Rochester Institute of Technology
RIT Digital Institutional Repository

Theses

4-2022

Learning Multi-step Robotic Manipulation Tasks through Visual Planning

Sulabh Kumra
sk2881@rit.edu

Follow this and additional works at: https://repository.rit.edu/theses

Recommended Citation
Kumra, Sulabh, "Learning Multi-step Robotic Manipulation Tasks through Visual Planning" (2022). Thesis. Rochester Institute of Technology. Accessed from

This Dissertation is brought to you for free and open access by the RIT Libraries. For more information, please contact repository@rit.edu.

Learning Multi-step Robotic Manipulation Tasks through Visual Planning

by

Sulabh Kumra

A Dissertation Submitted in Partial Fulfillment of
the Requirements for the Degree of
Doctor of Philosophy in Engineering

Supervised by

Dr. Ferat Sahin

Kate Gleason College of Engineering
Rochester Institute of Technology
Rochester, New York
April 2022

Learning Multi-step Robotic Manipulation Tasks through Visual Planning

by

Sulabh Kumra

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 Doctorate of Philosophy in Engineering.

Dr. Ferat Sahin
Department Head, Department of Electrical and Microelectronic Engineering
Primary Advisor

Dr. Andres Kwasinski
Professor, Department of Computer Engineering
Committee Member

Dr. Christopher Kanan
Associate Professor, Chester F. Carlson Center for Imaging Science
Committee Member

Dr. Jamison Heard
Assistant Professor, Department of Electrical and Microelectronic Engineering
Committee Member

Dr. Edward Hensel Jr
Director, PhD in Engineering program

© Copyright 2022 by Sulabh Kumra
All Rights Reserved

Biographical Note

Sulabh Kumra is a Roboticist. His research focuses on areas related to robotics, controls, and artificial intelligence. His primary research utilizes a combination of robotics, computer vision, and machine learning to develop an intelligent self-learning robotic system that can learn multi-step manipulation tasks in unstructured environments. Sulabh's research interests lie in collaborative robots in industrial automation, robotic manipulation, machine learning, and application of deep learning in robotics.

Sulabh Kumra received his Bachelors in Technology in the field of Electronics and Instrumentation Engineering from ITM University, India in 2013. There he built and led a team of 14 multidisciplinary engineers to develop India's first low-cost human-scale teleoperated humanoid robot called Dexto:Eka:. Sulabh graduated with a Master of Science in Electrical Engineering from Rochester Institute of Technology (RIT), Rochester, NY, USA in 2015. There, he worked as a teaching and research assistant at the Multi-Agent Biorobotics Laboratory. He then went on to pursue his Ph.D. in Engineering at RIT.

He is currently working as the Software Engineering Manager at OSARO Inc, San Francisco, CA, USA, where he is leading a team of engineers to build AI based robotic manipulation systems. He previously worked at Tesla and Xerox as Robotics and Controls Engineer. He has developed novel vision-based methods that enabled robots to interact intelligently with the physical world and improve themselves over time. His work has appeared in 19 international journal and conference publications. He is an active reviewer for IEEE Transactions on Robotics, IEEE Transactions on Systems, Man, and Cybernetics: Systems, IEEE Robotics and Automation Letters (RA-L), System of Systems Engineering Conference (SoSE), IEEE International Conference on Robotics and Automation (ICRA), IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), and IEEE International Conference on Automation Science and Engineering (CASE). He has received best paper award twice at international conferences.

Abstract

Multi-step manipulation tasks in unstructured environments are extremely challenging for a robot to learn. Such tasks interlace high-level reasoning that consists of the expected states that can be attained to achieve an overall task and low-level reasoning that decides what actions will yield these states. A model-free deep reinforcement learning method is proposed to learn multi-step manipulation tasks.

This work introduces a novel Generative Residual Convolutional Neural Network (GR-ConvNet) model that can generate robust antipodal grasps from n-channel image input at real-time speeds (20ms). The proposed model architecture achieved a state-of-the-art accuracy on three standard grasping datasets. The adaptability of the proposed approach is demonstrated by directly transferring the trained model to a 7 DoF robotic manipulator with a grasp success rate of 95.4% and 93.0% on novel household and adversarial objects, respectively.

A novel Robotic Manipulation Network (RoManNet) is introduced, which is a vision-based model architecture, to learn the action-value functions and predict manipulation action candidates. A Task Progress based Gaussian (TPG) reward function is defined to compute the reward based on actions that lead to successful motion primitives and progress towards the overall task goal. To balance the ratio of exploration/exploitation, this research introduces a Loss Adjusted Exploration (LAE) policy that determines actions from the action candidates according to the Boltzmann distribution of loss estimates. The effectiveness of the proposed approach is demonstrated by training RoManNet to learn several challenging multi-step robotic manipulation tasks in both simulation and real-world. Experimental results show that the proposed method outperforms the existing methods and achieves state-of-the-art performance in terms of success rate and action efficiency. The ablation studies show that TPG and LAE are especially beneficial for tasks like multiple block stacking.

Acknowledgements

Through my years here, there are many people I would like to thank.

Above all, I am grateful for the guidance and tutelage that has and continues to be provided by my advisor Dr. Ferat Sahin. His constant support and advice are what made this thesis possible.

My gratitude extends to my dissertation committee, Dr. Andres Kwasinski, Dr. Christopher Kanan, and Dr. Jamison Heard, who have each provided helpful feedback and guidance with this research.

I am very grateful to my wife, Shirin, who has stood by me through all my struggles, my absences, my ups and downs. She has been incredibly supportive of me throughout this endeavor. I am thankful to my parents, my sister, and my in-laws for their continued support and encouragement.

I am thankful to all my peers at the Multi-Agent Bio-robotics Laboratory, Rochester Institute of Technology. Each of you has contributed their unique roles in the completion of this work.

I acknowledge Research Computing at the Rochester Institute of Technology for providing computational resources and support that have contributed to the results reported in this work.

List of Contributions

Highlights of the Work

- A novel deep learning based framework to produce grasps by combining a heterogeneous collection of Convolutional Neural Network models trained using different objective functions.
- A uni-modal approach for detecting good robotic grasps for parallel plate grippers using the five-dimensional representation.
- A novel multi-modal architecture to predict grasp from RGB-D images. Experiments show that the proposed architecture outperforms current state-of-the-art methods in terms of both accuracy and speed.
- A dual-module robotic system that predicts, plans, and performs antipodal grasps for single or multiple objects in the scene. We open-sourced the implementation of the proposed inference[1] and control[2] modules.
- A novel generational residual convolutional neural network architecture (GR-ConvNet) that predicts suitable antipodal grasp configurations for objects in the camera's field of view at real-time speeds of 20ms.
- An end-to-end model architecture Robotic Manipulation Network (RoManNet) [3] to efficiently learn the action-value functions and generate accurate action candidates from visual observation of the scene.

[1] Available at https://github.com/skumra/robotic-grasping
[2] Available at https://github.com/skumra/baxter-pnp
[3] Available at https://github.com/skumra/romannet

- A Task Progress based Gaussian reward function that uses a sub-task indicator function and an overall task progress function to compute the reward for each action in a multi-step manipulation task.

- Address the challenge of balancing the ratio of exploration/exploitation by introducing a Loss Adjusted Exploration manipulation policy that selects actions according to the Boltzmann distribution of loss estimates.

Publications

- **Sulabh Kumra**, Shirin Joshi, Ferat Sahin. "Learning Multi-step Robotic Manipulation Policies from Visual Observation of Scene and Q-value Predictions of Previous Action", *2022 International Conference on Robotics and Automation (ICRA), pp. 8245-8251.*[1]

- **Sulabh Kumra**, Shirin Joshi, Ferat Sahin. "Learning Robotic Manipulation Tasks via Task Progress Based Gaussian Reward and Loss Adjusted Exploration", *IEEE Robotics and Automation Letters (RA-L) 7.1 (2021), pp. 534-541.* [2]

- **Sulabh Kumra**, Shirin Joshi, Ferat Sahin. "Antipodal Robotic Grasping using Generative Residual Convolutional Neural Network", *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 9626-9633.* [3]

- Shirin Joshi, **Sulabh Kumra**, Ferat Sahin. "Robotic Grasping using Deep Reinforcement Learning", *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE), pp. 1461-1466.* [4]

- **Sulabh Kumra** and Christopher Kanan. "Robotic grasp detection using deep convolutional neural networks", *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 769-776.* [5]

Contents

List of Figures

List of Tables

List of Abbreviations

Chapter 1

Introduction

Recent advances in robotics and automated systems have led to the expansion of autonomous capabilities and the use of more intelligent machines in various applications. The capability of adapting to changing environments is a necessary skill for task generalized robots. Machine learning plays a key role in creating such general purpose robotic solutions. However, most robots are still developed analytically and based on expert knowledge of the application background. Even though this is considered an effective method, it is an arduous and time-consuming approach and has limitations for generalized applicability. Due to the recent successful results of deep learning methods in computer vision and robotics applications, many robotics researchers have started implementing deep learning methods in their research.

The type of the learning that is applied varies according to the feedback mechanism, the process used for training data generation, and the data formulation. The learning problem can vary from perception to state abstraction, through to decision making. Deep Learning, a branch of machine learning, describes a set of modified machine learning techniques that, when applied to robotic systems, aims to enable robots to autonomously perform tasks that come naturally to humans. Inspired by the biological nervous system, a network of parallel and simultaneous mathematical operations are performed directly on the available data to obtain a set of representational heuristics between the input and output data. These heuristics are then used in decision making. Deep learning models have proven effective in diverse classification and detection problems, and there is a great deal of interest in expanding their utilization into other domains.

The grasp or grasping pose describes how a robotic end-effector can be arranged in a given

image plane to successfully pick up an object. The grasping pose for any given object is determined through a grasp detection system. Any suitable perception sensors, including cameras or depth sensors, can be used to visually identify grasping poses in a given scene. Grasp planning relates to the path planning process that is required to securely grab the object and maintain the closed gripper contacts to hold and lift the object from its resting surface. Planning usually involves mapping the image plane coordinates to the robot world coordinates for the detected grasp candidate. The control system describes certain closed-loop control algorithms that are used to control the robotic joints or Degrees of Freedom (DOF) to reach the grasping pose while maintaining a smooth reach.

1.1 Motivation

Traditionally analytical approaches, also known as hard coding, involve manually programming robots with the necessary instructions for performing a given task. These control algorithms are modelled based on the expert human knowledge of the robot and its environment during the specific task. The outcome of this approach explains the kinematic relationship between the robot's parameters and its world coordinates. Ju *et al.* further suggests that the kinematic model helps in further optimizing the control strategies. However direct mapping of results from a kinematic model to the robot joint controller is inherently open-loop and is identified to cause task space drifts. Therefore, Ju *et al.* further suggests the use of closed loop control algorithms to address these drifts.

Even though such hard coded manual teaching is known to achieve efficient task performance, such an approach has limitations; in particular the program is restricted to the situations predicted by the programmer, but in cases where frequent changes of robot programming is required, due to changes in the environment or other factors, this approach becomes impractical. Unstructured environments remain a large challenge for intelligent robots that would require a complex analytical approach to form the solution. While deriving of models requires a lot of data and knowledge of the physical parameters relating to the robotic task, use of more dynamic robotic actuators make it nearly impossible to model the physics, thus they conclude that manual teaching as an efficient but exhaustive approach. In such cases, empirical methods will provide an increased cognitive and adaptive capability to the robots while reducing or completely removing the need to manually model a robotic solution. Early work in empirical methods takes a classical form that explores the adaptive

and cognitive capability of robots to learn tasks from demonstration. Various non-linear Regression techniques, Gaussian process, Gaussian mixture models, and support vector machines are some of the popular techniques related to this context. Although these techniques provided some cognition for the robots, task replication is limited to the demonstrated tasks.

1.2 Outline

The chapters of this thesis were written so that they can be read independently. The rest of the thesis is structured as follows:

Chapter 2 gives an introduction to types of industrial manipulators and their applications. This chapter also discusses the current market for industrial robotics and the challenges faced with current industrial robotic manipulators.

Chapter 3 introduces the problem of robotic grasp detection and reviews current state-of-the-art algorithms based on deep learning in robotic grasping. The architectures and datasets presented in previous work are critically evaluated.

Chapter 4 presents a novel robotic grasp detection system that predicts the best grasp pose of a parallel plate robotic gripper for novel objects using the RGB-D image of the scene. Results produced by the proposed approach redefines the state-of-the-art for robotic grasp detection.

Chapter 5 presents a modular robotic system to tackle the problem of generating and performing antipodal robotic grasps for unknown objects from n-channel image of the scene. A novel generation residual convolutional neural network (GR-ConvNet) model is proposed that can generate robust antipodal grasps from n-channel input at real-time speeds.

Chapter 6 presents a model-free deep reinforcement learning method to learn multi-step manipulation tasks. A Robotic Manipulation Network (RoManNet), which is a vision-based model architecture, is proposed to learn the action-value functions and predict manipulation action candidates. A Task Progress based Gaussian (TPG) reward function that computes the reward based on actions that lead to successful motion primitives and progress towards the overall task goal is introduced.

Finally, in **chapter 7** conclusions are drawn and future work is discussed.

Chapter 2

Industrial Robot Manipulators

Industrial Robot Manipulators are machines which are used to manipulate or control material without making direct contact. Originally, it was used to manipulate radioactive or biohazardous objects which can be difficult for a person to handle. But now they have been widely applied in a broad range of fields such as assembly, production, electric welding, spraying and painting, food packaging, and component installation. In recent years, several companies have seen aggressive investment in related technologies or M&A activities, including Fanuc, ABB, KUKA, Yaskawa, Mitsubishi, Hitachi, Sony, Toyota, Honda, and Samsung.

The IFR has defined industrial robots as automatically controlled, reprogrammable, multipurpose manipulator programmable in three or more axes, which may be either fixed in place or mobile for use in industrial automation applications. The IFR has further classified industrial robots into six types based on their mechanical structures: articulated robots, Cartesian/linear/gantry robots, Selective Compliance Articulated Robot Arm (SCARA) robots, cylindrical robots, parallel robots, and others. Fig. 2.1 shows the most common configurations of industrial robot arm design.

2.1 Types of Manipulators

In industries many types of industrial manipulators are used according to their requirements. Some of them are listed below.

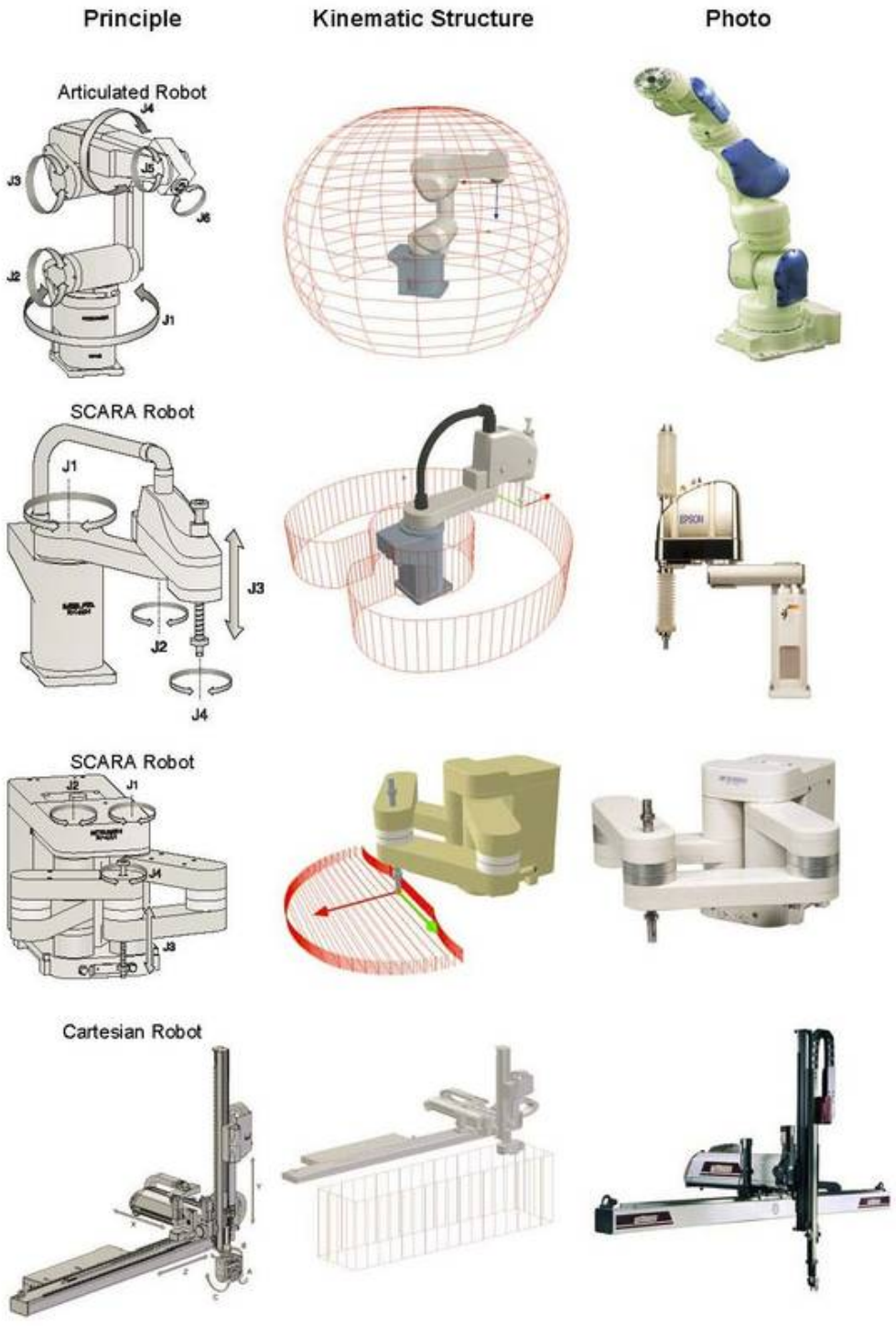


Figure 2.1: Robot Arm Design Configurations

2.1.1 Cartesian Coordinate Robot

In this industrial robot, its 3 principle axis have prismatic joints or move linearly through each other. Cartesian robots are best suited for dispensing adhesives as in automotive industries. The primary advantage of Cartesians is that they are capable of moving in multiple linear directions. And also they are able to do straight-line insertions and are easy to program. The disadvantages of Cartesian robot are that it takes too much space, as most of the space in this robot is unused.

2.1.2 SCARA Robot

SCARA robots have motions similar to that of a human arm. These machines comprise both a 'shoulder' and 'elbow' joint along with a 'wrist' axis and vertical motion. SCARA robots have 2 revolute joints and 1 prismatic joint. SCARA robots have limited movements but it is also its advantage as it can move faster than other 6 axis robots. It is also very rigid and durable. They are mostly used in precise application which require fast, repeatable and articulate point to point movements such as palletizing, DE palletizing, machine loading/unloading and assembly. Its disadvantages are that it has limited movements and it is not very flexible.

2.1.3 Cylindrical Robot

It is basically a robot arm that moves around a cylinder shaped pole. A cylindrical robotic system has three axes of motion: the circular motion axis and the two linear axes in the horizontal and vertical movement of the arm. So it has 1 revolute joint, 1 cylindrical and 1 prismatic joint. Today, Cylindrical Robots are less used and are replaced by more flexible and fast robots, but it has a very important place in history as it was used for grappling and holding tasks much before six axis robots were developed. Its advantage is that it can move much faster than the Cartesian robot if two points have the same radius. Its disadvantage is that it requires effort to transform from a Cartesian coordinate system to a cylindrical coordinate system.

2.1.4 Articulated Robot

Articulated Robot or Programmable Universal Manipulation Arm (PUMA) is the most commonly used industrial robot in assembly, welding operations, and university laboratories. It is more similar

to human arm than SCARA robot. It has great flexibility more than SCARA but it also reduces its precision. Therefore, they are used in less precision work such as assembling, welding and object handling. It has 3 revolute joints, but not all the joints are parallel, the second joint from the base is orthogonal to the other joints. This makes PUMA to be compliant in all three axis X, Y and Z. Its disadvantage is that it has less precision, so it cannot be used in critical and high precision applications.

2.2 Type of Path Generated

Industrial robots can be programmed from a distance to perform their required and preprogrammed operations with different types of paths generated through different control techniques. The three different types of paths generated are the point-to-point path, the controlled path, and the continuous path.

2.2.1 Point-to-Point Path

Robots programmed and controlled in this manner are programmed to move from one discrete point to another within the robot's working envelope. In automatic mode of operation, the exact path taken by the robot will vary slightly due to variations in velocity, joint geometries, and spatial locations of points. This difference in paths is difficult to predict and therefore can create a potential safety hazard to personnel and equipment.

2.2.2 Controlled Path

The path or mode of movement ensures that the end of the robot's arm follows a predictable (controlled) path and orientation as the robot travels from point to point. The coordinate transformations required for this hardware management are calculated by the robot's control system computer. Observations that result from this type of programming are less likely to present a hazard to personnel and equipment.

2.2.3 Continuous Path

A robot whose path is controlled by storing a large number or close succession of spatial points in memory during a teaching sequence is a continuous path controlled robot. During this time, and while the robot is being moved, the coordinate points in space of each axis are continually monitored on a fixed time basis, e.g., 60 or more times per second, and placed into the control system's computer memory. When the robot is placed in the automatic mode of operation, the program is replayed from memory and a duplicate path is generated.

2.3 Type of Industrial Robots

Over the last 20 years, the industrial market has been the primary adopter of robotics technology and accounted for the majority of all robot spend. The industrial robot market is characterized as robots used in manufacturing or assembly applications within automotive, electronic, or other machining industries. Largely due to the size and power of most industrial robots in use today, these systems are typically installed in caged environments with minimal human contact for various safety reasons. However, due to advancements in computer vision and motion sensing capabilities, a new type of industrial robot has emerged known as collaborative robots, or co-bots. Although the cobot market is small today, it is believed that this subcategory of the industrial market will see extraordinary growth over the next 10 years. The industrial market is broken down into two subcategories: traditional and collaborative robots (see Fig. 2.2).

2.3.1 Traditional Industrial Robots

Most traditional robots are installed in caged environments away from people, allowing them to handle heavy payloads and operate at fast speeds. However, the high integration costs associated with industrial robots limits the flexibility of these machines. Additionally, traditional robots typically require programming from advanced software engineers, which can drive the total cost of ownership well over $100K per system.

Figure 2.2: Industrial Robots

2.3.2 Collaborative Industrial Robots (Co-bots)

Co-bots are built with multiple motion and force detection sensors, which makes it safer for these systems to collaborate with humans. Programming co-bots is also less sophisticated than traditional robots, which lowers overall cost and improves flexibility. With the average selling price on co-bots ranging from $25 – 45k, robot automation is now accessible outside of large industrial manufacturing. That said, limited payload capacity and a slower operating speed are two drawbacks to co-bots in the market today.

2.4 Industrial Robotics Market

The Global Industrial Robotics Market was valued at $37,875 million in 2016, and is projected to reach $70,715 billion by 2023, growing at a CAGR of 9.4% from 2017 to 2023 [6].

The global industrial robotics market is driven by a surge in labor charges worldwide, which, in turn, has forced manufacturers to replace human labor with machines. Asia and Europe are the key growth regions of the world, with leading players, namely ABB, Fanuc, KUKA, Kawasaki, and the Yaskawa Electric Corporation being based out in the region.

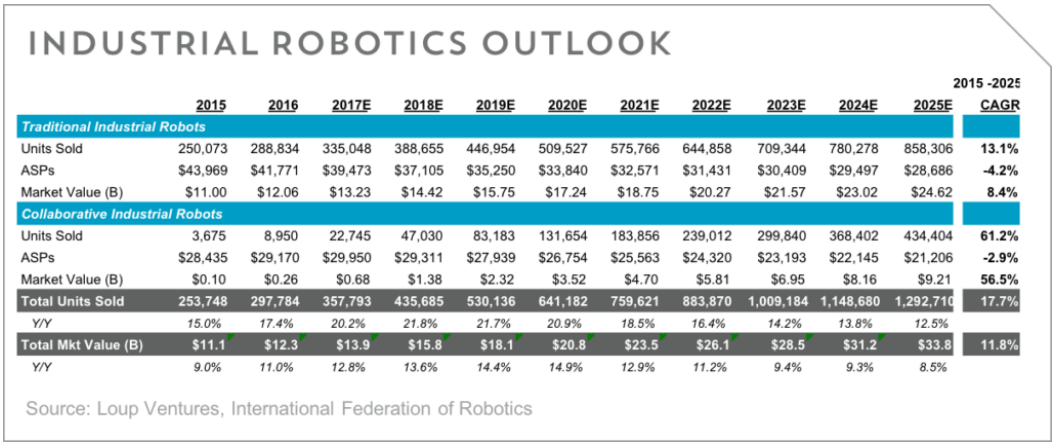
The global industrial production output is expected to witness moderate growth during the forecast period. The demand for industrial robotics is majorly observed in industries such as automobiles and heavy engineering. However, the increased need for automation in non-conventional areas, such as microelectronics, has increased the demand for industrial robotics. Hence, an auxiliary channel utilizing industrial robotics has surfaced in recent years. The heavy engineering sector is also responsible for the increased demand for industrial robotics.

Figure 2.3: IFR estimated worldwide operational stock of industrial robots [6]

The global industrial robotics market is affected by several factors, such as the usage of industrial robotics in the manufacturing industry, the increased demand for automation activities in the industry, reduction in custom duties, and evolving robotics & artificial intelligence industry. Furthermore, the high cost of industrial robotics solutions is a major hindrance to the growth of the industrial robotics market.

In terms of units, it is estimated that by 2020 the worldwide stock of operational industrial robots will increase from about 1,828,000 units at the end of 2016 to 3,053,000 units (see fig. 2.3). This represents an average annual growth rate of 14 percent between 2018 and 2020. In Australasia, the operational stock of robots is estimated to increase by 16% in 2017, by 9% in the Americas, and by 7% in Europe. Since 2016, the largest number of industrial robots in operation has been in China. In 2020, this will amount to about 950,300 units, considerably more than in Europe (611,700 units). The Japanese robot stock will increase slightly in the period between 2018 and 2020. About 1.9 million robots will be in operation across Asia in 2020. This is almost equal to the global stock of robots in 2016.

The increase in demand for automation and rapid growth in industrialization foster the use of industrial robots. Increased adoption of these robots in automotive, food & beverages, machinery, and precision & optics industries for automating processes, leading to increased efficiency, reduced



INDUSTRIAL ROBOTICS OUTLOOK

	2015	2016	2017E	2018E	2019E	2020E	2021E	2022E	2023E	2024E	2025E	2015 -2025 CAGR
Traditional Industrial Robots												
Units Sold	250,073	288,834	335,048	388,655	446,954	509,527	575,766	644,858	709,344	780,278	858,306	13.1%
ASPs	$43,969	$41,771	$39,473	$37,105	$35,250	$33,840	$32,571	$31,431	$30,409	$29,497	$28,686	-4.2%
Market Value (B)	$11.00	$12.06	$13.23	$14.42	$15.75	$17.24	$18.75	$20.27	$21.57	$23.02	$24.62	8.4%
Collaborative Industrial Robots												
Units Sold	3,675	8,950	22,745	47,030	83,183	131,654	183,856	239,012	299,840	368,402	434,404	61.2%
ASPs	$28,435	$29,170	$29,950	$29,311	$27,939	$26,754	$25,563	$24,320	$23,193	$22,145	$21,206	-2.9%
Market Value (B)	$0.10	$0.26	$0.68	$1.38	$2.32	$3.52	$4.70	$5.81	$6.95	$8.16	$9.21	56.5%
Total Units Sold	253,748	297,784	357,793	435,685	530,136	641,182	759,621	883,870	1,009,184	1,148,680	1,292,710	17.7%
Y/Y	*15.0%*	*17.4%*	*20.2%*	*21.8%*	*21.7%*	*20.9%*	*18.5%*	*16.4%*	*14.2%*	*13.8%*	*12.5%*	
Total Mkt Value (B)	$11.1	$12.3	$13.9	$15.8	$18.1	$20.8	$23.5	$26.1	$28.5	$31.2	$33.8	11.8%
Y/Y	*9.0%*	*11.0%*	*12.8%*	*13.6%*	*14.4%*	*14.9%*	*12.9%*	*11.2%*	*9.4%*	*9.3%*	*8.5%*	

Source: Loup Ventures, International Federation of Robotics

Figure 2.4: Industrial robotics outlook in terms of traditional and collaborative robots [6]

human errors, and increased safety of the workforce. Some painting and pick & place functions are carried out by robotic arms, which reduce complexity and possibility of errors. Surveillance & security have improved due to the implementation of drones. Industrial robots deliver high-quality products and services, increase the capabilities of manual labor in terms of efficiency, provide better customer services, and efficiently manage processes.

According to IFR, a total of 253,748 industrial robots were delivered in 2015, and the total market value grew 9.0% y/y to $11.1B. Of all the industrial units shipped, we believe 250,073 of industrial robots were in the form of traditional systems, while the remaining 3,675 units were collaborative machines. Over the next 10 years, it is anticipated that the traditional industrial market will see healthy growth, but due to lower costs and higher flexibility, it is anticipated that the cobot market will see much faster adoption. The total amount of cobot units shipped is expected to increase from 8,950 in 2016 to 434,404 by 2025, representing a 61.2% CAGR. Over this time frame we anticipate costs to continue to come down, but believe the total co-bot market value will exceed $9.0B by 2025. While it is expected that co-bots will drive overall industrial growth, it is anticipated that the traditional market to also see steady adoption. It is believed that the traditional market alone will represent a $24 billion market by 2025. In total, the industrial robotics market is expected to grow 11.8% annually to more than $33 billion over the next 10 years.

2.5 Challenges in Industrial Robotics

It's pretty clear that robot use at home and in industry is accelerating. "Robot density", measured as the ratio of individual robots per 10,000 human employees, is rising at an annual rate of 5% to 9%, according to the IFR. Yet both industrial and personal robots have a long way to go before we really see the sci-fi future we have all been dreaming of and even then, that future is likely to look different than we have imagined.

Here is a list of top 10 challenges for a new wave of industrial robotics:

1. Motors: Most motors are high speed + low torque, and robots need low speed + high torque output.

2. UX: Robots that are viable for unstructured environments will have a person operating it as a tool. Is it possible to apply trainable AI to create a gradually more intuitive UX? People (without technology backgrounds) must be able to be trained quickly in supervising these machines.

3. Low-cost 3D Cameras: Localization and grasp planning needs 3D, and the most-loved 3D cameras are often taken off the market when their respective vendors are acquired by larger companies (ex: PrimeSense/Apple). Note, 3D cameras cannot replace "safety-rated" LiDAR as light bumpers.

4. Point-Cloud Data Interoperability between Different 3D Cameras: It would be nice to have point cloud fusion across different sensors because short-range and long-range cameras require different optics.

5. Grasping: Object surface detection with 3D cameras is essential for motion planning (planning how the robot arm delivers a gripper towards an object).

6. Grippers: The gripper problem isn't solved, but that's because the physics of a single gripper doesn't handle a wide range of objects very well (small grippers for small objects, large grippers for large objects, payload capacity, etc...)

7. Exception Handling: What happens when the robot makes a mistake? Does it puncture a hole in the workspace? Does an operator manually restart it while a digital management system

records the error, so the whole system recovers? Think this through.

8. WiFi: WiFi works terribly in large/crowded spaces and places with metal racking.

9. Security & Hackability: IT teams from many large companies do not want to risk having an asset on an open network that could be accessed by an outside entity.

10. Sensors (& Data): A robot's window to the world is only as good as its data input from sensors. All sensors need to have clean data that minimizes latency and power requirements for data processing.

2.6 Future Trend: Smart Factory

Industry 4.0, linking the real-life factory with virtual reality, will play an increasingly important role in global manufacturing. As obstacles like system complexities and data incompatibility are overcome, manufacturers will integrate robots into factory-wide networks of machines and systems. Robot manufacturers are already developing and commercializing new service models: these are based on real-time data collected by sensors that are attached to robots. Analysts predict a rapidly growing market for cloud robotics in which data from one robot is compared to data from other robots in the same or different locations. The cloud network allows these connected robots to perform the same activities. This will be used to optimize parameters of the robot's movement such as speed, angle or force. Ultimately, the advent of big data in manufacturing could redefine the industry boundaries between equipment makers and manufacturers.

Chapter 3

Deep Learning Methods in Robotic Manipulation

For robots to attain more general purpose utility, grasping is a necessary skill to master. Such general-purpose robots may use their perception abilities in order to visually identify grasps for a given object. A grasp describes how a robotic end-effector can be arranged on top of an object to securely grab it between the robotic gripper and successfully lift it without slippage. Traditionally, grasp detection requires expert human knowledge to analytically form the task-specific algorithm, but this is an arduous and time-consuming approach. During the last five years, deep learning methods have enabled significant advances in robotic vision, natural language processing, and automated driving applications. The successful results of these methods have driven robotics researchers to explore the application of deep learning methods in task generalised robotic applications. This chapter reviews the current state-of-the-art in regards to the application of deep learning methods to generalised robotic grasping and discusses how each element of the deep learning approach has improved the overall performance of robotic grasping.

Traditional analytical approaches, also known as hard coding, involve manually programming a robot with the necessary instructions to perform a given task. These control algorithms are modelled based on expert human knowledge of the robot and its environment in the specific task. The outcome of this approach explains the kinematic relationship between the robot's parameters and its world coordinates. Ju *et al.* [16] suggested that the kinematic model helps to further optimize the control

strategies. However, direct mapping of results from a kinematic model to the robot joint controller is inherently open-loop and is identified to cause task space drifts. Therefore, they have, in addition, recommended the use of closed loop control algorithms to address these drifts [17].

Even though such hard coded manual teaching is known to achieve efficient task performance, such an approach has limitations; in particular, the program is restricted to the situations predicted by the programmer, but in cases where frequent changes of robot programming is required, due to changes in the environment or other factors, this approach becomes impractical [18]. According to Ju *et al.* [17], unstructured environments remain a great challenge for intelligent robots that would require a complex analytical approach to form the solution. While the derivation of models requires a great deal of data and knowledge of the physical parameters relating to the robotic task, the use of more dynamic robotic actuators makes it nearly impossible to model the physics, thus they conclude that manual teaching is an efficient but exhaustive approach [17]. In such cases, empirical methods will provide an increased cognitive and adaptive capability to the robots, while reducing or completely removing the need to manually model a robotic solution [19]. Early work in empirical methods takes a classical form that explores the adaptive and cognitive capabilities of robots to learn tasks from demonstration. Non-linear Regression techniques, Gaussian process, Gaussian mixture models, and Support Vector Machines are some of the popular techniques related to this context [20]. Although these techniques have provided some level of cognition for the robots, the task replication is limited to the demonstrated tasks.

Deep learning has recently made significant advancements in the application of computer vision, scene understanding, robotic arts, and natural language processing. Due to the convincing results that have been achieved in the scope of computer vision, there is an increasing trend towards implementation of deep learning methods in robotics applications. Many recent studies show that the unstructured nature of a generalized robotics task makes it significantly more challenging. However, to advance the state-of-the-art of robotic applications, it is necessary to create a generalized robotic solution for various industries such as offshore oil rigs, remote mine sites, manufacturing assembly plants, and packaging systems where work environments and scenarios can be highly dynamic. A desired primary ability for these general-purpose robots is the ability to grasp and manipulate objects to interact with their work environment. Visual identification and manipulation of objects is a relatively simple task for humans to perform, but for a robot, this is a very challenging task that

involves perception, planning, and control. Grasping can enable robots to manipulate obstacles in the environment or change the state of the environment if necessary. Early work such as [7, 21] show how far researchers have advanced the research methods in robotic grasping. These studies discuss early attempts to grasp novel objects using empirical methods.

Object grasping is challenging due to the wide range of factors such as different object shapes and unlimited object poses. Successful robotic grasping systems should be able to overcome this challenge to produce useful results. Unlike robots, humans can almost immediately determine how to grasp a given object. Robotic grasping currently performs well below human object grasping benchmarks but is continually being improved given the high demand. A robotic grasping implementation has the following sub-systems:

- Grasp detection sub-system: To detect grasp poses from images of the objects in their image plane coordinates
- Grasp planning sub-system: To map the detected image plane coordinates to the world coordinates
- Control sub-system: To determine the inverse kinematics solution of the previous sub-system

The grasp detection sub-system is the key entry point for any robotic grasping research. A review of current deep learning methods in grasp detection is provided in the subsequent sections of this chapter. A popular deep learning method that has been applied in most related literature is the Deep Convolutional Neural Network (DCNN) or sometimes referred to as the CNN due to the heavy involvement of convolutional layers in their architectures. It is evident that there are two approaches to apply a CNN to a problem:

- Create an application specific CNN model
- Use the complete or part of a pre-existing CNN model through transfer learning

Creating a proprietary application-specific CNN model requires a deep understanding of the concept and a reasonable level of experience with CNNs. Therefore, most researchers that implement CNNs in their grasp detection work have opted for transfer learning given the reduced number of parameters to be dealt with. Training of such a CNN requires a large volume of data [22].

The data can be labeled or unlabeled depending on whether a supervised or unsupervised training method is used. Training is the process of tuning the network parameters according to the training data. Some studies [10] [9] focus on simplifying the problem of grasp detection and build on the transfer learning model to improve the results. While there are several platforms to implement deep learning algorithms, most studies have used TensorFlow, Theano, or Matlab. With the recent advancements of software applications and programming languages, there are now more streamlined tools such as Keras, Caffe or DarkNet to implement the same functionality of former deep learning frameworks but in an easier and more efficient way. Although most recent deep learning approaches for robotic grasping follow purely supervised learning, software platforms such as NVIDIA ISAAC encourage unsupervised learning methods with the support of virtual simulation capabilities.

3.1 Robotic Grasp Detection

Grasp detection is identified as the ability to recognise the grasping points or the grasping poses for an object in any given image [23]. As shown in Figures 1 and 2, a successful grasp describes how a robotic end-effector can be orientated on top of an object to securely hold the object between its gripper and pick the object up. As humans, we use our eyesight to visually identify objects in our vicinity and find out how to approach them in order to pick them up. Similarly, visual perception sensors on a robotic system can be used to produce information on the environment that can be interpreted in a useful format [9]. A mapping technique is necessary to classify each pixel of the scene on the basis of belonging or not belonging to a successful grasp. Recent robotic grasping work has used several different definitions for successful grasp configurations [10] [9]. In this regard, a representation or a definition of a good grasp is necessary. This section reviews some of the promising grasp representations and their method of detection. Section 3.1.1 discusses several grasp configurations and how they are represented in images. Section 3.1.2 discusses how these grasp representations are detected from images.

3.1.1 Grasp Representation

In most of the earlier works, grasps were represented as points on images of actual scenes or from 3D mesh models based on simulations. Using a supervised learning approach, Saxena *et al.* [7]

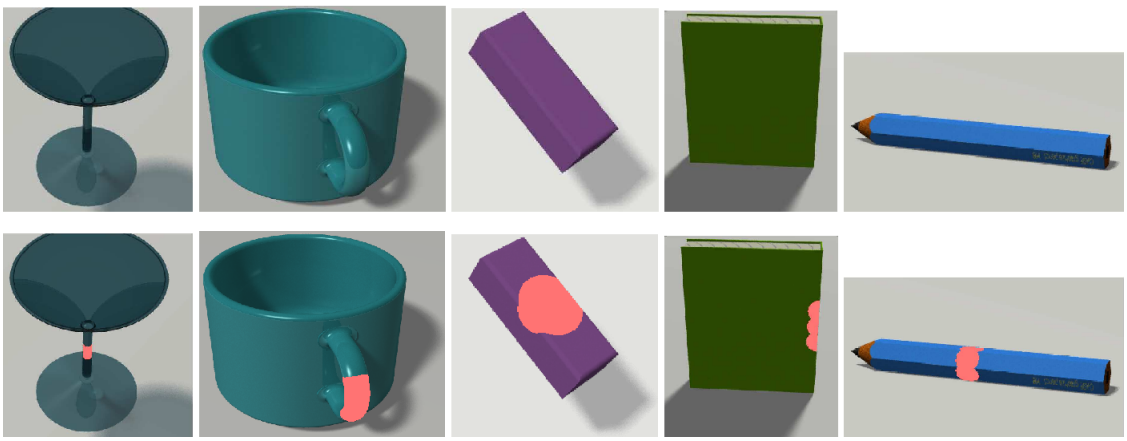


Figure 3.1: Five object classes used for training with their grasping points marked on the images. The objects are a Martini glass, a mug, an eraser, a book, and a pencil.[7]

investigated a regression learning method to infer the 3D location of a grasping point in a Cartesian coordinate system. They used a probabilistic model over possible grasping points while considering the uncertainty of the camera position. Extending their investigation, they had discretized the 3D workspace in order to find the grasping point g, given by $g = (x, y, z)$. They reported that by using two or more images captured from different angles it would simplify the grasp point inference and also referred to the smaller graspable regions on the images as grasp points as shown in Figure 3.1. In their reinforcement learning approach for grasp point detection, Zhang *et al.* [24] simply defined a grasp as a point in a 2D image plane. A major drawback of such point defined grasps, however, was that it only determined where to grasp an object and it did not determine how wide the gripper had to be opened or the required orientation for the gripper to successfully grasp the object.

As a way to overcome this limitation, another popular grasp representation that has been proposed is the oriented rectangle representation. According to Jiang *et al.* [8], their grasping configuration has a seven-dimensional representation containing the information of a Grasping point, Grasping orientation, and Gripper opening width. In world coordinates, their grasp representation, G, is stated as $G = (x, y, z, \alpha, \beta, \gamma, l)$. Their grasp representation is shown in Figure 3.2. The red lines represent the opening or closing width of the gripper along with the direction of the motion. The blue lines represent the parallel plates of the end-effector.

Simplifying the previously introduced seven-dimensional grasp rectangle representation from [8], Lenz *et al.* [10] proposed a five-dimensional representation. This was based on the assumption

Figure 3.2: Grasping rectangle representations. (a) The representation by Jiang *et al.* [8]: Top vertex (r_G, c_G), length m_G, width n_G and its angle from the x-axis, for a kitchen utensil. There can be multiple ground-truth grasps defined as shown. (b) The simplified representation by Redmon *et al.* [9] for a hammer, showing its grasp centre at (x, y) oriented by an angle of θ from its horizontal axis. The rectangle has a width and height of w and h respectively.

of a good 2D grasp being able to be projected back to 3D space. While they failed to evaluate their approach, Redmon *et al.* [9] confirmed the validity of the method with their own results. They further supported the statements by Jiang *et al.* [8] and Lenz *et al.* [10] that the detection of grasping points in this manner was analogous to object detection methods in computer vision but with an added term for gripper orientation. Adapting the method of [10] [8], they also presented a slightly updated representation of a grasp rectangle, as shown in Figure 3.2.

Another grasp representation introduced in more recent research is the combined location and orientation representation. In [11], the authors used the simple $G = (x, y, \theta)$ representation that dropped the dimensional parameters (h, w). The dimensional parameters provided a sense of the physical limitations for certain end-effectors. Similar representations are used in [25] [26]. This representation described a grasp in a 2D image plane. This representation was improved by Calandra *et al.* to include the 3D depth information by adding the z coordinates to the representation, resulting in a grasp representation, $G_z = (x, y, z, \theta)$ [27]. The G_z grasp representation was also used by Murali *et al.* [28] in their approach to detect robotic grasps through the use of tactile feedback and visual sensing.

From the three grasp representations described in this section, the rectangular representation can be identified as the most commonly used for any grasp detection applications. Although a detailed analysis of the relative suitability of the approaches has not been performed, the literature survey suggests that any preference is application specific. Lenz *et al.* [10] argued that, in most cases, the

depth information can be manually controlled specific to the application. Therefore the rectangle representation can be selected as the most suitable grasp representation in most cases.

Pixel-wise grasp representations were used when structured grasps were not useful. Ku *et al.* [29] used convolutional layer activations to find the anthropomorphic grasping points in images. They created a mask that represented the grasping points for the robotic index finger and the thumb. The mask contains all pixels that are part of the grasp. Their method only works for cuboid and cylindrical shaped objects. They reported that only one trial failed from the complete set of 50 trials, and they have managed to achieve an average success rate of 96%.

In applications where simple object localisation translates back to the simple pick-up points in the 2D image plane, dense captioning was used to localise objects in order to pick them up. In their work for the Amazon Picking Challenge, Schwarz *et al.* [30] used popular dense captioning [31] to localize objects in images. Dense captioning provides a textual description of each region of the image and it can be used to identify or localise objects in an image. During testing, they successfully picked up 10 objects from the 12 set of test objects and their fine-tuned system responded within 340 milliseconds during the testing.

These graspable region representations are widely used in picking or sorting objects in clutter when there are no particular requirements with respect to the order in which objects are picked up. More structured grasp representations are generally employed in conjunction with object recognition in order to grasp the identified objects [9]. The works discussed in this section demonstrate that a consistent grasp representation method must be adopted in order to start working with learning algorithms for the detection of robotic grasps. The ground truth labels should have the optimal number of parameters to represent a grasp while ensuring that it is not over-defined. The five-dimensional grasp representation originally presented by Lenz *et al.* [10] for a 2D image should, thus, be further explored.

3.1.2 Grasp Detection

The conventional analytical method of robotic grasp detection is performed on the premise that certain criteria such as object geometry, physics models, and force analytics are known [32]. The grasp detection applications are built based on a model developed with this information. The modelling of such information is often challenging due to the current fast-changing industry requirements. An

alternative approach is to use empirical methods, also known as data-driven approaches, that rely on previously known successful results. These methods are developed using existing knowledge of object grasping or by using simulations on real robotic systems [33]. A major drawback of analytical methods is that they rely on the assumption that the object parameters are known, therefore they cannot be used for a generalised solution [33]. There are two types of empirical approaches in robotic grasp detection:

1. Methods that use learning to detect grasps and use a separate planning system for grasp planning

2. Methods that learn a visuomotor control policy in a direct image-to-action manner

In the literature, direct grasp detection has been carried out using two different techniques. The most popular one is to detect structured grasp representations from images. An alternative approach is to learn a grasp robustness function. Both techniques require a separate grasp planning system to execute the grasp. During the last few years, there has been a growing interest into learning a visuomotor control policy using deep learning. The introduction of tools such as NVIDIA Isaac has enabled the extensive use of reinforcement learning in simulated environments with domain adaptation. These visuomotor control policy learning methods do not require a separate grasp planning system.

The most popular method for structured grasp detection was the sliding window approach proposed by Lenz *et al.* [10]. In their approach, a classifier is used to predict if a small patch of the image contains a potential grasp. The image is divided into a number of small patches and each patch is run through the classifier in an iterative process. The patches that contain higher ranking grasps are considered as candidates and pushed as outputs. This method yielded a detection accuracy of 75% and a processing time of 13.5 s per image. Similar results were reported from the studies by Wang *et al.* [34] and Wei *et al.* [35] who followed a similar approach. Guo *et al.* [36] used the reference rectangle method to identify graspable regions of an image. The locations of the reference rectangle were identified using the sliding window approach. Due to the repetitive scanning method for identifying graspable regions of images, this method was largely considered unsuitable where a real-time detection speed is necessary. As an alternative, Redmon *et al.* [9] proposed the one-shot detection method.

In most one-shot detection methods, a direct regression approach is used to predict a structured grasp output. In these approaches, the structured output represents the oriented grasp rectangle parameters in the image plane coordinates. In the first one-shot detection approach, the authors argued that a faster and more accurate method was necessary and proposed to use transfer learning techniques to predict grasp representation from images [9]. They reported a detection accuracy of 84.4% in 76 milliseconds per image. This result produced a large performance boost compared to the then state-of-the-art method, the sliding window. The one-shot detection method assumed that each image contained one graspable object and predicted one grasp candidate as opposed to the iterative scanning process of the sliding window approach. Therefore, it was evident that most work in one-shot detection followed deep transfer learning techniques to use pre-trained neural network architectures.

Although the preferred method for one-shot detection is the direct regression of the grasp representation, there were numerous occasions where the combined classification and regression techniques were employed for one-shot detection. While arguing that the orientation predictions of a structured grasp representation lay in a non-Euclidean space, where the standard regression loss (L2) had not performed well, Chu *et al.* [37] proposed to classify the orientation among 19 different classes in the range of [0 , 360]. They used a direct regression method to predict the bounding box of the grasp. The authors reported a detection accuracy of 94.4% with Red Green Blue (RGB) images. Building on the concept by Guo *et al.* [36], Zhou *et al.* [38] proposed to use anchor boxes for predefined regions of the images. Each image was divided into N × N regions. The orientation of the anchor box was classified between k classes. The authors argued that the k can be a variable integer. By default it was set to k = 6. The angles ranged between [-75, 75]. They achieved a detection accuracy of 97.74% for their work. The literature reasoned that these improvements were achieved as it was easier to converge to a classification during the training and the associated errors were minimal, but such a classification would limit the output to a predefined set of classes [11].

Learning a grasp robustness function has also been the central idea of many studies in deep grasp detection. The researchers used this function to identify the grasp pose candidate with the highest score as the output. Grasp robustness described the grasp probability of a certain location or an area of an image [26]. Binary classification was a well researched technique for this approach that classified the grasp points as valid or invalid (1 or 0). Using end-to-end learning, Ten Pas *et al.*

[39] performed a binary classification to identify graspable regions in a dense clutter of objects. They presented a 77% detection accuracy with passive point cloud data. In their work, Lu *et al.* [40] performed a CNN based grasp probability study to achieve a detection accuracy of 75.6% for previously unseen novel objects and 76.6% for previously seen objects during the training.

A method to learn an optimal grasp robustness function was proposed by Mahler *et al.* [23]. They considered the robustness as a scalar probability in the range of [0, 1]. The authors compiled a dataset known as Dex-Net 2.0 with 6.7 million point clouds and analytic grasp quality metrics with parallel-plate grippers planned using robust quasi-static grasp wrench space analysis on a dataset of 1500 3D object mesh models. They further trained a grasp quality convolutional network (GQ-CNN) that was used to learn a robustness metric for grasp candidates. They tested their CNN with their dataset which achieved an accuracy of 98.1% for grasp detection. Robust grasp detection is explored in [41]. Johns *et al.* reported that they achieved a grasp success rate of 75.2% with minor gripper pose uncertainties and 64.8% with major gripper pose uncertainties. They described the gripper pose uncertainties to be associated with varying shapes and contours associated with the objects.

The literature survey suggested that a direct mapping of images to robot actions could be predicted by learning a visuomotor control policy. This method would not require a separate grasp planning system, and thus was also considered a pixel-to-action method. Mahler *et al.* [26] proposed a method to find deep learnt policies to pick objects from clutter. The authors reported that by using a transfer learning technique with their previous findings in [23], they achieved a grasp detection accuracy of 92.4%. When they tested their learnt policies on robotic grasping, they achieved a success rate of 70% with five trials for each of 20 objects of the test dataset. The winning team from the Amazon Picking Challenge 2017, Zeng *et al.* [42] proposed a visuomotor control policy prediction method for images of objects in clutter. The authors proposed an action space with four individual actions: (a) suction down; (b) suction side; (c) grasp down; and (d) flush grasp. They reported a maximum accuracy of 96.7% for grasping and 92.4% for suction with the Top-1 confidence percentile [42]. Zhang *et al.* [24] proposed a method to use reinforcement learning [43] to determine the action to extend the robot end-effector to a point in a 2D image plane. Closely following the proposed deep Q network by Mnih *et al.* [43], the authors managed to adapt it to a robotic system using synthetic images. In testing, their system achieved a 51% success rate in reaching the

target point [24]. They further concluded that these results were largely affected by not having an optimal domain adaptation from synthetic to realistic scenes.

3.2 Grasp Training Data

Research has consistently shown that deep learning requires a large volume of labeled data to effectively learn the features during the training process [22]. This requirement is also apparent in supervised learning methods in robotic grasp detection [10, 9]. In recently published work, researchers either use training data from a third party or introduce their own application specific proprietary data sources or methods to automate the data generation [11, 23]. Johns *et al.* [41] highlighted that the major challenge with deep learning is the need for a very large volume of training data, thus they opted to generate and use simulated data for the training process. Another challenge of training deep neural networks is the lack of domain specific data as mentioned by Tobin *et al.* [44]. They proposed a method to generate generalised object simulations to address this challenge, although it has not yet been proven how effective the results can be. For real-time applications, the use of simulated data and the availability of 3D object models are not practically achievable. As a way to overcome this, there are reports of network pre-training as a solution when there is limited domain specific data [9].

Brownlee [45] specified that annotations of the available data will be more important if the learning was purely supervised and less important for unsupervised learning. He further described the importance of the three subsets of data for training, validation, and testing. In [9, 46], the authors had followed the same argument. A comprehensive explanation can also be found in [47]. The literature survey suggests that, in addition to larger training datasets, domain specific data are necessary for effective results.

3.2.1 Datasets

Goodfellow *et al.* [22] stated that the performance of a simple machine learning algorithm relied on the amount of training data as well as the availability of domain specific data. The recent publications suggested that the availability of training data is a prevailing challenge for this learning method. Some researchers combined datasets to create a larger dataset while others collected and

annotated their own data.

The CGD from is a popular grasp dataset that was compiled for most transfer learning approaches in robotic grasping. The CGD was created with grasp rectangle information for 240 different object types and it contained about 885 images, 885 point clouds and about 8019 labelled grasps including valid and invalid grasp rectangles. The grasps were specifically defined for the parallel plate gripper found on many robotic end-effectors. The CGD appeared in a number of research studies during the recent past, which might suggest that it has a reasonable diversity of examples for generalised grasps [9]. The recent trend of using Red Green Blue and Depth (RGB-D) for learning to predict grasps was covered with the CGD dataset through the inclusion of point cloud data by its creators. Lenz *et al.* [10] argued that having the depth information would result in a better depth perception for an inference system that was trained on depth data. A sense of good and bad grasps was also necessary to differentiate a better grasp from the alternatives [10, 9]. Therefore, the CGD could be selected as a suitable dataset for its quality and adaptability. The CGD was extensively used in [10, 9, 36].

In the grasp detection work by Wang *et al.* [34], the authors used the Washington RGB-D dataset [48] for its rich variety of RGB-D images. The authors self-annotated as they preferred to combine the resulting dataset with the CGD. The authors further stated that the combined Washington data instances of 25,000 with the 885 instances from the CGD would help in pre-training a deep network [34].

When application-specific data were necessary, researchers provided intuitive methods for data collection. Murali *et al.* [28] used a previously learned grasping policy to collect valid grasp data and performed random grasps to collect invalid grasp data. They have collected data for 52 different objects. Calandra *et al.* [27] collected data from 9269 grasp trials for 106 unique objects. Pinto *et al.* [11] stated how time consuming it was to collect data for robotic grasping and proposed a novel approach inspired from reinforcement learning. Their approach would predict centre points for grasps from a policy learned using reinforcement learning and the orientation was classified for 18 different classes using grasp probability. The authors scaled the data collection to 50,000 grasp trials using 700 robotic hours. They used a Mixture of Gaussians (MOG) background subtraction that identified graspable regions in images to avoid random object-less spaces in images. Levine *et al.* [49] further improved this approach through the collection of grasping data from nearly

900,000 grasp trials using 8 robots.

3.2.2 Multi-Modal Data

Use of multi-modal data has become popular in many research studies into robotic grasp pose detection. Early work from Saxena *et al.* [7] stated that most grasping work assumed prior knowledge of the 2D or 3D model of the object to be grasped, but such approaches encounter difficulties when attempting to grasp novel objects. The authors experimented with depth images for five different objects in their training. They reported the grasp success rates for basic objects such as mugs, pens, wine glasses, books, erasers, and cellphones. An overall success rate of 90% with a mean absolute error of 1.8 cm was reported.

Following this work, Jiang *et al.* [8] scaled the problem space to 194 images of nine classes. They stated that the availability of multi-modal data could be useful in identifying edges and contours in the images to clearly differentiate graspable regions. Lenz *et al.* [10] supported the same claim in their work that used multi-modal RGB-D data. In a few recent transfer learning applications, the authors used the multi-modality in a way that overcame the three-channel data limitation with existing pre-trained CNNs. The authors in [9, 46] replaced the Blue channel in RGB images with depth disparity images and created 3-channel RG-D images.

In [28], Murali *et al.* explored using tactile sensing to complement the use of visual sensors. This method involved a re-grasping step to accurately grasp the object. They reported a success rate of 85.92% with a deep network and 84.5% with an Support Vector Machine (SVM). A similar approach was followed by Calandra *et al.* [27] in their work on using tactile sensing in robotic grasp detection.

Our literature survey indicates there are several types of multi-modalities involved in grasp pose detection research with the most popular one being the RGB-D data. Evidence suggests that the added benefit of edge and contour information in RGB-D images has an advantage over uni-modal RGB images. There is not yet enough evidence to suggest whether tactile sensing has an added advantage over depth imaging for grasp detection work using RGB images. However, a major challenge with respect to using RGB-D data, however, is the access to suitable training datasets.

3.2.3 Domain Adaptation and Simulated Data

As pointed out by Tobin *et al.* [44], most of the applications lacked domain-specific data. While arguing the importance of a large volume of domain specific data, the authors proposed a method to use physics simulations to generate domain specific data using 3D mesh models for a set of primitive shapes. Most of the work that has used simulation and 3D model data relies on domain adaptation to its real-world equivalent set of objects. Bousmalis *et al.* [50] conducted several experiments to verify the domain adaptation capability of a deep grasp detection application that was trained on 3D mesh models randomly created by the authors. The authors randomly mixed simulated data with realistic data to compile a dataset of 9.4 million data instances. Using this a grasp success rate of 78% was achieved. In a similar approach, Viereck *et al.* [25] proposed a method for learning a closed-loop visuomotor controller from simulated depth images. The authors generated about 12,500 image-action pairs for the training. They reported a grasp pose detection success rate of 97.5% for objects in isolation, and 94.8% for objects in clutter. Mahler *et al.* [23] suggested populating a dataset containing physics based analyses such as caging, grasp wrench space (GWS) analyses and grasp simulation data for different types of object shapes and poses. They further suggested that cloud computing could be leveraged to train a convolutional neural network with this dataset that would in turn, predict a robustness metric for a given grasp instead of directly predicting a grasp. The proposed dataset was called Dex-Net 2.0 [23] and contained about 6.7 million point clouds and analytic grasp quality metrics with parallel-jaw grasps planned using robust quasi-static GWS analysis on a dataset of 1500 3D object models [23].

3.2.4 Summary

Mahler *et al.* [23] concluded that human annotation is a tedious process that requires months of work and the simulations would have to be run for a large number of iterations on a robotic system. With the limited availability of domain specific data, Redmon *et al.* [9] proposed to use pre-training. Pre-training assumes that by using the weights of the convolutional overhead of a CNN model that was trained on a large dataset such as ImageNet [51] would transfer the universal filtration capabilities to a smaller dataset, providing better results compared to the usual training approach. Even though most prior work used 3D simulations to find suitable grasp poses for objects, Redmon *et al.* [9] stated that, despite those previous works having performed well, they required a known 3D model

of the object to calculate a grasp. This 3D model would not be known a priority and the complex modelling techniques of forming the 3D model was beyond the capacity for most of the general purpose robots as their desired primary function was faster adaptation to dynamic scenarios [9]. In such cases, a learning algorithm would produce the necessary results provided that there were enough, domain-specific data instances for the training.

In conclusion, the number of training data plays a key role in the outcome of the trained algorithm. Some approaches (e.g., [49, 11]) try to reduce or completely avoid the challenges of compiling such huge datasets. In cases where an extended information set is necessary to produce a grasp prediction, a dataset such as the Dex-Net 2.0 [23] could be used. However, for most generalized grasp prediction networks, CGD would be an optimal starting point considering its adaptability to more generalized object shapes and poses with the added benefit of the inclusion of depth information.

3.3 Convolutional Neural Networks for Grasp Detection

Most recent work in robotic grasp detection apply different variations of convolutional neural networks to learn the optimal end-effector configuration for different object shapes and poses [23, 10, 11, 9]. They do so by ranking multiple grasp configurations predicted for each object image instance. Ranking is done based on the learned parameters from the representation learning capability of deep learning. As opposed to the manual feature design and extraction steps of classical learning approaches, deep learning can automatically learn how to identify and extract different application specific feature sets [22]. The authors of [22, 43] explained the importance of the CNN architecture towards learning.

In analytical approaches, various grasping application specific parameters such as closure properties and force analysis are combined to successfully model the grasps [32]. Closure properties describe the force and momentum exerted at the point of contact, also known as a Grasp Wrench. Depending on the level of friction at each of these points, the point of contact could be further elaborated. According to Bicchi *et al.* [32], force analysis describes the required grasping force that should be applied by the robotic gripper on the object to grasp it securely without slipping or causing damage. Kinematic modelling between the contact points is a function that describes the

relative motion between two different contact points. Reviews (e.g., [32]) suggest how practically impossible it would be to prepare a generalised grasping model using just analytical data. However, given how well certain learning algorithms [9, 10] performed in the past, it could be concluded that using a visual representation of successful grasps as training data with these learning algorithms would result in usable generalized solutions.

3.3.1 Architecture

A DCNN is built with multiple layers to extract information representations. Goodfellow *et al.* [22] has stated that the representation of the learning process of a deep neural network has similar attributes to the method through which information is processed by the human brain. During the last five years, there have been many active improvements for DCNN architectures. Most of these approaches use ImageNet tests for benchmarking. By inspecting many CNN methods that were originally evaluated on ImageNet data, it is evident that all of them have followed the general structure shown in Figure 5. The literature suggests that lower level features are identified using the convolutional layer, while application specific features are extracted by the fully connected portion of the network where pooling and activations are widely employed. This suggests that the results on the ImageNet data provide a reasonably useful evaluation of the architecture even though it is not specific to robotic grasping.

The literature survey suggests two types of convolutional layer placements in DCNNs. Early approaches used a stacked architecture where each layer was placed one after the other. More recent DCNNs have used convolutional layers in parallel. Szegedy *et al.* [52] reported that this trend was accelerated due to the availability of increased computational capabilities.

Both AlexNet and VGG-Net are stacked deep neural network architectures. With AlexNet, Krizhevsky *et al.* [53] produced a reduced error rate of 16.6%. Simonyan *et al.* further reduced the error rates to 7.0% with the introduction of VGG-Net [54]. Redmon *et al.* [9] were the first authors to implement AlexNet with their work in robotic grasp detection. They fine-tuned the DCNN architecture to accommodate their hardware. Their model is shown in Figure 3.3. Their direct regression model that was trained on RG-D images achieved an accuracy of 84.4% and the MultiGrasp model that divided an original image to N × N sub-images achieved an accuracy of 88%. Their work was later followed by Watson *et al.* [46] who achieved an accuracy of 78% with

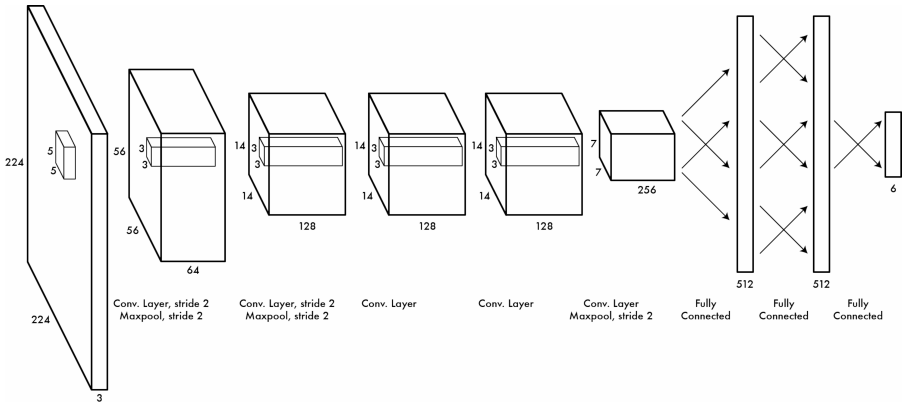


Figure 3.3: Neural network model proposed by Redmon *et al.* [9]

one fold cross validation.

Modern dense DCNN architectures are developed under the premise that deeper networks are capable of extracting more advanced features from data. Szegedy *et al.* [52] reported that the drawbacks of increasing the depth of a DCNN are two-fold. In order to train such deep network models there should be a distinguishable variation between the training data and this was challenging even with human labelling. When the depth of a DCNN is increased, the number of trainable parameters automatically increases, which requires higher computational power for training [52]. Therefore they suggested sparsely connected deep network architectures. They proposed their DCNN architecture known as GoogleNet, with a reduced error rate of 6.8% in ImageNet testing [51]. Following that, He *et al.* [55] proposed a DCNN architecture with skip connections that further reduced the error rates to 3.57% using their ResNet architecture. By combining [55] with their original approach in [56], Szegedy *et al.* proposed the Inception-ResNet architecture [57].

Lenz *et al.* [10] developed their two cascaded CNN models for grasp detection using a sliding window approach. The first neural network model extracted higher level features such as grasp locations whereas the larger second network verified the valid grasps from those detected. Figure 3.4 shows the architecture proposed by Lenz *et al.* . In the first stage, the authors used a variant of the Sparse Auto Encoder [58] to initialise the weights of the hidden layers. Pre-training in this way was a necessary step to avoid overfitting. The authors reported a grasp detection accuracy of 75% that was tested with a Baxter robot by carrying out object grasping. The creation of application specific DCNNs has received greater attention recently. When there has been enough higher quality training data available, researchers used their own custom neural network models. Most of these works were

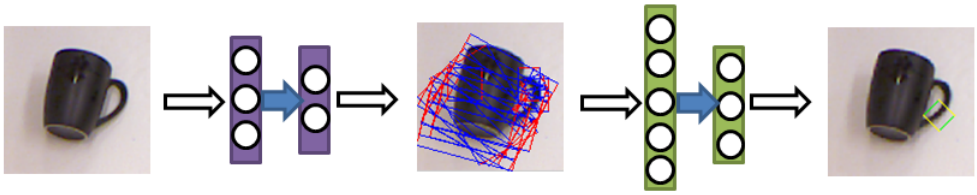


Figure 3.4: Two cascaded CNN based model by Lenz *et al.* [10]

motivated from recent DCNN success stories in ImageNet classification tests [51]. Lu *et al.* [59] used a custom architecture for their work in multi-fingered grasp prediction. The max-pooling and rectified linear units were used as activation functions in their work. They further concluded the adaptability of their work into the realm of two-fingered grasp detection. Detection accuracies of 75.6% for novel objects and 76.6% for previously seen objects were claimed.

Szegedy *et al.* in [52] stated that advances in the quality of image recognition had relied on newer ideas, algorithms, and improved network architectures as well as more powerful hardware, larger training datasets, and bigger learning models. They further commented that neither the deep networks nor the bigger models alone would result in such improvements but combining them n to create a deeper architecture would suggest these improvements over the classical theories. They experimentally presented that by increasing the depth (the number of deep levels) and the width (the number of units at each level) of a deep network would improve the overall network performance. However, it would also require a commensurately larger set of training data that would result in the following drawbacks:

1. Larger datasets would result in increased features to be extracted while limited datasets would result in overfitting.

2. Deeper networks would require increased computational resources during the training.

3.3.2 Transfer Learning Techniques

The most successful robotic grasp detection work has used transfer learning methods to achieve accuracies close to 90%. Any transfer learning approach includes the following steps:

1. Data pre-processing

2. Pre-trained CNN model

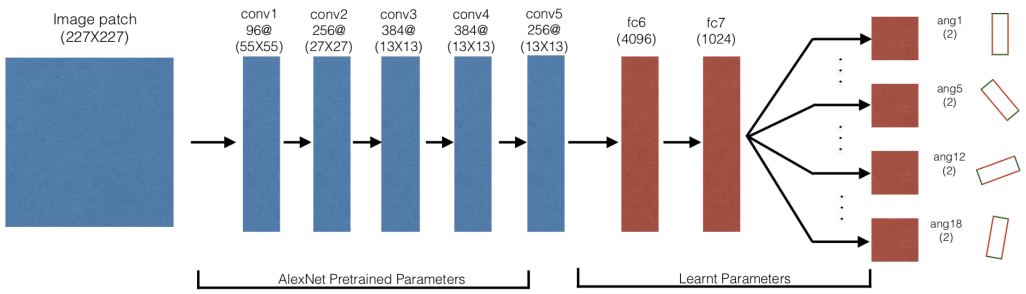


Figure 3.5: 18-way binary classifier by Pinto *et al.* [11]

Compared to image classification, robotic grasp detection requires the capability of a DCNN to identify grasp configurations for novel objects. This requires training on generalised object scene images. Therefore, most researchers limit their pre-processing techniques to just accommodate CNN input dimensions such as image width and the number of channels. Unlike in image classification, the ground truth data for grasping were less augmented. Redmon *et al.* [9] reported the minimum amount of necessary pre-processing for RGB-D datasets as centre cropping of images and replacing the blue channel of RGB images with depth data while normalising the depth data to [0, 255] range, which is the default RGB colour space range. While following the exact same procedure in [9], Watson *et al.* [46] normalised all RGB values and grasp labels to [0, 1] arguing that training targets should be in the same range as the training data. In their method, Pinto *et al.* [11] resized the images to 227×227 which was the input size for their model. Their network had a similar architecture to AlexNet [53] as shown in Figure 3.5.

Due to the problem of overfitting with the limited available datasets, the deep learning robotic grasp detection literature indicates that many authors have used pre-training. Pre-training was identified as a transfer learning technique where the deep network was pre-trained on larger datasets prior to the training on domain specific data. The weights of the convolutional layers that were originally learned during the pre-training were kept frozen during the training on domain specific data.

In their one-shot detection method, Redmon *et al.* [9] used the AlexNet [53] convolutional network architecture in compiling their network model [9] which achieved a grasp detection accuracy of 84.4%. The same approach was used in [46] with similar results reported. They modified the orientation parameter from the grasp representation while arguing that the angle predictions are

two-fold (positive or negative) [9]. Therefore, the authors replaced θ with $(sin2\theta, cos2\theta)$ following the trigonometric definitions. The argument was further supported in [8].

Even though the transfer learning made it less challenging to use a pre-trained model for the convolutional part of a DCNN, researchers still had to design the fully connected part of the DCNN. Although there is not enough evidence to determine the optimal number of units required, Redmon *et al.* [9] used two fully connected layers that had 512 units in each individual layer. In DCNN training applications, the popular learning optimiser is the Stochastic Gradient Decent (SGD). In deep grasp detection, most authors used the SGD but argued that it was not an optimal optimizer and reported that more advanced optimizers were necessary. Ruder provided a comprehensive overview of different learning optimizers in [60].

While most transfer learning approaches employed pre-trained network models in an end-to-end learning process some researchers used them as feature extractors for shallow network models. Chollet [61] stated that due to the two-step method of feature extraction, it was impossible to employ data augmentation techniques if necessary, as the learned features would not be the same as the training images. In addition, running end-to-end learning was costlier as it required the convolutional base of the network to be run on data repetitively.

Detecting object grasping configurations from images is still accurately solved using analytical methods but the use of empirical methods is exponentially increasing due to successful results in recent publications. One commonly used method is to train a visuomotor controller using deep learning that iteratively corrects the grasping point until the object is successfully grasped between the gripper jaws. The next best method is to learn a function that scores the possible grasps on an image and use it to select the highest scored grasp as the candidate. There are other methods that learn a certain heuristic and exhaustively search for possible grasps on the images. Training CNNs to detect grasps requires a high volume of manually labelled data. As a solution, most researchers opt to use simulated training data (e.g., [41, 25, 44]). Alternatively, data collection can be automated (e.g., [11]). Recent approaches have suggested network pre-training can help to avoid overfitting due to limited availability of training data [10, 9].

3.4 Deep Reinforcement Learning for Manipulation

In this section, the recent progress of Deep Reinforcement Learning (DRL) in the domain of robotic manipulation control is discussed.

3.4.1 Reinforcement Learning

Reinforcement learning [62] is a subfield of machine learning, concerned with how to find an optimal behavior strategy to maximize the outcome though trial and error dynamically and autonomously, which is quite similar with the intelligence of human and animals, as the general definition of intelligence is the ability to perceive or infer information, and to retain it as knowledge to be applied towards adaptive behaviors in the environment. This autonomous self-teaching methodology is actively studied in many 5 of 12 domains, like game theory, control theory, operations research, information theory, system optimization, recommendation system and statistics [63].

Reinforcement learning is different from supervised learning, where a training set of labeled examples is available. In interactive problems like robot control domain using reinforcement learning, it is often impractical to obtain examples of desired behavior that are both correct and representative of all the situations in which the agent has to act. Instead of labels, we get rewards which in general are weaker signals. Reinforcement learning is not a kind of unsupervised learning, which is typically about finding structure hidden in collections of unlabeled data. In reinforcement learning, the agent has to learn to behave in the environment based only on those sparse and time-delayed rewards, instead of trying to find hidden structure. Therefore, reinforcement learning can be considered as a third machine learning paradigm, alongside supervised learning and unsupervised learning and perhaps other future paradigms as well [64].

3.4.2 Deep Reinforcement Learning

DRL emerges from reinforcement learning and deep learning, and can be regarded as the bridge between conventional machine learning and true artificial intelligence, as illustrated in Figure 4. It combines both the technique of giving rewards based on actions from reinforcement learning, and the idea of using a neural network for learning feature representations from deep learning. Traditional reinforcement learning is limited to domains with simple state representations, while

DRL makes it possible for agents to make decisions from high-dimensional and unstructured input data [65] using neural networks to represent policies. In the past few years, research in DRL has been highly active with a significant amount of progress, along with the rising interest in deep learning.

The most commonly used DRL algorithms can be categorized in value-based methods, policy gradient methods and model-based methods. The value-based methods construct a value function for defining a policy, which is based on the Q-learning algorithm [66] using the Bellman equation [67] and its variant, the fitted Q-learning [68, 69]. The Deep Q-Network (DQN) algorithm used with great success in [43] is the representative of this class, followed by various extensions, such as double DQN [70], Distributional DQN [71, 72], etc. A combination of these improvements has been studied in [73] with state-of-the-art performance on the Atari 2600 benchmark, both in terms of data efficiency and final performance.

However, the DQN-based approaches are limited to problems with discrete and low dimensional action spaces, and deterministic policies, while policy gradient methods are able to work with continuous action spaces and can also represent stochastic policies. Thanks to variants of stochastic gradient ascent with respect to the policy parameters, policy gradient methods are developed to find a neural network parameterized policy to maximize the expected cumulative reward. Like other policy-based methods, policy gradient methods typically require an estimate of a value function for the current policy, and a sample-efficient approach is to use an actor-critic architecture that can work with off-policy data. The Deep Deterministic Policy Gradient (DDPG) algorithm [74] is a representation of this type of methods.

Both value-based and policy-based methods do not make use of any model of the environment and are also called model-free methods, which limits their sample efficiency. On the contrary, in the model-based methods, a model of the environment is either explicitly given or learned from experience by the function approximators [75, 76] in conjunction with a planning algorithm. In order to obtain advantages from both sides, there are many researches available integrating model-free and model-based elements [77, 78, 79], which are among the key areas for the future development of DRL algorithms.

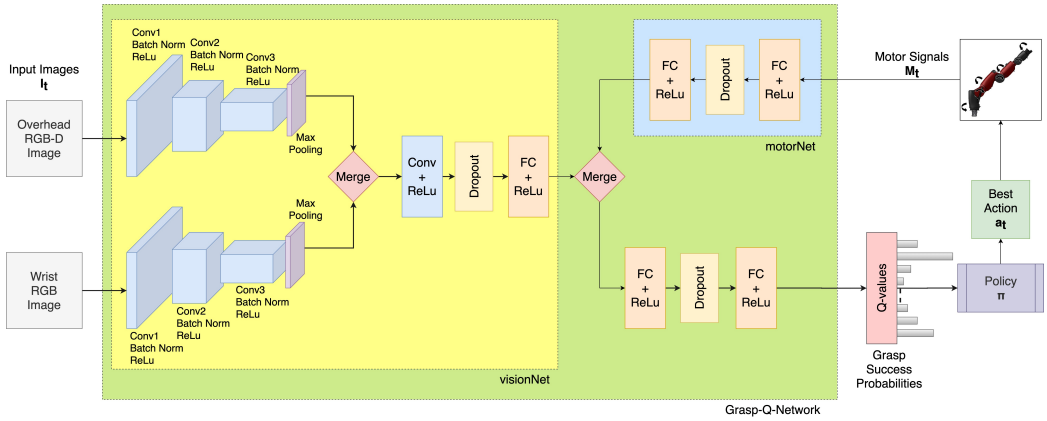


Figure 3.6: The architecture of Grasp-Q-Network proposed by Joshi *et al.* [4].

3.4.3 Deep Reinforcement Learning in Robotic Manipulation

In reinforcement learning, the learner is not informed which action to take but instead, it should decide which action will yield the most reward by trial and error. In most cases, the actions not only affect the subsequent reward but the next action thereby affecting all the subsequent rewards [64]. Thus, trial and error search and delayed reward are the two most prominent features of reinforcement learning. Several others claim to use rules for grasping that is based on the research that portrays human actions for grasping and manipulating objects. [80] presents early work in using reinforcement learning for robotic grasps in which the learning approach is adopted from human grasping an object. Three layers of functional modules that enable learning from a finite set of data along with maintaining a good generalization [81, 82] have shown successful implementations of reinforcement learning in the past.

However, due to complex issues such as memory complexity, sample complexity, as well as computational complexity, the user has to rely on deep learning networks. These networks use function approximations and representation learning properties to overcome the problems of using algorithms that require very high computational power and fast processing. [83] discusses the use of a reinforcement learning algorithm, Policy Improvement with Path Integrals (PI^2) that can be used when the state estimation is uncertain and this approach does not require a specific model and is therefore model-free. [84] discusses how a system can learn to reliably perform its task in a short amount of time by implementing a reinforcement learning strategy with a minimum amount of information provided for a given task of picking an object. Deep learning has enabled reinforcement

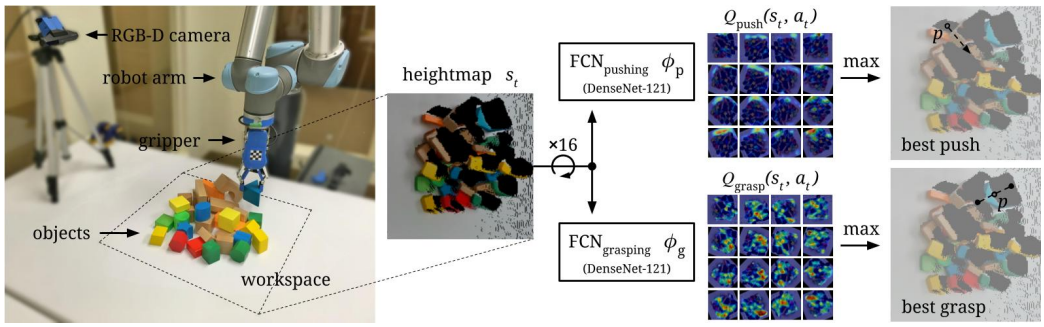


Figure 3.7: Q-learning framework proposed by Zheng *et al.* in [12].

learning to be used for decision-making problems such as settings with large dimensional state and action spaces that were once unmanageable. In recent years, a number of off-policy reinforcement learning techniques have been implemented. For instance, [85] uses deep reinforcement learning for solving Atari games, [86] uses a model-free algorithm based on the deterministic policy gradient to solve problems in the continuous action domain. Joshi *et al.* proposed a double deep Q-learning framework along with a Grasp-Q-Network (show in Fig. 3.6) to output grasp probabilities used to learn grasps that maximize the pick success. The observed input taken from the overhead camera and the wrist camera along with the current motor positions are fed into the Grasp-Q-Network, which returns the grasp success probabilities i.e. the Q values for all possible actions that the agent can take. These Q values are then used to select the best action based on the ϵ-greedy policy to find the optimal policy π^*.

The current research on DQN shows how deep reinforcement learning can be applied for designing closed-loop grasping strategies. [87] demonstrates this by proposing a Q-function optimization technique to provide a scalable approach for vision-based robotic manipulation applications. Zheng *et al.* used deep Q-learning framework (show in Fig. 3.7) for learning grasping strategies, along with pushing for applications comprising tightly packed spaces and cluttered environments [12].

3.5 Intermediate Conclusions

Deep learning has been showing remarkable success in the applications of computer vision such as classification, detection and localisation. Thus far, however, it has not been adopted very extensively

in robotic applications, although this is now a rapidly growing area of research. Due to the requirements of higher computational power and large volumes of training data, it is still challenging to implement an end-to-end learning approach for the complete robotic grasping activity. Despite this, the detection of successful grasping poses for robotic systems using deep learning methods has been investigated in a number of recent publications. In this paper, these emerging methods have been reviewed and several key elements of deep learning based grasp detection have been identified as: grasp representation, grasp detection, training, and CNN architectures.

Several methods have been discussed to represent successful grasps of a robotic system. A successful grasp is identified as the location within the work area that a robotic end-effector can be placed to securely grasp the target object between its parallel plate grippers and lift it without losing its grip. The information such as the coordinates of this location, the width the gripper needs to be opened, and the gripper orientation with reference to the horizontal axis are commonly included in this grasp representation.

Training a suitable neural network for grasp detection purposes usually requires a large amount of manually labeled data, such as the Cornell Grasps Dataset, but this requires even higher volumes of domain specific data which is not readily available at the time of writing. Therefore, researchers have opted to collect data with self-supervised methods. While most of these applications rely on realistic data such as images from objects in the real world, more recent literature discusses the use of simulation data and their domain adaptation capability. The one-shot detection algorithm follows the transfer learning technique to employ a pre-trained DCNN as its convolutional base.

Chapter 4

Robotic Grasp Detection using Deep Convolutional Neural Networks

Robotic grasping lags far behind human performance and is an unsolved problem in the field of robotics. When humans see novel objects, they instinctively know how to grasp to pick them up. A lot of work has been done related to robotic grasping and manipulation [21, 8, 88, 10, 9], but the problem of real-time grasp detection and planning is still a challenge. Even the current state-of-the-art grasp detection techniques fail to detect a potential grasp in real-time. The robotic grasping problem can be divided into three sequential phases: grasp detection, trajectory planning, and execution. Grasp detection is a visual recognition problem in which the robot uses its sensors to detect graspable objects in its environment. The sensors used to perceive the robot's environment are typically 3-D vision systems or RGB-D cameras. The key task is to predict potential grasps from sensor information and map the pixel values to real-world coordinates. This is a critical step in performing a grasp as the subsequent steps are dependent on the coordinates calculated in this step. The calculated real-world coordinates are then transformed to position and orientation for the robot's End-of-arm Tooling (EOAT). An optimal trajectory for the robotic arm is then planned to reach the target grasp position. Subsequently, the planned trajectory for the robotic arm is executed using either an open-loop or a closed-loop controller. In contrast to an open-loop controller, a closed-loop controller receives continuous feedback from the vision system during the entire grasping task. The additional processing needed to handle the feedback is computationally expensive and

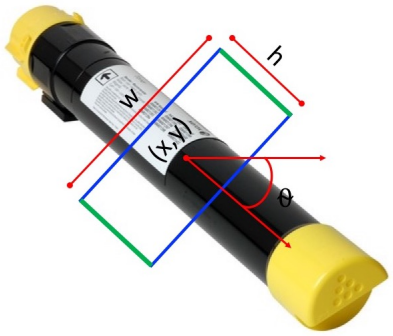


Figure 4.1: An example grasp rectangle for a potential good grasp of a toner cartridge. This is a five-dimensional grasp representation, where green lines represent parallel plates of the gripper, blue lines correspond to the distance between parallel plates of the grippers before grasp is performed, (x,y) are the coordinates corresponding to the center of the grasp rectangle, and θ is the orientation of the grasp rectangle with respect to the horizontal axis.

can drastically affect the speed of the task.

In this chapter, the problem of detecting a 'good grasp' from RGB-D imagery of a scene is addressed. Fig. 4.1 shows a five-dimensional grasp representation for a potential good grasp of a toner cartridge. This five-dimensional representation gives the position and orientation of a parallel plate gripper before the grasp is executed on an object. Although it is a simplification of the seven-dimensional grasp representation introduced by Jiang *et al.* [8], Lenz *et al.* showed that a good five-dimensional grasp representation can be projected back to a seven-dimensional grasp representation that can be used by a robot to perform a grasp [10]. In addition to the low computational cost, this reduction in dimension allows us to detect grasps using RGB-D images. In this work, this five-dimensional grasp representation is used for predicting the grasp pose.

A novel approach for detecting good robotic grasp for parallel plate grippers using the five-dimensional representation is introduced. The proposed approach uses two 50-layer deep convolutional residual neural networks running in parallel to extract features from RGB-D images, with one network analyzing the RGB component and the other analyzing the depth channel. The outputs of these networks are then merged, and fed into another convolutional network that predicts the grasp configuration. The proposed approach is compared to others in the literature, as well as a uni-modal variation of the proposed model that uses only the RGB component. The experiments are carried out on the standard CGD. Example images from the dataset are shown in Fig. 4.2. The experiments

Figure 4.2: Sample images from the CGD.

show that the proposed architecture outperforms the current state-of-the-art methods in terms of both accuracy and speed.

4.1 Background

Deep learning [89] has made significant progress in multiple problems in computer vision [53, 51, 90] and natural language processing [91, 92, 93]. These results have inspired many robotics researchers to explore the applications of deep learning to solve some of the challenging problems in robotics. For example, robot localization is moving from using hand-engineered features [94] to deep learning features [95], deep reinforcement learning is being used for end-to-end training for robotic arm control [96], multi-view object recognition has achieved state-of-the-art performance by deep learning camera control [97], reinforcement learning has been used to learn dual-arm manipulation tasks [98], and autonomous driving has been tackled using deep learning to estimate the affordances for driving [99].

A major challenge with deep learning is that it needs a very large volume of training data. However, large datasets with manually labeled images are unavailable for most robotics applications.

In computer vision, transfer learning techniques are used to pre-train deep convolutional neural networks on some large dataset, e.g., ImageNet, which contains 1.2 million images with 1000 categories [100], before the network is trained on the target dataset [101]. These pre-trained models are either used as an initialization or as a fixed feature extractor for the task of interest.

The most common approach for 2-D robotic grasp prediction is a sliding window detection framework. In this framework, a classifier is used to predict whether a small patch of the input image has a good potential grasp for an object. The classifier is applied to a number of patches on the image and the patches that get high scores are considered as good potential grasps. Lenz *et al.* used this technique with convolutional neural networks as a classifier and got an accuracy of 75% [10]. A major drawback of their work is that it runs at a speed of 13.5 seconds per frame, which is extremely slow for a robot to find where to move its EOAT in real-time applications. In [9], this method was accelerated by passing the entire image through the network at once, rather than passing several patches.

A significant amount of work has been done using 3-D simulations to find good grasps [102, 103, 104, 105]. These techniques are powerful, but most of them rely on a known 3-D model of the target object to calculate an appropriate grasp. However, general purpose robots should be able to grasp unfamiliar objects without object's 3-D model. Jincheng *et al.* showed that deep learning has the potential for 3-D object recognition and pose estimation, but their experiments only used five objects and their algorithm is computationally expensive [106]. Recent work by Mahler *et al.* uses a cloud-based robotics approach to significantly reduce the number of samples required for robust grasp planning [107]. Johns *et al.* generated their training data by using a physics simulation and depth image simulation with 3-D object meshes to learn grasp score which is more robust to gripper pose uncertainty [41].

Grasp point detection technique proposed by Jeremy *et al.* [108] has very high precision of 92%, but it only works with cloth towels and cannot be used as a general purpose grasp detection technique. Another grasp pose detection technique was introduced by Gualtieri *et al.* [109] for removing objects from a dense cluster. The technique was evaluated only on a small set of objects using a research robot.

The proposed method takes a different approach, instead of using AlexNet for feature extraction, as used in [10], [9] and [110], the current state-of-the-art deep convolutional neural network known

as ResNet [55] is used. A multi-modal model is also introduced, which extracts features from both RGB and Depth images to predict the grasp configuration.

4.2 Problem Formulation

The robotic grasp detection problem can be formulated as finding a successful grasp configuration g for a given image I of an object. A five-dimensional grasp configuration g is represented as:

$$g = f(x, y, h, w, \theta) \tag{4.1}$$

where (x, y) corresponds to the center of grasp rectangle, h is the height of the parallel plates, w is the maximum distance between the parallel plates, and θ is the orientation of the grasp rectangle with respect to the horizontal axis. h and w are usually fixed for a specific robot EOAT. An example of this representation is shown in Fig. 4.1.

This work focuses on planer grasps as Lenz *et al.* showed that a five-dimensional grasp configuration can be projected back to a seven-dimensional configuration for execution on a real robot. To solve this grasp detection problem, a different approach is taken, explained in section 4.3.

4.3 Approach

DCNNs have outperformed the previous state-of-the-art techniques to solve detection and classification problems in computer vision. In this work, DCNN is used to detect the target object from the image and predict a good grasp configuration. A single-step prediction technique is proposed instead of the two step approach used in [10] and [9]. These methods ran a simple classifier many times on small patches of the input image, but this is slow and computationally expensive. Instead, the entire image directly fed into DCNN to make grasp prediction on complete RGB-D image of the object. This solution is simpler and has less overhead.

Theoretically, a DCNN should have better performance with increased depth because it provides increased representational capacity. However, the currently used optimization method, Stochastic Gradient Decent (SGD) is not an ideal optimizer. In experiments, researchers found that increased depth brought increased training error, which is not in-line with the theory [55]. The increased

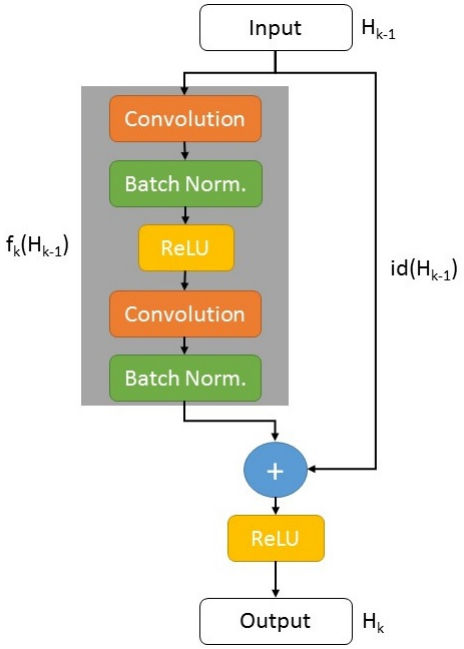


Figure 4.3: Example residual block in ResNet.

training error indicates that the ultra-deep network is very hard to optimize. This means that identity map is very hard to obtain in a convolutional neural network by end-to-end training using SGD. Therefore, residual layers as in ResNet [55] are used, which reformulates the mapping function between layers, using the function given by Eq.(4.2).

Similar to previous works, this work assumes that the input image contains only one graspable object and a single grasp has to be predicted for the object. The advantage of this assumption is that we can look at the complete image and make a global grasp prediction. This assumption may not be possible outside the experimental conditions and we would have to come up with a model that has to first divide the image into regions, so each region contains only one object.

4.3.1 Architecture

The proposed model is much deeper as compared to the previous approaches (e.g., [10, 110, 9]). Instead of using an eight-layer AlexNet, the ResNet-50, a fifty-layer deep residual model, is used to solve this grasp detection problem. The ResNet architecture uses the simple concept of residual learning to overcome the challenge of learning an identity mapping. A standard feed-forward CNN is modified to incorporate skip connections that bypass a few layers at a time. Each of these skip

connections gives rise to a residual block, and the convolution layers predict a residual that is added to the block's input. The key idea is to bypass the convolution layers and the non-linear activation layers in k^{th} residual block, and let through only the identity of the input feature in the skip connection. Fig. 4.3 shows an example of a residual block with skip connections. The residual block is defined as:

$$H_k = F(H_{k-1}, W_k) + H_{k-1} \tag{4.2}$$

where, H_{k-1} is the input to the residual block, H_k is the output of the block, and W_k are the weights learned for the mapping of function F. The readers are encouraged to see [55] for more details on the ResNet architecture.

Two different architectures are introduced for robotic grasp prediction: uni-modal grasp predictor and multi-modal grasp predictor. The uni-modal grasp predictor is a 2D grasp predictor that uses only single modality (e.g., RGB) information from the input image to predict the grasp configuration, where as the multi-modal grasp predictor is a 3-D Grasp Predictor that uses multi-modal (e.g., RGB and Depth) information. In the next two subsections, these two architectures are discussed in detail.

4.3.2 Uni-modal Grasp Predictor

Large-scale image classification datasets have only RGB images. Therefore, the DCNNs can be pre-trained with 3-channels only. This work introduces a uni-modal grasp predictor model which is designed to detect grasp using only three channels (RGB or Red Green Depth (RGD)) of the raw image. Fig. 4.4 shows the complete architecture of the proposed uni-modal grasp predictor. A ResNet-50 model that is pre-trained on ImageNet is used to extract features from the RGB channels of the image. For a baseline model, a linear SVM is used as classifier to predict the grasp configuration for the object using the features extracted from the last hidden layer of ResNet-50. In the proposed uni-modal grasp predictor, the last fully connected layer of ResNet-50 is replaced by two fully connected layers with rectified linear unit (ReLU) as activation functions. A dropout layer is also added after the first fully connected layer to reduce overfitting. SGD is used to optimize the training loss and Mean Squared Error (MSE) as is used as the loss function.

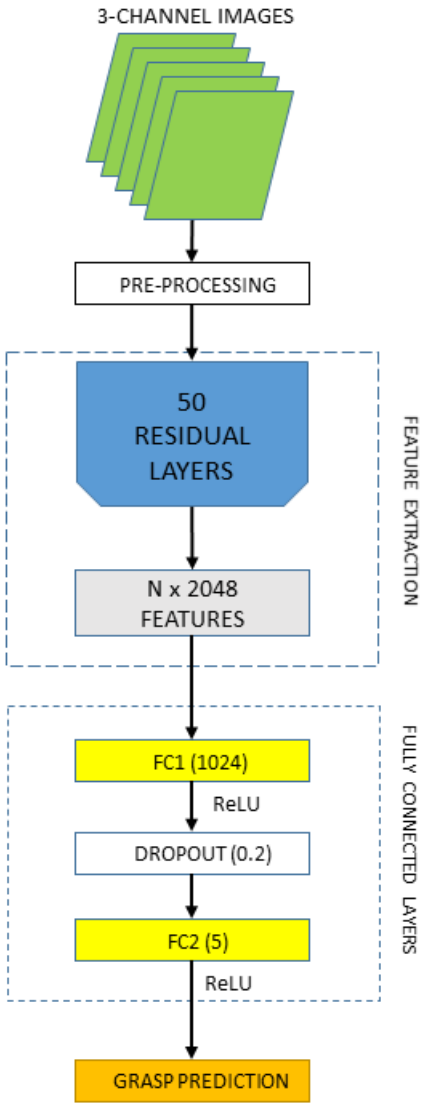


Figure 4.4: Complete architecture of the proposed uni-modal grasp predictor.

The 3-channel image is fed to the uni-modal grasp predictor, which uses the residual convolutional layers to extract features from the input image. The last fully connected layer is the output layer, which predicts the grasp configuration for the object in the image. During the training time, weights of convolutional layers in ResNet-50 are kept fixed and only the weights of last two fully connected layers are tuned. The weights of the last two layers are initialized using Xavier weight initialization.

4.3.3 Multi-modal Grasp Predictor

A multi-modal grasp predictor is also introduced, which is inspired by the RGB-D object recognition approach introduced by Schwarz *et al.* [111]. The multi-modal grasp predictor uses multi-modal (RGB-D) information from the raw images to predict the grasp configuration. The raw RGB-D images are converted into two images. The first is a simple RGB image, and the other is a depth image converted into a 3-channel image. This depth to 3-channel conversion is done similar to a gray to RGB conversion. These two 3-channel images are then given as input to two independent pre-trained ResNet-50 models. The ResNet-50 layers work as feature extractors for both images. Similar to the uni-modal grasp predictor, features are extracted from the second last layer of both the ResNet-50 networks. The extracted features are then normalized using L2-normalization. The normalized features are concatenated together and feed into a shallow convolutional neural network with three fully connected layers. The fully connected layers use ReLU activation functions. A dropout layer is added after first and second fully connected layers of the shallow network to reduce over-fitting. Similar to the uni-modal model, SGD is used as the optimizer and MSE as the loss function. Fig. 4.5 shows the complete architecture of the proposed multi-modal grasp predictor.

By using two DCNNs in parallel, the model was able to extract features from both RGB and depth images. Therefore, enabling the model to learn multimodal features from the RGB-D dataset. Weights of the two DCNNs are initialized using the pre-trained ResNet-50 models and the weights of the shallow network are initialized using Xavier weight initialization. The weights are fine-tuned during training.

As a simple baseline, a linear SVM classifier is also applied to the L2-normalized RGB DCNN and depth DCNN features to predict the grasp configuration for the object in the image.

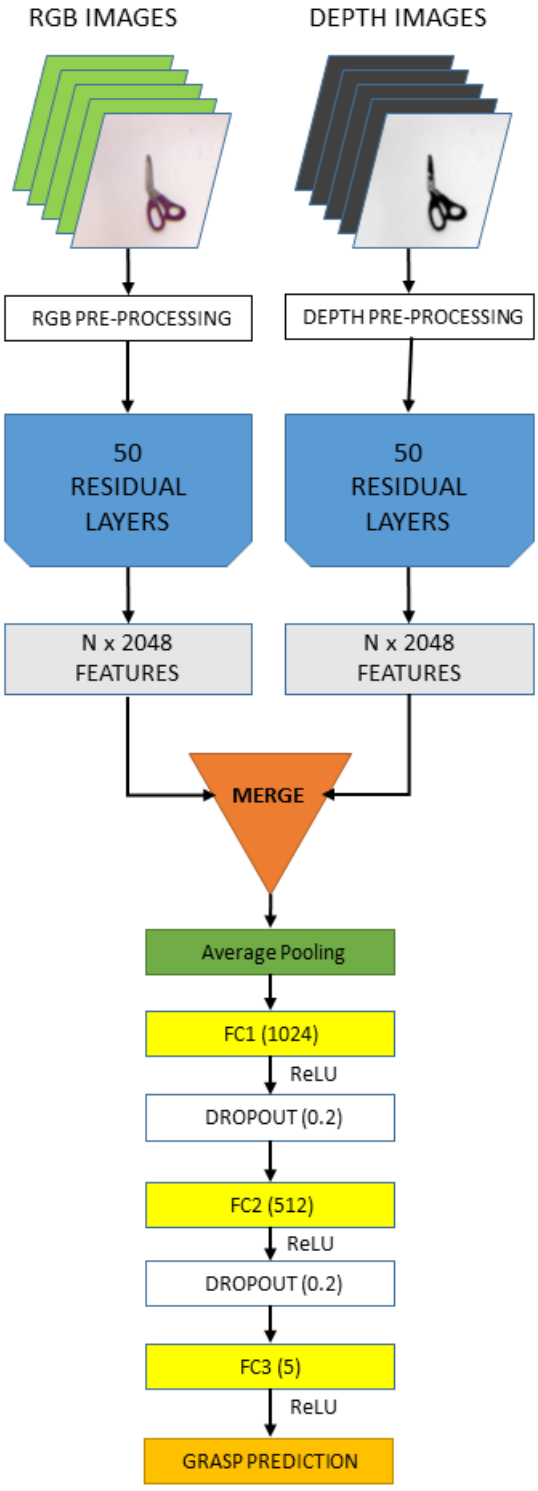


Figure 4.5: Complete architecture of the proposed multi-modal grasp predictor.

Figure 4.6: Ground truth grasps using the rectangular metric for a subset of the CGD.

4.4 Experiments

4.4.1 Dataset

For comparing the proposed method with others, the architecture is trained and tested on the standard CGD. The dataset is available at http://pr.cs.cornell.edu/grasping/rect_data/data.php. This dataset consists of 885 images of 240 different objects. Each image has multiple grasp rectangles labeled as successful (positive) or failed (negative), specifically selected for parallel plate grippers. In total, there are 8019 labeled grasps with 5110 positive and 2909 negative grasps. Fig. 4.6 shows the ground truth grasps using the rectangular metric for this dataset.

Similar to previous works, a five-fold cross validation is used for all experiments. The dataset is split in two different ways:

1. **Image-wise split**

   Image-wise splitting splits all the images in the dataset randomly into the five folds. This is helpful to test how well did the network generalize to the objects it has seen before in a

different position and orientation.

2. **Object-wise split**

   Object-wise splitting splits all the object instances randomly and all images of an object are put in one validation set. This is helpful to test how well did the network generalize to objects it has not seen before.

4.4.2 Data Pre-processing

Minimal amount of preprocessing of data is performed before feeding it into the DCNN. The input to the proposed DCNN is a patch around the grasp point, extracted from a training image. The patch is resized to 224×224, which is the input image size of the ResNet-50 model. The depth image is re-scaled to range 0 to 255. There are some pixels in depth image that have a NaN value as they were occluded in the original stereo image. These pixels with NaN value were replaced by zeros.

4.4.3 Pre-training

Pre-training is necessary when the domain-specific data available is limited as in the CGD. Therefore, ResNet-50 is first trained on ImageNet. It is assumed that most of the filters learned are not specific to the ImageNet dataset and only the layers near the top exhibit specificity for classifying 1,000 categories. The DCNN will learn universal visual features by learning millions of parameters during this pre-training process. Then, the features from the last layer are feed to the shallow convolutional neural network. It is important to note that the ImageNet dataset has only RGB images and thus the DCNN will learn RGB features only.

4.4.4 Training

For training and validation of the proposed models Keras deep learning library is used, which is written in Python and runs on top of Theano. Experiments were performed on a CUDA enabled NVIDIA GeForce GTX 645 GPU with Intel(R) Core(TM) i7-4770 CPU @ 3.40GHz. Although GPUs are currently not an integral part of robotic systems, they are becoming popular in vision-based robotic systems because of the increased computational power.

The training process was divided into two stages, in the first stage, only the shallow network is

Table 4.1: Grasp Prediction Accuracy on the CGD

Authors	**Algorithm**	**Accuracy (%)**	
		Image-wise split	Object-wise split
	Chance	6.7	6.7
Jiang *et al.* [8]	Fast Search	60.5	58.3
Lenz *et al.* [10]	SAE, struct. reg. two-stage	73.9	75.6
Redmon *et al.* [9]	AlexNet, MultiGrasp	88.0	87.1
Wang *et al.* [34]	Two-stage closed-loop, with penalty	85.3	-
Asif *et al.* [112]	STEM-CaRFs (Selective Grasp)	88.2	87.5
	Uni-modal Grasp Predictor		
	ResNet-50, SVM - RGB (*Baseline*)	84.76	84.47
	ResNet-50, ReLU, ReLU - RGB	88.84	87.72
Proposed	ResNet-50, tanh, ReLU - RGD	88.53	88.40
	Multi-Modal Grasp Predictor		
	ResNet-50x2, linear SVM - RGB-D	86.44	84.47
	ResNet-50x2, ReLU - RGB-D	**89.21**	**88.96**

trained, and in the second stage the complete network is trained end-to-end. To train the proposed uni-modal grasp predictor, SGD is used to optimize the model with hyper parameters in first stage set as: learning rate = 0.001, decay = 1e-6, momentum = 0.9, mini-batch size = 32 and maximum number of epoches = 30. For the multi-modal grasp predictor, the following hyper parameters are used in the first stage: learning rate = 0.0006, decay = 1e-6, momentum = 0.9, mini-batch size = 32 and maximum number of epoch = 50. For fine-tuning the network in the second phase, a much lower learning rate is used, and the learning rate is plateaued if the training loss does not decreases.

4.4.5 Evaluation

Prior works have used two different performance metrics for evaluating grasps on the CGD: rectangle metric and point metric. The point grasp metric compares the distance between the center point of predicted grasp and the center point of all the ground truth grasps. A threshold is used to consider the success of grasp, but past work did not disclose these threshold values. Furthermore, this metric does not consider the grasp angle, which is an essential parameter for grasping. The rectangle grasp metric consider complete grasp rectangle for evaluation and a grasp is considered to be a good grasp if the difference between the predicted grasp angle and the ground truth grasp angle is less than 30°,

and the Jaccard similarity coefficient of the predicted grasp and ground truth grasp is greater than 25%. Jaccard similarity coefficient or the Jaccard index measures similarity between the predicted grasp and ground truth grasp, and is defined as:

$$J(\widehat{\theta}, \theta) = \frac{|\widehat{\theta} \cap \theta|}{|\widehat{\theta} \cup \theta|} \tag{4.3}$$

where $\widehat{\theta}$ is the predicted grasp and θ is the ground truth grasp. As the rectangle metric is better at discriminating between 'good' and 'bad' grasp, this metric is used for the experiments. For all proposed models, the best scored grasp rectangle is selected using the rectangle metric for predicting the grasp.

4.5 Results

Table 4.1 shows a comparison of the results with the previous work for the rectangle metric grasp detection accuracy on the CGD. Across the board, both proposed models outperform the current state-of-the-art robotic grasp detection algorithms in terms of accuracy and speed. Results for the previous work are their self-reported scores. Tests were performed with image-wise split and object-wise split to test how well the network can generalize to different grasp features.

The results of various versions of uni-modal and multi-modal grasp predictors are collected by changing the information fed to the input channels. The RGB version of uni-modal grasp predictor uses only RGB channels of the input image. In the RGD version, the blue channel of the input image is replaced with the rescaled depth information. The baseline model of uni-modal grasp predictor got an accuracy of 84.76%. The uni-modal grasp predictor with RGB data got an accuracy of 88.84% and the same model with RGD data achieved an accuracy of 88.53%. In contrast to prior work, replacing blue channel with depth did not help the proposed model. This is mainly due to the fact that ResNet was trained with RGB images and the features learned in RGB are not the same as the features extracted from RGD.

The baseline multi-modal grasp predictor used RGB-D data and got an accuracy of 86.44%, which sets a new baseline for performance in RGB-D robotic grasp detection. The proposed multi-modal grasp predictor achieved an accuracy of 89.21%, which is the new state-of-the-art performance for RGB-D robotic grasp detection. Another experiment was performed by replacing the

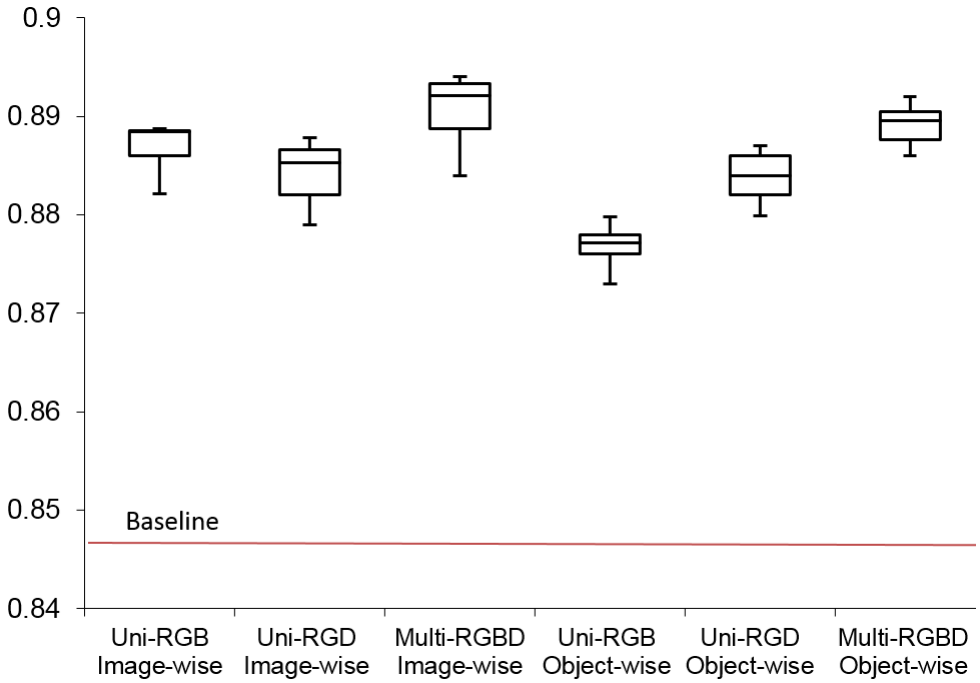


Figure 4.7: Accuracy comparison of models.

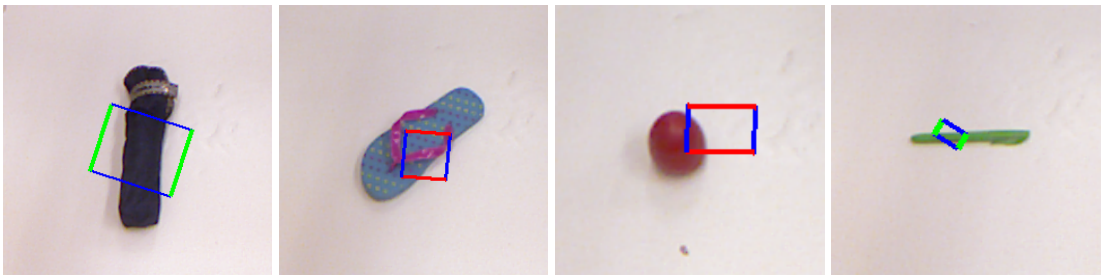
ResNet-50 model with a pre-trained VGG16 model. Although it performed better than previous models, it did not perform better than the proposed multi-modal model. Fig. 4.7 shows an accuracy comparison of all the proposed models in this work using 5-fold cross validation. Overall, the proposed multi-modal grasp predictor performed the best with the CGD.

Table 4.2 shows the grasp prediction speeds for the proposed models and compares it with previous work. Both proposed models are faster than previous methods. The uni-modal grasp predictor runs 800 times faster than the two-stage SAE model by Lenz *et al.* . The main reason for this boost in speed is replacing the sliding window classifier based approach by a single pass model. Also, GPU hardware is used to accelerate computation, and that can be another reason for faster computation.

Furthermore, a modification was made to the multi-modal model to predict the graspability, i.e. whether an object is graspable for a specific grasp rectangle or not. This was done by replacing the last fully connected layer by a dense layer with a binary output and using softmax as the activation function. It was able to achieve an accuracy of 93.4%, which is at par with the current state-of-the-art. Fig. 4.8 shows some examples of predicted graspability using the modified multi-modal

Table 4.2: Grasp Prediction Speed

Method	**Speed (fps)**
Lenz *et al.* [10]	0.02
Redmon *et al.* [9]	3.31
Wang *et al.* [34]	7.10
Uni-modal Grasp Predictor	16.03
Multi-modal Grasp Predictor	9.71



(a) True Positive (b) False Positive (c) True Negative (d) False Negative

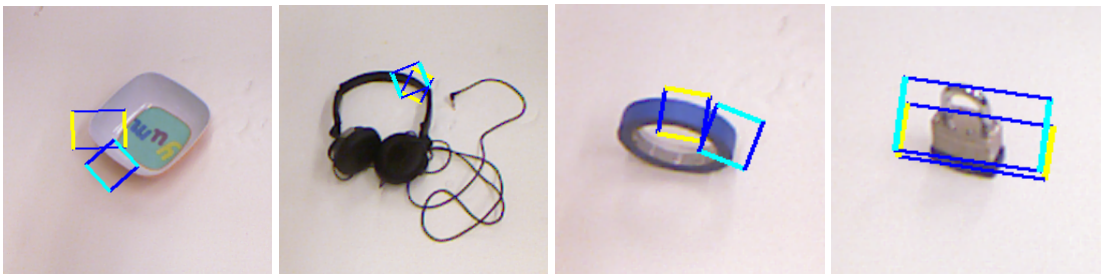
Figure 4.8: Examples of predicted graspability using the modified multi-modal grasp predictor.

grasp predictor. A green box means that a successful grasp was predicted and a red box means an unsuccessful grasp was predicted. The false negative (Fig. 4.8b) and false negative (Fig. 4.8d) are incorrect predictions. In Fig. 4.8b the model failed to understand the depth features of the slipper strap, using which the grippers can grasp the slipper. Whereas, in Fig. 4.8d the model failed to understand the orientation θ of the grasp rectangle with respect to the object. Other than some tricky examples such as these, the model predicts the graspability of different types of objects with high accuracy.

4.6 Discussion

The experimental results show that DCNN can be used to predict the graspability and grasp configuration for an object. The proposed network is 6 times deeper as compared to the previous work by Lenz *et al.* and it improved the accuracy by 14.94% for image-wise split and 13.36% for object-wise split. This shows that going deeper with network and using skip connections helps the model learn more features from the grasp dataset.

The results show that high accuracy can be achieved with the proposed multi-modal model and



(a) Bowl (b) Headphones (c) Tape (d) Lock

Figure 4.9: Uni-modal VS multi-modal grasp predictor.

that it can be used to predict the graspability and grasp configuration for the objects that the model has not seen before. The uni-modal model got the best accuracy when used with RGB data and image-wise split of dataset. Whereas, the multi-modal model performed the best with RGB-D data and object-wise split of the dataset.

Fig. 4.9 shows cases when multi-modal grasp predictor (cyan and blue grasp rectangles) produces a viable grasp, while the uni-modal grasp predictor (yellow and blue grasp rectangles) fails to produce a viable grasp for the same object. In some of these cases, the grasp produced by the uni-modal predictor might be feasible for a robotic gripper, but the grasp rectangle produced by the multi-modal predictor represents a grasp which would clearly be successful.

4.7 Limitations

Due to unavailability of a pre-trained ResNet model for depth data, both the ResNet-50 models used in the multi-modal model were pre-trained on ImageNet. This may not be the best model for the depth image as the model is only trained on RGB images and will not have depth specific features. In the future, we would like to pre-train the model on a large-scale RGB-D grasp dataset like [49] and then use it to predict RGB-D grasps. Moreover, if we have a large-scale RGB-D grasp dataset, we can modify the uni-modal model to take a four-channel input and predict grasps using all four channels. In this case, the input size for the network will be $(224 \times 224 \times 4)$ and we can pass RGB as first three channels and depth as the fourth channel.

Chapter 5

Antipodal Robotic Grasping using Generative Residual Convolutional Neural Network

Robotic manipulators are constantly compared to humans due to the inherent characteristics of humans to instinctively grasp an unknown object rapidly and with ease based on their own experiences. As increasing research is being done to make robots more intelligent, there exists a demand for a generalized technique to infer fast and robust grasps for any kind of object that the robot encounters. The major challenge is being able to precisely transfer the knowledge that the robot learns to novel real-world objects.

In this chapter, a modular robot agnostic approach to tackle this problem of grasping unknown objects is presented. A GR-ConvNet is proposed that generates antipodal grasps for every pixel in an n-channel input image. The term generative is used to distinguish the proposed method from other techniques that output a grasp probability or classify grasp candidates in order to predict the best grasp. Several experiments and ablation studies are provided in both standard benchmarking datasets and real settings to evaluate the key components of the proposed system.

Fig.5.1 shows an overview of the proposed system architecture. It consists of two main modules: the inference module and the control module. The inference module acquires RGB and aligned depth images of the scene from the RGB-D camera. The images are pre-processed to match the

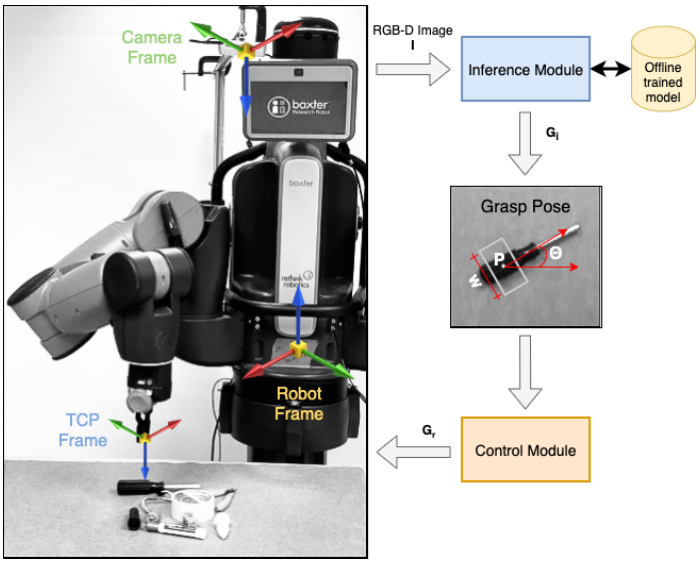


Figure 5.1: Overview. A real-time multi-grasp detection framework to predict, plan and perform robust antipodal grasps for the objects in the camera's field of view using an offline trained GR-ConvNet model. The proposed system can grasp novel objects in isolation as well as in clutter. Video: https://youtu.be/cwlEhdoxY4U

input format of the proposed GR-ConvNet model trained on an offline grasping dataset. The network generates quality, angle, and width images, which are then used to infer antipodal grasp poses. The control module consists of a task controller that prepares and executes a plan to perform a pick and place task using the grasp pose generated by the inference module. It communicates the required actions to the robot through a Robot Operating System (ROS) interface using a trajectory planner and controller.

In robotic grasping, it is very essential to generate grasps that are not just robust but also the ones that require the least amount of computation time. The proposed state-of-the-art technique demonstrates both of these from the outstanding results in generating robust grasps with lowest recorded inference time of 20ms on the CGD as well as the new G1BD. It is also demonstrated that the proposed technique works equally well in real-world with novel objects using a robotic manipulator. Fig. 5.2 shows the performance comparison of GR-ConvNet on CGD with prior work in terms of speed and accuracy.

Unlike the previous work done in robotic grasping [10, 113, 11, 5], where the required grasp is predicted as a grasp rectangle calculated by choosing the best grasp from multiple grasp probabilities, the proposed network generates three images from which we can infer grasp rectangles for

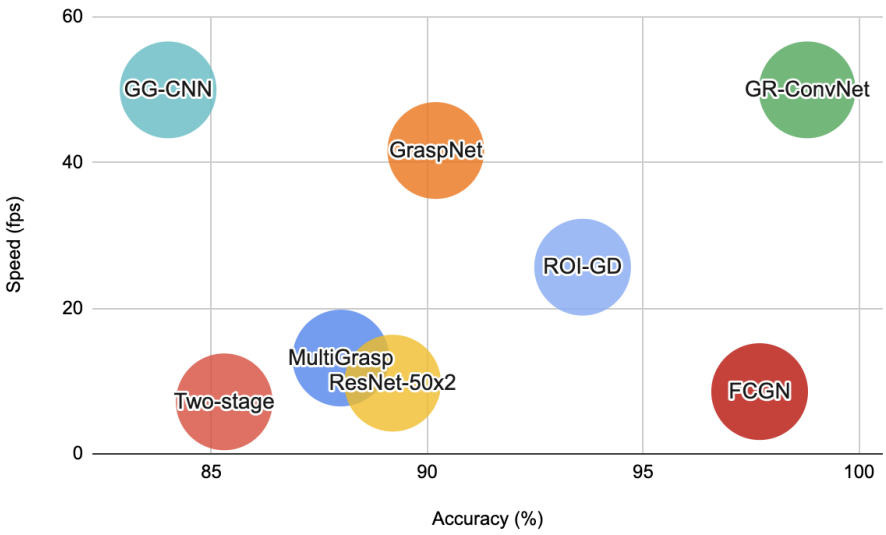


Figure 5.2: Performance comparison of GR-ConvNet on CGD with prior work.

multiple objects. Additionally, it is possible to infer multiple grasp rectangles for multiple objects from the output of GR-ConvNet in one-shot thereby decreasing the overall computational time.

5.1 Related Work

This work lies at the intersection of robotic grasping, computer vision, and deep learning. In this section, a brief review of the related work in these domains is presented. Table 5.1 provides a comparison of this work with recent related work in robotic grasping for unknown objects using learning-based approaches.

5.1.1 Robotic Grasping

There has been extensive on-going research in the field of robotics, especially robotic grasping. Although the problem seems to just be able to find a suitable grasp for an object, the actual task involves multifaceted elements such as- the object to be grasped, the shape of the object, physical properties of the object and the gripper with which it needs to be grasped among others. Early research in this field involved hand-engineering the features [108, 114], which can be a tedious and time-consuming task but can be helpful for learning to grasp objects with multiple fingers such as [115, 116].

Table 5.1: A comparison of related work

Author	Cornell Results	Jacquard Results	Graspnet Results	Household Objects	Adversarial Objects	Cluttered Objects	Open Source
Lenz *et al.* [10]	✓	✗	✗	✓	✗	✗	✓
Redmon *et al.* [113]	✓	✗	✗	✗	✗	✗	✗
Wang *et al.* [34]	✓	✗	✗	✗	✗	✗	✗
Kumra *et al.* [5]	✓	✗	✗	✗	✗	✗	✗
Depierre *et al.* [117]	✗	✓	✗	✓	✗	✗	✗
Chu *et al.* [37]	✓	✗	✗	✓	✗	✗	✓
Zhou *et al.* [38]	✓	✓	✗	✗	✗	✗	✗
Asif *et al.* [118]	✓	✗	✗	✓	✗	✓	✗
Morrison *et al.* [13]	✓	✗	✗	✓	✓	✓	✓
Zhang *et al.* [119]	✓	✓	✗	✓	✗	✗	✗
This work	✓	✓	✓	✓	✓	✓	✓

Initially for obtaining a stable grasp, the mechanics and contact kinematics of the end effector in contact with the object were studied and the grasp analysis was performed as seen from the survey by [32, 120]. Prior work [21] in robotic grasping for novel objects involved using supervised learning which was trained on synthetic data, but it was limited to environments such as offices, kitchen, and dishwasher. Satish *et al.* [121] introduced a fully convolutional gradient quality gradient neural network (FC-GQ-CNN) that predicted robust gradient quality using a data collection policy and a synthetic training environment. This method enabled an increase in the number of grasps considered to 5000 times in 0.625s. Bousmalis *et al.* [50] discussed domain adaptation and simulation in order to bridge the gap between simulated and real world data. In that pixel-level domain adaptation model, GraspGAN was used to generate adapted images that are similar to the real ones and are differentiated by the descriminator network. Trembley *et al.* [122] worked on a similar problem as Bousmalis *et al.* They used a deep network trained only on synthetic images with 6 DoF pose of known objects. However, this has been shown to work with household items only. James *et al.* [123] discuss about a Randomized to Canonical Adaptation Networks (RCANs) method that learns to translate images from randomized simulated environments to their equivalent simulated canonical images using an image-conditioned GAN. They then used this to train their RL algorithm for real-world images. Furthermore, an actor-critic network that combines the results obtained by the actor network is presented in [124] which samples grasp samples directly with the results obtained from a critic network which re-scores the results obtained from actor network to find stable and robust

grasps. However, the current research relies more on using the RGB-D data to predict grasp poses. These approaches depend wholly on deep learning techniques.

5.1.2 Deep Learning for Grasping

Deep learning has been a hot topic of research since the advent of ImageNet success and the use of GPU's and other fast computational techniques. Moreover, the availability of affordable RGB-D sensors enabled the use of deep learning techniques to learn the features of objects directly from image data. Recent experimentations using deep neural networks [113, 125, 126] have demonstrated that they can be used to efficiently compute stable grasps. Pinto *et al.* [11] used an architecture similar to AlexNet which shows that by increasing the size of the data, their CNN was able to generalize better to new data. Varley *et al.* [127] propose an interesting approach to grasp planning through shape completion where a 3D CNN was used to train the network on the 3D prototype of objects on their own dataset captured from various viewpoints. Guo *et al.* [36] used tactile data along with visual data to train a hybrid deep architecture. Mahler *et al.* [23] proposed a Grasp Quality Convolutional Neural Network (GQ-CNN) that predicts grasps from synthetic point cloud data trained with Dex-Net 2.0 grasp planner dataset. Levine *et al.* [128] discuss the use of monocular images for hand-to-eye coordination for robotic grasping using a deep learning framework. They use a CNN for grasp success prediction and further use continuous servoing to continuously servo the manipulator to correct mistakes. Antanas *et al.* [129] discuss an interesting approach known as a probabilistic logic framework that is said to improve the grasping capability of a robot with the help of semantic object parts. This framework combines high-level reasoning with low-level grasping. The high-level reasoning comprises object affordances, its categories, and task-based information while low-level reasoning uses visual shape features. This has been observed to work well in kitchen-related scenarios.

5.1.3 Grasping using Uni-modal Data

Johns *et al.* [41] used a simulated depth image to predict a grasp outcome for every grasp pose predicted and select the best grasp by smoothing the predicted pose using a grasp uncertainty function. A generative approach to grasping is discussed by Morrison *et al.* [13]. The Generative grasp CNN architecture generates grasp poses using a depth image and the network computes grasp on a

pixel-wise basis. [13] suggests that it reduces existing shortcomings of discrete sampling and computational complexity. Another recent approach that merely relies on depth data as the sole input to the deep CNN is as seen in [125].

5.1.4 Grasping using Multi-modal Data

There are different ways of handling objects multi-modalities. Many have used separate features to learn the modalities which can be computationally exhaustive. Wang *et al.* [34] proposed methods that consider multi-modal information as the same. Jiang *et al.* [8] used RGB-D images to infer grasps based on a two-step learning process. The first step was used to narrow down the search space and the second step was used to compute the optimal grasp rectangle from the top grasps obtained using the first method. Lenz *et al.* [10] used a similar two-step approach but with a deep learning architecture, which however could not work well on all types of objects and often predicted a grasp location that was not the best grasp for that particular object, such as in [8] the algorithm predicted grasp for a shoe was from its laces, which in practice failed when the robot tried to grasp using the shoelaces while in [10] the algorithm sometimes could not predict grasps which are more practical using just the local information as well as due to the RGB-D sensor used. Yan *et al.* [130] used point cloud prediction network to generate a grasp by first preprocessing the data by obtaining the color, depth, and masked images and then obtaining a 3D point cloud of the object to be fed into a critic network to predict a grasp. Chu *et al.* [37] propose a novel architecture that can predict multiple grasps for multiple objects simultaneously rather than for a single object. For this, they used a multi-object dataset of their own. The model was also tested on CGD. A robotic grasping method that consists of a ConvNet for object recognition and a grasping method for manipulating the objects is discussed by Ogas *et al.* [131]. The grasping method assumes an industry assembly line where the object parameters are assumed to be known in advance. Kumra *et al.* [5] proposed a Deep CNN architecture that uses residual layers for predicting robust grasps. The paper demonstrates that a deeper network along with residual layers learns better features and performs faster. Asif *et al.* [132] introduced a consolidated framework known as EnsembleNet in which the grasp generation network generates four grasp representations and EnsembleNet synthesizes these generated grasps to produce grasp scores from which the grasp with the highest score gets selected.

5.1.5 6-DoF Grasping

The 3-DOF grasp representation constrains the gripper pose to be parallel to the RGB image plane, which can be a challenge when grasping objects from a dense clutter. To overcome this, Liang *et al.* proposed PointNetGPD, which can directly process the 3D point cloud that locates within the gripper for grasp evaluation [133]. Similarly, Mousavian *et al.* introduced a 6-DOF GraspNet, which is a grasp evaluator network that maps a point cloud of the observed object and the robot gripper to a quality assessment of the 6D gripper pose. Moreover, they demonstrated that the gradient of GraspNet can be used to move the gripper out of collision and ensure that the gripper is well aligned with the object [134]. Murali *et al.* proposed a method that plans 6-DOF grasps for objects in a cluttered scene from partial point cloud observations. Their learned collision checking module was able to provide effective grasp sequences to retrieve objects that were not immediately accessible [135]. The two step deep geometry-aware grasping network (DGGN) proposed by Yan *et al.* first learns to build the mental geometry-aware representation by reconstructing the scene from RGB-D input, and then learns to predict the outcome of the grasp with its internal geometry-aware representation. The outcome of the model is used to sequentially propose grasping solutions via analysis-by-synthesis optimization [136]. A large-scale benchmark for object grasping called GraspNet-1Billion along with an end-to-end grasp pose prediction network to learn the approaching direction and operation parameters in a decoupled manner is introduced in [137].

5.2 Problem Formulation

In this work, the problem of robotic grasping is defined as predicting antipodal grasps for unknown objects from an n-channel image of the scene and executing it on a robot.

Instead of the five-dimensional grasp representation used in [10, 113, 5], an improved version of the grasp representation similar to the one proposed by Morrison *et al.* in [13] is used. The grasp pose in the robot frame is denoted as:

$$G_r = (\mathbf{P}, \Theta_r, W_r, Q) \tag{5.1}$$

where, $\mathbf{P} = (x, y, z)$ is tool tip's center position, Θ_r is tools rotation around the z-axis, W_r is the

required width for the tool, and Q is the quality score of the grasp.

A grasp from an n-channel image $\mathbf{I} \in \mathbb{R}^{n \times h \times w}$ with height h and width w is defined as:

$$G_i = (u, v, d, \Theta_i, W_i, Q) \tag{5.2}$$

where (u, v) corresponds to the center of grasp in image coordinates, d is the depth value, Θ_i is the rotation in camera's frame of reference, W_i is the required width in image coordinates, and Q is the same scalar as in equation (5.1).

The grasp quality score Q is the quality of the grasp at every point in the image and is indicated as a score value between 0 and 1, where a value that is in proximity to 1 indicates a greater chance of grasp success. Θ_i indicates the antipodal measurement of the amount of angular rotation required at each point to grasp the object of interest and is represented as a value in the range $[\frac{-\pi}{2}, \frac{\pi}{2}]$. W_i is the required width which is represented as a measure of uniform depth and indicated as a value in the range of $[0, W_{max}]$ pixels. W_{max} is the maximum width of the antipodal gripper.

To execute a grasp obtained in the image space on a robot, we can apply the following transformations to convert the image coordinates to robot's frame of reference.

$$G_r = T_{rc}(T_{ci}(G_i)) \tag{5.3}$$

where, T_{ci} is a transformation that converts image space into camera's 3D space using the intrinsic parameters of the camera, and T_{rc} converts camera space into the robot space using the camera pose calibration value.

This notation can be scaled for multiple grasps in an image. The collective group of all the grasps can be denoted as:

$$\mathbf{G} = (\mathbf{\Theta}, \mathbf{W}, \mathbf{Q}) \in \mathbb{R}^{3 \times h \times w} \tag{5.4}$$

where $\mathbf{\Theta}, \mathbf{W}$, and $\mathbf{Q}$ represents three images in the form of grasp angle, grasp width, and grasp quality score, respectively, calculated at every pixel of an image using equation (5.2).

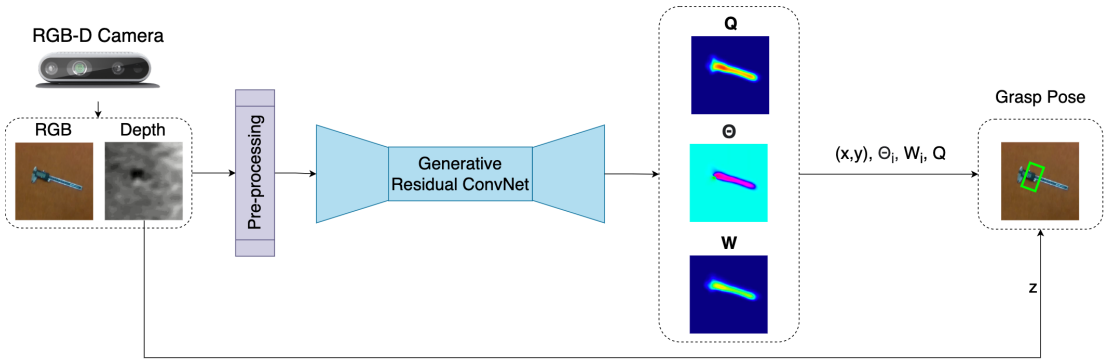


Figure 5.3: Inference module predict suitable grasp poses for the objects in the camera's field of view.

5.3 Proposed Approach

In this section, the proposed dual-module system to predict, plan, and perform antipodal grasps for novel objects in the scene is described. The overview of the proposed system is shown in Fig.5.1. The inference module is used to predict grasp poses in the image frame (G_i) for objects in the camera's field of view. The control module converts these grasp poses into robot frame (G_r) and then plans and executes robot trajectories to perform antipodal grasps.

5.3.1 Inference Module

Fig. 5.3 shows the inference module, which consists of three parts: image pre-processing, generation of pixel-wise grasp using GR-ConvNet v2, and computation of grasp pose(s). The input data is first pre-processed where it is cropped, resized, and normalized to suit the input requirements of GR-ConvNet. If the input has a depth image, it is inpainted to obtain a depth representation [138]. The 224×224 n-channel processed input image is fed into the GR-ConvNet v2. It uses n-channel input that is not limited to a particular type of input modality such as a depth-only or RGB-only image as the input image. Thus, making it generalized for any kind of input modality. The GR-ConvNet generates pixel-wise grasp in the form of grasp angle $\mathbf{\Theta}$, grasp width $\mathbf{W}$, and grasp quality score $\mathbf{Q}$ as the output using the features extracted from the pre-processed image.

The three output images are utilized to infer grasp poses in the image frame (G_i) using equation 5.2. In the case of a single grasp prediction, the pixel with the maximum value in $\mathbf{Q}$ is identified and the corresponding pixel location is used as (u, v) and the pixel value is used as Q. The same

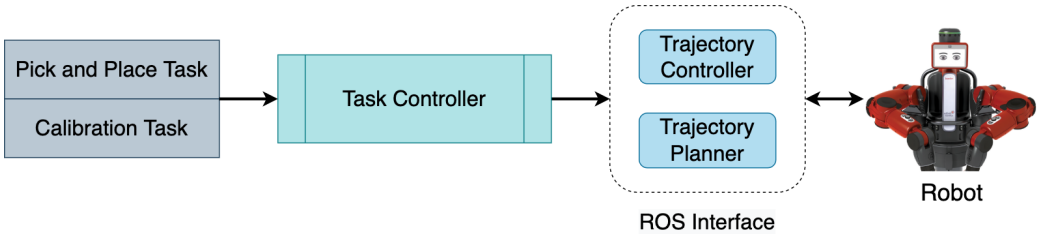


Figure 5.4: Control module uses the grasp poses generated by the inference module to plan and execute robot trajectories to perform antipodal grasps.

pixel locations in $\mathbf{\Theta}$, $\mathbf{W}$ and depth frame are used to determine Θ_i, W_i, and d, respectively. For multi-grasp prediction, local peaks are determined in $\mathbf{Q}$ using [139] to calculate all grasp poses.

5.3.2 Control Module

The control module mainly incorporates a task controller that performs tasks such as pick-and-place and calibration. The architecture of the control module is shown in Fig. 5.4. The task controller requests a grasp pose from the inference module, which returns the grasp pose with the highest quality score. The grasp pose is then converted from the camera frame into the robot frame using equation 5.3 and the transform calculated from an automatic hand-eye calibration process described in section 5.6.3. Further, the grasp pose in the robot frame (G_r) is used to plan a collision-free trajectory to perform the pick and place action using inverse kinematics through a ROS interface. The robot then executes the planned trajectory. Due to the modular approach and automatic hand-eye calibration process, this system can be adapted to any robotic manipulator and camera setup.

5.4 Generative Residual Convolutional Neural Network

Deep learning has redefined how robotic grasping was approached in the past. Further, CNN's have enhanced the way object detection and classification problems have been dealt with in computer vision. Furthermore, state-of-the-art results have been obtained by using residual networks for deeper architectures [5, 38]. These two deep learning techniques are the building blocks of the proposed architecture. In this section, two versions of GR-ConvNet are presented to approximate the complex function $\mathbf{I} \xrightarrow{f_\theta} G_i$, where f_θ denotes a neural network with θ being the weights.

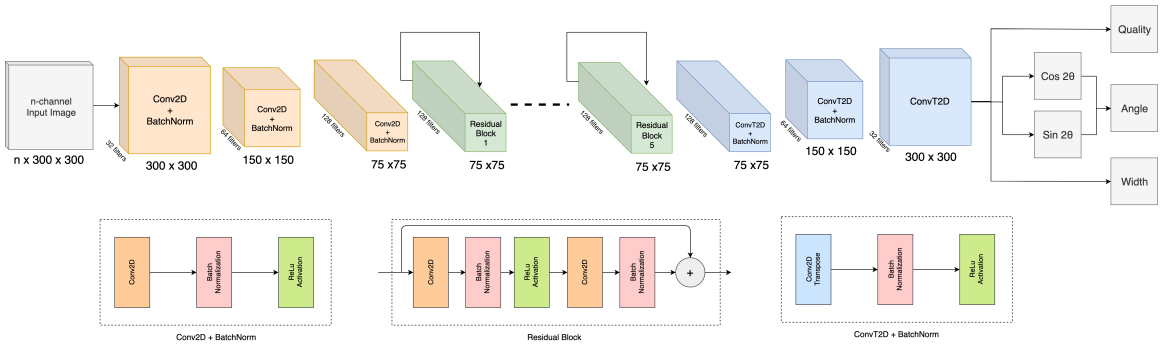


Figure 5.5: Network architecture of the Generative Residual Convolutional Neural Network v1, where n is the number of input channels, and k is the number of filters. The network takes in n-channel input image of size 224×224 and generates pixel-wise grasps in the form of grasp quality, grasp angle and grasp width.

5.4.1 Network Architecture

Fig. 5.5 shows the proposed GR-ConvNet v1 model, which is a generative architecture that takes in n-channel input image of size 224×224 and generates pixel-wise grasps in the form of four images of the same size. These output images consist of grasp quality score $\mathbf{Q}$, required angle $\mathbf{\Theta}$ in the form of $\cos 2\Theta$, and $\sin 2\Theta$ as well as the required width $\mathbf{W}$ of the end effector. Since the antipodal grasp is uniform around $\pm\frac{\pi}{2}$, the angle is extracted in the form of two elements $\cos 2\Theta$ and $\sin 2\Theta$ that produce distinct values that are combined to form the required angle.

The network consists of three parts: encoder, residual layers, and decoder. The n-channel image is passed through the encoder which consists of three convolutional layers, followed by five residual layers, and the decoder which consists of three convolution transpose layers to generate four images. The convolutional layers with a filter size of k extract the features from the input image. The output of the convolutional layer is then fed into five residual layers. As we know, the accuracy increases with increasing the number of layers. However, it is not true when you exceed a certain number of layers, which results in the problem of vanishing gradients and dimensionality error, thereby causing saturation and degradation in the accuracy. Thus, using residual layers enables us to better learn the identity functions by using skip connections [140]. After passing the image through these convolutional and residual layers, the size of the image is reduced to 56×56, which can be difficult to interpret. Therefore, to make it easier to interpret and retain spatial features of the image after convolution operation, the image is up-sampled by using a convolution transpose operation. Thus,

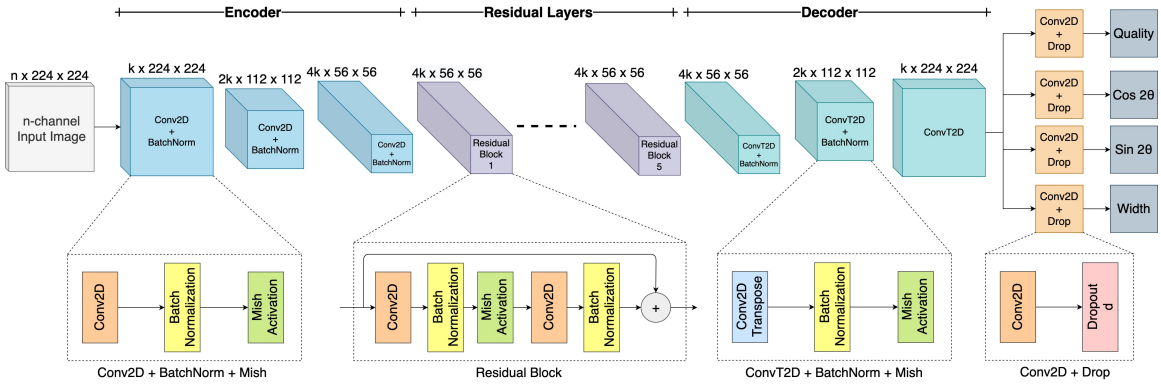


Figure 5.6: Network architecture of the Generative Residual Convolutional Neural Network v2, where n is the number of input channels, k is the number of filters, and d the dropout rate. The network takes in n-channel input image of size 224×224 and generates pixel-wise grasps in the form of grasp quality, grasp angle and grasp width.

obtaining the same size of the image at the output as the size of the input.

Fig. 5.6 shows the proposed GR-ConvNet v2 model. In this improved version of the network, a dropout layer is added after each of the outputs for regularization that favors rare but useful features. Also, the ReLU activation function is replaced with Mish throughout the network, which delivered across the board improvements in training stability. It is believed that the slight allowance for negative values in the Mish activation function allows for better gradient flow compared to the hard zero bound in ReLU.

The proposed network has only 1.9 million parameters with k=32 and n=4, which indicates that the network is comparatively shorter as opposed to other networks [5, 38, 132]. Thereby making it computationally less expensive and faster in contrast to other architectures using similar grasp prediction techniques that contain millions of parameters and complex architectures. The lightweight nature of the model makes it suitable for closed-loop control at a rate of up to 50 Hz.

5.4.2 Training Methodology

For a dataset having objects $D = \{D_1 \dots D_n\}$, input scene images $I = \{I^1 \dots I^n\}$ and ground truth grasp labels in image frame $\widehat{G_i} = \{g_1^1 \dots g_{m_1}^1 \dots g_1^2 \dots g_{m_n}^n\}$, the proposed GR-ConvNet model is trained end-to-end to learn the mapping function $f : (I, D) \rightarrow G_i$, where G_i is the grasp generated by the network in image frame.

The performance of various loss functions is analyzed for the proposed network, and after running a few trials it was found that in order to handle exploding gradients, the smooth L1 loss also known as Huber loss works best. The loss function is define as:

$$\mathcal{L}(y_i, \widehat{y_i}) = \frac{1}{n} \sum_{i}^{N} \textit{SmoothL1}(y_i - \widehat{y_i}) \tag{5.5}$$

where *SmoothL1* is given by:

$$\textit{SmoothL1}(x) = \begin{cases} 0.5x^2, & if\,|x| < 1 \\ |x| - 0.5 & otherwise \end{cases} \tag{5.6}$$

and $y_i \in (Q, \Theta_{cos}\Theta_{sin}, W)$ is the image generated by the model and $\widehat{y_i}$ is the ground truth image. The overall loss function denoted in equation 5.7 is a combined loss of the four output images generated by the model, which are in the form of quality, angle in cos and sin, and required width.

$$\mathcal{L} = \mathcal{L}_{quality} + \mathcal{L}_{cos} + \mathcal{L}_{sin} + \mathcal{L}_{width} \tag{5.7}$$

The training pipeline is improved, as compared to GR-ConvNet v1, by training the models using Ranger optimizer [141] instead of the Adam optimizer [142]. Ranger combines two latest breakthroughs in deep learning optimizers that builds on top of Adam — Rectified Adam, and LookAhead. Training with Rectified Adam gets off to a solid start intrinsically by adding in a rectifier that dynamically tamps down the adaptive learning rate until the variance stabilizes [143]. LookAhead lessens the need for extensive hyperparameter tuning while achieving faster convergence across different deep learning tasks with minimal computational overhead [144].

Instead of keeping the learning rate fixed at 10^{-3} throughout the training process, as for GR-ConvNet v1, the Flat + Cosine anneal is used as ramp-up and ramp-down curve for the learning rates during training. The learning rate is kept constant at 10^{-4} for first few epochs and then annealed to the target learning rate of 10^{-7} according to the law of cosine learning rate [145]. The ramp-up and ramp-down cycle is down twice during training as illustrated in Fig. 5.7.

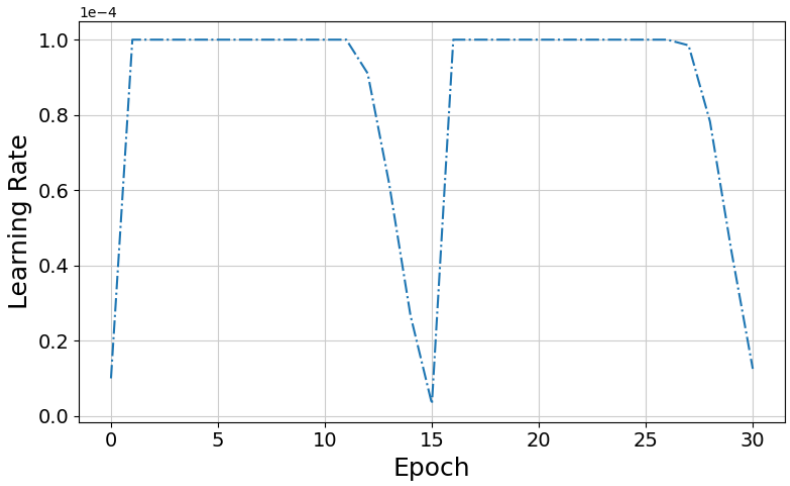


Figure 5.7: Flat + Cosine anneal learning rate curve used for training

5.4.3 Grasp Detection Metric

For a fair comparison of the results, the rectangle metric [8] proposed by Jiang *et al.* is used to report the performance of the proposed system. According to the proposed rectangle metric, a grasp is considered valid when it satisfies the following two conditions:

- The Jaccard index or intersection over union (IoU) score between the ground truth grasp rectangle and the predicted grasp rectangle is more than 25%.
- The offset between the grasp orientation of the predicted grasp rectangle and the ground truth rectangle is less than $30°$.

This IoU based metric requires a grasp rectangle representation, but the model predicts image-based grasp representation $\widehat{G_i}$ using equation 5.2. Therefore, in order to convert from the image-based grasp representation to the rectangle representation, the value corresponding to each pixel in the output image is mapped to its equivalent rectangle representation as shown in Fig. 5.8.

5.5 Network Evaluation

The two GR-ConvNets are evaluated on three publicly available datasets to examine the outcome for each of the datasets based on factors, such as the size of the dataset, type of training data, and demonstrate the model's capacity to generalize to any kind of object.

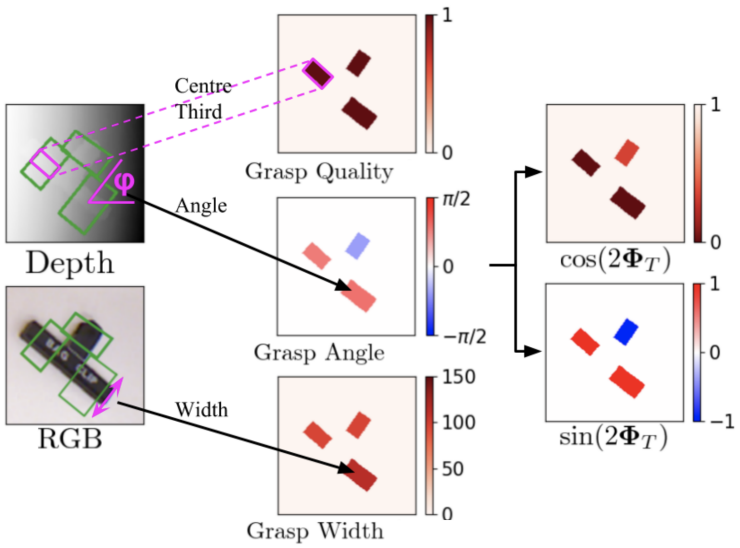


Figure 5.8: Generation of data to train and evaluate the models similar to [13]. **Left**: The cropped and rotated depth and RGB images from the dataset with the ground-truth positive grasp rectangles representing antipodal grasps (shown in green). **Right**: From the ground-truth grasps, the grasp angle **Θ**, grasp width **W**, and grasp quality **Q** images to train the network.

The model is trained using three random seeds and reported the average of the three seeds. The execution times for the proposed model are measured on a system running Ubuntu 18.04 with an Intel Core i7-7800X CPU clocked at 3.50 GHz and an NVIDIA GeForce GTX 1080 Ti graphics card with CUDA 11.

5.5.1 Datasets

There are a limited number of publicly available antipodal grasping datasets. Table 5.2 shows a summary of the publicly available antipodal grasping datasets. Three of these datasets are used for training and evaluating the proposed models. The first one is the CGD [8], which is the most common grasping dataset used to benchmark results, the second is a simulation JGD [117], which is more than 50 times bigger than the CGD, and the third is the more recent G1BD [137].

Cornell Grasp Dataset

The extended version of CGD comprises of 1035 RGB-D images with a resolution of 640×480 pixels of 240 different real objects with 5110 positive and 2909 negative grasps. The annotated ground

Table 5.2: Summary of antipodal robotic grasping datasets

Dataset	Modality	Type	Objects	Images	Grasps
Cornell [8]	RGB-D	Real	240	1035	8k
Multi-Object [37]	RGB-D	Real	-	96	2.9k
Jacquard [117]	RGB-D	Sim	11k	54k	1.1M
Dexnet [23]	Depth	Sim	1500	6.7M	6.7M
VR-Grasping [136]	RGB-D	Sim	101	10k	4.8M
VMRD [119]	RGB	Real	100	4.6k	100k
Graspnet [137]	RGB-D	Real	88	97k	1.2B

truth consists of several grasp rectangles representing grasping possibilities per object. However, it is a small dataset for training the GR-ConvNet model, therefore an augmented dataset is created using random crops, zooms, and rotations which effectively has 51k grasp examples. Only positively labeled grasps from the dataset were considered during training.

Jacquard Grasping Dataset

The JGD is built on a subset of ShapeNet which is a large CAD models dataset. It consists of 54k RGB-D images with a resolution of 1024×1024 pixels and annotations of successful grasping positions based on grasp attempts performed in a simulated environment. In total, it has 1,181,330 unique grasp annotations for 11,619 distinct objects in simulation.

Graspnet 1-billion Dataset

G1BD is a large-scale benchmark dataset that contains 190 cluttered and complex scenes captured by Kinect Azure and RealSense D435 cameras. In total, it contains 97,280 RGB-D images with over 1.1 billion grasp poses of 88 different objects. To use the raw rectangular images with resolution of 1280×720 pixels, a square image of size 720×720 pixels is cropped around the mean center of the ground truth bounding box. G1BD consists of 190 scenes, and each includes 256 images with annotations.

5.5.2 Evaluation on Cornell Dataset

As in previous works [10, 113, 5, 118, 36], a cross-validation setup is followed using image-wise, and object-wise data splits. The image-wise data split means that the training and validation sets are divided randomly, whereas the object-wise data split means that the objects in the validation set do not appear in the training set. Table 5.3 shows the performance of the proposed method compared to other techniques used for grasp prediction. The state-of-the-art accuracy of 98.8% on image-wise split and 97.7% on object-wise split is obtained using the GR-ConvNet v2 model, outperforming all competitive methods as seen in Table 5.3. The results obtained on the previously unseen objects in the dataset depict that the proposed network can predict robust grasps for different types of objects in the validation set. The data augmentation performed on the CGD improved the overall performance of the network. Furthermore, the recorded prediction speed of 20ms per image suggests that GR-ConvNet is suitable for real-time closed-loop applications.

The accuracy of the trained model is also evaluated at higher IoU thresholds. Table 5.4 contains the comparison of the results for CGD at different Jaccard thresholds. In contrast to previous work [36, 37, 149], the proposed approach maintains a high prediction accuracy even if the grasp detection metric is stricter. The proposed model outperforms the network proposed in [37] by 14% and in [149] by 11% at the 40% IoU threshold.

5.5.3 Evaluation on Jacquard Dataset

For the JGD, the network is trained on 90% of the dataset images and validated on 10% of the remaining dataset. As the JGD is much larger than the CGD, no data augmentation was required. Table 5.5 compares the results of the proposed network with other methods on the JGD. The IoU metric is used for grasp evaluation and an accuracy of 95.1% is observed using GR-ConvNet v2 with RGB-D data as the input and a batch size of 16.

Depierre *et al.* released a web based Simulated Grasp Trails (SGT) system to upload the scene index and corresponding grasp prediction [117]. The system rebuilds the scene in simulation and the grasp is executed by the simulated robot. The results of the execution are emailed to the user. These results for the proposed models are reported in Table 5.5. The results are the new state-of-the-art with an accuracy of 91.4% using the SGT metric.

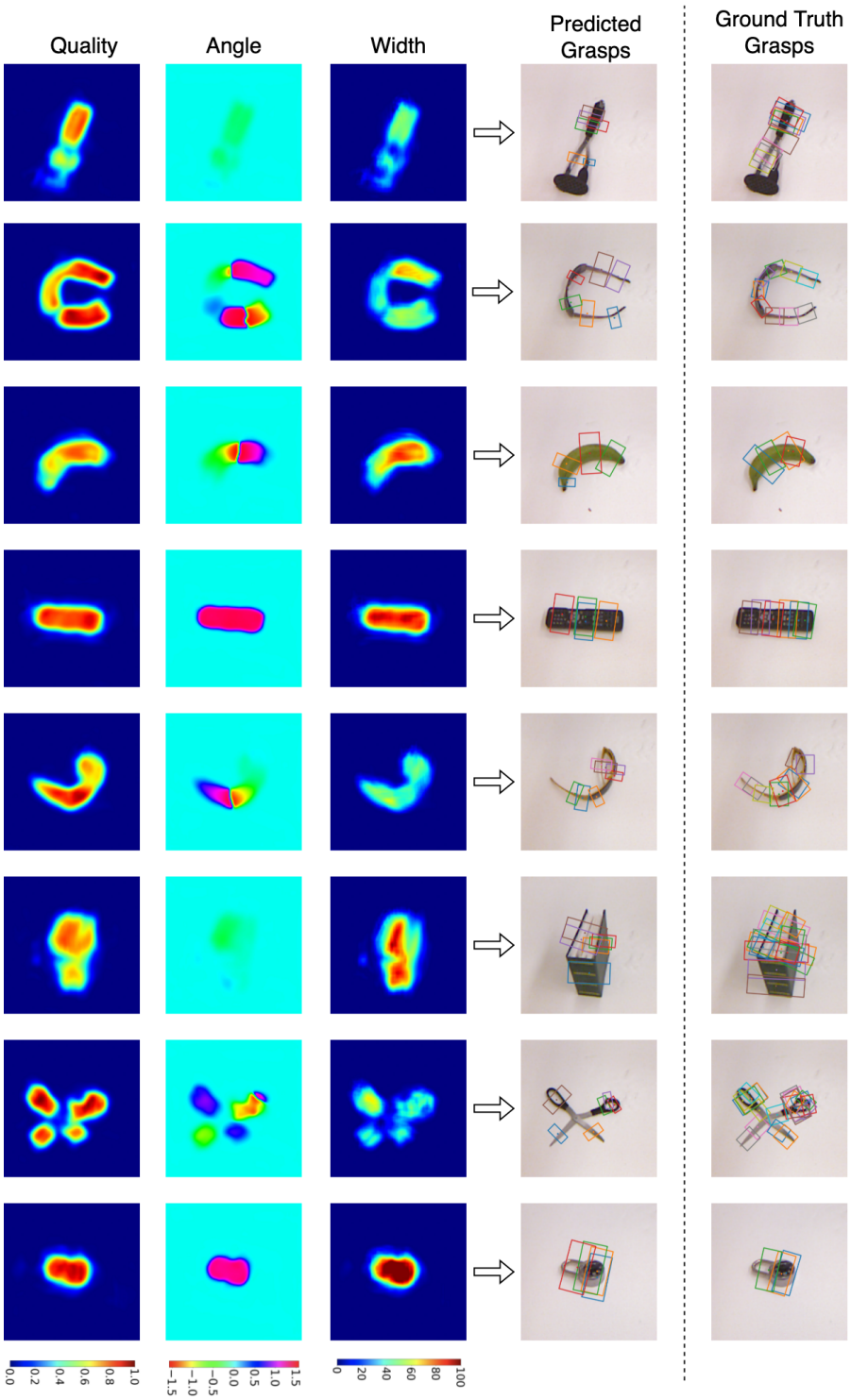


Figure 5.9: Qualitative results on CGD. The top three rows (quality, angle and width) are the output of GR-ConvNet. The bottom two rows are the predicted and ground truth grasps in rectangle grasp representation.

Table 5.3: Comparative results on CGD

Authors	**Algorithm**	**Accuracy (%)**		**Speed**
		IW	OW	(ms)
Jiang [8]	Fast Search	60.5	58.3	5000
Lenz [10]	SAE, struct. reg.	73.9	75.6	1350
Redmon [113]	AlexNet, MultiGrasp	88.0	87.1	76
Wang [34]	Two-stage closed-loop	85.3	-	140
Asif [112]	STEM-CaRFs	88.2	87.5	-
Kumra [5]	ResNet-50x2	89.2	88.9	103
Guo [36]	ZF-net	93.2	89.1	-
Zhou [38]	FCGN, ResNet-101	97.7	96.6	117
Asif [118]	GraspNet	90.2	90.6	24
Chu [37]	Multi-grasp Res-50	96.0	96.1	120
Morrison [146]	GG-CNN	73.0	69.0	**19**
Morrison [13]	GG-CNN2	84.0	82.0	20
Karaoguz [147]	GRPN	88.7	-	200
Zhang [119]	ROI-GD, ResNet-101	93.6	93.5	39
Wang [148]	DD-Net, Hourglass	97.2	96.1	-
This work	GR-ConvNet	97.7	96.6	20
	GR-ConvNet v2	**98.8**	**97.7**	20

Table 5.4: Grasp prediction accuracy (%) for CGD at different Jaccard thresholds

Approach	IoU>25%	IoU>30%	IoU>35%	IoU>40%
Guo *et al.* [36]	93.2	91.0	85.3	-
Chu *et al.* [37]	96.0	92.7	87.6	82.6
Wang *et al.* [149]	94.4	92.8	90.2	85.7
GR-ConvNet v2	**98.8**	**98.8**	**98.8**	**96.6**

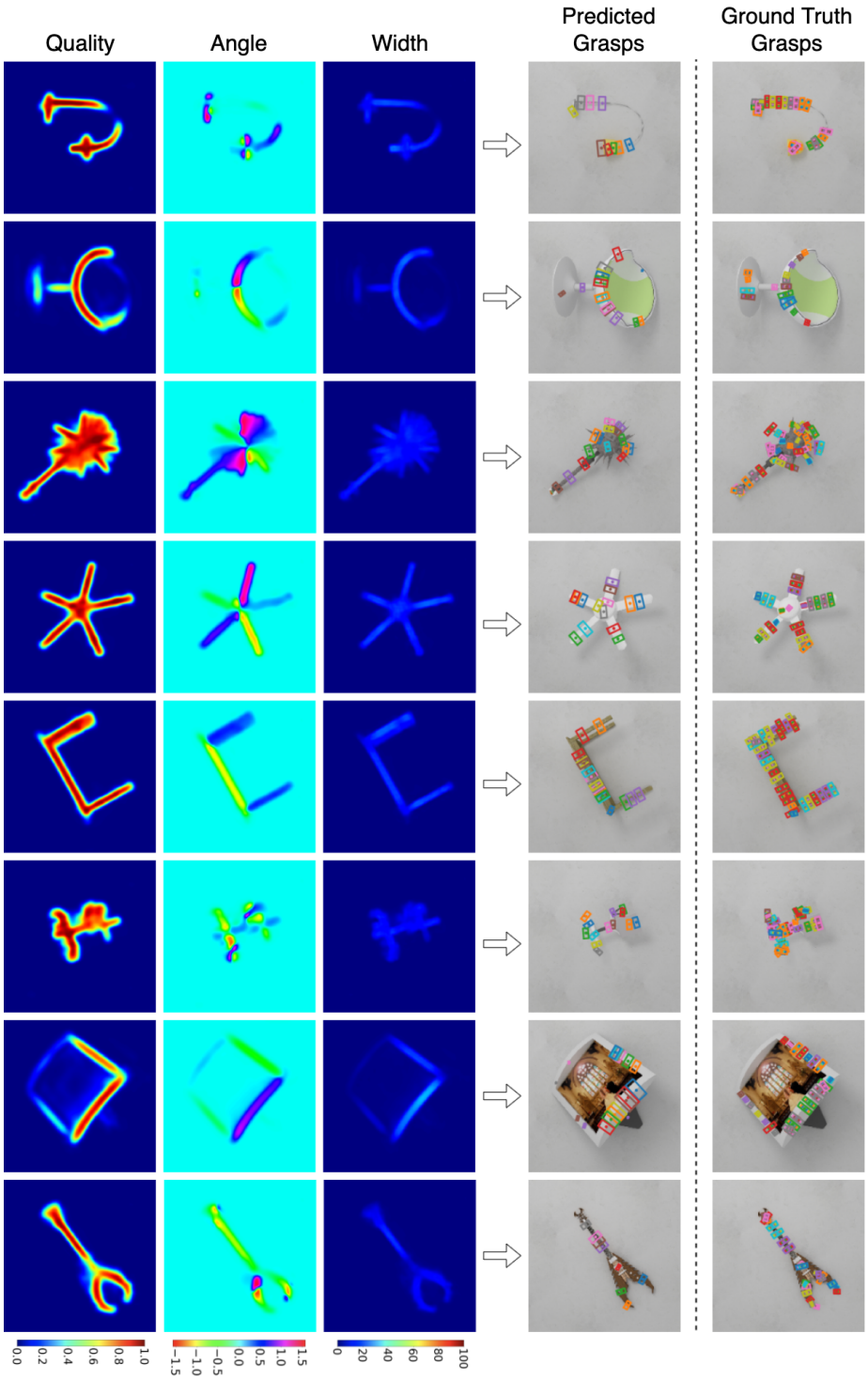


Figure 5.10: Qualitative results on JGD. The top three rows (quality, angle and width) are the output of GR-ConvNet. The bottom two rows are the predicted and ground truth grasps in rectangle grasp representation.

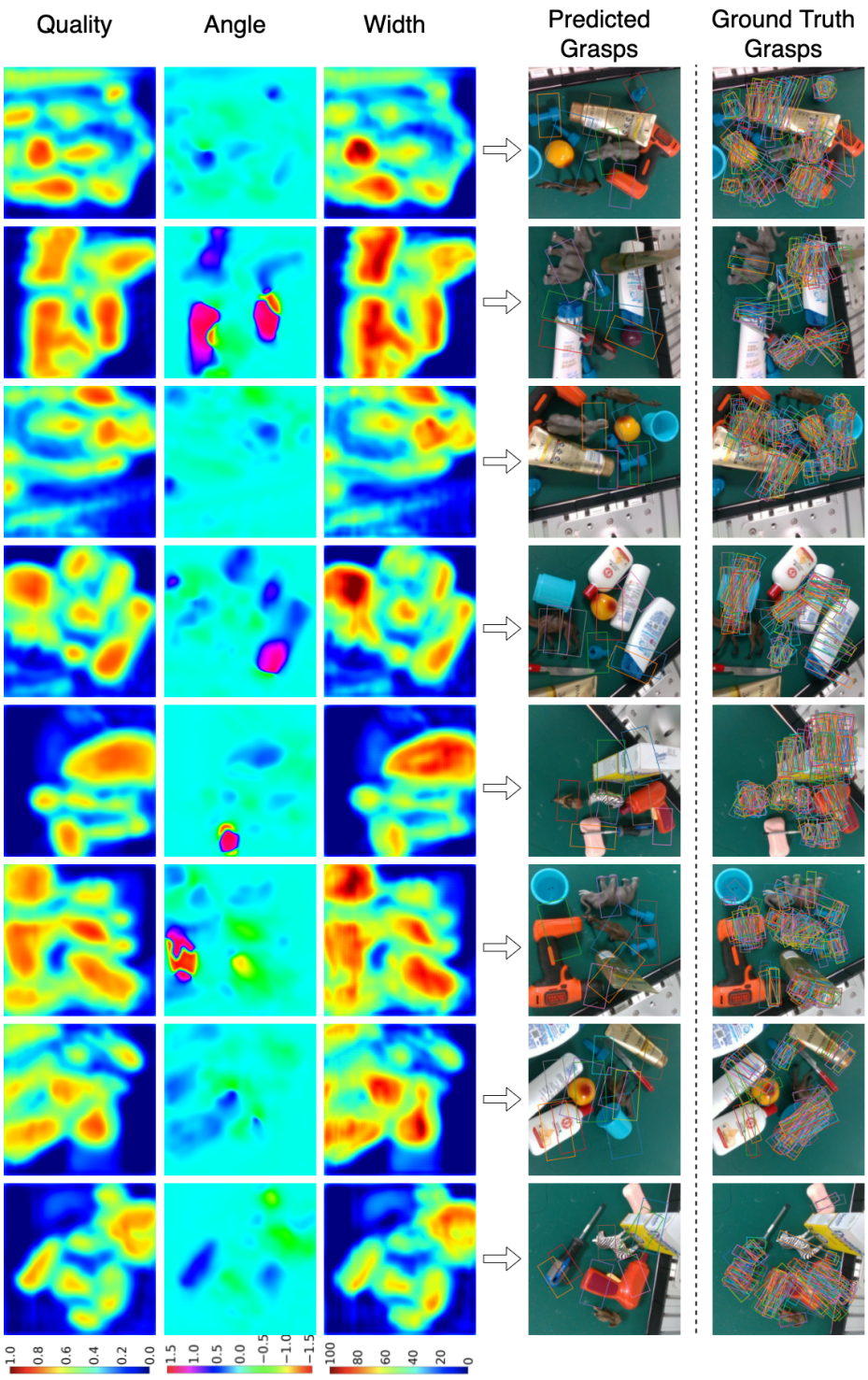


Figure 5.11: Qualitative results on G1BD. The top three rows (quality, angle and width) are the output of GR-ConvNet. The bottom two rows are the predicted and ground truth grasps in rectangle grasp representation.

Table 5.5: Comparative results on the JGD

Authors	**Algorithm**	**Accuracy (%)**	
		IoU	SGT
Depierre *et al.* [117]	Jacquard	74.2	72.4
Morrison *et al.* [13]	GG-CNN2	84.0	85.0
Zhou *et al.* [38]	FCGN, ResNet-101	92.8	81.9
Zhang *et al.* [119]	ROI-GD, ResNet-101	93.6	-
Wang [148]	DD-Net, Hourglass	**97.0**	89.4
	GR-ConvNet (b=8, d=0.0)	94.6	89.5
	GR-ConvNet v2 (b=16, d=0.1)	95.1	**91.4**

Table 5.6: Comparative results for grasp prediction accuracy (%) on G1BD for different validation splits

Approach	**5-fold**	**Seen**	**Similar**	**Novel**
GGCNN [13]	82.3	83.0	79.4	76.3
Multi-grasp [37]	86.0	82.7	77.8	72.7
GR-ConvNet (k=32, d=0.0)	96.1	96.2	94.8	87.9
GR-ConvNet v2 (k=32, d=0.1)	**98.7**	**97.9**	**96.0**	**90.5**

5.5.4 Evaluation on Graspnet Dataset

G1BD is gigantic, and the load on the computer when loading large amounts of grasps can cause problems. The load on compute resources is reduced by reducing the number of ground truth labels loaded per scene and pre-processing the dataset. Each grasp label in the G1BD has a quality measure associated with it, which is measured based on the friction coefficient μ. The ground-truth labels that are outside the cropped image and have poor grasp quality ($\mu < 0.4$) are discarded.

In addition to the 5-fold cross-validation split (similar to the one used for CGD), the predesignated data provided with G1BD is used for training and testing. There are a total of 190 scenes. The first 100 scenes are used for training, and the testing data has been split into three categories: (i) objects already seen (scenes 101-130), (ii) objects similar to training (scenes 131-160), and (iii) objects not seen before (scenes 161-190). The validation results compared to prior work are summarised in Table 5.6 for the three testing splits provided with the dataset and the 5-fold cross

Table 5.7: Results on Novel Objects

Object	**Accuracy (%)** (Trained on Cornell)	**Accuracy (%)** (Trained on Jacquard)
Toy	100	100
Earphones	100	95
Bottle	40	65
Opener	100	100
Shaker	100	100
Controller	100	100
Remote	100	90
Pen	100	100
Headphones	100	90
Scissor	95	90

validation method. It can be seen that the proposed GR-ConvNet outperforms the state-of-the-art counterparts ([13], [37]) by a large margin, with an improvement of 14.2% for the novel test set, which demonstrates its effectiveness in handling unseen scenarios.

5.5.5 Evaluation on Novel Objects

Furthermore, the performance was validated on actual objects by using an Intel RealSense depth camera. Random novel objects were chosen to authenticate the results obtained on both the datasets on which the network had been tested. The effectiveness of the network was tested in different types of environments including varying light conditions for grasping individual objects or grasping objects in a clutter.

Fig. 5.12 shows the qualitative results obtained on previously unseen objects. The figure consists of output in the image representation G_i in the form of grasp quality score Q, the required angle for grasping Θ_i, and the required gripper width W_i. It also includes the output in the form of a rectangle grasp representation projected on the RGB image and the ground truth grasps.

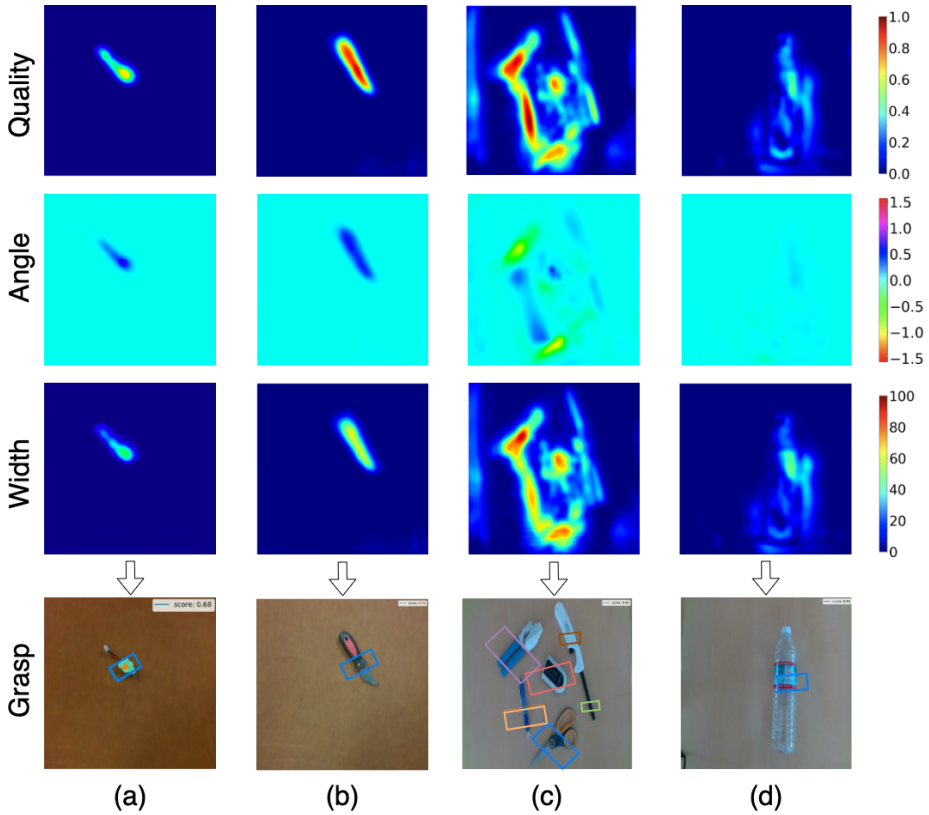


Figure 5.12: Qualitative results. Quality, angle and width are the output of GR-ConNet which are used to infer grasp rectangle. (a-b) Single grasp for single unseen object. (c) Multiple grasps for multiple objects. (d) Poor grasp for transparent object.

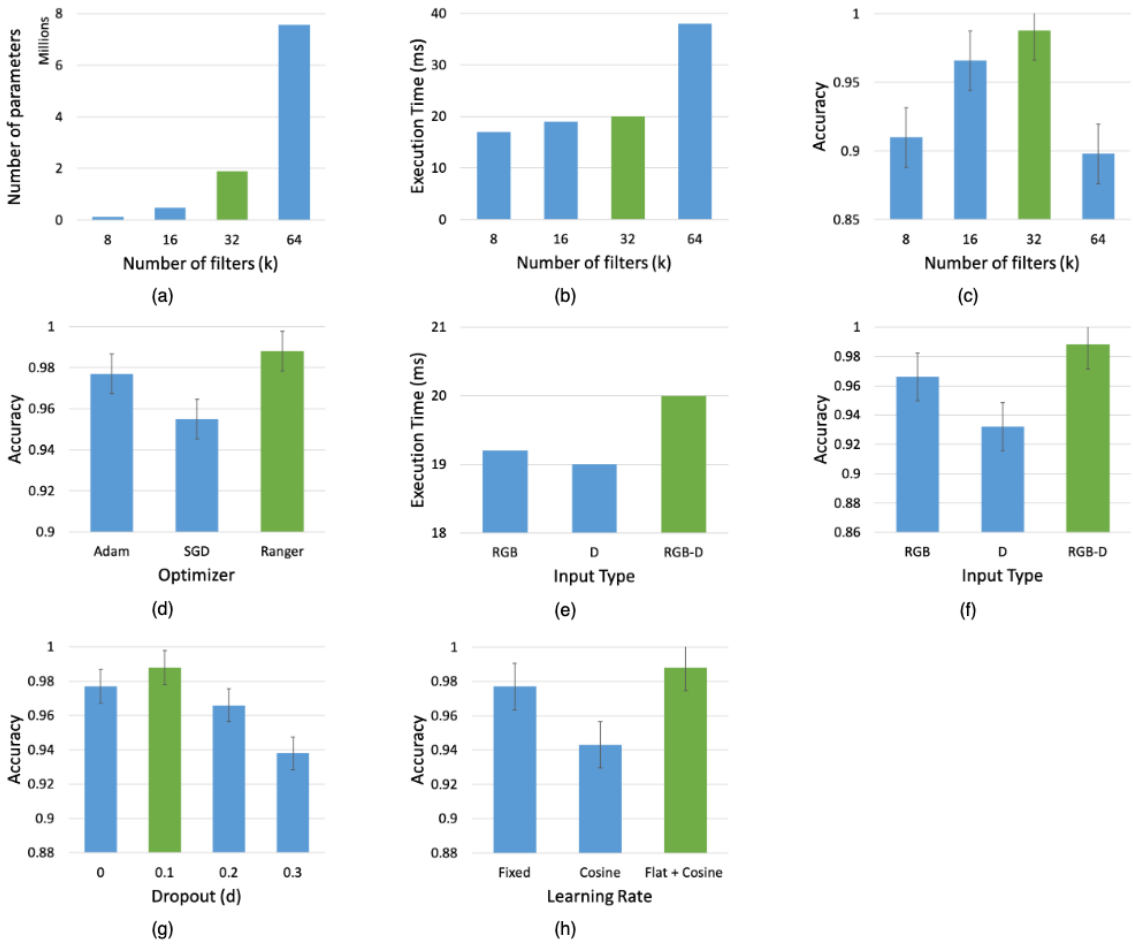


Figure 5.13: Ablation study results for GR-ConvNet by training on different filter sizes (k), input channels (n), dropout (d), optimizers and learning rates. The model is evaluated using the 25% IoU metric against the CGD. Green indicates the selected parameter.

5.5.6 Ablation Study

To better understand the performance of the proposed model, a series of experiments were carried out by tweaking a number of parameters, including filter size, batch size, learning rate, and varying the number of layers. After evaluating the performance of multiple parameters the architecture that gave us the highest grasp prediction accuracy along with the lowest recorded inference time is determined. This section discusses these experiments and elaborates on the contributions of each of the individual components and parameters that were chosen during the network design by evaluating the model on the CGD.

Firstly, the network is evaluated by varying the number of filters (k) at each layer as shown in

Fig. 5.13 (a-c). It can be seen from the figure that varying the number of filters plays a significant role in determining the accuracy of the network. It was found that, by increasing the number of filters (k), the accuracy increases proportionately until it reaches a certain value and then starts decreasing substantially. At this point, it was also observed that the number of parameters and execution time increased drastically. In comparison, increasing the number of filters had little impact on the execution time as opposed to providing higher accuracy. However, the accuracy dropped when the number of filters (k) was increased beyond k = 32 while also increasing the number of parameters and execution time. Thus, the set of parameters indicated in green are chosen, that yielded the maximum accuracy in comparison to an increased number of parameters and execution time.

Furthermore, the performance of the proposed network is evaluated on different input modalities. The modalities that the model was tested on included uni-modal input such as depth only and RGB only input images; and multi-modal input such as RGB-D images. Fig. 5.13 (e-f) shows the performance of the network on different modalities. Although the RGB-only input data had execution times lower than those of the RGB-D input data, it had a lower accuracy than the RGB-D input. The depth-only input data had the lowest execution times and the lowest accuracy compared to the RGB and the RGB-D input data. Thus, we observe that the network performed better on multi-modal data in comparison to uni-modal data since multiple input modalities enabled better learning of the input features.

Additionally, to study the effect of regularization on the network, dropout layers after the deconvolution layers are added. The model is tested with a dropout of 10%, 20%, and 30% feature drop against no dropout. From Fig. 5.13 (g) we can see that a dropout of 10% bumped the accuracy from 97.7% to 98.8% and a dropout of 20% and 30% reduced the accuracy below 97.7%. From the results, we can see that the model was slightly overfitting, and by dropping 10% of the features during training, it achieved an increase in the success rate by 1% on the validation set.

Finally, the impact of different optimizers and learning rate discussed in section 5.4.2 on the grasp prediction accuracy is studied. Fig. 5.13 (d) shows that Ranger optimizer improved the accuracy by 3.3% compared of the standard SGD optimizer. An improvement of 1.1% accuracy (shown in Fig. 5.13 (h)) is also observed when the model is trained using Flat + Cosine anneal as ramp-up and ramp-down curve for the learning rates instead of a fixed learning rate as in [3, 13].

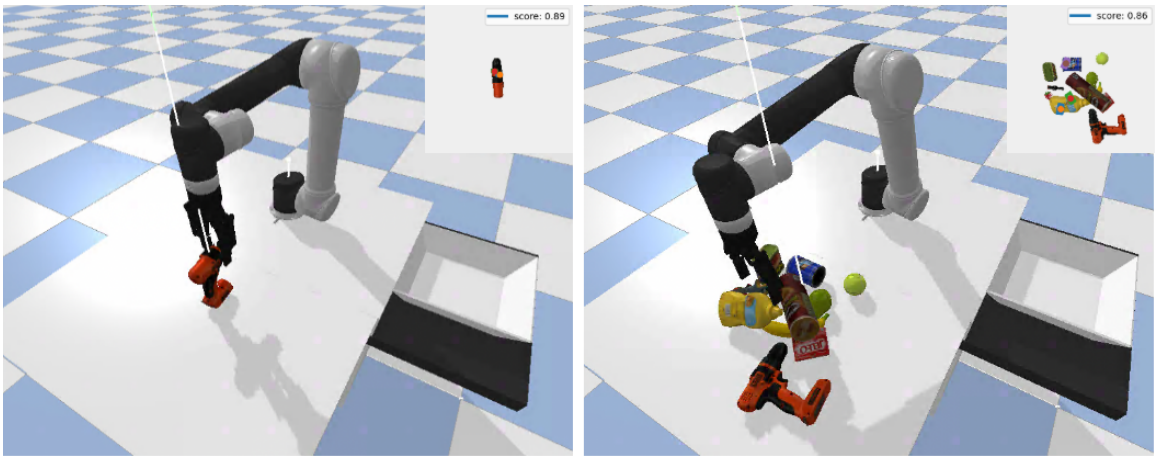


Figure 5.14: Examples of antipodal grasping in simulation. **Left**: Grasping YCB objects in isolation. **Right**: Grasping YCB objects in clutter.

5.6 Antipodal Grasping using Generative Residual Convolutional Neural Network

In this section, the antipodal robotic grasping experiments and results are discussed. Along with the state-of-the-art results on three standard datasets, it is also demonstrated that the proposed system equally outperforms in robotic grasping experiments for novel real-world objects. Furthermore, it is shown that the proposed model cannot only generate a single grasp for isolated objects but also multiple grasps for multiple objects in clutter. For a fair comparison, an open-loop grasping method similar to previous work ([10, 11, 23]) is implemented and evaluated the approach on: (i) household objects, (ii) adversarial objects and (iii) objects in clutter.

5.6.1 Simulation Setup

To evaluate antipodal robotic grasping in simulation, a simulation environment (shown in Fig. 5.14) is developed in PyBullet [150], where a UR5e with a Robotiq 2F-140 anitpodal gripper perceived the environment using an RGB-D camera looking over the robot's workspace. Simulated objects from the Yale-CMU-Berkeley (YCB) object set [151], a benchmarking object set for robotic grasping, are used for the simulation experiments. At the beginning of each experiment, the robot is set to a predefined pose, and randomly selected object(s) are placed in an arbitrary pose(s) inside the robot's workspace. In all experiments, the robot knows in advance about the placement pose in a basket,

Table 5.8: Pick success rate (%) on YCB objects in simulation

Approach	Training Dataset	Isolated	Cluttered
GGCNN	Cornell	79.0*	74.5*
GGCNN	Jacquard	85.5*	82.0*
GR-ConvNet v2	Cornell	**98.0**	92.0
GR-ConvNet v2	Jacquard	97.5	**96.5**

*The accuracy is calculated using the open source code and model.

while the GR-ConvNet model needs to predict the best graspable pose for the given scene and send it to the the robot to grasp the object, pick it up, and put it in the placement basket. A particular grasp is recorded as a success if the object is inside the basket at the end of the pick and place mission.

5.6.2 Simulation Experiments

The performance of GR-ConvNet trained on Cornell and Jacquard is evaluated in two different scenarios: isolated and cluttered. For the isolated object scenario, a randomly selected object is placed in an arbitrary pose inside the robot's workspace and the robot executed the pick and place mission. In the case of the cluttered scenario, to generate a simulated scene containing a cluttered pile of objects, 10 objects are randomly spawned into a box placed on top of the table. The box is removed once all objects become stable, and then the robot repeatedly executes pick-and-place missions until there are no objects left in the robot's workspace.

To report the performance of the model, the pick success rate is measured, which is the ratio of the number of successful grasps and the number of attempts. For each experiment, a total of 200 grasp attempts are run and the pick success rate is reported. Table 5.8 summarizes the results for different models tested with objects in isolation and clutter. It can be seen that the proposed GR-ConvNet performs significantly better than the GGCNN models in both isolated and cluttered scenarios, with improvements of 12.5% and 14.5% respectively.

5.6.3 Real-world Setup

The real-world experiments were conducted on the 7-DoF Baxter Robot by Rethink Robotics. A two-fingered parallel gripper was used for grasping the test objects. Intel RealSense Depth Camera

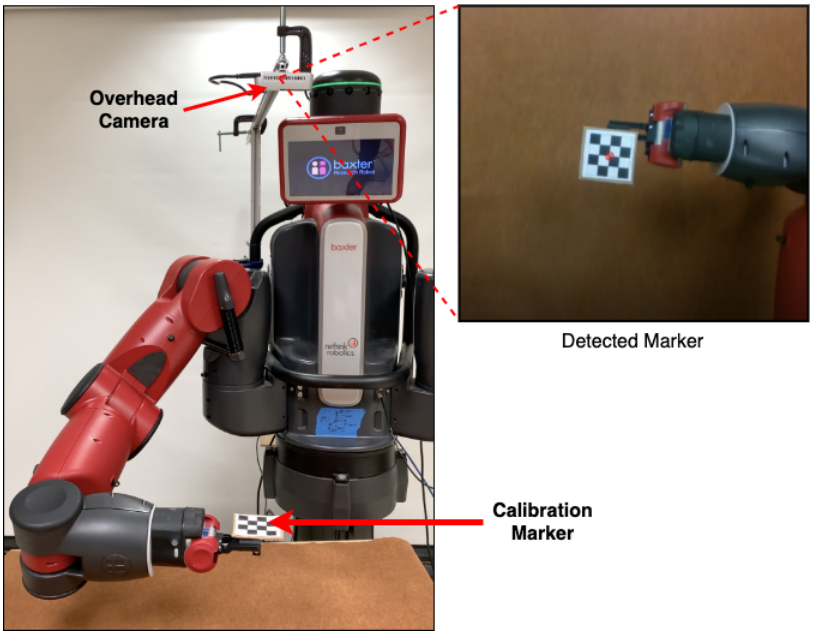


Figure 5.15: Setup for hand-eye calibration procedure.

D435 that uses stereo vision to calculate depth was used to get the scene image. The image bundle consists of a pair of RGB sensors, depth sensors, and an infrared projector. The camera was statically mounted behind the robot arm looking over the shoulder from where it captured 640 × 480 RGB-D images for each inference.

Hand-eye Calibration

The statically mounted overlooking camera is localized with respect to the robot frame using an automatic calibration task developed in the control module. Fig. 5.15 shows the setup used to perform the calibration procedure. The camera detects the location of the checkerboard pattern marker mounted on the robot TCP and optimizes the extrinsics as the robot's arm moves over a predefined grid of 3D locations in the camera's field of view. The procedure generates transformations T_{rc} and T_{ci}, which are used to convert the grasp poses in image frame (G_i) to robot's frame of reference (G_r).

(a) (b)

Figure 5.16: Objects used for robotic grasping experiments. (a) Household test objects. (b) Adversarial test objects.

Test Objects

A total of 35 household objects were chosen for testing the performance of the system. Each object was tested individually for 10 different positions and orientations which resulted in 350 grasp attempts. The objects were chosen such that each object represented a different shape, size, and geometry; and had minimum or no resemblance with each other. A mix of deformable, difficult to grasp, reflective, and small objects that need high precision is created. Fig. 5.16a shows the set of household objects that are used for the experiments.

Another set consisting of 10 adversarial objects with complex geometry was used to evaluate the accuracy of the proposed system. These 3D printed objects have abstract geometry with indefinite surfaces and edges that are hard to perceive and grasp. Each of these objects was tested in isolation for 10 different orientations and positions and made up of a total of 100 grasp attempts. Fig. 5.16b shows the adversarial objects used during the experiments.

Grasp Execution

Grasp poses predicted by the inference module are used to execute the grasps in an open-loop using a pick and place task. This task plans and executes open-loop collision-free trajectories considering the robot's arm motion for planning the trajectory towards a perch position with the gripper tip aligned with and approximately 15 cm above the grasp pose G_r. The arm then moves vertically

down until it reaches the required grasp pose, or a collision is detected by the robot using the force feedback. The robot then closes the antipodal gripper and moves back to the perch position. A grasp is successful if the robot lifts the object in the air at the perch position 15 cm above the grasp pose.

5.6.4 Real-world Experiments

Industrial applications such as warehouses require objects to be picked in isolation as well as from a clutter. To understand how well the model trained on the CGD generalizes to novel objects, grasping experiments with household and adversarial objects are performed in isolation and clutter.

Grasping in Isolation

For the experiment with objects in isolation, each object was tested for 10 different positions and orientations. The robot performed 334 successful grasps of the total 350 grasp attempts on household objects resulting in a grasp success rate of 95.4%, and 93 successful grasps out of 100 grasp attempts on adversarial objects giving a grasp success rate of 93%.

Grasping in Clutter

To evaluate the performance of the proposed system for cluttered objects, multiple trials were carried out with a set of 10 to 15 distinct objects for each run. The objects were shaken in a box and emptied into a pile in front of the robot to create a cluttered scene. The robot continuously attempted to grasp and remove the object from the scene after a successful grasp. Each run was terminated when there were no objects in the camera's field of view. An example of this is shown in Fig. 5.17 for household objects and in Fig. 5.18 for adversarial objects. Each run was carried out without object replacement, and a mean grasp success rate of 93.5% is recorded on household object clutter and 91.0% on adversarial object clutter. This shows the ability of the proposed method to maintain a high level of accuracy when grasping from a clutter of multiple objects.

Despite the model being trained only on isolated objects in the CGD, it is observed that it was able to efficiently predict grasps for objects in clutter. A comparison of the results for the proposed approach compared to other deep learning-based approaches in robotic grasping is shown in Table 5.9 and Table 5.10. These results indicate that GR-ConvNet can effectively generalize to new objects that it has never seen before. Furthermore, we can see the robustness of GR-ConvNet

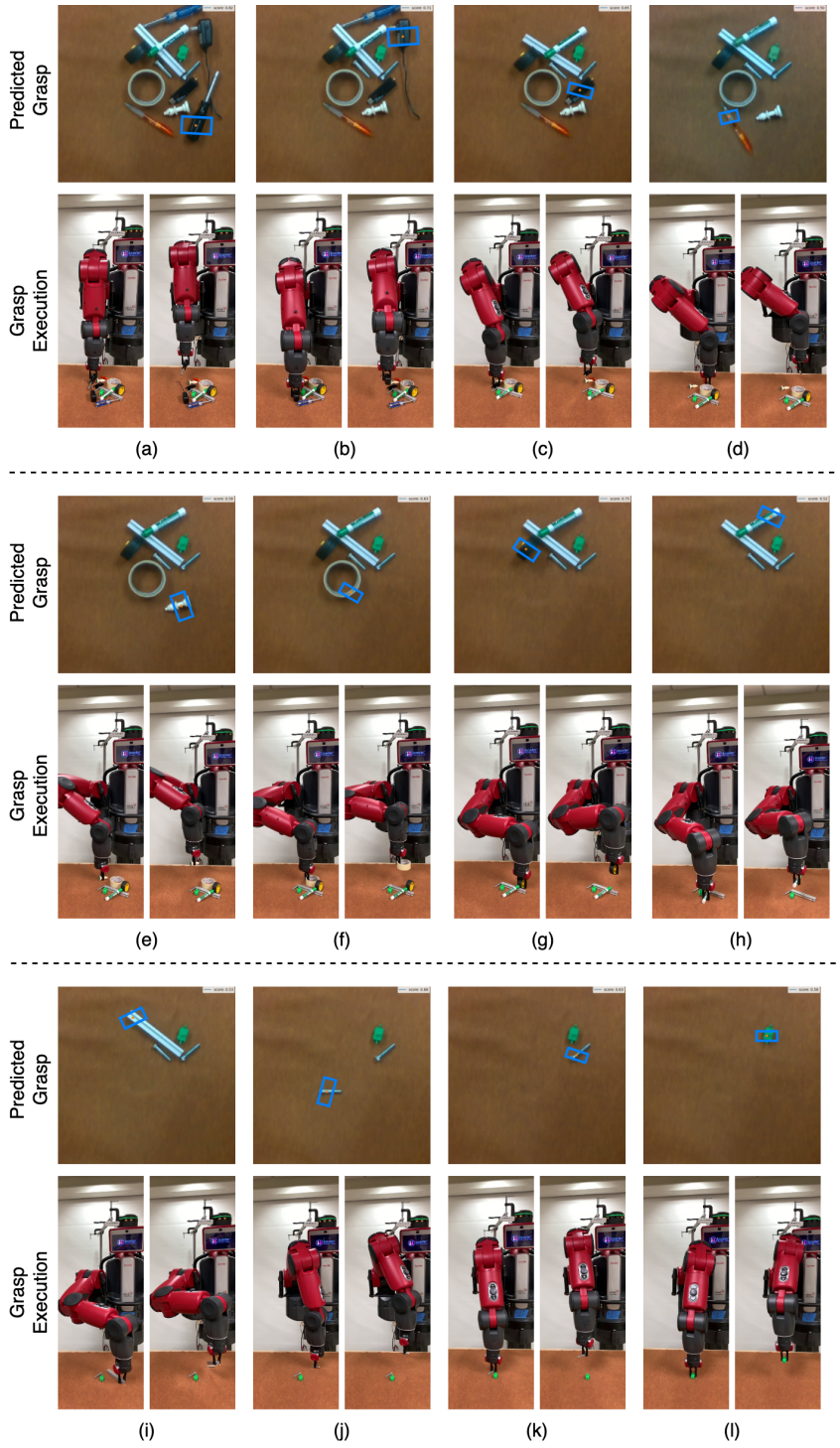


Figure 5.17: Visualization of the household objects clutter scene removal task in the real world using the proposed GR-ConvNet model trained on Cornell dataset. (a) - (l) show the grasp pose generated by inference module (top), robot grasping the object (bottom left), and robot retracting after successful grasp (bottom right) for each object.

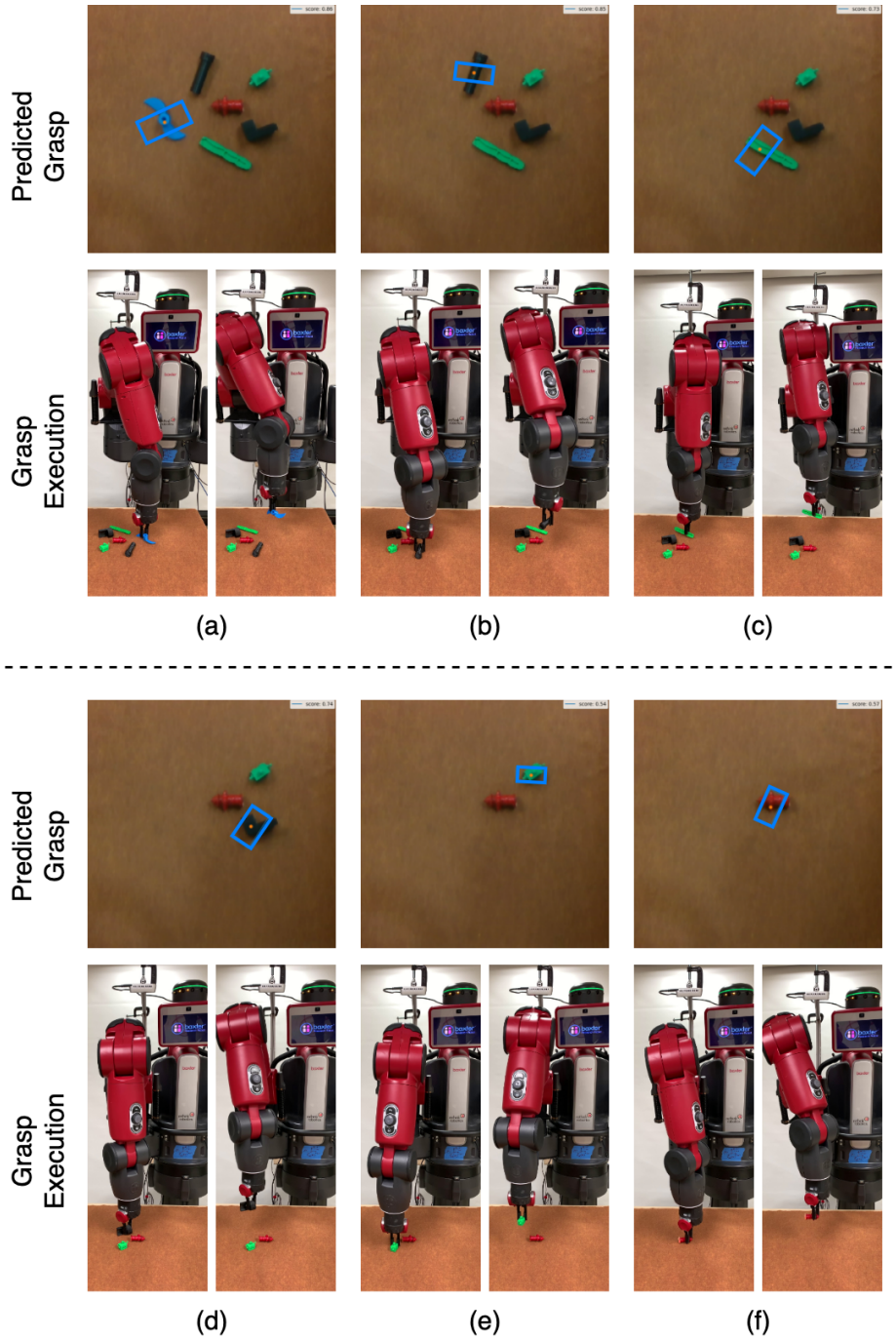


Figure 5.18: Visualization of the adversarial objects clutter scene removal task in the real world using the GR-ConvNet model trained on CGD. (a) - (l) show the grasp pose generated by inference module (top), robot grasping the object (bottom left), and robot retracting after successful grasp (bottom right) for each object.

Table 5.9: Comparative results for grasp success rate (%) in real-world for objects in isolation

Approach	Training Dataset	Household Objects	Adversarial Objects
SAE, struct. reg. [10]	Cornell	89.0 (89/100)	-
Alexnet based CNN [11]	Custom	66.0 (99/150)	-
Robust Best Grasp [41]	ModelNet in Sim	80.0 (80/100)	-
Multi-grasp Res-50 [37]	Cornell	89.0 (89/100)	-
DexNet 2.0 [23]	DexNet in Sim	80.0 (40/50)	92.5 (74/80)
CTR [25]	Custom in OpenRAVE	**97.5** (39/40)	-
GGCNN [13]	Cornell	91.6 (110/120)	83.7 (67/80)
GR-ConvNet	Cornell	95.4 (334/350)	**93.0** (93/100)

Table 5.10: Comparative results for grasp success rate (%) in real-world for cluttered scene removal

Approach	Training Dataset	Household Objects	Adversarial Objects
Alexnet based CNN [11]	Custom	38.4 (50/130)	-
GPD [109]	CAD models	77.3 (116/138)	-
CTR [25]	Custom in OpenRAVE	(66/74)	-
GGCNN [13]	Cornell	86.4 (83/96)	-
GR-ConvNet	Cornell	**93.5** (187/200)	**91.0** (91/100)

since it is capable of predicting antipodal grasps for multiple objects in a cluttered scene with a high accuracy of 93.5%. The performance of GR-ConvNet in isolated scenarios is comparable to CTR [25] and DexNet 2.0 [23] for household and adversarial objects, respectively. The performance reported for the work is statically more meaningful as the sample size is 8 times more as compared to [25]. Meanwhile, we can also notice that GR-ConvNet reaches the best grasp success rate in cluttered scenarios.

5.7 Failure case analysis

In the experimental results, there are only a few cases that can be accounted for as failures. Of them, the objects that had extremely low grasp scores and those that slipped from the gripper in spite of the gripper being closed were the most common ones. This could be attributed to the inaccurate depth information coming from the camera and the gripper misalignment due to collision between

the gripper and nearby objects.

Another case where the model was unable to produce a good grasp was for a transparent bottle as shown in Fig. 5.12(d). This could be due to inaccurate depth data captured by the camera due to possible object reflections. However, by combining depth data along with RGB data, the model was still able to generate a fairly good grasp for the transparent objects.

Chapter 6

Learning Multi-step Robotic Manipulation Tasks

Robotic manipulation tasks have been the backbone of most industrial robotic applications, e.g. bin picking, assembly, palletizing, or machine tending operations. In structured scenarios, these tasks have been reliably performed by the methods used in the existing work [116, 115]. Although, in unstructured scenarios, simple tasks, such as pick-only tasks, have been successfully performed using grasping approaches such as [125, 5, 152], complex tasks that involve multiple steps, such as clearing a bin of mixed items and creating a stack of multiple objects, remain a challenge.

Training end-to-end manipulation policies that map directly from image pixels to joint velocities can be computationally expensive and time exhaustive due to a large volume of sample space and can be difficult to adapt on physical setups [128, 87, 153, 154, 155]. To solve this, many have tried pixel-wise parameterization of both state and action spaces, which enables the use of a neural network as an approximator of Q-function [12, 156, 157]. However, these approaches have a low success rate, a long learning time, and cannot handle complex tasks consisting of multiple steps and long horizons.

Recent developments in deep reinforcement learning have shown promising results in several primitive tasks such as grasping, pushing, and pulling [158, 128, 87]. However, these approaches long learning time and cannot handle complex tasks consisting of multiple steps and long horizon. To solve this, many have tried model-free deep reinforcement learning based approaches that learn

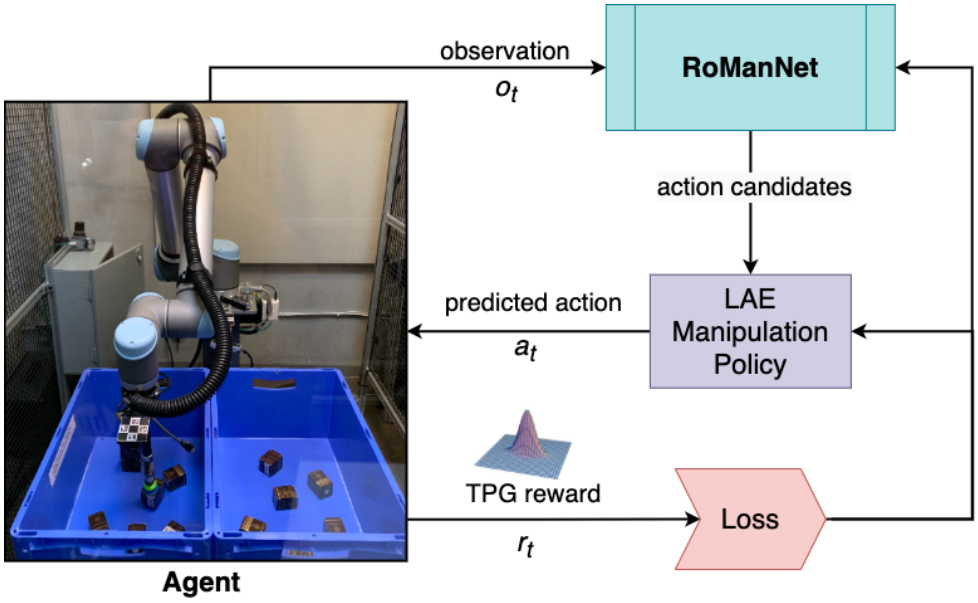


Figure 6.1: Proposed approach for training a vision-based deep reinforcement learning agent for efficiently learning multi-step manipulation tasks.

the coordinated behavior between intermediate actions and its consequences towards the advancement of an overall task goal [12, 156, 157].

In this chapter, a model-free deep reinforcement learning approach (shown in Fig. 6.1) to produce a deterministic policy that allows complex robot manipulation tasks to be effectively learned from pixel input is presented. The policy directs a low-level controller to perform motion primitives rather than regressing motor torque vectors directly by learning a pixel-wise action success likelihood map. An end-to-end model architecture RoManNet is introduced to efficiently learn the action-value functions and generate accurate action candidates from visual observation of the scene. A Task Progress based Gaussian (TPG) reward function is proposed to learn the coordinated behavior between intermediate actions and their consequences towards the advancement of an overall task goal. TPG reward function uses a sub-task indicator function and an overall task progress function to compute the reward for each action in a multi-step manipulation task. The challenge of balancing the ratio of exploration/exploitation is addressed by introducing a LAE manipulation policy that selects actions according to the Boltzmann distribution of loss estimates. Proposed LAE policy explores the action space and exploits the knowledge, which helps in reducing the learning time and improving the action efficiency.

The effectiveness of the proposed approach is demonstrated in simulation as well as in the real-world setting by training an agent to learn three vision-based multi-step robotic manipulation tasks. RoManNet trained with TPG reward and LAE policy performed significantly better than previous methods with only 2000 iterations in the real-world setting. A pick success rate of 92% for a mixed item bin-picking task and 84% action efficiency for a block-stacking task is observed.

6.1 Related Work

Robotic manipulation has always been an essential part of research in the field of robotics. There has been significant progress in recent years in robotic grasping that leverages deep learning and computer vision for generating grasp candidates for specific tasks that can be used to select suitable grasp poses for novel objects in fairly structured environments [159, 11, 3]. While most prior manipulation methods focus on singular tasks, this work focuses on learning multi-step manipulation tasks which generalize to a wide range of objects and tasks.

Learning based robotic grasping has been studied for the last decade and there has been a rise of deep learning-based approaches to tackle the problem of grasping novel objects [159, 11, 3]. Convolutional neural network-based approaches such as Grasp Quality Convolutional Neural Network (GQ-CNN) proposed by Mahler *et al.* predicts grasps from synthetic point cloud data trained on the Dex-Net dataset [160]. Levine *et al.* used learning-based hand-eye coordination by incorporating a CNN-based deep learning framework and continuous visual servoing for grasp success prediction [128]. Kumra *et al.* proposed a multi-modal deep learning-based architecture where a deep CNN extracts features from the scene and a shallow CNN predicts grasp configurations [5]. The paper demonstrates that a deeper network along with residual layers is more efficient at learning features. However, these approaches are computationally expensive. [3] introduced a generative residual convolutional neural network (GR-ConvNet) that not only predicts robust grasp poses but is also computationally less expensive. In this work, a variant of this network is used as the action pose generator.

Deep reinforcement learning can be used to learn complex robotic manipulation tasks by using model-free deep policies. Kalashnikov *et al.* demonstrate this by proposing a QT-opt technique to provide a scalable approach for vision-based robotic manipulation applications [87]. Riedmiller

et al. introduced a SAC-X method that learns complex tasks from scratch with the help of multiple sparse rewards where only the end goal is specified [161]. The agent learned these tasks by exploring its observation space and the results showed that this technique was highly reliable and robust. More recently, Xu *et al.* introduced a learning-to-plan method called Deep Affordance Foresight (DAF), which learns partial environment models of affordances of parameterized motor skills through trial-and-error [162]. Nematollahi *et al.* demonstrated that scene dynamics in the real-world can be learned for visuomotor control and planning [163].

Learning multi-step tasks with sparse rewards is particularly challenging because a solution is improbable through random exploration. Tasks such as clearing a bin with multiple objects [26, 87] do not include long-horizon and the likelihood of reverse progress is out of consideration. Many propose using a model-free method through self-supervised learning. One such method proposed by Zeng *et al.* uses a VPG framework that can discover and learn to push and grasp through model-free deep Q learning [12]. Similarly, Jeong *et al.* performed a stacking task by placing a cube over another cube using a two-stage self-supervised domain adaptation (SSDA) technique [164]. Zhu *et al.* presented a framework in which manipulation tasks were learned by using a deep visuomotor policy (DVP) that uses a combination of reinforcement learning and imitation learning to map RGB camera inputs directly into joint velocities [165]. Hundt *et al.* developed the Schedule for Positive Task (SPOT) framework [156], that explores actions within the safety zones and can identify unsafe regions even without exploring and can prioritize its experience to only learn what is useful. Zeng *et al.* proposed a Transporter Network trained using learning from demonstrations, which rearranges deep features to infer spatial displacements from visual input [15]. Kase *et al.* proposed a Deep Planning Domain Learning (DPDL) framework which learns a high-level model using sensor data to predict values for a large set of logical predicates consisting of the current symbolic world state and separately learns a low-level policy which translates symbolic operators into robust executable actions on a robot [166]. DPDL framework worked well on manipulation tasks in a photorealistic kitchen scenario. Similarly, Driess *et al.* proposed a network architecture consisting of a high-level reasoning network, an adaptation network, and low-level controllers to learn geometrically precise manipulation tasks for a dual robot arm system where the parameters of early actions are tightly coupled with those of later actions [167]. Kit assembly [168, 169], and cloth manipulation [170, 171] are some other tasks that involve multiple steps.

This work focuses on multi-step tasks that involve long-horizon planning and considers progress reversal. More closely related to this work is the VPG framework introduced by Zeng *et al.* which utilized a Fully Convolutional Network (FCN) as a function approximator to estimate the action-value function. However, this method had a low success rate and was sample inefficient. A sample efficient RoManNet framework is introduced to learn multi-step manipulation tasks, which is trained using a TPG reward function and LAE policy to address these problems. In addition to the grasp and pre-grasp primitives explored by Zeng *et al.* , the placement primitive is explored to evaluate the proposed approach on multi-step manipulation tasks like building a stack of blocks.

6.2 Problem Formulation

The problem of efficiently learning multi-step robotic manipulation for unknown objects in an environment with unknown dynamics is considered. Each manipulation task can be formulated as a Markov decision process (MDP) where at any given state $s_t \in \mathcal{S}$ at time t, the robot makes an observation $o_t \in \mathcal{O}$ of the environment and executes an action $a_t \in \mathcal{A}$ based on policy $\pi(s_t)$ and receives an immediate reward of $\mathcal{R}_{a_t}(s_t, s_{t+1})$. In the formulation for this work, $o_t \in \mathbb{R}^{4\times h\times w}$ is the visual observation of the robot workspace from RGB-D cameras, and the action space $\mathcal{A}$ is divided into two components: action type Φ and action pose Ψ. The underlying assumption is that the edges of o_t are the boundaries of the agent's workspace and o_t embeds all necessary state information, thus providing sufficient information of the environment to choose correct actions.

The goal is to find an optimal policy π^* in order to maximize the expected sum of future rewards i.e. γ-discounted sum on all future returns from time t to H, across planning horizon H. An off-policy Q-learning can be used to train a greedy deterministic policy $\pi(s_t)$ that chooses actions by maximizing the Q-function $Q^{\pi}(s_t, a_t)$, which estimates the expected reward $\mathbb{E}_{\pi}(s_{t+1}, a_t)$ of taking action a_t in state s_t at time t.

6.3 Approach

RoManNet and Previous Action Conditioned Robotic Manipulation Network (PAC-RoManNet) are introduced to approximate the action-value function Q_{μ}, which predict manipulation action candidates $\mathcal{A}_{s_t}$ from the observation o_t of the state s_t at time t. The proposed LAE manipulation policy

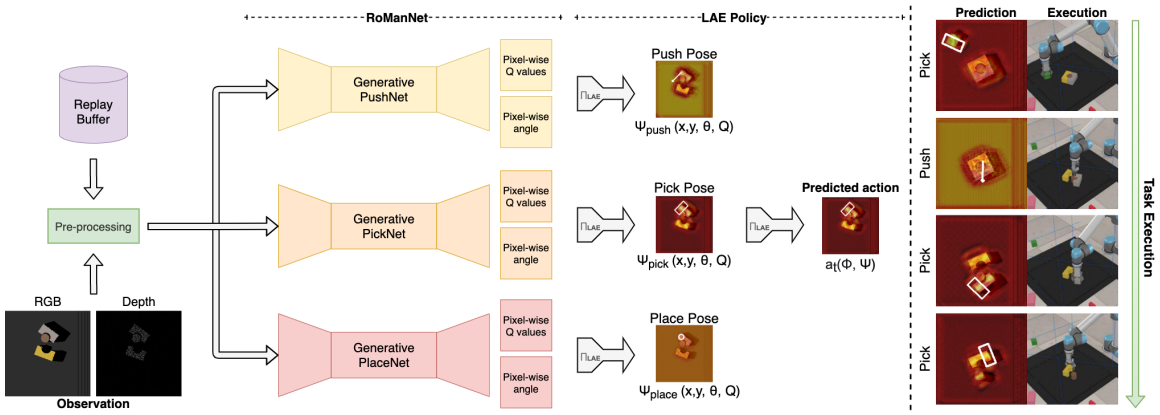


Figure 6.2: **Left**: Proposed RoManNet based framework for learning multi-step manipulation tasks. The pre-processed (cropped, resized, and normalized) inputs are fed into three generative networks which generate the action candidates. Each of these generative networks is a variant of GR-ConvNet. The LAE policy Π_{LAE} selects an action that maximizes the expected reward. **Right**: An illustration of an agent using the learned robot manipulation policy to execute a multi-step manipulation task, which requires the robot to clear a bin with challenging arrangement using pushing and picking actions. The robot can be seen pushing the item to de-clutter the tight arrangement before picking.

$\Pi_{LAE}(s_t, \mathcal{L}_t)$ determines the action a_t from the action candidates $\mathcal{A}_{s_t}$ that maximizes the reward $\mathcal{R}$. Once the agent executes a_t, the reward is computed using a TPG reward function. The parameters μ are updated by minimizing the loss function. An overview of the proposed learning framework is illustrated in Fig. 6.2.

6.3.1 Manipulation Action Space

Each manipulation action a_t is parameterized as two components: action type Φ, which consists of three high-level motion primitives $\{push, pick, place\}$, and action pose Ψ, which is defined by the pose at which the action is performed. Each manipulation action pose in the robot frame is defined as:

$$\Psi_r = (P, \theta_r, Q) \tag{6.1}$$

where, $P = (x, y, z)$ is the center position of the gripper, θ_r is the rotation of the gripper around the z-axis, and Q is an affordance score that represents the 'quality' of action. The manipulation action

pose in image frame Ψ_i is parameterized pixel-wise and defined as:

$$\Psi_i = (x, y, \theta_i, Q) \tag{6.2}$$

where (x, y) is the center of action pose in image coordinates, θ_i is the rotation in the image frame, and Q is the same affordance score as in equation (6.1).

The high-level motion primitive behaviors Φ are defined as follows:

- **Pushing:** $\Psi_{push} = (x, y, \theta, Q)$ denotes the starting pose of a 10cm push. A push starts with the gripper closed at (x, y) and moves horizontally at a fixed distance along angle θ.
- **Picking:** $\Psi_{pick} = (x, y, \theta, Q)$ denotes the middle position of a top-down grasp. During a pick attempt, both fingers attempt to move 3cm below Ψ_{pick} (in the $-z$ direction) before closing the fingers.
- **Placing:** $\Psi_{place} = (x, y, \theta, Q)$ denotes the middle position of a top-down placement. During a place attempt, both fingers open when the place pose Ψ_{place} is reached.

6.3.2 Learning the Action-Value Functions

The proposed algorithm to learn the action-value function is described in algorithm 1. Two different neural network architectures, RoManNet and PAC-RoManNet, are proposed to approximate the action-value function. In contrast to the DenseNet-121 fully connected networks as in [12, 156], both neural network architectures are built using three lightweight GR-ConvNet models PushNet, PickNet, and PlaceNet, one for each motion primitive behavior (pushing, picking, and placing respectively). In RoManNet architecture, each individual GR-ConvNet model takes as input the image representation $o_t \in \mathbb{R}^{4\times h\times w}$ of the state s_t and generates a pixel-wise map of Q values $\mathcal{Q}_\Phi(s_t, a_t) \in \mathbb{R}^{h\times w}$ and the corresponding rotation angle for each pixel $\theta_\Phi(s_t, a_t) \in \mathbb{R}^{h\times w}$ with the same image size and resolution as o_t. Whereas in PAC-RoManNet each individual GR-ConvNet model takes as input the image representation o_t of the state s_t and Q value prediction of previous action $a_{t-1}(\Phi, \Psi)$, and generates a pixel-wise map $Q(s_t, a_t) \in \mathbb{R}^{h\times w}$, where Q value prediction at each pixel represents the future expected reward $\mathbb{E}_\pi(s_{t+1}, a_t)$ of executing action $a_t(\Phi, \Psi)$. An overview of the proposed PAC-RoManNet based learning framework is illustrated in Fig. 6.3.

Algorithm 1 Learn action-value function $Q_\mu : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

Require: Manipulation MDP with states $\mathcal{S}$ and actions $\mathcal{A}$, Transition function $\mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$, and Reward function $\mathcal{R}_{tpg} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$

procedure LEARNING POLICY($\mathcal{S}$, $\mathcal{A}$, $\mathcal{R}$, $\mathcal{X}$, $\mathcal{P}$)

Initialize Q_μ with random weights

Initialize experience replay buffer $\mathcal{D}$

while Q_μ is not converged **do**

Initialize sub-task indicator function $\mathcal{X}$

Initialize overall task progress function $\mathcal{P}$

Start in state $s_t \in \mathcal{S}$

while s is not terminal **do**

Receive the observation o_t of the state s_t

Calculate $\Pi_{LAE}(s_t, \mathcal{L}_t)$ according to eq. 6.11

$a_t(\Phi, \Psi) \leftarrow \Pi_{LAE}(s_t, \mathcal{L}_t)$

Execute action on robot

Obtain $\mathcal{X}$ and $\mathcal{P}$

$r_t \leftarrow \mathcal{R}_{tpg}(s_{t+1}, a_t)$ according to eq. 6.8

Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$

$s_t \leftarrow s_{t+1}$ according to $\mathcal{T}$

Sample mini-batch from $\mathcal{D}$

$Y_t \leftarrow \mathbb{E}_\pi(s_{t+1}, a_t)$ according to eq. 6.3

Update Q_μ minimising the loss $\mathcal{L}_t$ in eq. 6.4

return Q_μ

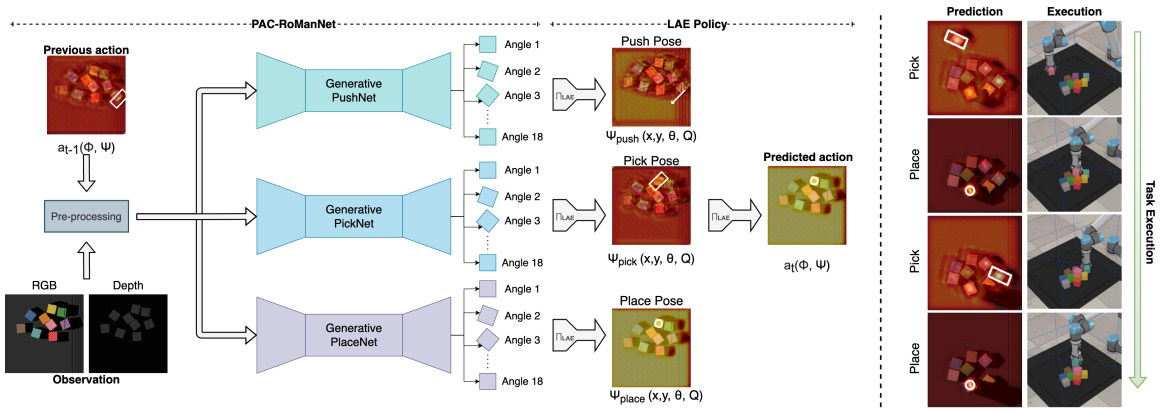


Figure 6.3: **Left**: Proposed Previous Action Conditioned Robotic Manipulation Network based framework for learning action-value function to predict manipulation action candidates from the observation of the state and Q value prediction of previous action. The LAE policy selects an action that maximizes the expected reward. **Right**: An illustration of an agent using the learned policy to execute a multi-step manipulation task, which requires the robot to create a stack of blocks with a goal stack height of 4. We can see the robot performing consecutive pick and place actions to build a stack using 10 cubes randomly placed in the bin. The robot learns not to pick blocks from the stack being built as it leads to progress reversal and thus a lower reward.

In this work, h and w are same and $\mathbb{R}^{h \times w}$ is represented as an image of resolution 224×224. The overall Q-function $Q_\mu(s_t, a_t)$ selects the action a_t that maximizes the Q-value over pixel-wise maps for all manipulation action poses: $\operatorname{argmax} Q_\mu(s_t, a_t) = \operatorname{argmax} \left(\mathcal{Q}_{push}(s_t), \mathcal{Q}_{pick}(s_t), \mathcal{Q}_{place}(s_t)\right)$, which gives the center position (x, y) of the action in image coordinates and corresponding pixel in the rotation angle pixel-wise map $\theta_\Phi(s_t, a_t)$ gives the rotation of the gripper θ_i.

The RoManNet is continuously trained to approximate the optimal policy with prioritized experience replay using stochastic rank-based prioritization and leverages future knowledge via a recursively defined expected reward function:

$$\mathbb{E}_\pi(s_{t+1}, a_t) = \mathcal{R}(s_{t+1}, a_t) + \eta(\gamma \mathcal{R}(s_{t+2}, a_{t+1})) \tag{6.3}$$

where, $\gamma \in [0, 1)$ is the discount factor and η is a reward propagation factor. A value of γ closer to 0 indicates that the agent will choose actions based only on the current reward and a value approaching 1 indicates that the agent will choose actions based on the long-term reward. The reward propagation factor η ensures that future rewards only propagate across time steps where subtasks are completed successfully. The value of η at time step t equals to 1 if $\mathcal{R}(s_{t+1}, a_t) > 0$, and 0 otherwise.

The loss of the network is computed using the Huber loss function at each time step t as:

$$\mathcal{L}_t = \begin{cases} \frac{1}{2}\delta(t)^2 & \text{for } |\delta(t)| \leq 1, \\ \left(|\delta(t)| - \frac{1}{2}\right) & \text{otherwise.} \end{cases} \tag{6.4}$$

where $\delta(t) = Q_{\mu_t}(s_t, a_t) - Y_t$ is the TD-error and μ_t are the weights of RoManNet at time step t. The gradients are passed only through the network from which the predictions of the executed action a_t were computed. For an antipodal gripper, the pick and place motion primitives are symmetrical around 0-180° , thus the gradients for the opposite orientation can be passed as well. The models are trained using Stochastic Gradient Descent with a fixed learning rate of 10^{-4}, a momentum of 0.9, and weight decay of 2^{-5}. The models are continuously trained on previous trials using a prioritized experience replay with future discount $\gamma = 0.5$.

6.3.3 Task Progress based Gaussian Reward

The reward function $\mathcal{R}(s_{t+1}, a_t) \in \mathbb{R}^{h \times w}$ operates on two principles: actions that advance overall task progress receive a reward proportional to the quantity of progress, but actions that reverse progress receive a reward of 0. The task progress is measured using: (i) a sub-task indicator function $\mathcal{X}(s_{t+1}, a_t)$, which equals to 1 if a_t leads to a successful primitive action and 0 otherwise, and (ii) an overall task progress function $\mathcal{P}(s_{t+1}, a_t) \in [0, 1]$, which is proportional to the progress towards an overall goal. The task progress based reward function is defined as:

$$\mathcal{R}_{tp}(s_{t+1}, a_t) = \mathcal{W}(\Phi)\mathcal{X}(s_{t+1}, a_t)\mathcal{P}(s_{t+1}, a_t) \tag{6.5}$$

where $\mathcal{W}(\Phi)$ is a weighting function that depends on the primitive motion action type Φ. In this work, $\mathcal{W}(\Phi)$ is set to 0.2 for pushing action, and 1 for picking and placing actions. In the experiments, to compute the sub-task indicator $\mathcal{X}(s_{t+1}, a_t)$, a push action is successful if it perturbs an object, a pick action is successful if an object is grasped and raised from the surface, and a place action is successful only if it increases the stack height. The overall task progress function $\mathcal{P}(s_{t+1}, a_t)$ is computed using the depth image. For item removal related tasks, it is the ratio of number of occupied pixels to the total number of pixels, and for the stacking task it is proportional to the height of the stack.

The task progress based reward function $\mathcal{R}_{tp}$ is a single pixel in $\mathbb{R}^{h \times w}$ and to make the reward function more robust, it is smoothed using an anisotropic Gaussian distribution [172] parameterized with standard deviations σ_x and σ_y. An anisotropy ratio of 2 (i.e., $\sigma_x/\sigma_y = 2$) is used, where x and y are the axis of the anisotropic Gaussian distribution and x is aligned with the x-axis of the gripper. The proposed TPG reward function is specified as follows:

$$G(x, y, \sigma_x, \sigma_y) = \frac{1}{2\pi\sigma_x\sigma_y} e^{-\left(\frac{x^2}{2\sigma_x^2} + \frac{y^2}{2\sigma_y^2}\right)} \tag{6.6}$$

$$\mathcal{R}_g(s_{t+1}, a_t) = \mathcal{R}_{tp}(s_{t+1}, a_t) \circledast G(x, y, \sigma_x, \sigma_y) \tag{6.7}$$

$$\mathcal{R}_{tpg}(s_{t+1}, a_t) = \max(\mathcal{R}_{tp}(s_{t+1}, a_t), \mathcal{R}_g(s_{t+1}, a_t)) \tag{6.8}$$

where $\circledast$ is the convolution operator, and G is the anisotropic Gaussian filter applied to $\mathcal{R}_{tp}$. The intuition here is that action poses in the local neighbourhood should yield similar reward. The advantages of using $\mathcal{R}_{tpg}$ as compared to $\mathcal{R}_{tp}$ are experimentally shown in section 6.4.5.

6.3.4 Loss Adjusted Exploration Policy

A LAE policy is introduced to reduce unnecessary exploration once knowledge about initial states has been sufficiently established. The LAE policy extends the ϵ-greedy policy similar to the Value-Difference based Exploration [173], and eliminates the need for hand-tuning the exploration rate ϵ as in [12, 156]. The update steps for loss dependent exploration probability $\mathcal{E}(\mathcal{L}_t)$ is defined as the following Boltzmann distribution of loss estimates:

$$f(\mathcal{L}_t, \sigma) = \frac{1 - e^{(\frac{-|\alpha \cdot \mathcal{L}_t|}{\sigma})}}{1 + e^{(\frac{-|\alpha \cdot \mathcal{L}_t|}{\sigma})}} \tag{6.9}$$

$$\mathcal{E}_{t+1}(\mathcal{L}_t) = \beta \cdot f(\mathcal{L}_t, \sigma) + (1 - \beta) \cdot \mathcal{E}_t(\mathcal{L}_t) \tag{6.10}$$

where $\mathcal{L}_t$ is the loss of the network computed using equation (6.4) at each time step t, σ is a positive constant called inverse sensitivity, α is a constant set to 0.5 in this work, and $\beta \in [0, 1)$ is a parameter determining the influence of the selected action on the exploration rate. The LAE policy is defined

as:

$$\Pi_{LAE}(s_t, \mathcal{L}_t) = \begin{cases} \mathcal{U}(\mathcal{A}_{s_t}) & \text{if } \xi < \mathcal{E}(\mathcal{L}_t), \\ \underset{a_t \in \mathcal{A}(s_t)}{\operatorname{argmax}} Q_\mu(s_t, a_t) & \text{otherwise.} \end{cases} \tag{6.11}$$

where $\mathcal{U}$ is a uniform distribution over action candidates $\mathcal{A}_{s_t}$ and $\xi \in [0, 1)$ is a random number drawn at each time step t from a uniform distribution.

6.4 Experiments

The experiments are conducted in both simulated and real settings to evaluate the proposed method across various tasks. The experiments are designed to investigate the following three questions:

- How well does the proposed method perform on different multi-step manipulation tasks?
- Does he proposed method improve the task performance compared to the baseline methods?
- What are the effects of the individual components of the proposedframework in solving multi-step manipulation tasks, i.e. without LAE or TPG reward?

6.4.1 Experimental Setups

For the simulated environment, the CoppeliaSim based setup used in [12] is extended to provide a consistent environment for fair comparisons and ablations. The environment simulates the agent using a UR5 robot arm with an RG2 gripper. Bullet Physics 2.83 is used to simulate the dynamics and CoppeliaSim's internal inverse kinematics module for robot motion planning. A statically mounted perspective 3D camera is simulated in the environment to capture the observations of the states. The color and depth images of size 640×480 are rendered with OpenGL using the simulated camera, without any noise models for depth or color.

For the real-world setup, a UR10 robot with a Piab suction cup as the EOAT is used. The RGB-D images of the scene were captured using an Intel RealSense D415 camera rigidly mounted above the workspace. The camera is localized with respect to the robot base by an automatic calibration procedure (similar to the one discussed in section 5.6.3), during which the camera tracks the location of a checkerboard pattern taped onto the gripper. The calibration optimizes for extrinsics as the robot

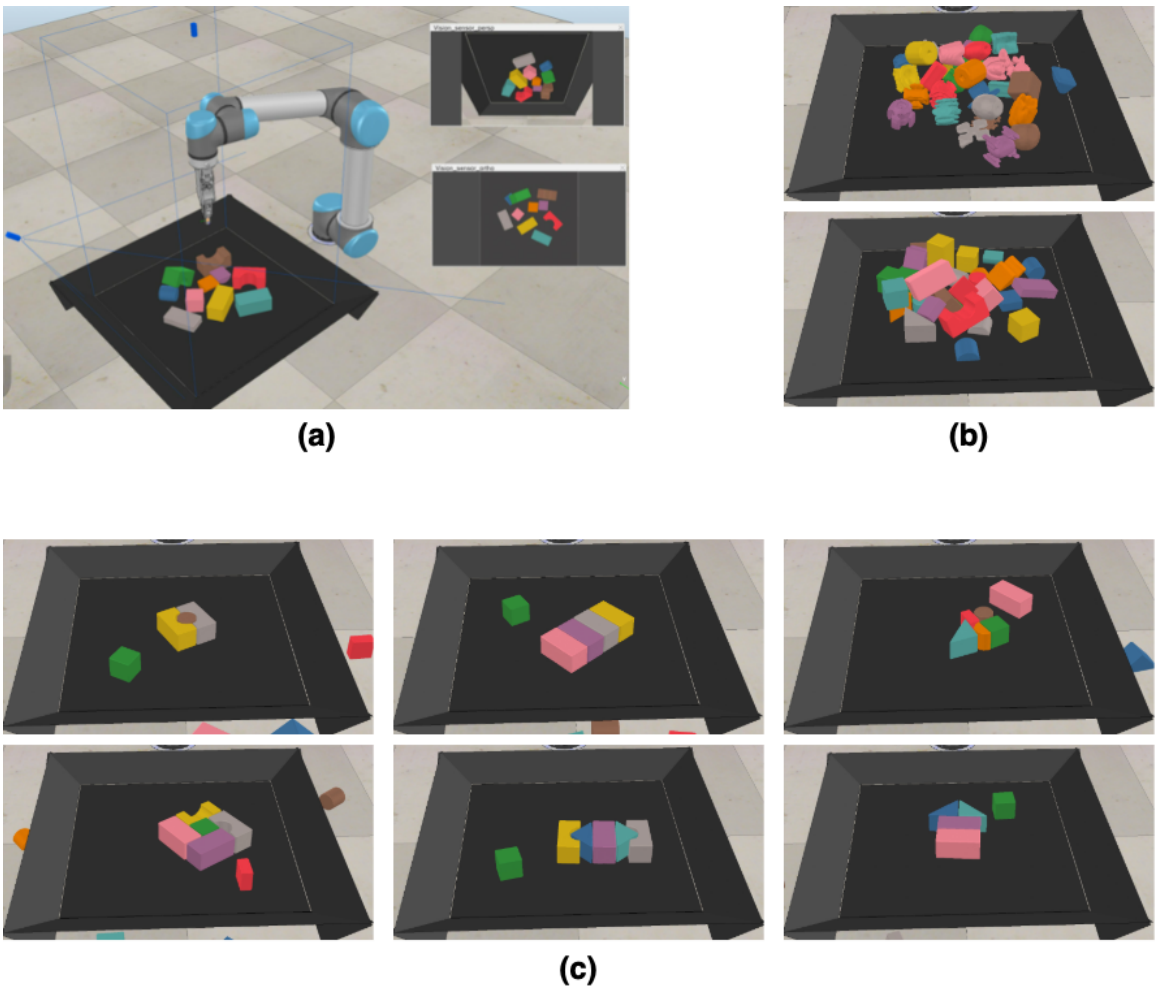


Figure 6.4: Setup for simulation experiments. (a) Trainig setup for bin clear task. (b) Dense clutter of objects using objects from VPG [12] and EGAD [14]. (c) Six examples of manually engineered test cases to reflect challenging real-world picking scenarios used in VPG [12].

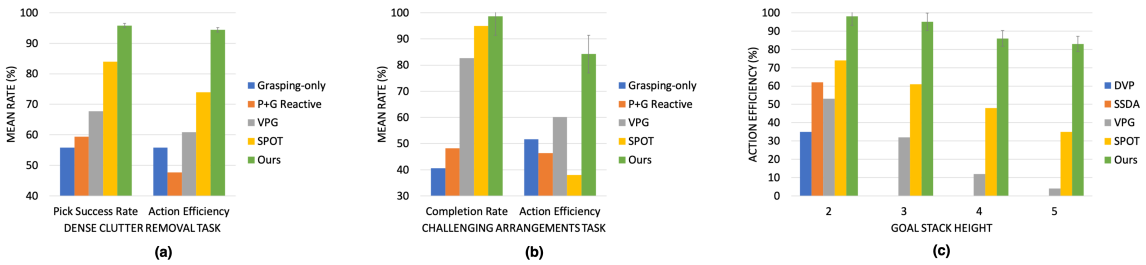


Figure 6.5: Quantitative comparisons with prior work for various tasks in similar simulation setups. Results for DVP and SSDA methods are borrowed from respective papers. All other results are reproduced in simulation using the open source code and models. (a) Performance for Dense Clutter Removal Task in terms of pick success rate and action efficiency. (b) Performance for Challenging Arrangements in terms of mean completion rate and action efficiency. (c) Performance for block stacking task in terms of mean action efficiency for various goal stack heights.

moves the gripper over a grid of 3D locations (predefined with respect to robot coordinates) within the camera field of view. For all experiments, the networks are trained from scratch, without any pre-training from vision datasets.

6.4.2 Evaluation Metrics

For evaluating the trained model, the policy is greedy deterministic and the model weights are reset to the trained weights at the start of each new test run. For each of the test cases, 30 runs with new random seeds are executed and the performance is evaluated with the following metrics found in [12, 156]:

- Completion rate: the average percentage of runs in which the policy completed the given task without 10 consecutive failed attempts.
- Pick success rate: the average percentage of object picking success per completion.
- Action efficiency: a ratio of the ideal to the actual number of actions taken to complete the given task.

6.4.3 Simulation Experiments

Three manipulation tasks are designed to evaluate the proposed method in simulation: dense clutter removal, clearing 11 challenging test cases with adversarial objects, and stacking multiple objects.

Table 6.1: Performance for Dense Clutter Removal Task (Mean %)

Approach	Completion Rate	Pick Success	Action Efficiency
Grasping-only [126]	90.9	55.8	55.8
P+G Reactive [12]	54.5	59.4	47.7
VPG [12]	100	67.7	60.9
SPOT [156]	100	84	74
PAC-RoManNet	**100**	**96.2**	**94.6**

All tasks share the same MDP formulation in section 6.2, while the object set, manipulation action space, and the reward function are different for each task. Three sets of objects are used in these tasks: (i) 9 different 3D toy blocks (same as VPG [12]), (ii) 49 diverse evaluation objects from EGAD [14], and (iii) cubes only for stacking task. To train the models for all tasks, objects with randomly selected shapes and colors are dropped in front of the robot at the start of each experiment. The robot then automatically performs data collection by trial and error. For object removal-related tasks in simulation, objects are removed from the scene after a successful pick action. The environment is reset at the termination of the task. Quantitative results for simulation experiments are summarized in Fig. 6.5.

Dense Clutter Removal Task

The proposed method is first evaluated in simulated environment where 30 objects are randomly dropped in a bin. The goal of the agent is to remove all objects from the bin in front of the robot by executing pushing or picking actions. The agent is trained with only 10 objects instead of 30 objects. This helps to test the generalization of policies to more cluttered scenarios. The first experiment compares the proposed method to previous methods in simulation using the adversarial objects from [12]. Comparison of the results with Grasping-only [126], P+G Reactive [12], VPG [12] and SPOT [156] are summarized in Table 6.1. We see that with a pick success rate of 95.8%, the proposed method outperforms all other methods across all metrics. It was observed that the proposed method learned to pick items from dense clutter without having to declutter them before picking. It is hypothesized that this led to the high action efficiency of 94.6%. The second experiment was with

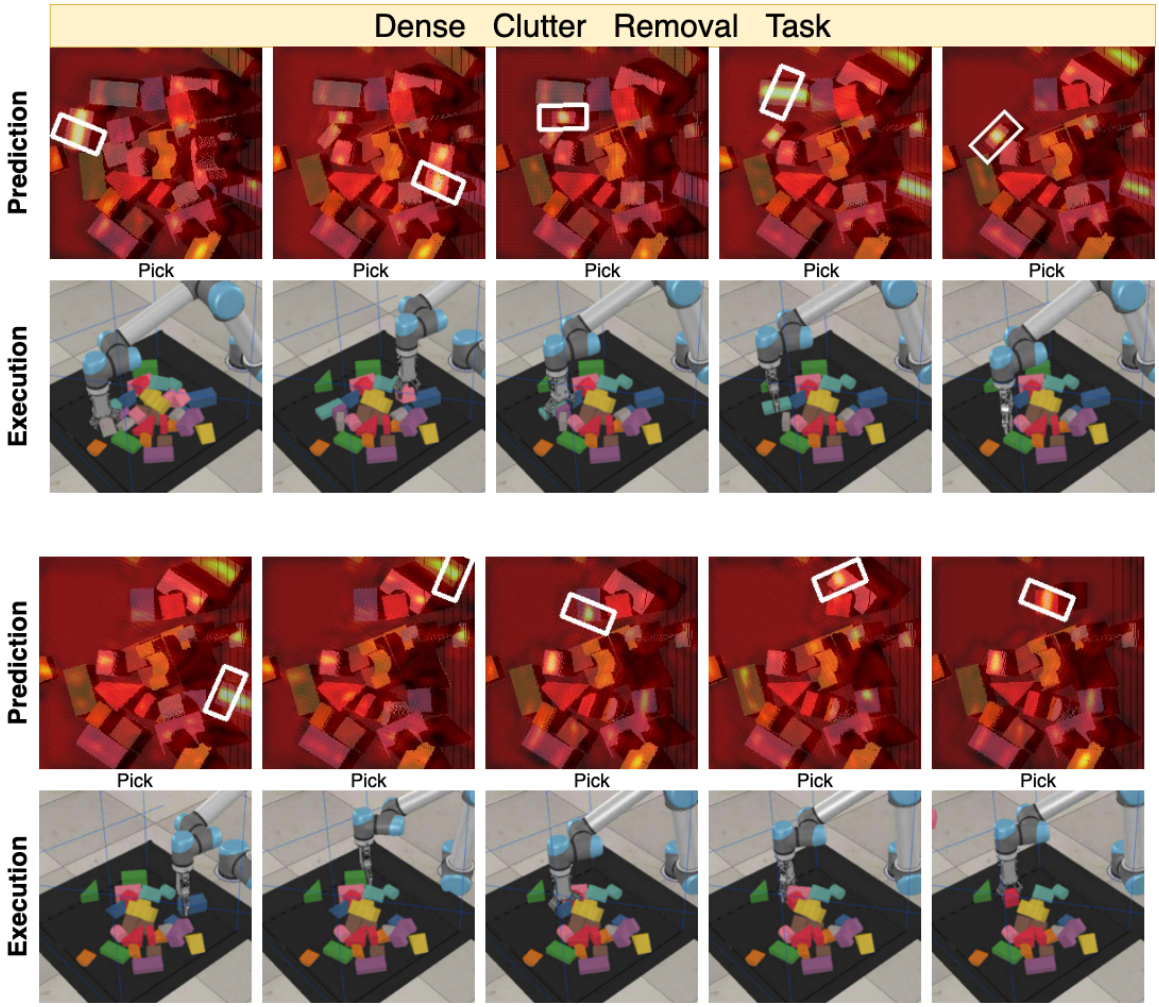


Figure 6.6: Visualization of the dense clutter removal task being executed in simulation using the trained model. The robot can be seen picking objects from the clutter of 30 objects in a bin and removing them from the bin one after another. We observe that the robot first picks any objects that are away from the clutter.

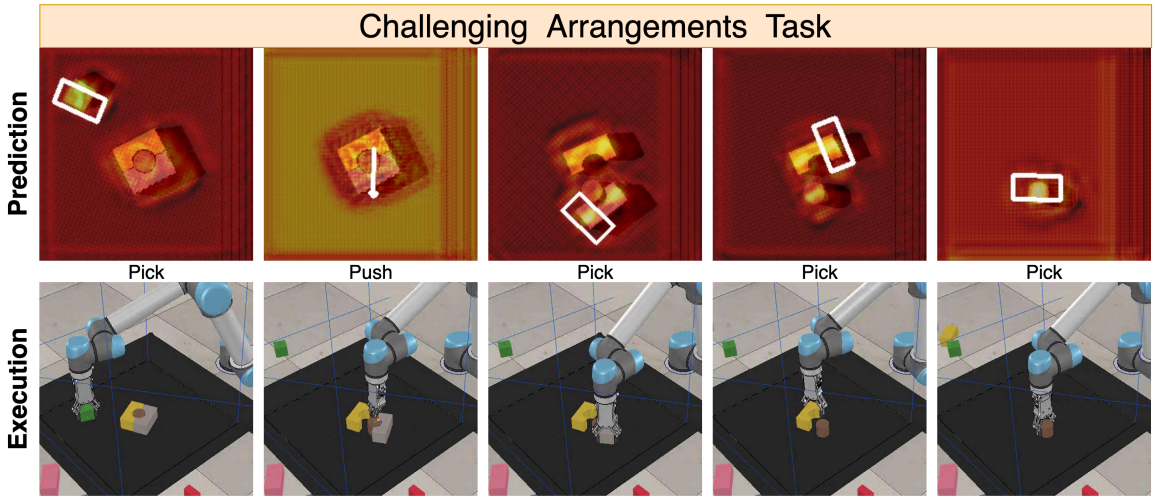


Figure 6.7: Visualization of one of the challenging arrangements task which has tightly packed objects being executed in simulation using the trained model. We observe that the robot first pushes the packed objects to de-clutter and then picks them up.

even more complex objects from the validation set of EGAD [14]. A high success rate of 92.2% is observed during testing on these complex objects, which suggests good generalizability of the proposed method.

Challenging Arrangements Task

The proposed method is also compared with other methods in simulation on 11 challenging test cases with adversarial clutter from [12]. These test cases are manually engineered to reflect challenging picking scenarios which allow us to assess the robustness of the model. Each test case

Table 6.2: Performance for Challenging Arrangements (Mean %)

Approach	**Cases 100% Completed**	**Completion Rate**	**Action Efficiency**
Grasp-only [126]	-	40.6	51.7
P+G Reactive [12]	-	48.2	46.4
VPG [12]	5/11	82.7	60.1
SPOT [156]	7/11	95	38
PAC-RoManNet	**10/11**	**98.8**	**89.4**

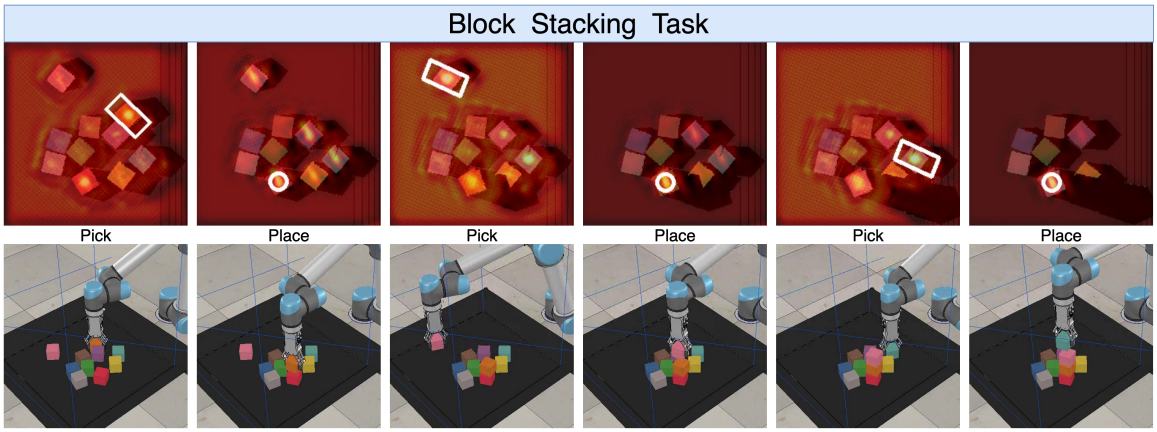


Figure 6.8: Visualization of the block stacking task with a goal stack height of 4 being executed in simulation using the trained model. We can see the robot performing consecutive pick and place actions to build a stack using 10 cubes randomly placed in the bin. The robot learns not to pick blocks from the stack being built as it leads to progress reversal and thus a lower reward.

consists of a unique configuration of 3 to 6 objects placed in a tightly packed arrangement that will be challenging even for an optimal picking-only policy as it will require de-cluttering them before picking. As a sanity check, a single isolated object is additionally placed separately from the main configuration. Similar to the dense clutter removal task, the policy is trained to learn push and pick actions with 10 objects randomly placed in a bin and tested on 11 challenging test cases. Performance comparison with previous work is shown in Table 6.2. Across the collection of test cases, we observe that the proposed method can successfully solve 10/11 cases with a 100% completion rate and an overall completion rate of 98.8%. Furthermore, the action efficiency of 89.4% with the proposed method indicates that it is significantly better than VPG [12] and SPOT [156] methods that attained action efficiencies of only 60.1% and 38% respectively.

Block Stacking Task

To truly evaluate the proposed multi-step task learning method, the task of stacking multiple blocks on top of each other is considered. This constitutes a challenging multi-step robotic task as it requires the agent to acquire several core abilities: picking a block from 10 blocks arbitrarily placed in the bin, precisely placing it on top of the second block, and repeating this process until a goal stack height is reached. Multiple experiments are performed with goal stack height in the range of 2 to 5. Table 6.3 summarizes the performance of the proposed method compared to previous

Table 6.3: Comparison of performance for block stacking task in terms of mean % action efficiency for various goal stack heights

Approach	Stack of 2	Stack of 3	Stack of 4	Stack of 5
DVP [164]	35	-	-	-
SSDA [12]	62	-	-	-
VPG [12]	53	32	12	4
SPOT [156]	74	61	48	35
PAC-RoManNet	**99**	**96**	**94**	**91**

work. Jeong *et al.* used self-supervised domain adaptation (SSDA) [164] and Zhu *et al.* used deep visuomotor policy (DVP) [165] and tested it with a goal stack height of 2. Hundt *et al.* used SPOT [156] and tested it with a goal stack height of 4. For this task with the highest complexity, the proposed approach seems to perform significantly better with an action efficiency of 99% and 94% for a goal stack height of 2 and 4, respectively. A possible reason is that, although the task is very complex, the TPG reward function helps learn optimal multi-step manipulation policy.

6.4.4 Real-World Experiments

The proposed method is validated in the real-world on two tasks: mixed item bin picking and block stacking. Fig. 6.9 and Fig. 6.10 illustrates the two tasks being executed by the UR10 robot using PAC-RoManNet trained for 2k iterations ($\sim$ 4 hours of robot run time).

Mixed Item Bin Picking

For the mixed item bin-picking task, the robot picks items from the source bin and places them into a destination bin until the source bin is empty, and then swaps the source and destination bin for the next run. 30 different items were used for training the model. To automate the training process, suction feedback is used to detect a successful pick and a binary classifier is used to detect if the bin is empty after each manipulation action. It is observed that the robot adopts diverse strategies to clear the clutter. The performance results in Table 6.4 shows that PAC-RoManNet performs consistently better than all previous methods. Pick success rate of 96% after training for only 2k iterations demonstrates the high sample efficiency of the proposed method. Generalization is a key

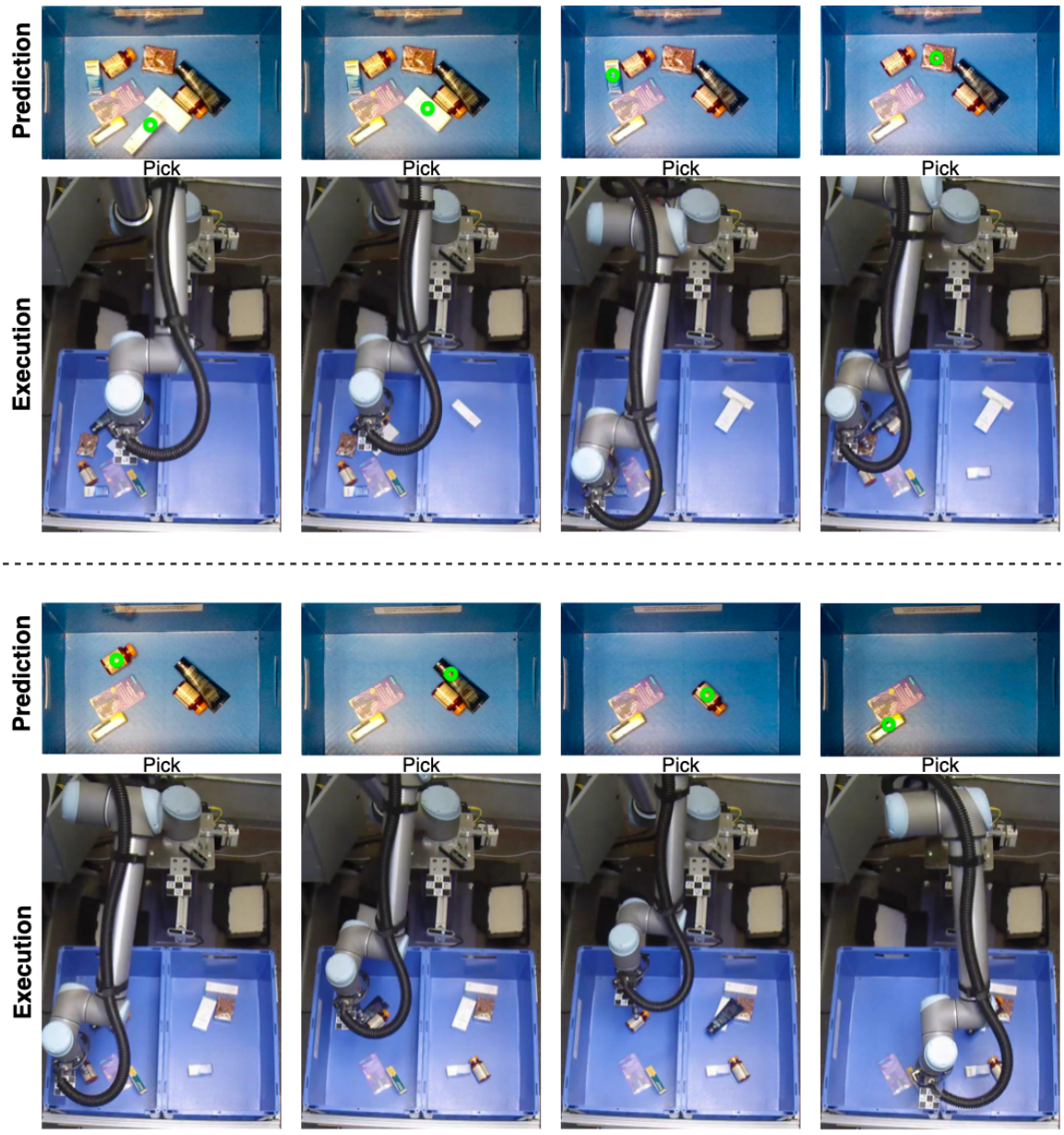


Figure 6.9: Illustration of mixed item bin picking task being executed by the UR10 robot using the model trained for approximately 4 hours in the real-world.

Table 6.4: Comparison of performance of state-of-the-art reinforcement learning based grasping methods in real-world settings

Approach	**Success %**	**Training Steps**	**Test Items**
VPG [12]	68	2.5k	20 seen
SPOT [156]	75	1k	20 seen
Levine *et al.* [128]	78	900k	-
QT-Opt [87]	88	580k	28 seen
Grasp-Q-Network [153]	89	7k	9 seen
Berscheid *et al.* [157]	92	27.5k	20 seen
PAC-RoManNet	**96** $\pm$ **1**	2k	20 seen
PAC-RoManNet	**93** $\pm$ **2**	2k	10 unseen

index for applications in industrial and logistics automation. The experiments with 10 unseen items show that the proposed method is extremely impressive as it can generalize to novel objects in a real-world setting and still achieve a success rate of 93%.

Real-World Block Stacking

The real-world block stacking task is similar to the one in simulation, where the robot performs manipulation actions until the goal stack height is reached. The depth measurements from the overhead camera are used to determine the current stack height. This task is particularly challenging as the agent needs to learn to align the position and orientation of the picked item with the stack during placement. Remarkably, 100% completion rate and 84% action efficiency is observed for a goal stack height of 4, outperforming the state-of-the-art action efficiency of 61% reported by Hundt *et al.* in [156].

6.4.5 Ablation Studies

In order to assess the necessity and efficacy of the different components of the proposed method, described in section 6.3, ablation experimental results are provided. Fig. 6.11(a) shows a comparison of the number of network parameters and the computation time for calculating the Q-values for a pushing and placing task of RoManNet with the network architectures used in VPG [12] and SPOT

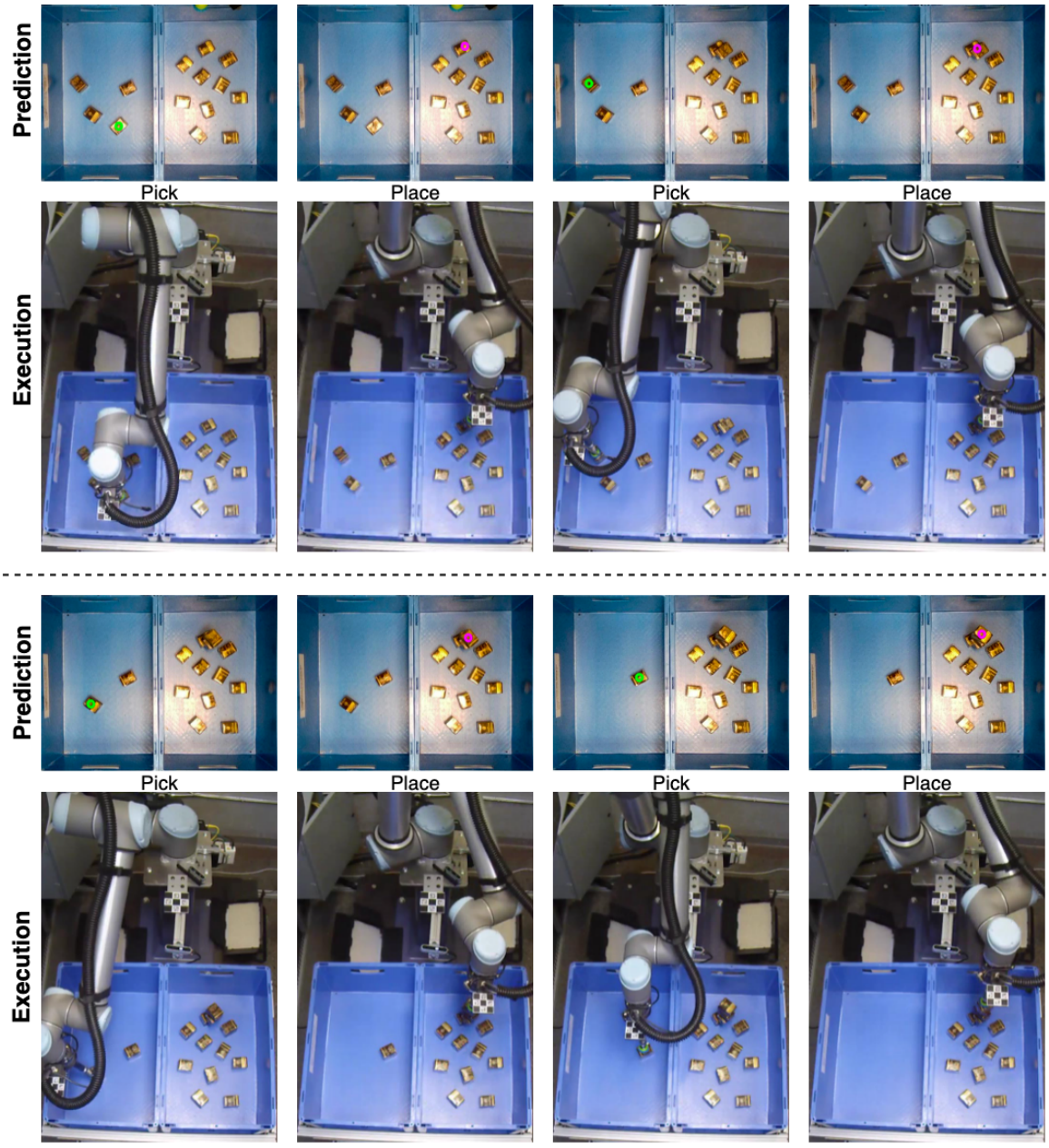


Figure 6.10: Illustration of block stacking task being executed by the UR10 robot using the model trained for approximately 4 hours in the real-world.

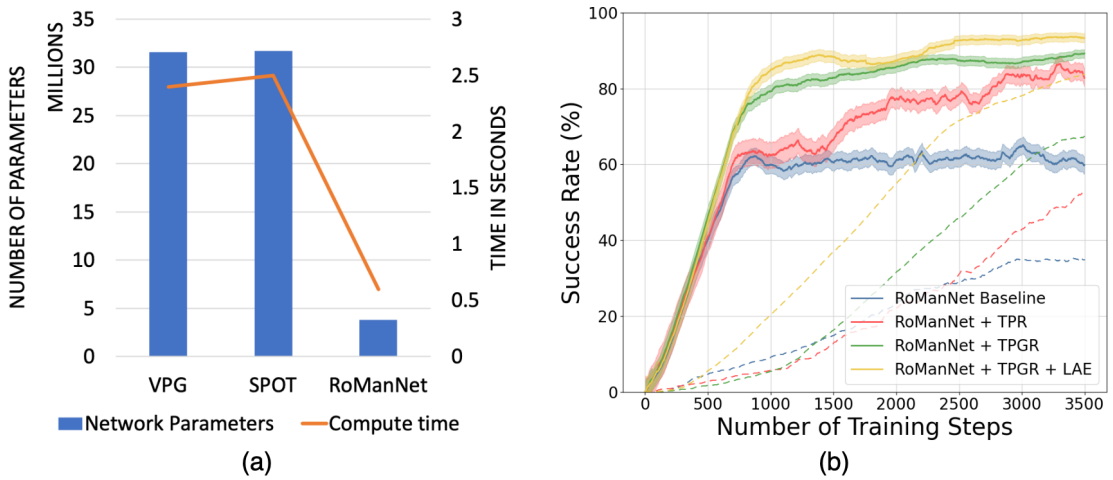


Figure 6.11: Ablation studies. (a) Comparison of network parameters and compute time of RoManNet with prior work (b) Learning curves for ablation of techniques used in conjunction with RoManNet for training agent on block stacking task. Solid lines indicate mean action success rates and dotted lines indicate mean action efficiency over training steps.

[156]. As RoManNet has significantly fewer network parameters, the compute time is significantly smaller (0.6 sec), making RoManNet more suitable for real-world manipulation tasks. Fig. 6.11(b) compares the learning curves for the underlying algorithm against the baseline approaches for the complex multi-step task of block stacking.

RoManNet Baseline

A baseline is established using the primary reward function found in VPG [12] and ϵ-greedy exploration policy. The baseline reward function is defined as function of sub-task indicator only:

$$\mathcal{R}_{base}(s_{t+1}, a_t) = \mathcal{X}(s_{t+1}, a_t) \tag{6.12}$$

The only case where this baseline model is on par with the full model is the clutter removal task, in which both the baseline and the full model achieved similar levels of performance. It is hypothesized that this is due to the short length of the task, where the task progress-based reward did not play a significant role.

RoManNet + TPG Reward

The second baseline is a combination of the TPG reward function and ϵ-greedy exploration policy. It is observed that TPG reward helped the agent learn to avoid task progress reversal for the multi-step block stacking task, thus improving the success rate by more than 25%. A significant improvement in action efficiency is also observed for the challenging arrangements and block stacking tasks. Moreover, it is observed that anisotropic gaussian smoothing improved manipulation action stability by removing maxima that were close to regions of low Q values. For example, RoManNet generated more stable action poses when trained with $\mathcal{R}_{tpg}$ as compared to $\mathcal{R}_{tp}$. As seen in Fig. 6.11(b), training with $\mathcal{R}_{tpg}$ reward function improved the action success rate by 9% compared to training with $\mathcal{R}_{tp}$ for block stacking task.

RoManNet + TPG Reward + LAE

For the complete proposed model, the proposed LAE policy is used instead of the ϵ-greedy exploration. An 11% improvement in action efficiency is observed for the block stacking task when RoManNet is trained with LAE policy instead of the baseline ϵ-greedy policy. This suggests that LAE policy improves the efficiency of exploration, which is critical in real-world applications as training for a large number of iterations on a physical robot is expensive.

6.4.6 Generalization

To demonstrate the generalizability of the proposed RoManNet architecture to various manipulation tasks, it is evaluated on the Ravens-10 benchmark tasks presented by Zeng *et al.* in [15]. The Ravens-10 benchmark consists of a wide variety of vision-based multi-step manipulation tasks like stacking a pyramid of blocks, manipulating deformable ropes, assembling kits with unseen objects, and pushing piles of small objects with closed-loop feedback. The 43-layer feed-forward residual networks for picking and placing is replaced by the proposed GR-ConvNet in the Transporter based framework [15] and the models are trained using behavior cloning.

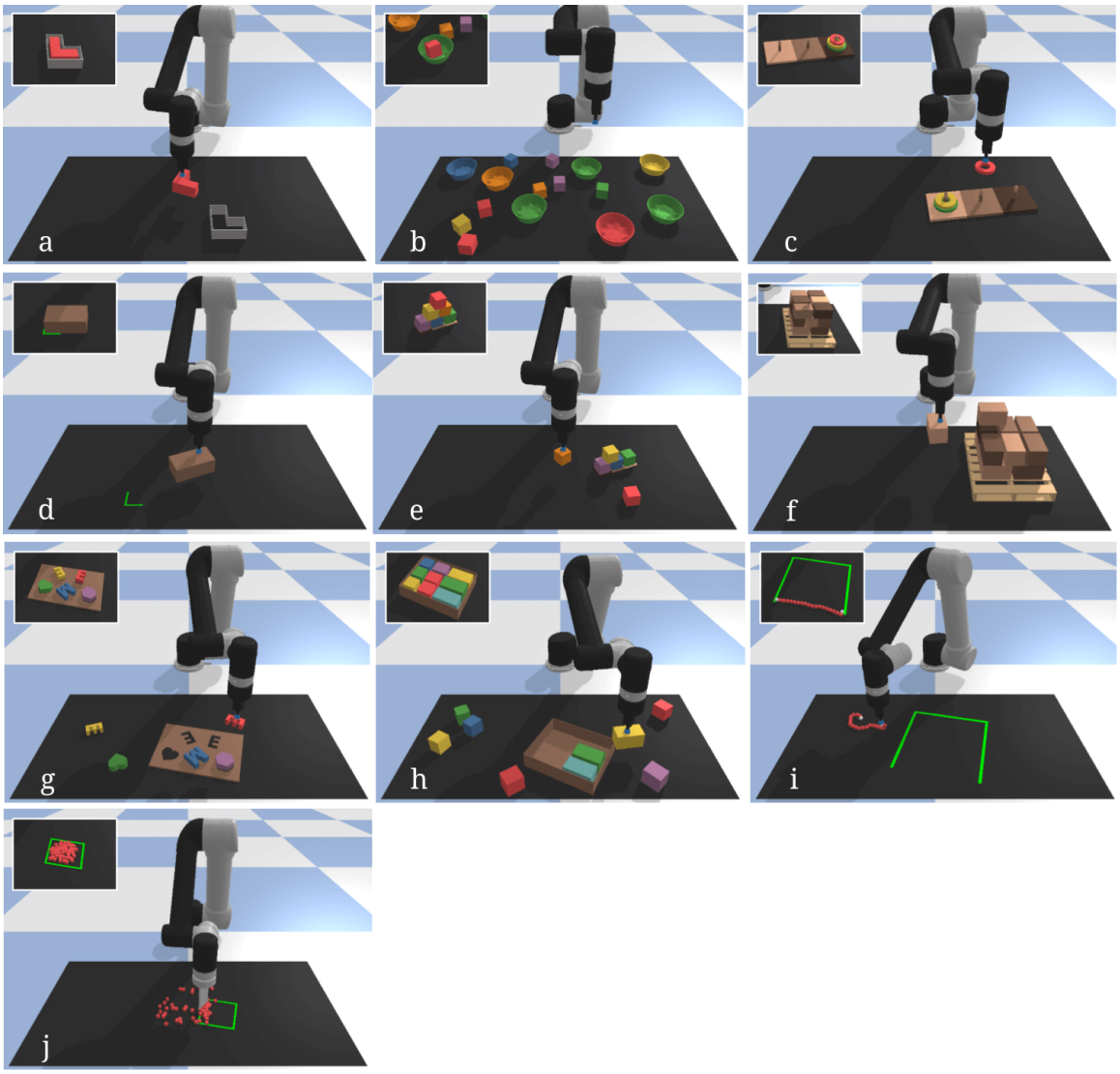


Figure 6.12: Set of 10 tasks in Ravens-10 benchmark [15].

Experimental Setup

Open-source simulation environment by Zeng *et al.* in [15] is used for fair comparison with baselines. The simulated environment is built with PyBullet [150], which consists of a UR5e robot with a suction gripper that overlooks the robot workspace with 3 RGB-D cameras pointing towards the workspace for improved visual coverage. For each task, objects are randomly spawned in the robot's workspace and the agent acts with motion primitives (pick, push, or place) parameterized by a sequence of two end effector poses. The task is completed when the agent receives a reward of 1 from the reward function that comes with each task. Partial reward is given during tasks for tasks that require multiple actions to be completed.

Examples of the Ravens-10 benchmark tasks are shown in Fig.6.12. The goal of each task is described as follows:

(a) block-insertion: pick up the L-shaped red block and place it into the L-shaped fixture.

(b) place-red-in-green: pick up the red blocks and place them into the green bowls amidst other objects.

(c) towers-of-hanoi: sequentially move disks from one tower to another—only smaller disks can be on top of larger ones.

(d) align-box-corner: pick up the randomly sized box and align one of its corners to the L-shaped marker on the tabletop.

(e) stack-block-pyramid: sequentially stack 6 blocks into a pyramid of 3-2-1 with rainbow colored ordering.

(f) palletizing-boxes: pick up homogeneous fixed-sized boxes and stack them in transposed layers on the pallet.

(g) assembling-kits: pick up different objects and arrange them on a board marked with corresponding silhouettes.

(h) packing-boxes: pick up randomly sized boxes and place them tightly into a container.

(i) manipulating-rope: rearrange a deformable rope such that it connects the two endpoints of a 3-sided square.

(j) sweeping-piles: push piles of small objects into a target goal zone marked on the tabletop.

During both training and testing, all objects (including target zones) are randomly positioned and oriented in the workspace. To succeed in the case where only a single demonstration is provided, the learner needs to be invariant to unseen configurations of objects. Information about the multi-modality of a task or its sequential permutations is only made available from the distribution of demonstrations. For example, moving disks in Towers of Hanoi may have only one correct sequence of states, but the distribution of how each disk can be picked or placed is learned from multiple demonstrations. Or, when palletizing homogeneous boxes, the arrangement of boxes across each layer on the pallet must be transposed, and boxes should be stacked stably on top other boxes already on the stack to avoid toppling

Results

The Ravens-10 benchmark tasks are difficult as most methods tend to over-fit to the training demonstration and generalize poorly with less than 100 demonstrations. The performance is evaluated using the same metric from 0 (failure) to 100 (success) as in [15]. For each task, the result averaged over 100 unseen test runs trained with 1, 10, 100 and 1000 demonstrations is reported. The performance results in Table 6.5 show that GR-ConvNet based Transporter framework can achieve state-of-the-art performance in terms of success rate on Ravens-10 benchmark tasks. While other methods require hundreds or thousands of demonstrations to achieve a task success rate of more than 90% for tasks such as *packing-boxes* and *sweeping-piles*, GR-ConvNet requires less than 1/10th of the number of demonstrations. This validates that the sampling efficiency of GR-ConvNet is extremely impressive when evaluated in unseen test settings. These results are consistent with the antipodal grasping experiments and demonstrate how GR-ConvNet generalizes across completely different manipulation tasks.

Table 6.5: GR-ConvNet performance on Ravens-10 benchmark tasks. Task success rate (mean %) vs demonstration used in training.

	align-box-corner				assembling-kits				block-insertion			
Approach	1	10	100	1000	1	10	100	1000	1	10	100	1000
GR-ConvNet	**62.0**	**91.0**	**100**	**100**	**56.8**	**83.2**	**99.4**	**99.6**	**100**	**100**	**100**	**100**
Transporter[15]	35.0	85.0	97.0	98.3	28.4	78.6	90.4	94.6	**100**	**100**	**100**	**100**
Form2Fit[168]	7.0	2.0	5.0	16.0	3.4	7.6	24.2	37.6	17.0	19.0	23.0	29.0
	palletizing-boxes				manipulating-rope				packing-boxes			
Approach	1	10	100	1000	1	10	100	1000	1	10	100	1000
GR-ConvNet	**84.2**	**98.2**	**100**	**100**	**25.7**	**87.0**	**99.9**	**100**	**96.1**	**99.9**	**99.9**	**100**
Transporter[15]	63.2	77.4	91.7	97.9	21.9	73.2	85.4	92.1	56.8	58.3	72.1	81.3
Form2Fit[168]	21.6	42.0	52.1	65.3	11.9	38.8	36.7	47.7	29.9	52.5	62.3	66.8
	place-red-in-green				stack-block-pyramid				sweeping-piles			
	1	10	100	1000	1	10	100	1000	1	10	100	1000
GR-ConvNet	**92.3**	**100**	**100**	**100**	**23.5**	**79.3**	**94.6**	**97.1**	**98.2**	**99.1**	**99.2**	**98.9**
Transporter[15]	84.5	**100**	**100**	**100**	13.3	42.6	56.2	78.2	52.4	74.4	71.5	96.1
Form2Fit[168]	83.4	**100**	**100**	**100**	19.7	17.5	18.5	32.5	13.2	15.6	26.7	38.4
	towers-of-hanoi											
	1	10	100	1000								
GR-ConvNet	**98.2**	**99.9**	**99.9**	**99.9**								
Transporter[15]	73.1	83.9	97.3	98.1								
Form2Fit[168]	3.6	4.4	3.7	7.0								

Chapter 7

Conclusion and Future Work

In this chapter, the conclusions and summary of this research is presented, along with the future direction of this research.

7.1 Conclusions

The research presented in this dissertation showed that a model-free deep reinforcement learning method can be used to effectively learn multi-step manipulation tasks.

Robotic Grasp Detection

In this work, a novel multi-modal robotic grasp detection system is presented that predicts the graspability of novel objects for a parallel plate robotic gripper using RGB-D images, along with a uni-modal model that uses RGB data only. Experiments showed that DCNNs can be used in parallel to extract features from multi-modal inputs and can be used to predict the grasp configuration for an object. It has been demonstrated that the use of deep residual layers helped extract better features from the input image, which were further used by the fully connected layers to output the grasp configuration. The proposed models improved the state-of-the-art performance on the CGD and run at real-time speeds. In future work, we would like to apply transfer learning concepts to use the trained model on the grasp dataset to perform grasps using a physical robot. Moreover, in an industrial setting, the detection accuracy can go even higher and can make grasp detection for pick and place related tasks robust to different shapes and sizes of parts.

Antipodal Robotic Grasping

A modular solution for grasping novel objects is presented using GR-ConvNet that uses n-channel input data to generate images that can be used to infer grasp rectangles for each pixel in an image. The lightweight nature of the proposed model makes it computationally less expensive and much faster compared to similar grasp prediction techniques. The GR-ConvNet is evaluated on three standard datasets, the CGD, the JGD and the G1BD, and obtained state-of-the-art results on all the datasets. Additionally, to test the robustness of the network, stricter IoU thresholds are used and obtained consistently outstanding results on all Jaccard thresholds for the CGD. Furthermore, ablation studies are performed to evaluate the effect of all individual parameters and components in the model. With the help of these experiments, we were able to identify the effect of adding dropout layers which further improved the performance of the network along with choosing the correct filter size (k) and examining the performance of the network on multiple input modalities.

The proposed system is also validated on novel real objects including household objects and adversarial objects in clutter by performing experiments using a robotic arm. The results demonstrate that the proposed system can predict and perform accurate grasps for previously unseen objects. In addition, the low inference time of the proposed model makes the system suitable for closed-loop robotic grasping. Furthermore, several experiments are performed on cluttered scene removal to show that the proposed system is capable of transferring in any industrial scenario and achieving exceptional results, even though the model was trained only on singular objects.

Learning Multi-step Manipulation Tasks

In this research, a vision-based deep reinforcement learning framework is presented to effectively learn complex manipulation tasks consisting of multiple steps and long-horizon planning. The experimental results indicate that the proposed TPG reward which computes reward based on the actions that lead to successful motion primitives and progress toward the overall task goal can successfully handle progress reversal in multi-step tasks. Moreover, it is shown that the proposed LAE policy can be used to curb unnecessary exploration which can occur after the initial states have already been explored. Compared to the previous work in multi-step manipulation, empirical results demonstrate that the proposed method outperformed all previous methods in various multi-step tasks in simulation as well as real-world settings. Several experiments for ablation studies

are also performed to further examine the effects of each component in the system. Finally, the high generalizability of RoManNet is demonstrated by showing that it can achieve state-of-the-art performance on Ravens-10 benchmark tasks.

7.2 Future Work

This thesis opens up a number of interesting directions in computer vision and robotics. In future work, this research can be extended to enhance other methods in broad contexts, such as reinforcement learning, imitation learning, meta-learning, and continuous learning.

In addition to the inference speed, using RGB-D images also simplifies the data, as it is in fewer dimensions and is easier to handle and modify. However, this increased speed also comes at a cost. The 2D representation of an object is flat compared to a point cloud. Therefore, the objects are only seen from one viewpoint, making it hard to determine the rotation of the gripper in space. Although GR-ConvNet was used to predict 4D grasps using RGB-D images in this work, it can potentially be extended to 6 DoF grasping in future work. The proposed GR-ConvNet model can also be used to explore manipulation tasks that require high precision. Another idea is to apply depth prediction techniques [174] to accurately predict depth for reflective objects, which can aid in improving the grasp prediction accuracy for reflective objects like the bottle as discussed in section 5.7.

In this work motion primitives are defined with parameters specified on a grid of pixels, which provides learning efficiency with DCNNs, but limits expressiveness. It would be interesting to explore other parameterizations that allow more expressive motions (without excessively inducing sample complexity), including more dynamic pushes, parallel rather than sequential combinations of pushing and picking action, and the use of more varied types of EOAT such as multiple suction cup and multi-fingered grippers.

Exploring multi-task training of RoManNet for more efficient generalization to new tasks is another interesting future work. The long term vision is to develop learning systems that can use prior knowledge to perform new tasks and continuously improve over time. Such learning systems will enable robots to perform a lot more tasks in industrial and household settings.

7.3 Last note

Over the course of four billion years, nature has witnessed an endless process of evolution, which has led to the emergence of us humans. In a certain sense, we are the product of the most powerful meta-learning algorithm to date. It is an exciting journey ahead of us to develop learning machines that can fully utilize the inductive bias of the world we live in, and thus reaching or even surpassing the pace of human learning.

Bibliography

[1] S. Kumra, S. Joshi, and F. Sahin, "Learning multi-step robotic manipulation policies from visual observation of scene and q-value predictions of previous action," in *International Conference on Robotics and Automation (ICRA)*, pp. 8245–8251, IEEE, 2022.

[2] S. Kumra, S. Joshi, and F. Sahin, "Learning robotic manipulation tasks via task progress based gaussian reward and loss adjusted exploration," *IEEE Robotics and Automation Letters*, vol. 7, no. 1, pp. 534–541, 2021.

[3] S. Kumra, S. Joshi, and F. Sahin, "Antipodal robotic grasping using generative residual convolutional neural network," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020.

[4] S. Joshi, S. Kumra, and F. Sahin, "Robotic grasping using deep reinforcement learning," *IEEE International Conference on Automation Science and Engineering, CASE 2020*, 2020.

[5] S. Kumra and C. Kanan, "Robotic grasp detection using deep convolutional neural networks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 769–776, IEEE, 2017.

[6] I. P. Releases, "Executive Summary World Robotics 2017 Industrial Robots," tech. rep., International Federation of Robotics, 09 2017.

[7] A. Saxena, J. Driemeyer, J. Kearns, and A. Y. Ng, "Robotic grasping of novel objects," in *Advances in neural information processing systems*, pp. 1209–1216, 2007.

[8] Y. Jiang, S. Moseson, and A. Saxena, "Efficient grasping from rgbd images: Learning using a new rectangle representation," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3304–3311, 2011.

[9] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1316–1322, May 2015.

[10] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[11] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pp. 3406–3413, IEEE, 2016.

[12] A. Zeng, S. Song, S. Welker, J. Lee, A. Rodriguez, and T. Funkhouser, "Learning synergies between pushing and grasping with self-supervised deep reinforcement learning," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4238–4245, IEEE, 2018.

[13] D. Morrison, P. Corke, and J. Leitner, "Learning robust, real-time, reactive robotic grasping," *The International Journal of Robotics Research*, p. 0278364919859066, 2019.

[14] D. Morrison, P. Corke, and J. Leitner, "Egad! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4368–4375, 2020.

[15] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, and J. Lee, "Transporter networks: Rearranging the visual world for robotic manipulation," *Conference on Robot Learning (CoRL)*, 2020.

[16] Z. Ju, C. Yang, and H. Ma, "Kinematics modeling and experimental verification of baxter robot," in *Control Conference (CCC), 2014 33rd Chinese*, pp. 8518–8523, IEEE, 2014.

[17] Z. Ju, C. Yang, Z. Li, L. Cheng, and H. Ma, "Teleoperation of humanoid baxter robot using haptic feedback," in *Multisensor Fusion and Information Integration for Intelligent Systems (MFI), 2014 International Conference on*, pp. 1–6, IEEE, 2014.

[18] J. Kober and J. Peters, "Imitation and reinforcement learning," *IEEE Robotics & Automation Magazine*, vol. 17, no. 2, pp. 55–62, 2010.

[19] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *The International Journal of Robotics Research*, vol. 31, no. 3, pp. 360–375, 2012.

[20] A. G. Billard, S. Calinon, and R. Dillmann, "Learning from humans," in *Springer handbook of robotics*, pp. 1995–2014, Springer, 2016.

[21] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 157–173, 2008.

[22] I. Goodfellow, Y. Bengio, A. Courville, and Y. Bengio, *Deep learning*, vol. 1. MIT press Cambridge, 2016.

[23] J. Mahler, J. Liang, S. Niyaz, M. Laskey, R. Doan, X. Liu, J. A. Ojea, and K. Goldberg, "Dex-net 2.0: Deep learning to plan robust grasps with synthetic point clouds and analytic grasp metrics," in *Robotics: Science and Systems (RSS)*, 2017.

[24] F. Zhang, J. Leitner, M. Milford, B. Upcroft, and P. Corke, "Towards vision-based deep reinforcement learning for robotic motion control," *arXiv preprint arXiv:1511.03791*, 2015.

[25] U. Viereck, A. Pas, K. Saenko, and R. Platt, "Learning a visuomotor controller for real world robotic grasping using simulated depth images," in *Conference on Robot Learning*, pp. 291–300, PMLR, 2017.

[26] J. Mahler and K. Goldberg, "Learning deep policies for robot bin picking by simulating robust grasping sequences," in *Conference on Robot Learning*, pp. 515–524, 2017.

[27] R. Calandra, A. Owens, M. Upadhyaya, W. Yuan, J. Lin, E. H. Adelson, and S. Levine, "The feeling of success: Does touch sensing help predict grasp outcomes?," in *Conference on Robot Learning*, pp. 314–323, PMLR, 2017.

[28] A. Murali, Y. Li, D. Gandhi, and A. Gupta, "Learning to grasp without seeing," in *International Symposium on Experimental Robotics*, pp. 375–386, Springer, 2018.

[29] L. Y. Ku, E. Learned-Miller, and R. Grupen, "Associating grasp configurations with hierarchical features in convolutional neural networks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2434–2441, IEEE, 2017.

[30] M. Schwarz, A. Milan, C. Lenz, A. Munoz, A. S. Periyasamy, M. Schreiber, S. Schüller, and S. Behnke, "Nimbro picking: Versatile part handling for warehouse automation," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 3032–3039, IEEE, 2017.

[31] J. Johnson, A. Karpathy, and L. Fei-Fei, "Densecap: Fully convolutional localization networks for dense captioning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4565–4574, 2016.

[32] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *ICRA*, vol. 348, p. 353, Citeseer, 2000.

[33] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis—a survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2014.

[34] Z. Wang, Z. Li, B. Wang, and H. Liu, "Robot grasp detection using multimodal deep convolutional neural networks," *Advances in Mechanical Engineering*, vol. 8, no. 9, 2016.

[35] J. Wei, H. Liu, G. Yan, and F. Sun, "Robotic grasping recognition using multi-modal deep extreme learning machine," *Multidimensional Systems and Signal Processing*, vol. 28, no. 3, pp. 817–833, 2017.

[36] D. Guo, F. Sun, H. Liu, T. Kong, B. Fang, and N. Xi, "A hybrid deep architecture for robotic grasp detection," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 1609–1614, IEEE, 2017.

[37] F.-J. Chu, R. Xu, and P. A. Vela, "Real-world multiobject, multigrasp detection," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3355–3362, 2018.

[38] X. Zhou, X. Lan, H. Zhang, Z. Tian, Y. Zhang, and N. Zheng, "Fully convolutional grasp detection network with oriented anchor box," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 7223–7230, IEEE, 2018.

[39] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt, "Grasp pose detection in point clouds," *The International Journal of Robotics Research*, vol. 36, no. 13-14, pp. 1455–1473, 2017.

[40] Q. Lu, K. Chenna, B. Sundaralingam, and T. Hermans, "Planning multi-fingered grasps as probabilistic inference in a learned deep network," in *Robotics Research*, pp. 455–472, Springer, 2020.

[41] E. Johns, S. Leutenegger, and A. J. Davison, "Deep learning a grasp function for grasping under gripper pose uncertainty," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4461–4468, Oct 2016.

[42] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, *et al.*, "Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching," in *2018 IEEE international conference on robotics and automation (ICRA)*, pp. 3750–3757, IEEE, 2018.

[43] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[44] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pp. 23–30, IEEE, 2017.

[45] J. Brownlee, "Deep learning with python: Develop deep learning models on theano and tensorflow using keras," *Machine Learning Mastery, Melbourne*, 2016.

[46] J. Watson, J. Hughes, and F. Iida, "Real-world, real-time robotic grasping with convolutional neural networks," in *Conference Towards Autonomous Robotic Systems*, pp. 617–626, Springer, 2017.

[47] J. Ruiz-del Solar, P. Loncomilla, and N. Soto, "A survey on deep learning methods for robot vision," *arXiv preprint arXiv:1803.10862*, 2018.

[48] K. Lai, L. Bo, X. Ren, and D. Fox, "A large-scale hierarchical multi-view rgb-d object dataset," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1817–1824, IEEE, 2011.

[49] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," in *2016 International Symposium on Experimental Robotics (ISER)*, 2016.

[50] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, *et al.*, "Using simulation and domain adaptation to improve efficiency of deep robotic grasping," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4243–4250, IEEE, 2018.

[51] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[52] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1–9, 2015.

[53] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[54] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *International Conference on Learning Representations*, 2015.

[55] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, June 2016.

[56] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2818–2826, 2016.

[57] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-resnet and the impact of residual connections on learning.," in *AAAI*, vol. 4, p. 12, 2017.

[58] I. Goodfellow, H. Lee, Q. V. Le, A. Saxe, and A. Y. Ng, "Measuring invariances in deep networks," in *Advances in neural information processing systems*, pp. 646–654, 2009.

[59] Q. Lu, K. Chenna, B. Sundaralingam, and T. Hermans, "Planning multi-fingered grasps as probabilistic inference in a learned deep network," in *International Symposium on Robotics Research*, 2017.

[60] S. Ruder, "An overview of gradient descent optimization algorithms," *arXiv preprint arXiv:1609.04747*, 2016.

[61] F. Chollet, *Deep learning with python*. Manning Publications Co., 2017.

[62] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *Journal of artificial intelligence research*, vol. 4, pp. 237–285, 1996.

[63] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[64] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[65] B. Dresp-Langley, O. K. Ekseth, J. Fesl, S. Gohshi, M. Kurz, and H.-W. Sehring, "Occam's razor for big data? on detecting quality in large unstructured datasets," *Applied Sciences*, vol. 9, no. 15, p. 3065, 2019.

[66] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.

[67] R. E. Bellman and S. E. Dreyfus, *Applied dynamic programming*. Princeton university press, 2015.

[68] G. J. Gordon, "Stable fitted reinforcement learning," *Advances in neural information processing systems*, vol. 8, 1995.

[69] M. Riedmiller, "Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method," in *European conference on machine learning*, pp. 317–328, Springer, 2005.

[70] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, 2016.

[71] M. G. Bellemare, W. Dabney, and R. Munos, "A distributional perspective on reinforcement learning," in *International Conference on Machine Learning*, pp. 449–458, PMLR, 2017.

[72] W. Dabney, M. Rowland, M. Bellemare, and R. Munos, "Distributional reinforcement learning with quantile regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018.

[73] M. Hessel, J. Modayil, H. Van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," in *Thirty-second AAAI conference on artificial intelligence*, 2018.

[74] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*, pp. 387–395, PMLR, 2014.

[75] J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh, "Action-conditional video prediction using deep networks in atari games," *Advances in neural information processing systems*, vol. 28, 2015.

[76] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7559–7566, IEEE, 2018.

[77] D. Silver, H. Hasselt, M. Hessel, T. Schaul, A. Guez, T. Harley, G. Dulac-Arnold, D. Reichert, N. Rabinowitz, A. Barreto, *et al.*, "The predictron: End-to-end learning and planning," in *International conference on machine learning*, pp. 3191–3199, PMLR, 2017.

[78] A. Tamar, Y. Wu, G. Thomas, S. Levine, and P. Abbeel, "Value iteration networks," *Advances in neural information processing systems*, vol. 29, 2016.

[79] V. François-Lavet, Y. Bengio, D. Precup, and J. Pineau, "Combined reinforcement learning via abstract representations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 3582–3589, 2019.

[80] M. A. Moussa and M. S. Kamel, "A connectionist model for learning robotic grasps using reinforcement learning," in *Proceedings of International Conference on Neural Networks (ICNN'96)*, vol. 3, pp. 1771–1776 vol.3, June 1996.

[81] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, vol. 3, pp. 2619–2624 Vol.3, April 2004.

[82] A. Y. Ng, A. Coates, M. Diel, V. Ganapathi, J. Schulte, B. Tse, E. Berger, and E. Liang, "Autonomous inverted helicopter flight via reinforcement learning," in *Experimental Robotics IX* (M. H. Ang and O. Khatib, eds.), (Berlin, Heidelberg), pp. 363–372, Springer Berlin Heidelberg, 2006.

[83] F. Stulp, E. Theodorou, J. Buchli, and S. Schaal, "Learning to grasp under uncertainty," in *2011 IEEE International Conference on Robotics and Automation*, pp. 5703–5708, May 2011.

[84] T. Lampe and M. Riedmiller, "Acquiring visual servoing reaching and grasping skills using neural reinforcement learning," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, Aug 2013.

[85] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013.

[86] L. Tai, G. Paolo, and M. Liu, "Virtual-to-real deep reinforcement learning: Continuous control of mobile robots for mapless navigation," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 31–36, IEEE, 2017.

[87] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, *et al.*, "Scalable deep reinforcement learning for vision-based robotic manipulation," in *Conference on Robot Learning*, pp. 651–673, 2018.

[88] M. Ciocarlie, K. Hsiao, E. G. Jones, S. Chitta, R. B. Rusu, and I. A. Şucan, "Towards reliable grasping and manipulation in household environments," in *Experimental Robotics*, pp. 241–252, 2014.

[89] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 05 2015.

[90] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2014.

[91] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[92] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural*

Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL (A. Moschitti, B. Pang, and W. Daelemans, eds.), pp. 1724–1734, ACL, 2014.

[93] R. Socher, E. H. Huang, J. Pennin, C. D. Manning, and A. Y. Ng, "Dynamic pooling and unfolding recursive autoencoders for paraphrase detection," in *Advances in Neural Information Processing Systems*, pp. 801–809, 2011.

[94] E. Johns and G.-Z. Yang, "Generative methods for long-term place recognition in dynamic scenes," *International Journal of Computer Vision*, vol. 106, no. 3, pp. 297–314, 2014.

[95] N. Sünderhauf, S. Shirazi, F. Dayoub, B. Upcroft, and M. Milford, "On the performance of convnet features for place recognition," in *Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on*, pp. 4297–4304, 2015.

[96] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[97] E. Johns, S. Leutenegger, and A. J. Davison, "Pairwise decomposition of image sequences for active multi-view recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3813–3822, June 2016.

[98] S. Kumra and F. Sahin, "Dual flexible 7 dof arm robot learns like a child to dance using q-learning," in *IEEE System of Systems Engineering Conference (SoSE)*, pp. 292–297, 2015.

[99] C. Chen, A. Seff, A. Kornhauser, and J. Xiao, "Deepdriving: Learning affordance for direct perception in autonomous driving," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2722–2730, 2015.

[100] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *CVPR09*, 2009.

[101] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," in *Advances in neural information processing systems*, pp. 3320–3328, 2014.

[102] J. Bohg and D. Kragic, "Learning grasping points with shape context," *Robotics and Autonomous Systems*, vol. 58, no. 4, pp. 362–377, 2010.

[103] M. Krainin, B. Curless, and D. Fox, "Autonomous generation of complete 3d object models using next best view manipulation planning," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 5031–5037, 2011.

[104] Q. V. Le, D. Kamm, A. F. Kara, and A. Y. Ng, "Learning to grasp objects with multiple contact points," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 5062–5069, May 2010.

[105] D. Song, K. Huebner, V. Kyrki, and D. Kragic, "Learning task constraints for robot grasping using graphical models," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 1579–1585, Oct 2010.

[106] J. Yu, K. Weng, G. Liang, and G. Xie, "A vision-based robotic grasping system using deep learning for 3d object recognition and pose estimation," in *Robotics and Biomimetics (ROBIO), 2013 IEEE International Conference on*, pp. 1175–1180, 2013.

[107] J. Mahler, F. T. Pokorny, B. Hou, M. Roderick, M. Laskey, M. Aubry, K. Kohlhoff, T. Kröger, J. Kuffner, and K. Goldberg, "Dex-net 1.0: A cloud-based network of 3d objects for robust grasp planning using a multi-armed bandit model with correlated rewards," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1957–1964, May 2016.

[108] J. Maitin-Shepard, M. Cusumano-Towner, J. Lei, and P. Abbeel, "Cloth grasp point detection based on multiple-view geometric cues with application to robotic towel folding," in *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2308–2315, 2010.

[109] M. Gualtieri, A. ten Pas, K. Saenko, and R. Platt, "High precision grasp pose detection in dense clutter," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 598–605, 2016.

[110] L. Pinto and A. Gupta, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3406–3413, May 2016.

[111] M. Schwarz, H. Schulz, and S. Behnke, "Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1329–1335, 2015.

[112] U. Asif, M. Bennamoun, and F. A. Sohel, "Rgb-d object recognition and grasp detection using hierarchical cascaded forests," *IEEE Transactions on Robotics*, 2017.

[113] J. Redmon and A. Angelova, "Real-time grasp detection using convolutional neural networks," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1316–1322, IEEE, 2015.

[114] D. Kragic and H. I. Christensen, "Robust visual servoing," *The international journal of robotics research*, vol. 22, no. 10-11, pp. 923–939, 2003.

[115] M. Kopicki, R. Detry, M. Adjigble, R. Stolkin, A. Leonardis, and J. L. Wyatt, "One-shot learning and generation of dexterous grasps for novel objects," *The International Journal of Robotics Research*, vol. 35, no. 8, pp. 959–976, 2016.

[116] J. Bohg, A. Morales, T. Asfour, and D. Kragic, "Data-driven grasp synthesis—a survey," *IEEE Transactions on Robotics*, vol. 30, no. 2, pp. 289–309, 2013.

[117] A. Depierre, E. Dellandréa, and L. Chen, "Jacquard: A large scale dataset for robotic grasp detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2018.

[118] U. Asif, J. Tang, and S. Harrer, "Graspnet: An efficient convolutional neural network for real-time grasp detection for low-powered devices.," in *IJCAI*, pp. 4875–4882, 2018.

[119] H. Zhang, X. Lan, S. Bai, X. Zhou, Z. Tian, and N. Zheng, "Roi-based robotic grasp detection for object overlapping scenes," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4768–4775, IEEE, 2019.

[120] K. B. Shimoga, "Robot grasp synthesis algorithms: A survey," *The International Journal of Robotics Research*, vol. 15, no. 3, pp. 230–266, 1996.

[121] V. Satish, J. Mahler, and K. Goldberg, "On-policy dataset synthesis for learning robot grasping policies using fully convolutional deep networks," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1357–1364, 2019.

[122] J. Tremblay, T. To, B. Sundaralingam, Y. Xiang, D. Fox, and S. Birchfield, "Deep object pose estimation for semantic robotic grasping of household objects," in *Conference on Robot Learning*, pp. 306–316, PMLR, 2018.

[123] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis, "Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 12627–12637, 2019.

[124] M. Yan, A. Li, M. Kalakrishnan, and P. Pastor, "Learning probabilistic multi-modal actor models for vision-based robotic grasping," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 4804–4810, IEEE, 2019.

[125] P. Schmidt, N. Vahrenkamp, M. Wächter, and T. Asfour, "Grasping of unknown objects using deep convolutional neural networks based on depth images," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6831–6838, IEEE, 2018.

[126] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, *et al.*, "Robotic pick-and-place of novel objects in clutter with multi-affordance

grasping and cross-domain image matching," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–8, IEEE, 2018.

[127] J. Varley, C. DeChant, A. Richardson, J. Ruales, and P. Allen, "Shape completion enabled robotic grasping," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2442–2447, IEEE, 2017.

[128] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, vol. 37, no. 4-5, pp. 421–436, 2018.

[129] L. Antanas, P. Moreno, M. Neumann, R. P. de Figueiredo, K. Kersting, J. Santos-Victor, and L. De Raedt, "Semantic and geometric reasoning for robotic grasping: a probabilistic logic approach," *Autonomous Robots*, vol. 43, no. 6, pp. 1393–1418, 2019.

[130] X. Yan, M. Khansari, J. Hsu, Y. Gong, Y. Bai, S. Pirk, and H. Lee, "Data-efficient learning for sim-to-real robotic grasping using deep point cloud prediction networks," *arXiv preprint arXiv:1906.08989*, 2019.

[131] E. Ogas, L. Avila, G. Larregay, and D. Moran, "A robotic grasping method using convnets," in *2019 Argentine Conference on Electronics (CAE)*, pp. 21–26, IEEE, 2019.

[132] U. Asif, J. Tang, and S. Harrer, "Ensemblenet: Improving grasp detection using an ensemble of convolutional neural networks," in *BMVC*, 2018.

[133] H. Liang, X. Ma, S. Li, M. Görner, S. Tang, B. Fang, F. Sun, and J. Zhang, "Pointnetgpd: Detecting grasp configurations from point sets," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 3629–3635, IEEE, 2019.

[134] A. Mousavian, C. Eppner, and D. Fox, "6-dof graspnet: Variational grasp generation for object manipulation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2901–2910, 2019.

[135] A. Murali, A. Mousavian, C. Eppner, C. Paxton, and D. Fox, "6-dof grasping for target-driven object manipulation in clutter," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6232–6238, IEEE, 2020.

[136] X. Yan, J. Hsu, M. Khansari, Y. Bai, A. Pathak, A. Gupta, J. Davidson, and H. Lee, "Learning 6-dof grasping interaction via deep geometry-aware 3d representations," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3766–3773, IEEE, 2018.

[137] H.-S. Fang, C. Wang, M. Gou, and C. Lu, "Graspnet-1billion: A large-scale benchmark for general object grasping," in *IEEE/CVF conference on computer vision and pattern recognition*, pp. 11444–11453, 2020.

[138] H. Xue, S. Zhang, and D. Cai, "Depth image inpainting: Improving low rank matrix completion with low gradient regularization," *IEEE Transactions on Image Processing*, vol. 26, no. 9, pp. 4311–4320, 2017.

[139] C. Arcelli and G. S. Di Baja, "Finding local maxima in a pseudo-euclidian distance transform," *Computer Vision, Graphics, and Image Processing*, vol. 43, no. 3, pp. 361–367, 1988.

[140] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[141] L. Wright, "Ranger - a synergistic optimizer.." https://github.com/lessw2020/Ranger-Deep-Learning-Optimizer, 2019.

[142] J. B. Diederik P. Kingma, "Adam: A method for stochastic optimization," *International Conference for Learning Representations*, 2015.

[143] L. Liu, H. Jiang, P. He, W. Chen, X. Liu, J. Gao, and J. Han, "On the variance of the adaptive learning rate and beyond," in *International Conference on Learning Representations*, 2020.

[144] M. Zhang, J. Lucas, J. Ba, and G. E. Hinton, "Lookahead optimizer: k steps forward, 1 step back," *Advances in Neural Information Processing Systems*, vol. 32, pp. 9597–9608, 2019.

[145] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," in *5th International Conference on Learning Representations (ICLR)*, 08 2017.

[146] D. Morrison, P. Corke, and J. Leitner, "Closing the loop for robotic grasping: A real-time, generative grasp synthesis approach," *Robotics: Science and Systems XIV*, 2018.

[147] H. Karaoguz and P. Jensfelt, "Object detection approach for robot grasp detection," in *2019 International Conference on Robotics and Automation (ICRA)*, pp. 4953–4959, IEEE, 2019.

[148] Y. Wang, Y. Zheng, B. Gao, and D. Huang, "Double-dot network for antipodal grasp detection," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[149] S. Wang, X. Jiang, J. Zhao, X. Wang, W. Zhou, and Y. Liu, "Efficient fully convolution neural network for generating pixel wise robotic grasps with high resolution images," in *2019 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 474–480, IEEE, 2019.

[150] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning." http://pybullet.org, 2016–2021.

[151] B. Calli, A. Singh, J. Bruce, A. Walsman, K. Konolige, S. Srinivasa, P. Abbeel, and A. M. Dollar, "Yale-cmu-berkeley dataset for robotic manipulation research," *The International Journal of Robotics Research*, vol. 36, no. 3, pp. 261–268, 2017.

[152] L. Yen-Chen, A. Zeng, S. Song, P. Isola, and T.-Y. Lin, "Learning to see before learning to act: Visual pre-training for manipulation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7286–7293, IEEE, 2020.

[153] S. Joshi, S. Kumra, and F. Sahin, "Robotic grasping using deep reinforcement learning," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*, pp. 1461–1466, IEEE, 2020.

[154] P. Florence, L. Manuelli, and R. Tedrake, "Self-supervised correspondence in visuomotor policy learning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 492–499, 2019.

[155] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," in *Proceedings of Robotics: Science and Systems*, (Pittsburgh, Pennsylvania), June 2018.

[156] A. Hundt, B. Killeen, N. Greene, H. Wu, H. Kwon, C. Paxton, and G. D. Hager, ""good robot!": Efficient reinforcement learning for multi-step visual tasks with sim to real transfer," *IEEE Robotics and Automation Letters*, vol. 5, no. 4, pp. 6724–6731, 2020.

[157] L. Berscheid, P. Meifβner, and T. Kröger, "Robot learning of shifting objects for grasping in cluttered environments," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 612–618, IEEE, 2019.

[158] S. Gu, E. Holly, T. Lillicrap, and S. Levine, "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates," in *2017 IEEE international conference on robotics and automation (ICRA)*, pp. 3389–3396, IEEE, 2017.

[159] I. Lenz, H. Lee, and A. Saxena, "Deep learning for detecting robotic grasps," *The International Journal of Robotics Research*, vol. 34, no. 4-5, pp. 705–724, 2015.

[160] J. Mahler, M. Matl, V. Satish, M. Danielczuk, B. DeRose, S. McKinley, and K. Goldberg, "Learning ambidextrous robot grasping policies," *Science Robotics*, vol. 4, no. 26, 2019.

[161] M. Riedmiller, R. Hafner, T. Lampe, M. Neunert, J. Degrave, T. Wiele, V. Mnih, N. Heess, and J. T. Springenberg, "Learning by playing solving sparse reward tasks from scratch," in *International Conference on Machine Learning*, pp. 4344–4353, PMLR, 2018.

[162] D. Xu, A. Mandlekar, R. Martín-Martín, Y. Zhu, S. Savarese, and L. Fei-Fei, "Deep affordance foresight: Planning through what can be done in the future," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021.

[163] I. Nematollahi, O. Mees, L. Hermann, and W. Burgard, "Hindsight for foresight: Unsupervised structured dynamics models from physical interaction," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5319–5326, IEEE, 2020.

[164] R. Jeong, Y. Aytar, D. Khosid, Y. Zhou, J. Kay, T. Lampe, K. Bousmalis, and F. Nori, "Self-supervised sim-to-real adaptation for visual robotic manipulation," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2718–2724, IEEE, 2020.

[165] Y. Zhu, Z. Wang, J. Merel, A. Rusu, T. Erez, S. Cabi, S. Tunyasuvunakool, J. Kramár, R. Hadsell, N. de Freitas, *et al.*, "Reinforcement and imitation learning for diverse visuomotor skills," *Robotics: Science and Systems XIV*, vol. 14, 2018.

[166] K. Kase, C. Paxton, H. Mazhar, T. Ogata, and D. Fox, "Transferable task execution from pixels through deep planning domain learning," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10459–10465, IEEE, 2020.

[167] D. Driess, J.-S. Ha, R. Tedrake, and M. Toussaint, "Learning geometric reasoning and control for long-horizon tasks from visual input," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021.

[168] K. Zakka, A. Zeng, J. Lee, and S. Song, "Form2fit: Learning shape priors for generalizable assembly from disassembly," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2020.

[169] S. Devgon, J. Ichnowski, M. Danielczuk, D. S. Brown, A. Balakrishna, S. Joshi, E. Rocha, E. Solowjow, and K. Goldberg, "Kit-net: Self-supervised learning to kit novel 3d objects into novel 3d cavities," in *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE)*, IEEE, 2021.

[170] J. Borràs, G. Alenyà, and C. Torras, "A grasping-centered analysis for cloth manipulation," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 924–936, 2020.

[171] D. Seita, A. Ganapathi, R. Hoque, M. Hwang, E. Cen, A. K. Tanwani, A. Balakrishna, B. Thananjeyan, J. Ichnowski, N. Jamali, *et al.*, "Deep imitation learning of sequential fabric smoothing from an algorithmic supervisor," *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[172] C. Tsiotsios and M. Petrou, "On the choice of the parameters for anisotropic diffusion in image processing," *Pattern recognition*, vol. 46, no. 5, pp. 1369–1381, 2013.

[173] M. Tokic, "Adaptive ε-greedy exploration in reinforcement learning based on value differences," in *Annual Conference on Artificial Intelligence*, pp. 203–210, Springer, 2010.

[174] B. Goodrich, A. Kuefler, W. D. Richards, C. Correa, R. Sharma, and S. Kumra, "Computer-automated robot grasp depth estimation," Mar. 18 2021. US Patent App. 17/020,565.