8-8-2022

InfoMixup : An Intuitive and Information-Driven Approach to Robust Generalization

Andrew H. Meyer
ahm7197@rit.edu

InfoMixup : An Intuitive and Information-Driven Approach to Robust Generalization

Andrew H. Meyer

InfoMixup : An Intuitive and Information-Driven Approach to Robust Generalization

Andrew H. Meyer

August 8, 2022

A Thesis Submitted
in Partial Fulfillment
of the Requirements for the Degree of
Master of Science
in
Computer Engineering

R·I·T | Kate Gleason
College of ENGINEERING

Department of Computer Engineering

InfoMixup : An Intuitive and Information-Driven Approach to Robust Generalization

Andrew H. Meyer

Committee Approval:

Dr. Cory Merkel *Advisor* Date
Assistant Professor
Department of Computer Engineering

Dr. Andres Kwasinski Date
Professor
Department of Computer Engineering

Dr. Matthew Wright Date
Professor
Department of Computing Security

Acknowledgments

It is well known that no great work is ever accomplished in isolation. The author would like to acknowledge the efforts of those who have come before him, and shared their findings for the benefit of their contemporaries and successors. The patience of the advising team, Drs. Cory Merkel and Andres Kwasinski in the development of this complex work in trying times and unfavorable circumstances is greatly appreciated.

Abstract

The discovery of Adversarial Examples — data points which are easily recognized by humans, but which fool artificial classifiers with ease, is relatively new in the world of machine learning. Corruptions imperceptible to the human eye are often sufficient to fool state of the art classifiers. The resolution of this problem has been the subject of a great deal of research in recent years as the prevalence of Deep Neural Networks grows in everyday systems. To this end, we propose *InfoMixup* , a novel method to improve the robustness of Deep Neural Networks without significantly affecting performance on clean samples. Our work is focused in the domain of image classification, a popular target in contemporary literature due to the proliferation of Deep Neural Networks in modern products. We show that our method achieves state of the art improvements in robustness against a variety of attacks under several measures.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Background

1.1 Introduction

The brittle nature of Deep Neural Networks is a well-known problem. Changes in image structure which are imperceptible to humans fool modern classifiers with ease, and often the incorrect prediction is given with high confidence. The prevalence of Adversarial Examples, and the development of algorithms to produce them, suggests that they are not statistical anomalies or outliers, but rather a side-effect of current models and training procedures.

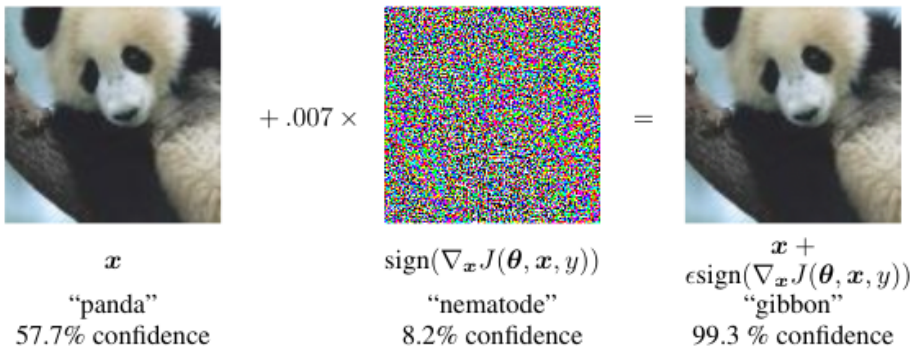


Figure 1.1: Creation of an adversarial example [1]

Recent studies have classified adversarial examples into different categories. Natural adversarial examples [2] are those which have not been modified by any algorithm, but readily fool a classifier. Hendrycks et. al. [2] have further classified natural adversarial examples into those which are in-distribution data, and those which are

out-of-distribution data. In-distribution adversarials are those whose correct classification is among the classes the model was trained on, but which the model fails to classify correctly. Out-of-distribution adversarials are data points which fall outside the set of classes known to the model, but which the model confidently predicts as being in one of the classes it was trained on.

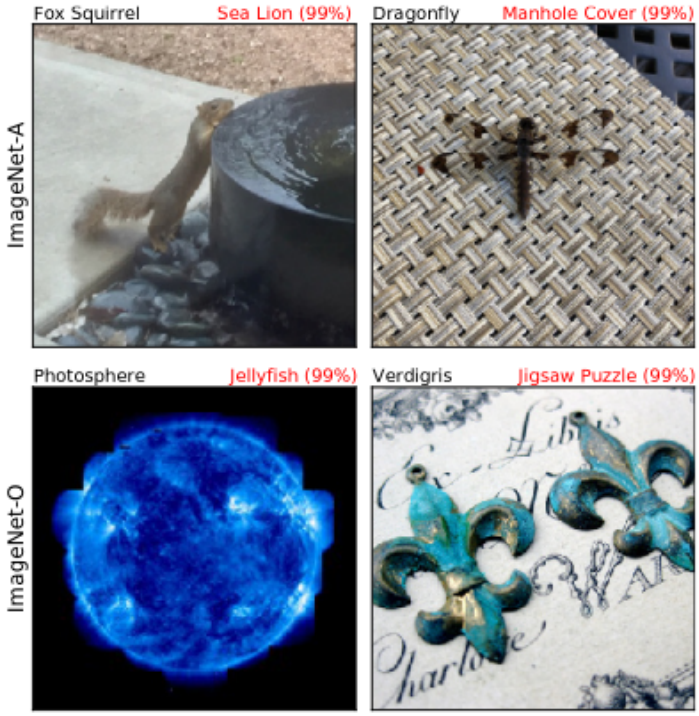


Figure 1.2: Natural Adversarial Examples [2]

One step up in difficulty while remaining near the world of natural adversarial examples, are "common corruptions". In the image classification domain, these are modifications to an image which make its contents less clear via artifacting caused by the image sensor, compression, transmission losses, or environmental conditions during image capture which have caused abberations in the image. The contents of the image are still recognizable to a human observer, but the quality of the image is reduced. Examples include rain on the lens of a camera, JPEG compression artifacts, salt-and-pepper noise, and posterization. Hendrycks et. al. have produced a database

of these so-called "common corruptions" on ImageNet-1k and TinyImageNet.[4] Their approach algorithmically generates abberations which follow similar trends to the sorts of corruptions observed in real-world images.

Adversarial examples can additionally be crafted algorithmically, via optimization techniques. By far the most difficult to overcome, it is algorithmically generated adversarial examples which granted these extreme edge case images their name. These adversarial *attacks* generate data which are edge cases for a model. Several subclasses of adversarial attack have been created. Attacks which have access to the model's predictions and gradients are considered white-box attacks. Attacks which have access only to the predictions are called black-box. Targeted adversarial attacks attempt to cause a certain sample of a source class to appear to the model to belong to a specific target class, while untargeted attacks simply aim to create any misclassification, regardless of the final classiciation result of the perturbed image.

There are two overarching classes of adversarial attack. *Untargeted attacks* are those which cause any misclassification, but do not attempt to make an example of a particular class appear to be an example of a specific class. *Targeted attacks*, then, are those which intend to cause an example of some class to appear to be an example of a different, but specific class. Within each of these classes, there are *black-box* and *white-box* attacks, where *black-box* attacks do not have knowledge of the model, whereas *white-box* attacks assume the adversary has full knowledge of the model architecture, and has access to more than classification results for a set of examples.

It is difficult to draw distinction between what is an adversarial example, and what is simply corrupted or unrecognizable. Adversarial examples generally have obvious classification to a human observer, but it is difficult to draw a boundary in a rigorous way under this intuition. Although mathematical bounds on what is considered adversarial have not been established, a concept for the general strength of an adversary has been developed. Common to all adversarial attacks is the concept

of a *maximum perturbation*, a rational positive number which is relative to a distance metric in the data space between the original and the adversarial image. Common distance metrics are l_0, l_2, and l_∞ norms. The maximum distance from the source data point the attacker is allowed to move the data point in space is referred to as the epsilon (ϵ) of the attack.

Strength of algorithms vary and are difficult to compare. FGSM is known to be fast and effective against unregularized networks, but can be defended against with Adversarial Training [5] and regularization. In contrast, attacks such as the method of Carlini and Wagner [6] are known to be very strong and difficult to defend against.

In our work we will consider all varieties of adversarial examples, but focus on those which are generated algorithmically.

1.2 Background

1.2.1 Adversarial Machine Learning

A common property of deep neural network architectures is the prevalence of "adversarial examples", or inputs which are imperceptibly altered to a human observer, but which are incorrectly classified by the neural network [7].Adversarial examples are observations which have been intentionally altered to be a worst-case example of a particular class as viewed from the perspective of the network.

Consider a topological space of n-dimensional vectors, $\mathcal{X}$. A classifier C takes vectors in $\mathcal{X}$ and maps them to a set of m-dimensional vectors $\mathcal{Y}$ which are class labels corresponding to a subset of vectors in $\mathcal{X}$. Let $\mathcal{A}$ and $\mathcal{B}$ be non-intersecting, open subsets of $\mathcal{X}$ with class labels $\mathcal{A}_t$ and $\mathcal{B}_t$, which contain all vectors in $\mathcal{X}$ which correspond to their associated labels in the true distribution. Let $\mathcal{A}_l$ and $\mathcal{B}_l$ be subsets of $\mathcal{A}_t$ and $\mathcal{B}_t$ which are the examples which C can map to their corresponding labels in $\mathcal{Y}$ correctly. The classifier $\mathcal{C}$ learns a boundary $\mathcal{B}_l$ between $\mathcal{A}_l$ and $\mathcal{B}_l$, which will

be dissimilar from the true boundary, $\mathcal{B}_t$ between $\mathcal{A}_t$ and $\mathcal{B}_t$. Using this construction, we define adversarial examples on $\mathcal{A}_t$ to be those vectors which are in $\mathcal{A}_t$, and also in $\mathcal{B}_l$.

Definition 1. *Adversarial Examples* are vectors which reside in the intersection of $\mathcal{A}_t$ and $\mathcal{B}_l$ for some class $\mathcal{A}$.

Under the traditional view of class boundaries, we consider there to be a "margin" region which resides between two classes, and it is believed that both natural and generated adversarial examples may lie in this space. While this view is not false, we propose a slightly modified view of the representation space, which lends itself to describing adversarial examples and potential solutions better than the traditional view.

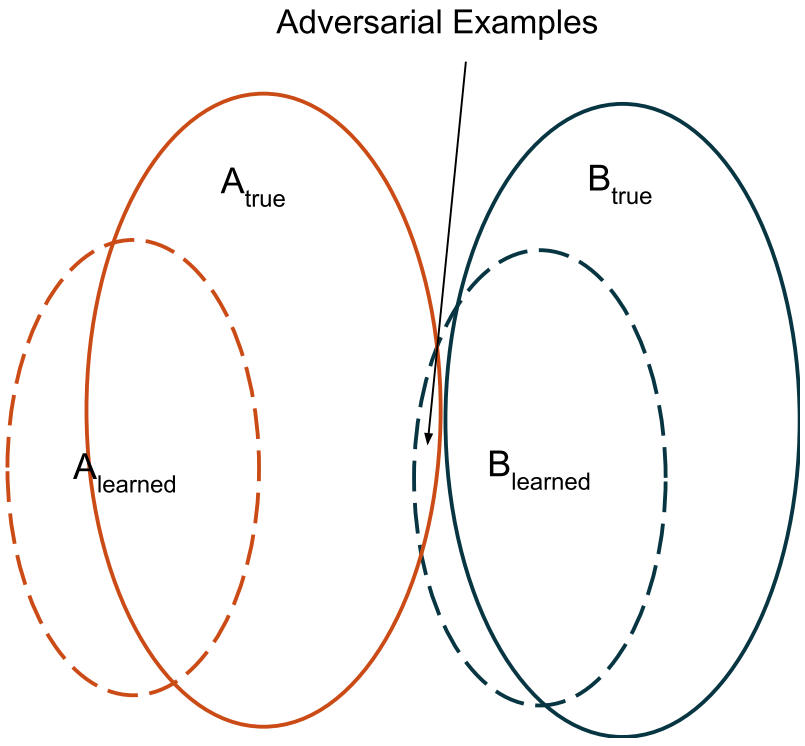


Figure 1.3: Boundary between hypothetical classes A and B.

Note that we can also consider naturally misclassified examples in this framework. Examples which are misclassified at all can be considered "adversarial", though they

are not perturbed in any way, but are simply outside the model's classification boundary for its target class. "Natural adversarial examples" are common. Hendrycks et. al. have created the ImageNet-A benchmark [2], a collection of images which correspond to the ImageNet-2012 labels, but which are all misclassified by state-of-the-art models.

Many explanations have been made for the existence of these examples. Goodfellow et. al claim that adversarial examples arise from networks being "too linear" [1], making them unable to draw sufficiently complex boundaries in the input space to classify a wide space of examples in each class. Another explanation for the existence of these adversarial examples is the presence of highly predictive, but non-robust features in a dataset. These non-robust features are an intrinsic property of the images in commonly-used datasets, and they are relied upon by classifiers when the objective function being optimized considers only classification accuracy [8].

Although generalization is arguably one of the most central concepts to modern machine learning, it is surprisingly difficult to make a rigorous definition of generalization that captures all ways in which it can be applied. Broadly, we can say that a model which generalizes well has learned boundaries which are "close" to the true boundaries for every subset in $\mathcal{X}$. Several tactics for indicating the similarity of the learned boundaries to the true boundaries arise from statistical modelling. A common measure of generalization is classification accuracy on a withheld set of testing data. Predicted class labels in a test set are compared to their true labels, and with a large test set, high similarity between the predicted labels and the true labels is considered to be an indication of a well-fit model. In applications that require more robust classifiers, validation procedures include cross-validation, and robust test set generation. The former splits the dataset into fractions of testing and training data such that each pair of train and testing sets is mutually exclusive, but the test data for one train-test pair may appear in the training data for a different train-test pair. This method allows the model to be compared against itself on different arrangements

of the data, to ensure the model performance is not a side effect of poorly chosen testing data. Robust testing involves withholding a further subset from a cross-validated dataset which is intentionally chosen to be different from the rest of the data. In the case of biometrics, this may be data from subjects whose training data is not included in cross-validation in any way.

Both of these techniques rely on random selection to ensure that the test set provides useful information about the class boundaries. When this assumption fails, models which appear to perform well on a particular task during development may fail in deployment. In adversarial settings, accuracy on test set examples is called "clean accuracy", to allow the term "accuracy" to apply to examples which have been perturbed by an attacker.

In this work we suggest that generalization and adversarial robustness are two facets of the same problem. Fundamentally, a classifier whose learned boundaries are identical to the true boundaries of each known class in the input space would have no adversarial examples, and no unexpected failures due to incorrect labeling. Training such a system would require infinite data, and so instead we seek a way to determine the degree to which the relevant information about a distribution of labels has been extracted by a model, thus providing a way to make informed estimations about the fitness and generalization of a model.

1.2.2 Information Theory in Machine Learning

Information theory is the field of mathematics concerned with the ways in which "messages" — elements of a set of possible messages, each with their own related meaning and information content, are recovered either exactly or approximately after some process has taken place. It was coined by Claude Shannon and his fellows, the first formal public appearance being the famous paper **A Mathematical Theory of Communication** [9]. Consider an observation in $\mathcal{X}$ with some corresponding label in

$\mathcal{Y}$. If $\mathcal{X}$ is a space of images, for example, then a single observation might be labeled as "panda", but contain context that one might expect to see along with an image of a panda, such as bamboo fronds, or blue skies, etc. All of this context is information that is present in the observation, but is useless to the classification problem at hand. Ideally, a well-fit classifier would be able to identify that a given observation contains a panda even if that panda were placed in a completely unfamiliar context. If the intent of the model is to obtain a label for the foreground of an image, then all information regarding the background can be discarded. In effect, the image can be compressed, removing all information from it that is non-essential to the classification task at hand. Since the label "panda" contains the same meaning as a context-free image of a panda in this context, but with many less bits required to represent it, it is clear that the image is not the most efficient way to represent this information.

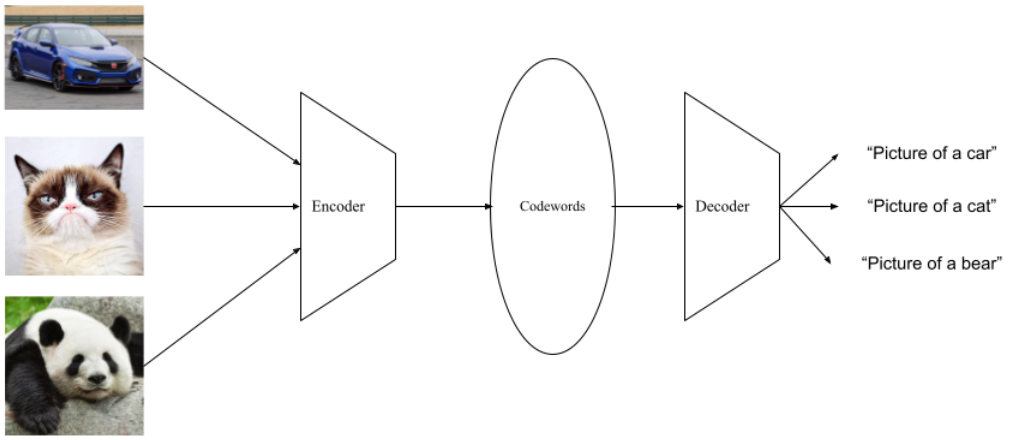


Figure 1.4: High-level diagram of encoding information into a set of codewords which capture the information content of a set of examples, and then decoding the codewords into a different representation.

Central to information theory is the idea that for a given set of possible messages, there is a minimal set of "codewords" that can represent each possible message while

allowing the original to be reconstructed in whole or in part. The loss of information can be useful, for example in providing more reliable communication channels by maximizing the use of channel capacity on information that is critical, and minimizing transmission of information which is not necessary. In the case of learning algorithms, minimizing information about the input space while maximizing information about the label space is critical to solving complex tasks due to both resource and representational ability constraints.

The concept of compression — the idea that a finite set of messages can be represented in a compact way, possibly at the expense of precision, can be applied to learning algorithms instead of communication channels, and is done so by the Information Bottleneck Method [10]. The Information Bottleneck (IB) Method restructures the learning problem into an encoding and decoding problem with a constrained optimization solution. Most classification tasks operate on high-dimensional, information-rich input spaces and yield vectors in relatively low-dimensional spaces. In the view of information theory, this means that the relevant information about the label distribution contained in each observation is small compared to the total amount of information contained in the observation. Thus, the model is trying to extract only the relevant information, or equivalently, create a compressed representation of the input from which the label of the observation can be easily obtained. The IB method creates an optimization problem, where the maximum amount of information about the label distribution $\mathcal{Y}$ is extracted from the input distribution $\mathcal{X}$ into a compressed representation $\mathcal{T}$, while minimizing the amount of information contained in $\mathcal{T}$ about $\mathcal{X}$.

The IB method translates to neural networks well. The compressed representation of $\mathcal{X}$ is exactly the hidden representation $\mathcal{H}$ or latent space of vectors observed at the output of the penultimate layer of the neural network. It is from this hidden representation that labels of the observations are predicted by a linear classifier. It is

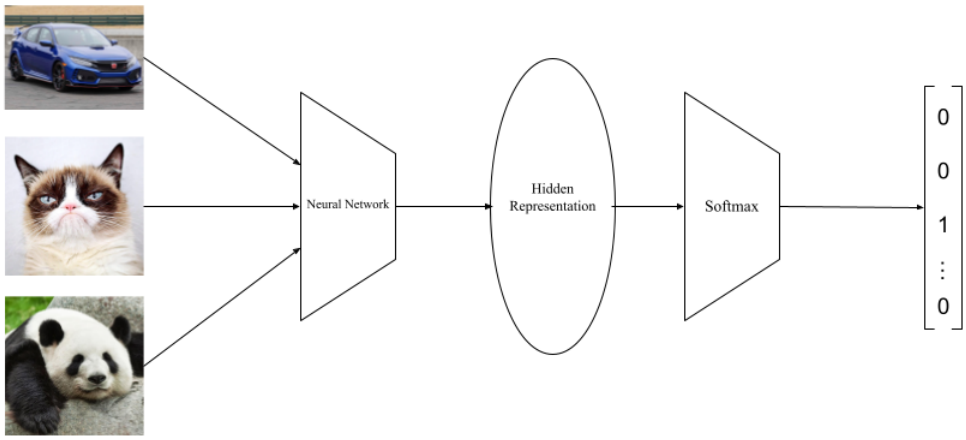


Figure 1.5: High-level diagram representing the use of an Information Bottleneck on a neural network as an encoder, and a softmax classifier as a decoder for a one-hot class representation.

easy to see then, that $\mathcal{H}$ is produced through the IB framework implicitly, since only so many bits of representation are available in $\mathcal{H}$, and standard training[8] optimizes the *usefulness* of $\mathcal{H}$ to the linear layer in predicting $\mathcal{Y}$.

The important side effect of standard training is that it encourages the model to learn *any* representation $\mathcal{H}$ that is useful for predicting $\mathcal{Y}$, without any constraint on the whether those features generalize well. Standard training will in fact choose predictive features over features which generalize well, even if they are brittle [8]. While the IB framework does limit the amount of information which can be retained in the hidden representation, which may discourage brittle features from being retained by the model, ultimately it does not eliminate the problem — sufficiently predictive, but brittle, features may still be retained in the compressed, hidden representation learned by the model.

A significant challenge in training robust models is to know which features are robust, and which are not. Often times, the features learned by deep neural networks

are unintelligible to humans, and thus the decision boundaries drawn by the network are similarly difficult to understand. Adversarial features are completely unintelligible to humans, as shown in the work of Ilyas et. al. [8], where images containing only adversarial features were produced by means of a Projected Gradient Descent adversary. To the human eye, these images appear to be static with a few silhouettes of unintelligible shapes within them, but the network can be confident that they correlate to one of its classes. It is often the case that extremely small perturbations of the input data cause catastrophic differences in the classification result.

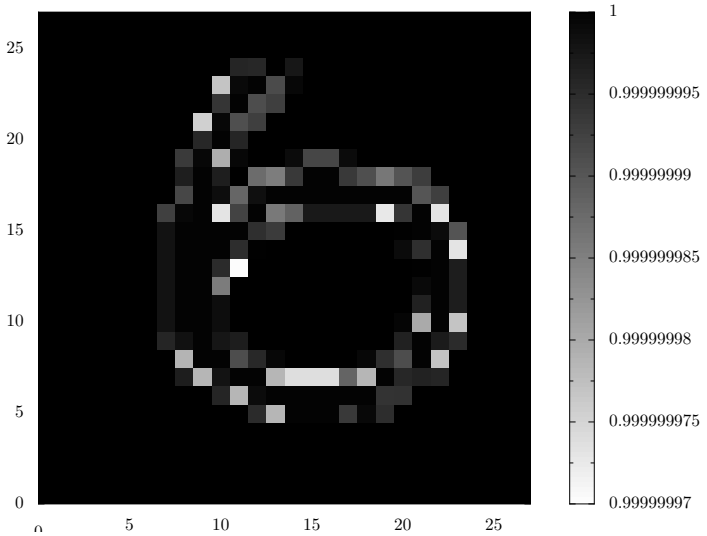


Figure 1.6: Image comprised of solely non-robust features, generated from an MNIST test image for "6" using the FGSM attack with $\epsilon = 0.1$. The above image is labeled "0" by an unsecured network. This image has been scaled for visibility the scale on the righthand side reveals the severity of this problem.

It was found in that work that these images containing only adversarial features suffice for standard classification; models trained on these images alone were able to acheive state-of-the-art accuracy on CIFAR-10 images from which the adversarial images were produced, which suggests that adversarial features are not the noise that they appear to be, but rather are highly predictive features present in natural settings, which are not perceptible to humans. This reality suggests that even an ideal model under information bottleneck, having only as many bits in the hidden

representation as in the target distribution, may contain adversarial features which are more predictive than robust features on the subset of the data distribution used during training, thus leading to a model which appears to perform optimally, but will fail in deployment.

Before adversarial examples were ever formally studied, overfitting was a well known failure mode of both statistical and deep neural models. Broadly, an overfit model is one which performs well on its training data, according to some metric of performance used to evaluate it, but performs significantly worse on related data which it has not seen before, by the same metric. Adversarial examples, then, are just an extreme case of overfitting, where nearly imperceptible differences between the training data and the adversarial data cause the same behavior as an overfit model.

Overfitting is a side-effect of the bias-variance tradeoff that is common to all statistical models. Any attempt to reduce the variance of a model will cause it to become biased towards the data used to develop the model. In the extreme case, the model perfectly learns the data distribution. The true distribution, from which the data was sampled, however, will in practice always be wider than the sampled distribution, and thus the model will fail when shown a sample from the true distribution.

An effective method of reducing overfitting is to make the model less complex, which widens the confidence interval and thus variance of the model, but causes it to be less biased towards the training data. This may have the effect of lowering the accuracy that the model can obtain on training data, but allows it greater flexibility to make predictions on samples outside the training data.

In deep neural networks, however, model complexity is not well understood. Reducing the number of trainable parameters of the model by decreasing the number of layers, removing connections between layers, or decreasing the number of nodes in each layer — may reduce the complexity of the network so much that it is unable to learn even the training data.

To combat this, rather than decreasing the number of parameters of the model, the model is instead *regularized*, or given additional constraints that prevent it from learning a strongly biased representation of the training data. Numerous regularization techniques have been developed in the study of robust neural network models, which can be broadly divided into three categories:

1. **Weight regularization schemes.** These techniques apply a constraint on the trainable parameters of the network through a penalty applied to the loss function. Manifold Mixup [11] implicitly apply a smoothing constraint on the weights of each layer by causing points around each transformed datapoint to cause similar predictions.

2. **Data augmentation schemes.** Methods of this class create additional data from only the training data, generally through transformations, such as scaling and rotation. Methods such as Mixup [12] perform interpolation between examples of different classes. Modifications like MixBoost [13] perform interpolation within a minority class to produce more examples of that class.

3. **Adversarial training schemes.** An extension of data augmentation, these methods explicitly extend the data distribution by use of adversarial examples [5] and harsh transformations [14].

In this work, we will focus on data augmentation and adversarial training, which are most closely related to our approach. We will show that carefully applied regularization can improve adversarial robustness without increasing model complexity, and propose a new method for improving the robustness of models, along with supporting empirical results.

1.3 Related Work

1.3.1 Data Augmentation and Adversarial Training Schemes

Perhaps the most effective and well-known way to improve adversarial robustness in deep neural networks is *adversarial training*, proposed by Madry et. al. [5]. Under this scheme, a trained model with similar or identical architecture to the target network to be implemented is trained on the clean dataset to desired performance. After it has been trained, the model is attacked by a variety of different adversaries, and the adversarial examples produced by the adversary are saved. After a large corpus of adversarial examples have been collected, a new model is trained on both the clean and adversarial data. The resulting model is significantly more robust to attack than models trained only on clean examples.

Their work builds on the idea that adversarial examples stem from features present in the input data which are predictive, but brittle, as supported by their later 2019 work [8]. By including adversarial examples in otherwise clean batches, these adversarial features are muted in the learned representation. In their work, they note that FGSM adversaries do little to improve the robustness of models in their method, indicating that FGSM attacks are relatively weak. This is supported by [15], which shows that models robustified against FGSM overfit to l_1 distance metrics.

Zhang et. al. proposed a novel and elegant method for improving both predictive performance and adversarial robustness with the now-famous *mixup* technique[12]. Mixup does not require any adversarial examples, but instead creates new, soft-labeled examples by linearly interpolating between examples from different classes with known labels. Mixup linearly interpolates according to Equation (1.1), where λ is drawn from a Beta distribution with equal parameters $\alpha = \beta \in (0, \inf)$. The single hyperparameter α controls the extremity of mixing. As α approaches zero, the values of lambda drawn as $\lambda = \beta(\alpha, \alpha)$ will have higher probability density towards zero

and one. As α approaches infinity, the drawn values will be more uniform on $[0, 1]$.

$$\begin{aligned} \hat{x} &= x_a \times \lambda + x_b \times (1 - \lambda) \\ \hat{y} &= y_a \times \lambda + y_b \times (1 - \lambda) \end{aligned} \tag{1.1}$$

The effect of this is that the interpolated examples produced by mixup are more similar to their ground-truth classes for lower values of α, and are more of a mixture for higher values of α. It is proposed that the examples produced by mixup are potentially outside the data manifold visible through the samples in the data set, and overlap with examples which might be chosen by an adversary [15].

The value chosen for α is strongly dependent on the model and the dataset used during training [16], which makes training with mixup difficult and time consuming for many modern architectures.

AugMix [14] is a data augmentation technique which combines several fixed-function transformations on image data, creating new examples which share the same label as the original image. AugMix creates new examples and labels in the same fashion as standard data augmentation techniques such as scaling, rotation, cropping, and posterization, but it interpolates between pairs of the transformed images to create a single new image which looks noticably distorted to the human eye, but is still clearly within the original class. These images are outside the distribution of clean images in pixel space, which places them in areas that an adversary may attempt to perturb an image towards. By including them in the training data, AugMix improves adversarial robustness by widening the learned distribution.

CutMix [17] is a data augmentation technique which interpolates between images by splicing them together in a picture-in-picture fashion, rather than linearly interpolating between them in pixel space as Mixup does. CutMix combines image by selecting randomly sized and randomly located squares from an image and splicing them into an equally-sized and randomly located space in a base image. The label

is mixed based on the proportion of the resulting image which is assigned to each underlying label. This strategy improves the localization of predictive features, encouraging the model to recognize foreground features anywhere in the image. An additional benefit is that images with multiple labels and foreground objects are shown to the model, which improves the localization of objects in the representation — a model pretrained with CutMix on a classification task can improve the performance of an image segmentation task when compared to a pretrained model that was not trained with CutMix. The randomly selected image segments are chosen via uniform distribution over the image dimensions, by default. This leaves the possibility that an entirely or primarily background-containing patch of the image could be cut out of one image, and placed directly over a primarily foreground patch of another image, thus obscuring the foreground features of both classes. The resulting label will still indicate that the resulting image is a mixture of the two original classes, which may cause the model to become confused, have degraded accuracy on clean samples, or produce confident predictions on information-free samples.

PuzzleMix [18] incrementally builds upon the concept of object localization from CutMix. The main improvement over CutMix is the use of *optimal transport* to place the spliced image segments such that foreground segments are always selected. Thus, PuzzleMix overcomes the main drawback of CutMix. PuzzleMix employs a pre-trained classifier which was trained in any way to produce a *class activation map* for each sample. The class activation map provides "saliency information" about the image. This saliency information indicates which areas of the image were most important in making the classification for that image. Puzzle mix does this for each image in a set of images to be mixed. The high-saliency areas of each image contain primarily foreground features for a well-trained model, and low-saliency areas should contain primarly background features. This allows PuzzleMix to create new training samples such that the entire image contains foreground features from many classes.

Similarly to CutMix, PuzzleMix labels the resulting image according to the fraction of the image assigned to each class. PuzzleMix produces localized feature recognition just as CutMix does, but without the potential for feature obfuscation.

The need for saliency information can make PuzzleMix computationally expensive compared to other data augmentation techniques. A pretrained classifier is required for PuzzleMix to operate, although it does not have to be of the same architecture as the model to be trained with PuzzleMix. Additionally, saliency maps could be precomputed and cached, allowing multiple training runs to proceed with only slightly increased computational load as compared to a standard training session. A significant drawback of PuzzleMix is that it does not specify how the foreground image sections are spliced together in the image. PuzzleMix subsamples the saliency map down to a 4×4 matrix, and selects elements from that matrix above a certain threshold to be mixed into the new image. The selected elements from the subsampled saliency map are expanded to match the original image dimensions, and in turn used to selectively mask background features. The tiles of foreground information are randomly emplaced in the mixed image, which has the drawback of potentially allowing foreground features to be split apart in a way that degrades their recognizability.

Co-Mixup [19] builds upon PuzzleMix and CutMix, and resolves the potentially hazardous dissociation of foreground features from PuzzleMix. The Co-Mixup algorithm solves an optimization problem for each minibatch, choosing how many images to mix and what regions from the source images should be chosen such that the patches contain most of the foreground information. The Co-Mixup mixing objective is significantly more complex than even PuzzleMix, but it automatically splices images together in a semantically-sensible manner such that the final image strikes a balance between maximizing salient information via foreground features, and maximizing diversity, by including foreground features from as many images as possible. Co-Mixup creates a jigsaw puzzle for as many images in each minibatch as possible,

where the edges of the mixed regions may not be square bounding boxes. Each jigsaw piece from a source image contains primarily foreground features, thus maximizing the amount of useful information in each image passed through the model. This maximization ensures that the model learns to recognize features in all regions of the input image, and that each image contains primarily predictive, foreground features.

It is unclear how useful any of CutMix, PuzzleMix, or Co-Mixup are with regard to adversarial robustness. While CutMix and PuzzleMix both show significant improvement against an FGSM adversary, Co-Mixup has no analysis relating to adversarial robustness. PuzzleMix claims to be able to include Adversarial Training in their method without additional cost, since their methods already employ gradients of the loss function to select images for mixing; The same gradient can be used for creating adversarial examples with a variety of attack algorithms. PuzzleMix does not perform adversarial analysis on ImageNet, which makes it difficult to compare against other methods who report adversarial accuracy on ImageNet.

Adversarial Vertex Mixup (AVM) [20] is a Mixup-inspired technique specifically designed to improve adversarial robustness. It combines Adversarial Training with Mixup. AVM seeks to improve Adversarial Training by reducing *Adversarial Feature Overfitting*, whereby the model learns the adversarial features in combination with robust features, causing it to overfit to the attacks, distance metrics, and attack strengths used during Adversarial Training. AVM operates by drawing a vector in the input space between an example and an adversarial example generated from it by an attacker, and creates new examples by sampling along that vector. In this way, AVM can create both mild and very strong adversarial examples easily, from relatively few base adversarial examples at very little computational cost compared to generating a similar number and variety of examples via directly attacking a network. AVM can be used with any adversary or combination thereof. AVM does not sacrifice significant performance on clean examples, but can significantly improve performance

on adversarial examples. No comparison against ImageNet is made, which makes comparison against VIB and other schemes impossible.

1.3.2 Regularization Schemes

In a similar vein as mixup, Manifold Mixup operates in much the same manner, but applies equation (1.1) to the hidden representation at various layers of the network [11]. The effect of Manifold Mixup is to flatten the representation learned at each layer, which has the effect of simplifying the boundaries and representations, which has the effect of improving classification performance, and minimally improving adversarial robustness.

The original Mixup technique draws from the dataset at random for each minibatch and mixes two minibatches against each other. Manifold Mixup, however, mixes within each batch for efficiency.

The authors of Manifold Mixup note that their method improves defense against FGSM attacks, but did not markedly improve performance against projected gradient descent (PGD) attacks.

Batch Normalization [21] is a well-known regularization strategy that is essential to modern network architectures, the most prevalent being the ResNet family of architectures [22]. ResNets employ batch normalization in their basic building blocks between each convolutional layer to prevent features from shifting in a particular direction at each layer, which hurts the generalization and learning speed of layers deeper in the network pipeline. Batch normalization effectively whitens the inputs to a particular layer on a per-batch basis, optionally with learned parameters which correct the mean and variance of a mini-batch based on the data observed for an entire epoch. This process has the same benefit as whitening the input data to a network — it enables networks to achieve better performance in less epochs, using a higher learning rate. On its own, batch normalization does not directly improve

adversarial robustness. A perturbed data sample will still be perturbed after its mean and variance have been corrected.

The Variational Information Bottleneck (VIB) [23] is the realization of The Information Bottleneck Method of Tishby et. al [10]. VIB adapts the work of Kingma and Welling on Variational Autoencoders [24], specifically the *reparameterization trick* to classification problems. The reparameterization trick allows the computation of gradients through a function involving a Gaussian random variable, which in turn allows a model to output the sufficient statistics for a random variable, instead of a direct prediction of a vector in the latent space. This prediction can be sampled and used during decoding, which aids the network in learning to represent similar inputs in a consistent manner.

VIB leverages this encoder/decoder paradigm to produce Gaussian random variables in the hidden representation space. The hidden representation outputs are considered in the objective function (1.2) along with the classification results.

$$J_{IB} = \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{\epsilon\sim p(\epsilon)} \left[-\log q\left(y_n | f\left(x_n, \epsilon\right)\right)\right] + \beta \mathrm{KL}\left[p\left(Z|x_n\right), r\left(Z\right)\right] \tag{1.2}$$

In the VIB objective, a surrogate distribution, $r\left(z\right)$ is used as the target distribution that all the class Gaussians are forced to be similar to via the VIB objective. The authors take $r\left(z\right)$ to be equal to $\mathcal{N}\left(0, I\right)$, the standard normal in the representation space. This choice of surrogate distribution acts as a weight regularizer, forcing the statistics predicted by the model to be near to zero and with low variance. VIB significantly improves adversarial robustness against the targeted version of C&W 2016 [6] on both ImageNet and MNIST due to its improved representation and well-regularized weights. However, much like Mixup, VIB requires careful tuning of its hyperparameter, β, which controls the strength of the regularization loss when compared to the classification loss.

From an information theoretic point of view, VIB constrains the number of bits

of mutual information between the data and the representation, $I(\mathcal{X};\mathcal{Z})$, as stated by the Information Bottleneck method. In standard training, no constraint is placed on how the model learns to predict $\mathcal{Y}$ from $\mathcal{X}$, and often, classifiers will have representation spaces capable of representing far more bits of information than are strictly necessary to encode $\mathcal{Y}$, which suggests that, with no constraint on the information in $\mathcal{Z}$, some information about $\mathcal{X}$ which is not strictly necessary for predicting $\mathcal{Y}$ is likely to be present. This ancillary information retained about the model causes it to overfit to $\mathcal{X}$. VIB places a constraint on the information retained about $\mathcal{X}$ by the model, which encourages the model to retain significantly less information about the training data.

1.4 Adversarial Machine Learning

1.4.1 Attack Strategies

Techniques used to train accurate models can also be used to attack them. In the simplest case, an adversarial example can be found by simply making random perturbations to a sample until the model incorrectly classifies it, but this brute force approach is often inefficient and may fail to produce images which are adversarial and minimally perturbed under some metric. A few classes of attack have emerged as new adversarial attack methods are developed.

- **Iterative Methods** Iterative methods start from an initial data point that is known to be within a particular class boundary, and query the model with this sample to learn the gradient of the model at the data point in the data space. The gradient information is then used to direct a search for adversarial examples near the initial data point.

  The Fast Gradient Sign Method (FGSM) [1] is one of the earliest and most well-known adversarial attacks, which moves against the direction of the gra-

dient sign in search of adversarial examples. Adversarials produced by FGSM appear to have a small amount of colored noise added to them and easily fool unsecured networks. FGSM is easily defended against, however, due to its dependency on quality gradient signal and iterative, unidirectional approach to finding adversarials.

- **Optimization-based Methods** Optimization-based methods use numerical methods to optimize a function which produces an adversarial example from a non-adversarial sample, and use the gradients of the model at multiple points to produce a minimally perturbed adversarial. These multi-step methods start at a known data point and make a small step against the gradient, and then query the model again with the modified sample. While these methods depend on being able to query the model and obtain gradients at many points, they are often extremely powerful and produce images with smaller perturbations than iterative methods in many cases. The most prominent optimization-based method is that of Carlini and Wagner in their 2016 work [6]. This method defeats even secured moedls with minimal effort, yielding images which fool even secured networks.

1.4.2 Transferability of Adversarial Examples

A salient feature of adversarial examples is their transferability between trained models and architectures [6]. Adversarial samples produced by attacking a ResNet are likely to also fool VGG networks, and vice versa. This would seem to indicate that adversarial examples are not necessarily a feature of the model, but of the training data.This poses a significant threat to "secured" models, or attack situations in which the defender assumes immunity from powerful white-box attacks due to their model being private. An attacker could produce adversarials which have a high probability of success on the black-box model by attacking a completely dissimilar model.

1.4.3 Obfuscated Gradients as a Defense

Many defenses which attempt to secure a model from adversarial attack do so by obfuscating the gradients of that model. Gradients can be made to point in the wrong direction, become numerically unstable, or become non-differentiable in critical points by training the model to have parameters causing these issues. Models with these obfuscated gradients are difficult to produce attacks for, since any optimization-based attacks will get stuck in local minima or fail to converge on an adversarial example for the specified input image. Although optimization based attacks will fail against these models, several authors have shown that this assumption is false in the general case.

1.4.4 Defeating Weak Defenses

Common defenses to adversarial attacks appear to perform well under a specified threat model and against specific attacks.

- **Approxmiation of Gradients**

  The work of Athalye et. al. [3] proposes that defenses which rely on changing the gradients of the model can be defeated through approximations of those gradients. They propose Backwards Pass Differentiable Approximation (BDPA), which relies on an adversarial $\hat{x}$ satisfying $\hat{x} \sim x$ in the data space. Then a linear approximation to the gradient at $\hat{x}$ can be made from the gradients at x.

- **Surrogate Models**

  Liu et. al.[25] show that it is possible to transfer adversarial examples from a surrogate model will transfer to a black-box model. Non-targeted examples transfer much more easily than targeted adversarials. Targeted adversarials may still cause misclassification, but may not cause the desired label to be

predicted. This is especially true considering that the choice of labels may be different among models that are trained on different data sets.

While the work of Liu et. al. considers transferrability between unsecured models, the changes necessary to make examples transfer from unsecured models to secured models may be small, and the work to transfer from one secured model to another may be even less.

Chapter 2

Theory and Algorithm Design

In this work, we propose *InfoMixup* —our novel regularizer, which we intend to be an intuitive and explainable way to improve the representation of a neural network in terms of separability, while simultaneously improving the robustness of that representation under attack. *InfoMixup* bridges between a data augmentation and a regularization scheme, by building on top of both Manifold Mixup and the Variational Information Bottleneck (VIB) methods.

2.1 Categorizing *InfoMixup*

2.1.1 *InfoMixup* vs. Data Augmentation

Data augmentation strategies are known to be expensive in training time. While many modern networks have pre-trained weights available for use in various settings, the errors and weaknesses of those weights are carried with them into deployment, possibly unbeknownst to the implementors. Not all implementors or researchers have the resources required to produce an adversarilly robust model using a data augmentation scheme, since methods like Adversarial Training add further examples to an already augmented dataset.

The authors are not aware of any guidance for how much augmentation benefits the parameterization of a network, rather, most data augmentation is either done

according to a previously-successful pattern, or via trial and error — an expensive process.

Our method requires no extra data, and does not require larger epochs due to data augmentation. Instead, *InfoMixup* chooses existing data points which are troublesome for the model and asks the optimizer to consider those data points more heavily. No augmentation strategy or schedule is necessary either — The model can simply be trained until it reaches its maximum level of convergence.

2.1.2 *InfoMixup* vs. Regularization

Regularization techniques are well known and have been common in learned models since long before data augmentation. Common regularization schemes for deep learning include dropout [26] and batch normalization [21].

InfoMixup does not rely on entropy to evenly distribute regularization throughout the network. Contemporary regularization strategies rely on entropy indirectly: Manifold Mixup [11] relies on entropy for random sampling of a β-distribution for the selection of selection of λ, and Dropout [26] relies on entropy to determine which connections are zeroed at each layer. *InfoMixup* chooses examples which are demonstrably difficult or adversarial to the model, and adjusts the label of those examples to guide the model towards a lower-confidence prediction while still producing a result. The smoother labels produced by *InfoMixup* cause models to produce smoother representations of the data, which necessarily widens the margin between classes where the model is uncertain. The outputs of an *InfoMixup* trained model display a smoother transition of confidence when moving between classes than an unregularized model.

While *InfoMixup* does not fit cleanly into either of these categories, the operation of *InfoMixup* is closer to that of a regularizer. Our theory and analysis are easily explained through the lens of regularization, and so we will refer to *InfoMixup* as a regularization strategy throughout this work.

2.2 A brief tour of *InfoMixup*

The *InfoMixup* algorithm is complex, but its operation is intuitive and grounded in rich mathematical foundations.

We use the VIB objective and variational inference layer to obtain Gaussian predictors for the class mean and variance, given a sample from the dataset, in exactly the same way as the VIB method. These predictors are collected throughout an entire epoch of training, and a mixture of Gaussians is produced which models the likelihood of a particular output from the network in the hidden representation. Using the mixture, the predictors of the next epoch are measured for their likelihood of being produced by the model based on the previous epoch's parameters. Predictors which are unlikely by a certain threshold are considered to have come from edge-case examples.

When an edge case is detected, *InfoMixup* considers where that point is located in the representation space, and finds the class which is nearest to that point, but not the ground truth class which the edge case example was drawn from. The edge case is pushed in a straight line drawn from the edge case itself toward the mean of that next most likely class by a small amount, and softmax classification proceeds on the adjusted point. The gradients from that prediction are propagated through the entire network. This process effectively gives higher weight to edge case examples in the information loss component of the VIB objective. The predictors are constrained toward the standard normal, just as in the VIB method.

We use the Mixup equation (1.1) to interpolate between the edge case example and the nearest incorrect class. The mixing is applied to both the example and the label, so that the decoder also learns a smoother mapping from the representation space to the label space. *InfoMixup* encourages the decoder to make lower confidence predictions for representation vectors in the margin between class clusters. We borrow

the hyperparameter α from Mixup for *InfoMixup* .

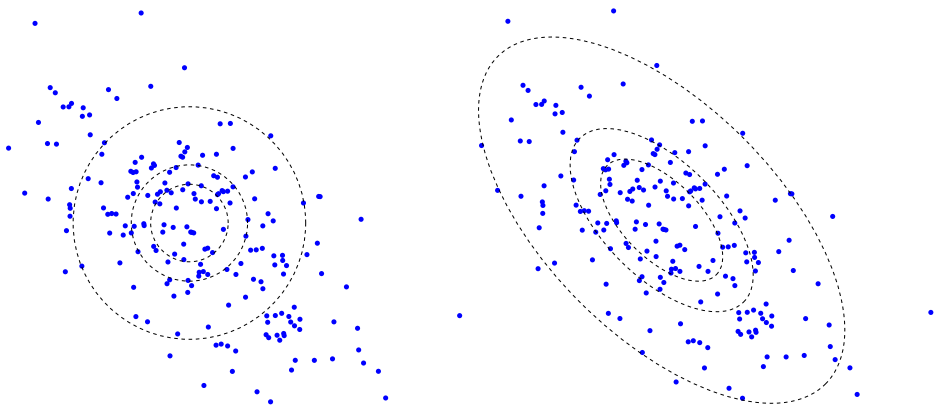
We use Mahalanobis distance for both determining which examples are edge cases, and for computing the next-nearest class. The Mahalanobis distance considers the probability density of an underlying distribution of points when measuring the distance between two points in space.

$$d = \sqrt{(\mathbf{x_i} - \mu_\mathbf{i})^T \Sigma^{-1} (\mathbf{x_i} - \mu_\mathbf{i})} \tag{2.1}$$

Where μ_i is the mean of class i, x_i is an example known to be in class i, and Σ^{-1} is the inverse of the covariance matrix for class i.

To understand the how this metric works, consider choosing two points from a space with a fixed Euclidean distance between them, and place a probability distribution with known mean and standard deviation in the space from which the points were chosen. The Mahalanobis distance between those chosen points will shrink as they are brought towards the direction of greatest variance, and will grow as they are brought closer to the mean and in the direction of least variance. The Euclidean distance between the points remains fixed, however. The Mahalanobis distance allows us to consider the underlying distribution of points when measuring distances between classes, the effect of which is that two classes whose directions of greatest variance are in the same direction appear closer together when computing the next-nearest class, and points which are near to the mean, but in the direction of least variance have similar uncertainty to points which are far from the mean but in the direction of greatest variance. This allows us to use a simple threshold for determining if a point is an edge case.

Another detail of *InfoMixup* is in how the threshold for edge cases is chosen. Intuitively, we would like to say "an example that is outside of the 95% probability mass of the cluster is an edge case". There does not exist a closed form solution for the cumulative distribution function for a multivariate Gaussian distribution, so instead



(a) Euclidian distance level lines **(b)** Mahalanobis distance level lines

Figure 2.1: Comparison of "level lines" for Euclidean and Mahalanobis distance metrics, using the same metric value. Mahalanobis distance stretches the level lines to mirror the probability density of points in a vector space.

we use an upper bound for it. The inverse quantile function of the χ^2_k distribution with k degrees of freedom gives an upper bound for the Mahalanobis distance from the mean of a multivariate Gaussian, given a desired probability of a point residing within that radius. To simplify the choice of hyperparameter, we use $\chi^2_k(p)$, where p is the hyperparameter selecting what probability mass a point must be outside of to be considered an edge case, and k is the dimension of the hidden representation.

The selection of p from a χ^2_k quantile function is not necessary for proper operation of *InfoMixup* , but it lends an intuitive meaning to the value of p.

At a high-level, *InfoMixup* is Manifold Mixup with some notable adjustments.

- *InfoMixup* does not allow intra-class mixing. Edge cases are mixed only with classes which are not their ground truth class.
- *InfoMixup* moves examples in the direction of a class mean, not of a particular example.
- *InfoMixup* selectively mixes the examples with only their nearest class, to lower the chance of a mixed example landing in a third, unrelated class.

Algorithm 1 *InfoMixup* Algorithm

```
if not training then return # InfoMixup plays no role in evaluation
init_flag ← False
for xb, yb in epoch do
    if not init_flag then
        mean_matrix ← 0
        variance_matrix ← 0
        init_flag ← True
     # Initialize class distribution tracking variables
        mean_matrix,variance_matrix←computeBatchMixtureStats(
            encoder_mean,encoder_var
        )
    else
     # Update class distribution tracking variables with data from current batch
        mean_matrix,variance_matrix← computeBatchMixtureStats(
            concat(encoder_mean,mean_matrix),
            concat(encoder_var,variance_matrix)
        )
    edge_cases ← filter(
        not withinInterval(xb,mean_matrix,variance_matrix)
    )
    class_dists ← mhDist(edge_cases,mean_matrix,variance_matrix)
    new_xb,new_yb ← bipartiteMixup(
        edge_cases,yb,mean_matrix,variance_matrix,λ
    )
```

Several approximations are made to speed up computation. The most impactful approximation is the mixture of two Gaussian distributions being approximately Gaussian. If we were to retain all the predictions for an entire batch, and then at the end of each epoch, compute the Gaussians of each class, the memory usage of *InfoMixup* would scale infeasibly for any modern applications. Since we are primarily concerned with the "perimeter" of the class Gaussian, and not with its exact density function, we instead assume that each batch is similar on a per-class basis, and compute the class Gaussian as a Gaussian whose mean is the mean of the batch means and whose variance spans the widest gap between the batch distributions. The benefits for computational resources are twofold: if the batch is sufficiently large that the likelihood of seeing all classes in a single batch is high, then we can assume that

Algorithm 2 `withinInterval` function

```
function WITHININTERVAL(sample_batch,ground_truth_labels,interval)
    dist ←MHDIST(
        sample_batch,mean_matrix[ground_truth_labels],
        variance_matrix[ground_truth_labels]
    )
return  interval > dist
```

Algorithm 3 `bipartiteMixup` procedure

```
    procedure BIPARTITEMIXUP(
         sample_batch,ground_truth_labels,mean,variance, λ
    )
        new_samples, new_targets ← 0
        for t in UNIQUE(targets) do
            others ← SET(targets) − t
            for other in UNIQUE(targets) where other ≠ t do
        # MH-Distance of sample from all other classes
                nearest_class ← ARGMIN(MHDIST(
                    sample_batch,mean[others],variance[others]
                ))
        # Mixup equation with non-intra-class constraint
                new_samples[t] ← sample_batch*λ + (1 − λ)mean[nearest_class]
                new_labels[t] ← ground_truth_labels*λ + (1 − λ)nearest_class_label
    return new_samples,new_targets
```

after a single batch all mean and variance parameters have been initialized. This allows us to parallelize computations throughout *InfoMixup* . We believe this tradeoff of precision for significant decrease in computational resource requirements is worthwhile and minimally affects the performance of *InfoMixup* . We also constrain the covariance matrix of each class to be diagonal as in VIB. Doing so greatly simplifies inversion of each covariance matrix when computing the Mahalanobis distance under each class.

Algorithm 4 `computeBatchMixtureStats` procedure

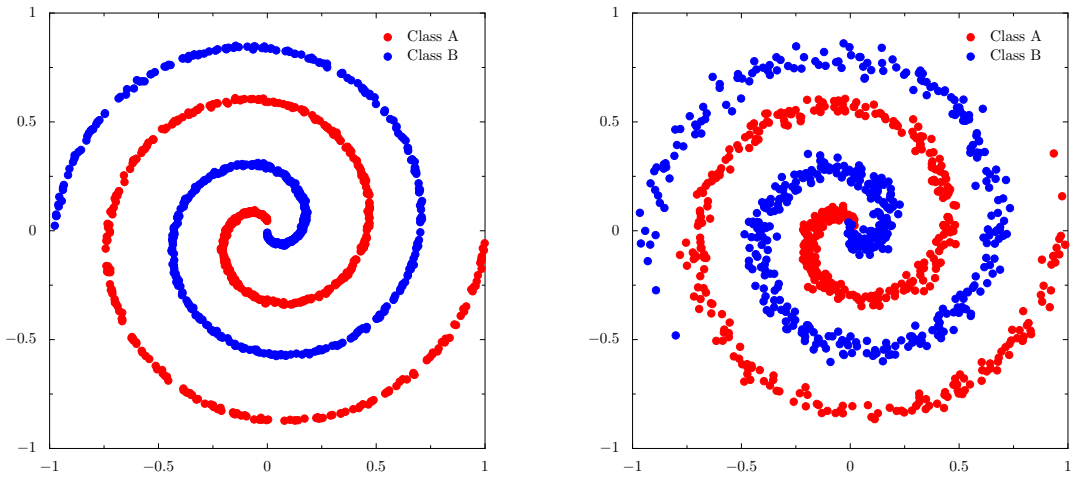
```
procedure computeBatchMixtureStats(
        batch_mean,batch_variance,targets
    )
    stats_accum ← 0
    for t in unique(targets) do
     # Compute a Gaussian that is similar to the mixture of two Gaussians
      stats_accum[t] = GaussianMixture(
          batch_mean,batch_variance,mean_matrix,variance_matrix
      )
    return stats_accum
```

Chapter 3

Empirical Methodology and Analysis

3.1 Visual Exploration: Spiral Dataset

As an initial exploration of regularization, we take inspiration from Verma et. al. [11] and visualize the effects of various regularizers on a common two-dimensional problem: the intertwined spirals dataset. This problem is nonlinear in Cartesian coordinates, and so cannot be learned by a linear classifier, but it is easily learned via the kernel trick or with a neural network, and due to the low dimension of the problem, trains quickly without hyperparameter tuning.

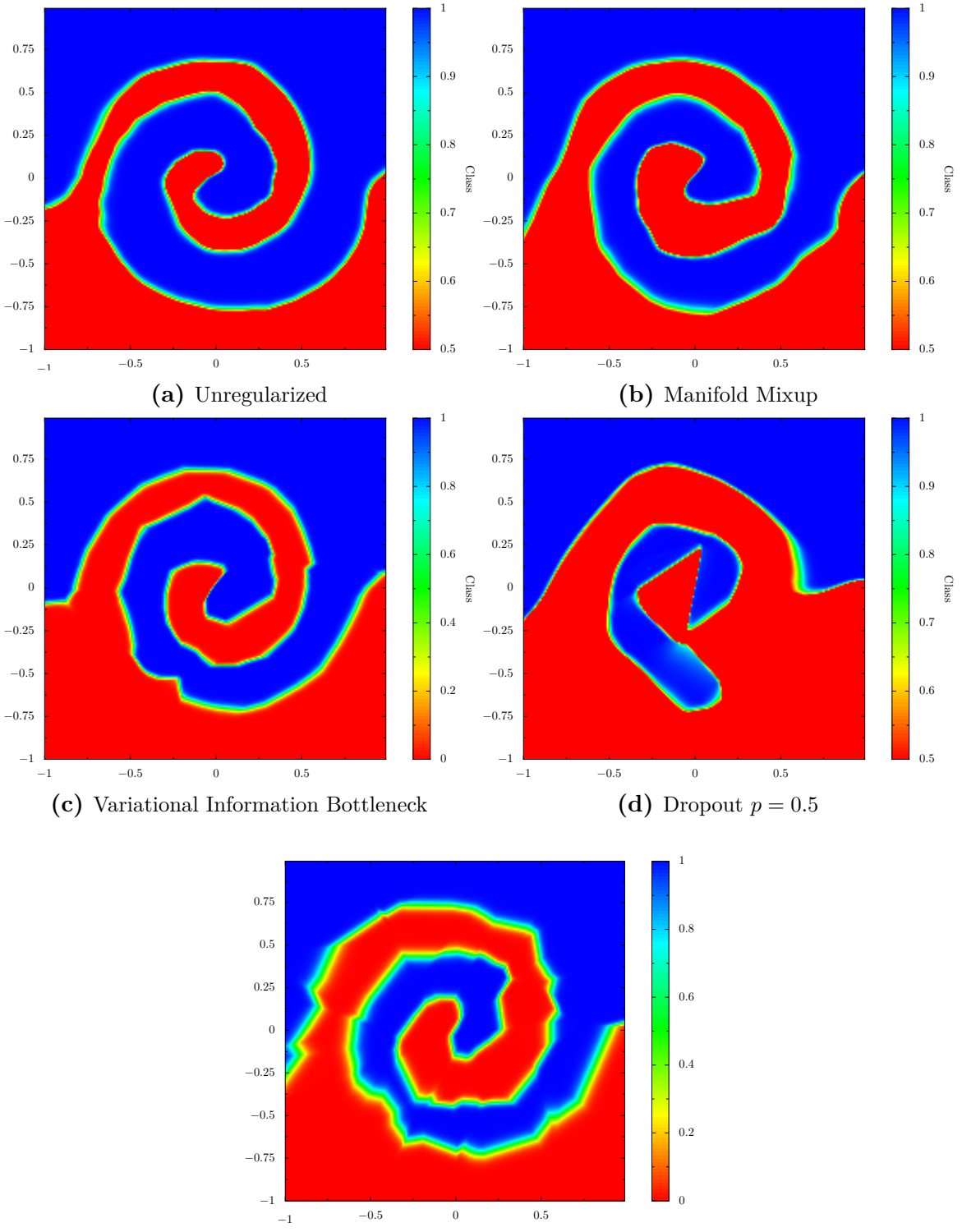


(a) Train **(b)** Test

Figure 3.1: Spiral Dataset

We evaluate all regularization strategies with a custom feed-forward network, trained for 2000 epochs with the regularization hyperparameters set by intuition and without tuning. Thus, our results are non-optimal for each regularizer, but the intent is to simply visualize the differences between each, and not to effectively regularize the spiral problem.

The spirals used as training data are normally distributed along each curve. After training, we evaluate the network over a range which contains a range of points wider than those in the training set and plot the model prediction as a color at each point, to display the gradient of confidence between the two classes. An ideal result from a given regularizer would be the average of a linear gradient between every possible pair of points with one point in each pair drawn from each class. In practice, we expect that a wider margin of low-confidence predictions between ranges of high-confidence will be displayed in better-regularized models, ideally without sacrificing accuracy in the high-confidence regions.



(a) Unregularized

(b) Manifold Mixup

(c) Variational Information Bottleneck

(d) Dropout $p = 0.5$

(e) *InfoMixup*

Figure 3.2: Comparison of Various Regularizers on the Spiral Dataset

Most regularizers appear to have a similar effect over the unregularized network

in Figure 3.2. The unregularized network discovers the "simplest" function, which corresponds nicely to the results of Hoffer et. al. [27]: a network with higher representational capacity learns a less complex representation of the data than a network with lower representational capacity, if given enough training time.

Manifold Mixup, combined with Input Mixup in this comparison, appears to have a smoothing effect, but does not make the representation appear more jagged, which may imply that its regularization strength is relatively low in comparison to the others. The margin is wider where the data points are naturally more spread out, and the margin appears to be wider everywhere compared to the unregularized baseline. This is as one might expect, given the random per-batch mixing performed by Mixup and Manifold Mixup.

Dropout is of particular interest. The margin regions are actually narrowed for Dropout, so much so that the network incorrectly predicts certain sections of the blue spiral. This is likely due to the ensemble effect of Dropout: multiple smaller networks are effectively created via random selection of nodes whose output is canceled. In this case, the resulting smaller networks may not have sufficient representational capacity to remain able to map the data correctly.

VIB much more drastically regularizes the network. The learned function has jagged edges, which may indicate either a reduction in representational capacity, or a significant and perhaps overbearing reduction in the amount of information retained by the network. Suppose the unregularized network has learned a sinusoidal function to represent each spiral: Then the VIB-regularized network appears to have learned a piecewise-linear approximation to that function. In low dimensions, this result is to be expected, since the optimal function to be learned is likely to be simple. In higher dimensional spaces, the effect of regularization may be to flatten jagged or discontinuous boundaries.

InfoMixup appears to perform the best. While it causes jagged edges similar to

Table 3.1: Comparison of Margin Between *InfoMixup* and Standard Training

Model	Mean Margin	Median Margin
Unregularized	0.4362	0.3986
InfoMixup	0.3694	0.4235

VIB, the margin region between classes is significantly wider in all locations compared to any of the other regularizers. Most notably, *InfoMixup* widens the margins evenly across the entire dataset, but does so in a fashion that reflects the separation between the classes at each point. In the training data, the two classes do not touch at the origin, however in the testing data, there is overlap near the origin. While all the regularizers have some margin in this area, only *InfoMixup* captures the overlap with a margin wider than that of the unregularized network. This implies that *InfoMixup* may perform well in situations where there are naturally neighboring or nearly-overlapping classes in the true distribution.

We consider the numerical width of the margin as well, as further evidence of the strength of *InfoMixup* as a regularizer. We consider every point in the test set as a complete bipartite graph, and for every edge sample 25 points in the data space along the line segment whose endpoints are in each class. The point at which the confidence of the model drops below 90% for the nearer endpoint is considered as the boundary of the margin region, and we then compute both the mean and the median length of this margin.

The margin is notably not widened on average by *InfoMixup* , which is obviated when observing the decision boundary. After our initial figures were created for Figure 3.2, we were unable to re-create an unregularized model which learned the true underlying function in an updated version of PyTorch. Instead, all trained models memorized blobs of points in the training distribution, which led to confusing results in margin width computation.

Since the results of the spiral dataset are not tuned, these comparisons are in-

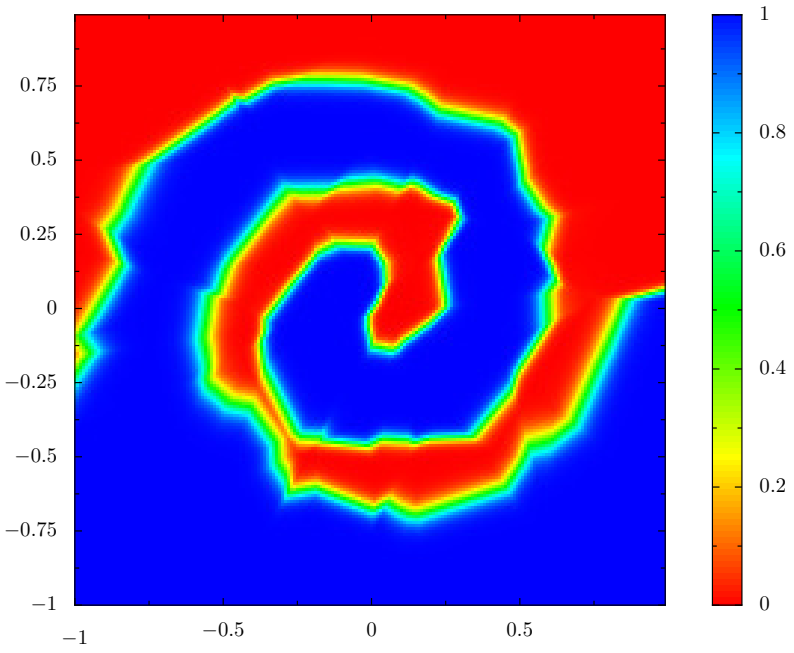


Figure 3.3: *InfoMixup* Decision Boundaries used in Margin Measurement

sufficient to perform a performance ranking. We move on to more realistic problems before making further comparisons.

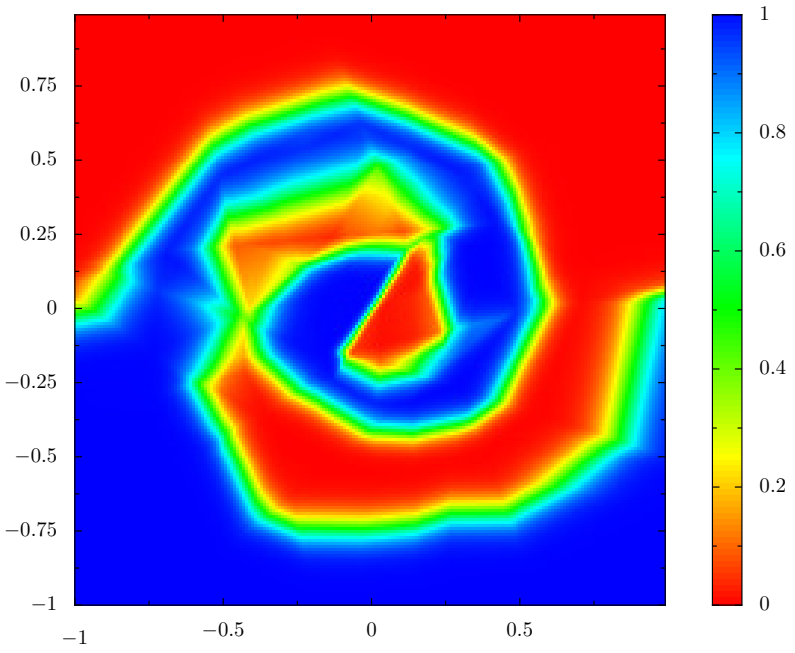


Figure 3.4: Unregularized Decision Boundaries used in Margin Measurement

3.2 Proof of Concept: *MNIST* and *LeNet5*

3.2.1 Regularizer Comparison

As an introductory exercise, we chose to develop *InfoMixup* on MNIST, a popular image recognition task based on small images of handwritten digits. A modern version of LeNet5 was used as the model, with ReLU activation. As a baseline, we trained the LeNet5 model for 30 epochs at a batch size of 256 images with Adam. Additional information about the training parameters for the baseline LeNet5 model are available in Appendix A. We then attacked the baseline model with a variety of state of the art methods. We obtained accuracy decay with respect to attack strength, and found that even weak attacks were able to easily fool an unregularized network, which is consistent with the findings of many other authors [11] [23]. We chose to use the Fast Gradient Sign Method [1] for our proof of concept, since it is simple, fast, and

effective. Additionally, a defense which is bested by FGSM is unlikely to generalize to stronger attacks.

Each attack yielded a corpus of new data in which every image is adversarial to a baseline network. In a classification task where the test data is drawn exclusively from this corpus, the baseline network acheieves 0% accuracy. We now look at methods through which we can improve classification accuracy on this dataset without substantially affecting the performance on clean data. Various state of the art methods which claim to improve adversarial accuracy were tested as a survey of existing methods to which our own novel method would be compared.

In order to visualize the operation of *InfoMixup* , we modify the LeNet5 architecture in Appendix A to have a two-dimensinoal encoder and representation, but still have ten output classes. We train this constrained model identically to a normal LeNet5 and then plot its embedding of the test set. We study both the actual embeddings of the test set with and without incorrectly-predicted data points, and the estimated class distributions produced by *InfoMixup* . We sample 1000 points from each class to draw its estimated density function.

Observation of figures 3.5 and 3.6, we can observe that the predicted class density functions are similar to the embedding produced by the network. Note that in testing mode, *InfoMixup* is switched off, allowing the network to operate without guidance from the estimated density functions. This observation aligns with the intended operation of *InfoMixup* , and demonstrates that it is able to estimate the class densities in the embedding space, which enables the selective mixing of high-information vectors.

Comparison of figures 3.6 and 3.7 shows that the margin between distributions contains some points which the classifier fails to predict, and that these points lie in the regions where the margin is already narrowest. This observation is concerning in that the VIB objective coerces all distributions towards the standard normal distribution; the effect of this is that the optimization objective tries to make all the class

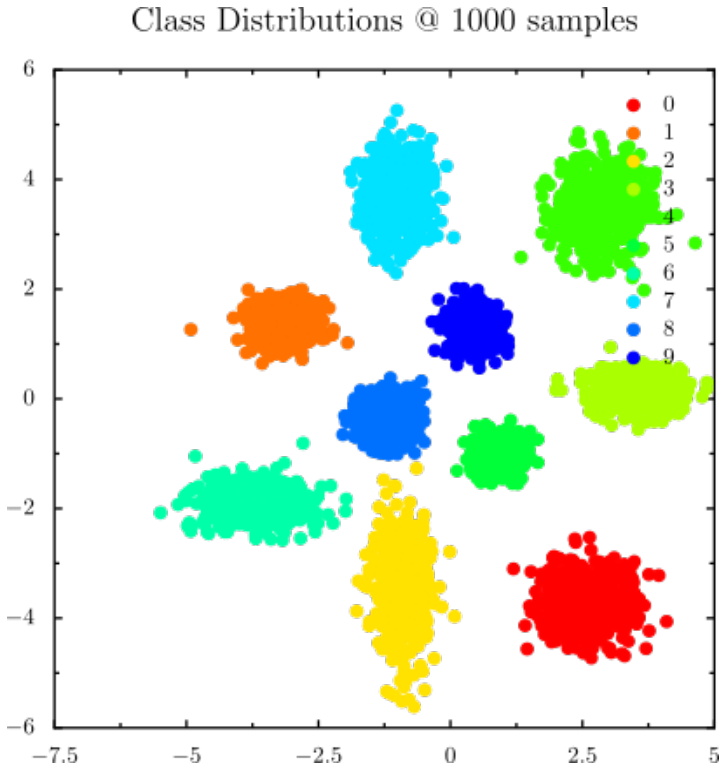


Figure 3.5: Estimated class distributions for a 2D embedding of MNIST.

distributions as close together as possible (minimizing the number of bits required to represent them), while also making them linearly separable. Some examples must fall in the margin region and be misclassified then, since no embedding is perfect.

In a higher-dimensional space, this problem may be less severe. The VIB authors achieve excellent performance on ImageNet and do not report a significant problem with the resulting embedding.

3.2.2 Accuracy Decay

We show that *InfoMixup* provides significant improvement in adversarial robustness to various attacks over other methods. Our methodology is simple and consistent. Every method is applied to the same LeNet5 model as seen in Appendix A. We use a training budget of 30 epochs, no data augmentation, Adam optimization with an initial learning rate of 0.1, a stepped learning rate decay of 0.1 every 5 epochs,

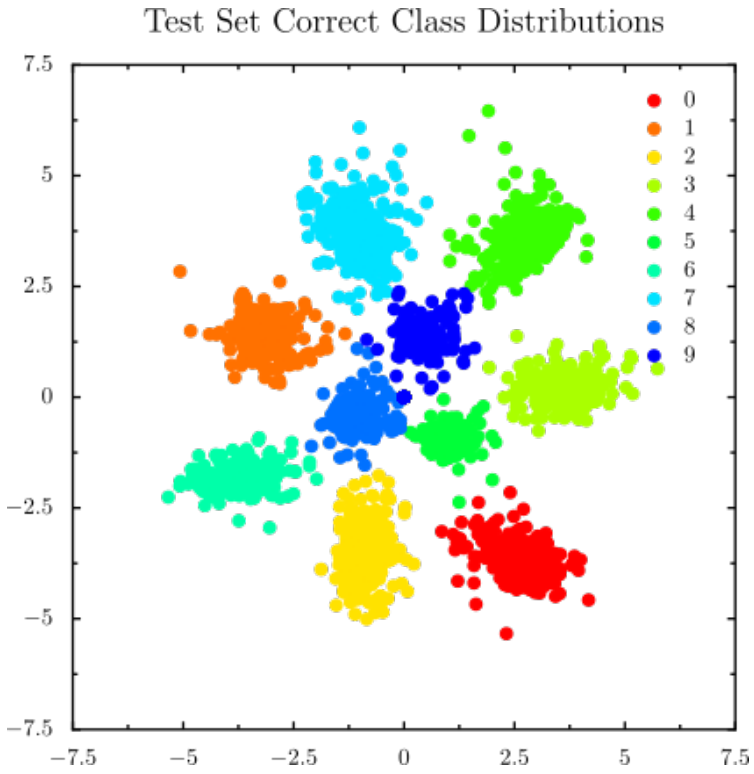


Figure 3.6: Embeddings of the MNIST test set produced by an *InfoMixup* - regularized LeNet5, with incorrect predictions removed.

cross-entropy loss, and batch size 256 for this experiment. We then consider the hyperparameters proposed by the authors of each method, and sweep through them, running 5 independent and identical training sessions for each hyperparameter. The resulting models are attacked with various methods, sweeping through epsilons to show how the accuracy of each resultant model decays with respect to the strength of the adversary. We then average the performance of all 5 models and compute the area under the accuracy decay curve using a trapezoidal integration. We use this area as a measure of overall robustness. We proposed this method of measuring robustness independently, however have since seen it used by [4], which lends credibility and precedent to this method. We independently select the best-performing hyperparameters for each method with the intent of giving each method the best possible chance in the competition, as our models are different in certain cases than those of the original authors. Additionally, some methods do not have significant investigation

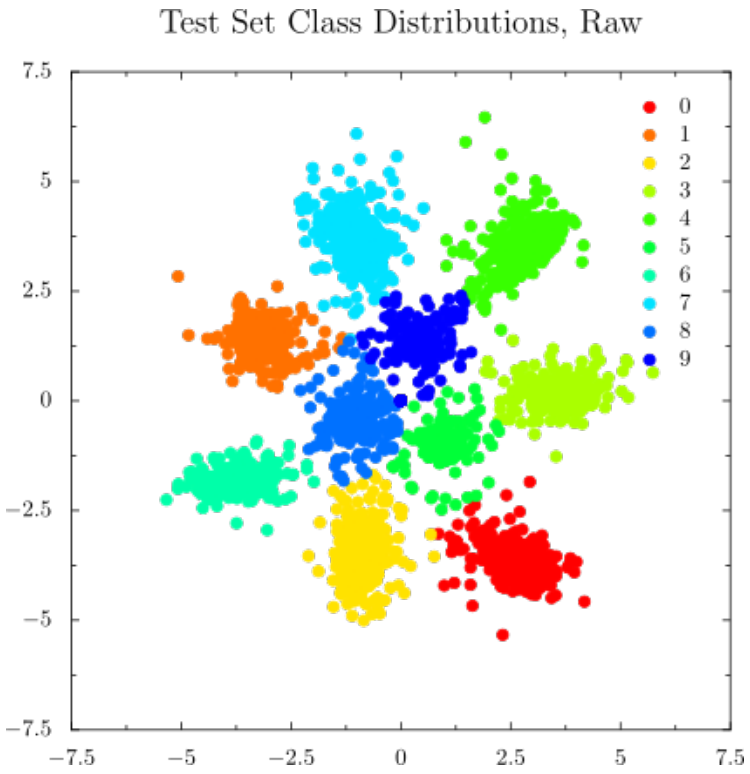


Figure 3.7: Embeddings of the MNIST test set produced by an *InfoMixup* -regularized LeNet5.

regarding adversarial robustness, so we opt to perform this work ourselves.

The best performing hyperparameter based on the average area-under-decay-curve for each model is selected, and we then compare each champion method against all other champions of other methods.

The same corpus of data through which the aforementioned methods were evaluated were also used for Adversarial Training [5]. The adversarial samples generated against the baseline network from all of the attacks were combined with the clean data samples to create a broader dataset, and a model identical to the baseline network was trained from scratch using the augmented dataset until convergence was reached.

This analysis yielded promising results, showing that *InfoMixup* consistently outperforms all other tested methods when optimal hyperparameters are selected for all methods. We observe that even the worst-case performance in five runs is notably more robust than the average performance of contemporary methods.

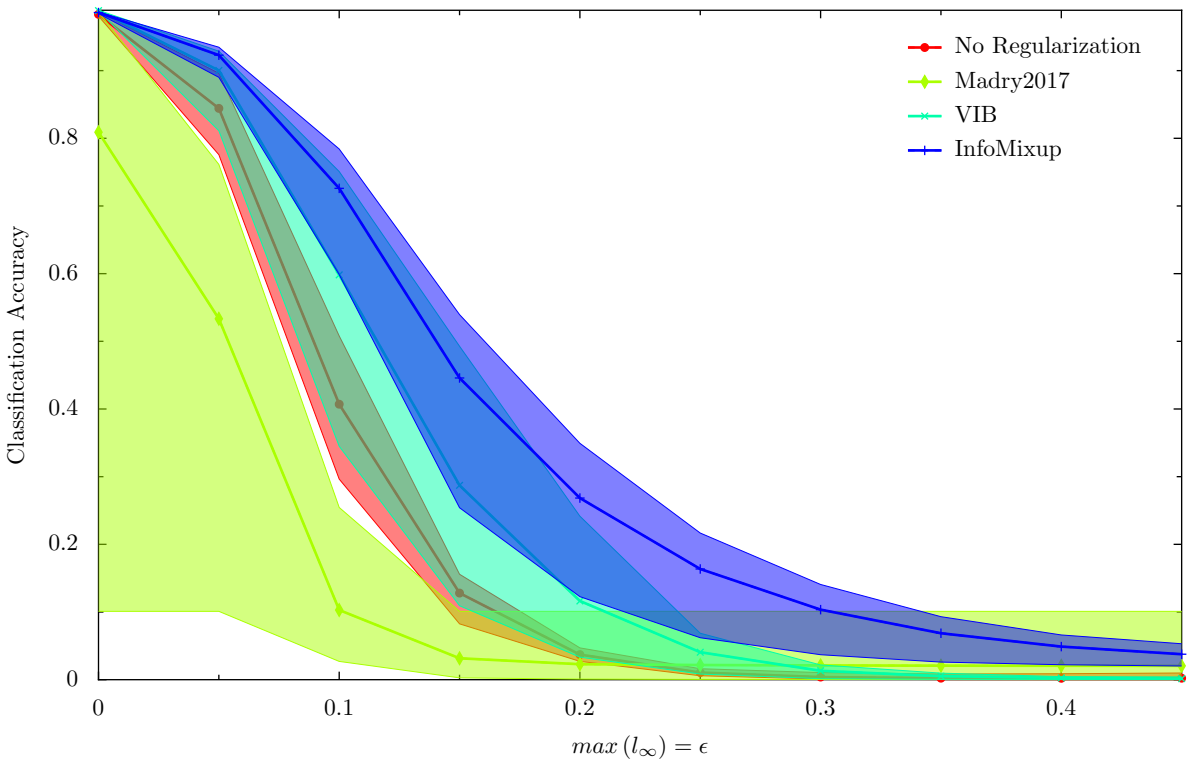


Figure 3.8: Accuracy Decay on FGSM for Baseline, VIB, Adversarial Training, and *InfoMixup* . We elide ManifoldMixup and Mixup due to low performance for readability.

The poor performance of Adversarial Training is reason for concern. A theoretical model of adversarial robustness would indicate that Adversarial Training is capable of making an arbitrarily strong network, assuming that there is sufficient representational capacity in the model and that a sufficient breadth of adversarial data is available for training. We believe the poor performance is due to our use of a different model with significantly less representational capacity than the model used in [5]. The average clean accuracy is significantly lower, denoting a failure to converge to the same degree as the other methods, all of which have nearly-perfect performance in clean accuracy. In this experiment, we opted to change only the regularizer, and no other training parameters, for the clearest possible comparison.

3.3 Methodology

3.3.1 InfoMixup Does Not Obfuscate Gradients

A common problem with defenses to adversarial attack is that they function by obfuscating the gradient signal used by a white-box attacker [3]. This defense is easily circumvented and makes the defense meaningless. To show that *InfoMixup* does not improve robustness by obfuscating gradients, we compare the performance of a black-box attack at the same epsilons as a white-box attack. BoundaryAttack and FGSM were chosen as the black-box and white-box methods respectively, and we observe that FGSM has a much higher success rate at fooling the network for every nonzero ϵ.

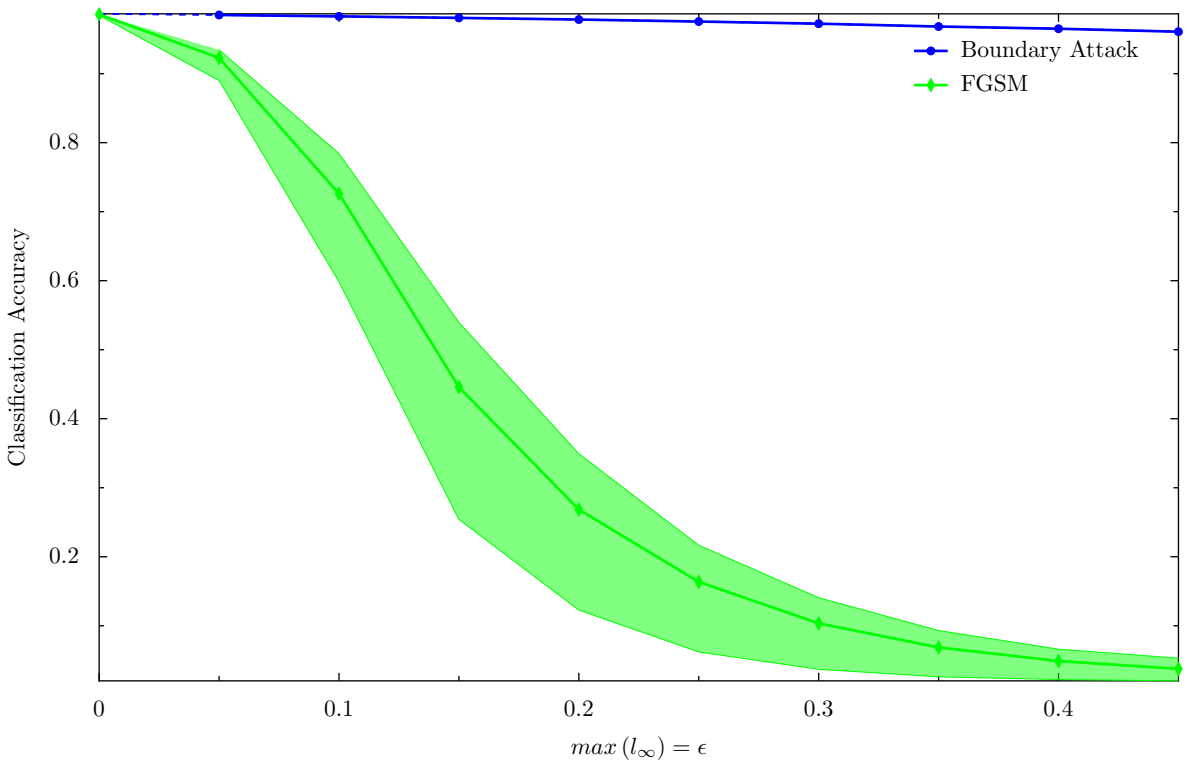


Figure 3.9: *InfoMixup* vs. FGSM and BoundaryAttack for the same epsilons. Per [3], the failure of BoundaryAttack and success of FGSM is a strong indication that *InfoMixup* does not obfuscate gradients.

The highest-performing hyperparameters from *InfoMixup* were used to train mod-

els on MNIST for this test. We then attacked the same networks with both FGSM and BoundaryAttack for epsilons ranging from $0.0 - 0.5$. We observe that indeed FGSM performs significantly better. The lack of errorbars on the BoundaryAttack accuracy decay in Figure 3.9 is not a mistake — BoundaryAttack only succeeded in finding adversarial examples in one out of the five models attacked, whereas FGSM found examples in all five and was able to fool the network for sufficiently large distortion.

3.3.2 *InfoMixup* and Transferability

Under the lens of adversarial examples arising from a data and optimization problem, where naturally occurring samples do not cover enough volume in the data space to allow for robust generalization, *InfoMixup* should cause significant improvements in robustness in the same fashion as VIB. Although it is impossible to create a perfectly secured network, a *InfoMixup* reduces the quantity of irrelevant information about the data space retained in the intermediate representation, thus decreasing the dependence of the classifier on brittle features.

In our testing however, we did not observe that *InfoMixup* defends against transferred adversarial examples. We choose a LeNet5 baseline, trained without any regularization or data augmentation, and attack it with FGSM. The resulting examples we then use to evaluate the best-performing *InfoMixup* model, which has not seen any adversarial data. The resulting performance was indistinguishable from that of the unregularized model.

This result is reason for concern about the performance of *InfoMixup* . Carlini et. al. suggest that strong adversarial defenses should be able to defeat the transfer of adversarial examples from one network to another [6]. This claim is rational, since transfer of adversarial examples is an effective side-channel attack that can be used to unlock black-box methods for attacking a model in a known problem domain when gradients are not available. *InfoMixup* does not appear to provide any

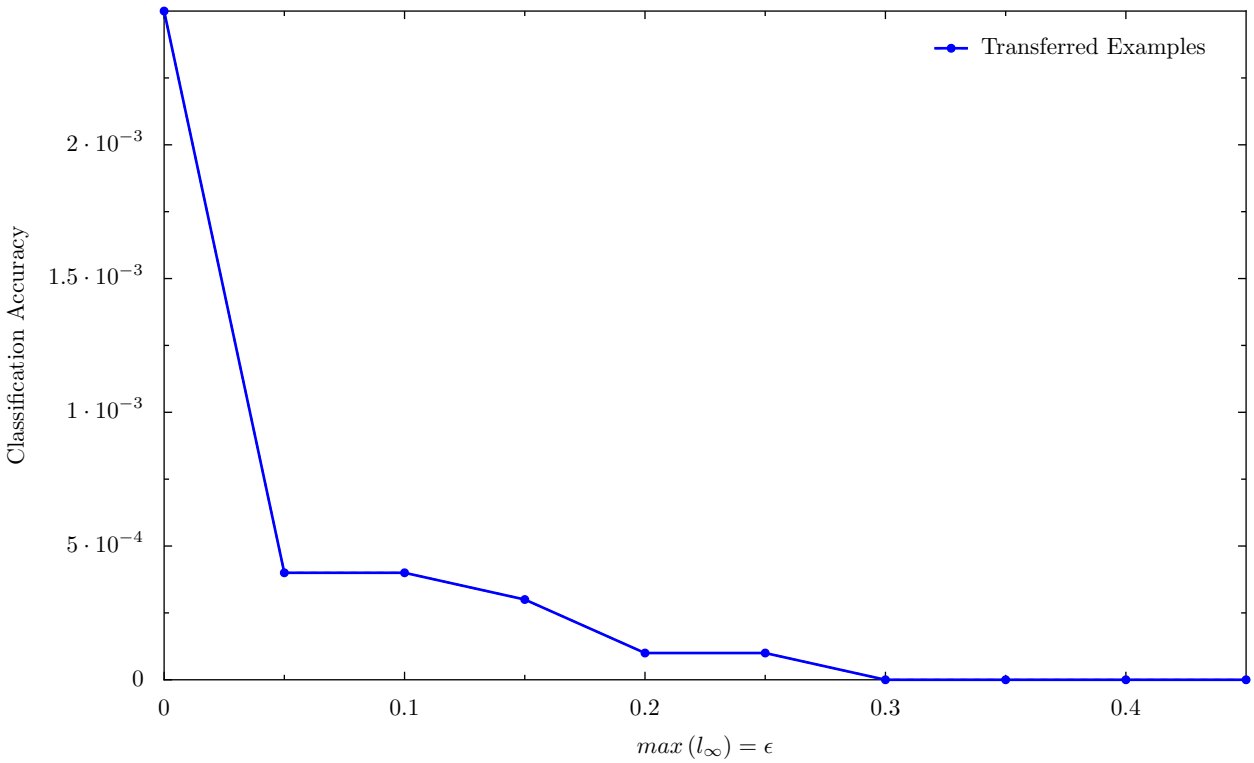


Figure 3.10: *InfoMixup* performance vs. transferred adversarial examples. Please note the graph scale on the dependent axis.

defense against transfer.

Another question is raised from this result: "If *InfoMixup* reduces the mutual information between the data and hidden representation, but does not resist transfer attacks, is it effectively performing this task?" We believe that more investigation is necessary in this domain.

In the theory of Ilyas et. al. [8], adversarial examples arise from the presence of features which correlate strongly with the label distribution within some radius, and under a certain measure away from the coordinates of an input sample in the data space, but which cease to provide correlation outside of that radius, with respect to the perturbation distance after a certain point. Thus, these features suffice for standard classification and are useful to the model in standard training, so they are incorporated into the model's "understanding" of the mapping between data and labels.

In order to perfectly defend against transfer examples, the model should have no dependence on these brittle features whatsoever. Given that the source of these brittle features is unknown and that the dimension of the subspace of brittle features in a particular dataset may not be possible to enumerate, nor the amplitude of those features possible to determine without the use of numerical optimization techniques to create perturbed samples, defending against transfer examples is extremely difficult.

Our experiment concerning transfer examples is significantly more difficult to obtain favorable results on than that of contemporary literature. We allow transfer examples to be generated with an attack which is different than the one used during training of the secured model, and additionally allow the use of a different metric. This allows an attacker to craft examples which use the brittle features in a different way than the examples used during training, which in turn allows the model to be fooled with ease.

3.4 CIFAR10 and *PreActResNet18*

With our initial experiments on MNIST showing promise, we turn our attention towards gathering empirical evidence supporting our theory in more complex and realistic scenarios. A popular next step from the MNIST dataset is CIFAR-10, a collection of 60000 32×32 pixel color images with ten labels. Compared to MNIST, a significantly larger proportion of the image plays a role in the classification compared to the black-and-white images in MNIST, which are nearly binary and thus have a much less complex decision boundary [28]. Considering this, CIFAR-10 generally needs larger networks to solve as compared to MNIST, and so from this experiment we can also learn about the behavior of *InfoMixup* on more complex convolutional neural networks.

Our experiments with CIFAR-10 use PreActResNet-18 configured with a single fully-connected linear layer. ResNets are generally easier to train due to the residual

paths allowing gradients to propagate backwards without vanishing, and having generally lower parameter counts than denser network architectures like SENet or VGG for similar representational capacity. The choice of a pre-activation ResNet is guided by their generally better performance over post-activated ResNets [22] of similar representational capacity. The authors of Manifold Mixup [11] use a PreActResNet-18 for their experiments with CIFAR-10 and CIFAR-100, which we use as a point of comparison for our own results. We use the training parameters detailed in [11] as a baseline, changing only the regularizer or the adversarial hardening scheme during training.

As a proof of concept, we train PreActResNet-18 for 1200 epochs using SGD with momentum of 0.9, Nesterov gradients, weight decay of 10^{-4} and a learning rate of 10^{-1}. We use a batch size of 100, to exactly match the baseline of Verma et. al [11]. The learning rate is multiplied by 0.1 at epochs 400 and 800, and we use binary cross entropy for the loss function. Binary cross entropy encourages the network to be highly selective of a single class by considering the cross entropy of the estimated and ground truth labels as well as the cross entropy of the inverse of the predicted and ground truth labels. The selectivity of binary cross entropy boosts performance in one-versus-all identification tasks.

We then trained a PreActResNet-18 with Manifold Mixup, using the best-performing parameters from its authors [11]. The authors claim 97.5% accuracy using $\alpha = 2$ with Manifold Mixup on PreActResNet-18. However, in our attempt to recreate this result, PreActResNet-18 reached an accuracy of 93% after 1200 epochs using identical parameters. This significant reduction in test set accuracy is reason for concern, however, a single training run was completed to verify these results, rather than a large batch from which the best performing model was selected.

With our baselines established, we run a parameter sweep of *InfoMixup* parameters α and β, covering 77 combinations drawn from the parameter sweeps in [11], [23]. We

record the clean accuracy on the test set for each set of hyperparameters after 1200 epochs, this time using an adjusted learning rate of 10^{-4}, which compensates for the increased magnitude of loss applied by *InfoMixup* 's objective function. Large values of β in particular need this adjustment, as the KL-divergence between the predicted and ideal class distributions is quite high in early epochs.

Upon completion of the grid search of hyperparameters, we were greeted with strikingly low top-1 accuracy on clean test data for all tested combinations, peaking at 60% with $\alpha = 2$ and $\beta = 10^{-8}$. These selections of hyperparameters that were automatically determined by the grid search align well with the selections made for CIFAR-10 from the constituent methods which *InfoMixup* is related to. An α value of two is the same as the highest performing α in ManifoldMixup, and $\beta = 10^{-8}$ is similar to the value chosen for MNIST in VIB, but is larger: we expect this due to the higher dimensional complexity of CIFAR-10 as compared to MNIST. This result suggests that *InfoMixup* does not generalize to complex problems despite promising results on MNIST. However, there is theory to suggest that CIFAR-10 does not contain enough information for robust generalization. The dimension of images increases significantly from MNIST and the images are no longer nearly binary. Schmidt et. al. [28] suggest that it is not possible to learn a p-robust classifier on CIFAR-10 for arbitrary p, with p being a measure of adversary strength. Since *InfoMixup* constrains the information content of the representation, we consider moving to a more complex dataset to see if *InfoMixup* performs better there.

3.5 TinyImageNet with *PreActResNet18*

We choose TinyImageNetfor our more complex dataset with which to test the generalization of *InfoMixup* . TinyImageNet consists of 100k images with a resolution of $64 \times 64 \times 3$, 500 images per class, with 50 from each class reserved for testing and validation. The test and validation sets for TinyImageNetare identical.

TinyImageNetis used by the authors of Manifold Mixup, which aids us in reducing the number of training sessions we must complete. Additionally, hardware and time constraints prevent us from testing on ImageNet-1k and others. We obtain several baselines, one from the authors of Manifold Mixup, and another from the course notes of CS231n from Stanford University, the course for which TinyImageNetwas created. Time constraints made testing with the parameters of Manifold Mixup difficult, so we attempted only a baseline unregularized training with this method, to determine if it would converge. We did not achieve convergence, with accuracies in the 3–4% range for this baseline.

We also attempted a method using Super Convergence [29], which uses large learning rates annealed quickly to train models in a fraction of the epochs required by standard training. Super Convergence also did not achieve a functional baseline, nor did any of the regularization techniques used here provide any increase in performance. All models trained did not learn whatsoever and stalled at random accuracy. Due to time and resource constraints, no further testing has been performed on larger datasets.

Chapter 4

Analysis of Results

4.1 Observations about Representational Capacity

When developing models to solve complex problems, one needs to select a model with sufficient representational capacity to solve the problem at hand. One might select LeNet5 for a handwriting recognition task similar to MNIST, its original problem domain, but a model on the scale of LeNet5 is unlikely to converge on ImageNet or similar classification tasks. Although defining model capacity and measures thereof it is outside the scope of this effort, it is important to consider the effect of representational capacity on robustness and generalization. It is well known that models with a large number of parameters require a large number of epochs to train, but that their resulting representations are likely to be simpler and to generalize better than a model which has a smaller number of parameters on the same dataset [27]. This is in contrast to the traditional view in statistical analysis, where the simplest model capable of representing the data will also generalize the best. This phenomenon of deep neural networks is not well understood, but it is nevertheless common to see models with more parameters than data points in large-scale machine learning. Increasingly popular methods involve convolutional layers which have a wide expansion ratio, such as MobileNets and WideResNet, prized for their parallel and efficient operation with notable performance benefits as well.

We seek to determine whether the additional dense linear layer used in VIB and

InfoMixup increases the representational capacity of the model in a way which affects adversarial robustness. This layer is used to provide a linear transformation from the column space of the hidden layer before the variational encoder to a space which is optimized by the VIB objective to have a probability density of points corresponding to classes in Y, one cluster for each class. Madry et. al. [5] claim that increasing model capacity provides non-negligible improvements in adversarial robustness. We aim to rule out the possibility that the advantage of information-guided approaches to improving robustness lies in additional representational capacity.

A theoretical outcome is simple to obtain. The constraint on the output of the variational layer is given by Equation 1.2, specifically the latter term multiplied by the constant β. The behavior of the variational encoder is deterministic when $\beta = 0$, since the objective at that point becomes identical to cross-entropy loss.

$$J_{IB,\beta=0} = \frac{1}{N} \sum_{n=1}^{N} \mathbb{E}_{\epsilon \sim p(\epsilon)} \left[-\log q \left(y_n | f \left(x_n, \epsilon \right) \right) \right] \tag{4.1}$$

The reparameterization trick is used to insert Gaussian noise such that the output of the dense linear layer within the variational layer becomes the parameters to a Gaussian distribution in the dimension of the representation space. If the source of noise is set to a constant value, then the variational layer becomes fully deterministic, and will learn a function:

$$f\left(x\right) = \mathbf{W}\overrightarrow{x} + \overrightarrow{b} \tag{4.2}$$

Where $\overrightarrow{b}$ contains the linear offset that would be learned with no constant offset added by the reparameterization, plus the offset added by reparameterization. Sampling and then averaging the output of the variational layer will still yield the same result, since all sampled points will be identical. Note, however, the lack of an activation function. The variational layer has no activation function applied, since its outputs are distributions in the representational space. Thus in this instance the

layer is only a linear transformation, and has little or no effect on the representational capacity of the network.

However, consider the case where $\beta = 0$, but the noise source is allowed to be Gaussian, and the noise is shifted by the location parameter and scaled by the variance parameter predicted by the variational layer. The function learned by the variational layer is now:

$$f(x) = \mathcal{N}(\mu_{var}, \sigma_{var}) \tag{4.3}$$

The normal distribution predicted by the variational layer to correspond to a given input is completely unconstrained with $\beta = 0$, so there is no restriction on the information content of the representation, since the parameters are allowed to vary freely. However, since the resulting distribution is sampled to produce a point in the representational space for the decoder to classify, the variational layer still cannot add representational capacity to the degree of a dense linear layer, since noise is still injected via the sampling operation.

Lastly, we consider the case where $\beta \neq 0$, and the noise source is allowed to be Gaussian. This is the case in normal operation of the variational layer during training. In this mode, the distributions produced by the variational layer are now penalized for having a nonzero KL-divergence against the standard normal. The effect of this is that the distributions are kept arbitrarily close to the origin, reducing the separation between distributions as compared to the unconstrained case. The output of the layer is still sampled in order to produce the point in the representational space for the decoder, thus the variational layer cannot act like a representational layer.

We also attempt to empirically demonstrate this fact. Three variations of LeNet5 are trained identically for this experiment, using the best-performing *InfoMixup* hyperparameters. The VIB-related hyperparameter of *InfoMixup* is data-dependent [23] and so requires no re-tuning for a new model. The Manifold Mixup parameter of *InfoMixup* is less clearly dependent on data or model, however, it is intuitive to

understand its value in terms of the dimension of the hidden representation. Higher dimensional spaces have lower point densities, and thus two data points in that space naturally appear further apart. Since we do not change the size of the representation layer in this experiment, we also do not re-tune α.

In addition, we train the MNIST model of [5] without regularization, and an unmodified LeNet5 without regularization, to back up the claims of [5] regarding model capacity and robustness. Indeed, we observe that increased model capacity of the model provided by Madry et. al. results in significantly lower validation loss than that of our LeNet5 implementation, which is consistent with the findings in [5].

The three LeNet5 variants we trained have one less layer, no changes, and one additional layer. In each model we maintain a taper in dimension as the layers progress towards the output, and do not allow any layers to have the same size, thus maximizing the ability of each layer to increase representational capacity. We observe that the model with an additional layer shows noticably lower validation loss, which is as we expect. The smaller model and the default show validation loss which are negligibly different, which raises a question about the continuity of performance increase with respect to network depth. We believe that this result does show that the representational capacity of information guided approaches does not lie in the linear layer within the variational encoder. If that were to be the case, we would expect to see that a model with one less layer would have higher validation loss. Instead, the representational capacity of a smaller model is actually increased significantly via the information-guided regularization.

Both networks were trained identically, and for 200 epochs, which was intended to cause extreme overfitting. However, neither the larger nor the smaller network overfit when trained with *InfoMixup* .

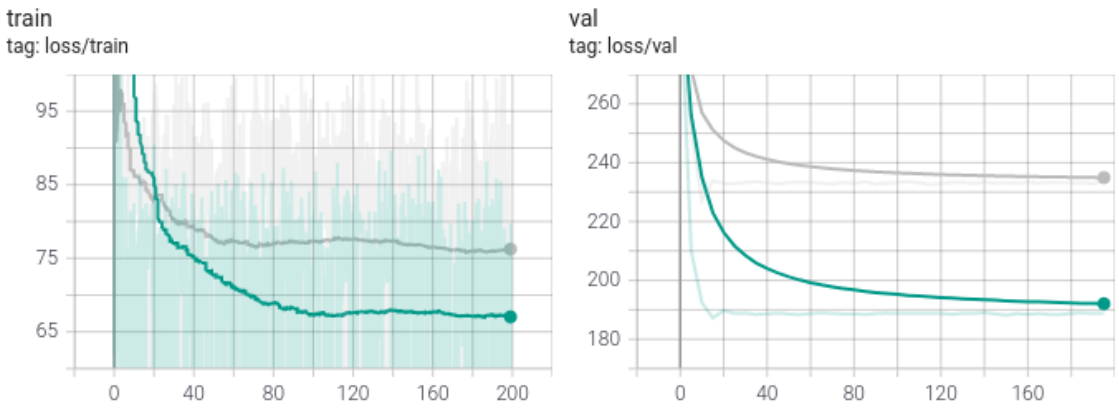


Figure 4.1: Training and validation loss of various model sizes. Extra layer pictured in green, withheld layer in gray.

Table 4.1: 30-Epoch MNIST Training Times for Contemporary Regularization

Method	Mean Training Time (s)	Std. Dev. (s)
Baseline	56.4	0.490
Mixup	58.6	0.490
Manifold Mixup	58.0	0.00[1]
VIB	61.6	0.490
InfoMixup	84.6	1.2

4.2 Compute Performance Comparison

InfoMixup has significant mathematical complexity compared to its contemporaries, which raises the concern about the scalability of *InfoMixup* to larger datasets and dimensions, especially when computing resources may be limited.

We test the complexity of *InfoMixup* via running several training runs on MNIST, using a dedicated workstation, with no other computational loads on the CPU or GPU. The recorded times are for an AMD Ryzen 7 3700X operating at 2.2GHz, and an NVIDIA RTX 2070 TI. Five duplicate runs were executed for each method.

1

[1]Manifold Mixup produced exceedingly consistent execution times, the variation for which is below the accuracy of the system clock. We report zero for the standard deviation, but some

Our testing shows that *InfoMixup* does indeed require more computation time, 30% more than VIB and 44% more than Manifold Mixup, on average. Over the baseline, VIB requires 9.2% more execution time, and Manifold Mixup requires 2.8% more time. *InfoMixup* requires 50% more compute time, which is a very significant increase.

A significant portion of this extra computation time is the non-parallel nature of many statistical computations, which make *InfoMixup* difficult to compute on GPU or other data-parallel acceleration schemes. The lower clock speed and thread grouping on highly data-parallel compute systems makes *InfoMixup* even more expensive on these systems, which is a strong disadvantage of our method. However, given the excellent performance of *InfoMixup* versus white-box adversaries, the cost may be worthwhile in critical cases. We also propose that there may be hope for *InfoMixup* and related methods in the near future, as custom computer architectures for both training and inference which speed up computations for neural networks are becoming commonplace. Although GPU acceleration fails to provide speedup for operations with decaying parallelism, future architectures which can perform statistical computations quickly may soon become available.

We also consider the performance of *InfoMixup* with respect to the percentile radius parameter, p, which controls the radius used as the cutoff between mixed and unperturbed examples. Decreasing the percentile radius should cause more examples to be included in the *InfoMixup* process, which will increase the necessary computation time.

Robustness of the model does not correlate with constrained percentile radius. Due to the nature of using a percentile as the cutoff, a non-constant number of examples are mixed in each minibatch, and different initialization may cause the network to simply have less outliers in the first place. Since our use of the quantile

variation is expected.

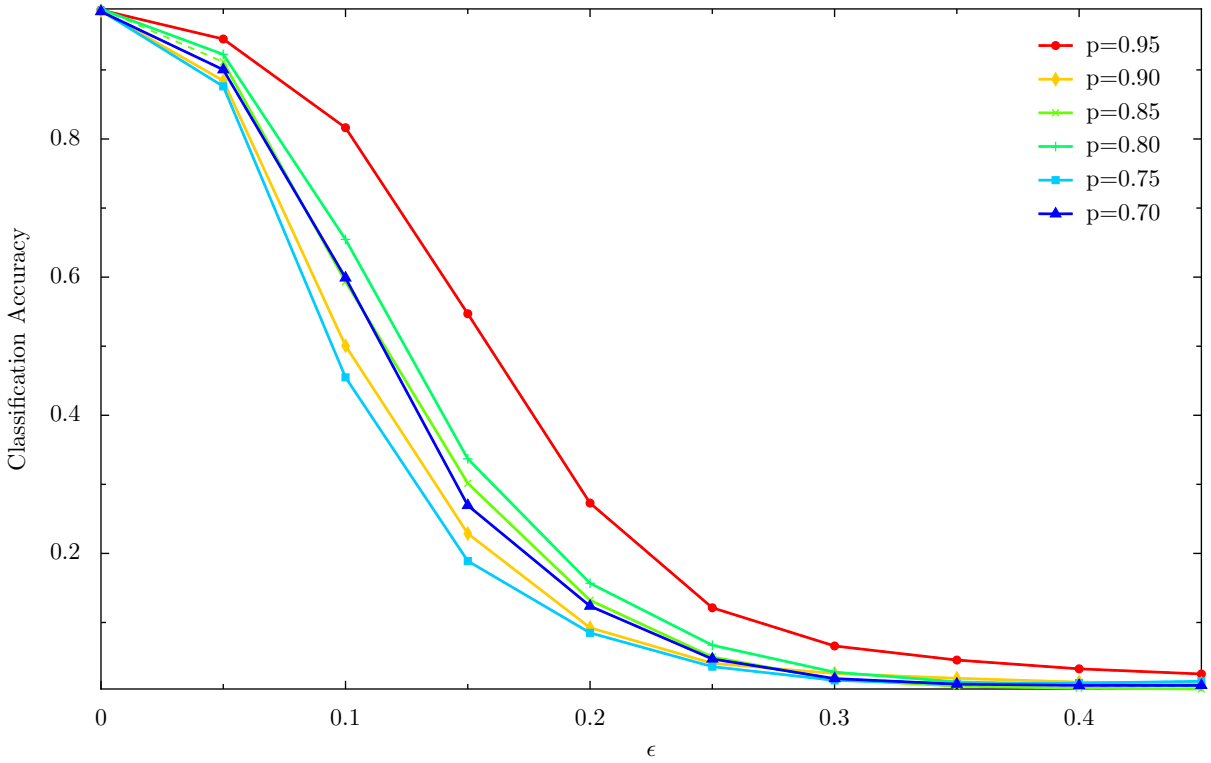


Figure 4.2: Robustness Evaluated at Various p

function of the χ^2 distribution is an upper-limit of the radius which contains $p\%$ of the examples, model initialization may change the precision of our estimate.

In a pathological case, the network learns to fit all the examples for a given label within the percentile radius, which effective disables *InfoMixup* , and from there the training will proceed as standard training with added distribution tracking, as no examples will be chosen for mixing.

We choose to empirically determine the compute cost. Models trained with optimal parameters, but with p swept from 0.70 to 0.95 were attacked with FGSM. One might expect that as the percentile boundary shrinks, such that more examples are included in mixing, the robustness of the model will increase. However, this is not what we observe. Instead, we see that 0.95 is the optimal value by a significant margin, and that the robustness oscillates, but trends downwards as the percentile drops. It is possible that the network is indeed learning to simply constrain its predictions

Table 4.2: Compute Time for Various p

p-value	25-epoch Compute Time (s)
0.95	134.727921
0.90	134.109429
0.85	133.655421
0.80	135.528113
0.75	88.591593
0.70	89.101397

for clean data to within the percentile radius, which will minimize the VIB objective in Equation (1.2) to a local minima which does not encourage robustness as intended. Since *InfoMixup* is effectively disabled for $p = 0.75$ and $p = 0.70$ in our experiments here, the compute time required drops significantly.

All times measured in this experiment were taken on a workstation with a Ryzen 7 3700X, 64 GB of 3600MHz DRAM, and an NVIDIA RTX 2070 TI, which was entirely dedicated to the task being measured for the duration of the experiment.

4.3 Analysis of Adversarial Training Performance

Our replication of the work of Madry et. al. [5] notably lacks the original performance claimed in their work.

LeNet5 as configured for our experiments, has 80,582 parameters. Madry et. al. use a model which has 3,233,034 parameters. Madry et. al. also claim that robustness increases with model capacity. Given the extreme difference in the number of parameters, a significant drop in overall performance is expected.

Lowering the correlation of brittle features with the label distribution by adding examples to the training set where these features do not correlate with the label distribution is in effect removing useful information that can be used by the model in its predictions. In addition, Madry et. al. train their network for 100 epochs,

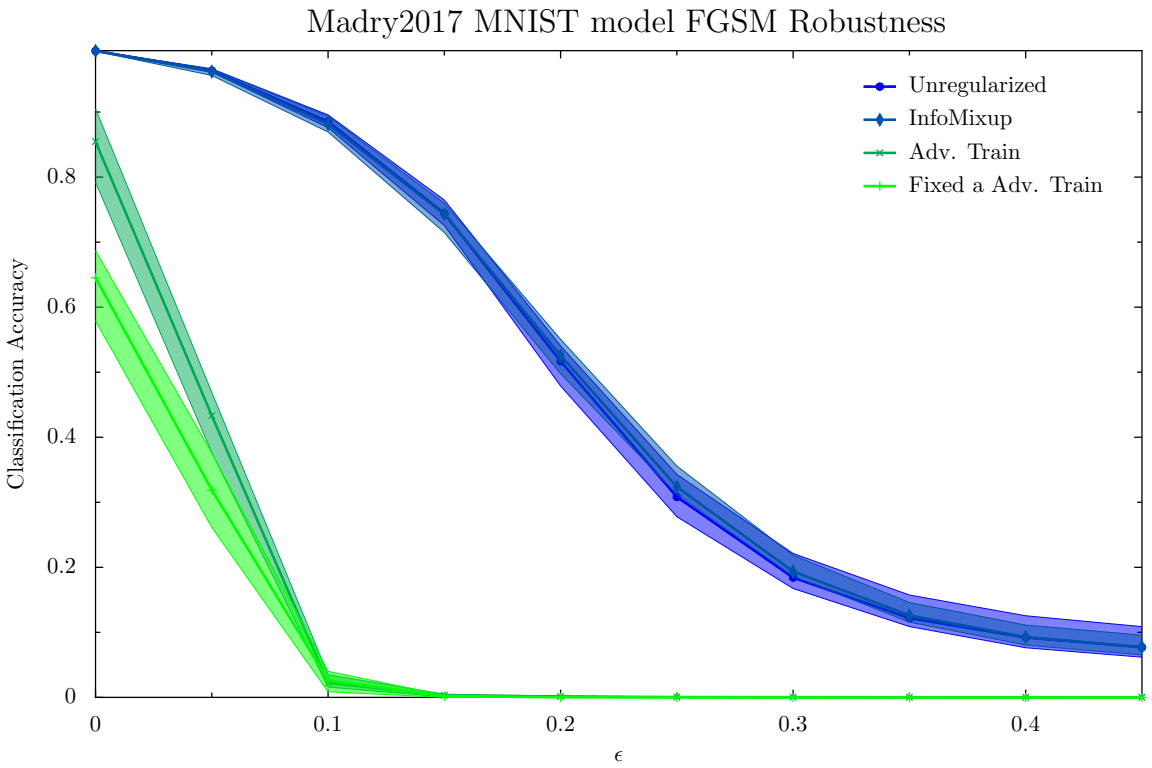


Figure 4.3: Poor performance of Adversarial Training using published parameters

whereas mine is trained for only 30, thus giving the optimizer less time to find the robust features and incorporate them into the model.

We clone the procedure of Madry et. al. exactly in an attempt to simply recreate their experimental results. We restricted training to include only adversarial examples, and use an independently initialized and trained copy of their MNIST model for generating the examples. Even so, we are unable to reproduce their results in our environment, which is using PyTorch 1.10.1. We use Foolbox [30][31] to perform all attacks. Our best-case results use a 50% blend of adversarial and clean examples in a LeNet5, which are the results presented in Figure 3.8.

There are several known problems with Adversarial Training, some of which are displayed here by our poor-performance models.

- Adversarial Training easily overfits to a metric [5]. During training, the model sees variations of each input image which have been perturbed a certain max-

imum radius under a fixed metric. A sufficiently powerful model may learn to represent the perturbations to each input sample in addition to the clean samples without common signs of overfitting. This makes the model secure against a specific kind of adversary and of a certain strength, but it remains fragile to other attack methods, potentially at lower ϵ.

- Adversarial Training can overfit to a metric [5]. A network trained with only l_2-bounded examples of a given ϵ is likely to be weak to examples generated under an l_∞ norm, even for the same ϵ. This result can be expected, as the meaning of ϵ is different under different metrics. Using several metrics and attacks during training greatly increases the computational cost of training.

- Distribution shifts can occur from large ϵ. When allowed a large perturbation, examples may become significantly different from the distribution of the training data. This is especially true for l_2-bound attacks. In this case, the overall performance of the network may decrease on clean examples.

This result is in line with the theory on which InfoMixup was born: adversarial features are highly predictive, but brittle [8]. After the initial publication, Madry et. al. have edited their paper to indicate that their results are skewed due to overfitting to the metric used to generate examples during training. In our experiments, we train with PGD as proposed in the original paper, but always attack with FGSM. Although both FGSM and PGD are using l_∞ norm in this experiment, the different nature of the attack is sufficient to decrease robustness of the secured model.

Chapter 5

Concluding Remarks

5.1 Future Directions

In this work we did not investigate several other possible uses of *InfoMixup* . Out-of-distribution data detection is a problem which is rising in popularity in recent years. Datasets for this use have been developed, namely ImageNet-O [32], which consists of images not contained within the ImageNet-1k challenge which are confidently predicted as classes within the Imagenet-1k class labels by state-of-the-art models. While not directly related to adversarial examples generated by optimization algorithms, certainly the phenomenon of deep networks placing high confidence predictions on out-of-distribution data is similarly a reason for concern.

The representation space of *InfoMixup* -regularized models has a point density with clusters for each class. Since the softmax classifier placed after *InfoMixup* considers the distance of a particular point from one of the class distributions as a standard softmax classifier would, *InfoMixup* does not inherently provide robustness to out-of-distribution data. However, since the density of points within the distribution of training data is well known, *InfoMixup* may be able to provide a signal to a side-channel mechanism for detecting out-of-distribution data points, either by detecting that the probability density at the point in the representation space corresponding to a particular example is lower than some learned threshold, or by detecting that the point resides in the margin between known class clusters. This information could be

used to raise suspicion on high-confidence predictions.

The same side-channel signal of out-of-distribution data could be used to detect Gaussian shifts in input data and potentially correct it, and then re-classify. This approach may allow learning a smaller model which can detect and correct Gaussian shifts in data, and determine whether correction is necessary before classification. Large-scale deployments of complex models will benefit from this detection, since re-training or incremental adjustments to the model may be expensive or simply not effective without enough data to meaningfully broaden the training distribution to include the real-world examples which are edge cases from the point of view of the model [33] [34]. Continual and lifelong learning applications may also benefit from this kind of signal. Robustness to distribution shifts is a significant problem in continual learning which is under active research today. Other potential benefits include automatic detection of new classes via clustering methods, which may be able to detect sub-classes in the output of *InfoMixup* -regularized models, or indicate to an engineer that an unknown class has appeared in deployment, or that there are deficiencies in the representation of a class in the existing model.

Robustness to adversarial examples, whether generated or natural, may also be improved by the use of non-softmax classifiers after *InfoMixup* regularization. Since the representation space of *InfoMixup* -regularized models has a known probability density and each class is Gaussian under some linear distortion in variance and mean, other classification methods, such as cluster-displacement classifiers or statistical Bayesian classifiers may be used in place of the standard softmax classifier, which boils down to a linear discriminant. Since *InfoMixup* produces a probability density in the representation space with distinct clusters for each class, it may also be possible to slightly perturb data to be classified, and compare the resulting sample distribution with the Gaussian mixture (albeit not a formal one) produced by *InfoMixup* for each class. If the variance of the sample distribution is very high

in a particular direction after being passed through *InfoMixup* , it is possible that a weakness in the model has been found, or that low confidence should be placed on the model's prediction for the original input. While this strategy is certainly applicable to non-*InfoMixup* models, our method enables more concrete statistical analysis of the results than contemporary methods due to the nature of the representation space.

The lack of resistance to transfer examples is reason for concern. A few hypotheses for why this is the case are known to the authors:

- *InfoMixup* dampens, but does not shatter gradients. It is possible that *InfoMixup* is modifying gradient information in a manner which confuses white-box attacks, but that does not cause them to be so confused as to perform worse than a black-box attack.

- *InfoMixup* may not protect against adversarial images which are out-of-distribution in a meaningful capacity. We believe that with a proper out-of-distribution detection mechanism, we may be able to flag adversarial inputs as "not classifiable" instead of misclassifying.

These items are excellent directions for future investigation, but may require significantly more analysis than can be done in this work. *InfoMixup* shows promise that more advanced information-guided regularization, and moreover classification techniques, are feasible in the realm of deep learning. The authors hope that this work will provide a foundation for further research in this area, and in an effort to ease developments have made the *InfoMixup* codebase open-source for inclusion in other projects. Our testing and training framework has also been open-sourced and made available for both verification of our results by our fellow researchers, and to ease the development of future methods.

Bibliography

[1] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," Dec. 2014.

[2] D. Hendrycks, K. Zhao, S. Basart, J. Steinhardt, and D. Song, "Natural adversarial examples," *CVPR*, 2021. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2021/papers/Hendrycks_Natural_Adversarial_Examples_CVPR_2021_paper.pdf

[3] A. Athalye, N. Carlini, and D. Wagner, "Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples," in *Proceedings of the 35th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, J. Dy and A. Krause, Eds., vol. 80. PMLR, 10–15 Jul 2018, pp. 274–283. [Online]. Available: https://proceedings.mlr.press/v80/athalye18a.html

[4] D. Hendrycks and T. Dietterich, "Benchmarking neural network robustness to common corruptions and perturbations," *Proceedings of the International Conference on Learning Representations*, 2019.

[5] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," Jun. 2017.

[6] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," Aug. 2016.

[7] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," Dec. 2013.

[8] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019, pp. 125–136. [Online]. Available: https://proceedings.neurips.cc/paper/2019/file/e2c420d928d4bf8ce0ff2ec19b371514-Paper.pdf

[9] C. E. Shannon, "A mathematical theory of communication," *The Bell System Technical Journal*, vol. 27, no. 3, pp. 379–423, 1948.

[10] N. Tishby, F. C. Pereira, and W. Bialek, "The information bottleneck method," Apr. 2000.

[11] V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, D. Lopez-Paz, and Y. Bengio, "Manifold mixup: Better representations by interpolating hidden states," in *Proceedings of the 36th International Conference on Machine*

Learning, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 6438–6447. [Online]. Available: http://proceedings.mlr.press/v97/verma19a.html

[12] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "mixup: Beyond empirical risk minimization," Oct. 2017.

[13] A. Kabra, A. Chopra, N. Puri, P. Badjatiya, S. Verma, P. Gupta, and B. Krishnamurthy, "Mixboost: Synthetic oversampling using boosted mixup for handling extreme imbalance," in *2020 IEEE International Conference on Data Mining (ICDM)*, 2020, pp. 1082–1087.

[14] D. Hendrycks, N. Mu, E. D. Cubuk, B. Zoph, J. Gilmer, and B. Lakshminarayanan, "Augmix: A simple data processing method to improve robustness and uncertainty," Dec. 2019.

[15] L. Schott, J. Rauber, M. Bethge, and W. Brendel, "Towards the first adversarially robust neural network model on mnist," May 2018.

[16] H. Guo, Y. Mao, and R. Zhang, "Mixup as locally linear out-of-manifold regularization," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, no. 01, pp. 3714–3722, Jul. 2019. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/4256

[17] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[18] J.-H. Kim, W. Choo, and H. O. Song, "Puzzle mix: Exploiting saliency and local statistics for optimal mixup," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 5275–5285. [Online]. Available: http://proceedings.mlr.press/v119/kim20b.html

[19] J. Kim, W. Choo, H. Jeong, and H. O. Song, "Co-mixup: Saliency guided joint mixup with supermodular diversity," in *International Conference on Learning Representations*, 2021.

[20] S. Lee, H. Lee, and S. Yoon, "Adversarial vertex mixup: Toward better adversarially robust generalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[21] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," Feb. 2015.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Dec. 2015.

[23] A. A. Alemi, I. Fischer, J. V. Dillon, and K. Murphy, "Deep variational information bottleneck," *Proceedings of the International Conference on Learning Representations (ICLR) 2017*, Dec. 2016.

[24] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," Dec. 2013.

[25] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," *CoRR*, vol. abs/1611.02770, 2016. [Online]. Available: http://arxiv.org/abs/1611.02770

[26] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[27] E. Hoffer, I. Hubara, and D. Soudry, "Train longer, generalize better: closing the generalization gap in large batch training of neural networks," in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, Eds., vol. 30. Curran Associates, Inc., 2017, pp. 1731–1741. [Online]. Available: https://proceedings.neurips.cc/paper/2017/file/a5e0ff62be0b08456fc7f1e88812af3d-Paper.pdf

[28] L. Schmidt, S. Santurkar, D. Tsipras, K. Talwar, and A. Madry, "Adversarially robust generalization requires more data," in *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., vol. 31. Curran Associates, Inc., 2018, pp. 5014–5026. [Online]. Available: https://proceedings.neurips.cc/paper/2018/file/f708f064faaf32a43e4d3c784e6af9ea-Paper.pdf

[29] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," 2017. [Online]. Available: https://arxiv.org/abs/1708.07120

[30] J. Rauber, W. Brendel, and M. Bethge, "Foolbox: A python toolbox to benchmark the robustness of machine learning models," in *Reliable Machine Learning in the Wild Workshop, 34th International Conference on Machine Learning*, 2017. [Online]. Available: http://arxiv.org/abs/1707.04131

[31] J. Rauber, R. Zimmermann, M. Bethge, and W. Brendel, "Foolbox native: Fast adversarial attacks to benchmark the robustness of machine learning models in pytorch, tensorflow, and jax," *Journal of Open Source Software*, vol. 5, no. 53, p. 2607, 2020. [Online]. Available: https://doi.org/10.21105/joss.02607

[32] A. Srivastava, S. Jain, and M. Thigle, "Out of distribution detection on imagenet-o," 2022. [Online]. Available: https://arxiv.org/abs/2201.09352

[33] S. Sodhani, S. Chandar, and Y. Bengio, "Toward Training Recurrent Neural Networks for Lifelong Learning," *Neural Computation*, vol. 32, no. 1, pp. 1–35, 01 2020. [Online]. Available: https://doi.org/10.1162/neco_a_01246

[34] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0893608019300231