related approaches is commonly limited. Trust is an important factor in FL, as it reflects the reliability, credibility, and even security of the units. However, most trust-related approaches in FL do not provide a clear definition or measurement of trust.
- Trust is not a fixed attribute but a dynamic state that can vary depending on multiple aspects. In FL, trust can change over time due to various factors, such as DQ, ML model performance, or malicious attacks. For example, unit's trust may increase in case of providing better updates, or decrease if it suffers from a malicious attack that poisons the local data and results in poorly performed ML model. However, the majority of trust-related approaches to FL do not provide a mechanism to monitor and trace trust values during the FL process.
- Malicious updates are one of the major threats to FL security and robustness, as they can cause the global model to learn incorrect or harmful information and jeopardize the overall

FL system. Malicious updates detection techniques typically require "a-priori" knowledge, such as ground truth or reference data. This is highly challenging and might not be feasible in real-world scenarios.

In addition, in sec. A.1, in detail we compare our approach to those already proposed in research to enhance security, privacy, and robustness of FL, and summarize the major novelties we introduce. Below we describe our approach capable of addressing the aforementioned flaws and challenges.

5.3.2 Developed Approach to Trust Evaluation in Federated Learning

In sec. 2.4, we described the concept of DQ and some examples of DQ metrics employed in practice. As we demonstrate in our real-world use cases (see sec. 4.2), maintaining high-quality data is of paramount importance for industrial data-driven and ML applications. In our research, we follow the approach to measure DQ in various industrial applications, such as: smartphones, developed by us in [107]; miscellaneous mobile devices equipped with multi-modal sensors (developed in sec. 3.2); and Autonomous Vehicles (AVs) communication, developed by us in [48,55]. In [107], we developed the DQ and security evaluation methodology and calculus for the sensor devices embedded into Android smartphones. In [44, 101] we extended the previously developed DQ calculus to multi-modal sensor devices and developed tools that allow intelligent selection of data sources based on their DQ. In [48], we combined the DQ metric with our Reputation and Trust indicators [53] and employed it to detect AVs that provide incorrect data due to intentional malicious attacks or unintentional failures.

In sec. 5.3.1, we discussed some weaknesses and challenges of the existing trust-driven FL methods, including their verification on rather simplistic limited datasets instead of complex industrial data, ill-defined trust definitions itself, limited trust quantification instruments, lack of trust traceability, and the requirement to rely on prior knowledge in order to delineate between anomalous and benign updates, and others. To address these challenges, we develop, implement and verify our novel Reputation and Trust-based mechanisms, introduced in sec. 2.5, and capable of detecting clients producing anomalous models that could be the results of the shifts between local datasets or malicious actions. In this work, we combine the concepts of DQ and Reputation and Trust evaluation in order to improve the security, privacy, and robustness of FL. To evaluate local units' DQ in the FL setting, we rely on clustering the ML models trained over a local data and provided by the units for the aggregation. We schematically represent the major flows and elements of our approach in Figure 5.5.

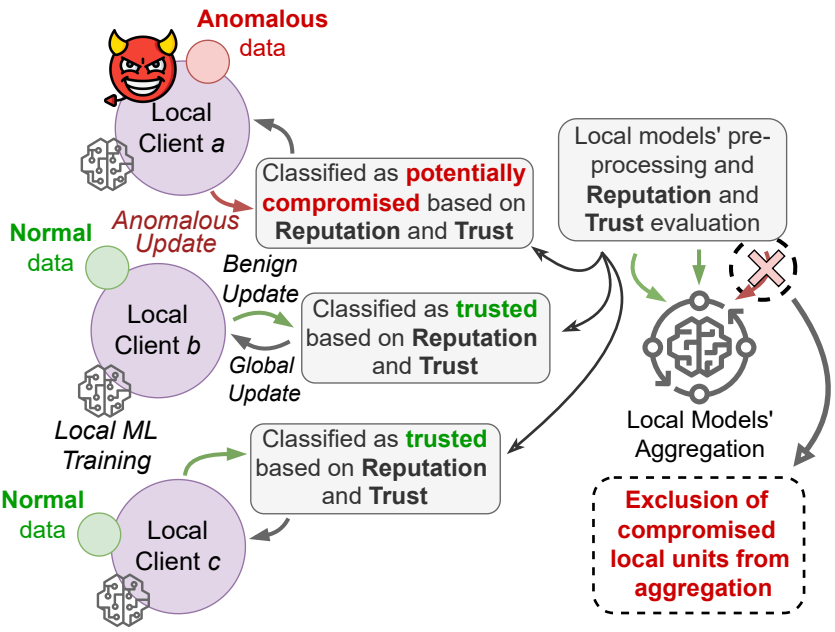


Figure 5.5: Our approach with Reputation and Trust-based techniques for detecting and excluding anomalous local models updates from FL aggregation procedure

The local units might generate data whose distribution strongly deviates from the majority as a result of the data source failure (e.g., improper sensor device operation) or malicious attacks against the local unit and as a result train deviating models. Our clustering approach allows to distinguish these models from the other majority. Hence, we assume that the majority of the models are trained on a high quality data, and the shift in the local data distribution is an indication of the anomalous data patterns. This assumption lets us define the quality of local models based on the local data source's DQ.

To preserve the local data privacy, we utilize models trained over the local data and analyze their parameters distribution [197]. Before the aggregation, we cluster the received local models in their parameter space, and calculate the distance from the cluster's center to each model. Following up our calculus developed in sec. 2.5, we introduce two indicators to evaluate trust toward the FL unit: *Reputation* (R), and $Trust$. However, in this practical use case, we calculate R based on the normalized Euclidean distance d from the major cluster center. The initial value of R is calculated according to (5.3).

$$R_i^{t_0} = 1 - d_i, \;\; R, d \in [0, 1] \subset \mathbb{R}, \tag{5.3}$$

where $R_i^{t_0}$ is the Reputation value calculated for the i-th unit in the initial t_0 time moment (after the first local training round); and d_i is a normalized Euclidean distance from the cluster's center towards the model provided by i-th unit for aggregation. The value of R is updated in each aggregation round, so if $d \geq \alpha$, where α is a specified threshold ($\alpha = 0.5$ in our case), R grows linearly. Otherwise, if $d < \alpha$, R decreases exponentially. The i-th unit's Reputation value for the current time moment t, other than t_0, is calculated according to (5.4) and (5.5).

$$R_i^t = \begin{cases} (R_i^{t-1} + d_i) - (R_i^{t-1}/t), & \text{if } d < \alpha \\ (R_i^{t-1} + d_i) - e^{-(1-d(R_i^{t-1}/t)}, & \text{if } d \geq \alpha, \end{cases} \tag{5.4}$$

$$R_i^t = \begin{cases} 1, \text{if } R \geq 1, \\ 0, \text{if } R \leq 0 \end{cases} \tag{5.5}$$

where R_i^{t-1} is i-th local unit's Reputation value calculated in the previous $t-1$ time moment (in the previous aggregation round). This feature allows to penalize local units heavily for providing low quality models and requires them to submit positive contributions for a considerable time to build their Reputation. We employ this distance to establish the Reputation and Trust indicators that are updated for each local client in each aggregation round.

Based on R, the $Trust$ indicator is calculated, which is a function of R that regulates how the change in R affects the trust toward the local unit. If $Trust < \beta$, where β is the established threshold ($\beta = 0.5$ in our case), the model provided by the unit is excluded from this and following aggregation rounds. In terms of the current aggregation round, the $Trust$ indicator is calculated according to (5.6) and (5.7), where $Trust_i^t$ is the Trust value toward the i-th local unit in the t time moment.

$$\begin{gathered} Trust_i^t = \sqrt{(R_i^t)^2 + d_i^2} - \sqrt{(1 - R_i^t)^2 + (1 - d_i)^2}, \\ Trust \in [0, 1] \subset \mathbb{R} \end{gathered} \tag{5.6}$$

$$Trust_i^t = \begin{cases} 1, \text{if } Trust \geq 1, \\ 0, \text{if } Trust \leq 0 \end{cases} \tag{5.7}$$

Below we discuss the advantages our DQ and trust evaluation-driven FL offers against the existing approaches.

1. DQ-focused: it prioritizes local training DQ, irrespective of how the degradation occurred, whether through malicious intent or unintentional failures. By integrating the local DQ and trust evaluations, our approach enables detecting local units that possess anomalous data and submit harmful updates that degrade the global model performance regardless of the reason. Moreover, our approach opens a great avenue for integrating the knowledge on various data source characteristics that can be employed to evaluate its DQ and security. The use of this knowledge in a combination with other units' trust evaluation methods contributes to a more informed and accurate trust measure. For example, the evaluation of DQ might not be limited to only clustering the model updates, and might include considering data source characteristics (e.g., the range, resolution, and accuracy of the sensor device that produce the data), as proposed in sec. 3.4.2. In addition, the data source computational platform security characteristics, which are also used in sec. 3.4.2, might be employed for the trust evaluation process. The integration of knowledge on data source characteristics, the models clustering results, and the calculated Reputation and Trust indicators enable more informed and accurate local units' trust estimation, which contributes to better detection of those units who possess anomalous data and train low quality models based on this data.

2. Trust-based filtering: instead of selecting local units for aggregation [95], we decide which units should be discarded from aggregation based on trust towards them. We develop calculus that quantifies Reputation and Trust indicators, and allows to evaluate them in a specified numerical range. This allows the FL system user to establish a specified level of trust deemed acceptable for their application. This makes our calculus more flexible and personalized according to the particular data, ML model, and context requirements. Untrusted local units are excluded from further communication, preventing them from receiving further global updates, which enhances privacy and security of FL.

3. Historical tracking: this feature allows accumulating and tracing the changes in trust toward local units' over time in FL system. The retrospective data on the quality of models local units provide for aggregation is employed for trust evaluation, enabling the identification of untrusted units based on their "prior behavior". This feature might be highly useful and employed, for example, for extensive security analytics and audit [173].

4. Unsupervised clustering: our approach does not require any prior knowledge on the training data and its distribution. It solely relies on the models' updates sent by the local units to the aggregation unit. These updates are clustered in an unsupervised manner, and Reputation and Trust indicators are calculated based on these clustering results. Our approach does not need access to ground truth or the local data distribution in advance for trust estimation,

which allows to enhance FL privacy. In addition, the employment of unsupervised clustering requires only the model updates as an input, which makes the approach adaptable to manifold data types and FL applications.

Despite the advantages listed above, our approach does have several limitations. As in majority of other existing FL approaches, it requires an aggregation unit responsible for calculating the Reputation and Trust indicators, which introduces a centralized point of failure into the FL framework. Additionally, our solution may not be effective when a significant portion of local units possess low quality training data, as it heavily relies on the availability of high quality models for accurate trust evaluation.

5.3.3 Developed Approach Verification on Industrial Application

Industry Case Study Setup

To verify our novel trust-based approach to FL, we employ the industrial dataset provided by SWIFT[3] and employed in U.S. PETs Prize Challenge[4] hosted by NIST and NSF. The collection incorporates about 4 million records on financial transactions between various international banking organizations performed over a 30 days interval. The records are composed of more than 20 attributes and contain anomalous transactions, which are the targets for the ML classification. Initially, we analyze the data and pre-process it in order to remove unimportant attributes that have no effect on the target variable. New attributes like "*sender_currency_frequency*" and "*sender_currency_amount_average*" are feature engineered from existing attributes to better describe a particular transaction in the context of ML classification. As the original dataset is highly imbalanced with almost 95% of the data being benign transactions, its direct use without re-sampling may result in a low ML performance in the testing stage. We utilize the SMOTE library [198] to modify the training data in order to avoid oversampling and then to normalize it. This results in a balanced dataset that can be employed to train more generalizable model.

To select the appropriate basic ML topology, we compare various models based on their performance achieved after a centralized ML training: DNN and a Random Forest (RF) classifier. The performances demonstrated by these two ML architectures are given in Figure 5.7. From these results, one can see that DNN is able to achieve about 74% AUC in contrast to ≈63% showed by

[3] `https://www.swift.com/`

[4] `https://www.drivendata.org/competitions/98/nist-federated-learning-1/page/522/`

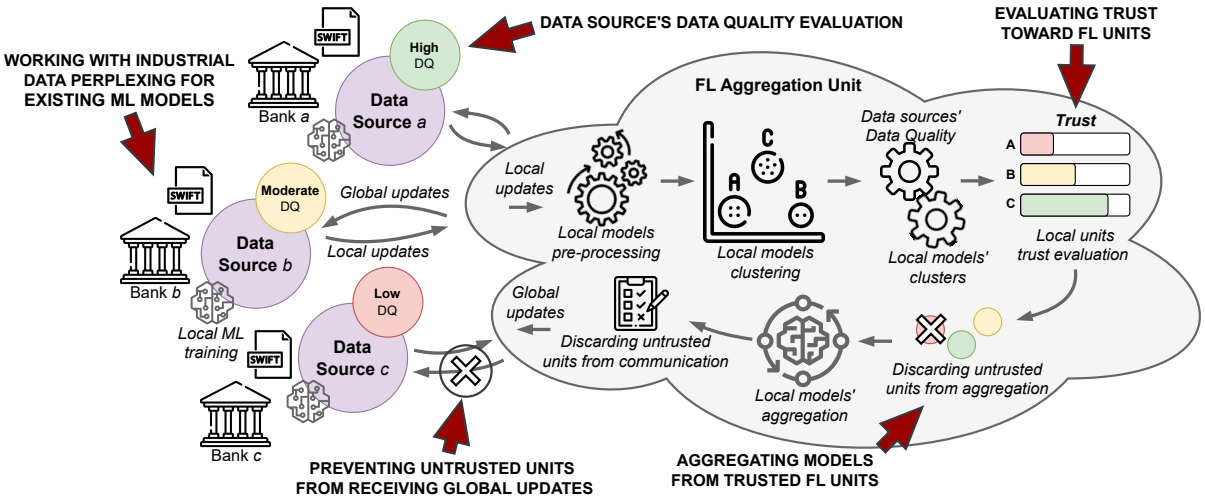


Figure 5.6: The proposed DQ-driven trust evaluation approach for robust FL in industrial applications on the example of the employed dataset provided by SWIFT

RF. Hence, we select DNN for the further FL empirical study. The best DNN topology, based on the the Area under the ROC Curve (AUC) metric, incorporates four dense layers: the first layer has 200 neurons, takes an input shape of 9 features, and employs the Rectified Linear Unit (ReLU) activation function; the next three layers have 100, 50, and 25 neurons, respectively, all employing ReLU as well. We also employ a dropout rate of 0.5, which results in preventing model's overfitting. The last layer has a single neuron with the Sigmoid activation function, which is typically used for binary classification problems. For our FL training, we divide our data into multiple cohorts distributed over 10 distinct FL units. The data is divided into cohorts based on the records pertaining to the particular bank origin. Each of these cohorts has a roughly equal number of records and is composed of similar attributes. In Figure 5.6, we schematically represent how the DQ and trust evaluation steps are integrated into the FL process in our industrial use case and how they benefit it.

Federated Learning with Trust: Data Poisoning Attacks Detection

To recreate data poisoning attack scenario, we maliciously augment the original SWIFT data collection with label flipping attacks on two of the local units. Specifically, we explore two types of label flipping attacks. On unit 2, we invert labels in the local training data, i.e., we change the labels pertaining to anomalous transactions to benign ones and vice versa. On unit 3, we change labels of all the transactions in the stored local training data to anomalous ones. We employ these

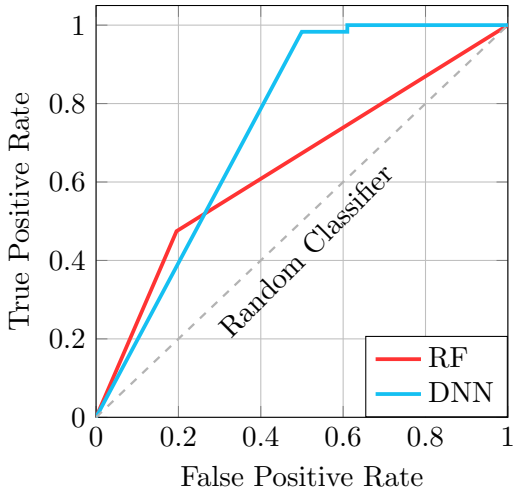


Figure 5.7: Performance comparison of various models based on the ROC curve: DNN vs RF trained in a centralized manner

two types of malicious manipulations to augment local data on unit 2 and unit 3 respectively, while leaving the other clients' data intact. We assume that the attackers have full control over the local data and models of these clients, but not over the communication or aggregation process.

Initially, the constructed DNN model is distributed across 10 distinct local clients from the aggregation server via encrypted socket communications. After the model is distributed, the local training process is initiated by each unit. The model is trained for 100 consequent epochs. In each training iteration, the fitting of each local model on a batch of unit's data occurs locally. The fitting process updates the local model's weights based on the data samples it encounters. The training data on each client is split into 80% training and 20% validation subsets. The early stopping procedure is used with a patience of 7 epochs in order to prevent overfitting as well.

At the end of the local training iteration, the parameters of the produced model are conveyed to the aggregation unit. In this case, we evaluated the Reputation and Trust indicators for the units submitted models for aggregation after the first local training round. As can be seen from Table 5.1, units 2 and 3 received the lowest trust values, which indicates that they provided models that lie out of the major distribution. In Table 5.1, the units are sorted according to the trust values towards them calculated after the first local training round. The small differences in trust values between the units 4, 10, 3, and 2 are justified by the retrospective nature of the Trust indicator – it needs to accumulate historical data during several aggregation rounds to reflect changes in the Reputation more accurate. In this particular case, just after the first training round, it is

Table 5.1: Reputation And Trust values calculated for each local unit after the first local training round

d **from cluster center**	**Client ID**	R	$Trust$
0	Unit 9	1	1
0.139	Unit 7	0.860	1
0.184	Unit 6	0.815	0.893
0.289	Unit 5	0.710	0.596
0.296	Unit 8	0.703	0.576
0.325	Unit 1	0.674	0.494
0.444	Unit 4	0.555	0.156
0.461	Unit 10	0.536	0.109
0.866	**Unit 3**	**0.133**	**0**
1	**Unit 2**	**0**	**0**

more reasonable to use Reputation as the decision-making criteria to detect compromised local units. Our approach is capable of detecting the compromised units even before aggregating the local updates, which means that, based on the Reputation and Trust values, they can be excluded from the further aggregation procedure and from the global model distribution. Discarding from the aggregation procedure will prevent the influence of harmful updates on the global model, and preventing compromised local units from receiving the global model will enhance the security and privacy of FL.

Federated Learning with Trust: Training in Normal Conditions

In this case, we continue to train the FL model under normal conditions after excluding the poisoned units. The same communication channel, over which the models were sent for aggregation, is employed for global updates distribution. The local updates are aggregated using the FedAvg function [143] to produce a global model. This global model is then distributed back to the clients, and a new local training round is initiated. The resulting ML global model is updated with the weights produced after the aggregation. This process is iterated in each aggregation round.

First, we evaluated the model trained in a conventional FL manner [112], without employing the Reputation and Trust indicators. Orange line in Figure 5.8(a) represents the performance demonstrated by this model. Comparing Figures 5.7 and 5.8(a), one can see that distributing data over

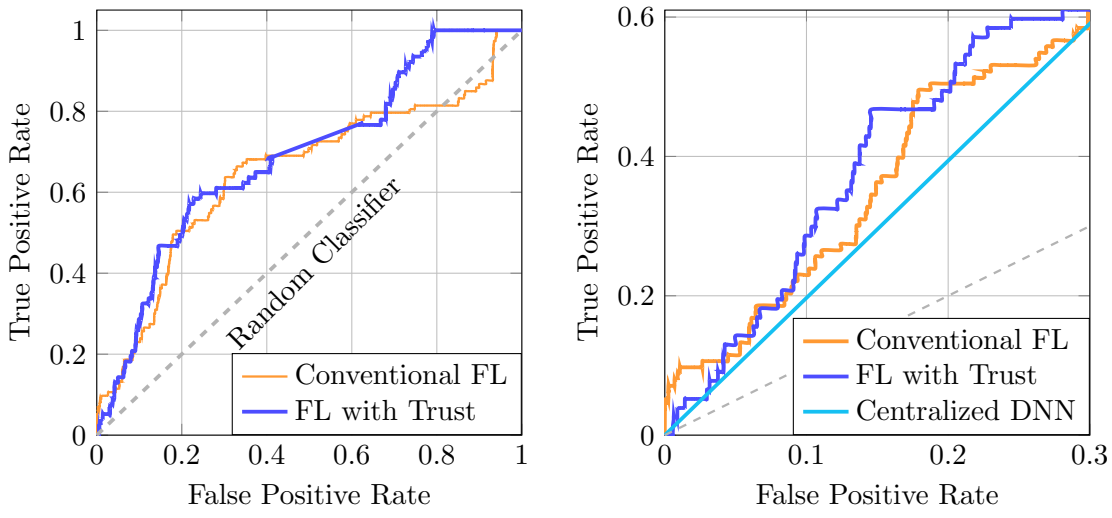


Figure 5.8: Performance comparison of various models based on the ROC curve: (a) – DNN trained in a FL manner with and without the proposed Reputation and Trust-based indicators; (b) – performance of the advanced and conventional FL model vs centralized ML model within the most important in practical applications False Positive Rate region

the local clients and aggregating their models results in commensurate classifications in contrast to the centralized model scenario. According to the results obtained, FL model demonstrated AUC of around 72%, which was comparable to one achieved in the centralized learning case.

Then, we introduced the Reputation and Trust indicators into our FL process in order to detect and exclude the units that employ deviating local data for training. To calculate the Reputation values, we performed clustering of the local models' parameters transmitted to the aggregation server. Using K-means algorithm, we found the major cluster's center and calculated the Euclidean distances d from this center to each of the models. We normalized d in the range between 0 and 1, and employed it to initialize the Reputation indicator R for each client. The initial Reputation was calculated according to (5.3), which means that the farther the model lies from the cluster center, the lower Reputation it receives. Then, the Trust indicator was calculated based on the R, as described in sec. 5.3.2. The Reputation and Trust indicators were updated in each aggregation round. In our experiments, we excluded the local units whose $Trust < \beta$, $\beta = 0.5$. This means that the excluded local clients were not supplied with the produced global model in the current aggregation round, and their models were not aggregated in the future rounds as well.

During our FL experiments, two local units were excluded from the further aggregation rounds,

as Trust towards them dropped below the established threshold. Blue line in Figure 5.8(a) represents the performance results for the FL process that employs the proposed Reputation and Trust mechanisms. In Figure 5.8(b) one can see the scaled version of these results over the more practical False Positive Rate values interval. The produced model evaluated over the testing data was able to achieve AUC of 77%, which outperformed both models trained in centralized and conventional FL manners. Based on the results, the employed Reputation and Trust-based mechanisms demonstrated the FL model performance improvement.

5.4 Discussion

In this chapter, we focused on addressing the challenge of varying input DQ over ML applications' operation through mechanisms applied in the learning stage. We demonstrated that open source industrial domain pre-trained models could be employed in these applications' design but require further re-training to improve their robustness to input DQ drops over operation. For re-training purposes, we investigated two approaches, TL and FL. In sec. 5.2.1, we specifically examined the case of varied DQ in an ITS, where real traffic sign images were transmitted over a wireless network with Quality of Service degradation. Through our empirical evaluations, we found that both strategies can effectively mitigate the impact of DQ variations. When only the high-quality data was available for training, FL demonstrated greater robustness to testing DQ variations compared to TL. Training ML models solely on the low DQ, without including high-quality data, resulted in higher ML performance. Additionally, we observed that FL with the GM as the aggregation strategy exhibited greater robustness to DQ variations, so we recommend to employ GM whenever the corresponding resources are available. By utilizing GM, the FL aggregator can effectively produce a more reliable global model under adverse local data conditions, making FL with GM a preferable approach when DQ is a concern. FL's ability to leverage distributed data sources and aggregate models from multiple participants enabled it to handle variations in data distributions more effectively.

Overall, our investigation results on the example of the ITS application for both TL and FL cases emphasized the critical role of DQ in re-training available ML models in order to make them meet the requirements of industrial applications. ML models specifically trained to address DQ variation may exhibit lower performance not only on degraded data but also on high-quality data compared to models trained solely on high-quality data. This underscores the need to address and assure DQ by employing additional mechanisms in the ML execution stage.

In sec. 5.3, we proposed, developed, and verified our novel approach to improve communication, security, and robustness in FL with Trust and DQ evaluation for distributed industrial applications. The approach integrates the developed in sec. 2.4 data source's DQ evaluation with the Reputation and Trust-based mechanisms, presented in sec. 2.5, which allows quantifying and calculating trust towards the FL unit based on its contributions towards the FL performance throughout the learning process. In contrast to other existing trust-driven FL approaches, ours leverages the following advantages. First, it relies on unsupervised clustering of the local models, provided for aggregation, which avoids the necessity of prior knowledge on the local data or access to it. In addition, it formally defines the concept of trust and enables numerical estimation of the trust value, which extends its capabilities and makes it more flexible for various industrial applications and their requirements. Moreover, our approach allows to monitor how the trust value of each local unit deviates over the FL system operation, which might be employed for extended security analysis and audit. Our solution enhances several FL aspects:

1. Communication, as it allows to exclude untrusted local units from the interaction, which reduces the amount data to be conveyed over the network and the quantity of local units to communicate with.

2. Security, as it permits detecting potentially compromised FL local units based on the quality of models they provide for the aggregation, their local DQ characteristics, and the trust towards them accumulated during the FL operation.

3. Robustness to poor local DQ, as detecting and excluding the models provided by these FL units from the aggregation increases testing performance of the resulting global model.

4. Privacy, as detecting and excluding potentially compromised local units from the current and the following aggregation rounds prevent them from receiving the global models, which reduces the risk of implementing data inference and membership attacks.

In contrast to many other studies aimed at enhancing FL security and robustness, we verified our approach on the industrial SWIFT financial transactions collection, which is initially highly imbalanced and possesses tangled data interrelationships.

5.5 How Robustness through Learning Helps to Enhance ML Robustness in Execution

Enhancing ML robustness through learning is important for ML models as it enables them to cope with real-world challenges and uncertainties in terms of the input DQ. As we discussed in sec. 3.1, in the real-world, input data is subject to various usually random factors that can lead to noise, incompleteness, inconsistencies, imbalance, or out-of-distribution samples, which can affect the model's performance and generalization ability. In sec. 2.3, we developed and presented the approach, aimed at adjusting MLIN structure based on the requirements to the ML application performance. This approach is by definition reactive, which means that the adjustment is enforced when the pattern of interest, provided by the requirements, occurs, e.g., when the ML application performance drops below the established threshold. The reactive ML robustness assurance measures, applied in the ML execution stage, are advantageous in terms of their capabilities to adapt to the changes in the operation environment. For instance, they commonly not require to interrupt the ML application to update the requirements in contrast to the preventive measures like re-training or ML model reconfiguration. However, as we demonstrated in our examination, some measures applied at the learning stage commonly contribute to improving the overall ML performance and increasing its tolerance to adversarial attacks.

In our work, we see multiple advantages of combining methods applied in the learning and execution stages to further ensure and enhance ML robustness, especially for real-world applications. By incorporating preventive measures during the training phase, which are focused on pre-training and fine-tuning the model parameters and architecture, and reactive measures during execution, which address unseen challenges in real-world data, a more comprehensive and holistic approach to robustness emerges. This synergy equips MLIN systems to handle a diverse range of scenarios and data variations effectively. Requirements to ML robustness depend on the user and ML application and can vary significantly across various domains. Methods applied at the learning stage can be selected based on insights gained from reactive measures, and vice versa. For instance, if the particular DQ variation pattern occur in the input data, samples that represent this pattern can be collected and employed for the ML model re-training. This adaptability ensures that the ML system can flexibly respond to specific needs and evolving conditions.

MLIN systems can face unexpected challenges, and no single approach can guarantee complete robustness to DQ variations. A combination of methods allows to decrease the ML performance degradation risks more comprehensively. Measures applied at the learning stage enable the initial generalizability and robustness to the known DQ variations, while reactive measures continuously

adapt ML applications to dynamic conditions, enhancing their ML robustness over the whole operation period. While preventive measures may be computationally intensive and require the MLIN system interruption, reactive measures are often more resource-efficient, as they target specific issues as they arise, conserving computational resources. Hence, utilizing both methods employed in the learning and execution stages aligns with the challenges of the real-world, where ML applications need to navigate uncertainties and respond to unforeseen events.

Our research contributes to enhancing ML applications' robustness to DQ variations, especially for ones employed in industry, where the DQ might highly deviate. TL and FL, despite their distinct origins and purposes, offer valuable approaches to address this challenge. By leveraging these approaches, ML applications can better adapt to varied DQ conditions, ultimately leading to improved performance and increased ML robustness against the inputs of varied quality.

5.6 Conclusion

To assure their robustness, industrial ML applications need to navigate the challenges of frequently unpredictable landscape of real-world DQ variations. In this chapter, we explored the ML robustness enhancement methods and techniques with a specific emphasis on strategies deployed during the learning phase. Through our industrial use cases leveraging real-world datasets, we presented practical insights into the efficacy of TL and FL as a measures to enhance ML robustness employed during the training process. Our findings revealed that these techniques, though not originally designed to enhance ML robustness to DQ variations, exhibited effectiveness in mitigating the impact of such variations. Below we represent the major findings and contributions developed in this chapter, each of significant import to enhancing ML robustness in MLIN systems.

- **TL to enhance ML robustness:** One of our major contributions is the exploration of TL as a robust solution for addressing DQ variations. Through our empirical study, **we showcased the effectiveness of employing the low DQ for the re-training of the model initially trained on a good DQ only.** This approach has proven its effectiveness in **enhancing ML robustness toward DQ variation during the execution stage.** Re-training on degraded data helped the model to adapt to the challenges posed by varying DQ, resulted in improved performance over the varied DQ.
- **FL to enhance ML robustness:** In our investigation, alongside the TL we also analyzed the effectiveness of FL in terms of enhancing ML robustness. Specifically, **we highlighted**

the substantial advantages of employing GM as the aggregation strategy in FL, given sufficient computational resources. This approach empowers FL aggregators to generate more robust global models. Our findings **emphasize the significance of employing GM strategy in the conventional FL to mitigate the impact of DQ variations** on the resulting model performance.

- **Analysis of TL and FL:** Based on the obtained empirical results, **we analyzed strengths and weaknesses of both TL and FL in the cases of varying training DQ.** FL, particularly when re-trained on high-quality data, **demonstrated superior tolerance to DQ variations.** Its inherent capacity to employ distributed data sources and aggregate models enabled greater adaptability to shifting data distributions. However, **TL demonstrated comparable results in specific scenarios.** These findings are beneficial for the practitioners in the area, as they provide reasoning to choose between these two strategies, depending on the unique demands of their application.

- **Enhancing FL Security and Robustness:** In addition to analyzing the effects of TL and FL, we focused on further enhancing FL through the incorporation of innovative Reputation and Trust techniques. Our approach integrates DQ evaluation and Reputation and Trust indicators, which **enables us to measure and quantify trust based on training data possessed by local units without accessing this data.** This integration **strengthens FL in several dimensions, including the reduce of communication burdens, and enhancing security, robustness, and privacy.** Our approach allows **detecting compromised local units, and excluding them from the training procedure,** which thereby **enhances the overall FL security.**

- **Holistic Approach to ML Robustness:** Considering our empirical results, we developed and represented **the holistic approach to ML robustness.** By integrating techniques applied during both the learning and execution stages, **we introduced a complex strategy that enhances ML application robustness to DQ variations in diverse scenarios and conditions.** This synergy bridges the gap between preventive measures implemented during training and reactive ones, applied in the execution stage. This holistic approach **allows to cover challenges arising on various MLIN stages and improves robustness to DQ variations over extended periods of operation.**

Chapter 6

Conclusion

AI/ML and communication are two fields that have undergone tremendous evolution in recent years, transforming our world and reshaping our everyday lives. The emergence of intelligent data-driven systems, which actively have been employed in such diverse industrial domains as healthcare, transportation, and military, has enabled new possibilities and opportunities for solving complex and challenging problems, improving service efficiency and quality, and creating value and impact. These systems commonly leverage large-scale and diverse data sources, such as sensors, cameras, mobile devices, and online platforms, to make decision based on the data being continuously produced. According to a report by IDC, the worldwide spending on AI-centric systems is expected to reach $154 billion by the end of 2023, which is a 26.9% increase over the amount spent in 2022, and the global market is forecasted to continually grow and reach $300 billion by 2026[1].

However, applying ML systems in practice also poses significant challenges that need to be addressed. One of the main challenges is the DQ, which can be defined as the degree to which data satisfies the requirements of the application and the end user. DQ can be affected by various factors, such as unpleasant environmental conditions, cyberinfrastructure failures, or malicious and adversarial attacks [46]. Poor DQ can compromise the performance, robustness, and reliability of ML systems, leading to inaccurate outcomes and poor decisions that can result in harmful consequences. Therefore, enhancing and maintaining the required DQ is essential for data-driven systems. The challenge of maintaining the required ML performance in the event of DQ drop became known as the ML robustness problem. Our current situation review, conducted in sec. 2.1

[1] https://www.businesswire.com/news/home/20230307005050/en/Worldwide-Spending-on-AI-Centric-Systems-Forecast-to-Reach-154-Billion-in-2023-According-to-IDC

and 2.2 where we classify and discuss the existing approaches, demonstrated that ML robustness is defined, evaluated, and quantified in various ways. The most widely adopted definition refers to the ability of an ML system to maintain its performance when faced with uncertainties or adversarial effects on the input data. In this case, ML robustness is typically evaluated based on the distance measure between the original and the perturbed data sample, where the perturbed data is obtained by applying some noise, transformations, or adversarial manipulations to the original data. The larger the distance that an ML system can tolerate with the minimal performance loss, the more robust it is. However, this distance-based approach does not fully capture the quality of the data input, as DQ is a complex concept that encompasses intrinsic and contextual attributes related to the data. Moreover, this approach is not applicable in real-time ML systems, where ML system has to process the continuous data flow. In this case, the "original" data is not available, and it is not possible to calculate the distance measure between the "original" and the actual data.

In this work, we addressed these challenges by pursuing the integration perspective in the consideration of modern ML systems. We focused on ML systems that are integrated with network infrastructure and incorporate various components from different domains, such as data sources, network devices, and ML applications. We referred to these systems as ML with Integrated Network (MLIN) systems. In practical applications, MLIN systems commonly rely on data collected by the sensors and then transferred over the network to be processed by the ML application that can be implemented on the cloud. At present, MLIN systems are often designed, implemented, and maintained with a focus on separate components without considering the interrelationships between them. Unfortunately, this approach is limited by the scope of each particular component, which leads to system disintegration and prevents employing the interactions between the MLIN components in order to address the DQ variation and ML application robustness challenges.

We proposed and developed a novel approach to assure robustness towards the DQ variation in MLIN systems, which is based on using the relationship between the input DQ and ML performance demonstrated over this input. In contrast to others, our ML application robustness evaluation enables measuring robustness during the ML execution phase when the "ground truth" data is not available. Our approach adapts the developed comprehensive DQ calculus that integrates intrinsic and contextual DQ metrics with the security of the data sources. The developed ML application robustness calculus enables quantifying the robustness to a particular data sample or to a set of the processed data samples over the MLIN operation period. We verified the solutions we developed on a multitude of real-world use cases, which demonstrated the high practical value of the obtained results. Alongside the solutions aimed at ensuring ML application robustness, we also developed methods to enhance the security and privacy of the industrial ML applications.

Considering our empirical results, we developed and represented the holistic approach to ML application robustness evaluation. This integral approach allowed us to cover the challenges related to the varied DQ arising throughout distinct data life-cycle stages and MLIN components. With the MLIN integration aspects in mind, we developed our novel approach to ML application robustness evaluation that is capable of representing the vital relationship between the input DQ and ML performance. We conducted a thorough empirical study, which involved employing real facilities and instrumentation. The obtained results demonstrated our approach's feasibility and advantages of implementing it in practical applications. Alongside ML application robustness definition and calculus itself, we developed approaches to ensure ML application robustness in MLIN. We verified these approaches in multiple real-world use cases that demonstrated their effectiveness. Our results and findings accommodate our answers to the important RQs we formulated in sec. 1.1. Below, relying on all the contributions, results, and findings developed in our work so far, we derive our answers to the RQs posed.

- **RQ1: What methods and techniques should be employed to design the integrated MLIN system and formalize the interrelationships between its component's interactions and ML application robustness?**

  Our research yielded a comprehensive understanding of a generic integrated MLIN architecture. For the integrated MLIN system design, system analysis methods and techniques should be employed, as they enable to holistically incorporate various components and reflect their intricate relationships within an MLIN environment. The architecture can be described as an organized system comprising components such as data sources, network facilities, ML application, ML application performance evaluation, and MLIN adjustment feedback. Our designed architecture not only identifies these components but also elucidates their composition and functionality within the MLIN system.

  - **What classification is feasible for the existing approaches to define, assess, and quantify ML application robustness, and what major benefits and disadvantages they posses?**

    In sec. 2.2, we classified the existing approaches to address ML application robustness according to the component they are concentrated on: data, ML model, and network. In the context of our work, we see this classification feasible as we consider MLIN from the system integration perspective, and we emphasize the importance of considering the interrelationships between various MLIN components instead of considering each component in separation. The majority of the reviewed approaches define and quantify ML

robustness based on the distance metrics between the original and adversarial or manipulated data, and its relationship with the ML performance demonstrated over this data. One of the major advantages here is their ability to establish the practical threshold on how the particular ML model is robust to a certain type of data manipulations. However, they posses several disadvantages, such as: the need to have "ground truth" data in order to calculate the distance from, which does not allow to determine the ML robustness for systems operating in real-time; in contrast to the DQ indicator we employed in our approach, the distance measure does not allow to comprehensively evaluate the quality of the data, especially in terms of how it satisfies the user and application requirements; as well as other disadvantages we analyze and discuss in chapter 2.

– **How effective are the existing approaches to ML robustness in addressing the DQ variation in MLIN systems?**

  The existing approaches to enhance ML robustness are usually concentrated on separate MLIN components, e.g., data cleaning or filtering, ML model re-training or hyperparameters tuning, etc. However, as we discussed in sec. 2.4, DQ in MLIN might be affected by distinct components on various data life-cycle stages. In addition, in real-time systems, DQ variation factors are very dynamic and conventional approaches focused on separate components might not be effective in ensuring ML application robustness in this case. Additional measures have to be implemented in order to ensure ML application robustness in MLIN systems.

– **Which methods and techniques should be employed to design the integrated MLIN system architecture and incorporate the ML adjustment feedback component into this architecture?**

  Since we considered MLIN from the system integration perspective, system engineering and analysis methods and techniques should be employed to reflect the interrelationships between MLIN components, and represent how they affect ML application robustness.

– **What calculus and metrics are feasible to measure and represent the DQ variations in the integrated MLIN system?**

  In this work, we followed up our research [107] and developed DQ generic calculus that considers DQ variation caused by various MLIN components. We formulated the overall DQ as the integral indicator, which incorporates metrics related to each MLIN component that processes the data during the system operation. This approach allowed us to encompass the dynamic DQ variation factors in MLIN related to each of the components, and take them into account while the overall DQ evaluation.

– **How do the introduction and use of Reputation and Trust metrics contribute**

to increasing MLIN security?

In our research, we followed up our investigations [48,53,55], and developed the Reputation and Trust generic calculus, which we employed for enhancing security and robustness of FL. The Reputation and Trust metrics allowed us to effectively detect local units possessing anomalous training data and discard them from the aggregation and further FL communication. Hence, applying FL equipped with our Reputation and Trust indicators allows to enhance MLIN system security.

– **What calculus should be developed to effectively measure ML robustness that integrates the input DQ and ML performance demonstrated over this input?**

The developed calculus should address the disadvantages of the conventional distance-based ML robustness measures, such as the requirement to have a "ground thruth" data and the incapability of providing quantifiable ML robustness metric in real-time conditions. We defined ML application robustness as the relationship between the input DQ and the ML performance demonstrated over this input. This approach allowed us to address the disadvantages faced by conventional approaches by considering the interrelationship between the input data and its variation over the MLIN operation period, and the ML performance demonstrated over this input.

- **RQ2: How to improve ML application robustness by data sources selection and adaptation?**

Conventional approaches deem DQ as the complex indicator that combine intrinsic and contextual metrics. In our work, we integrated the conventional DQ with the data sources security metrics to consider them in the DQ evaluation. In sec. 3.1, we discussed various security factors that can deteriorate DQ and, in turn, the ML application performance if not paid the proper attention. DQ variation, caused by the security violations, results in ML application robustness decrease. To address the data source-related security challenges, an approach is needed that takes into account the security characteristics of the data source and enables selecting those data sources that provide the best possible DQ. Alternatively, depending on the MLIN system configuration, the data sources can be adapted to the current conditions in order to improve the DQ they provide. In practical applications, the challenges of data sources selection or adaptation are substantially exaggerated by the multi-modality and diverse platforms these data sources are embedded into. To address these challenges and to improve ML robustness, a novel methodological framework should be developed for automatic data sources selection in MLIN, which enables selecting or adapting data sources

providing the best possible DQ.

- **Which novel methodological framework should be developed for automatic data sources selection in MLIN that provides integration of platforms, data sources modalities, and diverse metrics?**

  In sec. 3.2, we proposed and developed our novel multi-level Integration Framework for Data Sources Selection, which enables selecting and adapting multi-modal data sources embedded into diverse platforms. The Framework employs GA for the real-time selection and our DQ calculus that integrates quality of the provided data with security metrics of the data source platform in order to provide an integral indicator utilized as a major optimization function. The Framework contributes to improving ML application robustness by selecting data sources that provide DQ resulting in better ML performance.

- **How to verify the developed novel framework in practice?**

  To verify the developed Framework, we designed a practical use case related to the ML application used for the medical research based on the data crowd-sourced from the diverse mobile devices. To facilitate our use case, we collected an extensive knowledge base on technical and system characteristics of around ten thousands diverse sensor-embedded mobile devices. We employed the collected characteristics in our empirical study, in which we evaluated our Framework's effectiveness and efficiency against the conventional brute force search. As we employed data collected from an extensive number of real mobile devices, we deem our use case as a valid practical verification of the developed Framework.

- **How effective and efficient is the developed novel Framework in comparison to the conventional approaches to data sources selection?**

  Our verification on the practical use case showcased that the developed Framework demonstrated similar sensor selection effectiveness while showcased substantially higher efficiency compared to the conventional brute force-based technique. The Framework with the incorporated GA-based sensor selection allowed to return the results comparable in terms of the effectiveness on 97% faster on average.

- **How to assist the community in adopting the developed framework in practice?**

  To facilitate the use of our Framework in practice and accommodate the benefits gained from it by the community, we implemented the developed sensor selection methods and techniques in multiple publicly available Android OS applications, described in sec. 3.6.

These applications enable employing our solutions both for the research or personal purposes related to sensor selections objectives.

- **RQ3: How to improve ML application robustness by the network infrastructure adaptation?**

  In sec. 2.3, we proposed and developed the MLIN feedback component as a vital element within the integrated MLIN architecture. It relies on the requirements provided by the user and application to perform constant monitoring and respond to the variations in DQ. It acts as a reactive control mechanism, ensuring that the MLIN system remains adaptive and responsive to the DQ variations. The feedback is aimed at restructuring MLIN components and adapting them in order to improve ML performance to the level specified by the provided requirements, which also contributes to enhancing ML application robustness. The results of the practical examples we developed illustrated how this feedback mechanism can be effectively realized in MLIN systems.

  - **How to realize the integrated MLIN architecture in practice?**

    In sec. 2.3.5, we provided theoretical description of the integrated MLIN architecture. Further, in sec. 4.2.1, we implemented this architecture in practice to facilitate our real-world use cases investigation. To accommodate the practical realization, we employed the POWDER platform, which provides remote access to the real network facilities and allows to use them for our research purposes. We setup the POWDER profile with the following components: data source node, responsible for retrieving the data and transmitting it over the communication channel to the receiver; the LTE communication channel between the data source and the receiver; and the receiver that incorporated the installed ML application responsible for processing the input data and returning the ML performance results. This setup accommodated the transition of the theoretical integrated MLIN architecture into practice and allowed us to investigate our real-world use cases.

  - **How to examine the interrelationships between various MLIN components and their impact on the input DQ and ML application performance demonstrated over this input?**

    Our empirical study, conducted across diverse real-world use cases, provided critical insights into the interrelationships between various MLIN components. We explored how changing network infrastructure conditions within MLIN (e.g., packet loss and network resources availability) influenced DQ and ML performance. In our research, we utilized

real-world data collections, industrial and foundation ML models, and real network facilities. In all the considered use cases, we found that DQ variation, caused by the network QoS changes, resulted in the ML application performance degradation, and the degree of this degradation deviated depending on the particular application and domain.

- **How to incorporate MLIN feedback adjustment component into the MLIN architecture in practice?**

  In sec. 4.3, we demonstrated a practical example of how the MLIN feedback adjustment component can be realized in practice and incorporated into the real MLIN architecture. Based on the interrelationships between MLIN components, studied in our real-world use cases, we designed the system that employs rule-based logic to generate recommendations aimed at ensuring ML performance in the cases of DQ variation due to changing network conditions. This rule-based system employed the requirements provided by the user and application, to which the ML performance should satisfy. In our example, we demonstrated how the network protocol adjustment, in case of the increasing packet loss, contributed to enhancing ML performance in the case of DQ variation.

- **What flexible and feasible calculus and indicators should be developed to realize the introduced ML application robustness generic calculus in practice?**

  In sec. 2.6, we developed novel ML application robustness definition and generic calculus. However, depending on various practical scenarios, this calculus can be adjusted or modified in order to satisfy the user and application requirements. In 4.4, we enhanced the generic calculus functionality and demonstrated what indicators can be developed for various practical cases. On the example of sound classification, we showcased two types of ML application robustness indicators: local ($\mathcal{RB}_L$) and global ($\mathcal{RB}_G$). The former one enables representing how the ML application robustness changes between the previous and the current time moments. The latter one accumulates all the changes between the input DQ and the ML performance demonstrated over this input from the system initialization moment. Depending on the practical scenario, the user may select the indicator which better covers their needs. Additionally, the generic calculus is flexible enough to be modified according to the particular application and its requirements.

- **RQ4: How to improve ML robustness by enhancing effectiveness, security, and privacy over the ML training phase?**

  In our work, we investigated several strategies applied at the ML training phase that enable further enhancing ML application robustness during the system execution. TL enables adapting pre-trained ML models to varying DQ by employing the data of varied quality

for re-training while also significantly reducing the training time. FL provides a distributed framework that not only addresses the challenge of training data variations but also enhances the ML system security and privacy. FL's decentralized data processing minimizes the risks of privacy violation and adapts effectively to DQ variations. To further improve the effectiveness, security, and privacy of FL, we incorporated it with our Reputation and Trust-based techniques that allow establishing and quantifying trust towards the local units. We see TL and FL as a proactive measures applied through the ML model training that allow improving ML application robustness towards the DQ variations in the ML execution phase.

- **Are TL and FL feasible to be employed over the ML training phase in order to enhance ML application robustness towards DQ variations within ML execution?**

  Our industrial use case investigations, conducted in sec. 5.2.1 and 5.3.3, demonstrated that TL and FL are indeed feasible techniques to be employed during the ML training phase in order to enhance ML application robustness towards DQ variations during the execution. According to our TL case results, further training the pre-trained ML model on the degraded data only appeared to be the most effective approach in comparison to other training data configurations, and helped to improve the ML performance on the degraded DQ samples in comparison to the baseline. For the FL case, the presence of the aggregation function allowed it to better tolerate the variations in the training data, which enhanced the overall ML performance over the diverse data during the model execution.

- **How to evaluate if TL and FL are feasible in enhancing ML application robustness in practice?**

  In sec. 5.2, we examined the ITS industrial case that incorporated traffic sign images classification. We studied how the various combination of diverse quality training data affected the performance of the resulting ML model that have to process samples of various DQ. We considered two types of training: further training the ML model using TL in a centralized manner; and further training the ML model using FL on the data distributed over a number of local units. We compared the results of the produced ML model performance demonstrated over the data of varied DQ in both cases. Both TL and FL appeared to be effective in improving ML performance over the varied DQ samples, which contributes to enhancing the ML application robustness.

- **What conventional FL features make it vulnerable to malicious attacks against local units?**

During our investigations, we revealed the methodological FL vulnerability that jeopardizes security and privacy of the ML system. In the conventional FL, the local units participating in the aggregation round do not undergo any verification procedures when the global model is distributed back to them. If the local unit was compromised, an adversary may gain access to the global model distributed to the local unit after the aggregation, and may implement malicious data inference or membership attacks. Since the global model is based on a combination of all the local models submitted for aggregation, such a vulnerability endangers the security and privacy of all the local units.

– **What methods and techniques should be developed to further enhance the security and robustness of the FL?**

To further enhance the security and robustness of ML applications within MLIN, we introduced innovative Reputation and Trust techniques. We integrated these techniques with FL to address the discovered methodological vulnerability in the conventional FL processes. By equipping FL with Reputation and Trust techniques, we enhanced FL security and improved learning effectiveness. Practical validation in real industrial applications, such as digital payment systems, demonstrated the effectiveness and feasibility of our approach. Our findings offer additional methods to accommodate both security and robustness challenges in MLIN systems, particularly when dealing with sensitive applications where the privacy of data is of paramount importance.

– **What reactive and preventive methods and techniques should be integrated to address the challenges posed by the DQ variation in practical ML applications?**

Our research showed that addressing the challenges posed by the DQ variations in practical ML applications requires a balanced integration of reactive and preventive measures. The proposed MLIN feedback adjustment component operates reactively, continuously monitoring MLIN parameters during ML execution and offering real-time recommendations to improve the ML performance and ML application robustness to DQ variations. On the preventive front, we examined TL and FL as valuable techniques during the ML training phase. TL allows adapting models to DQ variations by leveraging pre-trained knowledge, enhancing learning efficiency and robustness of the resulting model towards DQ variations. FL, on the other hand, contributes significantly to security and privacy improvement by keeping the training data confidential and enhances ML application robustness to DQ variations through the aggregation procedure. By incorporating these reactive and preventive strategies, ML systems can better adapt to DQ variations, maintaining ML application robustness, security, and privacy throughout their life-cycle.

Our research positively impacts the areas of ML and AI, communication, and the computing in overall by enhancing ML applications robustness and security. In an era where data-driven systems are becoming the core of critical decision-making processes across various domains, the issue of ML robustness has never been more relevant. Our approach not only addresses the DQ and robustness challenges in ML applications but also engenders trust in modern ML- and AI-based systems. Our approach contributes to developing more dependable data-driven systems by enabling their robust and widespread deployment across diverse applications and industries. These benefits align with the evolving needs and challenges of the global ML landscape, making our approach a valuable contribution to the community.

Bibliography

[1] Fireworks Sound Effect. (Date last accessed 10-September-2023).

[2] Overview of Open Images V6. (Date last accessed 10-September-2023).

[3] Ultimate Military / Weapon Gun Shot Sound Effect Pack! [200+ Sounds for 3 HOURS]. (Date last accessed 10-September-2023).

[4] Ieee standard glossary of software engineering terminology. *IEEE Std 610.12-1990*, pages 1–84, 1990.

[5] Safetynet, 6 2018. . Accessed: September 10, 2023.

[6] Durmus Alp Emre Acar, Yue Zhao, Ramon Matas Navarro, Matthew Mattina, Paul N Whatmough, and Venkatesh Saligrama. Federated learning based on dynamic regularization. *arXiv preprint arXiv:2111.04263*, 2021.

[7] Zeeshan Ahmed, Salima Hamma, and Zafar Nasir. An optimal bandwidth allocation algorithm for improving qos in wimax. *Multimedia Tools and Applications*, 78(18):25937–25976, 2019.

[8] Khaled Walid Al-Sabbagh, Miroslaw Staron, Regina Hebig, and Wilhelm Meding. Improving data quality for regression test selection by reducing annotation noise. In *2020 46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 191–194. IEEE, 2020.

[9] Guillaume Alain and Yoshua Bengio. What regularized auto-encoders learn from the data-generating distribution. *The Journal of Machine Learning Research*, 15(1):3563–3593, 2014.

[10] Mohammed Aledhari, Rehma Razzak, Reza M Parizi, and Fahad Saeed. Federated learning: A survey on enabling technologies, protocols, and applications. *IEEE Access*, 8:140699–140725, 2020.

[11] Ahmed Ali and Steve Renals. Word error rate estimation for speech recognition: e-wer. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 20–24, 2018.

[12] Rosana Ardila, Megan Branson, Kelly Davis, Michael Henretty, Michael Kohler, Josh Meyer, Reuben Morais, Lindsay Saunders, Francis M Tyers, and Gregor Weber. Common voice: A massively-multilingual speech corpus. *arXiv preprint arXiv:1912.06670*, 2019.

[13] Matthew Arnold, Rachel KE Bellamy, Michael Hind, Stephanie Houde, Sameep Mehta, Aleksandra Mojsilović, Ravi Nair, K Natesan Ramamurthy, Alexandra Olteanu, David Piorkowski, et al. Factsheets: Increasing trust in ai services through supplier's declarations of conformity. *IBM Journal of Research and Development*, 63(4/5):6–1, 2019.

[14] Otmane Azeroual, Joachim Schöpfel, Dragan Ivanovic, and Anastasija Nikiforova. Combining data lake and data wrangling for ensuring data quality in cris. *Procedia Computer Science*, 211:3–16, 2022.

[15] Xianglin Bao, Cheng Su, Yan Xiong, Wenchao Huang, and Yifei Hu. Flchain: A blockchain for auditable federated learning with trust and incentive. In *2019 5th International Conference on Big Data Computing and Communications (BIGCOM)*, pages 151–159. IEEE, 2019.

[16] Alcardo Alex Barakabitze, Nabajeet Barman, Arslan Ahmad, Saman Zadtootaghaj, Lingfen Sun, Maria G Martini, and Luigi Atzori. Qoe management of multimedia streaming services in future networks: A tutorial and survey. *IEEE Communications Surveys & Tutorials*, 22(1):526–565, 2019.

[17] Osbert Bastani, Yani Ioannou, Leonidas Lampropoulos, Dimitrios Vytiniotis, Aditya Nori, and Antonio Criminisi. Measuring neural net robustness with constraints. *Advances in neural information processing systems*, 29, 2016.

[18] Aharon Ben-Tal, Laurent El Ghaoui, and Arkadi Nemirovski. *Robust optimization*, volume 28. Princeton university press, 2009.

[19] Abhijit Bendale and Terrance Boult. Towards open world recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1893–1902, 2015.

[20] OI Berezovskaya, SS Chuprov, EA Neverov, and ER Sadreev. Review and comparison of lightweight modifications of the aes cipher for a network of low-power devices. *Automatic Control and Computer Sciences*, 56(8):994–1006, 2022.

[21] Mykola Beshley, Natalia Kryvinska, Marian Seliuchenko, Halyna Beshley, Elhadi M Shakshuki, and Ansar-Ul-Haque Yasar. End-to-end qos "smart queue" management algorithms and traffic prioritization mechanisms for narrow-band internet of things services in 4g/5g networks. *Sensors*, 20(8):2324, 2020.

[22] Peva Blanchard, El Mahdi El Mhamdi, Rachid Guerraoui, and Julien Stainer. Machine learning with adversaries: Byzantine tolerant gradient descent. *Advances in Neural Information Processing Systems*, 30, 2017.

[23] Joe Breen, Andrew Buffmire, Jonathon Duerig, Kevin Dutt, Eric Eide, Anneswa Ghosh, Mike Hibler, David Johnson, Sneha Kumar Kasera, Earl Lewis, et al. Powder: Platform for open wireless data-driven experimental research. *Computer Networks*, page 108281, 2021.

[24] Flavio Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. Optimized pre-processing for discrimination prevention. *Advances in neural information processing systems*, 30, 2017.

[25] Ebru Celikel Cankaya. *Bell-LaPadula Confidentiality Model*, pages 71–74. Springer US, Boston, MA, 2011.

[26] Di Cao, Shan Chang, Zhijian Lin, Guohua Liu, and Donghong Sun. Understanding distributed poisoning attack in federated learning. In *2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS)*, pages 233–239. IEEE, 2019.

[27] Haotong Cao, Shengchen Wu, Gagangeet Singh Aujla, Qin Wang, Longxiang Yang, and Hongbo Zhu. Dynamic embedding and quality of service-driven adjustment for cloud networks. *IEEE Transactions on Industrial Informatics*, 16(2):1406–1416, 2019.

[28] Xiaoyu Cao, Minghong Fang, Jia Liu, and Neil Zhenqiang Gong. Fltrust: Byzantine-robust federated learning via trust bootstrapping. *arXiv preprint arXiv:2012.13995*, 2020.

[29] Yulong Cao, Ningfei Wang, Chaowei Xiao, Dawei Yang, Jin Fang, Ruigang Yang, Qi Alfred Chen, Mingyan Liu, and Bo Li. Invisible for both camera and lidar: Security of multi-sensor fusion based perception in autonomous driving under physical-world attacks. In *2021 IEEE Symposium on Security and Privacy (SP)*, pages 176–194. IEEE, 2021.

[30] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[31] A Carter, S Imtiaz, and GF Naterer. Review of interpretable machine learning for process industries. *Process Safety and Environmental Protection*, 2022.

[32] Anirban Chakraborty, Manaar Alam, Vishal Dey, Anupam Chattopadhyay, and Debdeep Mukhopadhyay. A survey on adversarial attacks and defences. *CAAI Transactions on Intelligence Technology*, 6(1):25–45, 2021.

[33] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002.

[34] Minmin Chen, Kilian Weinberger, Fei Sha, and Yoshua Bengio. Marginalized denoising auto-encoders for nonlinear representations. In *International conference on machine learning*, pages 1476–1484. PMLR, 2014.

[35] Yizheng Chen, Shiqi Wang, Yue Qin, Xiaojing Liao, Suman Jana, and David Wagner. Learning security classifiers with verified global robustness properties. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 477–494, 2021.

[36] Yu Chen, Fang Luo, Tong Li, Tao Xiang, Zheli Liu, and Jin Li. A training-integrity privacy-preserving federated learning scheme with trusted execution environment. *Information Sciences*, 522:69–79, 2020.

[37] Zhe Chen, Fu Xiao, Fangzhou Guo, and Jinyue Yan. Interpretable machine learning for building energy management: A state-of-the-art review. *Advances in Applied Energy*, page 100123, 2023.

[38] Zheng Chen, Wenli Zhou, Shuo Wu, and Li Cheng. An adaptive on-demand multipath routing protocol with qos support for high-speed manet. *IEEE Access*, 8:44760–44773, 2020.

[39] Li Sze Chow and Raveendran Paramesran. Review of medical image quality assessment. *Biomedical signal processing and control*, 27:145–154, 2016.

[40] Sergei Chuprov, Pavel Belyaev, Ruslan Gataullin, Leon Reznik, Evgenii Neverov, and Ilia Viksnin. Robust autonomous vehicle computer-vision-based localization in challenging environmental conditions. *Applied Sciences*, 13(9):5735, 2023.

[41] Sergei Chuprov, Igor Khokhlov, Leon Reznik, and Srujan Shetty. Influence of transfer learning on machine learning systems robustness to data quality degradation. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2022.

[42] Sergei Chuprov, Moinuddin Memon, and Leon Reznik. Federated learning with trust evaluation for industrial applications. (Accepted for publication in IEEE CAI 2023 proceedings, in press).

[43] Sergei Chuprov, Leon Reznik, and Garegin Grigoryan. Study on network importance for ml end application robustness. In *ICC 2023-IEEE International Conference on Communications*, pages 6627–6632. IEEE, 2023.

[44] Sergei Chuprov, Leon Reznik, Igor Khokhlov, and Karan Manghi. Multi-modal sensor selection with genetic algorithms. In *2022 IEEE Sensors*, pages 1–4. IEEE, 2022.

[45] Sergei Chuprov, Leon Reznik, Antoun Obeid, and Srujan Shetty. How degrading network conditions influence machine learning end systems performance? In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2022.

[46] Sergei Chuprov, Leon Reznik, Antoun Obeid, and Srujan Shetty. How degrading network conditions influence machine learning end systems performance? (*will be published soon*). In *The 9th International Workshop on Computer and Networking Experimental Research using Testbeds (CNERT)*, pages 1–6. IEEE, 2022.

[47] Sergei Chuprov, Akshaya Nandkishor Satam, and Leon Reznik. Are ml image classifiers robust to medical image quality degradation? In *2022 IEEE Western New York Image and Signal Processing Workshop (WNYISPW)*, pages 1–4. IEEE, 2022.

[48] Sergei Chuprov, Ilia Viksnin, Iuliia Kim, Timofey Melnikov, Leon Reznik, and Igor Khokhlov. Improving knowledge based detection of soft attacks against autonomous vehicles with reputation, trust and data quality service models. In *2021 IEEE International Conference on Smart Data Services (SMDS)*, pages 115–120. IEEE, 2021.

[49] Sergey Chuprov, Ruslan Gataullin, Evgenii Neverov, Pavel Belyaev, Iuliia Kim, and Ilia Viksnin. Police office model performance and security evaluation in a simulated group of mobile robots. In *The 5th International Conference on Future Networks & Distributed Systems*, pages 606–615, 2021.

[50] Sergey Chuprov, Egor Marinenkov, Ilya Viksnin, Leon Reznik, and Igor Khokhlov. Image processing in autonomous vehicle model positioning and movement control. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, pages 1–6. IEEE, 2020.

[51] Sergey Chuprov, Ilia Viksnin, Iuliia Kim, Nikita Tursukov, and Gleb Nedosekin. Empirical study on discrete modeling of urban intersection management system. *International Journal of Embedded and Real-Time Communication Systems (IJERTCS)*, 11(2):16–38, 2020.

[52] Sergey Chuprov, Ilya Viksnin, and Iuliia Kim. Urban intersection management with connected infrastructure objects and autonomous vehicles. In *2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE)*, pages 1–4. IEEE, 2019.

[53] Sergey Chuprov, Ilya Viksnin, Iuliia Kim, Egor Marinenkov, Maria Usova, Eduard Lazarev, Timofey Melnikov, and Danil Zakoldaev. Reputation and trust approach for security and safety assurance in intersection management system. *Energies*, 12(23):4527, 2019.

[54] Sergey Chuprov, Ilya Viksnin, Iuliia Kim, and Gleb Nedosekin. Optimization of autonomous vehicles movement in urban intersection management system. In *2019 24th Conference of Open Innovations Association (FRUCT)*, pages 60–66. IEEE, 2019.

[55] Sergey Chuprov, Ilya Viksnin, Iuliia Kim, Leon Reznik, and Igor Khokhlov. Reputation and trust models with data quality metrics for improving autonomous vehicles traffic security and safety. In *2020 IEEE systems security symposium (SSS)*, pages 1–8. IEEE, 2020.

[56] Sergey Chuprov, Ilya I Viksnin, Iuliia Kim, and Maria Usova. Intersection management tasks in mobile robotic system with decentralized control. In *MICSECS*, 2018.

[57] JB Copas. Plotting p against x. *Applied statistics*, pages 25–31, 1983.

[58] Oscar Day and Taghi M Khoshgoftaar. A survey on heterogeneous transfer learning. *Journal of Big Data*, 4(1):1–42, 2017.

[59] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[60] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[61] Zhun Deng, Linjun Zhang, Kailas Vodrahalli, Kenji Kawaguchi, and James Y Zou. Adversarial training helps transfer learning via better representations. *Advances in Neural Information Processing Systems*, 34:25179–25191, 2021.

[62] Egor Domnitsky, Vladimir Mikhailov, Evgeniy Zoloedov, Danila Alyukov, Sergey Chuprov, Egor Marinenkov, and Ilya I Viksnin. Software module for unmanned autonomous vehicle's on-board camera faults detection and correction. In *MICSECS*, 2020.

[63] Mitchell L Doucette, Christa Green, Jennifer Necci Dineen, David Shapiro, and Kerri M Raissian. Impact of shotspotter technology on firearm homicides and arrests among

large metropolitan counties: A longitudinal analysis, 1999–2016. *Journal of urban health*, 98(5):609–621, 2021.

[64] Nathan Drenkow, Numair Sani, Ilya Shpitser, and Mathias Unberath. A systematic review of robustness in deep learning for computer vision: Mind the gap? *arXiv preprint arXiv:2112.00639*, 2021.

[65] Lisa Ehrlinger, Verena Haunschmid, Davide Palazzini, and Christian Lettner. A daql to monitor data quality in machine learning applications. In *Database and Expert Systems Applications: 30th International Conference, DEXA 2019, Linz, Austria, August 26–29, 2019, Proceedings, Part I 30*, pages 227–237. Springer, 2019.

[66] Arif Tanju Erdem and Ali Özer Ercan. Fusing inertial sensor data in an extended kalman filter for 3d camera tracking. *IEEE Transactions on Image Processing*, 24(2):538–548, 2014.

[67] Laura Erhan, M Ndubuaku, Mario Di Mauro, Wei Song, Min Chen, Giancarlo Fortino, Ovidiu Bagdasar, and Antonio Liotta. Smart anomaly detection in sensor systems: A multi-perspective review. *Information Fusion*, 67:64–79, 2021.

[68] Emanuel Falkenauer. Applying genetic algorithms to real-world problems. In *Evolutionary algorithms*, pages 65–88. Springer, 1999.

[69] Abolfazl Farahani, Behrouz Pourshojae, Khaled Rasheed, and Hamid R Arabnia. A concise review of transfer learning. In *2020 International Conference on Computational Science and Computational Intelligence (CSCI)*, pages 344–351. IEEE, 2020.

[70] Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Mitigating sybils in federated learning poisoning. *arXiv preprint arXiv:1808.04866*, 2018.

[71] Raghu K Ganti, Fan Ye, and Hui Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, 2011.

[72] Antonio Ghezzi, Donata Gabelloni, Antonella Martini, and Angelo Natalicchio. Crowdsourcing: a review and suggestions for future research. *International Journal of Management Reviews*, 20(2):343–363, 2018.

[73] Anousheh Gholami, Nariman Torkzaban, and John S Baras. Trusted decentralized federated learning. In *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–6. IEEE, 2022.

[74] Einollah Jafarnejad Ghomi, Amir Masoud Rahmani, and Nooruldeen Nasih Qader. Load-balancing algorithms in cloud computing: A survey. *Journal of Network and Computer Applications*, 88:50–71, 2017.

[75] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[76] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068*, 2014.

[77] Nitin Gupta, Hima Patel, Shazia Afzal, Naveen Panwar, Ruhi Sharma Mittal, Shanmukha Guttula, Abhinav Jain, Lokesh Nagalapatti, Sameep Mehta, Sandeep Hans, et al. Data quality toolkit: Automatic assessment of data quality and remediation for machine learning datasets. *arXiv preprint arXiv:2108.05935*, 2021.

[78] Leif Hancox-Li. Robustness in machine learning explanations: Does it matter? In *Proceedings of the 2020 conference on fairness, accountability, and transparency*, pages 640–647, 2020.

[79] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, et al. Deep speech: Scaling up end-to-end speech recognition. *arXiv preprint arXiv:1412.5567*, 2014.

[80] Awni Hannun, Carl Case, Jared Casper, Bryan Catanzaro, Greg Diamos, Erich Elsen, Ryan Prenger, Sanjeev Satheesh, Shubho Sengupta, Adam Coates, and Andrew Y. Ng. Deep speech: Scaling up end-to-end speech recognition, 2014.

[81] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[82] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *arXiv preprint arXiv:1610.02136*, 2016.

[83] Shawn Hershey, Sourish Chaudhuri, Daniel PW Ellis, Jort F Gemmeke, Aren Jansen, R Channing Moore, Manoj Plakal, Devin Platt, Rif A Saurous, Bryan Seybold, et al. Cnn architectures for large-scale audio classification. In *2017 ieee international conference on acoustics, speech and signal processing (icassp)*, pages 131–135. IEEE, 2017.

[84] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015.

[85] Jeroen D Hol, Thomas B Schön, Henk Luinge, Per J Slycke, and Fredrik Gustafsson. Robust real-time tracking by fusing measurements from inertial and vision sensors. *Journal of Real-Time Image Processing*, 2(2):149–160, 2007.

[86] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[87] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*, 2018.

[88] Chin-Lin Hu, Chao-Yu Hsu, Sod-Erdene Khuukhenbaatar, Yamkhin Dashdorj, and Yongqiang Dong. Path selection with joint latency and packet loss for edge computing in sdn. In *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 1–4. IEEE, 2019.

[89] Zeou Hu, Kiarash Shaloudegi, Guojun Zhang, and Yaoliang Yu. Fedmgda+: Federated learning meets multi-objective optimization. *arXiv preprint arXiv:2006.11489*, 2020.

[90] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[91] Xiaowei Huang, Marta Kwiatkowska, Sen Wang, and Min Wu. Safety verification of deep neural networks. In *International conference on computer aided verification*, pages 3–29. Springer, 2017.

[92] Ben Hutchinson, Andrew Smart, Alex Hanna, Emily Denton, Christina Greer, Oddur Kjartansson, Parker Barnes, and Margaret Mitchell. Towards accountability for machine learning datasets: Practices from software engineering and infrastructure. In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 560–575, 2021.

[93] Sohei Itahara, Takayuki Nishio, and Koji Yamamoto. Packet-loss-tolerant split inference for delay-sensitive deep learning in lossy wireless networks. In *2021 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2021.

[94] Jiawen Kang, Zehui Xiong, Dusit Niyato, Shengli Xie, and Junshan Zhang. Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory. *IEEE Internet of Things Journal*, 6(6):10700–10714, 2019.

[95] Jiawen Kang, Zehui Xiong, Dusit Niyato, Yuze Zou, Yang Zhang, and Mohsen Guizani. Reliable federated learning for mobile networks. *IEEE Wireless Communications*, 27(2):72–80, 2020.

[96] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. Scaffold: Stochastic controlled averaging for federated learning. In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.

[97] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient smt solver for verifying deep neural networks. In *International conference on computer aided verification*, pages 97–117. Springer, 2017.

[98] Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina CS Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, et al. Identifying medical diagnoses and treatable diseases by image-based deep learning. *Cell*, 172(5):1122–1131, 2018.

[99] Tran Duy Khanh, Igor Komarov, Radda Iureva, Sergey Chuprov, et al. Tra: effective authentication mechanism for swarms of unmanned aerial vehicles. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1852–1858. IEEE, 2020.

[100] Igor Khokhlov. *Integrated framework for data quality and security evaluation on mobile devices*. PhD thesis, 2020.

[101] Igor Khokhlov, Sergei Chuprov, and Leon Reznik. Integrating security with accuracy evaluation in sensors fusion. In *2022 IEEE Sensors*, pages 1–4, 2022.

[102] Igor Khokhlov, Akshay Pudage, and Leon Reznik. Sensor selection optimization with genetic algorithms. In *IEEE Sensors 2019, Montreal, Canada*, 2019.

[103] Igor Khokhlov, Akshay Pudage, and Leon Reznik. Sensor selection optimization with genetic algorithms. In *2019 IEEE SENSORS*, pages 1–4. IEEE, 2019.

[104] Igor Khokhlov, Leon Reznik, and Sahil Ajmera. Sensors in mobile devices knowledge base. *IEEE Sensors Letters*, 4(3):1–4, 2020.

[105] Igor Khokhlov, Leon Reznik, and Rohit Bhaskar. The machine learning models for activity recognition applications with wearable sensors. In *18th IEEE International Conference on Machine Learning and Applications - ICMLA 2019*. IEEE, 2019.

[106] Igor Khokhlov, Leon Reznik, Justin Cappos, and Rohit Bhaskar. Design of activity recognition systems with wearable sensors. In *2018 IEEE Sensors Applications Symposium (SAS)*, pages 1–6. IEEE, 2018.

[107] Igor Khokhlov, Leon Reznik, and Sergey Chuprov. Framework for integral data quality and security evaluation in smartphones. *IEEE Systems Journal*, 15(2):2058–2065, 2020.

[108] Igor Khokhlov, Leon Reznik, and Sergey Chuprov. Framework for integral data quality and security evaluation in smartphones. *IEEE Systems Journal*, 15(2):2058–2065, 2021.

[109] Igor Khokhlov, Leon Reznik, Suresh Babu Jothilingam, and Rohit Bhaskar. What can data analysis recommend on design of wearable sensors? In *2018 15th IEEE Annual Consumer Communications & Networking Conference (CCNC)*, pages 1–2. IEEE, 2018.

[110] Hyesung Kim, Jihong Park, Mehdi Bennis, and Seong-Lyun Kim. Blockchained on-device federated learning. *IEEE Communications Letters*, 24(6):1279–1283, 2019.

[111] Iuliia Kim, João Pedro Matos-Carvalho, Ilya Viksnin, Luís Miguel Campos, José M Fonseca, André Mora, and Sergey Chuprov. Use of particle swarm optimization in terrain classification based on uav downwash. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 604–610. IEEE, 2019.

[112] Jakub Konečnỳ, H Brendan McMahan, Felix X Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[113] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). Accessed on July 30, 2023.

[114] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55(5), 2014.

[115] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012.

[116] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *Artificial intelligence safety and security*, pages 99–112. Chapman and Hall/CRC, 2018.

[117] Asif Laghari, Rashid Laghari, Asif Wagan, and Aamir Umrani. Effect of packet loss and reorder on quality of audio streaming. *EAI Endorsed Transactions on Scalable Information Systems*, 7(24), 2019.

[118] Asif Ali Laghari, Rashid Ali Laghari, Asif Ali Wagan, and Aamir Iqbal Umrani. Effect of packet loss and reorder on quality of audio streaming. *EAI Endorsed Transactions on Scalable Information Systems*, 7(25), 2020.

[119] Himabindu Lakkaraju and Osbert Bastani. " how do i fool you?" manipulating user trust via misleading black box explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pages 79–85, 2020.

[120] Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. Faithful and customizable explanations of black box models. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, pages 131–138, 2019.

[121] Adam Langley, Alistair Riddoch, Alyssa Wilk, Antonio Vicente, Charles Krasic, Dan Zhang, Fan Yang, Fedor Kouranov, Ian Swett, Janardhan Iyengar, et al. The quic transport protocol: Design and internet-scale deployment. In *Proceedings of the conference of the ACM special interest group on data communication*, pages 183–196, 2017.

[122] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[123] Ga Young Lee, Lubna Alzamil, Bakhtiyar Doskenov, and Arash Termehchy. A survey on data cleaning methods for improved machine learning model performance. *arXiv preprint arXiv:2109.07127*, 2021.

[124] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *Advances in neural information processing systems*, 31, 2018.

[125] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020.

[126] Shiyu Liang, Yixuan Li, and Rayadurgam Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*, 2017.

[127] Chienhung Lin, Kuochen Wang, and Guocin Deng. A qos-aware routing in sdn hybrid networks. *Procedia Computer Science*, 110:242–249, 2017.

[128] Chienhung Lin, Kuochen Wang, and Guocin Deng. A qos-aware routing in sdn hybrid networks. *Procedia Computer Science*, 110:242–249, 2017. 14th International Conference on Mobile Systems and Pervasive Computing (MobiSPC 2017) / 12th International Conference on Future Networks and Communications (FNC 2017) / Affiliated Workshops.

[129] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. corr abs/1312.4400 (2013). *arXiv preprint arXiv:1312.4400*, 2013.

[130] Zachary C Lipton. The mythos of model interpretability: In machine learning, the concept of interpretability is both important and slippery. *Queue*, 16(3):31–57, 2018.

[131] Haochen Liu. Trustworthy machine learning: Fairness and robustness. In *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining*, pages 1553–1554, 2022.

[132] Xiaoyuan Liu, Hongwei Li, Guowen Xu, Zongqi Chen, Xiaoming Huang, and Rongxing Lu. Privacy-enhanced federated learning against poisoning adversaries. *IEEE Transactions on Information Forensics and Security*, 16:4574–4588, 2021.

[133] Yin Lou, Rich Caruana, and Johannes Gehrke. Intelligible models for classification and regression. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 150–158, 2012.

[134] Lucy Ellen Lwakatare, Aiswarya Raj, Ivica Crnkovic, Jan Bosch, and Helena Holmström Olsson. Large-scale machine learning systems in real-world industrial settings: A review of challenges and solutions. *Information and software technology*, 127:106368, 2020.

[135] Julia Lyakhovenko, Ilya I Viksnin, and Sergey Chuprov. Integrating smart contracts into smart factory elements' informational interaction model. In *MICSECS*, 2020.

[136] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, Jiyuan Sun, Minhui Xue, Bo Li, Chunyang Chen, Ting Su, Li Li, Yang Liu, et al. Deepgauge: Multi-granularity testing criteria for deep learning systems. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 120–131, 2018.

[137] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

[138] Umer Majeed and Choong Seon Hong. Flchain: Federated learning via mec-enabled blockchain network. In *2019 20th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 1–4. IEEE, 2019.

[139] Dimitrios Michael Manias and Abdallah Shami. Making a case for federated learning in the internet of vehicles and intelligent transportation systems. *IEEE Network*, 35(3):88–94, 2021.

[140] Egor Marinenkov, Sergei Chuprov, Nikita Tursukov, Iuliia Kim, and Ilia Viksnin. Study on destructive informational impact in unmanned aerial vehicles intergroup communication. *Symmetry*, 14(8):1580, 2022.

[141] Egor Marinenkov, Sergey Chuprov, Ilya I Viksnin, and Iuliia Kim. Empirical study on trust, reputation, and game theory approach to secure communication in a group of unmanned vehicles. In *MICSECS*, 2019.

[142] Mohammad Masoud, Yousef Jaradat, Ahmad Manasrah, Ismael Jannoud, et al. Sensors of smart devices in the internet of everything (ioe) era: big opportunities and massive doubts. *Journal of Sensors*, 2019, 2019.

[143] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[144] Jacob Patrick McManus, Timothy George Day, and Zachary J Mailloux. The effects of latency, bandwidth, and packet loss on cloud-based gaming services. *Interact Qualif Proj (All Years)*, 58, 2019.

[145] Tahir Mehmood, Alfonso E Gerevini, Alberto Lavelli, and Ivan Serina. Combining multi-task learning with transfer learning for biomedical named entity recognition. *Procedia Computer Science*, 176:848–857, 2020.

[146] Timofey Melnikov, Eduard Lazarev, Olesya Berezovskaya, Sergey Chuprov, and Ilia Viksnin. Empirical study on premises monitoring algorithm implementation in mobile robotic system. In *2020 International Conference Nonlinearity, Information and Robotics (NIR)*, pages 1–6. IEEE, 2020.

[147] Timofey Y Melnikov, Sergey S Chuprov, Eduard A Lazarev, Ruslan I Gataullin, Ilia I Viksnin, et al. Improving reputation and trust-based approach with reliability indicators for autonomous vehicles intergroup communication. 2022.

[148] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. *arXiv preprint arXiv:1702.04267*, 2017.

[149] Jonathan K. Millen. *Biba Model*, pages 81–82. Springer US, Boston, MA, 2011.

[150] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2574–2582, 2016.

[151] Mahmoud Moshref, Rizik Al-Sayyed, and Saleh Al-Sharaeh. An enhanced multi-objective non-dominated sorting genetic routing algorithm for improving the qos in wireless sensor networks. *IEEE Access*, 9:149176–149195, 2021.

[152] Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.

[153] Štefica Mrvelj and Marko Matulin. Impact of packet loss on the perceived quality of udp-based multimedia streaming: a study of user quality of experience in real-life environments. *Multimedia systems*, 24(1):33–53, 2018.

[154] Milad Nasr, Reza Shokri, and Amir Houmansadr. Comprehensive privacy analysis of deep learning: Passive and active white-box inference attacks against centralized and federated learning. In *2019 IEEE symposium on security and privacy (SP)*, pages 739–753. IEEE, 2019.

[155] Zeinab Nazemi Absardi and Reza Javidan. A qoe-driven sdn traffic management for iot-enabled surveillance systems using deep learning based on edge cloud computing. *The Journal of Supercomputing*, pages 1–26, 2023.

[156] Evgenii A Neverov, Ilia I Viksnin, and Sergei S Chuprov. The research of automl methods in the task of wave data classification. In *2023 XXVI International Conference on Soft Computing and Measurements (SCM)*, pages 156–158. IEEE, 2023.

[157] Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 427–436, 2015.

[158] Huy T Nguyen, Nguyen Cong Luong, Jun Zhao, Chau Yuen, and Dusit Niyato. Resource allocation in mobility-aware federated learning networks: A deep reinforcement learning approach. In *2020 IEEE 6th World Forum on Internet of Things (WF-IoT)*, pages 1–6. IEEE, 2020.

[159] Lan N Nguyen and My T Thai. Network resilience assessment via qos degradation metrics: An algorithmic approach. *Proceedings of the ACM on Measurement and Analysis of Computing Systems*, 3(1):1–32, 2019.

[160] Adrian Nilsson, Simon Smith, Gregor Ulm, Emil Gustavsson, and Mats Jirstrand. A performance evaluation of federated learning algorithms. In *Proceedings of the second workshop on distributed infrastructures for deep learning*, pages 1–8, 2018.

[161] Owusu Nyarko-Boateng, Faith Edem Bright Xedagbui, Adebayo Felix Adekoya, and Benjamin Asubam Weyori. Fiber optic deployment challenges and their management in a developing country: A tutorial and case study in ghana. *Engineering Reports*, 2(2):e12121, 2020.

[162] Mohammad S Obaidat, Faouzi Zarai, and Petros Nicopolitidis. *Modeling and simulation of computer networks and systems: Methodologies and applications*. Morgan Kaufmann, 2015.

[163] Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa. Oblivious multi-party machine learning on trusted processors. In *USENIX Security Symposium*, volume 16, pages 10–12, 2016.

[164] Steve Olenski. The state of crowdsourcing, 2015. https://www.forbes.com/sites/steveolenski/2015/12/04/the-state-of-crowdsourcing/. Accessed: April 17, 2019.

[165] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings, 2015.

[166] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, pages 582–597. IEEE, 2016.

[167] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. Deepxplore: Automated whitebox testing of deep learning systems. In *proceedings of the 26th Symposium on Operating Systems Principles*, pages 1–18, 2017.

[168] Cristina Perfecto, Mohammed S Elbamby, Javier Del Ser, and Mehdi Bennis. Taming the latency in multi-user vr 360°: A qoe-aware deep learning-aided multicast framework. *IEEE Transactions on Communications*, 68(4):2491–2508, 2020.

[169] Jeremy Petch, Shuang Di, and Walter Nelson. Opening the black box: the promise and limitations of explainable machine learning in cardiology. *Canadian Journal of Cardiology*, 38(2):204–213, 2022.

[170] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *arXiv preprint arXiv:1912.13445*, 2019.

[171] Krishna Pillutla, Sham M Kakade, and Zaid Harchaoui. Robust aggregation for federated learning. *IEEE Transactions on Signal Processing*, 70:1142–1154, 2022.

[172] Ivan Miguel Pires, Nuno M Garcia, Nuno Pombo, and Francisco Flórez-Revuelta. From data acquisition to data fusion: a comprehensive review and a roadmap for the identification of activities of daily living using mobile devices. *Sensors*, 16(2):184, 2016.

[173] John P. Pironti. Five key considerations when applying a trust, but verify approach to information security and risk management, 2021. Accessed on July 30, 2023.

[174] Forough Poursabzi-Sangdeh, Daniel G Goldstein, Jake M Hofman, Jennifer Wortman Wortman Vaughan, and Hanna Wallach. Manipulating and measuring model interpretability. In *Proceedings of the 2021 CHI conference on human factors in computing systems*, pages 1–52, 2021.

[175] Atieh Poushneh. Humanizing voice assistant: The impact of voice assistant personality on consumers' attitudes and behaviors. *Journal of Retailing and Consumer Services*, 58:102283, 2021.

[176] Maria Priestley, Fionntán O'Donnell, and Elena Simperl. A survey of data quality requirements that matter in ml development pipelines. *ACM Journal of Data and Information Quality*, 2023.

[177] Tong Qin, Shaozu Cao, Jie Pan, and Shaojie Shen. A general optimization-based framework for global pose estimation with multiple sensors. *arXiv preprint arXiv:1901.03642*, 2019.

[178] Youyang Qu, Md Palash Uddin, Chenquan Gan, Yong Xiang, Longxiang Gao, and John Yearwood. Blockchain-enabled federated learning: A survey. *ACM Computing Surveys*, 55(4):1–35, 2022.

[179] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv preprint arXiv:1707.04131*, 2017.

[180] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[181] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[182] L. Reznik and Elisa Bertino. Poster: Data quality evaluation: integrating security and accuracy. pages 1367–1370, 11 2013.

[183] Leon Reznik. *Intelligent Security Systems: How Artificial Intelligence, Machine Learning and Data Science Work for and Against Computer Security.* John Wiley & Sons, 2021.

[184] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[185] Greg Ridgeway, David Madigan, Thomas Richardson, and John O'Kane. Interpretable boosted naïve bayes classification. In *KDD*, pages 101–104, 1998.

[186] Salah Rifai, Grégoire Mesnil, Pascal Vincent, Xavier Muller, Yoshua Bengio, Yann Dauphin, and Xavier Glorot. Higher order contractive auto-encoder. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 645–660. Springer, 2011.

[187] Gaith Rjoub, Omar Abdel Wahab, Jamal Bentahar, and Ahmed Bataineh. Trust-driven reinforcement selection strategy for federated learning on iot devices. *Computing*, pages 1–23, 2022.

[188] Andrew Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[189] Abhijit Guha Roy, Shayan Siddiqui, Sebastian Pölsterl, Nassir Navab, and Christian Wachinger. Braintorrent: A peer-to-peer environment for decentralized federated learning. *arXiv preprint arXiv:1905.06731*, 2019.

[190] Andras Rozsa, Manuel Günther, and Terrance E Boult. Are accuracy and robustness correlated. In *2016 15th IEEE international conference on machine learning and applications (ICMLA)*, pages 227–232. IEEE, 2016.

[191] Andras Rozsa, Ethan M Rudd, and Terrance E Boult. Adversarial diversity and hard positive generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 25–32, 2016.

[192] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[193] Cynthia Rudin. Why black box machine learning should be avoided for high-stakes decisions, in brief. *Nature Reviews Methods Primers*, 2(1):81, 2022.

[194] Esmaiel Saadatzadeh, Rahim Ali Abbaspour, and Alireza Chehreghan. An improvement in smartphone-based 3d indoor positioning using an effective map matching method. *Journal of Ambient Intelligence and Humanized Computing*, 14(10):13741–13771, 2023.

[195] Khaled Salah, M. Habib Ur Rehman, Nishara Nizamuddin, and Ala Al-Fuqaha. Blockchain for ai: Review and open research challenges. *IEEE Access*, 7:10127–10149, 2019.

[196] Muhammad Salah ud din, Muhammad Atif Ur Rehman, Rehmat Ullah, Chan-Won Park, Dae Ho Kim, and Byung seo Kim. Improving resource-constrained iot device lifetimes by mitigating redundant transmissions across heterogeneous wireless multimedia of things. *Digital Communications and Networks*, 2021.

[197] Wojciech Samek, Felix Sattler, Thomas Wiegand, and Klaus-Robert Müller. Concepts for federated learning, client classification and training data similarity measurement, April 7 2022. US Patent App. 17/526,739.

[198] Widi Satriaji and Retno Kusumaningrum. Effect of synthetic minority oversampling technique (smote), feature representation, and classification algorithm on imbalanced sentiment analysis. In *2018 2nd International Conference on Informatics and Computational Sciences (ICICoS)*, pages 1–5, 2018.

[199] Vikash Sehwag, Arjun Nitin Bhagoji, Liwei Song, Chawin Sitawarin, Daniel Cullina, Mung Chiang, and Prateek Mittal. Analyzing the robustness of open-world machine learning. In *Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security*, pages 105–116, 2019.

[200] Gökhan Şengül, Erol Ozcelik, Sanjay Misra, Robertas Damaševičius, and Rytis Maskeliūnas. Fusion of smartphone sensor data for classification of daily user activities. *Multimedia Tools and Applications*, 80(24):33527–33546, 2021.

[201] Uri Shaham, Yutaro Yamada, and Sahand Negahban. Understanding adversarial training: Increasing local stability of neural nets through robust optimization. *arXiv preprint arXiv:1511.05432*, 2015.

[202] Muhammad Shayan, Clement Fung, Chris JM Yoon, and Ivan Beschastnikh. Biscotti: A ledger for private and secure peer-to-peer machine learning. *arXiv preprint arXiv:1811.09904*, 2018.

[203] Shiqi Shen, Shruti Tople, and Prateek Saxena. Auror: Defending against poisoning attacks in collaborative deep learning systems. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 508–519, 2016.

[204] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[205] Johannes Stallkamp, Marc Schlipsing, Jan Salmen, and Christian Igel. The german traffic sign recognition benchmark: a multi-class classification competition. In *The 2011 international joint conference on neural networks*, pages 1453–1460. IEEE, 2011.

[206] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826, 2016.

[207] Pedro Tabacof and Eduardo Valle. Exploring the space of adversarial images. In *2016 international joint conference on neural networks (IJCNN)*, pages 426–433. IEEE, 2016.

[208] Jihoon Tack, Sihyun Yu, Jongheon Jeong, Minseon Kim, Sung Ju Hwang, and Jinwoo Shin. Consistency regularization for adversarial robustness. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 8414–8422, 2022.

[209] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International Conference on Machine Learning*, pages 6105–6114. PMLR, 2019.

[210] Sheng Tan, Yili Ren, Jie Yang, and Yingying Chen. Commodity wifi sensing in ten years: Status, challenges, and opportunities. *IEEE Internet of Things Journal*, 9(18):17832–17843, 2022.

[211] Data Quality Lab. Data collected by the Data Quality Lab. Accessed: September 10, 2023.

[212] Data Quality Lab. Sensor Selector Android OS Application. Accessed: September 10, 2023.

[213] Data Quality Lab. System Security Evaluation Android OS Application. Accessed: September 10, 2023.

[214] Data Quality Lab. Sensor Quality Assessment Android OS Application. Accessed: September 10, 2023.

[215] Linguistic Data Consortium. 2000 HUB5 English Evaluation Speech LDC2002S09, 2002. Web Download. Philadelphia: Linguistic Data Consortium. (Date last accessed 10-September-2023).

[216] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. Deeptest: Automated testing of deep-neural-network-driven autonomous cars. In *Proceedings of the 40th international conference on software engineering*, pages 303–314, 2018.

[217] Florian Tramer and Dan Boneh. Adversarial training and robustness for multiple perturbations. *Advances in neural information processing systems*, 32, 2019.

[218] Stacey Truex, Nathalie Baracaldo, Ali Anwar, Thomas Steinke, Heiko Ludwig, Rui Zhang, and Yi Zhou. A hybrid approach to privacy-preserving federated learning. In *Proceedings of the 12th ACM workshop on artificial intelligence and security*, pages 1–11, 2019.

[219] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Proceedings. 1991 IEEE computer society conference on computer vision and pattern recognition*, pages 586–587. IEEE Computer Society, 1991.

[220] Nikita O Tursukov, Ilya I Viksnin, Evgeny A Neverov, Elena L Sheinman, and Sergei S Chuprov. Evaluation of the effectiveness of neural networks based on the criteria for completing the object classification task. In *2023 XXVI International Conference on Soft Computing and Measurements (SCM)*, pages 120–122. IEEE, 2023.

[221] Udacity. Udacity self-driving car driving data 10/3/2016 (dataset-2-2.bag.tar.gz). Accessed on July 30, 2023.

[222] Maria Usova, Sergey Chuprov, Ilya Viksnin, Ruslan Gataullin, Antonina Komarova, and Andrey Iuganson. Model of smart manufacturing system. In *Intelligent Distributed Computing XIII*, pages 356–362. Springer, 2020.

[223] Maria Usova, Sergey Chuprov, Ilya I Viksnin, and Oksana Baranova. Model of secure informational messages for ensuring informational interaction in smart factory. In *MICSECS*, 2019.

[224] Maria Usova, Ilya I Viksnin, and Sergey Chuprov. Informational messages and space models application in smart factory concept. In *MICSECS*, 2020.

[225] Saeed Vahidian, Mahdi Morafah, and Bill Lin. Personalized federated learning by structured and unstructured pruning under data heterogeneity. In *2021 IEEE 41st International Conference on Distributed Computing Systems Workshops (ICDCSW)*, pages 27–34. IEEE, 2021.

[226] Kush R Varshney. Trustworthy machine learning and artificial intelligence. *XRDS: Crossroads, The ACM Magazine for Students*, 25(3):26–29, 2019.

[227] Daniel Vela, Andrew Sharp, Richard Zhang, Trang Nguyen, An Hoang, and Oleg S Pianykh. Temporal quality degradation in ai models. *Scientific Reports*, 12(1):11654, 2022.

[228] I Viksnin, Julia Lyakhovenko, Nikita Tursukov, Sergey Chuprov, and Ekaterina Sozinova. Empirical study on modeling of people behavior in emergency. In *CEUR Workshop Proceedings*, 2020.

[229] Ilia I Viksnin, Egor D Marinenkov, and Sergey S Chuprov. A game theory approach for communication security and safety assurance in cyber-physical systems with reputation and trust-based mechanisms. *Scientific and Technical Bulletin of Information Technologies, Mechanics and Optics*, 22(1):47–59, 2022.

[230] Ilya Viksnin, Sergey Chuprov, Maria Usova, and Danil Zakoldaev. Police office model for multi-agent robotic systems. In *IOP Conference Series: Materials Science and Engineering*, volume 497, page 012036. IOP Publishing, 2019.

[231] Ayush Vora, Leon Reznik, and Igor Khokhlov. Mobile road pothole classification and reporting with data quality estimates. In *2018 Fourth International Conference on Mobile and Secure Services (MobiSecServ)*, pages 1–6. IEEE, 2018.

[232] Grega Vrbančič and Vili Podgorelec. Transfer learning with adaptive fine-tuning. *IEEE Access*, 8:196197–196211, 2020.

[233] Hongyi Wang, Mikhail Yurochkin, Yuekai Sun, Dimitris Papailiopoulos, and Yasaman Khazaeni. Federated learning with matched averaging. *arXiv preprint arXiv:2002.06440*, 2020.

[234] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE journal on selected areas in communications*, 37(6):1205–1221, 2019.

[235] Tianhao Wang, Yuheng Zhang, and Ruoxi Jia. Improving robustness to model inversion attacks via mutual information regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11666–11673, 2021.

[236] Xiaoshun Wang and Yajing Yu. Wasserstein distance transfer learning algorithm based on matrix-norm regularization. In *2022 2nd International Conference on Algorithms, High Performance Computing and Artificial Intelligence (AHPCAI)*, pages 8–13. IEEE, 2022.

[237] Yuhao Wang, Vlado Menkovski, Ivan Wang-Hei Ho, and Mykola Pechenizkiy. Vanet meets deep learning: The effect of packet loss on the object detection performance. In *2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring)*, pages 1–5. IEEE, 2019.

[238] Mitchell Webber and Raul Fernandez Rojas. Human activity recognition with accelerometer and gyroscope: A data fusion approach. *IEEE Sensors Journal*, 21(15):16979–16989, 2021.

[239] Xuekai Wei, Mingliang Zhou, Sam Kwong, Hui Yuan, Shiqi Wang, Guopu Zhu, and Jingchao Cao. Reinforcement learning-based qoe-oriented dynamic adaptive streaming framework. *Information Sciences*, 569:786–803, 2021.

[240] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. A survey of transfer learning. *Journal of Big data*, 3(1):1–40, 2016.

[241] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295. PMLR, 2018.

[242] Kelvin Wong, Yanlei Gu, and Shunsuke Kamijo. Mapping for autonomous driving: Opportunities and challenges. *IEEE Intelligent Transportation Systems Magazine*, 13(1):91–106, 2020.

[243] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

[244] Pulei Xiong, Scott Buffett, Shahrear Iqbal, Philippe Lamontagne, Mohammad Mamun, and Heather Molyneaux. Towards a robust and trustworthy machine learning system development: An engineering perspective. *Journal of Information Security and Applications*, 65:103121, 2022.

[245] Hongli Xu, Zhuolong Yu, Xiang-Yang Li, Liusheng Huang, Chen Qian, and Taeho Jung. Joint route selection and update scheduling for low-latency update in sdns. *IEEE/ACM Transactions on Networking*, 25(5):3073–3087, 2017.

[246] Lei Xu, Alfredo Cuesta-Infante, Laure Berti-Equille, and Kalyan Veeramachaneni. R&r: Metric-guided adversarial sentence generation. *arXiv preprint arXiv:2104.08453*, 2021.

[247] Runhua Xu, Nathalie Baracaldo, Yi Zhou, Ali Anwar, and Heiko Ludwig. Hybridalpha: An efficient approach for privacy-preserving federated learning. In *Proceedings of the 12th ACM workshop on artificial intelligence and security*, pages 13–23, 2019.

[248] Weilin Xu, David Evans, and Yanjun Qi. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155*, 2017.

[249] Yonghui Xu, Sinno Jialin Pan, Hui Xiong, Qingyao Wu, Ronghua Luo, Huaqing Min, and Hengjie Song. A unified framework for metric transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 29(6):1158–1171, 2017.

[250] Zeli Xue. Routing optimization of sensor nodes in the internet of things based on genetic algorithm. *IEEE Sensors Journal*, 21(22):25142–25150, 2021.

[251] Timothy Yang, Galen Andrew, Hubert Eichner, Haicheng Sun, Wei Li, Nicholas Kong, Daniel Ramage, and Françoise Beaufays. Applied federated learning: Improving google keyboard query suggestions. *arXiv preprint arXiv:1812.02903*, 2018.

[252] Dong Yin, Yudong Chen, Ramchandran Kannan, and Peter Bartlett. Byzantine-robust distributed learning: Towards optimal statistical rates. In *International Conference on Machine Learning*, pages 5650–5659. PMLR, 2018.

[253] Xiaoyong Yuan, Pan He, Qile Zhu, and Xiaolin Li. Adversarial examples: Attacks and defenses for deep learning. *IEEE transactions on neural networks and learning systems*, 30(9):2805–2824, 2019.

[254] Huan Zhang, Minhao Cheng, and Cho-Jui Hsieh. Enhancing certifiable robustness via a deep model ensemble. *arXiv preprint arXiv:1910.14655*, 2019.

[255] Jie M Zhang, Mark Harman, Lei Ma, and Yang Liu. Machine learning testing: Survey, landscapes and horizons. *IEEE Transactions on Software Engineering*, 2020.

[256] Xialei Zhang, Xinyu Yang, Jie Lin, Guobin Xu, and Wei Yu. On data integrity attacks against real-time pricing in energy-based cyber-physical systems. *IEEE Transactions on Parallel and Distributed Systems*, 28(1):170–187, 2016.

[257] Xinglin Zhang, Zheng Yang, Wei Sun, Yunhao Liu, Shaohua Tang, Kai Xing, and Xufei Mao. Incentives for mobile crowd sensing: A survey. *IEEE Communications Surveys & Tutorials*, 18(1):54–67, 2016.

[258] Yuqian Zhang, Cun Mu, Han-Wen Kuo, and John Wright. Toward guaranteed illumination models for non-convex objects. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 937–944, 2013.

[259] Juncen Zhu, Jiannong Cao, Divya Saxena, Shan Jiang, and Houda Ferradi. Blockchain-empowered federated learning: Challenges, solutions, and future directions. *ACM Computing Surveys*, 55(11):1–31, 2023.

[260] Fuzhen Zhuang, Zhiyuan Qi, Keyu Duan, Dongbo Xi, Yongchun Zhu, Hengshu Zhu, Hui Xiong, and Qing He. A comprehensive survey on transfer learning. *Proceedings of the IEEE*, 109(1):43–76, 2020.

Appendices

Appendix A

First Appendix

A.1 Summary of novelties and differences of our FL Reputation and Trust-based approach compared to state-of-the-art ones

Authors	Title	They Describe	We Describe	Difference – Our Novelty
Konecný, J., McMahan, H. B., Yu, F. X., Richtárik, P., Suresh, A. T., & Bacon, D.	Federated Learning: Strategies for Improving Communication Efficiency	The FL model which includes the strategy of training a model locally on a number of distributed clients and the communication of model updates between the clients and the aggregation server. The paper proposed communication improvements for slow and unreliable networks.	We describe the invention that allows excluding potentially compromised local units from the global updates distribution, which allows to enhance training data privacy and robustness of the FL system. Moreover, we employ Reputation and Trust-based mechanisms that allow knowledge accumulation on the quality of updates sent by local units throughout the FL system operation.	We propose to analyze local model updates on the aggregation unit and exclude potentially compromised local units from the global model distribution. We address the potential privacy violation by decreasing the risk of data inference attacks implementation on the global model.
McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A.	Communication-Efficient Learning of Deep Networks from Decentralized Data	The employment of Federated Average (FedAvg) function for the FL environment, which allows performing a number of training rounds on the local device and then transmitting the local update for the aggregation. The approach allows decreasing the communication burden in comparison to FederatedSGD baseline algorithm.	We describe the invention that allows excluding potentially compromised local units from the global updates distribution, which allows to enhance training data privacy and robustness of the FL system. Moreover, we employ Reputation and Trust-based mechanisms that allow knowledge accumulation on the quality of updates sent by local units throughout the FL system operation.	We propose to analyze local model updates on the aggregation unit and exclude potentially compromised local units from the global model distribution. We address the potential privacy violation by decreasing the risk of data inference attacks implementation on the global model.
Fang, H., & Qian, Q.	Privacy Preserving Machine Learning with Homomorphic Encryption and Federated Learning	The approach of partially homomorphic encryption for the FL system. Using the approach, all the training data on the local clients is obfuscated, and all the ML-training operations are performed over this obfuscated data.	We describe the invention that allows excluding potentially compromised local units from the global updates distribution, which allows to enhance training data privacy and robustness of the FL system. Moreover, we employ Reputation and Trust-based mechanisms that allow knowledge	We propose to analyze local model updates on the aggregation unit and exclude potentially compromised local units from the global model distribution. Even if the data is obfuscated or encrypted, an adversary can still perform malicious manipulations over it,

			accumulation on the quality of updates sent by local units throughout the FL system operation.	so the global model performance can deteriorate.
Vahidian, S., Morafah, M., & Lin, B.	Personalized Federated Learning by Structured and Unstructured Pruning under Data Heterogeneity	The approach that allows obtaining personalized model from a client-level objective. A novel model's parameters aggregation technique, Sub-FedAvg, is proposed. The technique allows to find all the subnetwork(s) of clients with similar parameters and aggregate them.	We describe the invention that allows excluding potentially compromised local units from the global updates distribution, which allows to enhance training data privacy and robustness of the FL system. Moreover, we employ Reputation and Trust-based mechanisms that allow knowledge accumulation on the quality of updates sent by local units throughout the FL system operation.	We propose to analyze local model updates on the aggregation unit and exclude potentially compromised local units from the global model distribution. We address the potential privacy violation by decreasing the risk of data inference attacks implementation on the global model.
Cao, D., Chang, S., Lin, Z., Liu, G., & Sun, D.	Understanding Distributed Poisoning Attack in Federated Learning	The approach to eliminate poisoned local models from malicious participants during training. The poisoned models are excluded from the aggregation procedure.	We describe the invention that allows excluding potentially compromised local units from the global updates distribution, which allows to enhance training data privacy and robustness of the FL system. Moreover, we employ Reputation and Trust-based mechanisms that allow knowledge accumulation on the quality of updates sent by local units throughout the FL system operation.	In our approach, complementing to potentially compromised local units detection, exclude them from the global updates distribution procedure. By doing this, we decrease the risk of gaining unauthorized access to the global model by the adversary.
Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., ... & Seth, K.	Practical Secure Aggregation for Privacy-Preserving Machine Learning	They propose to enhance the security of communication between the local and aggregation units by incorporating cryptographic primitives. The local units should prove its identity with the help of specified cryptographic operation. The approach requires	We describe the invention that allows excluding potentially compromised local units from the global updates distribution, which allows to enhance training data privacy and robustness of the FL system. Moreover, we employ Reputation and Trust-based mechanisms that allow knowledge	We propose to analyze local model updates on the aggregation unit and exclude potentially compromised local units from the global model distribution. We address the potential privacy violation by decreasing the risk of data inference attacks implementation on the global model.

		Public Key Infrastructure establishment.	accumulation on the quality of updates sent by local units throughout the FL system operation.	
Bhowmick, A., Duchi, J., Freudiger, J., Kapoor, G., & Rogers, R.	Protection Against Reconstructio n and Its Applications in Private Federated Learning	They proposed to employ a Differential Privacy approach to encrypt the data at the data source and perform ML model training over the encrypted data.	We describe the invention that allows excluding potentially compromised local units from the global updates distribution, which allows to enhance training data privacy and robustness of the FL system. Moreover, we employ Reputation and Trust-based mechanisms that allow knowledge accumulation on the quality of updates sent by local units throughout the FL system operation.	We propose to analyze local model updates on the aggregation unit and exclude potentially compromised local units from the global model distribution. We address the potential privacy violation by decreasing the risk of data inference attacks implementation on the global model.
Flach, P.	Performance Evaluation in Machine Learning: The Good, the Bad, the Ugly, and the Way Forward	They describe the existing approaches to ML performance evaluation. The major advantages, flaws and peculiarities of these approaches are discussed.	We describe the invention that allows excluding potentially compromised local units from the global updates distribution, which allows to enhance training data privacy and robustness of the FL system. Moreover, we employ Reputation and Trust-based mechanisms that allow knowledge accumulation on the quality of updates sent by local units throughout the FL system operation.	N/A - we refer this paper to provide background on ML performance evaluation approaches.
Rathee, M., Shen, C., Wagh, S., & Popa, R. A.	ELSA: Secure Aggregation for Federated Learning with Malicious Actors	They propose a secure aggregation protocol for federated learning based on the concept of distributed trust with two aggregation servers.	We describe the invention that allows excluding potentially compromised local units from the global updates distribution, which allows to enhance training data privacy and robustness of the FL system. Moreover, we employ Reputation and Trust-based mechanisms that allow knowledge	We propose to analyze local model updates on the aggregation unit and exclude potentially compromised local units from the global model distribution. We address the potential privacy violation by decreasing the risk of data inference attacks implementation on the global model.

			accumulation on the quality of updates sent by local units throughout the FL system operation.	
Gehlhar, T., Marx, F., Schneider, T., Suresh, A., Wehrle, T., & Yalame, H.	SafeFL: MPC-friendly Framework for Private and Robust Federated Learning	They propose a framework for evaluating the effectiveness and performance of FL techniques that protect against both privacy inference and data poisoning attacks. They perform an evaluation of various aggregation schemes and report the accuracy through their framework. They also evaluate the computational and communication overhead.	We describe the invention that allows excluding potentially compromised local units from the global updates distribution, which allows to enhance training data privacy and robustness of the FL system. Moreover, we employ Reputation and Trust-based mechanisms that allow knowledge accumulation on the quality of updates sent by local units throughout the FL system operation.	We propose to analyze local model updates on the aggregation unit and exclude potentially compromised local units from the global model distribution. We address the potential privacy violation by decreasing the risk of data inference attacks implementation on the global model.
Mansouri, M., Önen, M., Jaballah, W. B., & Conti, M.	SoK: Secure Aggregation based on cryptographic schemes for Federated Learning	They propose a formal definition of secure aggregation based on cryptographic schemes and compare existing secure aggregation solutions in relation to federated learning.	We describe the invention that allows excluding potentially compromised local units from the global updates distribution, which allows to enhance training data privacy and robustness of the FL system. Moreover, we employ Reputation and Trust-based mechanisms that allow knowledge accumulation on the quality of updates sent by local units throughout the FL system operation.	We propose to analyze local model updates on the aggregation unit and exclude potentially compromised local units from the global model distribution. We address the potential privacy violation by decreasing the risk of data inference attacks implementation on the global model.
Corrigan-Gibbs, H., & Boneh, D.	Prio: Private, Robust, and Scalable Computation of Aggregate Statistics	They introduce a privacy-preserving system for collection of aggregate statistics. Their system is based on a new zero-knowledge cryptographic proof which facilitates private aggregation. This paper does not concentrate solely on Federated	We describe the invention that allows excluding potentially compromised local units from the global updates distribution, which allows to enhance training data privacy and robustness of the FL system. Moreover, we employ Reputation and Trust-based mechanisms	We propose to analyze local model updates on the aggregation unit and exclude potentially compromised local units from the global model distribution. We address the potential privacy violation by decreasing the risk of data inference attacks

		Learning. Rather, it presents a method to compute aggregate statistics in a secure way, which is applicable to Federated Learning. However, their approach does not rely on quantifiable definitions of Trust to facilitate a secure aggregation process.	that allow knowledge accumulation on the quality of updates sent by local units throughout the FL system operation.	implementation on the global model.
Cao, X., Fang, M., Liu, J., & Gong, N. Z.	FLTrust: Byzantine-robust Federated Learning via Trust Bootstrappin g	They propose a federated learning method to achieve Byzantine robustness against malicious clients. In their approach, the server collects a small portion of the initial training set and constructs its own model to bootstrap trust. Then they compare the server's model with the clients' models to determine trust-worthy clients and perform aggregation.	We describe the invention that allows excluding potentially compromised local units from the global updates distribution, which allows to enhance training data privacy and robustness of the FL system. Moreover, we employ Reputation and Trust-based mechanisms that allow knowledge accumulation on the quality of updates sent by local units throughout the FL system operation.	We propose to analyze local model updates on the aggregation unit and exclude potentially compromised local units from the global model distribution. We address the potential privacy violation by decreasing the risk of data inference attacks implementation on the global model.
Wen, D., Jeon, K. J., & Huang, K.	Federated Dropout — A Simple Approach for Enabling Federated Learning on Resource Constrained Devices	They propose the Federated Dropout scheme to tackle the communication and computation overhead in federated learning when the clients are communicating with the aggregation server. In their method, they generate several subnets with dropout at the server, and then communicate those models back to clients for updates. They claim their approach reduces both communication and computation overhead.	We describe the invention that allows excluding potentially compromised local units from the global updates distribution, which allows to enhance training data privacy and robustness of the FL system. Moreover, we employ Reputation and Trust-based mechanisms that allow knowledge accumulation on the quality of updates sent by local units throughout the FL system operation.	We propose to analyze local model updates on the aggregation unit and exclude potentially compromised local units from the global model distribution. We address the potential privacy violation by decreasing the risk of data inference attacks implementation on the global model.
Ma, Y., Woods, J.,	Flamingo: Multi-Round	Their proposed method tackles the problem of	We describe the invention that allows	We propose to analyze local model updates on

Angel, S., Polychroniadou, A., & Rabin, T.	Single-Server Secure Aggregation with Applications to Private Federated Learning	secure aggregation in federated learning by utilizing known cryptographic primitives. They claim that their design facilitates a more efficient training process.	excluding potentially compromised local units from the global updates distribution, which allows to enhance training data privacy and robustness of the FL system. Moreover, we employ Reputation and Trust-based mechanisms that allow knowledge accumulation on the quality of updates sent by local units throughout the FL system operation.	the aggregation unit and exclude potentially compromised local units from the global model distribution. We address the potential privacy violation by decreasing the risk of data inference attacks implementation on the global model.
Horvath, S., Laskaridis, S., Almeida, M., Leontiadis, I., Venieris, S., & Lane, N.	FjORD: Fair and Accurate Federated Learning under heterogeneous targets with Ordered Dropout	They employ the proposed ordered drop out technique in order to provide a fairer training process in federated learning process, so that even clients with very limited computational resources can participate in the federated learning process and contribute model updates.	We describe the invention that allows excluding potentially compromised local units from the global updates distribution, which allows to enhance training data privacy and robustness of the FL system. Moreover, we employ Reputation and Trust-based mechanisms that allow knowledge accumulation on the quality of updates sent by local units throughout the FL system operation.	We propose to analyze local model updates on the aggregation unit and exclude potentially compromised local units from the global model distribution. We address the potential privacy violation by decreasing the risk of data inference attacks implementation on the global model.
Lin, J., Du, M., & Liu, J.	Free-riders in Federated Learning: Attacks and Defenses	They propose a new "free-rider" attack and defense methods against federated learning, where attackers exploit the system without contributing any data. Two attack methods are presented: gradient-based and model-inversion. They introduce STD-DAGMM defense mechanism to detect and mitigate these attacks. Local updates are compressed, analyzed for	We describe the invention that allows excluding potentially compromised local units from the global updates distribution, which allows to enhance training data privacy and robustness of the FL system. Moreover, we employ Reputation and Trust-based mechanisms that allow knowledge accumulation on the quality of updates sent by local units throughout the FL system operation.	We propose to analyze local model updates on the aggregation unit and exclude potentially compromised local units from the global model distribution. Beyond clustering the local models provided for aggregation based on their parameters, we also employ Reputation and Trust indicators for knowledge accumulation. Our Reputation and Trust indicators provide the quantifiable knowledge-based

	reconstruction error, and normalized with activation ranking standard deviation. This combined vector is fed to a DAGMM for anomaly detection. Clients flagged as anomalies in multiple rounds are excluded from global update calculation. Only legitimate clients' weighted updates are used for global model aggregation.	measure of trust towards the local unit based on the model updates it provides for the aggregation throughout the learning process. We address the potential privacy violation by decreasing the risk of data inference attacks implementation on the global model, as we discard the local units with low trust from the aggregation and further communication.