Certifying Properties of Deep Networks by Taking them into Shallow Waters

Certifying Properties of Deep Networks by Taking them into Shallow Waters

Master Thesis

Konstantinos Barmpas
Master of Engineering (M.Eng.)

Electrical Engineering & Information Technology, ETH Zürich
Electrical & Electronic Engineering, Imperial College London

Supervisor: Prof. Dr. Thomas Hofmann

Advisors: Kevin Roth (ETH), Yannic Kilcher (ETH),
David Haber (Daedalean AI)

Data Analytics Lab
Department of Computer Science, ETH Zürich

Daedalean AI, AG

This thesis is dedicated to my niece

Abstract

Over the last decades, complex deep neural networks have revolutionized Artificial Intelligence (AI) research. These models can now achieve impressive performances on various complex tasks like recognition, detection and image semantic segmentation, achieving accuracy close to, or even better, than human perception. However, these neural networks require to be both deep and complex and this complexity constitutes a danger for the safety verification (certification) and interpretability of a neural network model.

This project explores the certification properties of complex neural networks by taking them into ”shallow waters”. First, a detailed investigation of efficient model distillation techniques is conducted. Then, using the shallow models trained with these distillation methods, several of their properties are further explored, among them adversarial robustness and their performance under parameter reduction procedures. Finally, by combining network’s convex relaxation with model compression, the certification area of shallow student models (derived from either normally or robustly trained teacher networks) is researched. Through all of these experimental results, it is empirically demonstrated and proved that model distillation leads to shallow models with larger certification areas than their equivalent complex teacher networks. Therefore, based on this thesis evidence, shallow distillated networks constitute a possible solution to the safety and interpretability issues that modern complex Artificial Intelligence (AI) models face.

Acronyms

In this report the following abbreviations are widely used especially in the experimental section of the project (Chapter 8):

- **AI**: Artificial Intelligence.
- **DNN**: Deep Neural Network.
- **GPU**: Graphics Processing Unit.
- **CV**: Convolutional Layer.
- **MP**: Max-Pooling.
- **FC**: Fully-Connected.
- **LR**: Learning Rate.
- **CNN**: Convolutional Neural Network.
- **ReLU**: Rectified Linear Unit.
- **SGD**: Stochastic Gradient Descent.

Acknowledgements

First and foremost, I would like to thank my supervisor Prof. Thomas Hofmann for providing me with the opportunity to conduct this incredible project. It has been a true honour to be his student. I would also like to thank my advisors (David, Kevin and Yannic) for their invaluable guidance, advice and continuous support throughout the course of this project, especially during these uncertain times.

Since this project marks the end of my university degree, I would like to thank Prof. Alessandro Astolfi for his nomination to attend my year abroad at ETH Zürich as well as Dr. Christos Papavassileiou for opening the door to Imperial College London for me four years ago.

A big thanks to all the best friends that I have made throughout my time at university. I wish them all the best in their future endeavours. A special thanks goes to Omar who has always been my support and source of encouragement over the last four years.

Last but not least, I would like to thank my family (especially my mom, dad, aunt and sister) and my ”second” famliy (Ntinos, Nikos, Paulos, Katerina, Anna-Maria, Sotiria and Rania) for their unconditional love and support. Without them, I would not have had the opportunity and strength to attain an exceptional education at both ETH Zürich and Imperial College London.

Contents

Chapter 1

Introduction

In recent years, the field of Machine Learning has demonstrated impressive performances on various tasks, especially visual tasks, that have applications in unpredictable environments such as self-driving cars and autonomous flying vehicles. State-of-the-art neural networks can now efficiently solve complex problems like object recognition, object detection and image semantic segmentation with very high confidence. Efficient Machine Learning algorithms have been the object of study for several decades but due to a significant increase in the number of available data as well as recent advancements in hardware, there is now the necessary computational power to make these algorithms feasible. But this boost in performance comes at the cost of deeper and more complex models and this complexity poses serious threats to safety and interpretability. As complex machine learning models have slowly started controlling elements of our daily lives and are in charge of cars or airplanes, the issue of trust becomes more and more relevant. In the past few years, progress in the field of Artificial Intelligence (AI) has involved increasingly data-driven, black box approaches. Now more than ever, there is the need to define a regulatory framework and standards for the safe use of machine learning-based systems.

In light of this, the **European Union** has launched their first version of the Ethics Guidelines for Trustworthy Artificial Intelligence [1] in 2019. These guidelines have started to influence rule making across industries. For the aviation industry, which poses many interesting examples of safety critical applications, the European Union Aviation Safety Agency (**EASA**) has derived their own roadmap to create a certification framework that enables the introduction of AI applications without compromising safety.

Daedalean AI [2], a startup based in Zurich, builds relevant safety-critical applications using machine learning which will have to be compliant with future certification standards. Their operation covers industries from general aviation to urban air mobility. Using machine learning and computer vision, Daedalean's visual system enables autonomous flying capabilities to completely and reliably replace the human pilot. Their autopilot software is developed according to strict software regulatory rules and standards under the "Software Considerations in Airborne Systems and Equipment Certification" (DO-178C).

In 2019, Daedalean AI started a collaboration with EASA (European Union Aviation Safety Agency) in an Innovation Partnership Contract (IPC) on the Concepts of Design Assurance for Neural Networks (CoDANN). The purpose of this project was "to investigate ways to gain confidence in the use of products embedding machine learning-based systems (and more specifically neural networks) and develop general guidelines towards the certification of deep neural networks in order to expand the current aviation regulatory framework". The first

[1] https://ec.europa.eu/digital-single-market/en/news/ethics-guidelines-trustworthy-ai

[2] Daedalean AI - official website: https://www.daedalean.ai

round of this IPC was concluded with the publication of a joint report on the "Learning Assurance for Neural Network" [3]. In 2020, the second round of the IPC was launched to further investigate and extend these guidelines for autonomous autopilot systems for aircrafts of near future. The motivation of this Master Thesis Project is to contribute to the above work and the second round of the IPC through an investigation and comparison of the certification properties of deep neural networks and shallow distillated models.

The remainder of this report is structured as follows:

- *Chapter 2* presents background information related to the project, including related work and a brief introduction to CNNs followed by fundamental principles of model compression, adversarial attacks as well as certification area.
- *Chapter 3* captures and specifically analyses the project objectives.
- In *Chapter 4*, a number of model distillation techniques is presented and discussed in detail. A complete discussion about the reasons behind distillation's success is also presented in this chapter.
- *Chapter 5* is oriented around parameter reduction. More specifically, it lays down the theoretical background behind "Pruning Filters" [30] method and its benefits in terms of parameter reduction as well as computational cost.
- In *Chapter 6*, a wide variety of adversarial attacks are presented and described in full detail.
- *Chapter 7* is based on the work conducted by Eric Wong et al. in [47] and describes a method based on network's relaxation to design and build robust ReLU-based classifiers.
- *Chapter 8* is concerned with the experimental implementation to investigate the project objectives as described in Chapter 3 using the theoretical knowledge and methods described in Chapters 4, 5, 6 and 7.
- *Chapter 9* includes a critical appraisal of the work presented in this Master Thesis project.

[3]https://www.easa.europa.eu/newsroom-and-events/news/easa-artificial-intelligence-roadmap-10-published

Chapter 2

Background

This part of the report serves as the literature review of the project and the place where we lay down some of the fundamental principles as well as the notation used in this thesis. In this chapter, the concept of model distillation is presented, followed by a discussion on adversarial attacks and their danger for a neural network. Finally, mathematical details of certification are also presented here.

2.1 Related Work

As mentioned in the previous chapter, the idea of model distillation is not a new concept to the machine learning community. In the late 80s, Cybenko [18] mathematically demonstrated that a neural network with only one hidden layer consisting of units with sigmoid activations can efficiently approximate any target function with the only requirement to be a sufficiently wide hidden layer. Empirical evidence though disproves Cybenko since shallow neural networks are often less accurate than deep neural network models. Zeng and Martinez [51] used neural networks and synthetic data produced by the RANDOM algorithm to approximate ensembles of classifiers. Following their work, Bucilå, Caruana and Nicolescu-Mizil [17] presented a method for compressing large ensembles into faster and smaller models of equivalent performance. In their experiments, the shallow mimic networks were trained on synthetic examples generated using the MUNGE algorithm.

Dauphin and Bengio ([19] & [23]) made use of SIFT features in their attempt to train efficient large, high-accuracy shallow models on the large-scale ImageNet dataset. A number of research papers (e.g. [20],[34],[46],[45],[22],[26],[36]) suggest that deep neural networks are more efficient and accurate compared to shallow models for tasks like image analysis and speech acoustic modeling. Furthermore, Cohen and Shashua [16] as well as Liang and Srikant [37] proved empirically that the representational efficiency of deep neural networks grows exponentially with depth favoring deep complex neural networks over shallow models. More recently, Ba and Caruana (in [7] & [27]) demonstrated that shallow neural networks can approximate / learn the target functions that more complex deep neural networks have learnt during training. In their work, the distillation of knowledge occurs by minimising the squared difference between the logits (the input to the output softmax layer of the classifier model) produced by the deep models and the logits produced by the shallow models. Hinton, Vinyals and Dean ([25] & [32]), building on Caruana's previous work, demonstrated a new way of knowledge distillation by ”raising the temperature of the output softmax layer of the deep neural network model” to produce soft targets and finally proved that ”matching the logits” is a specific form of this more generic distillation method.

During the last years, the topic of adversarial attacks has been an active field of research in the machine learning community. Recent work by Goodfellow et al. [31] has demonstrated that adversarial examples (data points that are indistinguishable to the human eye from correct examples) can be produced and successfully fool the machine learning models. Different strategies have been followed over the years to design models that withstand adversarial perturbations ([28] & [15]). In recent years, model distillation has been considered multiple times as an effective method to design models robust to adversarial attacks. Currently, there is no conclusive evidence of distillation's success since it has been both empirically proved ([43] & [42]) and disproved [13] as an efficient design method for adversarial robustness.

As it is obvious, there is the need for robust neural networks, models that can demonstrate conclusive robustness against norm-bounded perturbations. In other words, networks where a certificate of robustness exists. Different robust optimization methods have been employed over the years ([31], [38], [48], [44], [11]) that focus on linear neural network models. Singh et. al [24] presented DeepZ, a method to certify neural network robustness based on abstract interpretation, while Eric Wong and Zico Kolter [47] proposed a method to train robust ReLU classifiers via the "convex outer adversarial polytope" technique.

In summary, the aforementioned works have proposed different model distillation techniques and properties. However, they only consider model distillation as a method to only train accurate shallow neural networks. Effective defenses against adversarial attacks have been focused only on adversarial training or robust optimization principles. Our work in this thesis is fundamentally associated with the field of distillation ([7], [25] & [27]) but it also relates to the connection between distillation and adversarial robustness. Finally, building on the work by Eric Wong and Zico Kolter [47] and using the empirical results from different distillation techniques, we will examine the effect that distillation has on the certification area of both normally and robustly trained models.

2.2 Notation

As in any other machine learning model, the objective of a deep neural network is to approximate some unknown target function f^*. The network defines a mapping $y = f(x;\theta)$ and learns the set of parameters θ that yields the best approximation function. In this thesis, the predictive neural network model is denoted as f_θ with θ being the set of the network's parameters. The predictive models used in our analysis are Deep Neural Networks (DNNs) with K-hidden layers (artificial neural networks (ANNs) with multiple layers between their inputs and outputs). More precisely, Convolutional Neural Networks (CNNs) consist the main neural network tool throughout the present thesis report. The set of the learned parameters θ consists of the weights as well as the biases of each layer. Mathematically, $\theta := (W_i, b_i)$ with W_i and b_i denoting the weights and bias of the i-th layer of network respectively. Each unit in the network has a non-linear activation function in their output denoted by h. Furthermore, the labelled training and test datasets are denoted by $(\mathcal{X}, \mathcal{Y})$, and $(\mathcal{X}_{test}, \mathcal{Y}_{test})$ respectively.

For evaluation purposes of the models, clean accuracy is the metric frequently used. It is the fraction of predictions that the predictive model classifies correctly over the total number of data points being tested.

Definition 2.1 Let f_θ be a predictive model and the test set $(\mathcal{X}_{test}, \mathcal{Y}_{test})$=$(x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$, the clean accuracy is defined as :

$$CleanAccuracy := \frac{\sum_{(x,y)\in(\mathcal{X}_{test},\mathcal{Y}_{test})} \mathcal{I}(f_\theta = y)}{N} \tag{2.1}$$

where $\mathcal{I}$ is the indicator function:

$$\mathcal{I}(y = x) = \begin{cases} 1 & x = y \\ 0 & x \neq y \end{cases}$$

Finally, in different parts of our analysis perturbations, and more precisely norm-bounded perturbations, are used.

Definition 2.2 A perturbation ball in input space is denoted by $\mathcal{B}_\varepsilon$ and for norm-bounded perturbations around point x is defined as:

$$\mathcal{B}_\varepsilon(x) := \{x' \mid \|x - x'\| < \varepsilon\} \tag{2.2}$$

where $\varepsilon > 0$.

2.3 Convolution Neural Network (CNN)

This section presents a detailed discussion on Convolution Neural Networks (CNNs). Starting with an overview of their architecture, followed by a detailed analysis of the fundamental convolutional layer and the max-pooling layer which are both used in our experiments later on. Finally, an analysis of the use of the CNN architecture for classification tasks is presented since image classification is the main focus of the experimentation part of this report.

2.3.1 Overview

Convolutional Neural Network (CNN) is a machine learning architecture with significant applications in the field of computer vision and "inspired by the natural visual perception mechanism of the living creatures" [33]. CNN is a variant of deep neural network that involves convolutional and fully-connected layers which share their weights and diminish the amount of total parameters. It uses feature maps to extract significant features and retain important information. Sub-sampling and pooling layers are also widely used with the main purpose to be dimensionality reduction.

2.3.2 Convolutional Layer

The convolutional layers aim to learn feature representations of the inputs. Each convolutional layer consists of multiple kernels used to convolve the input and produce the feature maps. In a feature map, each neuron is linked to a neighbourhood of neurons in the previous layer of the CNN. This neighbourhood is also known as the neuron's receptive field.

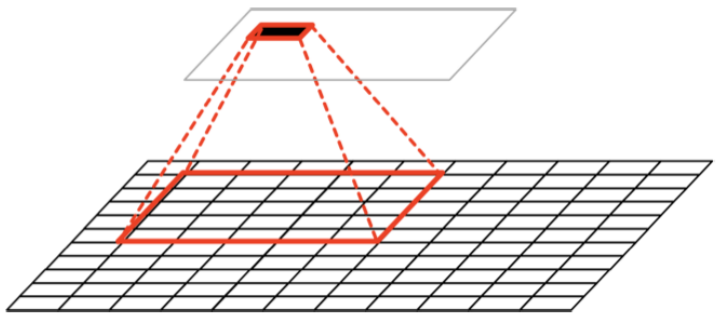


Figure 2.1: Illustration of Convolutional Layer & Receptive Field (Source: [33])

Each feature map is computed in two steps: first a convolution between the input with a known kernel takes place. Then, the result is passed through a non-linear activation function.

Definition 2.3 Using the notation described above, the (i,j) element of m-feature map in the k-layer is derived mathematically as:

$$y^k_{i,j,m} = h_k((\mathbf{w}^k_m)^T * \mathbf{x}^k_{i,j} + b^k_m) \tag{2.3}$$

where $\mathbf{x}^k_{i,j}$ is the input centered around (i,j) in the k-th layer, $\mathbf{w}^k_m$ and b^k_m are the weights and bias of the k-th layer respectively and h_k is the activation in the k-th layer of the network.

2.3.3 Max-Pooling Layer

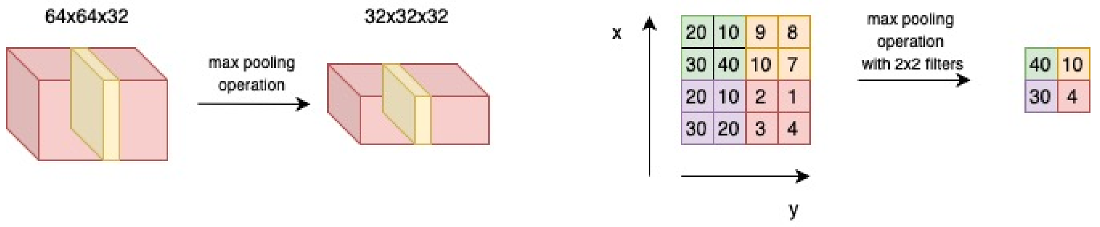


Figure 2.2: (Left): The input has size of [64x64x32] and is max-pooled with filter size 2x2 and stride 2 into output of size [32x32x32]. Notice that the depth / number of channels is preserved. (Right): Mathematical example in a single channel slice. The max pooling operation takes place with filter 2x2 and stride 2. In each pooling region, the maximum number is passed on the next layer.

The pooling operation constitutes a vital component of the CNN architecture. Pooling significantly decreases the amount of connections between convolutional layers. It keeps the computational cost at a low level while preserving all the important information. There are several pooling methods but throughout this report only max-pooling layers are frequently used.

Max-pooling layers, similar to convolutional layers, apply a kernel to the input feature map and generate an output feature map which is composed by the maximum value elements in each region defined by the kernel size.

Definition 2.4 In the max-pooling operation, the (i,j) element of m-feature map is derived mathematically as:

$$a_{i,j,m} = \max_{k,l \in \mathcal{J}_{ij}} y_{k,l,m} \tag{2.4}$$

where $a_{i,j,m}$ is the pooling output at (i,j), $y_{k,l,m}$ is the input value at (k,l) inside the pooling region $\mathcal{J}_{ij}$ in the (i,j) neighbourhood defined by the pooling kernel in the m-feature map.

2.3.4 Classification Task

Convolutional Neural Networks (CNNs) represent a huge breakthrough in image analysis, especially in the fields of classification and recognition. "Their architecture resembles the visual perception mechanism of the living creatures whose cells in the visual cortex are responsible for detecting light in receptive fields" as Hubel and Wiesel described in their study [21].

A typical CNN architecture for classification (an illustration example of which appears in the graph below) is composed of three main parts:

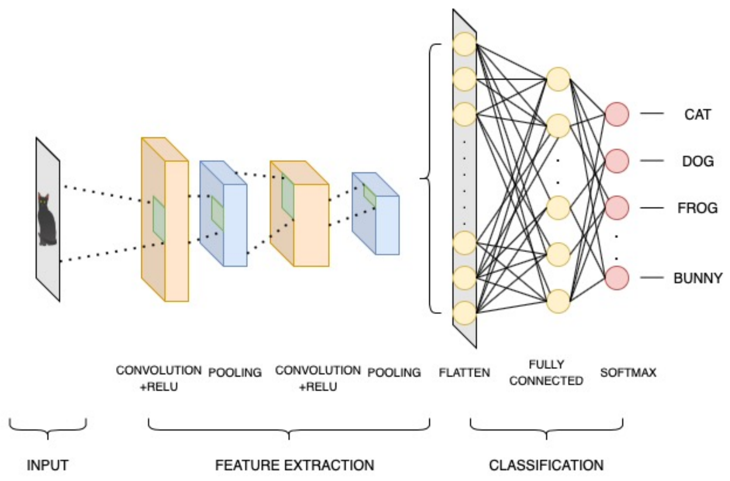


Figure 2.3: Illustration of the different parts of a CNN architecture tasked with image classification

1. The input part: The image that needs to be classified by the CNN architecture
2. The feature extraction part: This part of the CNN involves convolution and pooling operations to create multiple feature maps. In each layer, different filters are applied along with a non-linear activation function to generate the different feature maps. For image classification, ReLU activation function, a non-linear and non-saturated function, is frequently chosen for this task.

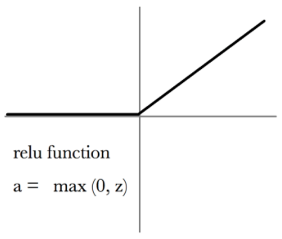


Figure 2.4: ReLU non-linear activation function graph

Definition 2.5 ReLU is a non-linear function that sets the negative values to zero and maintains the positive values. Since it uses only the maximum operation is faster compared to other non-linear functions. ReLU activation function is mathematically described as:

$$a_{i,j,m} = \max(0, z_{i,j,m}) \tag{2.5}$$

where $a_{i,j,m}$ and $z_{i,j,m}$ are the output and input of the function respectively at location (i,j) in the m-th feature map / m-th channel.

The final step of the feature extraction part includes the flattening of the last feature maps into one long vector which consists the characteristic features of the input image.

3. The classification part: It is the final stage in the classification pipeline. It is composed of a fully-connected artificial neural network (ANN) whose input is the characteristic feature vector of the feature extraction part. The number of the neuron units in the final fully-connected layer (also known as output layer) equals to the number of possible classification classes. While the hidden units can use different non-linear activation functions (ReLU is a popular choice like in the feature extraction part), the output layer

uses softmax activation function to map the non-normalized output of a network to a probability distribution over possible classification classes.

Definition 2.6 Let N the possible classification classes. The softmax operation is denoted by σ and is defined mathematically as:

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \tag{2.6}$$

for i $\in$ (1,2,...N) and $\mathbf{z} = (z_1, z_2, ..., z_N)$. The standard exponential function is applied to each element z_i of the output layer and is normalized by the sum of all the exponentials. The softmax operation ensures that the sum of the outputs in the final layer of the CNN is equal to 1.

As in any other machine learning network, the learning of the model's parameters happens through training. For the model's training, a loss function is required. The loss function quantifies how close the predictions of the CNN model are to the true labels of the images. Let θ denote all the parameters of a CNN model with labelled training set $(\mathcal{X}, \mathcal{Y})$=$(x_1, y_1), ..., (x_N, y_N)$.

Definition 2.7 The total loss of a CNN model can be calculated as follows:

$$\mathcal{L} = \frac{1}{N} * \sum_{i=1}^{N} l(\theta, \mathbf{x}^n, \mathbf{y}^n) \tag{2.7}$$

The goal of any training algorithm is to minimize the value of the loss function by adjusting the weights and biases of the neurons. The best set of parameters are computed through the minimization of the loss function. Ideally, this should be zero and in that case the CNN model perfectly classifies all the samples. In practice though, this would lead to overfitting the training data, in other words the CNN model has memorised the data rather than learn the target function and tends to perform poorly on new unseen data samples. For that reason, a different set of images known as validation set is used during training with the only goal to efficiently estimate the performance of the model in unseen data (not encountered at the training stage) and stop the training process at the right time.

2.4 Model Compression

The conception of model compression (distillation) is to produce a shallow neural network model (a neural network with a small number of hidden layers) that can mimic (approximate) more complex models. The deep complex models are known as teacher models and the shallow models as student models.

Definition 2.8 Let f^* be an unknown target function, f_θ a K-layer predictive model and g_θ a L-layer predictive model where $L << K$. Distillation is defined as the process to achieve the following approximation:

$$f^* \approx f_\theta \approx g_\theta \tag{2.8}$$

For reasons discussed later on in the report, the shallow student networks can learn more precise functions (closer to the true target function) when trained via model compression rather than when trained on the original dataset used to train the complex models. Model distillation is one of the main fields of focus in this thesis report. Therefore, different strategies as well as a discussion about the reasons behind distillation's success will be analysed in the next chapters.

2.5 Adversarial Attacks

Adversarial attacks are small undetectable perturbations of the input data points that are sufficient to fool the neural networks and change their output prediction.

Definition 2.9 Let f_θ be a well-trained predictive model and $\mathcal{B}_\varepsilon$ a norm-bounded perturbation ball around the data point x $\in \mathcal{X}$. x′ is a adversarial example if and only if:

$$f_\theta(x') \neq p \tag{2.9}$$

where $f_\theta(x) = p$ and $x' \in \mathcal{B}_\varepsilon$.

There has been a great research interest in the design of adversarial attacks. According to Barreno (in ([8],[9] & [10]), and Ozdag [41], the adversarial attacks of machine learning models can be described in categories as they are illustrated in figure 2.5.

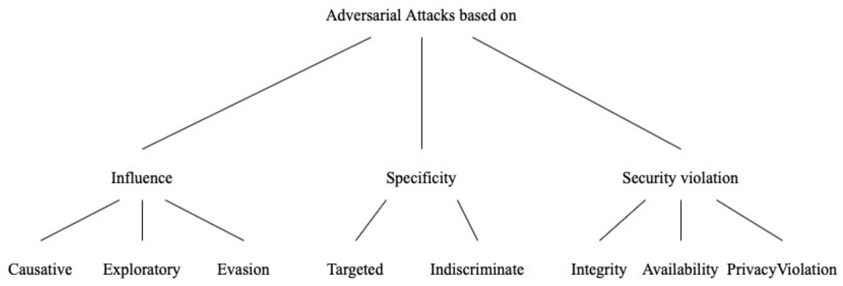


Figure 2.5: Adversarial attacks categorized based on Influence, Specificity and Security Violation

The influence of the adversarial attacks is an important taxonomy factor. Causative attacks target the training process while exploratory and evasion attacks take place after the training is completed with the former method exploiting misclassifications without altering the training process and the latter method modifying the input data. For the security violation criterion, attacks are characterized based on the state of the system after the attack (for instance false positives). Targeted attacks target specific data points while indiscriminate attacks exploit in a random non-targeted manner.

Furthermore, another categorization strategy has been described extensively in literature (Biggio and Roli [12], Chakraborty et al. [14], Yuan [50]). This categorization strategy (illustrated in figure 2.6) takes into account the different capabilities that adversarial attacks might have, their goals and their knowledge (complete or partial) of the machine learning system.

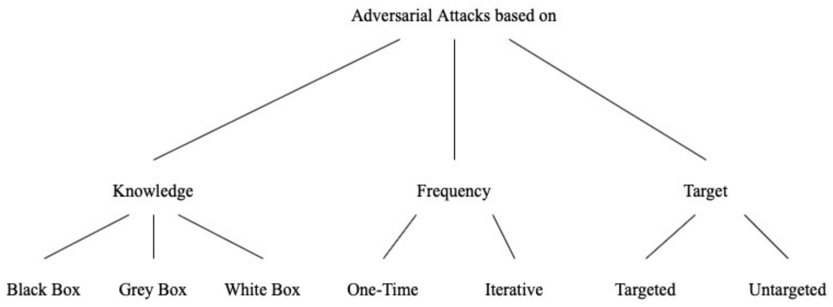


Figure 2.6: Adversarial attacks categorized based on Target, Frequency and Knowledge of the system

White box attacks require a complete knowledge of the machine learning systems, while Grey box and Black box attacks have some and no knowledge respectively. Based on their frequency adversarial attacks can be one-time or iterative (multiple times) attacks. Finally, a targeted attack targets a specific class while a untargeted attack does not.

For the evaluation of the models' robustness, different adversarial attacks are used throughout our analysis and they will be detailedly described in the next chapters.

2.6 Certification

As it is obvious, there is a growing interest in the challenging topic of accessing and verifying (certifying) robustness properties of neural networks. In other words, given a f_θ trained model on a training set $(\mathcal{X}, \mathcal{Y})$, certification is interested in measuring the model's robustness by examining for each point x whether its neighbourhood produces the same classification result. In other words, what is the maximum distance ε in the neighbourhood of the data point x such that the point can not be turned into an adversarial example.

Definition 2.10 A verification problem for any predictive model f_θ can be defined as the following robust optimisation problem [29]:

$$\mathcal{J}(x, f_\theta, \varepsilon) = \min_{x' \in \mathcal{B}_\varepsilon} c * f_\theta(x') \tag{2.10}$$

where c is a matrix encoding the specification or the relation between the outputs of the network that needs to verified (certified).

Using c := $e_{y^{correct}}$ - $e_{y^{wrong}}$, the sign of J indicates if there is any point in the $\mathcal{B}_\varepsilon$ ball that flips the model's prediction. More specifically, if J $>$ 0, there exists no input in $\mathcal{B}_\varepsilon$ ball while if J $<$ 0, then there exists at least one input in the ε-neighbourhood of x where model prediction is incorrect.

There are two ways to verify any predictive model f_θ given a dataset $\mathcal{X}$:

1. Verify that all points $x \in \mathcal{X}$ over the same fixed local perturbation ball $\mathcal{B}_\varepsilon$ remain unchanged to the perturbations.
2. For every point $x \in \mathcal{X}$, find a different local perturbation ball $\mathcal{B}_\varepsilon$ for which the network's output is guaranteed to not change.

Certification is the last part of our experimental results and these methods will be analyzed in detail in the next chapters.

Chapter 3

Requirements Capture

The primary objective of this thesis project is to investigate model distillation as well as its applications in areas like adversarial robustness and certification. Up until this point, there has been a short introduction of the project's motivation and an overview of the background knowledge that is required for the project. Potential advantages and disadvantages of model distillation have been briefly discussed in the literature review section. The project investigation will involve different model distillation techniques and will be oriented around models' robustness as well as their certification area. Although the aforementioned objectives are quite broad, for each task research can be oriented around the following issues:

Model Distillation

1. Implement different model distillation techniques that have been described in the literature as efficient model compression methods.
2. Investigate the factors that contribute to successful knowledge distillation.
3. How do the results of different distillation techniques compare with each other ?
4. Are there any disadvantages of distillating complex deep neural networks ?

Parameter Reduction

1. Investigate whether a decrease in the amount of parameters of complex deep neural networks is possible without, or slightly, affecting the overall clean accuracy of the model.
2. Having an efficient method in place, study whether the combination of model distillation and parameter reduction can result in shallow models with small number of parameters and accuracy comparable to complex deep neural networks.

Adversarial Robustness

1. Implement different adversarial attack techniques on both deep neural networks and shallow distillated neural networks.
2. Study the relationship of clean accuracy and adversarial robustness.
3. Investigate with empirical evidence whether model distillation results in robust models. In other words, compare the performance of deep teacher neural networks and shallow distillated student neural networks under the same settings of adversarial attacks.

Certification Area

1. Implement a normally trained and a robustly trained classifier with the same architecture that will be used for comparison.
2. Make use of the conclusions in the previous research areas (Model Distillation, Parameter Reduction and Adversarial Robustness) and create efficient shallow distillated models of the two classifiers.
3. Study the different factors that might affect the certification area of a deep neural network model.
4. Investigate whether model distillation has any effect on the certification area of the models compared to their equivalent complex teacher deep neural networks.
5. Are the empirical results independent of the method (robustly or non-robustly technique) that the teacher model has been trained with?

To answer the aforementioned questions and successfully execute the different research tasks, our analysis will be focused on image classification on publicly available research image datasets, more specifically in the SVHN dataset and the CIFAR-10 dataset. While the predictive models used throughout this report will be primarily Convolutional Neural Networks (CNNs) as they are described in Chapter 2. In the next chapters, we will lay down the theoretical knowledge needed for the experimental section of the project (Chapters 4, 5, 6 & 7) and the experimental setup as well as the research's empirical results (Chapters 8 & 9).

Chapter 4

Model Distillation

This chapter presents a detailed discussion on model distillation. Starting with the fundamental principles behind compression, followed by a detailed description of the distillation techniques that are used in the experimental analysis in the next chapters. Finally, a discussion about the reasons behind distillation's success is also presented here.

4.1 Overview

As it was mentioned in a previous chapter, the conception of model compression (distillation) is to create a shallower, more concise model to resemble the behaviour of a deep complex neural network, as it is described by the [2.8] equation. Distillation works by passing a large unlabelled dataset through the well-trained deep complex neural network, also known as the teacher model, and collect its predictions on these data points. These predictions will be used as the training labels for the shallow network, known as the student model. The target of distillation is not to make the student model as accurate as possible but to distillate the teacher's knowledge into it. In other words, the student model should mimic / approximate the function learned by the complex network by making the same correct and wrong predictions as the teacher model.

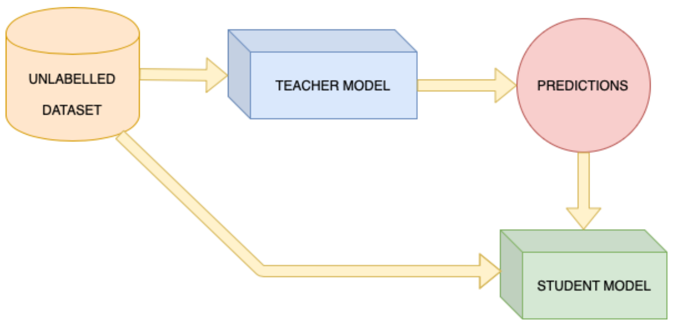


Figure 4.1: Model Compression Pipeline: A large unlabelled dataset is passed through the teacher model. The teacher's predictions are collected and used along with the large unlabelled dataset for the training of the shallow student network.

Research in the field of model distillation has demonstrated that shallow models actually do have the capacity to efficiently learn complex functions. Empirical data, though, shows that shallow models can not achieve accuracy comparable to the teacher models, if they are trained only on the original dataset. The intermediate step of training a high accurate teacher model and collecting its predictions on a new large dataset constitutes a necessary stage in the model distillation process.

4.2 Distillation Techniques

This section analyzes three techniques that are used in the experimental part of the project to study model distillation. However, these methods have not been chosen randomly. All of the techniques offer easy and fast implementation. In particular, the first two techniques, ”Matching the Logits” & ”Ranging the Teacher’s Temperature”, can be frequently found in different research papers on the field and they have both demonstrated significant results in image classification, which is the area that the experimental analysis is focused on. Finally, the ”Adversarial Distillation” technique is inspired by the Adversarial Training of a neural network model and is used only in the final set of experiments.

4.2.1 Matching the Logits

This approach is described and primarily used by [7]. In this technique, the teacher networks are trained as for any classification task, using softmax activation in their output layer and cross entropy loss function. The shallow student networks are trained on the logarithmic probability values z, known as logits, which are inputs to the output softmax layer of the network. For the students’ training, mean square error loss function is used instead of cross entropy loss, effectively turning the classification task into a regression problem. For this technique to produce good quality results, the student models need to be trained on the predictions of the trained teacher models on a new large unlabelled dataset $\mathcal{X}^*$.

We can formulate this distillation technique as the learning objective of a regression problem given the training dataset $\mathcal{X} = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$ and a large unlabelled dataset $\mathcal{X}^* = (x_1^*, x_2^*, ..., x_M^*)$ where $M >> N$. The teacher model f_θ is initially trained on the training dataset $\mathcal{X}$. Using the trained teacher model’s predictions on the unlabelled dataset $\mathcal{X}^*$, the teacher logit $z_i^{f_\theta}$ for each datapoint $x_i^* \in \mathcal{X}^*$ is produced. Then, the learning objective function is defined as [7]:

$$\mathcal{L} = \frac{1}{M} * \sum_{i=1}^{M} (z_i^{f_\theta} - z_i^{g_\theta})^2 \tag{4.1}$$

where $z_i^{g_\theta}$ is the logit produced by the student model for each datapoint $x_i^* \in \mathcal{X}^*$.

With this technique, the shallow student model learns the teacher’s function by placing emphasis on all prediction targets. Instead of just training using only the teacher’s final predictions as the labels, training using the logits provides the logarithmic relationships of probability predictions between the class labels learned by the teacher model.

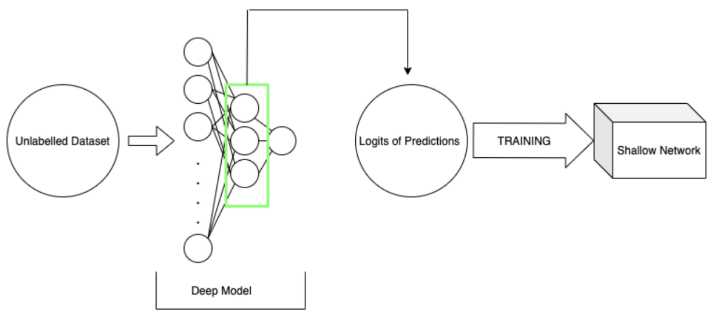


Figure 4.2: Graphical Illustration of the ”Matching the Logits” Distillation Technique. The logits crafted by the teacher model on the large unlabelled dataset are used as the target labels during the student model’s training.

4.2.2 Ranging the Teacher's Temperature

Deep neural networks deployed in classification tasks output probabilities using a softmax activation output layer that turns the logit z_i computed for each class into a probability q_i by comparing z_i with all the other z logits.

$$q_i = \frac{exp(z_i)}{\sum_j exp(z_j)} \tag{4.2}$$

Temperature is a value (T) that the logit is divided by before the softmax activation layer, as described in [4.3]. When the temperature is 1, the softmax activation function is computed directly on the logits, as shown in [4.2]. Using a temperature less than 1, the model computes the softmax function on $\frac{logits}{T}$ resulting in larger arithmetic values compared to those using temperature of 1. Performing softmax activation on larger values creates "hard targets" and makes the model more confident but, at the same time, also more conservative. Using a temperature higher than 1, the model computes the softmax activation on $\frac{logits}{T}$ resulting in smaller arithmetic values compared to those using temperature of 1. Using a higher temperature creates "soft targets" and produces a softer probability distribution over the classes making the model more easily excited and as a result greater diversity but also more mistakes are allowed.

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)} \tag{4.3}$$

This distillation technique is based on tuning this temperature parameter (T) to an ideal region, which depends on the model's architecture as well as the data points used during training, to produce the best soft or hard targets. The shallow student models are trained using cross-entropy loss function on these targets generated by the teacher network when passing a unlabelled dataset through it.

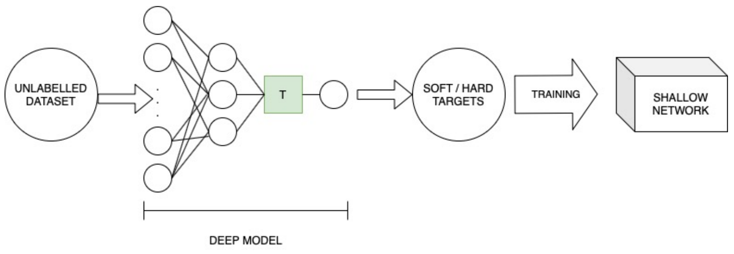


Figure 4.3: Graphical Illustration of the "Ranging the Teacher's Temperature" Distillation Technique. The soft / hard targets generated by dividing the teacher network's logits on the large unlabelled dataset with the temperature parameter (T). These targets are used as target labels during the training of the student model.

This technique is a simpler modified version of the method proposed in [25]. In this paper, the proposed method includes a weighted averaged of these "soft / hard" targets as well as the correct labels (if they are known) of the unlabelled dataset. Interestingly enough, it can be proved mathematically that "Matching the Logits" is a special case of this paper's proposed method in the high temperature limit (high temperature compared to the magnitude of the logits).

However, only the aforementioned simplified method is used throughout the experimental analysis of the project since the labels of the unlabelled datasets are not usually known when conducting model compression.

4.2.3 Adversarial Distillation

This technique is inspired by adversarial training and described in [39]. Adversarial Distillation constitutes a modified version of "Matching the Logits" distillation method. In adversarial training, predictive models are trained using perturbed samples instead of normal data points in order to gain some level of robustness against adversarial attacks during training.

Mathematically, if $\mathcal{X} = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$ is the training set, in adversarial training the data points endure small perturbations and a new training set is produced that consists of $\mathcal{X}_\delta = (x_1^*, y_1), ..., (x_N^*, y_N)$ where

$$x_i^* = x_i + \delta \tag{4.4}$$

with δ being the small perturbation in the space. The predictive models are trained using this new $\mathcal{X}_\delta$ dataset and the clean label targets $((y_1), ..., (y_N))$.

Adversarial distillation is designed around an analogous pipeline. Similar to the "Matching the Logits" method, the shallow student networks are trained on the logarithmic probability values z, known as logits, which are inputs to the output softmax layer of the network. The prediction logits of the teacher model on a new unlabelled dataset $\mathcal{X}^*$ are used as the logit targets for the distillation procedure. The only difference with the previous method lies on the unlabelled dataset that the student models use for training. Instead of using the same unlabelled dataset $\mathcal{X}^*$ with the teacher model, the student model is trained on the perturbed unlabelled dataset $\mathcal{X}_\delta^*$ but still trying to match the "clean logit targets".

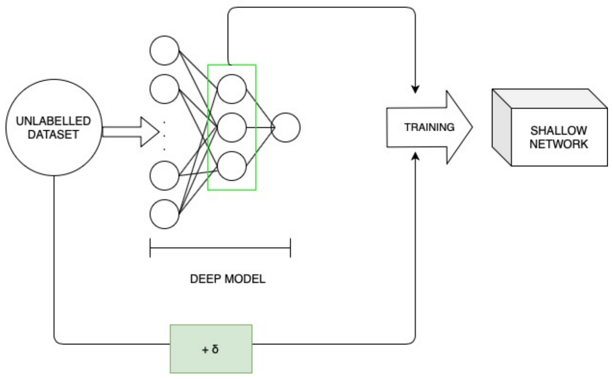


Figure 4.4: Graphical Illustration of the "Adversarial Distillation" Technique. The logits produced by the teacher model on the large unlabelled dataset are used as the target labels for the student model. During the student model's training, the perturbed unlabelled dataset is used.

Adversarial distillation can be formulated as the learning objective of a regression problem given the training dataset $\mathcal{X} = (x_1, y_1), (x_2, y_2), ..., (x_N, y_N)$, a large unlabelled dataset $\mathcal{X}^* = (x_1^*, x_2^*, ..., x_M^*)$ where $M >> N$ and the perturbed $\mathcal{X}_\delta^*$ derived from the unlabelled dataset, as described in [4.4]. The teacher model f_θ is initially trained on the training dataset $\mathcal{X}$. Using the trained teacher model's predictions on the unlabelled dataset $\mathcal{X}^*$, the teacher logit $z_i^{f_\theta}$ for each datapoint $x_i^* \in \mathcal{X}^*$ is produced. Then, the learning objective function is defined as:

$$\mathcal{L} = \frac{1}{M} * \sum_{i=1}^{M} (z_i^{f_\theta} - z_i^{g_\theta})^2 \tag{4.5}$$

where $z_i^{g_\theta}$ is the logit produced by the student model for each datapoint $x_i^* \in \mathcal{X}_\delta^*$.

4.3 Discussion

As it has been shown in the literature, model distillation enables networks to achieve higher accuracy compared to the same architecture networks trained directly on the original data. There is a variety of reasons why this can happen:

1. Some datasets include errors in their training labels that might confuse the network. A well-trained teacher model can eliminate some of these errors making effectively the learning process easier for the shallow student model.
2. With methods like ”Matching the Logits” and ”Adversarial Distillation” where logits are used as the targets for the distillation procedure, the student model learns to mimic the complex function by having more information from the deep complex teacher model. Instead of just using the original labels, the student models are given now information about the different classes and their relationship with each other. E.g. the logits [15, 10,20,30] and [5,-10,0,10] will both result in class 4 prediction but the differences between the classes are completely different and contain a significant amount of knowledge information that boosts the performance of the student network.
3. Learning from the original labels can be extremely difficult sometimes for a shallow neural network, especially if the complexity of the target function is really high. Distillation targets, on the other hand, are far easier to learn, since the complex teacher model through methods like ”Ranging the Teacher's Temperature” can produce and provide the student model with simpler labels.
4. In methods like ”Matching the Logits” and ”Adversarial Distillation”, even if the teacher models make wrong predictions, the uncertainty of the complex network and its produced logits can be far more informative to guide the student network to learn a more accurate target function than the original labels because of the extra information they convey.

Although model distillation has the potential produce shallow models with high accuracy comparable to complex deep neural networks, research has demonstrated that high accuracy comes with an increase in the amount of parameters. In their paper Caruana et al. (as described in [27]) have showed (as we also investigate in the experimental section of the project in the next chapters) that the student models, although have fewer number of layers, they need to be wider (need to have more hidden units for fully-connected layers or more channels for convolutional layers). To overcome this drawback and be able to produce shallow neural networks with a small amount of parameters, parameter reduction methods will be used. In the next chapter, the parameter reduction technique, that is used in our experimental analysis in Chapter 8, is described in detail.

Chapter 5

Parameter Reduction

The previous chapter analysed model distillation of complex deep neural networks and described different distillation techniques in detail. As it was clearly stated, shallower models tend to need a much larger number of parameters in order to be efficient and have a clean accuracy comparable to their teacher complex neural networks. What happens, though, if the neural networks need to operate under a certain parameter budget ? This chapter addresses this problem and investigates whether neural networks can go under a certain parameter reduction without affecting their overall performance.

5.1 Overview

Over the last years, there has been a broad tendency in the machine learning world to design, build and train deeper neural networks with an overall increase in the number of parameters and convolution operations. These deep complex neural networks, although extremely accurate, come with significant costs when employed in mobile devices, sensors or embedded systems where power and computational resources may be limited.

Different methodologies have been developed over the years to decrease the storage (number of parameters) of deep neural network models. In the early years, research was focused on removing weights with small magnitudes from the fully-connected part (also known as classification part) of a Convolutional Neural Network (CNN). This process could be followed by a retraining of the smaller model to regain its original accuracy [4]. However, some of the problems remained intact. Although, with these methods a significant amount of parameters could potentially be eliminated, since the majority of parameters is located in the fully-connected part of a convolutional neural network, in principle many disadvantages were still present. The parameters in the fully-connected part of a CNN constitute the majority of the network's total parameters but they contribute only a small amount to the total computational time of the network. Nowadays, many deep complex convolutional neural networks are located in an online server and they have to function under a certain time budget. Recent efforts to reduce parameters have been focused on removing unnecessary filter weights from the convolutional layers located in the feature extraction part of a CNN. In this way, they reduce both the number of parameters of the network along with its total computational cost.

In this section, Pruning Filters [30], a method to efficiently remove parameters from the feature extraction part of a CNN, is described in detail. This method is used in the experimental part of the project that is presented in the Chapter 8. There have been several reasons for choosing to make use of this particular method:

1. Deep CNNs with large capacity tend to have a large amount of redundancy among its different filters and feature channels.

2. By removing non-essential filters, both the number of parameters as well as the computational cost of the network are significantly reduced.

3. In the previous chapter, it was stated that distilled models need a increase in their parameter budget in order to have accuracies comparable to the teacher model. A significant amount of the experimental section of the project is oriented around the relationship between the number of parameters in the feature extraction part of a Convolutional Neural Network and its effect on the distillation accuracy. Under these conditions, it is only logical to investigate parameter reduction methods targeting the convolutional layers of a CNN and create models which combine the benefits of model distillation and parameter reduction.

4. This method is both efficient and easy to implement. Unlike pruning weights, pruning filters has no requirement of extra specialized hardware.

5.2 Pruning Filters

Pruning Filters of a Convolutional Neural Network is a parameter reduction method that is described in [30] and is used in the experimental part of this thesis project for reasons that are elaborated above. This section targets to outline the pruning process as well as its advantages and provide a mathematical analysis of this method as it is detailedly described in [30].

5.2.1 Method Description

”Pruning filters” [30] is a method to ”efficiently prune filters of well-trained Convolutional Neural Networks (CNNs) with relatively low weight magnitudes to produce new CNNs with reduced computation costs without introducing irregular sparsity while minimising the accuracy drop”. The method is based on the computation of L1-norm on the different filters. In each layer of the CNN, the sum of its absolute weights is calculated. The method involves pruning a certain number of filters with the smallest sum of absolute weights. ”When a filter is pruned, its feature map and its corresponding kernel in the next convolutional layer are also removed from the new network”.

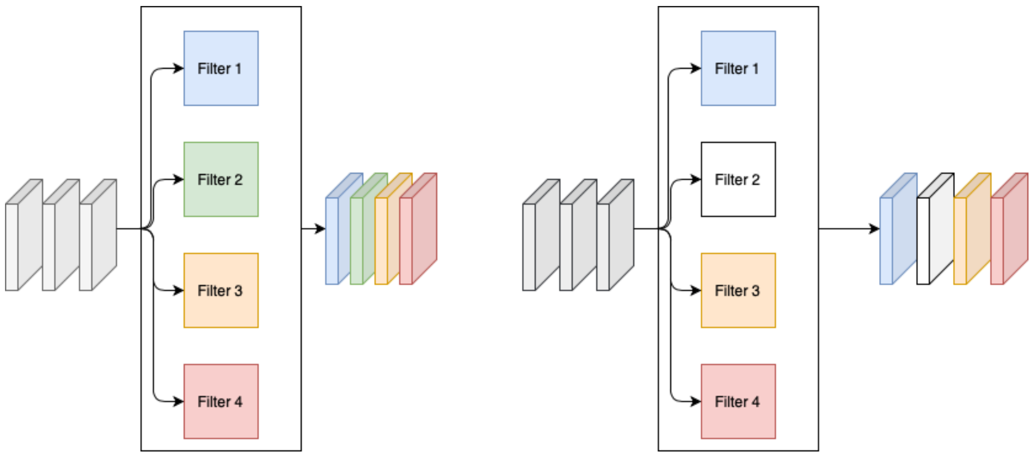


Figure 5.1: Graphical Illustration of Pruning Filters Method. (Left) The convolution process from the ith layer to i+1 layer of a well-trained network without filter pruning. (Right) The convolution process from the ith layer to i+1 layer of a well-trained network where Filter2 is pruned. White color indicates that this filter and its corresponding filter map have been removed.

5.2.2 Method Mathematical Analysis

In the i-th convolutional layer, let n_i denote the input channels and with h_i and w_i the height and width of the input feature maps respectively. Each convolutional layer takes the feature maps $x_i \in R^{n_i * h_i * w_i}$ as the inputs, convolves it and produces the output feature maps $x_{i+1} \in R^{n_{i+1} * h_{i+1} * w_{i+1}}$. This convolutional operation is achieved, as it is described in Chapter 2, by applying filters denoted by $\mathcal{F} \in R^{n_i * k * k}$ on the n_i input channels where each filter generates a feature map and consists of n_i kernels $\mathcal{K} \in R^{k*k}$.

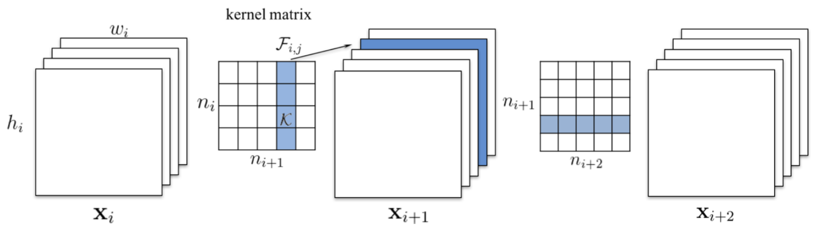


Figure 5.2: Graphical Illustration of Pruning Filters Method. In the figure, the blue filter is pruned resulting in the removal of its corresponding feature map and related kernels in the next layer. Source: [30]

According to this method, the filter's significance in each layer is determined by its L1-norm $||\mathcal{F}_{i,j}||_1 = \sum |\mathcal{F}_{i,j}|$. "The importance of this sum comes from the fact that small sums tend to produce feature maps with insignificant contribution to the network's output. The parameter reduction is achieved by pruning these weak filters, their corresponding filter maps and kernels in the next convolutional layer".

Algorithm 1: Pseudo-algorithm for the Pruning Filters Method [30]

```
Step 1:
For every filter $\mathcal{F}_{i,j}$, compute the sum of its absolute kernel weights $s_j = \sum_{k=1}^{n_i} \sum |\mathcal{K}_k|$;
while (There is no drop in the model's accuracy) do
    Step 2: Prune k filters with the smallest sum of absolute kernel weights. Remove also
      their associated kernels in the next convolutional layer ;
    Step 3: Generate a new kernel matrix for both layers i and i+1. Duplicate the
      remaining kernel weights to the new network ;
    Step 4: Increase the k filters to be pruned ;
end
```

5.2.3 Computational and Parameter Benefits

The total number of operations from the ith layer to the next equal $n_{i+1} * n_i * k^2 * h_{i+1} * w_{i+1}$. As demonstrated in Figure 5.2.2, when one filter $\mathcal{F}_{i,j}$ is pruned, its corresponding map and the associated kernel in the next layer are eliminated as well. These actions save $n_i * k^2 * h_{i+1} * w_{i+1}$ operations for the corresponding map and $n_{i+2} * k^2 * h_{i+2} * w_{i+2}$ operations for the associated kernel. In terms of computation cost, pruning k filters of layer i results in a reduction of k/ni+1 of the computation cost for both layers (layer i and layer i + 1).

Chapter 6

Adversarial Robustness

In Chapter 2, there has been a short introduction to the different categories of adversarial attacks grouped by factors like their influence, their specificity, their knowledge of the system under attack as well as their frequency and target. In this chapter, a wide range of adversarial attacks are presented and described in full detail. These attacks are used to measure adversarial robustness of a complex deep neural network as well as its distillated shallow model in the experimental section in Chapter 8.

6.1 Overview

Adversarial examples, as described in the equation [2.9], are input samples to machine learning models created by performing small perturbations to the original clean data points with the intent of misleading and fooling the machine learning algorithms. The majority of models can be fooled by perturbations so small in magnitude to be perceptible to the human eye. As it is described in [31] by Goodfellow, a variety of research experiments has demonstrated that "different network architectures trained on different subsets of the same training dataset $\mathcal{X}$ tend to misclassify the same adversarial example", effectively exposing weaknesses in the current machine learning training algorithms. Initial explanations of the adversarial examples' source were oriented around the idea of non-linearity of the deep neural networks. However, Goodfellow et al. in their work disproved this hypothesis.

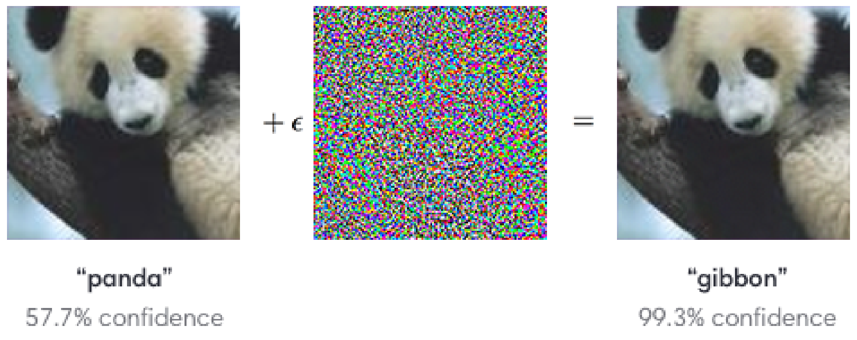


Figure 6.1: Adversarial Example that causes the misclassification of the input image with high confidence. Source: [31]

Many state-of-the-art deep neural networks, even though they have excellent performance on their respective test sets, fail to learn the true underlying principles, a fact that makes them susceptible to adversarial attacks.

Effective defense against adversarial examples is difficult to find since there is no theoretical model of the construction process of adversarial examples.

Definition 6.1 In general, adversarial examples are solutions to the following non-linear and non-convex optimisation problem:

$$\begin{gathered}\min_{\delta} ||\delta|| \\ \text{such that } f_\theta(x+\delta) = p \\ \text{with } x+\delta \in \mathcal{B}_\varepsilon\end{gathered} \tag{6.1}$$

where f_θ be a well-trained predictive, $f_\theta(x) = p$ and $\mathcal{B}_\varepsilon$ a norm-bounded perturbation ball around the data point x $\in \mathcal{X}$.

Since this optimization problem is non-linear and non-convex, there is no efficient theoretical tool for fully describing its solutions. Another reason for neural networks' weak robustness is the amount of data they use for training and testing, a very small amount of all the many possible inputs they might encounter in the real world. Finally, although different types of defenses have been designed over the years, these are neither adaptive nor universal. In other words, they may prevent one adversarial attack method while they leave other vulnerabilities exposed and ready to be used in another type of adversarial attack.

6.2 Types of Adversarial Attacks

In this section, the different types of attacks used in the experimental section in Chapter 8 are described in detail. These methods will be used to assess and compare the adversarial robustness of both the complex teacher neural network and its distillated shallow student model.

6.2.1 Fast Gradient Sign Method

The Fast Gradient Sign Method (FGSM) is described by Goodfellow et. al. in [31] and is one of the most computationally efficient adversarial attack and very easy to implement. Its implementation involves the computation of the sign of cost function of the predictive model. By adding a small amount of error value to each data sample following with the direction of the sign of the cost function, this method can produce efficient adversarial examples.

Definition 6.2 Mathematically, Fast Gradient Sign Method (FGSM) can be represented by the following equation [31]:

$$x^* \Leftarrow x + \varepsilon * sgn(\nabla_x J(x, f, \theta)) \tag{6.2}$$

where x is the original sample, $J(x, f, \theta)$ is the cost function and the parameter ε refers to the l_∞ perturbation and defined as $\varepsilon := ||x^* - x||_\infty$.

Early experiments have shown that this type of attack, despite its simplicity, is remarkably powerful. As it is obvious from the equation above, the FGSM attack is designed to attack neural networks by exploiting their learning procedure through gradients. The idea behind this technique is to modify the input data sample in order to maximize, instead of minimizing (target of model's training), the cost loss function based on the same backpropagated gradients. Alternatively stated, this technique makes use of the gradients of the loss function to create adversarial examples that maximize the cost function $J(x, f, \theta)$.

6.2.2 Projected Gradient Descent

The Projected Gradient Descent (PGD) method is described in [3] and on the surface looks very similar to the Fast Gradient Sign Method (FGSM) attack. The main difference between the two methods is the frequency of attack. Fast Gradient Sign Method is a One-Time attack whereas Projected Gradient Descent is Iterative. Instead of calculating the gradient of the cost function once and adding a large error value, Projected Gradient Descent calculates a new gradient in each iteration and adds a much smaller perturbation to each data point. In short, Projected Gradient Descent constitutes a more powerful, multi-step version of the Fast Gradient Sign Method.

Definition 6.3 Mathematically, Projected Gradient Descent (PGD) can be represented by the following equation [3]:

$$x^*_{t+1} \Leftarrow \Pi_{x+\mathcal{S}}(x^*_t + \alpha * sgn(\nabla_x J(x, f, \theta)) \tag{6.3}$$

where x is the original sample, $J(x, f, \theta)$ is the cost function, $\mathcal{S}$ the set of all samples in the dataset and the parameter α refers to the small l_∞ perturbation constant.

Further research on the field of adversarial attacks has demonstrated that all the local maxima found by PGD have similar loss values, both for normally trained and adversarially trained neural networks. This finding (in [3]) suggests that robustness against the PGD adversary yields robustness against all first-order adversaries (in other words attacks that rely only on first-order information). It is proven that as long as the adversary uses only gradients of the loss function with respect to the input, there is no better local maxima than the ones found by PGD. Therefore in the community, PGD attack with sufficient iterations is considered a strong state-of-the-art attack. For that reason, PGD robustness will be the main concentration in the experimental part of the project.

6.2.3 Basic Iterative Method

The Basic Iterative Method is described in [6] and is another variant of the Fast Gradient Method (FGSM) mostly targeting images. Like the Projected Gradient Descent, this method is iterative. The technique involves the computation of the gradient numerous times using a small step size and clipping the pixel values to guarantee that are in an ε-neighbourhood of the original input.

Definition 6.4 Mathematically, Basic Iterative Method (BIM) can be represented by the following equation [6]:

$$x^*_{t+1} \Leftarrow Clip_{x,\varepsilon}(x^*_t + a * sgn(\nabla)_x J(x^*_t, f, \theta)) \tag{6.4}$$

where x is the original sample, $J(x^*_t, f, \theta)$ is the cost function and the parameter α refers to the small l_∞ perturbation constant.

If **X** is a source image, the Clip function is a function which performs per-pixel clipping of that image **X′**, so the result will be in the l_∞ ε-neighbourhood of the source image **X**. In

$$Clip_{x,\varepsilon}(\mathbf{X'})(x, y, z) = \min(255, \mathbf{X}(x, y, z) + \varepsilon, \max(0, \mathbf{X} - \varepsilon, \mathbf{X'})) \tag{6.5}$$

where $\mathbf{X}(x, y, z)$ is the value of channel z of the image **X** at coordinates (x, y).

6.2.4 Momentum Iterative Method

The Momentum Iterative Method is described in [49] and constitutes an improvement of the fast gradient method (FGM), a generalization of FGSM which is defined as [49]:

Definition 6.5

$$x^* \Leftarrow x + \varepsilon * \frac{\nabla_x J(x, f, \theta))}{||\nabla_x J(x, f, \theta))||_2} \tag{6.6}$$

where x is the original sample, $J(x, f, \theta)$ is the cost function and the parameter ε refers to the l_∞ perturbation and defined as $\varepsilon := ||x^* - x||_\infty$ and adversarial example meets the L2-norm bound $||x^* - x||_2 \leq \varepsilon$.

In mathematics, the momentum technique accelerates the solution of gradient algorithms by making use of a correction vector. In each iteration, this vector gets updated and the accumulation of the previous gradients contribute to the success of this attack by helping the algorithm to avoid essentially week local maxima or minima.

Definition 6.6 Mathematically, the Momentum Iterative Method (MIM) can be represented by the following equation [49]:

$$\begin{aligned} x^* &\Leftarrow x + \alpha * sign(g_{t+1}) \\ \text{with } g_{t+1} &\Leftarrow g_t + \frac{\nabla_x J(x^*, f, \theta))}{||\nabla_x J(x^*, f, \theta))||_1} \end{aligned} \tag{6.7}$$

where x is the original sample, $J(x_t^*, f, \theta)$ is the cost function and the parameter α refers to the small l_∞ perturbation constant.

6.3 Adversarial Robustness

Robustness against adversarial attacks is a desired property for the contemporary state-of-the-art neural networks and therefore it is an active field of research in the area of Artificial Intelligence. In machine learning, the goal of any predictive f_θ model trained on a training set $\mathcal{X}$ with an underlying distribution $\mathcal{D}$ is to minimize the Empirical Risk Minimization (ERM) which is defined as:

$$\mathcal{E}_{(x,y)\sim\mathcal{D}}[\mathcal{L}(x, f, \theta)] \tag{6.8}$$

where $\mathcal{L}$ is the loss function.

Unfortunately, solutions to the Empirical Risk Minimization (ERM) problem, although they provide high accuracy predictive models for a certain task, they leave many exposed weaknesses which can be exploited by modern adversarial attack algorithms (like the aforementioned ones described in section 6.2).

The ultimate goal of this field of research is to provide the framework where predictive models can both learn the target function and also be robust to adversarial attacks, in other words gain adversarial robustness. To achieve adversarial robustness though, the neural network's training process needs to change and the Empirical Risk Minimization (ERM) optimization problem needs to be slightly modified.

Definition 6.7 Let f_θ be a predictive model trained on a training set $\mathcal{X}$ with an underlying distribution $\mathcal{D}$. Adversarial Robustness can be seen as the following optimization problem of an adversarially modified Empirical risk minimization (ERM) function:

$$\begin{gathered}\min_{\theta} H(\theta) \\ \text{where } H(\theta) = \mathcal{E}_{(x,y)\sim\mathcal{D}}[\max_{\delta\in\mathcal{S}} \mathcal{L}(x+\delta, f, \theta)]\end{gathered} \tag{6.9}$$

For each data point $x \in \mathcal{X}$, $\mathcal{S}$ is the set of allowed perturbations.

In Chapter 8, using the aforementioned types of adversarial attacks, a comparison of the adversarial robustness between the teacher and the student model is presented. The reason behind this experiment is to examine whether model distillation alters the adversarial robustness of the teacher model by making the network shallower. In other words, if model distillation boosts, aggravates or leaves intact the adversarial robustness of the shallow student model compared to the teacher model under a series of adversarial attacks in a wide variety of settings.

Chapter 7

Robust Classifier & Certification Properties

The previous chapter analysed adversarial examples, presented some of the most common adversarial attack methods and provided a review of the concept of adversarial robustness. This chapter addresses the area of verification of neural networks. Emphasis will be given on designing and building robust ReLU-based classifiers with a method based on network's relaxation. Finally, detailed definitions of different certification properties of a neural network are also presented in this chapter.

7.1 Overview

Adversarial examples, as they are described in Chapter 6, are specifically designed data points which are indistinguishable from the authentic to the human eye but they are built in a way that enables them to fool the machine learning systems. No matter the defenses that are used in today's machine learning models, the potential risk of a classifier being fooled is extremely high. Therefore, classifiers need to be designed such as they are guaranteed to be robust to different types of adversarial perturbations. As it is described in Chapter 2, verification is an optimization problem that effectively examines if there is any input in the perturbation ball $\mathcal{B}_\varepsilon$ for which the model's prediction belongs to an incorrect class. To solve this optimization equation [2.10], the problem needs to be "relaxed" at some point whether this is in its primal formulation (equivalently the forward pass through the network) or in its dual formulation (equivalently in the backward pass through the network). Based on the tightness of the solution, verification can be categorized into "Tight Verification" and "Non-tight Verification".

Definition 7.1 Let x be any input in $\mathcal{X}$. Tight verification can be expressed mathematically as [35]:

$$f_\theta(x') \begin{cases} = f_\theta(x), & \forall x' \in \mathcal{B}_\varepsilon \\ \neq f_\theta(x) \end{cases} \tag{7.1}$$

where f_θ is a well-trained predictive model and $\mathcal{B}_\varepsilon$ denotes a norm-bounded perturbation ball around the data point x $\in \mathcal{X}$. If the equality holds, the network certifies while in the second case there is an adversarial example in the perturbation ball $\mathcal{B}_\varepsilon$ for this particular f_θ predictive model.

In tight verification, the relaxed adversarial polytope has one point in common with the boundary of the true adversarial polytope while this differs in non-tight verification methods. This chapter is oriented around non-tight verification and its use to design classifiers which are "provably robust against norm-bounded adversarial perturbations via a convex outer adversarial polytope" [47]. This method will be used to train robust classifiers in the experimental section of this project and compare their certification area with their equivalent shallow distillated models. Finally, this method has been chosen as it provides a way to design provably robust classifiers with some of the best test error bounds in the literature.

7.2 Network's Relaxation

Definition 7.2 For a neural network, the robust optimization problem [2.10] can be expressed mathematically as follows [29]:

$$\begin{gathered}\min_{\hat{z}_k} c * \hat{z}_k + d \\ \text{subject to } z_1 \in \mathcal{B}_\varepsilon \text{ (input constraint)} \\ \hat{z}_{i+1} = W_i * z_i + b_i,\ \text{i = 1,..,K (affine constraints)} \\ z_i = h(\hat{z}_i),\ \text{i = 2,..,K (activation constraints)}\end{gathered} \tag{7.2}$$

where $\mathcal{Z}_\varepsilon$ is the adversarial polytope and is defined as $\mathcal{Z}_\varepsilon(x) := (z|z = f_\theta(x')\ \ \forall x' \in \mathcal{B}_\varepsilon$.

If the minimum value of the objective function $c * \hat{z}_k + d$ for all $\hat{z}_k \in \mathcal{Z}_\varepsilon$ is positive then it guarantees that the networks certifies. In other words, there exists no input in $\mathcal{B}_\varepsilon(x)$ that can change the prediction of the network for a given input x. If the activation function h is not convex then both the above mentioned optimization problem as well as the adversarial polytope $\mathcal{Z}_\varepsilon(x)$ are non-convex. By replacing the activation function h with a set of constraints defining a convex set $\hat{h}(\hat{z}_i, l_i, u_i)$, where l_i , u_i are the upper and lower bounds in the range of $\hat{z}_i$, the problem can be turned into convex. This approximation $\hat{h}(\hat{z}_i, l_i, u_i)$ is referred to as the Convex Approximation technique [29].

Definition 7.3 If the perturbations are norm-bounded, the $\mathcal{Z}_\varepsilon(x)$ adversarial polytope is defined as:

$$\mathcal{Z}_\varepsilon(x) = \{f_\theta(x + \Delta) : ||\Delta||_\infty \leq \varepsilon\} \tag{7.3}$$

In the experimental part of the report, ReLU based classifiers are extensively used and therefore the rest of the optimization problem's solution analysis will be oriented around ReLU activations. If the activation h is ReLU and the predictive model is a k-layer feedforward neural network then the [7.2] problem equations can be re-written as:

$$\begin{gathered}z_1 \in \mathcal{B}_\varepsilon \\ \hat{z}_{i+1} = W_i * z_i + b_i,\ \text{i = 1,..,K} \\ z_i = \max(\hat{z}_i, 0),\ \text{i = 2,..,K}\end{gathered} \tag{7.4}$$

The following figure demonstrates the linear relaxation of the ReLU activation function:

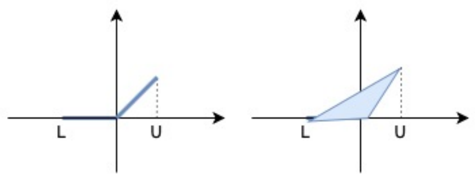


Figure 7.1: Convex ReLU function relaxation over the bounded [L,U]

”Given known lower and upper bounds for the pre-ReLU activations, the ReLU-activation h can be replaced in the aforementioned equations” by [47] (see [47] Appendix for proof):

$$z \geq 0, z \geq \hat{z}, -u * \hat{z} + (u - l) * z \leq -u * l \tag{7.5}$$

7.3 Provably Robust ReLU-based Classifier

This sections presents the methodology and mathematical derivations of the method described in [47] to train deep ReLU networks that are provably robust against norm-bounded adversarial perturbations. The method uses relaxation of the ReLU activation function and consists of two steps: construct the ”outer convex adversarial polytope” and efficiently optimize over this bound using the dual problem formulation.

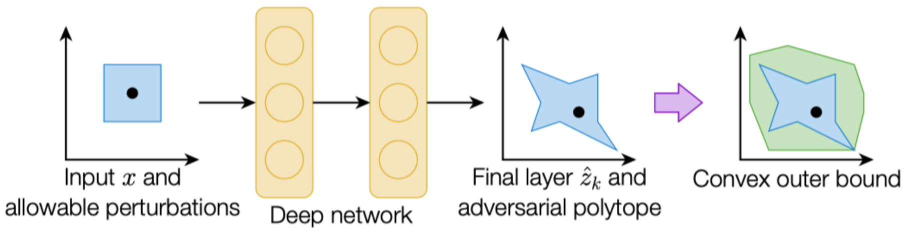

Input x and allowable perturbations
Deep network
Final layer $\hat{z}_k$ and adversarial polytope
Convex outer bound

Figure 7.2: Demonstration of the real adversarial polytope and the external convex polytope. Source: [47]

This convex outer bound is effectively used to provide certification guarantees for the classifier. As it is easily noticeable from the illustration above, if no data point inside this external convex approximation (green region) exists that can fool the network and change its prediction, then it is guaranteed that there is also no data point inside the true polytope (blue region) that can achieve that. The robustness of the classifier is derived by the minimization of the worst-case loss over the external convex polytope or equivalently by solving the following optimization problem [47]:

Definition 7.4

$$\begin{gathered} \min_{\hat{z}_k} (\hat{z}_k)_{y^*} - (\hat{z}_k)_{y^(target)} = c^T * \hat{z}_k \\ \text{subject to } \hat{z}_k \in \hat{\mathcal{Z}}_\varepsilon \text{ where } c := e_{y^*} - e_{y^(target)} \end{gathered} \tag{7.6}$$

where $x \in \mathcal{X}$ is a sample, y^* its label and $\hat{\mathcal{Z}}_\varepsilon$ denotes the convex outer adversarial polytope.

In other words, given an example point x, by solving [7.6] the point in $\hat{\mathcal{Z}}_\varepsilon$ that minimizes the true class and maximizes some other target class can be computed. This constitutes a Linear Program (LP). If the solutions of the Linear Program for all target classes are positive, there is no adversarial norm-bounded perturbation of the given example point x that could potentially mislead the network. This method though has two main drawbacks:

1. Although it can detect all adversarial example points, it might misclassify some normal example points as well. In other words, this method provides zero false negatives but might classify some normal points as possible adversarial examples.
2. This Linear Program’s solution is neither efficient nor tractable for every target class in large deep complex neural networks since the amount of variables of the problem equals to the total amount of all network’s activations.

7.4 Efficient Optimization

The optimization problem [7.4] can be solved efficiently by making use of the dual formulation of the problem. As described in [47], the dual problem's form resembles an adjusted version of the backprogration neural network. This effectively translates to the fact that a provable robust lower bound of the primal formulation [7.4] can be efficiently computed with only one backward pass of this modified neural network.

Definition 7.5 The dual problem is given by the following formulation (the following equations and solutions to the problem are derived by [47]):

$$\begin{gathered} \max_{\alpha} \mathcal{J}_{\varepsilon}(x_{\varepsilon}, g_{\theta}(c, \alpha)) \\ \text{where } \alpha_{i,j} \in [0,1], \quad \forall i,j \end{gathered} \tag{7.7}$$

The $g_{\theta}(c,\alpha))$ is the k-layer backpropagation network defined for $i = k-1, ..., 2$ as:

$$\begin{gathered} v_k = -c \\ v_i = W_i^T v_{i+1}, \quad \text{for } i = k-1, ..., 1 \\ v_{i,j} = \begin{cases} 0, & j \in \mathcal{I}_i^- \\ v_{i,j}, & j \in \mathcal{I}_i^+ \\ \dfrac{u_{i,j}}{u_{i,j} - l_{i,j}}[\hat{v}_{i,j}]_+ - \alpha_{i,j}[\hat{v}_{i,j}]_-, & j \in \mathcal{I}_i \end{cases} \end{gathered} \tag{7.8}$$

where $\mathcal{I}_i^-, \mathcal{I}_i^+, \mathcal{I}_i$ denote the sets of activations in layer i in which the bounds are both negative, positive or span zero respectively.

While the $\mathcal{J}_{\varepsilon}(x,v)$ is equal to:

$$-\sum_{i=1}^{k-1} v_{i+1}^T b_i - x^T \hat{v}_1 - \varepsilon ||\hat{v}_1||_1 + \sum_{i=2}^{k-1} \sum_{j \in \mathcal{I}_i} l_{i,j} |v_{i,j}|_+ \tag{7.9}$$

As it is obvious, the dual formulation network is similar to the backpropagation network with one additional free variable $\alpha_{i,j}$ that can be optimized to improve the overall solution. For reasons that are beyond the scope of this project, the $\alpha_{i,j}$ is fixed in the method [47] that will be used in our experiments and is equal to:

$$\alpha_{i,j} = \frac{u_{i,j}}{u_{i,j} - l_{i,j}} \tag{7.10}$$

Final step to conclude the mathematical description of this method to train robustly ReLU classifiers is the computation of the neccessary bounds to form the $\mathcal{I}_i^-, \mathcal{I}_i^+, \mathcal{I}_i$ sets. These element-wise bounds can be computed using the "Computing Activation Bounds" algorithms described below. The algorithm (described in [47]) uses a matrix to cumulatively calculate the backward passes layer-by-layer and compute the lower and upper bounds in each pass.

Algorithm 2: Computing Activation Bounds Algorithm [47]

```
input: Given a network $f_\theta$, its parameters $(W_i, b_i)$, data point x and ball size $\varepsilon$;
//initialization;
$\hat{v}_1 := W_1^T$ , $\gamma_1 = b_1^T$;
$l_2 = x^T W_1^T + b_1^T - \varepsilon||W_1^T||_{1,:}$, $u_2 = x^T W_1^T + b_1^T + \varepsilon||W_1^T||_{1,:}$ ;
// $|| * ||_{1,:}$ for a matrix denoted $l_1$ norm for all columns ;
for $i = 2, ..., k-1$ do: ;
    //initialize new terms ;
    form $I_i^-, I_i^+, I_i, D_i$ ;
    $\nu_{i,j} = (D_i)_{I_i} W_i^T, \gamma_i = b_i^T$ ;
    //propagate existing terms ;
    $\nu_{j,\mathcal{I}_i} = \nu_{j,\mathcal{I}_i} D_i W_i^T$ for $j = 2, ...i-1$ ;
    $\gamma_j = \gamma_j D_i W_i^T$ for $j = 1, ...i-1$ ;
    $\hat{v}_1 = \hat{v}_1 D_i W_i^T$ ;
    //compute bounds ;
    $\varphi_i = x^T \hat{v}_1 + \sum_{j=1}^{i} \gamma_j$ ;
    $l_{i+1} = \varphi_i - \varepsilon||\hat{v}_1||_{1,:} + \sum_{j=2}^{i} \sum_{i' \in \mathcal{I}_\rangle} l_{j,i'}[-\nu_{j,i'}]_+$;
    $u_{i+1} = \varphi_i + \varepsilon||\hat{v}_1||_{1,:} - \sum_{j=2}^{i} \sum_{i' \in \mathcal{I}_\rangle} l_{j,i'}[\nu_{j,i'}]_+$;
end for ;
output: bounds $(l_i, u_i)$
```

where D a diagonal matrix defined as:

$$(D_i)_{i,j} = \begin{cases} 0, & j \in \mathcal{I}_i^- \\ 1, & j \in \mathcal{I}_i^+ \\ a_{i,j}, & j \in \mathcal{I}_i \ \ and \ \ a_{i,j} \ \ as \ \ in \ \ [7.10] \end{cases} \tag{7.11}$$

Using the lower bounds developed through the dual formulation of the optimization problem and the "Computing Activation Bounds" algorithm, a robust classifier to norm-bounded perturbations can be designed and built by minimizing the worst case loss over convex outer polytope [47]:

Definition 7.6 Training a robust classifier to norm-bounded perturbations is an optimization problem which can be expressed as [47]:

$$\begin{gathered}\min_{\theta} \mathcal{H}(\theta) \\ \text{where: } \mathcal{H}(\theta) = \sum_{i=1}^{N} \max_{||\Delta||_\infty \leq \varepsilon} L(f_\theta(x_i + \Delta), y_i)\end{gathered} \tag{7.12}$$

where f_θ is the classifier, the dataset $\mathcal{X} = (x_1, y_1), ..., (x_N, y_N)$ ε defines the ball distance around each point $x_i \in \mathcal{X}$, L the loss function while Δ is the perturbation.

7.5 Certification Properties

In this section, certification properties of neural networks (using the bounds analyzed in detail in the previous section) are discussed. The methods are described in [47] and include Robust Error Bound and Certification Area. These properties will be extensively evaluated in the last experimental section of the project and their definitions are presented in this section. For completeness a full proof of the two properties' definitions can be found in the Appendix A2.

7.5.1 Robust Error Bound

Definition 7.7 For a data point $x \in \mathcal{X}$ and label y^*, the model is guaranteed to be robust in the perturbation ball $\mathcal{B}_\varepsilon$ around x if and only if [47]:

$$\mathcal{J}_\varepsilon(x, g_\theta(e_{y^*} 1^T - I)) \geq 0 \tag{7.13}$$

where $\mathcal{J}$ and g_θ are defined in 7.7.

In other words, if the above relationship holds there is no x^* example in the perturbation ball $\mathcal{B}_\varepsilon$ around x, $||x^* - x||_\infty \leq \varepsilon$ for $\varepsilon > 0$, such that the output of the predictive is an incorrect class ($f_\theta \neq y^*$).

7.5.2 Certification Area

Certification area of a dataset $\mathcal{X}$ is defined as the set of ε distances for which a certification guarantee exists for each point $x \in \mathcal{X}$. In other words, the maximum ε value for each point $x \in \mathcal{X}$ in which the classifier will not change its prediction.

Definition 7.8 The certification area can be expressed mathematically as the following optimization problem [47]:

$$\begin{gathered}\text{maximize } \varepsilon \\ \text{such that } \mathcal{J}_\varepsilon(x, g_\theta(e_{f_\theta(x)} 1^T - I)) \geq 0\end{gathered} \tag{7.14}$$

The computation of the certificate of robustness, largest ε value, for each data point $\in \mathcal{X}$ can be computed using numerical methods. In the experimental section of the project, Newton's method (see Appendix A1) will be used.

Chapter 8

Experiments

In this chapter, the experiments in the field of model distillation, parameter reduction and adversarial robustness (as described in Chapters 4, 5 & 6) are presented. Using the lessons learnt from these experiments, a robust classifier, as discussed in Chapter 7, is implemented and is used as the teacher model in a series of distillation experiments to compute the certification properties of the shallow networks and answer the questions outlined in Chapter 3. Development is undertaken in Python, a high level programming language which supports a large number of Machine Learning languages which are essential for the development and evaluation of deep learning models. During the following experiments, two of these languages are used: Keras (Tensorflow Backend) and PyTorch.

8.1 Datasets

SVHN: SVHN [2] is a real-world image dataset for developing machine learning and object recognition algorithms. It contains over 600,000 digit images that come from a significantly hard, unsolved, real world problem (recognizing digits and numbers in natural scenery images). ”SVHN dataset is acquired from house numbers in Google StreetView images”. It includes 73257 digits for training, 26032 digits for testing, and a 531131 images additional extra training data that comes with labels. In the experiments, the 73257 digits training set is referred to as the standard training set, the 26032 digits testing set as the test set and 531131 additional extra training data as the unlabelled or extra dataset.

Figure 8.1: Samples from the SVHN Dataset. Source: [2]

The SVHN dataset consists of 32-by-32 colored (with 3 channels) images. Although the dataset can be used as it stands, the following modification is performed: A majority of the images usually include three numbers in their ”cropped-digits” version but their labels refer only to the middle digit. As a result in the following experiments, the images are cropped to contain only the middle number and the dataset is reformed to 32-by-17 colored images.

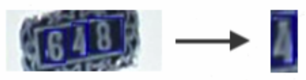


Figure 8.2: Example of the SVHN Dataset Image Pre-processing

CIFAR-10: CIFAR-10 is another dataset for developing object recognition algorithms based on machine learning models. ”It is a labeled subset of the 80 million tiny images dataset [1] collected by Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. The CIFAR-10 dataset consists of 60000 32x32 colour images in 10 classes, with 6000 images per class. There are 50000 training images and 10000 test images” [5]. The classes are completely mutually exclusive. In other words, there is no overlap between the classes.

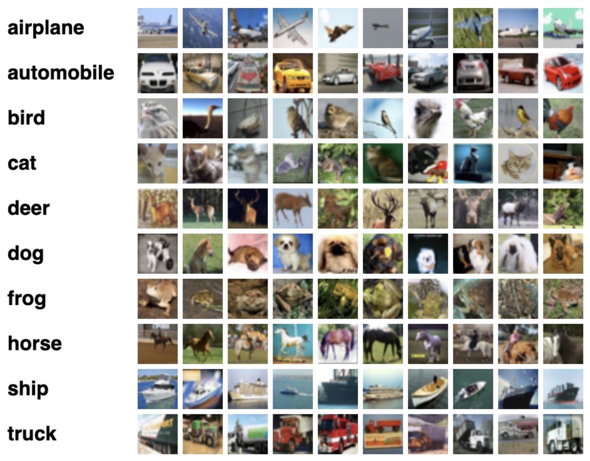


Figure 8.3: Samples from the CIFAR-10 Dataset. Source: [5]

In the following experiments, the 50000 images training set is referred to as the standard training set, the 10000 testing images as the test set. An extra set is needed in order to be used as the unlabelled set during the distillation process. Unkile SVHN, CIFAR-10 does not come with an extra set prepared. Luckily, CIFAR-10 is a subset of the 80 million tiny images dataset [1] part of which can be used as our extra dataset. In his papers [[7], [27]], Caruana uses the original CIFAR-10 dataset combined with 0.5M random images from the 80 million tiny images dataset. CIFAR-100 is another labelled subset of the 80 million tiny images dataset which is included in the Tensorflow (or Keras) API. This dataset is just like the CIFAR-10, except it has 100 classes containing 600 images each. The two datasets combined consists of 110.000 images (50.000 training images from CIFAR-10 & 60.000 images from CIFAR-100). This 110.000 images dataset acts as the extra or unlabelled dataset in the experiments that make use of the CIFAR-10 dataset.

8.2 Model Distillation

8.2.1 Teacher Models

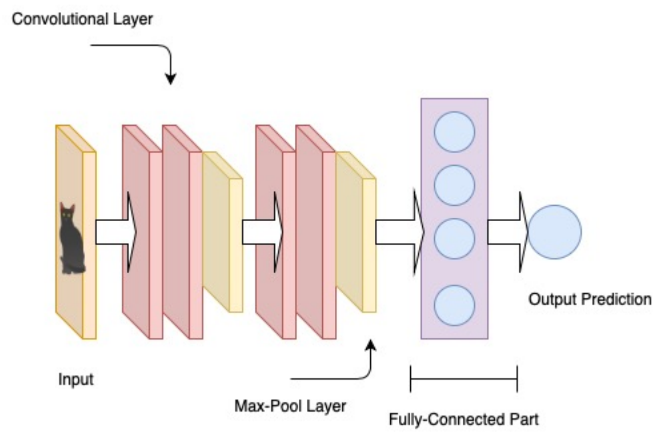


Figure 8.4: Teacher CNN's Architecture - SVHN

SVHN: Using the standard training set, a Convolutional Neural Network (CNN) with 4 convolutional layers is trained with the following architecture (CV-CV-MP-CV-CV-MP-FC).

The CNN is trained using three different optimizers: Adam (with lr=0.001), SGD (with lr=0.01) and RMSProp (with lr=0.001) using Cross-Entropy loss function and the results are presented in the following table.

Table 8.1: CNN Experiments - SVHN Dataset

Name	Optimizer	Accuracy Test Set	Accuracy Extra Set
CNN	Adam	**92.528%**	**96.281%**
CNN-SGD	Sgd	91.368%	95.658%
CNN-RMSProp	Rmsprop	91.403%	95.728%

Since the correct labels of the extra dataset are provided, the accuracy in the extra set can also be computed. It is worth mentioning that the trained CNNs in all three cases achieve a higher accuracy in the extra dataset. The extra dataset comes from the same data distribution but contains "somewhat less difficult samples" according to the SVHN documentation and this constitutes the reason for the increase in the model accuracy since the test set contains "harder" data points. Two points can be made here:

1. The CNN teacher model, although not extremely complex, can generalize well and hence work as the top performance network for this series of experiments in the SVHN dataset.
2. Being able to achieve a high score in these sets means we can expect high scores for the compressed models as well since distillation's target is to make shallow models mimic the function learned by the teacher model.

CIFAR-10: Using the standard training set, a Convolutional Neural Network with 6 convolutional layers is trained with the following architecture (CV-CV-MP-CV-CV-MP-CV-CV-MP-FC).

The CNN is trained using Adam optimizer (with lr=0.001) and Cross-Entropyy loss function.

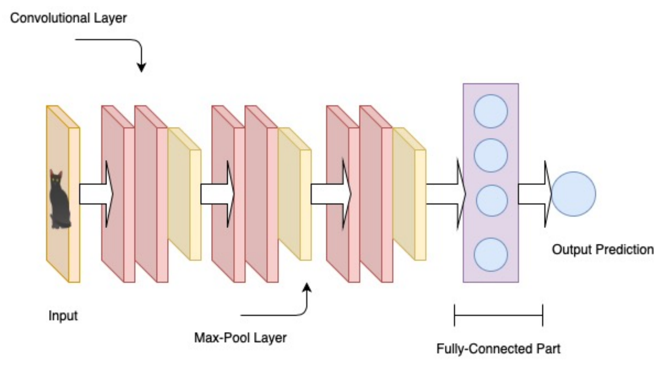


Figure 8.5: Teacher CNN's Architecture - CIFAR10

It is worth mentioning that since CIFAR-10 is a more difficult dataset, a deeper model is used (6 convolutional layers) in order to achieve a high accuracy of **82.560%** in the test set.

For the rest of the experiments, these two **CNN networks** are used as the state-of-the-art teacher models for each dataset trained with Adam optimizer.

8.2.2 Matching the Logits

This section contains the experimental results of distillating the aforementioned teacher models into three different shallow architectures consisting of either one fully-connected layer, one convolutional layer prior to the fully-connected part or two convolutional layers prior to the fully-connected part. Using the corresponding CNN teacher model for each dataset, ”Matching the Logits” distillation technique is implemented to train the shallow networks to mimic the function learned by the teacher models. First step of the experiments is to train the shallow networks with a similar number of parameters to the corresponding CNN teacher model using the standard training set of each dataset, its hard labels and Cross-Entropy loss function.

In the next phase of the experiment for both datasets, using the corresponding teacher CNN model, the teacher's logits of the two following sets are extracted:

1. Standard training set
2. Extra training set

The same series of experiments (described in the sub-sections below) are performed twice, once with the logits from the standard training set (denoted as Experiment A) and once with the logits from the extra training set (denoted as Experiment B). The models are trained using the same optimizer (Adam optimizer) as in stage one but with Mean Square Error (MSE) loss function. Through these experiments, several shallow network architectures are employed, each one of them with different number of parameters. All architectures are tested on their corresponding test sets.

Shallow Neural Net (SNN)

In this experiment, a standard shallow net (SNN) architecture consisting only of one hidden fully-connected layer is implemented, trained and distillated as described above.

CIFAR-10: A SNN model with only one fully connected layer proved to be inadequate to give a shallow model with a decently high accuracy since CIFAR-10 is a more complex problem compared to the SVHN digit classification.

SVHN: The SNN model with similar number of parameters to the teacher model (around 220k parameters) is trained on the standard training dataset and achieves an classification accuracy of **69.197%** in the test set which is around 20% less than the performance of the teacher CNN. The distillation results using the "Matching the Logits" technique for various SNNs with different number of parameters are displayed in the [8.2] table and the [8.6] graph.

Table 8.2: SNN Experiments - SVHN Dataset

Name	Number of Parameters	Number of hidden units	Accuracy (a)	Accuracy (b)
SNN-180	220k	180	75.073%	79.041%
SNN-512	850k	512	78.092%	82.191%
SNN-2000	3.5M	2000	81.000%	83.908%
SNN-6000	9.5M	6000	81.404%	84.223%
SNN-10000	16M	10000	79.982%	**85.057%**
SNN-20000	32.5M	20000	**81.635%**	84.314%
SNN-180	220k	180	**69.197%**	-
CNN	220k	-	**92.528%**	-

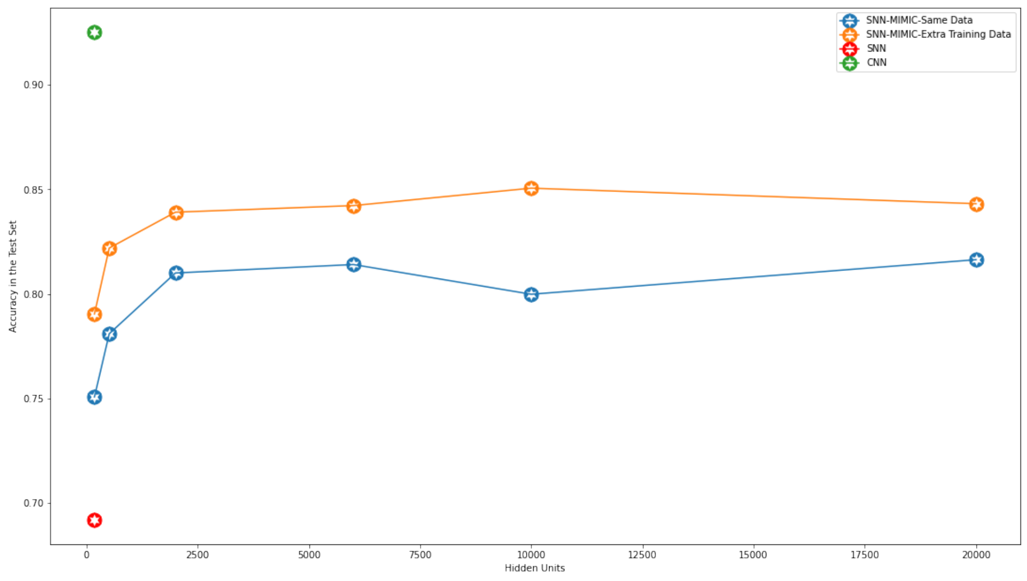


Figure 8.6: Graphical Illustration of SNN Accuracy Results - SVHN Dataset. The Green point represents the accuracy of the CNN Teacher model while the Red point the accuracy of the SNN with same number of parameters with the CNN and trained on the original data. Finally, the Orange and the Blue lines represent the series of distillation experiments using the extra and the standard data respectively.

Remarks from these experiments:

1. The SNN networks trained on the original standard training set with hard labels achieve a lower accuracy score than the ones (with same number of parameters) trained via distillation.
2. Between the networks that are trained via distillation, those SNNs that make use of the unlabelled dataset achieved a higher accuracy score.
3. The larger the width of the hidden layer (greater number of parameters / hidden units), the better the accuracy that the networks achieve.

Shallow Convolutional Neural Net (SCNN) - One Convolutional Layer

In this experiment, a shallow convolutional neural net (SCNN) architecture consisting of one convolutional layer (CV-MP-FC) prior to the fully connected part is implemented.

SVHN: The SCNN model with similar number of parameters to the teacher model (around 220k) is trained on the original dataset and achieves a classification accuracy of **80.612%** in the test set which is around 12% less than the performance of the teacher CNN. The distillation results using the "Matching the Logits" technique for various SCNNs with different number of parameters are displayed in the [8.3] table and the [8.7] graph.

Table 8.3: SCNN Experiments One Convolutional Layer - SVHN

Name	Number of Parameters	Number of channels	Accuracy (a)	Accuracy (b)
SCNN-h4	220k	4	82.049%	87.900%
SCNN-h16	850k	16	87.830%	89.409%
SCNN-h64	3.5M	64	87.061%	90.147%
SCNN-h172	9.5M	172	87.611%	89.717%
SCNN-h300	16M	300	87.481%	89.847%
SCNN-h600	32.5M	600	**87.711%**	**90.062%**
SCNN-SVHN	220k	-	**80.612%**	-
CNN	220k	-	**92.528%**	-

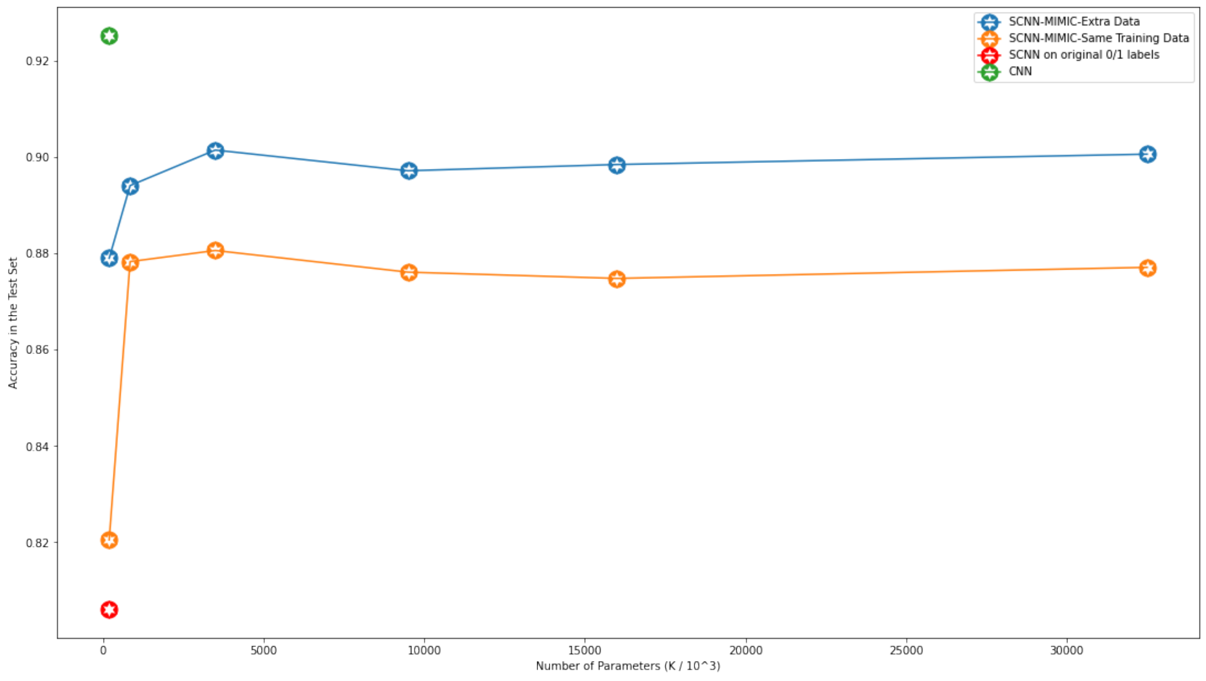


Figure 8.7: Graphical Illustration of SCNN Accuracy Results - SVHN Dataset. The Green point represents the accuracy of the CNN Teacher model while the Red point the accuracy of the SCNN with same number of parameters with the CNN and trained on the original data. Finally, the Blue and the Orange lines represent the series of distillation experiments using the extra and the original data respectively.

CIFAR-10: The SCNN model with similar number of parameters to the teacher model (around 550k) and trained on the original dataset, achieved an accuracy of **63.970%** in the test set which is around 20% less than the performance of the teacher CNN. The distillation results using the ”Matching the Logits” technique for various SCNNs with different number of parameters are displayed in the [8.4] table and the [8.8] graph.

Table 8.4: SCNN Experiments One Convolutional Layer - CIFAR10

Name	Number of Parameters	Number of channels	Accuracy (a)	Accuracy (b)
SCNN-h16	450k	16	64.300%	64.460%
SCNN-h64	1.8M	64	67.170%	67.310%
SCNN-h256	7M	256	**68.000%**	68.100%
SCNN-h384	12M	384	67.890%	**68.650%**
SCNN-CIFAR10	450k	-	**63.970%**	-
CNN	550k	-	**82.560%**	-

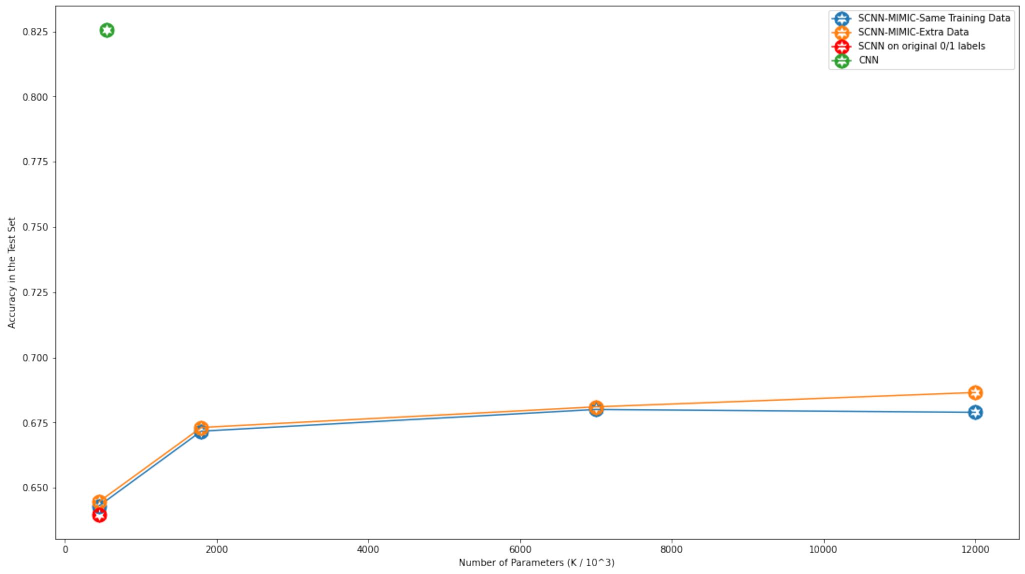


Figure 8.8: Graphical Illustration of SCNN Accuracy Results - CIFAR10 Dataset. The Green point represents the accuracy of the CNN Teacher model while the Red point the accuracy of the SCNN with same number of parameters with the CNN and trained on the original data. Finally, the Blue and the Orange lines represent the series of distillation experiments using the extra and the original data respectively.

Remarks from these experiments:

1. The SCNNs with one convolutional layer trained on the original standard training set with hard labels achieve a lower accuracy score than the ones (with same number of parameters) trained via distillation.
2. Between the networks that are trained via distillation, those SCNNs with one convolutional layer that make use of the unlabelled dataset achieve a higher accuracy score.
3. The larger the number of channels (greater number of parameters), the better the accuracy that the networks achieve.
4. The SCNNs with one convolutional layer trained via distillation achieve a higher accuracy than the SNNs in the previous experiment.

Shallow Convolutional Neural Net (SCNN) - Two Convolutional Layers

In this experiment, a shallow convolutional neural net (SCNN) architecture consisting of two convolutional layers (CV-MP-CV-MP-FC) prior to the fully connected part is implemented.

SVHN: The SCNN model model, with similar number of parameters to the teacher model (around 220k) and trained on the original dataset, achieved an accuracy of **82.916%** in the test set which is around 10% less than the performance of the teacher CNN. The results are displayed in the [8.5] table and the [8.9] graph.

Table 8.5: SCNN Experiments Two Convolutional Layers - SVHN Dataset

Name	Number of Parameters	Number of channels	Accuracy (a)	Accuracy (b)
SCN-32-32	220k	32-32	89.159%	**90.646%**
SCN-64-128	850k	64-128	90.101%	91.361%
SCN-64-512	3.5M	64-512	90.485%	91.583%
SCN-350-1024	9.5M	350-1024	90.496%	**91.637%**
SCNN	220k	-	**82.916%**	-
CNN	220k	-	**92.528%**	-

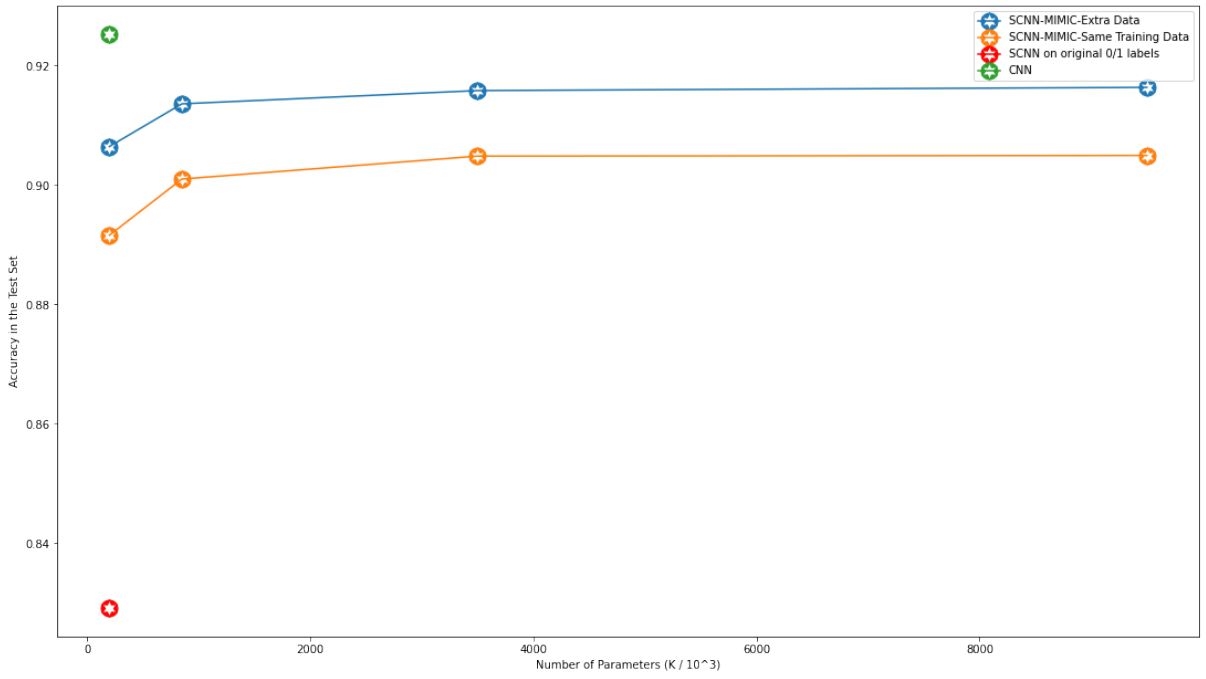


Figure 8.9: Graphical Illustration of SCNN Accuracy Results - SVHN Dataset. The Green point represents the accuracy of the CNN Teacher model while the Red point the accuracy of the SCNN with same number of parameters with the CNN and trained on the original data. Finally, the Blue and the Orange lines represent the series of distillation experiments using the extra and the original data respectively.

CIFAR10: The SCNN model model, with similar number of parameters to the teacher model (around 550k) and trained on the original dataset, achieved an accuracy of **65.890%** in the test set which is around 10% less than the performance of the teacher CNN. The results are displayed in the [8.6] table and the [8.10] graph.

Table 8.6: SCNN Experiments Two Convolutional Layers - CIFAR10

Name	Number of Parameters	Number of channels	Accuracy (a)	Accuracy (b)
SCN-64-128	650k	64-128	71.990%	73.450%
SCN-64-512	2.5M	64-512	74.790%	75.120%
SCN-128-1024	6M	128-1024	75.140%	75.460%
SCNN-256-2048	12M	256-2048	**75.190%**	**75.490%**
SCNN	650k	-	**65.890%**	-
CNN	550k	-	**82.560%**	-

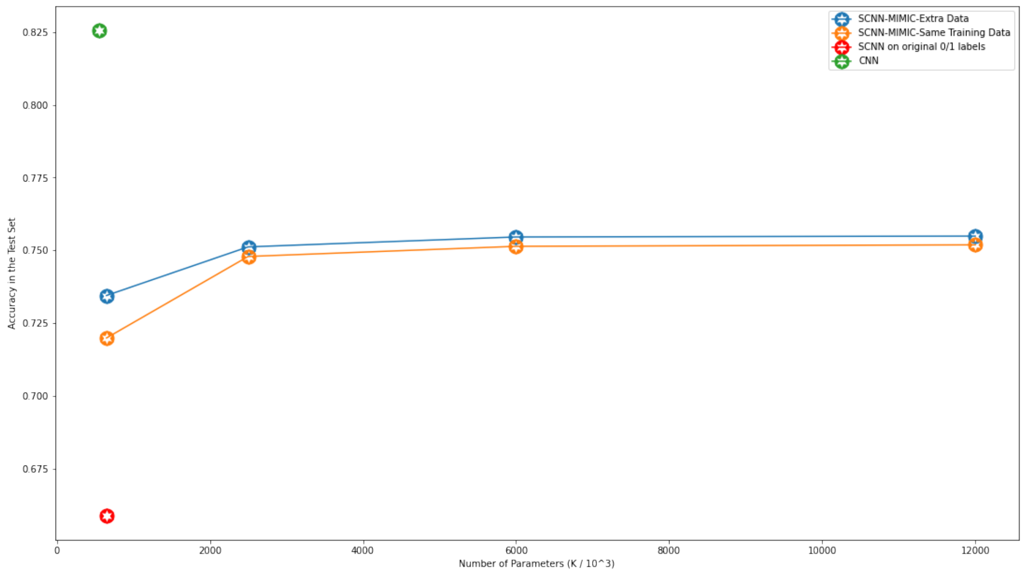


Figure 8.10: Graphical Illustration of SCNN Accuracy Results - CIFAR10 Dataset. The Green point represents the accuracy of the CNN Teacher model while the Red point the accuracy of the SCNN with same number of parameters with the CNN and trained on the original data. Finally, the Blue and the Orange lines represent the series of distillation experiments using the extra and the original data respectively.

Remarks from these experiments :

1. The SCNNs with two convolutional layers trained on the original standard training set with hard labels achieve a lower accuracy score than the ones (with same number of parameters) trained via distillation.
2. Between the networks that are trained via distillation, those SCNNs with two convolutional layers that make use of the unlabelled dataset achieved a higher accuracy score.
3. The larger the number of channels (greater number of parameters), the better the accuracy that the networks achieve.
4. The SCNNs with two convolutional layers trained via distillation achieve a higher accuracy than the SNNs or the SCNNs with one convolutional layer from the previous experiments. Finally, the best model has accuracy comparable to the CNN's score.

8.2.3 Ranging the Teacher's Temperature

In this section, "Ranging the Teacher's Temperature" distillation technique is implemented. This experiment examines the effect that ranging the temperature of the teacher model has on the classification accuracy of the student model. Since "Two Convolutional Layers" is the architecture with the best performance of all shallow networks tested so far, the student models in these experiments also employ only this approach. By ranging the teacher's temperature between 0 and 1, a output array with more conservative probabilities for the teacher model is produced while by raising the temperature between 1 and 5 a output array with softer probabilities for the teacher model is crafted. For the student model, the temperature is kept at 1 for the creation of its output probabilities array. During the training of the models, Cross-Entropy loss function is used to compare these two output arrays (from the teacher and student model).

Various SCNNs with different number of parameters and temperature values are used as the student models to distillate the teacher CNN models for each dataset. In the following subsections for each dataset, only the students with the best accuracy for each SCNN architecture are presented and compared with their equivalent best student model using the "Matching the Logits" distillation technique. The detailed distillation results for both datasets are described in Appendix B3.

SVHN

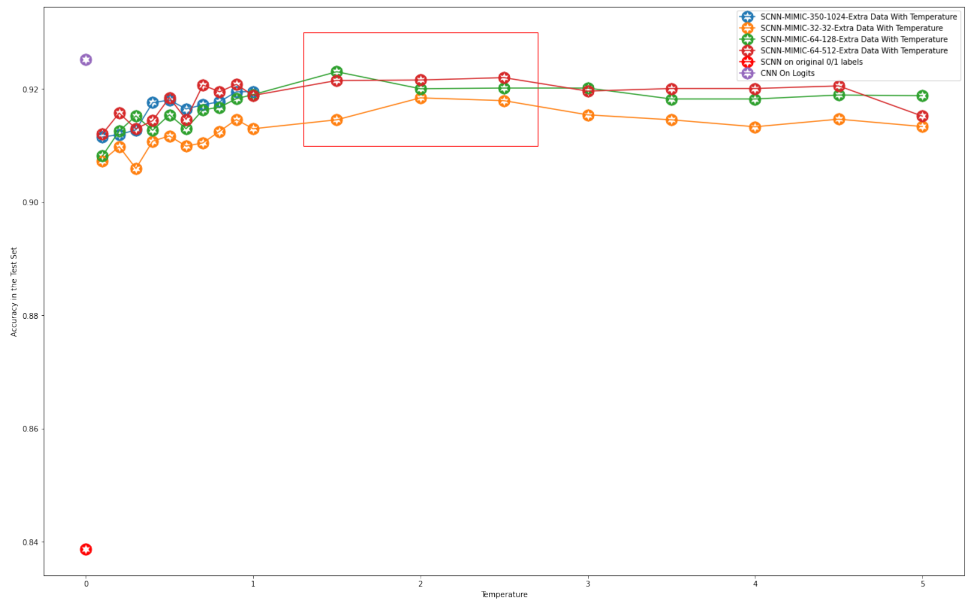


Figure 8.11: Graphical Illustration of SCNN Accuracy Results For Different Temperatures - SVHN Dataset. The Purple point represents the accuracy of the CNN Teacher model while the Red point the accuracy of the SCNN with same number of parameters with the CNN and trained on the original data. The colored lines represent the series of distillation experiments using various architectures and temperature values. Finally, the Red box represent the "Ideal Region": the temperature region where all student models achieve the best classification accuracy.

Table 8.7: SCNN Experiments Two Convolutional Layers - SVHN

Name	Number of Parameters	Temperature	Temp. Accuracy	Logits Accuracy
SCN-32-32	220k	2.0	91.845%	90.646%
SCN-64-128	850k	1.5	92.306%	91.361%
SCN-64-512	3.5M	2.5	92.202%	91.583%

CIFAR10

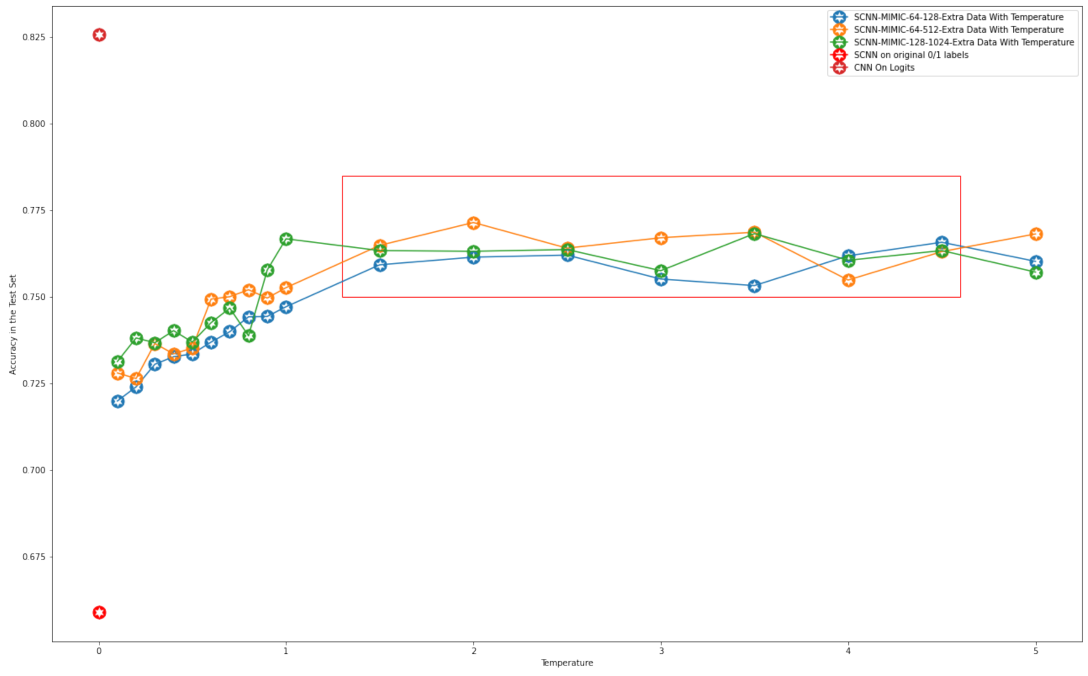


Figure 8.12: Graphical Illustration of SCNN Accuracy Results For Different Temperatures - CIFAR-10 Dataset. The Dark Red point represents the accuracy of the CNN Teacher model while the Light Red point the accuracy of the SCNN with same number of parameters with the CNN and trained on the original data. The colored lines represent the series of distillation experiments using various architectures and temperature values. Finally, the Red box represent the "Ideal Region": the temperature region where all student models achieve the best classification accuracy.

Table 8.8: SCNN Experiments Two Convolutional Layers - CIFAR10

Name	Number of Parameters	Temperature	Temp. Accuracy	Logits Accuracy
SCN-64-128	650k	4.5	76.570%	73.450%
SCN-64-512	2.5M	2.0	77.140%	75.120%
SCN-128-1024	6M	3.5	76.820%	75.460%

Remarks from these experiments:

1. The shallow student models trained via "Ranging the Teacher's Temperature" distillation technique achieve higher accuracy compared with their equivalent networks trained via "Matching the Logits" distillation technique.
2. Softer targets produce models with higher classification accuracy for both datasets.
3. Choosing the ideal temperature is necessary to produce the best accuracy possible student model. In other words, this distillation technique has an extra hyper-parameter (compared to the "Matching the Logits" distillation technique) that needs to be fine-tuned during training.

8.3 Parameter Reduction

Up until this point in the experimental part of the project, two state-of-the art teacher CNNs for both datasets were well-trained to achieve a high classification accuracy. Using the ”Matching the Logits” distillation technique, different shallow student models were trained with various widths and number of channels by matching the logits that both the teacher model and the training model produced. With this method, it was empirically demonstrated that shallow student neural networks need to be both deep and convolutional (since models with two convolutional layers achieve higher accuracy than models with one convolutional layer and in turn models with one convolutional layer achieve higher accuracy than models with one fully connected layer). Furthermore, by raising the temperature of the teacher model and choosing the optimal temperature value for each architecture and dataset, it was shown that all student models trained with ”Ranging the Teacher's Temperature” distillation technique can surpass in performance their equivalent models trained directly on the logits. It is obvious that, although the best distillated models have a comparable accuracy to their corresponding state-of-the art CNNs and they are shallower (two convolutional layers instead of four for SVHN dataset or six for the CIFAR10 dataset), there is a trade-off in the number of parameters which is significantly increased. The next step is to investigate whether the number of parameter can be decreased while maintaining the accuracy of the best student distillated models. For that reason, the ”Pruning Filters” method (described in Chapter 5) is implemented to the best accuracy student models for each dataset separately.

8.3.1 Best Student Model - SVHN

During the distillation experiments in the SVHN dataset described in the previous section, the best accuracy student model was the **SCNN-MIMIC-SVHN-h64-128_1.5** (see Appendix B3). For this student model, the L1-norm for its weights is computed and is displayed in the following two graphs:

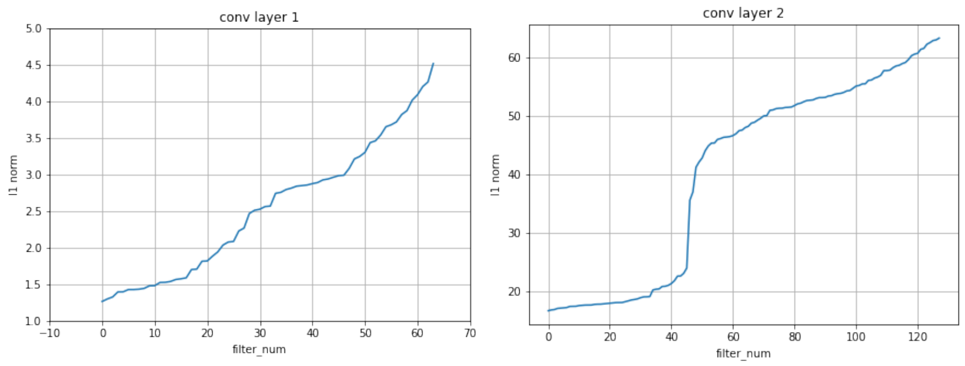


Figure 8.13: Computation of the L1-norm of the weights of the 1st and the 2nd convolutional layer in the best accuracy student model distillated for the SVHN dataset.

Based on the L1-norm the filters with value less than 25 in the 2nd convolutional layer (or equivalently 46/128 filters in the 2nd convolutional layer) and the filters with value less than 2.5 in the 1st convolutional layer (or equivalently 27/64 filters) are removed. Further decrease in the number of filters results in significant accuracy drop.

The resulted model **SCNN-MIMIC-SVHN-h64-128_1.5_new** has the same accuracy in both the validation and the test set but only 500k parameters, a reduction of around 30% in the number of parameters.

Table 8.9: Pruning Filters Experiments - SVHN Dataset

Name	Number of Parameters	Accuracy Test Set
CNN	220k	92.528%
SCNN-MIMIC-SVHN-h64-128_1.5	850k	92.306%
SCNN-MIMIC-SVHN-h64-128_1.5_new	500k	92.306%

8.3.2 Best Student Model - CIFAR10

During the distillation experiments using the CIFAR-10 dataset described in the previous section, the best accuracy student model was the **SCNN-MIMIC-CIFAR10-h64-512_2.0** (see Appendix B3). For this student model, the L1-norm for its weights is computed and is displayed in the following two graphs:

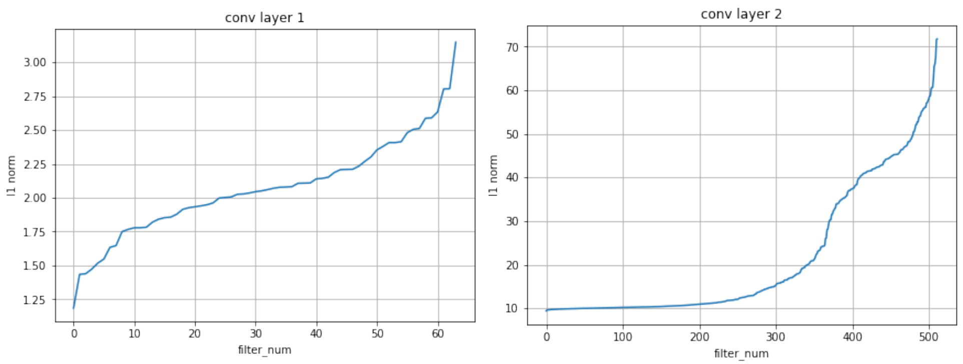


Figure 8.14: Computation of the L1-norm of the weights of the 1st and the 2nd convolutional layer in the best accuracy student model distillated for the CIFAR-10 dataset.

Based on the L1-norm the filters with value less than 20 in the 2nd convolutional layer (or equivalently 342/512 filters in the 2nd convolutional layer) and the filters with value less than 1.4 in the 1st convolutional layer (or equivalently 1/64 filters in the 1st convolutional layer) are removed. Further decrease in the number of filters results in significant accuracy drop.

Table 8.10: Pruning Filters Experiments - CIFAR10 Dataset

Name	Number of Parameters	Accuracy Test Set
CNN	550k	82.560%
SCNN-MIMIC-CIFAR10-h64-512_2.0	2.5M	77.140%
SCNN-MIMIC-CIFAR10-h64-512_2.0_new	880k	77.140%

The resulted model **SCNN-MIMIC-CIFAR10-h64-512_2.0_new** has the same accuracy in both the validation and the test set but only 880k parameters, a reduction of around 65% in the number of parameters.

8.4 Adversarial Robustness

In this part of the experiment, the shallow distillated models (filter pruned) with the best accuracy in both SVHN and CIFAR10 datasets are tested for their out-of-the-box robustness to adversarial non-targeted attacks. Their performance is compared to the performance of their corresponding teacher models under the same series of attacks.

Using Cleverhans library [40] adversarial examples from the SVHN and CIFAR-10 test sets are crafted using the methods (described in detail in Chapter 6) and the settings described in the table below.

Table 8.11: Adversarial Attacks - Experiments' Settings

Name	Formulation	Experimental Setup
Projected Gradient Descent [3]	$x_0^* = x$ $x_{t+1}^* \Leftarrow \Pi_{x+\mathcal{S}}(x_t^* + \alpha * sgn(\nabla_x J(x, f, \theta))$	$\varepsilon \subseteq [0.0, 0.2]$ iterations = 100 & $\alpha = \frac{\varepsilon}{2}$
Basic Iterative Method [6]	$x_0^* = x$ $x_{t+1}^* \Leftarrow Clip_{x,\varepsilon}(x_t^* + a * sgn(\nabla)_x J(x_t^*, f, \theta))$	$\varepsilon \subseteq [0.0, 0.2]$ iterations = 100 & $\alpha = \frac{\varepsilon}{2}$
Momentum Iterative Method [49]	$x_0^* = x$ $x^* \Leftarrow x + \alpha * sign(g_{t+1})$ $g_{t+1} \Leftarrow g_t + \frac{\nabla_x J(x^*, f, \theta))}{\|\|\nabla_x J(x^*, f, \theta))\|\|_1}$	$\varepsilon \subseteq [0.0, 0.2]$ iterations = 100 $\alpha = \frac{\varepsilon}{2}$

From these adversarial robustness experiments whose results are displayed in the following graphs, it is obvious that for all tested attacks, the teacher models and the student models show a similar behaviour (their accuracy after the attack is very close) for both datasets. Sometimes the student model actually performs slightly better than the teacher model for small values of ε, a fact that needs further investigation.

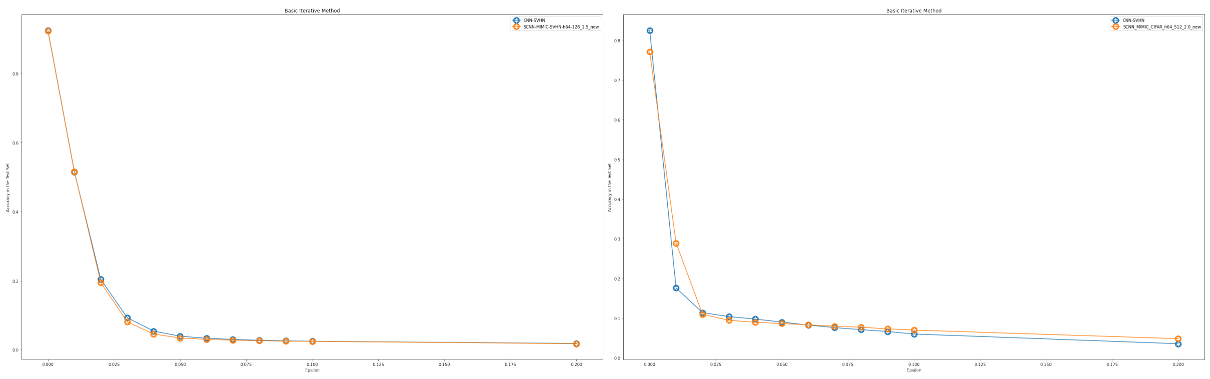


Figure 8.15: Basic Iterative Method: Blue Teacher - Orange Student (Left: SVHN Dataset & Right: CIFAR10 Dataset)

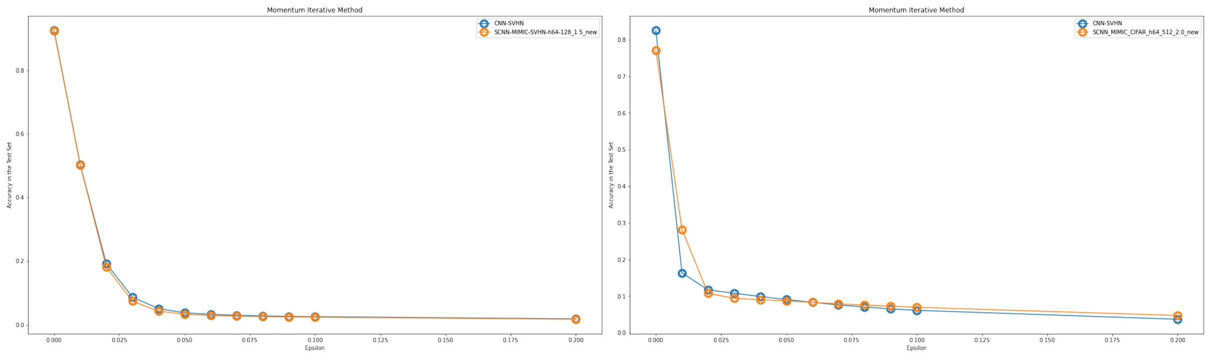


Figure 8.16: Momentum Iterative Method: Blue Teacher - Orange Student (Left: SVHN Dataset & Right: CIFAR10 Dataset)

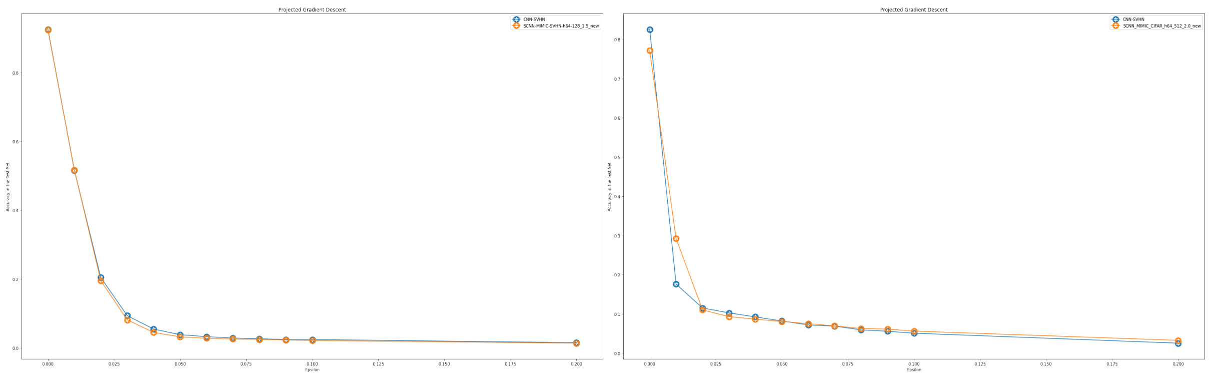


Figure 8.17: Projected Gradient Descent: Blue Teacher - Orange Student (Left: SVHN Dataset & Right: CIFAR10 Dataset)

Project Gradient Descent (PGD) is considered (for reasons discussed in Chapter 6) to be one of the hardest adversarial attacks for neural networks. This method is used in this report to further evaluate and more closely compare the teacher's and student's robustness. Small values of ε are chosen between 0 and 0.1 and step size 0.001. Choosing small values of ε is necessary since the networks are normally (not adversarially or robustly) trained. For each ε value, the PGD adversarial examples of the test set are calculated with maximum 100 iterations and $\alpha = \frac{\varepsilon}{2}$.

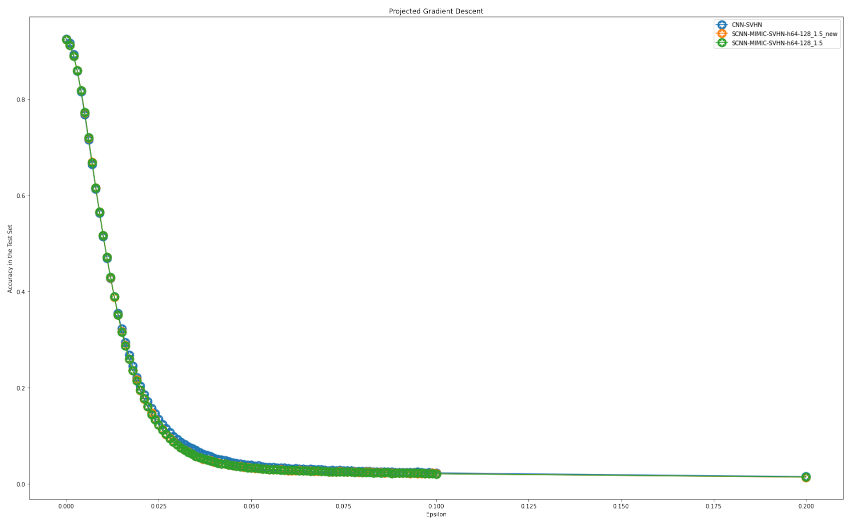


Figure 8.18: Projected Gradient Descent (PGD) Attacks in the SVHN dataset. Orange is the student model after filter pruning, Green is the student model prior to filter pruning and Blue is the teacher model.

In the graph above, the best student model prior filter pruning and the best student model after filter pruning are tested under the same adversarial PGD attacks. Empirically, it is proven that filter pruning results only in a parameter decrease rather than an actual benefit in terms of robustness since their performances are identical under the PGD attacks. Therefore, for the rest of the experiments only the robustness of filter pruned models is tested.

Using PGD attacks in the SVHN dataset, for small ε values [0.0, 0.014], hence small perturbations, the student model performs slightly better (maximum 0.5% accuracy difference between student and teacher model). ε in [0.0, 0.014] is actually the area of interest since both models have a classification accuracy greater than 35% in the test set.

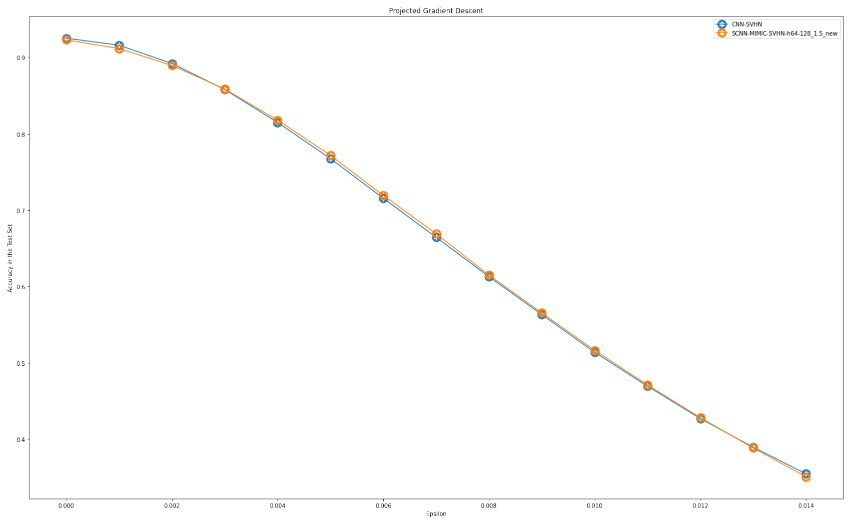


Figure 8.19: Projected Gradient Descent (PGD) Attacks in the SVHN dataset with ε at [0.0 - 0.014]. Orange is the student model after filter pruning and Blue is the teacher model.

For large ε values hence large perturbations, the student model performs slightly worse (maximum 0.9% accuracy difference between teacher and student model) than the teacher model. Nevertheless, in this region the accuracy drops dramatically in every step and reaches a plateau at around 5% classification accuracy.

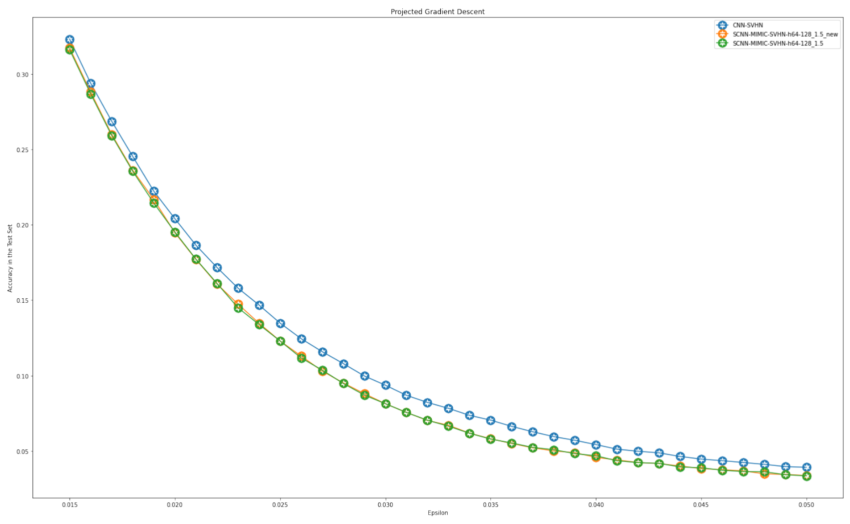


Figure 8.20: Projected Gradient Descent (PGD) Attacks in the SVHN dataset with ε at [0.015 - 0.050]. Orange is the student model after filter pruning, Green is the student model prior to filter pruning and Blue is the teacher model.

A similar behaviour was observed when repeating the PGD experiments with the student and teacher models trained on CIFAR10 dataset. Using PGD attacks in the CIFAR-10 dataset, for small ε values [0.0, 0.02], hence small perturbations, the student model performs better (maximum 20% accuracy difference between student and teacher model). For large ε values hence large perturbations, the student model performs slightly better (maximum 0.2% accuracy difference between teacher and student model) than the teacher model. Nevertheless, in this region the accuracy drops dramatically in every step and reaches a plateau at around 8% classification accuracy.

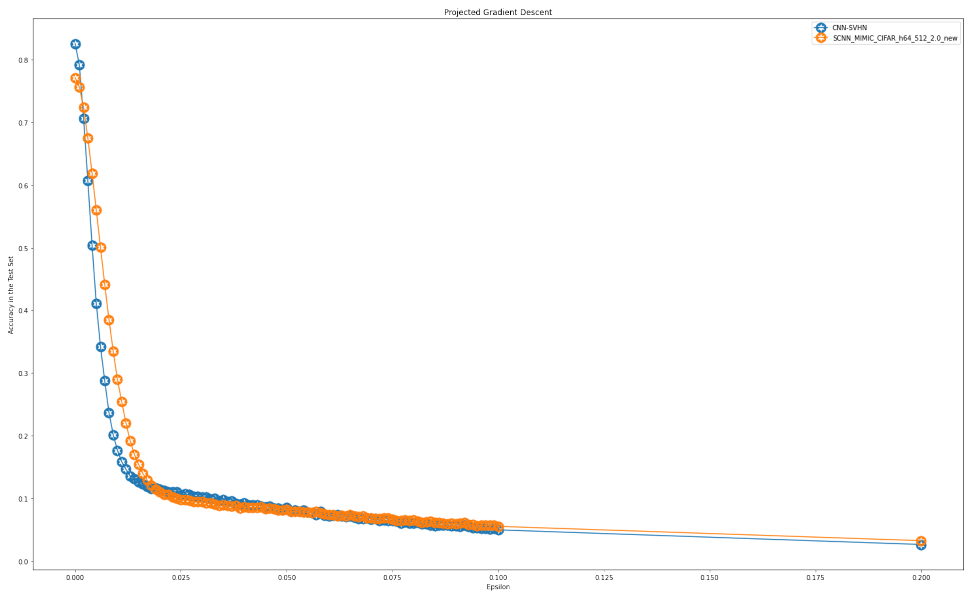


Figure 8.21: Projected Gradient Descent (PGD) Attacks in the CIFAR-10 dataset with ε at [0.0 - 0.1]. Orange is the student model after filter pruning and Blue is the teacher model.

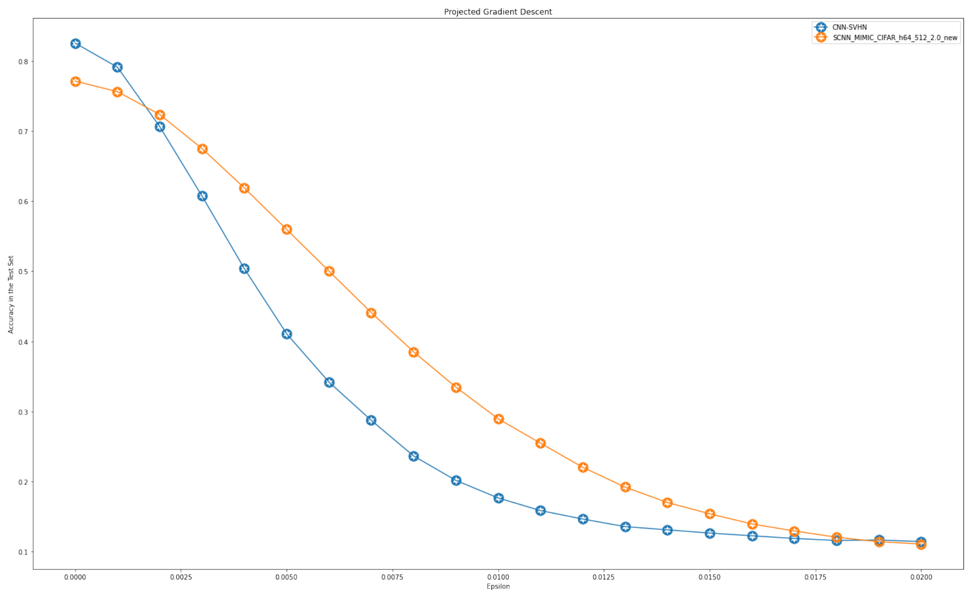


Figure 8.22: Projected Gradient Descent (PGD) Attacks in the CIFAR-10 dataset with ε at [0.0 - 0.02]. Orange is the student model after filter pruning and Blue is the teacher model.

8.5 Provably Robust Classifier

In this section, making use of the conclusions from the experiments in the previous sections, two models with the same architecture are trained (one normally and one robustly using the "Convex Outer Adversarial Polytope" technique) and are used as the teacher models for distillation experiments in the SVHN and CIFAR-10 datasets. Since the "Convex Outer Adversarial Polytope" technique (as described in detail in Chapter 7) is based only on ReLU networks, the previously used architectures of the CNN teacher models in the sections above cannot be used since they contain maximum pooling layers. Therefore, for each dataset a new neural network architecture of four convolutional layers followed by a fully connected part is defined. Instead of maximum pooling layers though, stride of two as well as padding of one are used to reduce the dimensions of the network's layers accordingly.

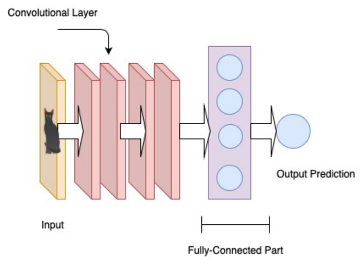


Figure 8.23: Teacher Classifier CNN's Architecture

8.5.1 Baseline & Robust Models

SVHN: For the SVHN dataset, a CNN of four convolutional layers prior to the fully-connected part (total 45k parameters) is used as the teacher's architecture. During the models' training, Adam optimizer with learning rate of 0.001, kernel size of 4 and Cross-Entropy loss function are used. The Robust model is trained in minibatches of size 20 (to avoid running out of the available GPU memory) and $\varepsilon = 0.01$ using the "Convex Outer Adversarial Polytope" technique. For both models, early stopping is also applied to avoid overfitting. The Baseline model achieves an accuracy of 85.960% and the Robust model has an accuracy of 74.220%.

Table 8.12: Accuracy Results, Robust Error Bounds and PGD Accuracy - SVHN

Name	Robust	ε	Accuracy Test Set	Robust Error Bound	PGD Accuracy at $\varepsilon = 0.01$
Baseline Model	No	-	85.960%	100%	32.268%
Robust Model	Yes	0.01	74.220%	46.7%	66.741%

CIFAR-10: For the CIFAR-10 dataset, a CNN of four convolutional layers prior to the fully-connected part (total 250k parameters) is used as the teacher's architecture. The models are trained using the same settings as in the SVHN experiment above. The Baseline model achieves an accuracy of 60.330% and the Robust model has an accuracy of 53.180%.

Table 8.13: Accuracy Results, Robust Error Bounds and PGD Accuracy - CIFAR-10

Name	Robust	ε	Accuracy Test Set	Robust Error Bound	PGD Accuracy at $\varepsilon = 0.01$
Baseline Model	No	-	60.330%	100%	34.680%
Robust Model	Yes	0.01	53.180%	68.7%	50.760%

8.5.2 Models' Distillation

In order to successfully distillate the neural networks (Baseline & Robust models), "Matching the Logits" distillation technique is used. Since the networks' architectures are now simpler than before, "Matching the Logits" distillation technique prove to be enough to create powerful shallow nets that even manage to surpass the accuracy of their deep teacher networks.

Based on the experiments in the previous sections, architectures with one or two convolutional layers produce student models that achieve high accuracy results (comparable to the teacher models). Therefore in this subsection, the extra sets from both datasets are used. Student models with two different architectures of one and two convolutional layers prior to the fully-connected part and various numbers of channels are implemented during the distillation process. The distillation results are detailedly described in the following tables and they confirm the observations made in the previous set of distillation experiments.

Table 8.14: SCNN Experiments One Convolutional Layer - SVHN

Training on Extra Training SVHN Dataset			
Name	# of Parameters / # of channels	Baseline Accuracy	Robust Accuracy
SCNN-MIMIC-h2	200k / 2	79.418%	73.882%
SCNN-MIMIC-h4	400k / 4	84.227%	74.635%
SCNN-MIMIC-h32	850k / 32	84.162%	74.577%
SCNN-MIMIC-h64	1.6M / 64	**85.053%**	**74.781%**
Training on Normal Training SVHN Dataset			
Name	# of Parameters / # of channels	Accuracy	
Baseline	45k / -	**85.960%**	
Robust	45k / -	**74.220%**	
SCNN-MIMIC-h2	200k / 2	**79.179%**	

Notation: # = number

Table 8.15: SCNN Experiments Two Convolutional Layers - SVHN

Training on Extra Training SVHN Dataset			
Name	# of Parameters / # of channels	Baseline Accuracy	Robust Accuracy
SCNN-MIM-4-8	50k / 4-8	84.669%	73.955%
SCNN-MIM-h16-32	200k / 16-32	86.359%	**74.589%**
SCNN-MIM-h16-64	400k / 16-64	**86.631%**	74.474%
SCNN-MIM-h32-128	850k / 32-128	86.628%	74.569%
Training on Normal Training SVHN Dataset			
Name	# of Parameters / # of channels	Accuracy	
Baseline	45k / -	**85.960%**	
Robust	45k / -	**74.220%**	
SCNN-MIM-h4-8	50k / 4-8	**84.869%**	

Notation: # = number

Table 8.16: SCNN Experiments Two Convolutional Layers - CIFAR10

Training on Extra Training CIFAR-10 Dataset			
Name	# of Parameters / # of channels	Baseline Accuracy	Robust Accuracy
SCNN-MIM-4-8	260k / 4-8	56.220%	50.710%
SCNN-MIM-h16-32	1M / 16-32	**59.250%**	52.430%
SCNN-MIM-h16-64	2M / 16-64	58.920%	**52.880%**
Training on Normal Training CIFAR-10 Dataset			
Name	# of Parameters / # of channels	Accuracy	
Baseline	250k / -	**60.330%**	
Robust	250k / -	**53.180%**	
SCNN-MIM-h4-8	260k / 4-8	**56.550%**	

Notation: # = number

8.5.3 Prune Filtering of the Best Student Models

In both datasets, for each one of the teacher networks (Baseline & Robust models) as well as their corresponding best accuracy student models, the L1-norm of the filter weights is computed. Based on the L1-norm, the filters with small values are removed without damaging the network's performance (as described in Chapter 5). Unlike, in the previous experiments where the accuracy of the pruned models remained unchanged, the accuracy of some of the models are slightly increased as it is described in the two following tables:

Table 8.17: Summary of Best (Baseline & Robust) Models - SVHN

Name	# of Conv. Layers / # of Params.	Accuracy Test Set
Baseline	- / 45k	85.960%
SCNN-MIM-h16-64	16-64 / 400k	86.631%
SCNN-MIM-h16-64-new (pruned)	16-64 / 380k	86.631%
Robust	- / 45k	74.220%
SCNN-MIM-h16-32	16-32 / 200k	74.589%
SCNN-MIM-h16-32-new (pruned)	16-32 / 160k	74.589%

Notation: # = number

Table 8.18: Summary of Best (Baseline & Robust) Models - CIFAR-10

Name	# of Conv. Layers / # of Params.	Accuracy Test Set
Baseline	- / 250k	60.330%
SCNN-MIM-h16-32	16-32 / 1M	59.250%
SCNN-MIM-h16-32-new (pruned)	16-32 / 450k	59.250%
Robust	- / 250k	53.180%
SCNN-MIM-h16-64	16-64 / 2M	52.880%
SCNN-MIM-h16-64-new (pruned)	16-64 / 550k	52.880%

Notation: # = number

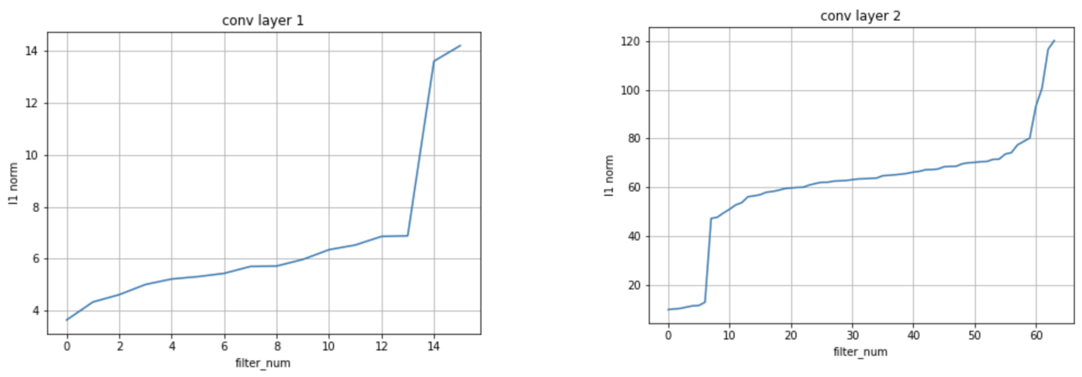


Figure 8.24: Computation of the L1-norm of the weights of the 1st and the 2nd convolutional layer in the best accuracy Baseline student model distillated for the SVHN dataset.

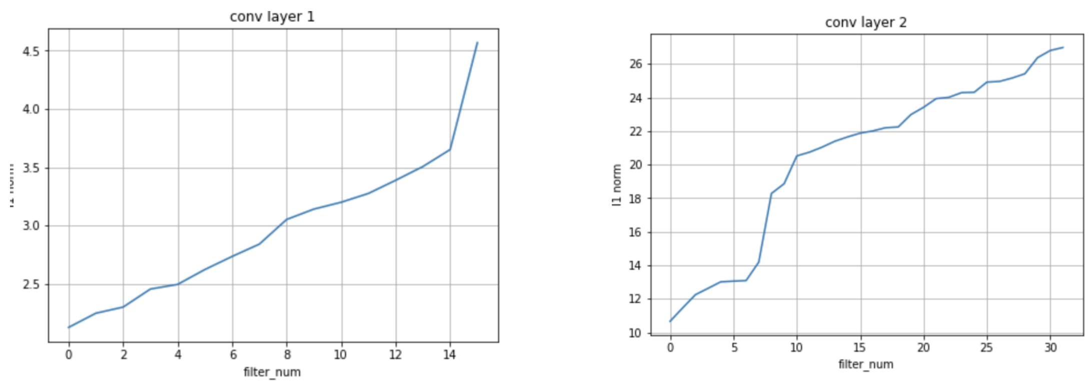


Figure 8.25: Computation of the L1-norm of the weights of the 1st and the 2nd convolutional layer in the best accuracy Robust student model distillated for the SVHN dataset.

8.5.4 Models' Progressive Gradient Descent (PGD) Robustness

As in the previous sections, Progressive Gradient Descent (PGD) method is used to evaluate and compare the teacher's with the student's adversarial robustness. Small values of ε are chosen between 0.0 and 0.1 with step size of 0.001. For each ε value, the PGD adversarial examples of the test set are calculated with maximum 100 iterations and $\alpha = \frac{\varepsilon}{2}$ for the Baseline and Robust models for both datasets.

From the results (presented in the graphs below), the following observations can be drawn:

1) As expected, the Robust model is far more robust to PGD attacks compared to the Baseline model in both datasets.

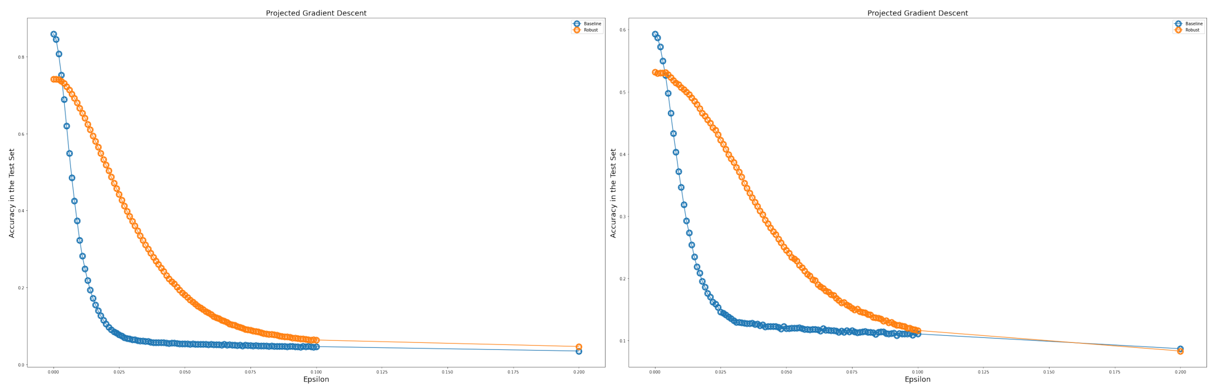


Figure 8.26: Accuracy in the test set under PGD Attacks with different ε values. (Orange Line: Robust Model and Blue Line: Baseline) - Left: SVHN dataset and Right: CIFAR-10 dataset.

2) The distillated models follow the same robustness order as their corresponding teacher models above. The accuracy difference between the models though is significantly smaller.

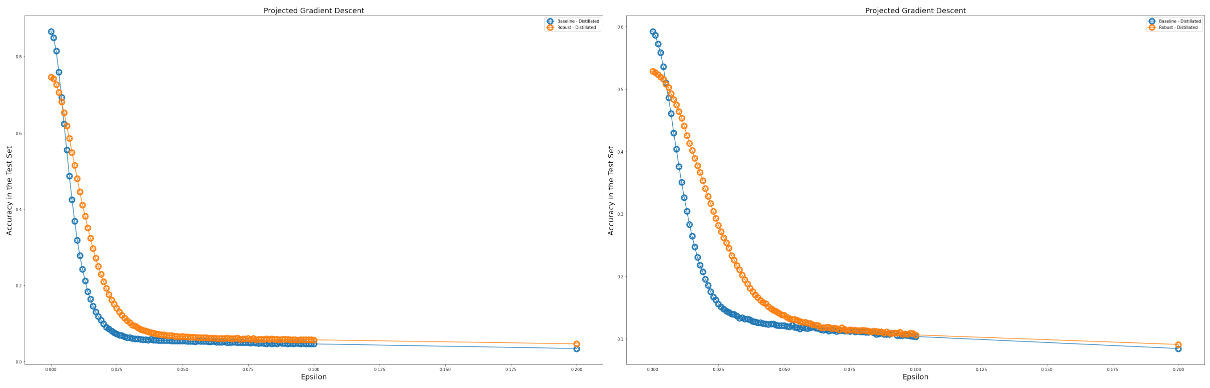


Figure 8.27: Accuracy in the test set under PGD Attacks with different ε values. (Orange Line: Robust Model (Distillated) and Blue Line: Baseline (Distillated)) - Left: SVHN dataset and Right: CIFAR-10 dataset.

3) As in the experiments in the previous sections, the distillated models of the Baseline models prove to be more robust than their corresponding teacher models for small values of ε.

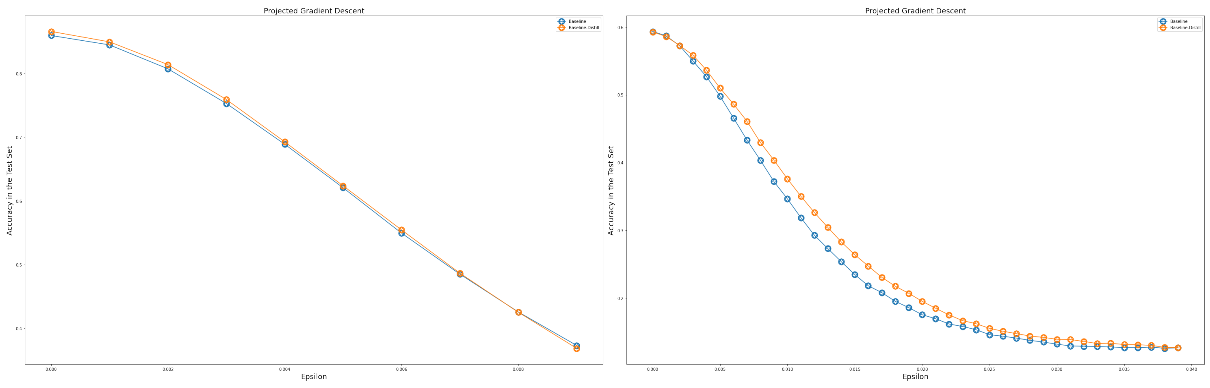


Figure 8.28: Accuracy in the test set under PGD Attacks with different ε values. (Orange Line: Robust Model (Distillated) and Blue Line: Baseline (Distillated)) - Left: SVHN dataset and Right: CIFAR-10 dataset.

4) None of the distillated models (not even the student model that uses the Robust model as its teacher) can surpass the adversarial robustness performance of the Robust model trained with the "Convex Outer Adversarial Polytope" technique.

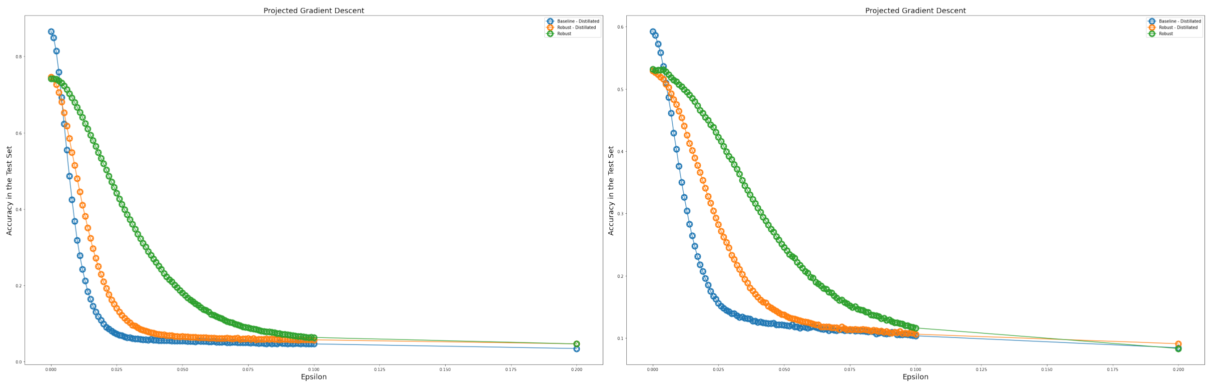


Figure 8.29: Accuracy in the test set under PGD Attacks with different ε values. (Orange Line: Robust (Distillated), Blue Line: Baseline (Distillated) and Green Line: Robust Model) - Left: SVHN dataset and Right: CIFAR-10 dataset.

8.6 Certification Area

This section's research goal is to investigate the effect that distillation has on the certification area of a neural network. According to the [47], "for each example x on the network with learnt function $f_\theta(x)$, the largest value of ε can be computed for which a certificate of robustness exists such that the output of $f_\theta(x)$ provably cannot be flipped within this ε ball". The certification areas of the Baseline and Robust models as well as their respective distillated networks for both datasets are calculated and compared.

8.6.1 Certification Area for Baseline Models

Using Newton's method as described in [7.14], the certification areas for both the Baseline model and the Baseline Distillated model are calculated and displayed on the graphs below:

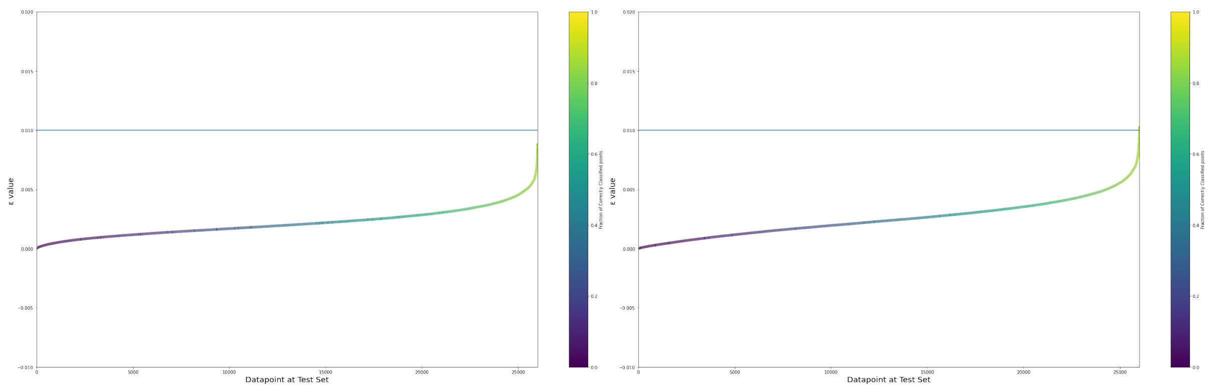


Figure 8.30: Maximum ε distances to the decision boundary of each data point of the test set in increasing ε order for the Baseline and the Best Accuracy Baseline Distillated Models. The blue line denotes the ε value that the Robust model has been trained to be robust on (in our experiments $\varepsilon = 0.01$). Left: Baseline Model and Right: Best Accuracy Baseline Distillated Model - SVHN Dataset

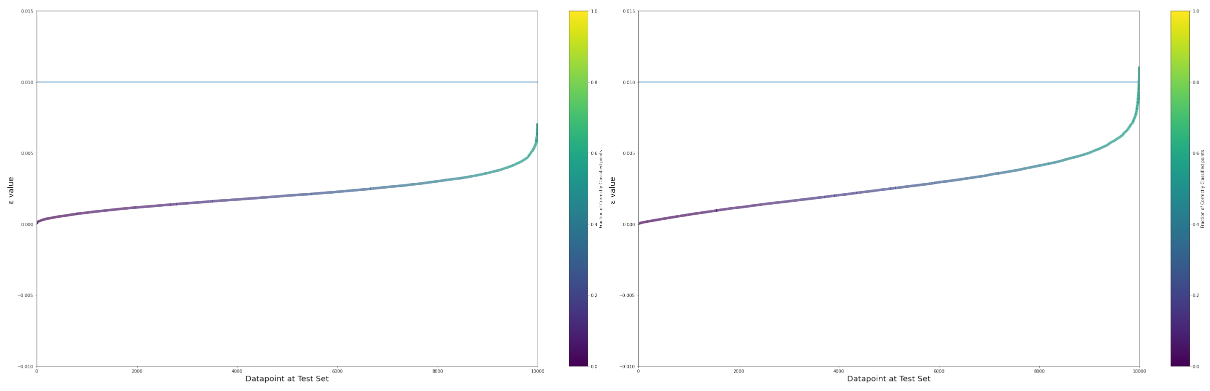


Figure 8.31: Maximum ε distances to the decision boundary of each data point of the test set in increasing ε order for the Baseline and the Best Accuracy Baseline Distillated Models. The blue line denotes the ε value that the Robust model has been trained to be robust on (in our experiments $\varepsilon = 0.01$). Left: Baseline Model and Right: Best Accuracy Baseline Distillated Model - CIFAR-10 Dataset

8.6.2 Certification Area for Robust Model

Using Newton's method as described in [7.14], the certification areas for both the Robust model and the Robust Distillated model are calculated and displayed on the graphs below:

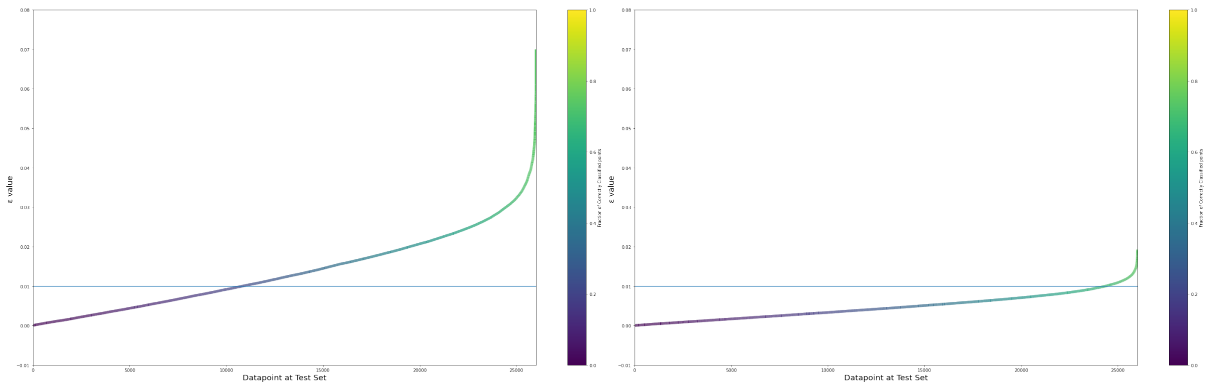


Figure 8.32: Maximum ε distances to the decision boundary of each data point of the test set in increasing ε order for the Robust and Best Accuracy Robust Distillated models. The blue line denotes the ε value that the Robust model has been trained to be robust on (in our experiments $\varepsilon = 0.01$). Left: Robust Model and Right: Best Accuracy Robust Distillated Model - SVHN Dataset

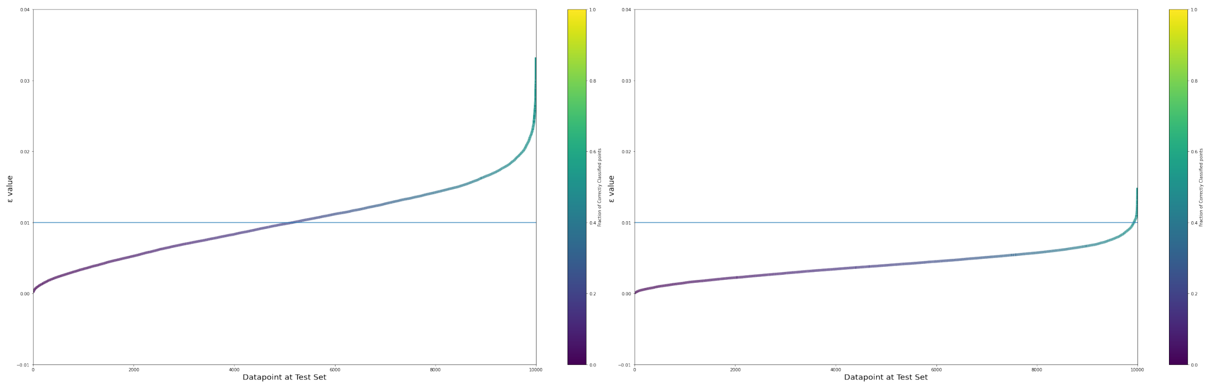


Figure 8.33: Maximum ε distances to the decision boundary of each data point of the test set in increasing ε order for the Robust and Best Accuracy Robust Distillated models. The blue line denotes the ε value that the Robust model has been trained to be robust on (in our experiments $\varepsilon = 0.01$). Left: Robust Model and Right: Best Accuracy Robust Distillated Model - CIFAR-10 Dataset

Remarks from these experiments:

1. The certification area of the Baseline Distillated model is larger compared to the Baseline network in both datasets. This observation shows that another advantage of the distillation process of normally trained models is an increase in their certification area.
2. On the contrary, the certification area of the Robust models is much larger compared to the Robust Distillated models.

8.6.3 Certification Area & Number of Parameters

In this subsection, the relationship between the number of parameters of the student model and the certification area is investigated. For each distillated model (produced by either the Baseline or Robust networks in both datasets) the certification area is calculated and the average ε value is computed (for completeness the graphical results are included in Appendix C). From these experiments, although there is a general trend indicating that smaller networks have larger certification areas, there is no conclusive evidence about the existence of an absolute direct relationship between the number of parameters of the student model and the certification area, since for the SVHN dataset, smaller networks have larger certification areas while for the CIFAR-10, the exact opposite holds. It is worth mentioning though that for both datasets all distillated Baseline models have a larger certification area compared to the Baseline model while all distillated Robust models have a smaller certification area compared to the Robust model. This empirical evidence further backs the claim that distillation does increase the adversarial robustness of the student models compared to their teacher networks if the models are normally (not robustly) trained.

Table 8.19: Average ε Certification Area - SVHN Dataset

Name	Trained Method	Number of Parameters	Average ε
BaselineModel	Baseline	45k	0.00215
SCNN-MIMIC-h4-8	Baseline	50k	0.00313
SCNN-MIMIC-h16-32	Baseline	200k	0.00308
SCNN-MIMIC-h16-64	Baseline	400k	0.00254
RobustModel	Robust	45k	0.01367
SCNN-MIMIC-h4-8	Robust	50k	0.00646
SCNN-MIMIC-h16-32	Robust	200k	0.00474
SCNN-MIMIC-h16-64	Robust	400k	0.00343

Table 8.20: Average ε Certification Area - CIFAR-10 Dataset

Name	Trained Method	Number of Parameters	Average ε
BaselineModel	Baseline	250k	0.00211
SCNN-MIMIC-h4-8	Baseline	260k	0.00259
SCNN-MIMIC-h16-32	Baseline	1M	0.00270
SCNN-MIMIC-h16-64	Baseline	2M	0.00274
RobustModel	Robust	250k	0.00995
SCNN-MIMIC-h4-8	Robust	260k	0.00442
SCNN-MIMIC-h16-32	Robust	1M	0.00444
SCNN-MIMIC-h16-64	Robust	2M	0.00408

8.6.4 Improve Robustness & Certification Area of the Distillated Models

Ideally, we would like the distillated models to have the certification area and the robustness of the Robust model. In the previous section, it was proven that using "Matching the Logits" distillation technique, the student models of the Robust network did not inherit these properties. In this subsections, three different ways to improve the robustness and certification area of the Distillated Robust model are outlined:

1. Distillation using Ranging the Teacher's Temperature Technique.

2. Re-training Robustly the best accuracy Distillated Robust model.
3. Distillation using Adversarial Distillation Technique.

Ranging the Teacher's Temperature Technique

In these experiments, the distillation procedure follows the "Ranging the Teacher's Temperature" Technique. By ranging the teacher's temperature between 0 and 1, a output array with more conservative probabilities for the teacher model is produced while by raising the temperature between 1 and 5 a output array with softer probabilities for the teacher model is produced. For the student model, the temperature is kept at 1 for the creation of its output probabilities array. During the training of the models, Cross-entropy loss function is used to compare these two output arrays (from the teacher and student model) and and Adam optimizer with learning rate of 0.001.

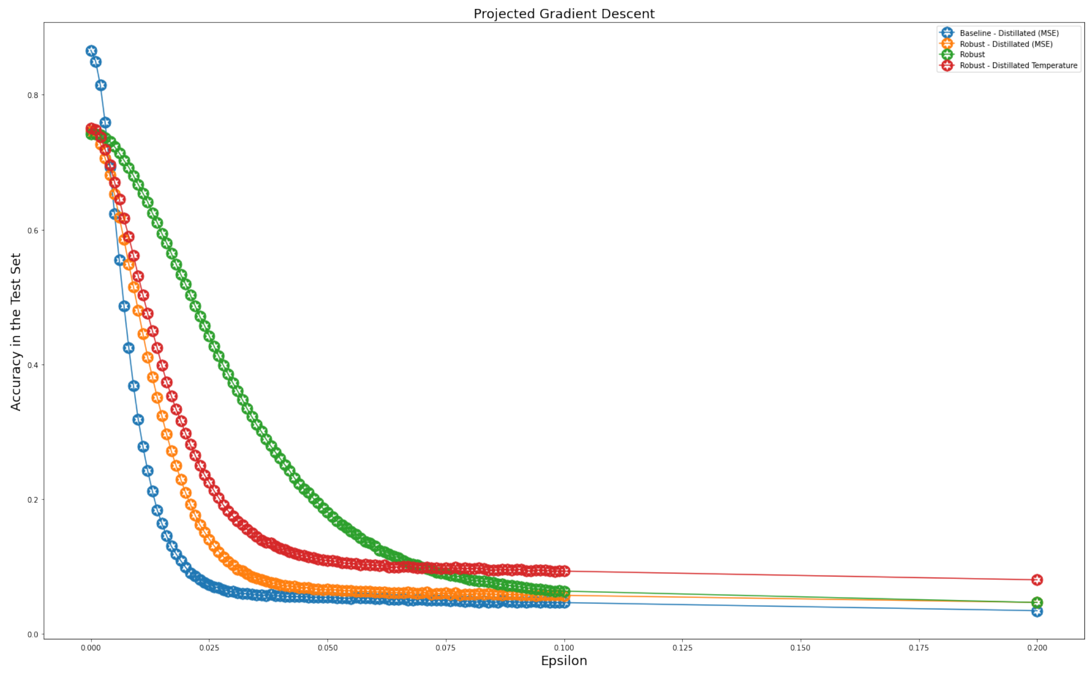


Figure 8.34: Accuracy in the test set under PGD Attacks with different ε values. (Orange Line: Robust (Distillated - Logits), Blue Line: Baseline (Distillated - Logits), Green Line: Robust Model and Red Line: Robust (Distillated - Temperature)) - SVHN Dataset

By ranging the temperature in the ideal range for this model, models with higher accuracy than using "Matching the Logits" technique can be produced (the detailed results are presented in the Appendix B2). The best model from ranging the teacher's temperature is the **SCNN-MIMIC-h32-128_0.1** model with accuracy of **74.846%**. The same PGD attacks on this model are performed and the certification area using Newton's method is calculated. Although, the adversarial robustness is slightly increased (results in the graph above), the certification area remains almost the same as in the distillated model using "Matching the Logits" distillation technique.

Re-training Robustly

From both distillated models either using ”Matching the Logits” technique or ”Ranging the Teacher’s Temperature” technique, it can be concluded that a model **does not inherit** the robustness nor the certification area of a robustly trained teacher network. By retraining robustly using the ”Convex Outer Adversarial Polytope” technique [47] for $\varepsilon = 0.01$ and for only 3 epochs:

1) The lost robustness is re-gained (in fact the re-trained model is more robust than the original robust model in some cases)

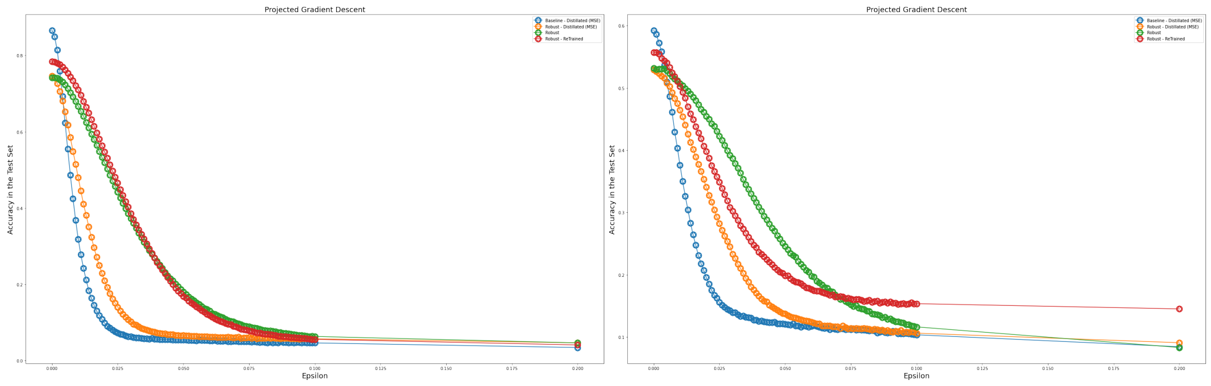


Figure 8.35: Accuracy in the test set under PGD Attacks with different ε values. (Orange Line: Robust (Distillated), Blue Line: Baseline (Distillated), Green Line: Robust Model and Red Line: Robust Model (Distillated Retrained Robustly)) Left: SVHN dataset and Right: CIFAR-10 dataset.

2) The clean accuracy of the model is increased compared to the teacher Robust model.

Table 8.21: Accuracy Results, Robust Error Bounds and PGD Accuracy - SVHN Dataset

Name	Robustly Trained	ε	Accuracy Test Set	PGD Accuracy at $\varepsilon = 0.01$
Baseline Model	No	-	85.960%	32.268%
Robust Model	Yes	0.01	74.220%	66.741%
Robust Model (MSE Distillated)	No	-	74.589%	47.984%
Robust Model (Retrained Distillated)	Yes	0.01	78.423%	70.974%

Table 8.22: Accuracy Results, Robust Error Bounds and PGD Accuracy - CIFAR-10 Dataset

Name	Robustly Trained	ε	Accuracy Test Set	PGD Accuracy at $\varepsilon = 0.01$
Baseline Model	No	-	60.330%	34.680%
Robust Model	Yes	0.01	53.180%	50.760%
Robust Model (MSE Distillated)	No	-	52.880%	46.430%
Robust Model (Retrained Distillated)	Yes	0.01	56.660%	50.290%

3) The certification area is increased as well since the distillated model is shallower and robustly re-trained.

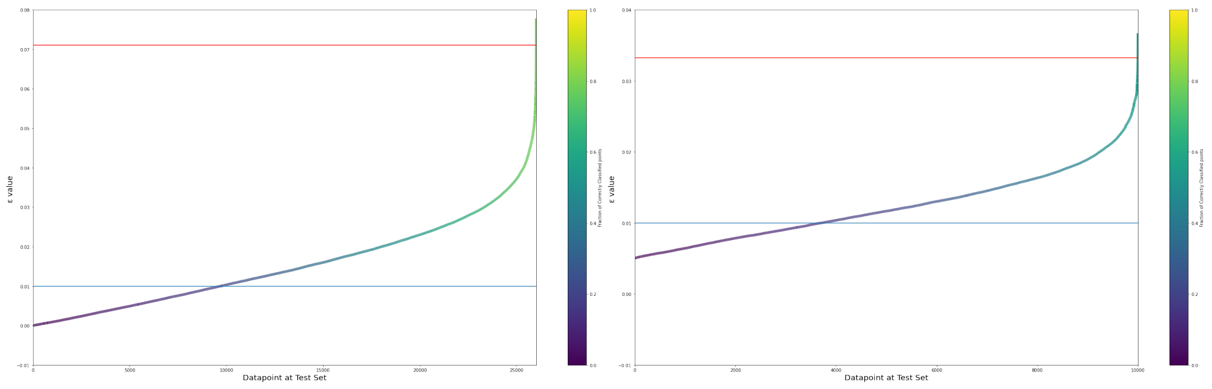


Figure 8.36: Maximum ε distances to the decision boundary of each data point of the test set in increasing ε order for the Robust Distillated Re-Trained models on the two datasets. The blue line denotes the ε value that the Robust model has been trained to be robust on (in our experiments $\varepsilon = 0.01$) and the red line denotes the maximum ε value of the respective teacher Robust Model. Left: SVHN Dataset and Right: CIFAR-10 Dataset.

Re-training Robustly vs. Robustly Trained Shallow Net

The question we try to answer here is if distillation is an important part after all. For this section, the shallow network with the same architecture as the Robust Model (Retrained Distillated) model is trained robustly until a similar accuracy is achieved in the test set for a valid and fair comparison and calculated its certification area. From the results, it can be shown that the Robust Model (Retrained Distillated) achieves a much larger certification area compared to Robust Shallow Network indicating that distillation constitutes a vital part to the increase of a model's certification area.

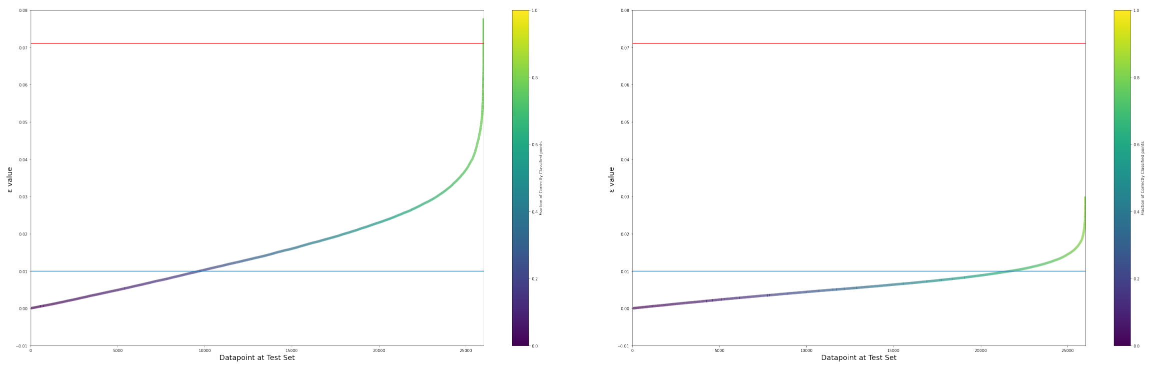


Figure 8.37: Maximum ε distances to the decision boundary of each data point of the test set in increasing ε order. The blue line denotes the ε value that the Robust model has been trained to be robust on (in our experiments $\varepsilon = 0.01$) and the red line denotes the maximum ε value of the respective teacher Robust Model. Left: Robust Distillated Re-Trained model (Accuracy: 78.423%) and Right: Straight Robustly Trained Shallow Network (Accuracy: 80.015%) - SVHN Dataset.

Table 8.23: Average ε Certification Area - SVHN Dataset

Name	Trained Method	Average ε
RobustModel	Robust	0.01367
Shallow SCNN	Robust	0.00612
ReTrained SCNN	Robust	0.01534

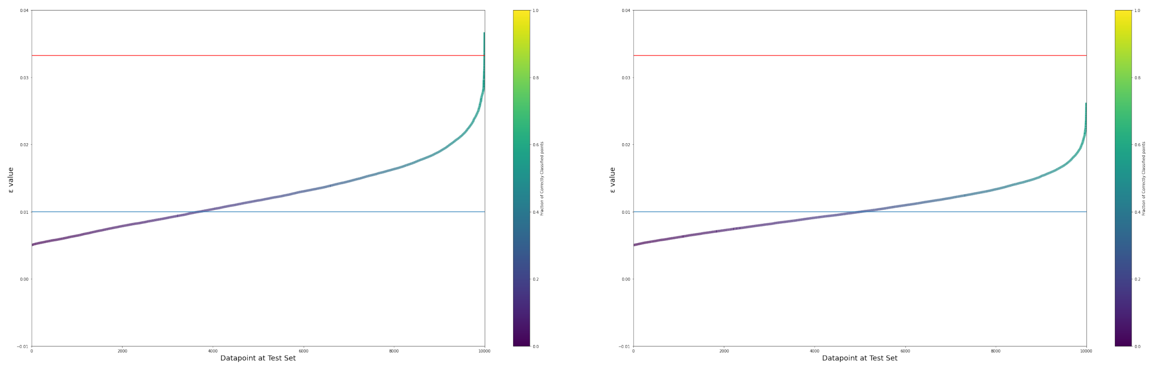


Figure 8.38: Maximum ε distances to the decision boundary of each data point of the test set in increasing ε order. The blue line denotes the ε value that the Robust model has been trained to be robust on (in our experiments $\varepsilon = 0.01$) and the red line denotes the maximum ε value of the respective teacher Robust Model. Left: Robust Distillated Re-Trained model (Accuracy: 55.670%) and Right: Straight Robustly Trained Shallow Network (Accuracy: 56.660%) - CIFAR-10 Dataset.

Table 8.24: Average ε Certification Area - CIFAR-10 Dataset

Name	Trained Method	Average ε
RobustModel	Robust	0.00995
Shallow SCNN	Robust	0.01044
ReTrained SCNN	Robust	0.01232

Adversarial Distillation

As described in Chapter 4, Adversarial Distillation works by matching the clean logits of the teacher model for the training data points with the logits that the student model generates during training from their equivalent adversarial examples. In order to adversarially distillate the teacher model, the adversarial examples are crafted using the architectures of the different student models. More specifically, in every epoch the given state of the student model is used to generate the adversarial examples (on-the-fly).

Table 8.25: Adversarial Distillation - Student On-The-Fly - Accuracy Results

Name	Channels	Accuracy Test Set
SCNN-MIMIC-h4-8	4-8	74.692%
SCNN-MIMIC-h16-32	16-32	**74.720%**
SCNN-MIMIC-h16-64	16-64	74.149%

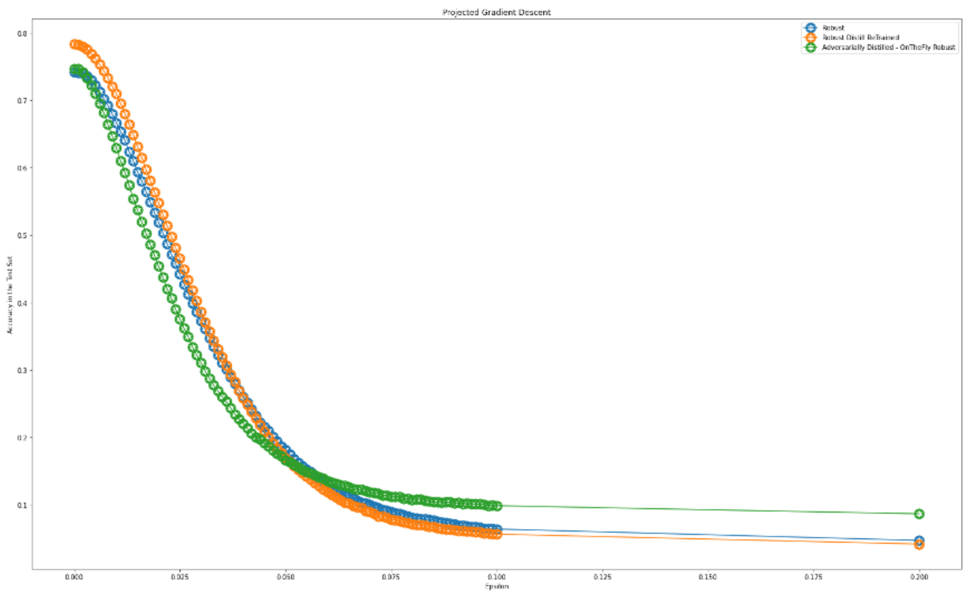


Figure 8.39: Accuracy in the test set under PGD Attacks with different ε values. (Orange Line: Robust (Distillated), Blue Line: Robust Model and Green Line: Adversarially Distillated Model) - SVHN dataset

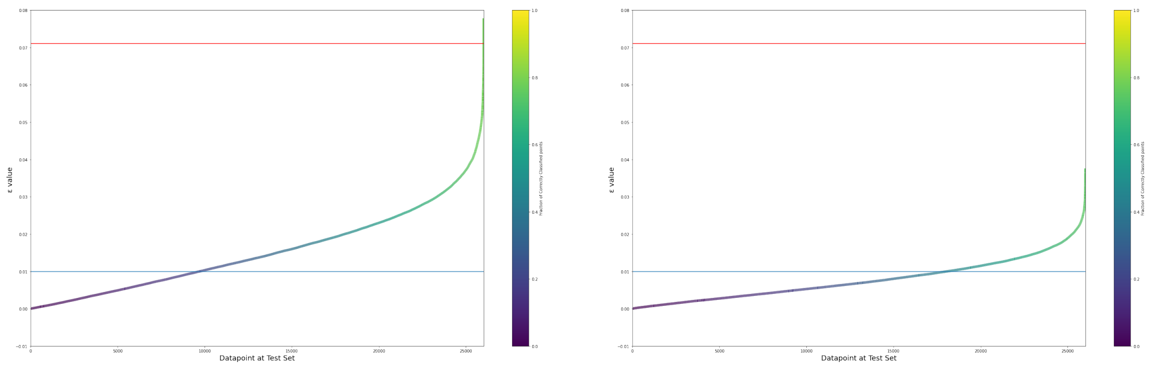


Figure 8.40: Maximum ε distances to the decision boundary of each data point of the test set in increasing ε order. The blue line denotes the ε value that the Robust model has been trained to be robust on (in our experiments $\varepsilon = 0.01$) and the red line denotes the maximum ε value of the respective teacher Robust Model. Left: Robust Distillated Re-Trained model (Accuracy: 78.423%) and Right: Adversarially Distillated Model (Accuracy: 74.720%) - SVHN Dataset.

Table 8.26: Average ε Certification Area - SVHN Dataset

Name	Trained Method	Average ε
RobustModel	Robust	0.01367
Distillated Robust Model	-	0.00646
ReTrained SCNN	Robust	0.01534
Adversarial Distill. SCNN	Adversarially	0.00770

From these experiments, it is shown that adversarial distillation does result in distillated models with larger certification areas compared to the other two distillation techniques ("Matching the Logits" and "Ranging the Temperature of the Teacher model") previously used. Nevertheless, its certification performance is significantly lower compared to the Robust Model (Retrained Distillated). In summary, from all three possible tested solutions to improve the certification performance and robustness of the network, only "Re-Training" generates a network with significantly improved certification and robustness results.

Chapter 9

Evaluation

This chapter aims to conduct a critical appraisal of the work conducted in this Master Thesis project compared to the original objectives. Starting from the objectives set out in the requirements capture in Chapter 3, the primal purpose of this project was to investigate certification properties of compressed shallow neural networks. Thereinafter, this task was broken into four major objective areas:

1. Model Distillation
2. Parameter Reduction
3. Adversarial Robustness
4. Certification Area

Model Distillation

The majority of the project work is oriented around model distillation techniques to compress and train efficient shallow neural networks. Two different techniques were implemented based on prior work by Caruana et al. ([7] & [27]) and Dean et al. ([25] & [32]). The two distillation techniques ("Matching the Logits" & "Ranging the Teacher's Temperature") were used along with two different image datasets to efficiently compress deep complex neural network models. These complex teacher models included either 4 convolutional layers (for the SVHN dataset) or 6 convolutional layers (for the CIFAR-10 dataset) prior to their fully connected part. Various network architectures with different number of parameters consisting of either one fully connected layer, one convolutional layer or two convolutional layers prior to the fully connected part were implemented and used as the student models in a series of distillation experiments. Through these experiments, the following conclusions can be drawn:

1. In order to achieve a high classification accuracy score, the student model needs to be **convolutional**. It was proven that architectures that include one or two convolutional layers have a significant higher performance than student models with only one fully connected layer.
2. The student model needs to be **deep**: the more convolutional layers, the better the accuracy. From the experiments, student models with two convolutional layers achieve higher accuracy than student models with only one convolutional layer prior to the fully connected part.
3. Both distillation techniques produce models with accuracy comparable to their teacher models when they have a large number of channels hence **large number of parameters**.

4. Raising the temperature of the teacher model to the **"ideal region"** (which depends on the network's architecture and the dataset) can result in better accuracy student models compared to the same architecture student models using "Matching the Logits" distillation technique.

Although the experiments on both datasets confirm the distillation conclusions described above, the final accuracy of the best student model in CIFAR-10 dataset is not very close to the teacher model. Unlike in the SVHN dataset experiments where the best student model achieved an accuracy of 92.306% (close to the teacher model's accuracy of 92.528%), the best student model in CIFAR-10 experiments achieved an accuracy of 77.140% while the teacher model has an accuracy of 82.560% in the test set. This is the result of two main factors:

1. The CIFAR-10 is a more difficult dataset and a deeper teacher model was used (6 convolutional layers) in order to achieve a high accuracy (higher than 80%) compared to SVHN dataset where a shallower teacher model (4 convolutional layers) model resulted in an accuracy 92%. One of the aforementioned conclusions is that the deeper the student model (more convolutional layers), the better the accuracy it achieves. In the experiments, Shallow CNNs of only up to two convolutional layers were explored. While for the SVHN dataset this consists 50% of the teacher's convolutional layers, for the CIFAR-10 it is only 1/3 of the teacher's convolutional layers. It is justifiable to assume that a SCNN with 3 convolutional layers could result in a better accuracy student model.

2. Unlike the SVHN dataset where more than 0.5M unlabelled images are provided, for the CIFAR-10 experiments only 110.000 images (from the CIFAR-100 combined with the CIFAR-10 training set) were used as the unlabelled dataset during the distillation experiments. If a larger portion of images from the 80 tiny images dataset was used, there would be a performance boost in all student models regardless their architecture.

Parameter Reduction

As noted in the distillation experiments, shallow distillated models require an increase in their parameter budget in order to achieve a classification accuracy comparable to their teacher models. Part of the project was about the investigation of parameter reduction methods to create models which combine the benefits of model distillation and parameter reduction. Using the "Pruning filters" [30] method, the number of parameters of the student Convolutional Neural Networks (CNNs) was drastically diminished. Through these experiments, the computational cost of the student CNNs was decreased by removing a large number of insignificant filters based on their L1-norm values. The experiments demonstrated that the shallow distillated student models can, in fact, have less parameters (close to their teacher models' number of parameters) without affecting (in some cases even improving) their overall performance.

Adversarial Robustness

Adversarial robustness of the distillated models has been investigated in this experimental task. Three different adversarial attack methods (Progressive Gradient Descent, Basic Iterative Method and Momentum Iterative Method) were implemented to craft adversarial examples with various perturbations. Both shallow student models and deep complex teacher models were undertaken under the same series of attacks using the aforementioned methods. It was found that shallow distillated neural networks demonstrate some robustness compared to their teacher models. In particular, for small perturbations the best student in both datasets proved to be more robust compared to their teacher models. While for large perturbations their performance was almost identical to their teacher networks.

Certification Area

The major objective of this task was to investigate the certification properties of shallow neural networks and determine under which circumstances distillation benefits the certification area of a neural network. To a large extend, this task was based on the prior work of Eric Wong et al. [47]. Two models with the same architecture were trained: one normally trained and one robustly trained using the ″Outer Convex Adversarial Polytope″ technique described in [47]. These models were used as the teacher models in a series of distillation and certification properties experiments. It was found that distillation plays an important role in increasing the certification area of a neural network. More specifically, the certification area of normally trained neural networks is increased when distillation is applied (all student models demonstrated a higher certification area compared to their teacher model). On the other hand, distillation of robustly trained teacher models resulted in shallow models with a smaller certification area compared to their teacher model. Although this evidence disproves the claim that distillation benefits the certification area of shallow models, through different experiments it was found that by re-training the student model robustly results in an important increase in their certification area compared to their teacher model. To further back the claim of the importance of distillation, two shallow models with the same architecture were trained until they reached the same accuracy in the two following ways:

1. The shallow neural network was trained straight robustly using the ″Outer Convex Adversarial Polytope″ technique (Model A).
2. The shallow neural network was first trained through distillation of a robustly trained deep neural network. At a subsequent stage, it was re-trained robustly using the ″Outer Convex Adversarial Polytope″ technique (Model B).

It was found that Model B has a significant larger certification area compared to Model A as well as its teacher model. In other words, it was shown that if distillation is part of a neural network's training process, it will enhance the network's certification area.

Summary of Contributions

Finally, a summary of the main contributions of this project is listed below:

1. The empirical affirmation of distillation conclusions described by Caruana et al. in [27].
2. A clean comparison between two distillation techniques: ″Matching the Logits″ [27] and ″Ranging the Teacher's Temperature [25].
3. Empirical demonstration of the dominance in terms of adversarial robustness of shallow distillated networks compared to deep neural networks for small adversarial perturbations.
4. The benefits of parameter reduction of shallow neural networks in terms of their computational cost and certification area.
5. The extension of the prior work by Eric Wong et al. [47]: Proving that distillation can result in shallow neural networks with enhanced certification area properties compared to their deep complex teacher models.

Chapter 10

Conclusion and Further Work

This project [1] explores the certification properties of deep neural networks by taking them into ”shallow waters”. The thesis brings together different research topics in the Machine Learning field and carefully connects them in order to efficiently train shallow neural networks and properly investigate and compare their certification properties. Throughout this process, the project has presented some of the major challenges posed by model distillation as well as possible solutions to address them.

As discussed in Chapter 9 in detail, starting with comparing and contrasting different model distillation techniques, several conclusions common for all tested distillation methods were drawn. It was found that in order to have a shallow student model with accuracy close to its complex teacher model, the student network has to be convolutional (include at least one convolutional layer) and deep (include several convolutional layers - less in number than the teacher model). An apparent drawback of model distillation is the necessary increase in the number of parameters of the student model. Pruning filters is an efficient solution to this problem which reduces not only the number of parameters without affecting the student network’s performance but also its computational cost. Furthermore, it was empirically demonstrated that distillation has significant benefits in adversarial robustness (for small perturbations). Finally, through a series of experiments, it was shown that model distillation can result in shallow neural networks with enhanced certification area properties compared to their deep complex teacher models, regardless of the method that their teacher networks have been trained with (normally or robustly).

Although a significant amount of work has been presented in this project, further research can be conducted. First of all, it seems only natural to consider different network architectures as well as possible alternatives to the training of robust classifiers. In this project, the ”Outer Convex Adversarial Polytope” technique by [47] has been considered as a method to train provably robust classification networks. Other methods that may or may not include convex looseness of bounds can be considered to train the teacher models which will be used in a series of distillation and certification experiments. Furthermore, as discussed in Chapters 4 and 6, methods for crafting adversarial examples can be also used to adversarially train neural networks. Additional and more complex adversarial attack methods should be considered and explored to adversarially train complex teacher models. Finally, by distillating these adversarially trained networks, a further evaluation of the adversarial robustness and certification properties of their corresponding shallow distillated neural networks should be conducted.

[1]The source code for the Master Thesis project will be made available at `https://github.com/KonstantinosBarmpas/Shallow-Waters`

Bibliography

[1] *80 million tiny images: a large dataset for non-parametric object and scene recognition.*

[2] *The Street View House Numbers (SVHN) Dataset - NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011.*

[3] Ludwig Schmidt Dimitris Tsipras Adrian Vladu Aleksander Madry, Aleksandar Makelov. *Towards Deep Learning Models Resistant to Adversarial Attacks*. 2019.

[4] Ilya Sutskever Alex Krizhevsky and Geoffrey E Hinton. *Imagenet Classification with Deep Convo- lutional Neural Networks. In NIPS*. 2012.

[5] Vinod Nair Alex Krizhevsky and Geoffrey Hinton. *CIFAR-10 Dataset.* 2017.

[6] Ian Goodfellow Alexey Kurakin and Samy Bengio. *Adversarial examples in the physical world.* 2016.

[7] Jimmy Ba and Rich Caruana. *Do deep nets really need to be deep? In NIPS*. 2014.

[8] M. A. Barreno. *Evaluating the Security of Machine Learning Algorithms (Master's Thesis). University of California, Berkeley, Berkeley, CA.* 2008.

[9] Nelson B. Joseph A. D. Tygar J. D. Barreno, M. *The security of machine learning. Machine Learning*. 2010.

[10] Nelson B. Sears R. Joseph A. D. Tygar J. D. Barreno, M. *Can machine learning be secure? Proceedings of the 2006 ACM Symposium on Information, computer and communications security, Taipei, Taiwan*. 2006.

[11] El Ghaoui L. Ben-Tal, A. and A. Nemirovski. *Robust optimization. Princeton University Press*. 2009.

[12] Roli F. Biggio, B. *Wild patterns: Ten years after the rise of adversarial machine learning.* 2018.

[13] N. Carlini and Wagner. *Towards evaluating the robustness of neural networks. In Security and Privacy (SP), 2017 IEEE Symposium on, pp. 39–57. IEEE.* 2017b.

[14] Alam M. Dey V. Chattopadhyay A. Mukhopadhyay D. Chakraborty, A. *Adversarial attacks and defenses: A Survey*. 2018.

[15] Perona P. Chalupka, K. and F. Eberhardt. *Visual Causal Feature Learning. ArXiv e-prints*. 2014.

[16] Nadav Cohen and Amnon Shashua. *Convolutional rectifier networks as generalized tensor decompositions. preprint arXiv:1603.0016*. 2016.

[17] Rich Caruana Cristian Bucila and Alexandru Niculescu-Mizil. *Model Compression, In Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, pages 535–541, New York, NY, USA, 2006. ACM*. 2006.

[18] George Cybenko. *Approximation by superpositions of a sigmoidal function. Mathematics of Control, Signals, 2(4):303–314*. 1989.

[19] Yann N. Dauphin and Yoshua Bengio. *Big neural networks waste capacity. arXiv:1301.3583*. 2013.

[20] Rob Fergus David Eigen, Jason Rolfe and Yann LeCun. *Understanding deep architectures using a recursive convolutional network. In ICLR (workshop track)*. 2014.

[21] T.N.Wiesel D.H.Hubel. *Receptive fields and functional architecture of monkey striate cortex ,The Journal of physiology 215–243*. 1968.

[22] Gang Li Frank Seide and Dong Yu. *Conversational speech transcription using context-dependent deep neural networks. In INTERSPEECH*. 2011.

[23] Razvan Pascanu James Bergstra Ian J. Goodfellow Arnaud Bergeron Nicolas Bouchard Frédéric Bastien, Pascal Lamblin and Yoshua Bengio. *Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop*. 2012.

[24] Matthew Mirman Markus Püschel Martin Vechev Gagandeep Singh, Timon Gehr. *Fast and Effective Robustness Certification*. 2018.

[25] Geoffrey Hinton Geoffrey Hinton and Jeff Dean. *Distilling the Knowledge in a Neural Network*. 2015.

[26] Krzysztof J. Geras and Charles Sutton. *Scheduled denoising autoencoders. In ICLR*. 2015.

[27] Samira Ebrahimi Kahou Ozlem Aslan Shengjie Wang Abdelrahman Mohamed Matthai Philipose Matt Richardson Rich Caruana Gregor Urban, Krzysztof J. Geras. *DO DEEP CONVOLUTIONAL NETS REALLY NEED TO BE DEEP AND CONVOLUTIONAL?* 2017.

[28] Shixiang Gu and Luca Rigazio. *Towards deep neural network architectures robust to adversarial examples. In NIPS Workshop on Deep Learning and Representation Learning*. 2014.

[29] Harleen Hanspal. *Provable robustness and analysis of Neural Networks against adversarial attacks.(Master's Thesis). ETH Zurich*. 2019.

[30] Igor Durdanovic Hanan Samet Hans Peter Graf Hao Li, Asim Kadav. *PRUNING FILTERS FOR EFFICIENT CONVNETS*. 2017.

[31] Jonathon Shlens Christian Szegedy Ian J. Goodfellow. *EXPLAINING AND HARNESSING ADVERSARIAL EXAMPLES*. 2015.

[32] R. Monga K. Chen M. Devin Q. V. Le M. Z. Mao M. Ranzato A. Senior P. Tucker K. Yang J. Dean, G. S. Corrado and A. Y. Ng. *Large scale distributed deep networks. In NIPS*. 2012.

[33] Jason Kuenb Lianyang Mab Amir Shahroudyb Bing Shuaib Ting Liub Xingxing Wangb Li Wangb Gang Wangb Jianfei Caic Tsuhan Chenc Jiuxiang Gua, Zhenhua Wangb. *Recent Advances in Convolutional Neural Networks*. 2017.

[34] Shaoqing Ren Kaiming He, Xiangyu Zhang and Jian Sun. *Deep residual learning for image recognition. arXiv:1512.03385*. 2015.

[35] Alexander Madry Zico Kolter. *Tutorials on adversarial robustness- theory and practise.* 2019.

[36] Rich Caruana Gregor Urban Shengjie Wang Ozlem Aslan Matthai Philipose Matthew Richardson Krzysztof J. Geras, Abdel-rahman Mohamed and Charles Sutton. *Blending LSTMs into CNNs. arXiv:1511.06433*. 2015.

[37] Shiyu Liang and R Srikant. *Why deep neural networks? arXiv preprint arXiv:1610.04161*. 2016.

[38] Makelov A.-Schmidt L. Tsipras D. Madry, A. and A. Vladu. *Towards deep learning models resistant to adversarial attacks. arXiv preprint arXiv:1706.06083*. 2017.

[39] Soheil Feizi Micah Goldblum, Liam Fowl and Tom Goldstein. *Adversarially Robust Distillation.* 2019.

[40] Nicholas Carlini Ian Goodfellow Reuben Feinman Alexey Kurakin Cihang Xie Yash Sharma Tom Brown Aurko Roy Alexander Matyasko Vahid Behzadan Karen Hambardzumyan Zhishuai Zhang Yi-Lin Juang Zhi Li Ryan Sheatsley Abhibhav Garg Jonathan Uesato Willi Gierke Yinpeng Dong David Berthelot Paul Hendricks Jonas Rauber Rujun Long Nicolas Papernot, Fartash Faghri and Patrick McDaniel. *Technical Report on the cleverhans v2.1.0 Adversarial Examples Library*. 2018.

[41] M. Ozdag. *Adversarial attacks and defenses against deep neural networks: a survey. Procedia Computer Science*. 2018.

[42] McDaniel P. Goodfellow I. Jha S. Celik Z. B. Papernot, N. and A. Swam. *Practical black-box attacks against deep learning systems using adversarial examples. In Proceedings of the 2017 ACM Asia Conference on Computer and Communications Security*. 2017.

[43] McDaniel P. Wu X. Jha S. Papernot, N. and Swami A. *Distillation as a defense to adversarial perturbations against deep neural networks. In Security and Privacy (SP), 2016 IEEE Symposium on, pp. 582–597. IEEE*. 20176.

[44] Steinhardt J. Raghunathan, A. and P. Liang. *Certified defenses against adversarial examples. In International Conference on Learning Representations*. 2018.

[45] Klaus Greff Rupesh K Srivastava and Juergen Schmidhuber. *Training very deep networks. In NIPS*. 2015.

[46] Karen Simonyan and Andrew Zisserman. *Very deep convolutional networks for large-scale image recognition. In ICLR*. 2014.

[47] Eric Wong and J. Zico Kolter. *Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope*. 2018.

[48] Caramanis C. Xu, H. and S. Mannor. *Robustness and regularization of support vector machines. Journal of Machine Learning Research, 10(Jul):1485–1510*. 2018.

[49] Tianyu Pang Hang Su Jun Zhu Xiaolin Hu Yinpeng Dong, Fangzhou Liao and Jianguo Li. *Boosting adversarial attacks with momentum.* 2017.

[50] He P. Zhu Q. Li X. Yuan, X. *Adversarial examples: Attacks and defenses for deep learning*. 2018.

[51] X. Zeng and T. R. Martinez. *Using a neural network to approximate an ensemble of classifiers. Neural Processing Letters, 12(3):225–237*. 2000.

Appendix A

Mathematical Derivations

A.1 Newton's Method

Let f be a well-defined function. Newton's method calculates an approximate solution to the equation

$$f(x) = 0 \tag{A.1}$$

using an initial approximation of the solution which is denoted by x_0. Iteratively, the Newton's Method calculates the tangent line at the approximation point x_0 and computes the point that the tangent line crosses the x-axis (in the graph x_1 for the tangent line at point x_0) and uses that crossing point as the new approximation for the next iteration.

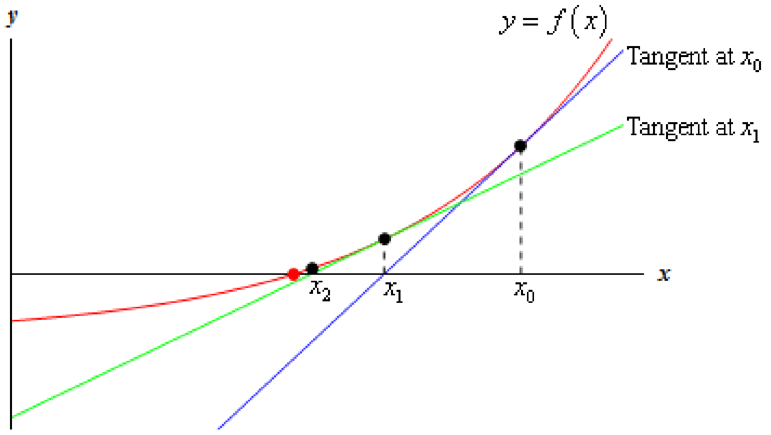


Figure A.1: Newton's Method Graphical Illustration
Source: https://tutorial.math.lamar.edu/classes/calci/newtonsmethod.aspx

Definition A.1 Newton's Method: If x_n is an approximation of the solution of $f(x) = 0$ and if $f'(x_n) \neq 0$, the next best approximation is given by:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{A.2}$$

where $f'(x_n)$ is the derivative of the function f at the point x_n.

A.2 Proof of Robust Error Bound

In this section, the proof for the Robust Error Bound explored in Chapter 7 of the thesis is presented. Note that the content of this section is heavily derived from [47] and is presented in this report for completeness.

Theorem Let L be a monotonic, translationally invariant loss function. For any data point (x,y) and $\varepsilon > 0$, the worst case adversarial loss from 7.12 can be bounded by:

$$\max_{||\Delta||_\infty \leq \varepsilon} L(f_\theta(x+\Delta), y) \leq L(-\mathcal{J}_\varepsilon(x, g_\theta(e_y 1^T - I)), y) \tag{A.3}$$

where $\mathcal{J}_\varepsilon$ and g_θ are defined in Chapter 7.

Proof: Let's rewrite the problem using the adversarial polytope $\mathcal{Z}_\varepsilon(x)$.

$$\max_{||\Delta||_\infty \leq \varepsilon} L(f_\theta(x+\Delta), y) = \max_{\hat{z}_k \in \mathcal{Z}_\varepsilon(x)} (L(\hat{z}_k, y)) \tag{A.4}$$

$$\max_{\hat{z}_k \in \mathcal{Z}_\varepsilon(x)} (L(\hat{z}_k, y)) \leq \max_{\hat{z}_k \in \mathcal{Z}_\varepsilon(x)} (L(\hat{z}_k - (\hat{z}_k)_y 1, y)) = \max_{\hat{z}_k \in \mathcal{Z}_\varepsilon(x)} (L(\hat{z}_k(I - e_y I^T), y)) \tag{A.5}$$

Let $C = I - e_y I^T$. Since L is monotonic loss function, the upper bound can be found by using the element-wise maximum over $[C_k]_i$ for all $i \neq y$ and the minimum for $i = y$ where $[C_k]_i = 0$. So,

$$\max_{\hat{z}_k \in \mathcal{Z}_\varepsilon(x)} (L(C\hat{z}_k, y)) \leq L(h(\hat{z}_k) \tag{A.6}$$

where:

$$h(z_k)_i = \max_{\hat{z}_k \in \mathcal{Z}_\varepsilon(x)} C_i \hat{z}_k \tag{A.7}$$

This is the primal formulation of the problem described in Chapter 7. Using the dual formulation it can be shown that:

$$\mathcal{J}_\varepsilon(x, g_\theta(-C_i)) \leq \min_{\hat{z}_k \in \mathcal{Z}_\varepsilon(x)} -C_i \hat{z}_k \tag{A.8}$$

Multiplying with -1:

$$-\mathcal{J}_\varepsilon(x, g_\theta(-C_i)) \geq \min_{\hat{z}_k \in \mathcal{Z}_\varepsilon(x)} C_i \hat{z}_k \tag{A.9}$$

Applying h:

$$h(z_k)_i \leq -\mathcal{J}_\varepsilon(x, g_\theta(-C_i)) \tag{A.10}$$

Applying this to all the elements of h gives the final upper bound:

$$\max_{||\Delta||_\infty \leq \varepsilon} L(f_\theta(x+\Delta), y) \leq L(-\mathcal{J}_\varepsilon(x, g_\theta(e_y 1^T - I)), y) \tag{A.11}$$

Definition A.2 For a data point $x \in \mathcal{X}$ and label y^*, the model is guaranteed to be robust in the perturbation ball $\mathcal{B}_\varepsilon$ around x if and only if [47]:

$$\mathcal{J}_\varepsilon(x, g_\theta(e_{y^*}1^T - I)) \geq 0 \tag{A.12}$$

where $\mathcal{J}$ and g_θ are defined in Chapter 7.

Proof: J is lower bound on the LP problem described in Chapter 7. Combining this fact with the certificate in equation above, for all $y \neq f(x)$:

$$\min_{\hat{z}_k \in \mathcal{Z}_\varepsilon(x)} (\hat{z}_k)_{f(x)} - (\hat{z}_k)_y \geq 0 \tag{A.13}$$

Appendix B

Experiments

B.1 Learning Curves for Teacher Models in SVHN Distillation Experiments

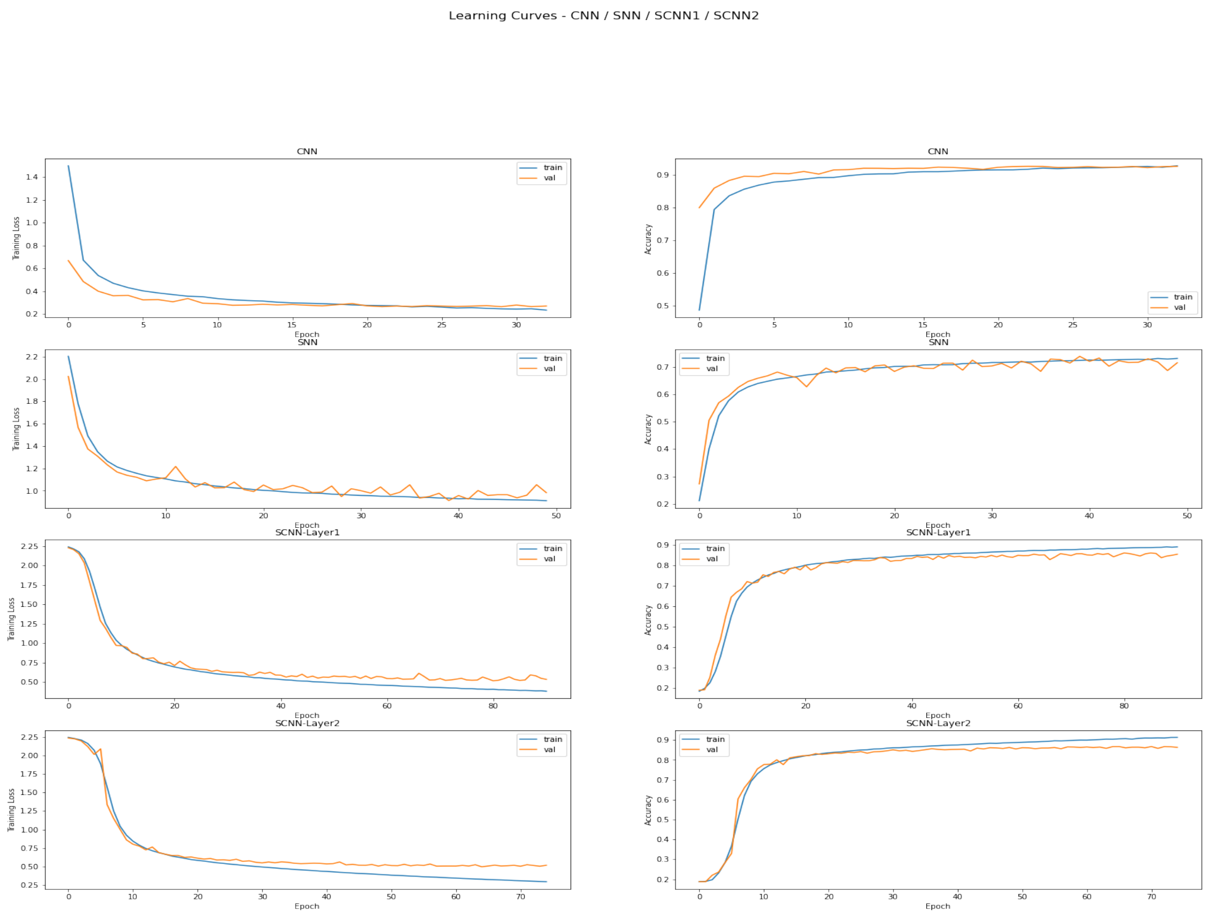


Figure B.1: Learning Curves for the CNN teacher model, the Shallow Neural Network (SNN), the one-convolutional layer Shallow Network (SCNN) and the two-convolutional layers Shallow Network (SCNN) trained on the original data

B.2 Ranging the Temperature Detailed Accuracy Results - Robust Model

Table B.1: SCNN Experiments Two Convolutional Layers with Teacher Temperature - Robust Model (SVHN) (1)

Name	Teacher's Temperature	Accuracy (Temperature)	Accuracy (MSE)
Number of Channels (L1-L2): 4-8 / Number of Parameters: 50k			
SCNN-MIMIC-h4-8_0.1	0.1	74.289%	-
SCNN-MIMIC-h4-8_0.2	0.2	74.216%	-
SCNN-MIMIC-h4-8_0.3	0.3	**74.685%**	-
SCNN-MIMIC-h4-8_0.4	0.4	74.458%	-
SCNN-MIMIC-h4-8_0.5	0.5	74.343%	-
SCNN-MIMIC-h4-8_0.6	0.6	74.155%	-
SCNN-MIMIC-h4-8_0.7	0.7	74.163%	-
SCNN-MIMIC-h4-8_0.8	0.8	74.182%	-
SCNN-MIMIC-h4-8_0.9	0.9	74.174%	-
SCNN-MIMIC-h4-8_1.0	1.0	74.213%	-
SCNN-MIMIC-h4-8_1.5	1.5	74.155%	-
SCNN-MIMIC-h4-8_2.0	2.0	74.643%	-
SCNN-MIMIC-h4-8_2.5	2.5	74.439%	-
SCNN-MIMIC-h4-8_3.0	3.0	74.424%	-
SCNN-MIMIC-h4-8_3.5	3.5	74.285%	-
SCNN-MIMIC-h4-8_4.0	4.0	74.566%	-
SCNN-MIMIC-h4-8_4.5	4.5	74.305%	-
SCNN-MIMIC-h4-8_5.0	5.0	73.901%	-
SCNN-MIMIC-h4-8	On Logits	-	**73.955%**
Number of Channels (L1-L2): 16-32 / Number of Parameters: 200k			
SCNN-MIMIC-h16-32_0.1	0.1	74.293%	-
SCNN-MIMIC-h16-32_0.2	0.2	**74.547%**	-
SCNN-MIMIC-h16-32_0.3	0.3	74.405%	-
SCNN-MIMIC-h16-32_0.4	0.4	74.424%	-
SCNN-MIMIC-h16-32_0.5	0.5	74.435%	-
SCNN-MIMIC-h16-32_0.6	0.6	74.293%	-
SCNN-MIMIC-h16-32_0.7	0.7	74.547%	-
SCNN-MIMIC-h16-32_0.8	0.8	74.274%	-
SCNN-MIMIC-h16-32_0.9	0.9	74.309%	-
SCNN-MIMIC-h16-32_1.0	1.0	74.316%	-
SCNN-MIMIC-h16-32_1.5	1.5	74.458%	-
SCNN-MIMIC-h16-32_2.0	2.0	74.213%	-
SCNN-MIMIC-h16-32_2.5	2.5	74.451%	-
SCNN-MIMIC-h16-32_3.0	3.0	74.213%	-
SCNN-MIMIC-h16-32_3.5	3.5	74.324%	-
SCNN-MIMIC-h16-32_4.0	4.0	74.105%	-
SCNN-MIMIC-h16-32_4.5	4.5	74.489%	-
SCNN-MIMIC-h16-32_5.0	5.0	74.520%	-
SCNN-MIMIC-h16-32	On Logits	-	**74.589%**

Table B.2: SCNN Experiments Two Convolutional Layers with Teacher Temperature - Robust Model (SVHN) (2)

Name	Teacher's Temperature	Accuracy (Temperature)	Accuracy (MSE)
Number of Channels (L1-L2): 16-64 / Number of Parameters: 400k			
SCNN-MIMIC-h16-64_0.1	0.1	74.350%	-
SCNN-MIMIC-h16-64_0.2	0.2	**74.650%**	-
SCNN-MIMIC-h16-64_0.3	0.3	74.501%	-
SCNN-MIMIC-h16-64_0.4	0.4	74.566%	-
SCNN-MIMIC-h16-64_0.5	0.5	74.504%	-
SCNN-MIMIC-h16-64_0.6	0.6	74.335%	-
SCNN-MIMIC-h16-64_0.7	0.7	74.458%	-
SCNN-MIMIC-h16-64_0.8	0.8	74.581%	-
SCNN-MIMIC-h16-64_0.9	0.9	74.343%	-
SCNN-MIMIC-h16-64_1.0	1.0	74.393%	-
SCNN-MIMIC-h16-64_1.5	1.5	74.458%	-
SCNN-MIMIC-h16-64_2.0	2.0	74.493%	-
SCNN-MIMIC-h16-64_2.5	2.5	74.451%	-
SCNN-MIMIC-h16-64_3.0	3.0	74.566%	-
SCNN-MIMIC-h16-64_3.5	3.5	74.451%	-
SCNN-MIMIC-h16-64_4.0	4.0	74.504%	-
SCNN-MIMIC-h16-64_4.5	4.5	74.589%	-
SCNN-MIMIC-h16-64_5.0	5.0	74.216%	-
SCNN-MIMIC-h16-64	On Logits	-	**74.474%**
Number of Channels (L1-L2): 32-128 / Number of Parameters: 850k			
SCNN-MIMIC-h32-128_0.1	0.1	**74.846%**	-
SCNN-MIMIC-h32-128_0.2	0.2	74.612%	-
SCNN-MIMIC-h32-128_0.3	0.3	74.431%	-
SCNN-MIMIC-h32-128_0.4	0.4	74.320%	-
SCNN-MIMIC-h32-128_0.5	0.5	74.389%	-
SCNN-MIMIC-h32-128_0.6	0.6	74.362%	-
SCNN-MIMIC-h32-128_0.7	0.7	74.393%	-
SCNN-MIMIC-h32-128_0.8	0.8	74.504%	-
SCNN-MIMIC-h32-128_0.9	0.9	74.351%	-
SCNN-MIMIC-h32-128_1.0	1.0	74.362%	-
SCNN-MIMIC-h32-128_1.5	1.5	74.335%	-
SCNN-MIMIC-h32-128_2.0	2.0	74.574%	-
SCNN-MIMIC-h32-128_2.5	2.5	74.654%	-
SCNN-MIMIC-h32-128_3.0	3.0	74.462%	-
SCNN-MIMIC-h32-128_3.5	3.5	74.431%	-
SCNN-MIMIC-h32-128_4.0	4.0	74.439%	-
SCNN-MIMIC-h32-128_4.5	4.5	74.547%	-
SCNN-MIMIC-h32-128_5.0	5.0	74.600%	-
SCNN-MIMIC-h32-128	On Logits	-	**74.569%**

B.3 Ranging the Temperature Detailed Accuracy Results

Table B.3: SCNN Experiments Two Convolutional Layers with Teacher Temperature - SVHN Dataset

Name	Teacher's Temperature	Accuracy (d)	Accuracy (b)
Number of Channels (L1-L2): 32-32 / Number of Parameters: 220k			
SCNN-MIMIC-SVHN-h32-32_1.5	1.5	91.457%	-
SCNN-MIMIC-SVHN-h32-32_2.0	2.0	**91.845%**	-
SCNN-MIMIC-SVHN-h32-32_2.5	2.5	91.795%	-
SCNN-MIMIC-SVHN-h32-32_3.0	3.0	91.545%	-
SCNN-MIMIC-SVHN-h32-32_3.5	3.5	91.457%	-
SCNN-MIMIC-SVHN-h32-32_4.0	4.0	91.334%	-
SCNN-MIMIC-SVHN-h32-32_4.5	4.5	91.468%	-
SCNN-MIMIC-SVHN-h32-32_5.0	5.0	91.338%	-
SCNN-MIMIC-SVHN-h32-32	On Logits	-	**90.646%**
Number of Channels (L1-L2): 64-128 / Number of Parameters: 850k			
SCNN-MIMIC-SVHN-h64-128_1.5	1.5	**92.306%**	-
SCNN-MIMIC-SVHN-h64-128_2.0	2.0	92.006%	-
SCNN-MIMIC-SVHN-h64-128_2.5	2.5	92.018%	-
SCNN-MIMIC-SVHN-h64-128_3.0	3.0	92.018%	-
SCNN-MIMIC-SVHN-h64-128_3.5	3.5	91.825%	-
SCNN-MIMIC-SVHN-h64-128_4.0	4.0	91.825%	-
SCNN-MIMIC-SVHN-h64-128_4.5	4.5	91.895%	-
SCNN-MIMIC-SVHN-h64-128_5.0	5.0	91.883%	-
SCNN-MIMIC-SVHN-h64-128	On Logits	-	**91.361%**
Number of Channels (L1-L2): 64-512 / Number of Parameters: 3.5M			
SCNN-MIMIC-SVHN-h64-512_1.5	1.5	92.152%	-
SCNN-MIMIC-SVHN-h64-512_2.0	2.0	91.160%	-
SCNN-MIMIC-SVHN-h64-512_2.5	2.5	**92.202%**	-
SCNN-MIMIC-SVHN-h64-512_3.0	3.0	91.964%	-
SCNN-MIMIC-SVHN-h64-512_3.5	3.5	92.010%	-
SCNN-MIMIC-SVHN-h64-512_4.0	4.0	92.010%	-
SCNN-MIMIC-SVHN-h64-512_4.5	4.5	92.056%	-
SCNN-MIMIC-SVHN-h64-512_5.0	5.0	91.522%	-
SCNN-MIMIC-SVHN-h64-512	On Logits	-	**91.583%**

(*) The SCNN-MIMIC-SVHN-h350-1024 experiments were not conducted due to the huge training time each model required. We expect similar results though.

Table B.4: SCNN Experiments Two Convolutional Layers with Teacher Temperature - SVHN Dataset

Name	Teacher's Temperature	Accuracy (d)	Accuracy (b)
Number of Channels (L1-L2): 32-32 / Number of Parameters: 220k			
SCNN-MIMIC-SVHN-h32-32_0.1	0.1	90.734%	-
SCNN-MIMIC-SVHN-h32-32_0.2	0.2	90.984%	-
SCNN-MIMIC-SVHN-h32-32_0.3	0.3	90.596%	-
SCNN-MIMIC-SVHN-h32-32_0.4	0.4	91.080%	-
SCNN-MIMIC-SVHN-h32-32_0.5	0.5	91.172%	-
SCNN-MIMIC-SVHN-h32-32_0.6	0.6	90.992%	-
SCNN-MIMIC-SVHN-h32-32_0.7	0.7	91.049%	-
SCNN-MIMIC-SVHN-h32-32_0.8	0.8	91.253%	-
SCNN-MIMIC-SVHN-h32-32_0.9	0.9	**91.461%**	-
SCNN-MIMIC-SVHN-h32-32_1.0	1.0	91.299%	-
SCNN-MIMIC-SVHN-h32-32	On Logits	-	**90.646%**
Number of Channels (L1-L2): 64-128 / Number of Parameters: 850k			
SCNN-MIMIC-SVHN-h64-128_0.1	0.1	90.819%	-
SCNN-MIMIC-SVHN-h64-128_0.2	0.2	91.257%	-
SCNN-MIMIC-SVHN-h64-128_0.3	0.3	91.526%	-
SCNN-MIMIC-SVHN-h64-128_0.4	0.4	91.272%	-
SCNN-MIMIC-SVHN-h64-128_0.5	0.5	91.545%	-
SCNN-MIMIC-SVHN-h64-128_0.6	0.6	91.299%	-
SCNN-MIMIC-SVHN-h64-128_0.7	0.7	91.633%	-
SCNN-MIMIC-SVHN-h64-128_0.8	0.8	91.679%	-
SCNN-MIMIC-SVHN-h64-128_0.9	0.9	91.829%	-
SCNN-MIMIC-SVHN-h64-128_1.0	1.0	**91.902%**	-
SCNN-MIMIC-SVHN-h64-128	On Logits	-	**91.361%**
Number of Channels (L1-L2): 64-512 / Number of Parameters: 3.5M			
SCNN-MIMIC-SVHN-h64-512_0.1	0.1	91.207%	-
SCNN-MIMIC-SVHN-h64-512_0.2	0.2	91.576%	-
SCNN-MIMIC-SVHN-h64-512_0.3	0.3	91.299%	-
SCNN-MIMIC-SVHN-h64-512_0.4	0.4	91.441%	-
SCNN-MIMIC-SVHN-h64-512_0.5	0.5	91.848%	-
SCNN-MIMIC-SVHN-h64-512_0.6	0.6	91.464%	-
SCNN-MIMIC-SVHN-h64-512_0.7	0.7	92.071%	-
SCNN-MIMIC-SVHN-h64-512_0.8	0.8	91.952%	-
SCNN-MIMIC-SVHN-h64-512_0.9	0.9	**92.083%**	-
SCNN-MIMIC-SVHN-h64-512_1.0	1.0	91.887%	-
SCNN-MIMIC-SVHN-h64-512	On Logits	-	**91.583%**
Number of Channels (L1-L2): 350-1024 / Number of Parameters: 9.5M			
SCNN-MIMIC-SVHN-h350-1024_0.1	0.1	91.153%	-
SCNN-MIMIC-SVHN-h350-1024_0.2	0.2	91.195%	-
SCNN-MIMIC-SVHN-h350-1024_0.3	0.3	91.280%	-
SCNN-MIMIC-SVHN-h350-1024_0.4	0.4	91.760%	-
SCNN-MIMIC-SVHN-h350-1024_0.5	0.5	91.806%	-
SCNN-MIMIC-SVHN-h350-1024_0.6	0.6	91.649%	-
SCNN-MIMIC-SVHN-h350-1024_0.7	0.7	91.722%	-
SCNN-MIMIC-SVHN-h350-1024_0.8	0.8	91.776%	-
SCNN-MIMIC-SVHN-h350-1024_0.9	0.9	91.952%	-
SCNN-MIMIC-SVHN-h350-1024_1.0	1.0	**91.956%**	-
SCNN-MIMIC-SVHN-h350-1024	On Logits	-	**91.637%**

Table B.5: SCNN Experiments Two Convolutional Layers with Teacher Temperature - CIFAR10

Name	Teacher's Temperature	Accuracy (d)	Accuracy (b)
Number of Channels (L1-L2): 64-128 / Number of Parameters: 650k			
SCNN-MIMIC-CIFAR10-h64-128_0.1	0.1	71.990%	-
SCNN-MIMIC-CIFAR10-h64-128_0.2	0.2	72.400%	-
SCNN-MIMIC-CIFAR10-h64-128_0.3	0.3	73.050%	-
SCNN-MIMIC-CIFAR10-h64-128_0.4	0.4	73.270%	-
SCNN-MIMIC-CIFAR10-h64-128_0.5	0.5	73.350%	-
SCNN-MIMIC-CIFAR10-h64-128_0.6	0.6	73.690%	-
SCNN-MIMIC-CIFAR10-h64-128_0.7	0.7	73.990%	-
SCNN-MIMIC-CIFAR10-h64-128_0.8	0.8	74.420%	-
SCNN-MIMIC-CIFAR10-h64-128_0.9	0.9	74.430%	-
SCNN-MIMIC-CIFAR10-h64-128_1.0	1.0	**74.710%**	-
SCNN-MIMIC-CIFAR10-h64-128	On Logits	-	**73.450%**
Number of Channels (L1-L2): 64-512 / Number of Parameters: 2.5M			
SCNN-MIMIC-CIFAR10-h64-512_0.1	0.1	72.800%	-
SCNN-MIMIC-CIFAR10-h64-512_0.2	0.2	72.640%	-
SCNN-MIMIC-CIFAR10-h64-512_0.3	0.3	73.650%	-
SCNN-MIMIC-CIFAR10-h64-512_0.4	0.4	73.350%	-
SCNN-MIMIC-CIFAR10-h64-512_0.5	0.5	73.520%	-
SCNN-MIMIC-CIFAR10-h64-512_0.6	0.6	74.930%	-
SCNN-MIMIC-CIFAR10-h64-512_0.7	0.7	75.000%	-
SCNN-MIMIC-CIFAR10-h64-512_0.8	0.8	75.200%	-
SCNN-MIMIC-CIFAR10-h64-512_0.9	0.9	74.960%	-
SCNN-MIMIC-CIFAR10-h64-512_1.0	1.0	**75.260%**	-
SCNN-MIMIC-CIFAR10-h64-512	On Logits	-	**75.120%**
Number of Channels (L1-L2): 128-1024 / Number of Parameters: 6M			
SCNN-MIMIC-CIFAR10-h128-1024_0.1	0.1	73.130%	-
SCNN-MIMIC-CIFAR10-h128-1024_0.2	0.2	73.810%	-
SCNN-MIMIC-CIFAR10-h128-1024_0.3	0.3	73.670%	-
SCNN-MIMIC-CIFAR10-h128-1024_0.4	0.4	74.020%	-
SCNN-MIMIC-CIFAR10-h128-1024_0.5	0.5	73.690%	-
SCNN-MIMIC-CIFAR10-h128-1024_0.6	0.6	74.250%	-
SCNN-MIMIC-CIFAR10-h128-1024_0.7	0.7	74.670%	-
SCNN-MIMIC-CIFAR10-h128-1024_0.8	0.8	73.870%	-
SCNN-MIMIC-CIFAR10-h128-1024_0.9	0.9	75.770%	-
SCNN-MIMIC-CIFAR10-h128-1024_1.0	1.0	**76.670%**	-
SCNN-MIMIC-CIFAR10-h128-1024	On Logits	-	**75.460%**

(*) The SCNN-MIMIC-CIFAR10-256-2048 experiments were not conducted due to the huge training time each model required. We expect similar results though.

Table B.6: SCNN Experiments Two Convolutional Layers with Teacher Temperature - CIFAR10

Name	Teacher's Temperature	Accuracy (d)	Accuracy (b)
Number of Channels (L1-L2): 64-128 / Number of Parameters: 650k			
SCNN-MIMIC-CIFAR10-h64-128_1.5	1.5	75.920%	-
SCNN-MIMIC-CIFAR10-h64-128_2.0	2.0	76.140%	-
SCNN-MIMIC-CIFAR10-h64-128_2.5	2.5	76.200%	-
SCNN-MIMIC-CIFAR10-h64-128_3.0	3.0	75.510%	-
SCNN-MIMIC-CIFAR10-h64-128_3.5	3.5	75.320%	-
SCNN-MIMIC-CIFAR10-h64-128_4.0	4.0	76.180%	-
SCNN-MIMIC-CIFAR10-h64-128_4.5	4.5	**76.570%**	-
SCNN-MIMIC-CIFAR10-h64-128_5.0	5.0	76.020%	-
SCNN-MIMIC-CIFAR10-h64-128	On Logits	-	**73.450%**
Number of Channels (L1-L2): 64-512 / Number of Parameters: 2.5M			
SCNN-MIMIC-CIFAR10-h64-512_1.5	1.5	76.480%	-
SCNN-MIMIC-CIFAR10-h64-512_2.0	2.0	**77.140%**	-
SCNN-MIMIC-CIFAR10-h64-512_2.5	2.5	76.400%	-
SCNN-MIMIC-CIFAR10-h64-512_3.0	3.0	76.700%	-
SCNN-MIMIC-CIFAR10-h64-512_3.5	3.5	76.860%	-
SCNN-MIMIC-CIFAR10-h64-512_4.0	4.0	75.480%	-
SCNN-MIMIC-CIFAR10-h64-512_4.5	4.5	76.300%	-
SCNN-MIMIC-CIFAR10-h64-512_5.0	5.0	76.810%	-
SCNN-MIMIC-CIFAR10-h64-512	On Logits	-	**75.120%**
Number of Channels (L1-L2): 128-1024 / Number of Parameters: 6M			
SCNN-MIMIC-CIFAR10-h128-1024_1.5	1.5	76.330%	-
SCNN-MIMIC-CIFAR10-h128-1024_2.0	2.0	76.310%	-
SCNN-MIMIC-CIFAR10-h128-1024_2.5	2.5	76.360%	-
SCNN-MIMIC-CIFAR10-h128-1024_3.0	3.0	75.750%	-
SCNN-MIMIC-CIFAR10-h128-1024_3.5	3.5	**76.820%**	-
SCNN-MIMIC-CIFAR10-h128-1024_4.0	4.0	76.050%	-
SCNN-MIMIC-CIFAR10-h128-1024_4.5	4.5	76.330%	-
SCNN-MIMIC-CIFAR10-h128-1024_5.0	5.0	75.710%	-
SCNN-MIMIC-CIFAR10-h128-1024	On Logits	-	**75.460%**

(*) The SCNN-MIMIC-CIFAR10-256-2048 experiments were not conducted due to the huge training time each model required. We expect similar results though.

Appendix C

Certification Area & Number of Parameters

C.1 CIFAR-10 Dataset Certification Area Graphs

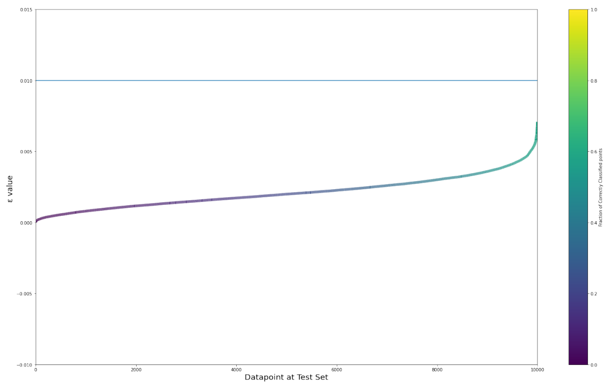


Figure C.1: Baseline Model

Accuracy: 60.330%
250k parameters

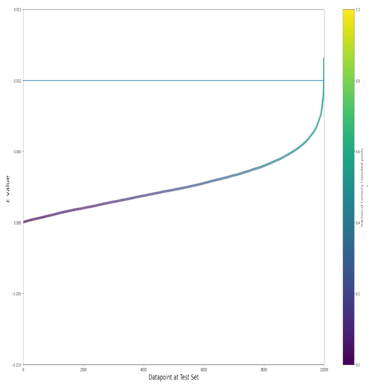


Figure C.2: Distillated Model

Accuracy: 56.220%
260k parameters

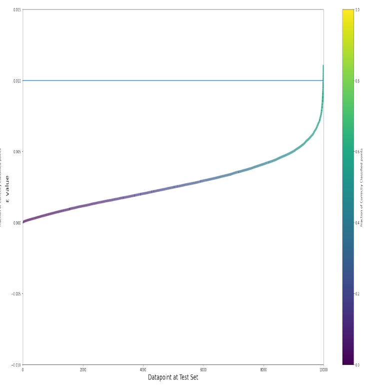


Figure C.3: Distillated Model

Accuracy: 59.250%
1M parameters

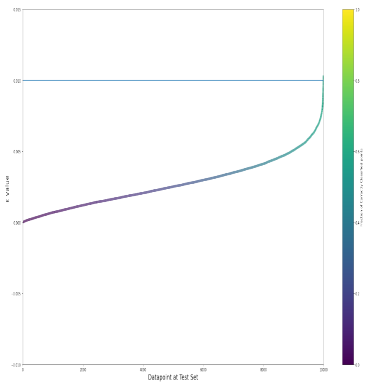


Figure C.4: Distillated Model

Accuracy: 58.920%
2M parameters

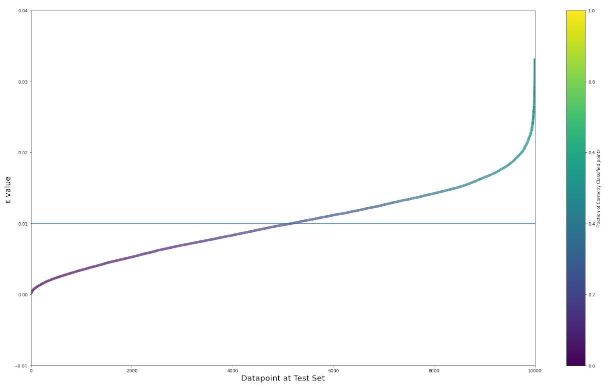
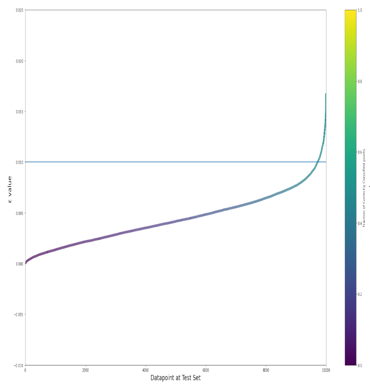
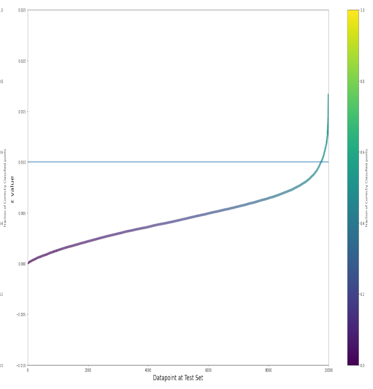


Figure C.5: Robust Model

Accuracy: 53.180%
250k parameters





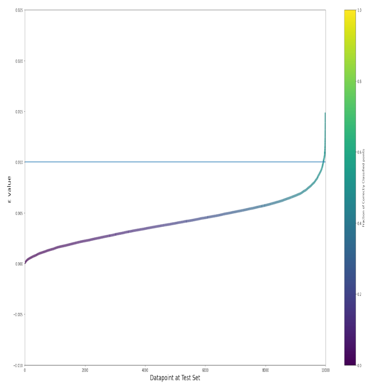


Figure C.6: Distillated Model

Accuracy: 50.710%
260k parameters

Figure C.7: Distillated Model

Accuracy: 52.430%
1M parameters

Figure C.8: Distillated Model

Accuracy: 52.880%
2M parameters

C.2 SVHN Dataset Certification Area Graphs

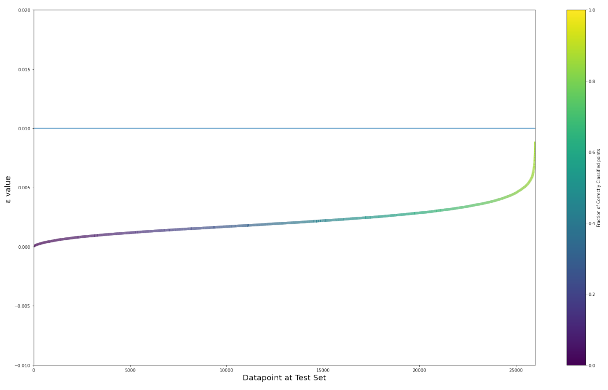


Figure C.9: Baseline Model

Accuracy: 60.330%
250k parameters

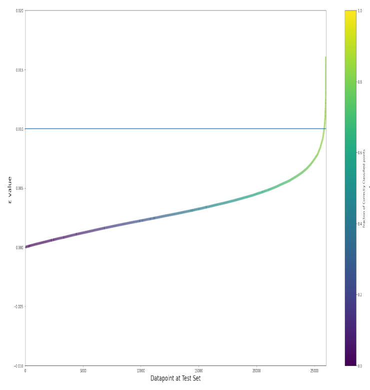


Figure C.10: Distillated Model

Accuracy: 56.220%
260k parameters

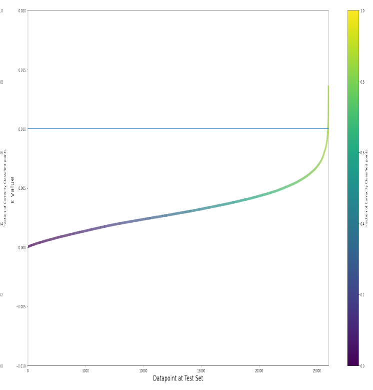


Figure C.11: Distillated Model

Accuracy: 59.250%
1M parameters

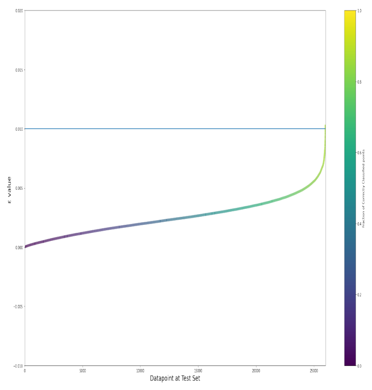


Figure C.12: Distillated Model

Accuracy: 58.920%
2M parameters

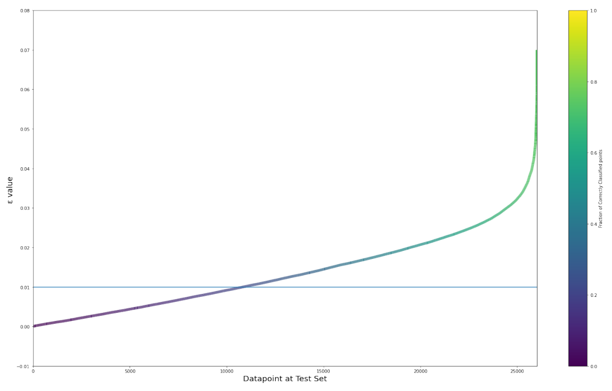


Figure C.13: Robust Model

Accuracy: 53.180%
250k parameters

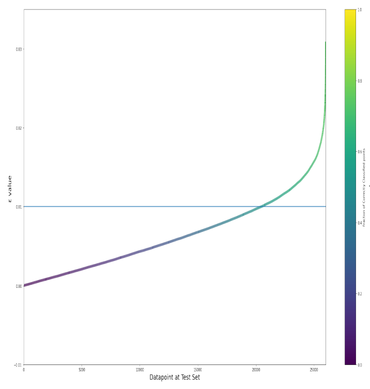


Figure C.14: Distillated Model

Accuracy: 50.710%
260k parameters

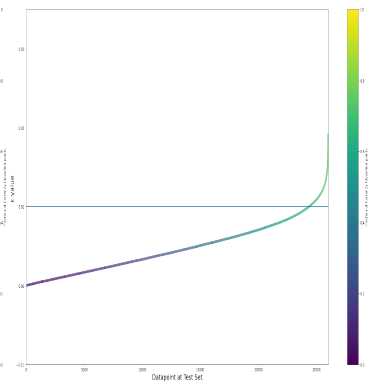


Figure C.15: Distillated Model

Accuracy: 52.430%
1M parameters

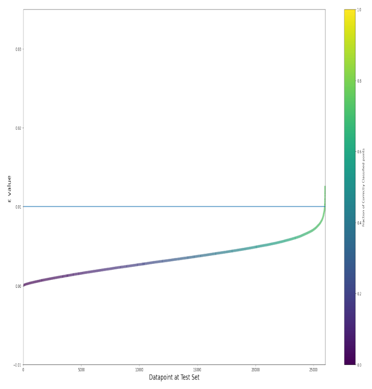


Figure C.16: Distillated Model

Accuracy: 52.880%
2M parameters

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

CERTIFYING PROPERTIES OF DEEP NEURAL NETWORKS BY TAKING THEM INTO SHALLOW WATERS

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
BARMPAS	KONSTANTINOS

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
ZURICH, AUGUST 15 2020	Barmpas K

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.