Margin-based regularization for deep neural networks

WD Brooks

 orcid.org/ 0000-0001-4373-8350

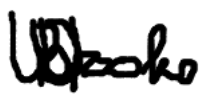
Dissertation accepted in fulfilment of the requirements for the degree *Master of Engineering in Computer and Electronic Engineering* at the North-West University

Supervisor:	Prof MH Davel
Co-supervisor:	Dr C Mouton

Graduation: June 2025

Declaration

I, William David Brooks, hereby declare that the dissertation entitled "Margin-based regularization for deep neural networks" is my own original work and has not already been submitted to any other university or institution for examination.



WD. Brooks

Student number: 31637043

Signed on the 27th day of January 2024 at Potchefstroom.

Abstract

Deep Neural Networks (DNNs) have achieved significant success in various tasks, yet understanding the factors that enhance their generalization remains a challenge. This study aims to explore whether recent advances in post hoc margin-based generalization prediction measures can be applied to effectively enhance the generalization ability of modern DNNs. Specifically, we explore the use of constrained margins as a regularization technique to improve generalization, focusing on their application within robustness-enhancing methods.

Large margins – the minimum distance between data points and decision boundaries – in the input space are known to correlate with increased robustness. In this study, we focus on two margin-maximizing approaches: a simple approach which modifies the loss function to increase an approximation of the margin (Large margin loss), and a more complex state-of-the-art method (Dynamics-Aware Robust Training) which builds upon this approach.

We first study standard margin maximization methods aimed at improving the robustness of DNNs. Our experiments, conducted on the CIFAR-10 dataset, identify and assess the effectiveness of the components introduced in the large margin loss and DyART in improving robustness. Model performance is evaluated under well-known adversarial attacks, including AutoAttack and Projected Gradient Descent (PGD). We find that while more precise margin approximations do not improve the generalization ability of DNNs, calculating the gradient of the margin with respect to model parameters leads to substantial robustness improvements.

Following this, we adapt constrained margins, a metric which has shown to be more correlated with generalization than standard margins, to both the large margin loss function and DyART. Our aim is to investigate how maximizing these constrained margins impact generalization during training in approaches traditionally designed to enhance robustness using standard margins. We found that maximizing constrained margins during training neither improves generalization performance nor improves adversarial robustness of DNNs.

Keywords: *Deep Neural Network, Adversarial robustness, Generalization, Margins, Dynamics-Aware Robust Training, Large margin loss, Adversarial attacks, Constrained Margins*

Contents

List of Figures

List of Tables

List of Abbreviations

DNN Deep Neural Network

ML Machine Learning

SGD Stochastic Gradient Descent

SLT Statistical Learning Theory

HP Hyperparameter

PC Principal Component

AT Adversarial Training

PGD Projected Gradient Descent

FAB Fast Adaptive Boundary Attack

SVM Support Vector Machine

PCA Principal Component Analysis

MM Margin Maximization

DyART Dynamics-Aware Robust Training

WRN Wide Residual Network

CNN Convolutional Neural Network

FGSM Fast Gradient Sign Method

CW Carlini and Wagner attack

SOTA State-of-the-Art

APGD Auto Projected Gradient Descent

APGD-t Targeted Auto Projected Gradient Descent

CMP Constrained Margin Plane

PGDL Predicting Genralization in Deep Learning

List of Symbols & Subscripts

List of Symbols

x	Inputs
y	Labels
X	Input set
Y	Label set
f	Deep classifier
θ	Internal parameters
Δ	Minimal perturbation
δ	Perturbation
ϵ	Perturbation bound
S	Perturbation set
L	Loss
∇	Gradient
η	Step size/extrapolation step
ϕ	Output margin
z	Logits
R	Margin
C	Constraint
l	Lower bound
u	Upper bound
μ	Weighting factor
γ	Minimum margin
λ	Cross weight
B	Batch

N	Batch size
α	Exponential penalization term
β	Temperature
P	Principal component matrix

List of Subscripts

p	Distance norm
q	Dual norm
i	Iteration/index
T	Transposed of Matrix
m	Top principle components

Chapter 1

Introduction

Matthew 6:33: "But seek first the kingdom of God, and His righteousness and all these things shall be added to you."

1.1 Introduction

Deep Neural Networks (DNNs) have transformed numerous fields and have achieved remarkable success in tasks ranging from image classification to natural language processing. The concept of generalization of DNNs, a measure of how well a model performs on unseen data [1]–[3], remains an area of ongoing research. Adversarial robustness is a topic closely related to generalization. It can be defined as the ability of DNNs to withstand adversarial attacks: visually imperceptible perturbations to input data that cause a model to misclassify that sample [4], [5].

Work from various studies [1], [6]–[8] indicate an inherent trade-off between generalization and adversarial robustness. These studies found that methods used to increase adversarial robustness typically lead to a decrease in the generalization ability of models. We aim to investigate whether the concept of margin maximization, which is actively used to improve adversarial robustness [9], [10], can also be used as a regularization technique to improve generalization.

Section 1.1.1 introduces the concept of generalization and examines the role of regularizers in

enhancing the generalization ability of DNNs. Section 1.1.2 focuses on adversarial robustness, presenting methods such as Adversarial Training (AT) and Margin Maximization (MM) to improve robustness. Section 1.1.3 builds on these concepts by introducing margins, highlighting their relationship with both generalization and adversarial robustness, and emphasizing how maximizing margins enhances robustness. Finally, Section 1.1.4 explores current margin maximization techniques for improving robustness and their impact on DNN performance.

1.1.1 Generalization and regularization

One of the core challenges in modern DNNs is achieving generalization, the ability of a model to perform well on new, unseen data. A model that generalizes can apply what it has learned in various scenarios, which is critical for deploying models in real-world applications. Despite their widespread adoption, our understanding of why these models generalize as well as they do despite great overparameterization is still limited and remains an open research question. This emphasizes the importance of understanding generalization in the context of traditional Machine Learning (ML) which is well understood within the framework of Statistical Learning Theory (SLT), before discussing generalization in DNNs.

A model that generalizes well exhibits a small 'generalization gap': a small difference between a model's train and generalization error (also known as test error). Figure 1.1 from Goodfellow et al. [11, p. 113] provides an illustration of this concept.

The capacity of a model can be defined as the ability of the model to fit a variety of complex functions [12] which affects both its ability to fit data and generalize well [11, p. 110]. Fitting the data involves adjusting the model's internal parameters, typically through optimizers such as Stochastic Gradient Descent (SGD). In the case of DNNs, determining the effective capacity is challenging because it is limited by the optimizers' ability to find the set of parameters that minimize the loss. The loss functions in DNN are highly complex and nonlinear, making finding the global minimum a difficult task.

From Figure 1.1, we can see that a model that generalizes well has less capacity, i.e., is represented by simpler functions. However, we still need a low training error. This leads us to explain two critical aspects: overfitting and underfitting, which are often discussed in terms of the bias-variance trade-off in SLT.

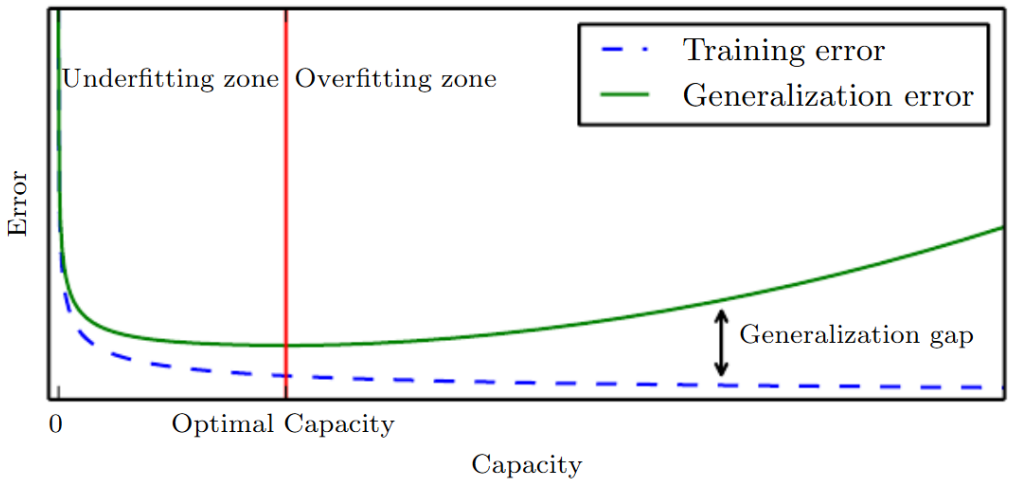


Figure 1.1: Illustration of the expected generalization gap in ML, highlighting the concepts of overfitting and underfitting as they relate to model capacity and error [11].

Overfitting describes when a model performs well on the training data but fails to capture the underlying patterns needed to generalize to new, unseen inputs, i.e. the model effectively memorizes the training data (models with high capacity) i.e. high variance and low bias. In contrast, underfitting occurs when a model's capacity is too low to capture the underlying patterns in the data, resulting in poor performance on both the training and test data i.e. high bias and low variance. We can mitigate the effect of underfitting by increasing training times or increasing the model's complexity. However, increasing the complexity of the model can also lead to overfitting if not controlled properly.

Regularization techniques are designed to prevent overfitting and reduce the gap between training and test error, assisting a model to generalize better. In modern ML, regularization refers to any addition that improves generalization. In DNNs, regularization typically involves adding an explicit term to the loss function which aids in parameter selection. Many regularization techniques have been proposed such as L_2 (weight decay) and L_1 regularization, dropout, data augmentation, label smoothing and early stopping [11]. These methods aim to improve generalization, by reducing a model's test error but not its training error [11], [13]. Techniques that stabilise training, such as batch normalization, have also been introduced, allowing models to train faster and generalize better, exhibiting a regularization effect [14].

Despite this theoretical understanding of generalization, DNNs sometimes contradict these intuitions, and these models exhibit a phenomenon called *double decent*. This phenomenon typically

occurs as a "second phase" to what we see in Figure 1.1. As the capacity of the model increases and the generalization gap increases, we reach a point of interpolation (where all the training samples have been fitted). As the models capacity increases beyond this point[15], [16] of interpolation, it can both memorize the noise in the data and generalize well. The exact reason for this remains unknown.

1.1.2 Adversarial robustness

DNNs are inherently vulnerable to adversarial perturbations – small perturbations to the input data that lead a deep learning model to misclassify the input, often with high confidence [4], [5]. This vulnerability poses challenges where reliability and security of machine learning models are paramount.

The exact reason for the susceptibility of DNNs to these perturbations remains unknown and is an ongoing area of research. Several methods have been developed to generate adversarial examples and evaluate the robustness of DNNs [17]–[20]. Popular attack algorithms, such as Projected Gradient Descent (PGD) [17], have become standard tools for testing the robustness of neural networks, as discussed in more detail in Section 2.2.2. These attack algorithms aim to exploit the vulnerability of neural networks to adversarial examples and current research aims to improve the robustness of DNNs, by building defenses to such perturbations [21]–[24].

Popular defenses against adversarial attacks, such as AT [12], [17], aim to enhance the robustness of these models by explicitly exposing them to adversarial examples during training. Despite being effective, this approach is computationally expensive as it requires generating adversarial perturbations for each batch of training data and furthermore does not guarantee robustness against all types of attacks.

Margin-maximization techniques, while often similarly computationally expensive, have been explored as an alternative due to their ability to offer a better robustness-accuracy trade-off compared to methods like PGD adversarial training [9], [10]. We elaborate more on margin-maximization techniques in the following sections.

1.1.3 Margins

Margins in modern DNNs can be evaluated in various parts of the network, including input layers (input margins), hidden layers (hidden margins), or output layers (output margins). A typical margin (hidden or input) is defined as the shortest distance of a sample to the decision boundary, which is a hypersurface that separates data points belonging to different classes. While extensive research has been conducted on output margins [25]–[27], output margins simply correspond to the model's certainty for a given prediction. Instead, we are interested in the true distance of a sample to the separating hyperplane. Therefore, we will primarily focus on related work concerning input and hidden margins.

Margins have been well studied in models such as Support Vector Machines (SVMs) [28], [29] before the arrival of DNNs [30]. However, the decision boundaries of modern DNNs exhibit strong non-linearity and operate in high-dimensional spaces [31]. This poses challenges in calculating the exact margin and is considered intractable for modern DNNs [9], [32]–[35]. Due to the high dimensionality in DNNs we define an oversimplified example with two dimensions to explain the concept of a margin. Figure 1.2 shows this oversimplified visual representation, which consists of two classes with two features separated by a decision boundary. The arrows indicate the margin, i.e. the distance to the decision boundary, for various samples. Due to

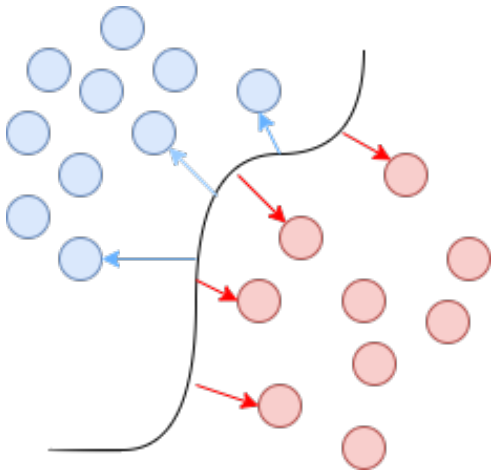


Figure 1.2: The samples of two classes (blue and red) separated by a decision boundary. Arrows indicate the margin for each sample.

the non-linearity of DNNs, we cannot calculate these margins directly; we can only approximate these margins. A first-order Taylor approximation is a well-recognized and often used

equation for approximating margins in modern DNNs. Adversarial attack algorithms, such as DeepFool [20], or the Fast Adaptive Boundary Attack (FAB) [19], which are important attack algorithms in this study, utilize this approximation at its core. Much research has been done to approximate these margins for modern DNNs and we will discuss this in detail in Section 2.4.1.

Robustness and margins

Margin-maximization techniques focus on increasing the margin – the minimum distance between a sample and its decision boundary – in the input space. Large margins in the input space have been shown to be correlated with improved adversarial robustness of DNNs [36]. Having a large margin ensures the correct classification of each class despite small input perturbations resulting in improved adversarial robustness. The intuition behind this is that if the margin is large, the adversary will have to generate larger perturbations to fool the model. Therefore, for perturbations smaller than the margin, the sample will still be correctly classified. Conversely, a smaller margin suggests that if the perturbation of the adversary is larger than the margin of that sample, it will be incorrectly classified. However research have shown that there exists a trade off: increased adversarial robustness often leads to a reduction in the generalization ability of DNNs. This is commonly referred to as the "robustness-accuracy trade-off". The reasons behind this phenomenon are not entirely understood and will be discussed further in Section 2.5.1.

Margins predicting generalization

As mentioned above, increased margins are linked to increased robustness. However, as evidenced by the existence of the robustness-accuracy trade-off, the link between margins and generalization is less clear. We now turn our attention to the application of margins in predicting generalization in DNNs.

Mouton et al. [33] consider whether large input margins are predictive of the test performance of pre-trained models. They find that this is not the case, and furthermore note that larger margins are slightly negatively correlated with generalization in some cases. To address this, Mouton et al., propose a novel concept called 'constrained margins'. This measure confines the margin measurement for each sample to highly specific directions in the input space. Specifically,

they measure the margins in the directions of the principal components of the training data, i.e. the directions which explain the greatest variance. Their results show a high correlation between large constrained margins and test set performance across various models and datasets. This indicates that constrained margins provide a much more reliable predictor of generalization performance than 'standard' input margins.

In summary, while large standard input margins are beneficial for robustness, they are generally not indicative of improved generalization. Conversely, when measured in high variance directions, large input margins are correlated with improved generalization performance.

1.1.4 Margin-aware training

To increase the adversarial robustness of DNNs, several authors have developed methods that attempt to increase the margins during training. Specifically, we focus on two methods that have been proposed to increase margins during model training: the work proposed by Elsayed et al. [9] and Xu et al. [10], both of which prioritize large margins to improve a model's adversarial robustness during training. Each introduces loss functions designed for maximizing margins in the input space to improve the adversarial robustness of DNNs and are described in more detail in Section 2.4.2.

The work by Xu et al. [10] and Elsayed et al. [9] primarily focused on increasing margins, utilizing innovative approaches to MM of DNNs. Elsayed et al. introduced a novel loss function based on the first-order Taylor approximation of the margin, which maximizes the margin at any layer, demonstrating improvements in adversarial robustness. Xu et al. built on this concept by identifying the vulnerability of data points with smaller margins and proposing Dynamics-Aware Robust Training (DyART), a technique that prioritizes increasing these smaller margins in the input space. DyART achieved near state-of-the-art performance on benchmarks such as RobustBench [37] in 2023 improving on recognized baselines such as AT [17], TRADES [38], MMA [36], GAIRAT [39], MAIL [40] and AWP [41] under various perturbation bounds.

Despite these gains in robustness, neither method leads to improved generalization on natural, non-adversarial data.

1.2 Problem statement

The generalization ability of DNNs remains an area of ongoing research. Standard input margins have demonstrated a connection to increased robustness, however they have been shown to *not* be predictive of generalization performance. In fact, increased input margins are often associated with decreased generalization performance. *Constrained input margins*, on the other hand, a novel technique for estimating margins, has shown potential in predicting a model's generalization ability. Current margin-maximization approaches focus primarily on standard input margins and constrained margins have not been investigated in this setting. It is not known whether constrained margin maximization can be used as an effective regularizer to improve generalization, what the optimal algorithmic approach for implementing such a regularizer would be, or which factors might influence its performance.

1.3 Research questions

Research questions to consider:

- Elsayed et al. and Xu et al. introduced novel margin maximization techniques for improving robustness. Which components from their work contribute mostly to robustness?
- Can these margin maximization methods be adapted to increase *constrained* margins?
- Does increasing constrained margins lead to better generalization performance and/or robustness?
- How do other regularization methods, such as weight decay, interact with constrained margins?

1.4 Objectives

To address our research questions on using constrained margins, as initially introduced by Mouton et al., as a potential regularizer for deep neural networks, we set the following objectives.

1. **Understanding the components of margin maximization techniques:** Our objective is to analyze specific components of the margin maximization approaches introduced by Elsayed et al. and Xu et al., identifying the elements that contribute most to improving adversarial robustness.

2. **Exploring large constrained margins for generalization and robustness:** Prior research on constrained margins has applied them post-training to predict generalization. Building on the insights from Elsayed et al. and Xu et al. on maximizing margins during training, we aim to adapt these approaches to incorporate constrained margins and evaluate its effect on generalization and robustness in deep neural networks.

By conducting this investigation, the study aims to not only improve the generalization of DNNs, but also contribute to a deeper understanding of DNNs' generalization capabilities.

1.5 Study scope

Given the problem statement above, to properly evaluate the generalization ability of a model, good baselines need to be established. This requires evaluating performance on well-known datasets and architectures.

- **Dataset:** We use the CIFAR-10 dataset which is widely utilised for image classification tasks, providing labeled data with multiple classes, making them valuable benchmarks for evaluating machine learning algorithms. This dataset is publicly available and widely adopted within the field of adversarial robustness. While the dataset is sometimes regarded as a 'toy problem', creating an adversarially robust CIFAR10 classifier is still an open and actively studied problem.

- **Architectures:** There are multiple architectures used within the field of adversarial robustness such as ResNet-18s, Wide Residual Networks (WRNs) and VGG-16 type models. All these architectures are deep learning architectures based on Convolutional Neural Networks (CNNs). However, Resnets and WRNs are computationally too expensive for our experiments and we will mainly focus on VGG-16 architectures.

1.6 Research methodology

This is an experimental study that investigates the effectiveness and viability of utilising constrained margins in deep neural network training to improve model generalization. The study utilizes approaches such as adapting large-margin training, and dynamics-aware robust training, and adversarial training. The experimental setup involves establishing strong baselines with well known architectures and datasets, followed by a comparative analysis to evaluate the impact of constrained margins on DNNs. Overall, the study aims to provide insights into the potential benefits of incorporating constrained margins in neural network training. The following steps are taken:

- **Experimental development:** Create a baseline model that serves as a reference for comparing the performance of the proposed constrained margin methods. Set up experiments to apply the constrained margin training methods to the selected neural network architectures and datasets identified in Section 1.5. Ensure proper Hyperparameter (HP) tuning and training procedures for each method to ensure fair comparisons.
- **Training with standard margins:** Determine which elements of the MM methods are most beneficial towards maximizing margins.
- **Training with constrained margins:** Explore the use of constrained margins during DNN training using the margin maximization approach from Elsayed et al.'s and Xu et al.'s methods.
- **Comparative analysis:** Conduct a thorough comparative analysis between the baseline model and the DNN models trained with constrained margins. Evaluate the models using relevant performance metrics and incrementally increase the complexity of experiments to assess the impact of different techniques on model performance.
- **Assessment of experimental findings:** After completing all experiments, analyze and interpret the results to determine the effectiveness of constrained margin training techniques in improving the generalization and performance of DNNs.

1.7 Dissertation overview

This dissertation has the following structure: Chapter 2 provides the necessary background information and discusses related work, setting the stage for the rest of the dissertation. In Chapter 3, we evaluate the components that lead to improved robustness when using margin maximization loss functions. Chapter 4, applies constrained margins to these components, assesing their effect on generalization. Finally, Chapter 5, concludes with a summary of the key findings of the investigation.

1.8 Publications

Sections of Chapter 3 of this dissertation has been published at SACAIR titled “Does simple trump complex? Comparing strategies for adversarial robustness in DNNs”.

Chapter 2

Background

Proverbs 16:9: "The heart of man plans his way, but the Lord establishes his steps."

2.1 Introduction

In this chapter, we provide the necessary background for this dissertation, focusing on key concepts related to robustness in DNNs. We begin by introducing adversarial examples and how the robustness of models can be evaluated (Section 2.2). We then discuss two main techniques for improving model robustness during training, namely those that directly utilize adversarial examples (Section 2.3) and margin maximization (Section 2.4), which utilize a loss function to encourage larger margins during training. Finally, we discuss how robustness characteristics relate to performance on unseen inputs (Section 2.5).

2.2 Adversarial examples and robustness evaluation

Adversarial examples pose a considerable challenge to the robustness of DNNs. The vulnerability of DNNs to these adversarial examples has sparked significant research on methods that can mitigate their effect. In the next Sections, we first define adversarial examples before discussing

adversarial attacks and defenses against these attacks, respectively.

2.2.1 Adversarial examples

Adversarial examples are specially crafted data samples that appear normal to humans, however, consistently cause misclassification when evaluated by neural networks. These examples are created using adversarial attack algorithms to fool a classifier by slightly perturbing clean examples. These attack algorithms aim to calculate the minimal perturbation that will cause misclassification. Consider a deep classifier f parameterized by θ, with inputs $\boldsymbol{x} \in X$ and corresponding labels $y \in Y = \{1, 2, \ldots, n\}$, that generates a prediction score to classify the input vector $\boldsymbol{x}$ to class y. The minimal adversarial perturbation Δ, describes the smallest distance that changes a sample's classification and is given by:

$$\Delta(\boldsymbol{x}; f_\theta) := \min_{\boldsymbol{\delta}} \|\boldsymbol{\delta}\|_p \text{ subject to } \arg\max_{\boldsymbol{\delta}} f_\theta(\boldsymbol{x} + \boldsymbol{\delta}) \neq \arg\max f_\theta(\boldsymbol{x}) \text{ and } \|\boldsymbol{\delta}\|_p \leq \epsilon. \tag{2.1}$$

where $\boldsymbol{x}$ represents the original input image (example is Figure 2.1), while $\boldsymbol{\delta}$ is the calculated perturbation (examples are Figures 2.2a and 2.2b). We will define the adversarial example to be $\hat{\boldsymbol{x}}$ where $\hat{\boldsymbol{x}} = \boldsymbol{x} + \delta$. The term $\|\cdot\|_p$ denotes the p-norm, which measures the magnitude of the perturbation. The function $f_\theta(\boldsymbol{x})$ is the model's prediction, and $f_\theta(\hat{\boldsymbol{x}})$ represents the model's prediction for the perturbed input. In practice, a perturbation bound, ϵ, a measure of the intensity of a perturbation, is often specified to limit the magnitude of the allowed perturbation, thereby controlling the strength of the adversary. The higher this bound, the more difficult it becomes to interpret the actual class of a given image. To demonstrate this concept, we provide

Figure 2.1: The original input image $\boldsymbol{x}$ classified as "whale" [20]

an example from Moosavi et al. [20], comparing two prominent attack methods: DeepFool [20] and Fast Gradient Sign Method (FGSM) [12]. Figure 2.1, is the original input and Figure

2.2 illustrates both the perturbations ($\boldsymbol{\delta}$) and their resulting adversarial examples ($\hat{\boldsymbol{x}}$) for each method[1]. Notice that these adversarial perturbations are virtually imperceptible to the human eye, yet can easily fool DNNs to misclassify the example with high confidence.

As previously mentioned (recall Section 1.1.2), DNNs that are able to withstand such adversarial attacks are known as 'adversarially robust' models. Identifying such minimal perturbations is crucial for measuring model robustness, as the smaller the perturbation required to induce misclassification, the less robust the model. We elaborate on this further in the following Section.

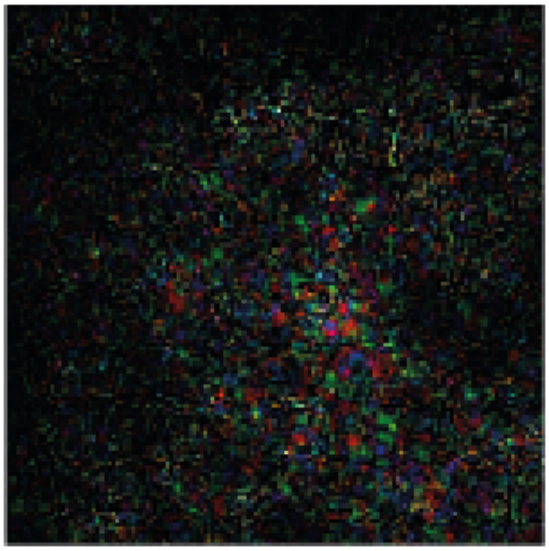
2.2.2 Adversarial attacks for evaluating robustness

Various attack algorithms have been proposed for generating stronger adversarial examples to fool modern DNNs, examples of which are PGD [17], FAB [19], DeepFool [20], Fast Gradient Sign Method (FGSM) [12], and the Carlini and Wagner attack (CW) [42]. At the same time, techniques have been developed to leverage these adversarial attacks to enhance the robustness of DNNs against such perturbations [10], [17], [36], [38]. These attacks can be used to generate adversarial examples in which you can vary the strength of the perturbation ϵ during training to improve the robustness of models, but can also be used to evaluate the robustness of DNNs to these attacks under various perturbation bounds.

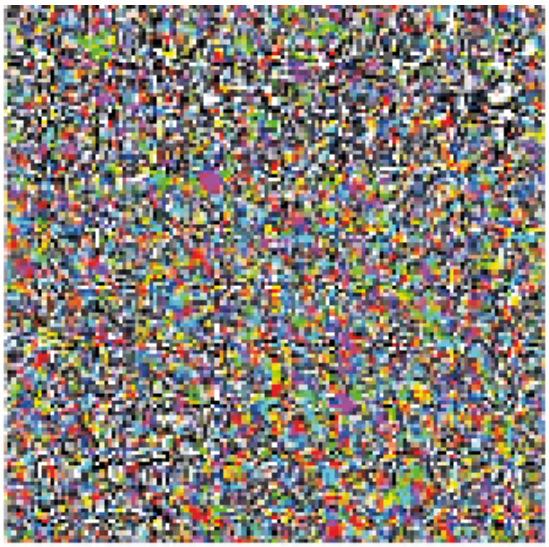
We aim to utilize adversarial attacks to evaluate the robustness of models. The robustness of model are evaluated by generating adversarial examples (usually using unseen data, i.e. the test set) with the above mentioned attacks and then measuring the model's accuracy on this set. The classification accuracy of DNNs on these adversarial examples provides insight into the model's robust accuracy: a higher prediction accuracy on these adversarial examples indicates greater robustness.

Robustness is generally evaluated with respect to a specific perturbation bound, i.e. a maximum perturbation size. Common choices are $\frac{4}{255}$ and $\frac{8}{255}$ for the l_∞ norm, although similar sizes adjusted for the l_2 norm are also sometimes used [37]. The reason for these choices of perturbation bound is that when larger values are used the samples can often no longer be considered "adversarial", i.e. a perturbed dog "misclassified" as a cat might visually appear like a cat.

[1]The magnitude of the perturbations in Figures 2.2a and 2.2b, is increased for visualization purposes



(a) Perturbation $\boldsymbol{\delta}$ calculated using DeepFool [20]



(b) Perturbation $\boldsymbol{\delta}$ calculated using FGSM [12]

(c) Adversarial example $\hat{\boldsymbol{x}}$ using DeepFool

(d) Adversarial example $\hat{\boldsymbol{x}}$ using FGSM

Figure 2.2: Adapted from [20]. Comparison of perturbations and adversarial examples generated using DeepFool and FGSM methods. Top row shows the calculated perturbations $\boldsymbol{\delta}$, while bottom row shows the resulting adversarial examples $\hat{\boldsymbol{x}}$.

When evaluating robustness, modern evaluations use a combination of attacks to ensure that the model is not only robust to a single specific attack. Therefore, it is important to measure the robustness of the models using current State-of-the-Art (SOTA) approaches that combine multiple attacks for proper evaluation as one can overfit to a single attack. For example, a model trained with AT (explained in the following Section) might be robust against PGD attacks, it might not generalize well to other, potentially unknown, attacks [43].

AutoAttack [18] has become the standard benchmark for evaluating the adversarial robustness of modern DNNs. AutoAttack consists of an ensemble of four attacks. These four attacks include two variations of the standard PGD attack developed by Madry et al. [17]: an enhanced version of PGD known as Auto Projected Gradient Descent (APGD) and the Targeted Auto Projected Gradient Descent (APGD-t) [18]. Additionally, it includes the targeted FAB [19] attack, and the black-box Square attack [44]. We will elaborate on the attack methods relevant to our study in the following Sections.

Croce et al. established a framework, RobustBench [37], with the intention of being a standardized benchmark to measure the robust accuracy of DNNs, and to compare findings and results. The benchmark measures and compares the robustness of user submitted models on the CIFAR10/100 datasets using AutoAttack with various perturbation bounds. The most commonly applied bound in current literature is the l_∞ perturbation bound of $\frac{8}{255}$ which has become a standard benchmark on platforms like RobustBench [37].

2.3 Adversarial training

Adversarial Training (AT) [17] is the standard approach to improving the robustness of classifiers to defend against adversarial attacks. This method is seen as a form of (online) data augmentation, where models are trained on adversarial examples generated during each training iteration. Here we provide a brief explanation of this method.

Adversarial training is formulated as a saddle point problem (min-max optimization problem); where we first maximize the loss by generating strong adversarial examples (the attack), then minimize the loss by training the model on these adversarial examples (the defense). This inner

maximization and outer minimization problem is formalized in Equation 2.2.

$$\min_{\theta} \max_{\boldsymbol{\delta} \in \mathcal{S}} \mathcal{L}(f_\theta(\boldsymbol{x} + \boldsymbol{\delta}), y) \tag{2.2}$$

The inner maximization function aims to find the perturbation $\boldsymbol{\delta}$ within the set of allowed perturbations ($\mathcal{S}$) for a given sample $\boldsymbol{x}$ that achieves the highest loss ($\mathcal{L}$) w.r.t the label y. This is equivalent to an adversarial attack algorithm that generates the strongest possible perturbation within a specified perturbation bound. The outer minimization then aims to find model parameters that minimize the maximum loss achieved by these adversarial perturbations, making the model robust against such worst-case attacks.

The set $\mathcal{S}$, as defined in Equation 2.3, defines the set of all perturbations $\boldsymbol{\delta}$ that the adversary is allowed to apply to a given input $\boldsymbol{x}$. This is generally defined as a norm constraint, as to enforce a specified maximum perturbation bound, which ensures that the perturbations are within a certain distance from the original input. Formally,

$$\mathcal{S} = \{\boldsymbol{\delta} \in X : \|\boldsymbol{\delta}\|_p \leq \epsilon\} \tag{2.3}$$

where $\boldsymbol{\delta}$ represents the adversarial perturbation applied to the input $\boldsymbol{x}$. The term $\|\cdot\|_p$ denotes the p-norm, often chosen as the ℓ_∞ norm. The parameter ϵ is the maximum perturbation size which limits how much the adversary can change the input. Thus S, represents the set of possible perturbations around $\boldsymbol{x}$, keeping $\boldsymbol{x} + \boldsymbol{\delta}$ close to the original input.

Algorithm 1 shows the high-level steps of adversarial training. In short, the model is trained as normally done with gradient descent, except that the loss is minimized on adversarial examples that are generated before each update, instead of on natural examples. We now consider two methods that are commonly used to generate such adversarial examples, i.e. bounded adversarial attack methods.

2.3.1 Projected Gradient Decent

Madry et al. solved the above-mentioned saddle point problem using Projected Gradient Descent (PGD) [17]. PGD is a first-order attack, which means that it can effectively maximize the loss of the inner maximization function using gradient information, i.e. only uses first-order derivatives. Thus, PGD knows how the output changes to small changes in the input x. PGD starts from a random perturbation (point in the epsilon ball as shown in Equation 2.3) around each natural

Algorithm 1 Adversarial Training

Require: Training data $\mathcal{D}$, model f_θ, loss function $\mathcal{L}$, perturbation bound ϵ, number of epochs E

for epoch $= 1$ **to** E **do**
  for $(\boldsymbol{x}, y)$ in $\mathcal{D}$ **do**
    // Inner maximization (generate adversarial example)
    $\boldsymbol{\delta} \leftarrow \arg\max_{\delta \in \mathcal{S}} \mathcal{L}(f_\theta(\boldsymbol{x} + \boldsymbol{\delta}), y)$
    $\hat{\boldsymbol{x}} \leftarrow \boldsymbol{x} + \boldsymbol{\delta}$
    // Outer minimization (update model parameters)
    Update θ with gradient descent to minimize $\mathcal{L}(f_\theta(\hat{\boldsymbol{x}}), y)$
  end for
end for
return Robust model f_θ

example in the set $\mathcal{S}$. For each training batch, PGD performs the following steps to solve the inner maximization problem over a fixed number of iterations (e.g., 7 or 20 iterations, depending on the desired strength of the adversarial example):

1. **Initialize**: Start from a randomly chosen point within the ϵ-ball surrounding the original input $\boldsymbol{x}$.
2. **Calculate Gradient Step**: Determine the sign of the gradient of the loss function, $\nabla_x \mathcal{L}(f_\theta(\boldsymbol{x}))$, which indicates the direction that maximizes the loss.
3. **Update Perturbation**: Adjust the input based on the gradient direction with a step size η, which controls the magnitude of the change at each iteration.
4. **Project if Necessary**: Apply the projection operator $\text{proj}_{\hat{\boldsymbol{x}}}$, to ensure that the perturbed example remains within the ϵ-ball around $\boldsymbol{x}$. This projection is an optimization process finding the closest point to the updated perturbation that is within the e-ball.

The mathematical formulation for PGD is shown in Equation 2.4:

$$\hat{\boldsymbol{x}}_{PGD}^{i+1} = \text{proj}_{\hat{\boldsymbol{x}}} \left(\hat{\boldsymbol{x}}_i + \eta \, \text{sign} \left(\nabla_x \mathcal{L}(f_\theta(\boldsymbol{x})) \right) \right). \tag{2.4}$$

Here, $\hat{\boldsymbol{x}}_{PGD}^{i+1}$ represents the adversarial example at the next PGD iteration, while $\hat{\boldsymbol{x}}_i$ is the example at the current PGD iteration. The projection operator $\text{proj}_{\hat{\boldsymbol{x}}}$ ensures that the adversarial

examples remain within the set $\hat{\boldsymbol{x}}$, where $\hat{\boldsymbol{x}} = \{\boldsymbol{x} + \boldsymbol{\delta} : \boldsymbol{\delta} \in S\}$.

2.3.2 Auto Projected Gradient Decent

The idea behind APGD is to address the limitations of using a fixed step size, η, which is a highly influential hyperparameter for PGD to perform well. Utilizing a fixed step size does not guarantee that the algorithm consistently reaches higher loss regions in the loss plane. This essentially means that PGD does not reliably find the strongest adversarial examples. Croce et al. [18] mitigate this by adjusting the step size based on the trend of the optimization i.e adjusting the η, based on how the loss changes. This means that if the loss increases for the current step size at iteration i, then it is sufficient, but if not, then it needs to be reduced during the next iteration to possibly establish a stronger adversarial example. This requires four components:

- **Define a large initial step size** to explore the set S and find good initial points.
- **Define an update step**, defined in Equation 2.5, to calculate the adversarial example by considering important directions to maximize the loss. This is achieved using PGD, as defined in Equation 2.4, to calculate the adversarial example for the next iteration. A weight μ (a hyperparameter defined between 0 and 1, set to 0.75 in their paper) is applied to balance two components:

  1. The influence of the current gradient information on the update, establishing the important direction of the current update $\eta(\hat{\boldsymbol{x}}^{i+1}_{PGD} - \hat{\boldsymbol{x}}_i)$.
  2. The influence of the gradient information from the previous update, ensuring consistency in the search direction by considering past directions to maintain "momentum" in these important directions $(1-\eta)(\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_{i-1})$.

  Thus, the update step leverages both the current gradient and momentum from past updates to efficiently optimize the loss function.

$$\hat{\boldsymbol{x}}_{i+1} = \text{proj}_{\hat{\boldsymbol{x}}} \left(\hat{\boldsymbol{x}}_i + \eta(\hat{\boldsymbol{x}}^{i+1}_{PGD} - \hat{\boldsymbol{x}}_i) + (1-\eta)(\hat{\boldsymbol{x}}_i - \hat{\boldsymbol{x}}_{i-1}) \right), \tag{2.5}$$

- **Identify fixed checkpoints** at which to reduce the step size. These checkpoints are defined based on the total number of iterations and their positioning are determined based

on a set of parameters called p_j (period length which can be between 0 and 1) to determine where the checkpoints will be. The amount of checkpoints cannot be more than the total number of iterations. Therefore, the checkpoints divide the number of iterations into smaller segments.

- **Establish specific conditions at the checkpoints** to ensure effective progress in the optimization process. Croce et al. count the number of successful updates since the last checkpoint and define a hyperparameter ρ which represents a step size reduction threshold (set to 0.75 in the paper). If the loss increases in at least 75% of the steps, the step size remains unchanged; otherwise, it is reduced.

They therefore explore the set S and reduce the step size at predefined checkpoints when necessary, allowing more effective exploration of the set and the generation of stronger adversarial examples compared to standard PGD.

2.4 Margin maximization

When employing adversarial training, it is necessary to distinguish between techniques that directly utilize adversarial examples, as discussed in Section 2.3 and those that focus on margin maximization (MM). Prominent MM approaches include TRADES [38], work by Elsayed et al. [9], DyART [10], and MMA [36].

These approaches leverage the concept of a margin, denoted as R from here onward, defined as the minimum distance of a sample to its nearest decision boundary. Large margins in the input space naturally imply better robustness, as a larger perturbation is required for the sample to be misclassified. Therefore, finding a way to increase these margins will allow us to build models that perform better against stronger attacks. The intuition behind MM approaches is similar to the min-max optimization problem in the sense that we aim to minimize a loss function aimed at maximizing the minimum margin.

We now formalize a margin measured in the input space of a DNN. Consider a classifier f_θ and an input sample $\boldsymbol{x}$. We can define the prediction of the classifier for the sample as the class y corresponding to the maximum output logit $z_\theta^y(\boldsymbol{x})$ given by $f_\theta(\boldsymbol{x}) = \arg\max_{y \in Y} z_\theta^y(\boldsymbol{x})$. To define the decision boundary, we introduce ϕ, which represents the output margin – the

difference between the logit value for the true class y and the highest logit value among all other classes $y' \neq y$. Formally,

$$\phi_\theta(\boldsymbol{x}) = z_\theta^y(\boldsymbol{x}) - \max_{y' \neq y} z_\theta^{y'}(\boldsymbol{x}) \tag{2.6}$$

A sample $\boldsymbol{x}$ is deemed to be correctly classified when $\phi_\theta^y(\boldsymbol{x}) > 0$ and incorrectly classified otherwise. The decision boundary exists where there is a tie, that is, $\phi_\theta^y(\boldsymbol{x}) = 0$. We can then define the margin, R_θ, as the distance of a sample $\boldsymbol{x}$ to the nearest sample $\hat{\boldsymbol{x}}$ on the decision boundary:

$$R_\theta(\boldsymbol{x}) = \min_{\hat{\boldsymbol{x}}} \|\hat{\boldsymbol{x}} - \boldsymbol{x}\|_p \quad \text{s.t.} \quad \phi_\theta^y(\hat{\boldsymbol{x}}) = 0 \tag{2.7}$$

Here $\|.\|_p$ is some chosen l_p norm. We typically use the l_∞ norm as is convention [10], [17]. Since we cannot directly calculate this margin due to non linearity, we need to estimate this margin value.

2.4.1 Estimating margins

As mentioned in Chapter 1, the decision boundaries of contemporary DNNs exhibit strong non-linearity and operate in high-dimensional spaces, which presents challenges when trying to calculate the exact margin. Consequently, we rely on measures that estimate the margin. However, bounded attacks such as PGD (Section 2.3.1) or APGD (Section 2.3.2) are not suited for this purpose, as the perturbation size is predetermined by the chosen perturbation bound. These attacks focus on finding an adversarial example within a predefined set $\mathcal{S}$, rather than directly estimating the margin. In contrast, unbounded attack algorithms like FAB, which utilize the first-order Taylor approximation, allow for more flexible perturbations and can be used to provide a more accurate estimation of the margin for modern DNNs, as they directly consider the decision boundary and generate the smallest possible perturbation.

Taylor approximation

Estimating margins requires approximating the distance of a point to the decision boundary for modern DNNs. A first-order Taylor approximation is a well-recognized and popular equation for approximating margins in modern DNNs [9], [19], [20]. Let us consider the commonly used first-order Taylor approximation of the margin, as shown in Equation 2.8. The margin $R(\boldsymbol{x})$ for

point $\boldsymbol{x}$ is given by

$$R_\theta(\boldsymbol{x}) = \frac{z_\theta^y(\boldsymbol{x}) - z_\theta^{y'}(\boldsymbol{x})}{\left\| \nabla_{\boldsymbol{x}} z_\theta^y(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} z_\theta^{y'}(\boldsymbol{x}) \right\|_q}, \tag{2.8}$$

where the numerator in Equation 2.8 is given by the difference between the output logits of the true class, y, and some other class, y' (i.e. the same as Equation 2.6). The denominator computes the difference between the gradients $\nabla_{\boldsymbol{x}}$ of $z_\theta^y(\boldsymbol{x})$ and $z_\theta^{y'}(\boldsymbol{x})$ with regard to the input $\boldsymbol{x}$, and then calculates the norm $\|.\|_q$. Calculating the gradient terms requires calculating the Jacobian matrix which gives the partial derivatives of the logits with respect to the input $\boldsymbol{x}$. Here, $\|.\|_q$ denotes the dual norm of $\|.\|_p$ where $\frac{1}{q} + \frac{1}{p} = 1$. Specifically, when $p = \infty$, then $q = 1$ [9]. Furthermore, it should be clear that if the sample is incorrectly classified, the numerator will be negative, and the resulting margin will also be a negative value.

In this work, we will often rely on this first-order Taylor approximation to estimate margins. In addition, we also consider the FAB attack, which uses this approximation.

Fast Adaptive Boundary attack

FAB is an iterative adversarial attack that aims to find the smallest possible perturbation (adversarial example close to the original input) by adaptively exploring the decision boundary[19]. The high-level process of FAB requires starting from an original input x, and incrementally moving it in a direction that reduces the model's confidence. With each iteration, Croce et al. [19] adapt the search direction based on the local decision boundary. FAB is conducted in three phases to achieve this: initialization, the core iteration process (bot described as Algorithm 2), and final refinement (Algorithm 3).

Algorithm 2, assists in explaining the core functioning of FAB in its simplest form. FAB iteratively applies a first-order Taylor approximation to find the class s (which is not the original class c), as shown in Equation 2.9

$$s = \arg\min R_\theta(\boldsymbol{x}_{(i)}) \tag{2.9}$$

with the closest decision boundary to a sample $\boldsymbol{x}_{(i)}$[2]. FAB then performs a projection step which projects a sample onto this decision boundary and applies a constraint around this sample (ensures that the perturbed example stays within the defined input region).

[2]I explicitly do not refer to $\boldsymbol{x}_{(i)}$ as $\hat{\boldsymbol{x}}_{(i)}$ as FAB provides the adversarial example $\hat{\boldsymbol{x}}$ at output. However, the projection of this point $\boldsymbol{x}_{(i)}$(which is a sample on the decision boundary) will be noted as $\hat{\boldsymbol{x}}$

Algorithm 2 FAB Attack Core Process

Input: Original sample $\boldsymbol{x}_{orig}$, Target classifier f
Output: Adversarial example with minimal perturbation
for each restart **do** ▷ Initialization Phase: Sets up starting points
    if first restart **then**
        Initialize using original point
    else
        Sample a starting point within a controlled radius around the original sample. The size of the radius depends on the current best perturbation.
    end if
    for each iteration i **do** ▷ Core Iterations
        Estimate nearest decision boundary from current sample using Taylor approximation
        Project current point onto this boundary (with perturbation constraint)
        Project original point onto this boundary (with perturbation constraint)
        Calculate step size using projection distances
        Generate next candidate using convex combination between the current and original adversarial example applying the step size.
        if candidate not classified as original class **then**
            if perturbation smaller than best found **then**
                Return adversarial example
                Update minimum perturbation found
            else
                Apply linear step towards the original (size of step controlled by HP β)
                Return next candidate example
            end if
        end if
    end for
end for

To ensure that the perturbed example stays within the defined input region, Croce et al. [19] defines a ϵ-ball constraint, C, around $\boldsymbol{x}_{(i)}$ and $\boldsymbol{x}_{orig}$ to ensure that their perturbations remain within the allowable range of pixel values. Specifically, for a given input $\boldsymbol{x} \in X$, the constraint is expressed as:

$$C = \{\hat{\boldsymbol{x}} \in X : l_i \leq \hat{\boldsymbol{x}}_{(i)} \leq u_i\}, \tag{2.10}$$

where l_i and u_i are the lower and upper bounds for the i-th component of $\hat{\boldsymbol{x}}$, which typically represent the allowable range of pixel values. These bounds ensure that the perturbation does not cause any input features to exceed these defined values. Other techniques typically clip $\hat{\boldsymbol{x}}$ to ensure it remains within the bound constraints.

The projection step with norm p ($\text{proj}_p(\boldsymbol{x}, \phi, C)$ where $\boldsymbol{x}$ is the input to be projected) is then used to project both the current sample $\boldsymbol{x}_{(i)}$ and the original sample $\boldsymbol{x}_{orig}$ onto the decision boundary ϕ^s_θ which provides the perturbation δ. Calculating the norm of the perturbation $\|\delta\|_p$ provides us with the distance between the sample and its projection. If a point $\hat{\boldsymbol{x}}_i$ violates the constraint, FAB adjusts $\hat{\boldsymbol{x}}_i$ such that it lies within the defined limits of the constraint. This guarantees that every perturbed input remains both adversarial and within the valid input domain.

We then utilize the norms of the perturbations to calculate the step size dynamically based on the relationship between the perturbation of the current sample $\boldsymbol{\delta^{(i)}}$ and the original sample $\boldsymbol{\delta^{(orig)}}$ as shown in Equation 2.11.

$$\mu = \min\left(\frac{\|\boldsymbol{\delta}^{(i)}\|_p}{\|\boldsymbol{\delta}^{(i)}\|_p + \|\boldsymbol{\delta}^{(i)}_{orig}\|_p}, \mu_{\max}\right) \in [0, 1], \tag{2.11}$$

where $\|\boldsymbol{\delta}^{(i)}\|_p$ is the distance of $\boldsymbol{x}_{(i)}$ to the decision boundary for class s and $\|\boldsymbol{\delta}^{(i)}\|_p$ is the distance of $\boldsymbol{x}_{orig}$ to the decision boundary of class s. The ratio therefore adjusts the weight between $\boldsymbol{x}_{(i)}$ and $\boldsymbol{x}_{orig}$ based on their distances to the decision boundary.

FAB also includes an extrapolation step to increase the likelihood of finding a successful adversarial example. This final step to find the next candidate is shown in Equation 2.12:

$$\hat{\boldsymbol{x}}_{i+1} = \text{proj}_C\left((1-\mu)(\boldsymbol{x}_{(i)} + \eta\boldsymbol{\delta}^{(i)}) + \mu(\boldsymbol{x}_{orig} + \eta\boldsymbol{\delta}^{(i)}_{orig})\right), \tag{2.12}$$

where η is an extrapolation parameter to explore directions beyond the convex combination of the projections and which respect the constraint C. This final step ensures that $\hat{\boldsymbol{x}}_{i+1}$ is not

only on the decision boundary but also incorporates an extrapolation factor to explore more adversarial directions.

FAB then checks whether $\hat{\boldsymbol{x}}_{i+1}$ is misclassified (that is, no longer classified as the original class) and compares the size of the perturbation for this sample, $\|\hat{\boldsymbol{x}}_{i+1} - \boldsymbol{x}_{\text{orig}}\|_p$, with the size of the previous perturbation found. If the new perturbation is not smaller, Croce et al. apply a backward step, as shown in Equation 2.13

$$\boldsymbol{x}_{i+1} = (1-\beta)\boldsymbol{x}_{\text{orig}} + \beta\boldsymbol{x}_{(i+1)}, \quad \beta \in (0,1), \tag{2.13}$$

where they define $\beta = 0.9$, such that $\boldsymbol{x}_{(i+1)}$ moves 90% closer towards the original sample. This allows the model to essentially "try again" and find another adversarial example close to $\boldsymbol{x}_{orig}$ with a smaller perturbation than previous adversarial examples.

After completing all restarts and iterations and returning the adversarial example $\hat{\boldsymbol{x}}_{out}$, Croce et al. apply a final step to improve the quality of the adversarial example, as the points close to the decision boundary can be slightly off because the Taylor approximation is not exact. They therefore apply a refinement step as shown in Algorithm 3. The goal of Algorithm 3 is to refine

Algorithm 3 FAB Final Refinement

Input: Best adversarial example $\hat{\boldsymbol{x}}_{\text{out}}$, Original sample $\boldsymbol{x}_{\text{orig}}$
Definitions: $f_s(\cdot)$: score for target class s, $f_c(\cdot)$: score for original class c
for 3 refinement steps **do**
    Calculate $\boldsymbol{x}_{\text{temp}} = \hat{\boldsymbol{x}}_{\text{out}} - \frac{(f_s(\hat{\boldsymbol{x}}_{\text{out}}) - f_c(\hat{\boldsymbol{x}}_{\text{out}}))(\hat{\boldsymbol{x}}_{\text{out}} - \boldsymbol{x}_{\text{orig}})}{f_s(\hat{\boldsymbol{x}}_{\text{out}}) - f_c(\hat{\boldsymbol{x}}_{\text{out}}) + f_s(\boldsymbol{x}_{\text{orig}}) - f_c(\boldsymbol{x}_{\text{orig}})}$
    if $f_s(\boldsymbol{x}_{\text{temp}}) - f_c(\boldsymbol{x}_{\text{temp}}) > 0$ **then**
        Update $\hat{\boldsymbol{x}}_{\text{out}} \leftarrow \boldsymbol{x}_{\text{temp}}$ ▷ $\boldsymbol{x}_{\text{temp}}$ classified as target class s
    else
        Update $\boldsymbol{x}_{\text{orig}} \leftarrow \boldsymbol{x}_{\text{temp}}$ ▷ $\boldsymbol{x}_{\text{temp}}$ classified as original class c
    end if
end for
Ensure: $f_s(\hat{\boldsymbol{x}}_{\text{out}}) - f_c(\hat{\boldsymbol{x}}_{\text{out}}) > 0$ ▷ Final $\hat{\boldsymbol{x}}_{\text{out}}$ is classified as s
Ensure: $f_s(\boldsymbol{x}_{\text{orig}}) - f_c(\boldsymbol{x}_{\text{orig}}) < 0$ ▷ Final $\boldsymbol{x}_{\text{orig}}$ is classified as c
return Final adversarial example $\hat{\boldsymbol{x}}_{\text{out}}$

the adversarial example $\hat{\boldsymbol{x}}_{out}$ through iterative updates, while ensuring that it remains classified as the target s.

2.4.2 Margin maximization loss functions

The core idea of MM is to utilize a loss function that combines traditional cross-entropy loss with a MM term. This hybrid loss function, as shown in Equation 2.14, is designed to increase margins in the input space, thereby improving the model's adversarial robustness.

$$\mathcal{L} = \mathcal{L}_{\text{CE}}(y, \hat{y}) + \lambda \cdot \mathcal{L}_{\text{MM}}(\gamma, R) \tag{2.14}$$

Here, $\mathcal{L}_{\text{CE}}$ is the cross-entropy loss function, y is the true label of a sample, and $\hat{y}$ the model's corresponding prediction. The $\mathcal{L}_{\text{MM}}$ term, typically includes a user-defined hyperparameter, γ, which represents the minimum margin that should be maintained, making the model less susceptible to perturbations of an input. In addition, a hyperparameter λ is incorporated to weigh the margin maximization component of the loss function.

We will primarily focus on the MM loss functions proposed by Elsayed et al. and Xu et al., respectively. Both methods utilize hybrid loss functions that aim to maximize margins in the input space whilst minimizing the loss with respect to the inputs for correct classification [9], [10].

Margin maximization loss function by Elsayed et al.

The L_{MM} function introduced by Elsayed et al., for a batch $\mathcal{B}$ of size N, is defined as:

$$\mathcal{L}_{MM}(\gamma, R) = \frac{1}{N} \sum_{i \in \mathcal{B}} \mathscr{A}_{y' \neq y} \max\{0, \gamma - R_\theta(\boldsymbol{x}_i)\} \tag{2.15}$$

Here $\mathscr{A}_{y' \neq y}$ is some aggregation operator, and they experiment with choosing this as either the *max* operator or the *sum* operator. For each training sample $\boldsymbol{x}$ (we drop the index for clarity) and corresponding label y, the individual loss is computed for each class where $y' \neq y$. The margin of a sample $\boldsymbol{x}$ is therefore calculated to the decision boundary of every other class, $y' \neq y$. These individual losses are then aggregated using the max or sum operator.

Intuitively, this loss function aims to ensure that all samples have at least a margin of γ (the minimum margin) to the nearest decision boundary. Additionally, it penalizes samples with smaller margins more than those with larger margins, and also accommodates both correctly and incorrectly classified samples. Consider a correctly classified sample with a margin smaller than the minimum margin. The margin loss for this sample will then be equal to $\gamma - R$. However,

if the samples has a margin larger than the minimum margin, it will simply be assigned a loss of zero. On the other hand, if a sample is incorrectly classified, the margin (R) will be negative and the loss for that sample will be adjusted to account for both the distance the decision boundary must move for it to be correctly classified and have a positive margin of at least γ.

The behavior of this loss function is perhaps made more clear if one considers how Elsayed et al. measure the margin for each sample. While some authors measure the exact distance to an adversarial example, Elsayed et al. opt to rather approximate this distance. This greatly reduces the computational cost, as calculating adversarial examples for each batch of training data can be highly expensive. Specifically, they utilize a first order Taylor approximation, as shown in Equation 2.8, to approximate the margin.

Since there is a derivative in the MM loss function, it requires calculating the second-order gradient when calculating the gradient of the loss w.r.t. the model parameters for gradient descent, which is computationally too expensive. They therefore treat the denominator in Equation 2.8 as a constant during back-propagation.

Finally, Elsayed et al. also experiment with extending this loss function to the margin of each sample as measured at the hidden layers. Specifically, they replace the input $\boldsymbol{x}$ with its hidden representation h_l at a given layer l. Their goal is to enforce a large margin based on the data representation at each specific layer. We do not consider the maximization of margins measured at the hidden layers in this work.

Margin maximization loss function by Xu et al.

Similar to the approach by Elsayed et al., Xu et al. also designed a MM loss function. They identified an issue with other robust training methods where increasing the margin for one data-point might decrease the margin of another data point. They called this 'conflicting dynamics'. To mitigate this issue, they introduced DyART, an MM loss function designed to prioritize increasing smaller margins [10]. This ensures that all samples have a margin of at least γ. Their MM function is defined as follows:

$$MM(R) = \begin{cases} \frac{1}{\alpha}\exp(-\alpha R), & R < \gamma \\ 0, & \text{otherwise} \end{cases} \tag{2.16}$$

This function assigns an exponential loss to samples with a margin smaller than γ; otherwise, the loss is zero. The hyperparameter α smooths this exponential function. However, unlike the MM function of Elsayed et al., the loss function is only applied when $\phi_\theta^y(\boldsymbol{x}) > 0$, meaning incorrectly classified samples are ignored. Since most samples are still incorrectly classified at the start of training, a burn-in period or training warmup is recommended, where the model is trained naturally using cross-entropy for a certain number of epochs before applying the MM loss function. The loss for a training batch $\mathcal{B}$ of size N, with correctly classified samples $\mathcal{B}_\theta^+$ of size m, is defined as:

$$\mathcal{L}_{MM}(R) = \frac{1}{m} \sum_{i \in \mathcal{B}_\theta^+} MM(R_\theta(\boldsymbol{x}_i)) \tag{2.17}$$

Xu et al. also use a more accurate estimation of the margin for each sample than Elsayed et al. Instead of relying on the first-order Taylor approximation, DyART utilizes FAB [19] to find the closest boundary point $\hat{\boldsymbol{x}}$ from sample $\boldsymbol{x}$. The margin is then given by $||x - \hat{\boldsymbol{x}}||_\infty$. Recall from Section 2.4.1 that FAB is much more computationally expensive. In addition, if the margin is calculated to each class it needs to calculate the Jacobian at each iteration between the true class and every other class. Therefore, finding the closest point on the decision boundary becomes computationally prohibitive, especially for datasets with a large number of classes, such as CIFAR-100. FAB also requires multiple iterations to produce high-quality adversarial examples.

To lessen this computational burden, Xu et al. do not calculate the margin to each non-label class for each samples. Instead, they introduced an alternative method for calculating the margin, known as the soft margin, $R_\theta^{\text{soft}}(\boldsymbol{x})$, which is the minimum distance of sample to a point on the 'soft decision boundary'. The soft decision boundary is an approximation of the exact, or 'true', decision boundary. More specifically, the soft decision boundary is given by the points where $\phi_\theta^y(\boldsymbol{x}; \beta) = 0$ for a hyperparameter β, and $\phi_\theta^y(\boldsymbol{x}; \beta)$ is defined as:

$$\phi_\theta^y(\boldsymbol{x}; \beta) = z_\theta^y(\boldsymbol{x}) - \frac{1}{\beta} \log \sum_{y' \neq y} \exp\left(\beta z_\theta^{y'}(\boldsymbol{x})\right) \tag{2.18}$$

Therefore, instead of calculating the distance to the exact decision boundary, which requires evaluating the logits for all possible pairs of classes, as defined in Equation 2.6, the Log-Sum-Exp function of Equation 2.18, gives a weighted average which emphasizes the largest logits without needing to compare each logit. This reduces the number of operations required as it avoids the need to compute the margin to every possible pair of logits. This soft decision

boundary is controlled by a hyperparameter, β, and the higher this value, the closer the soft decision boundary represents the exact decision boundary.

Finally, recall that Elsayed et al. are unable to calculate the appropriate gradient of L_{MM} w.r.t. the model parameters during training due to computational constraints. Xu et al. encounter the same problem, since the margin calculation (recall Equation 2.7), is within itself an optimization problem. Inspired by MMA [36], Xu et al. derive a closed-form expression that allows efficiently calculating the gradient of L_{MM} w.r.t. the model parameters. This approach allows for the model parameters to be updated based on a more accurate representation of how the margin changes, rather than approximating it as done by Elsayed et al.[9] and Ding et al.[36].

2.5 Robustness and generalization

As previously discussed in Section 2.4, larger margins in the input space imply greater adversarial robustness. However, as previously mentioned, larger margins also often imply reduced generalization ability, which is known as the robustness accuracy trade-off. In this Section we elaborate on prior work that investigates this phenomenon.

2.5.1 Robustness accuracy trade-off

Although AT is a widely used approach to improve the robustness of classifiers i.e reducing the error on the adversarial examples, several studies have shown that it and other methods that improve robustness often lead to a decrease in clean accuracy i.e increase in the standard error, commonly referred to as the accuracy-robustness trade-off [1], [6]–[8]. This trade-off has been extensively studied, with numerous works investigating the inherent tension between enhancing robustness and maintaining generalization on standard, non-adversarial samples. However, studies by Rózsa et al. [45] and Gilmer et al. [46] suggest that better generalizing models can also exhibit improved robustness.

Raghunathan et al. [6] made several observations supporting the trade-off between generalization and robustness. They noted that the gap between the adversarial loss and standard loss reduces when increasing the size of the labeled dataset and that the trade-off is due to insufficient samples and that additional unlabeled data can mitigate this trade-off [6], [8]. Moreover, Raghunathan et

al. argue that in the case of overparameterized models (having more parameters θ than training data), the trade-off between standard and robust error arises because regularization, such as L2, controls weight size but ignores the data's underlying distribution. They state this trade-off can be eliminated in linear regression when incorporating a regularizer that utilizes the distribution of the inputs.

Tsipras et al. [1] provide another compelling explanation for the robust-accuracy trade-off, illustrating it as an inevitable consequence of the conflicting objectives between adversarial robustness and generalization. In their analysis, Tsipras et al. distinguish between robust features, features that are strongly correlated with the true label which remain stable in the presence of adversarial perturbations, and non-robust features, which are weakly correlated with the label and are vulnerable to manipulation by an adversary. For a standard classifier, any feature that is even slightly correlated with the label is considered useful, meaning the model will rely on both robust and non-robust features to maximize accuracy. However, this reliance on non-robust features becomes problematic in an adversarial setting, as these features can be easily manipulated, leading to a significant drop in adversarial accuracy. They argue that the trade-off between standard and adversarial accuracy is a fundamental trait of the data distribution. They observed that standard classifiers, which aim to maximize accuracy, often rely on non-robust features that can be exploited by adversarial attacks. In contrast, adversarially robust models focus on learning features that are invariant to perturbations, which are typically the more robust, strongly correlated features. This leads to a decrease in standard accuracy, as the model discards non-robust features that contribute to performance in non-adversarial conditions.

Tsipras et al. provided examples where they visualized the gradient of the loss with respect to the input pixels ($\frac{\partial \mathcal{L}}{\partial x}$) for several images, as shown in Figure 2.3. These visualizations show that the gradients of adversarially trained networks align with perceptually relevant features, indicating that such models learn to prioritize features that are truly representative of the class label. In contrast, standard models exhibit noisy gradients, relying on features that do not necessarily correspond to meaningful or robust patterns in the data. It is hypothesized that these noisy gradients are due to the model relying on non-robust features. In conclusion, the work by Tsipras et al. [1], along with other studies [6]–[8], highlights the inherent conflict between standard accuracy and adversarial robustness. The trade-off between these objectives is likely closely linked to the nature of the data distribution and the specific goals of adversarial

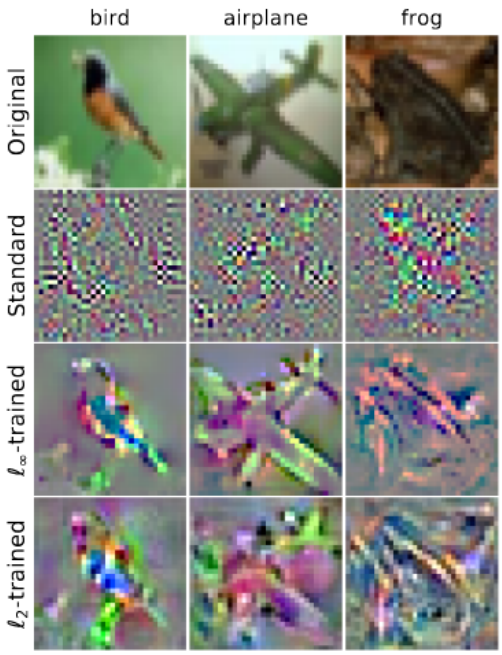


Figure 2.3: Visualizing the loss gradients with respect to individual features in the input space on the CIFAR-10 dataset by Tsipras et al. with the bottom two rows adversarially trained[1]

robustness.

2.5.2 Generalization prediction

As we elaborated in Section 2.5, increasing margins results in improved robustness, but can also lead to reduced generalization on non-adversarial data. That said, margin measurements have also been used to predict the generalization performance of DNNs. In this section, we elaborate on prior work that utilizes margins for generalization prediction, in order to make the link between margins and generalization more clear. Both hidden [32] (Section 2.5.2) and input margins [33] (Section 2.5.2) have been explored for the purpose of predicting the generalization of DNNs. We shall elaborate on both of these approaches.

Hidden margins

As previously mentioned, 'generalization' refers to a model's ability to perform well on unseen data. A key concept related to this is the generalization gap, which is defined as the difference between training and test performance. Jiang et al. [32] consider the problem of predicting

this gap for trained models when utilizing only the model parameters and training data. Their proposed solution relies on measuring the margins of a large number of training samples at several layers throughout the network. More specifically, their prediction process is as follows:

- Given a large set of trained models with known generalization gaps, the set is divided into a training and test set.
- For each network, the Taylor-approximated margin (recall Equation 2.8) is measured at the input layer as well as some selected hidden layers for a large number of training samples. This results in a distribution of margin values for each layer.
- The margin distributions are then normalized, as different layers can have different activation scales, which makes comparing the distributions difficult between layers.
- Each layer's margin distribution is summarized using several statistical measurements, e.g. the median and inter quartile range.
- A linear regression model is fit using the summary statistics from each model within the train set to predict the generalization gap.
- The resulting linear predictor is then evaluated by how well it predicts the generalization gap on the test set of models.

Their results show that these margin distributions serve as reliable features for predicting the generalization gap in this setting. They further emphasize the necessity of obtaining margin information from all layers to improve predictions of the generalization gap. This indicates that the input margin alone is not sufficient.

Constrained margins

Mouton et al. [33] also utilize margins to predict generalization gap, although in a different setting than that of Jiang et al. Fitting a linear regression model necessitates a training set of models for which the generalization gap is already known, so instead Mouton et al. consider the problem of *ranking* models within a set according to their estimated generalization gap, without using a training set. The measure is then evaluated by determining how well the proposed measure ranks the model in comparison to the true generalization gap ranking of the models.

The same setting has also been previously investigated in the Predicting Genralization in Deep Learning (PGDL) challenge [47], which serves as the test bed in their study.

Mouton et al. show that in the ranking setting, there are several limitations to the hidden margin approach of Jiang et al. For one, even though the margin distributions are normalized, comparison between different layers remains difficult. This is especially troublesome when comparing the margins of models with different architectural setups, for example a different number of layers or differently sized layers. To address these issues, Mouton et al. [33] adopt a different approach by confining their margin measurements to the input space. This alleviates the necessity of normalizing margins, as all models in the comparison share the same input space. However, they show that following the naive approach of simply ranking the models according to the mean input margin of many training samples is not predictive of the generalization gap, i.e. this margin ranking does not align with the true ranking.

This finding is consistent with the previously mentioned robustness-accuracy trade-off (Section 2.5.1): if greater robustness often implies reduced generalization performance, and larger margins lead to greater robustness, it would be expected that large margins do not necessarily correlate with generalization (positively or negatively). To address this issue, Mouton et al. leverage the idea of robust and non-robust features (recall Section 2.5). They argue that for a margin measurement in the input space to correlate with generalization, it should be measured in directions which do not rely on non-robust features (weakly correlated with the label), but rather rely on robust features (strongly correlated with the label). The intuition is that the margins in these directions should provide a better indication of how well the model separates the various classes in its training data.

Mouton et al. propose a new margin measurement called 'constrained margins'. Specifically, this metric measures the margin in a specific subspace in the input space that is spanned by the directions which explain most of the variance in the data, effectively filtering out these non-robust features. They used Principal Component Analysis (PCA) to identify these subspaces. Although PCA is typically used for dimensionality reduction, it can also be viewed as a tool for discovering a low-dimensional manifold that captures the true structure of the data.

Figure 2.4 is an illustration of the functionality of constrained margins in a three-dimensional input space, used for easier visualization. This figure shows a two-dimensional subspace spanned

by the top two principal components (PC1 and PC2), which explain the most variance. When the margin is measured, the search space is confined to this subspace, which effectively confines the directions in which the margin can be measured.

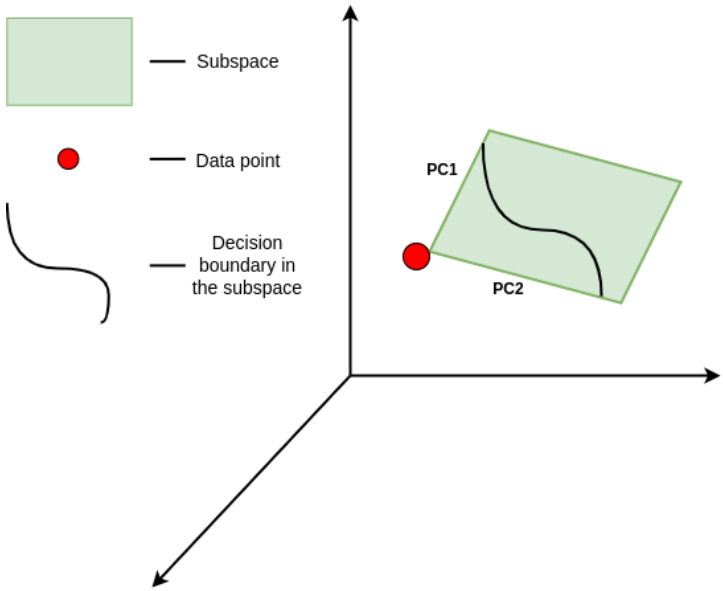


Figure 2.4: A three-dimensional example of the Constrained Margin Plane (CMP) for two principal components, PC_1 and PC_2.

To measure a constrained margin, Mouton et al. adapt the Taylor approximation to calculate the margin within these subspaces, as defined in Equation 2.19.

$$R_\theta(\boldsymbol{x}) = \frac{z_\theta^y(\boldsymbol{x}) - z_\theta^{y'}(\boldsymbol{x})}{\left\| \left[\nabla_{\boldsymbol{x}} z_\theta^y(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} z_\theta^{y'}(\boldsymbol{x}) \right] \mathbf{P}_m^T \right\|_q}, \tag{2.19}$$

where $\mathbf{P}_m$ is the $m \times n$ matrix formed by the top m principal components (with n input features). Mouton et al. performed their experiments using the $L2$ norm and evaluated the predictive power of constrained margins on the PGDL benchmark.

They show that the mean constrained margin of a model serves as a reliable indicator of the model's generalization gap ranking. Notably, Mouton et al. employed Kendall's rank correlation to assess the relationship between the mean constrained margins and generalization ability of the models. Their findings demonstrated a reduction in the correlation with generalization as the number of Principal Components (PCs) increased. This suggests that margins derived from the top principal components, those explaining more variance, are more predictive of generalization. Therefore, focusing on subspaces with higher variance is key to enhancing generalization

predictions. Unlike standard input margins, these constrained margins have shown effectiveness as post-hoc generalization predictors.

Constrained margins are shown to generally be more predictive of a model's generalization performance than other margin measurements and are not burdened by the necessity of normalization, as is the case with hidden margins. In particular, Mouton et al. have demonstrated that constrained input margins outperform standard input margins and hidden margins in the majority of cases.

2.6 Conclusion

In this chapter, we provided an overview of key concepts related to robustness, including adversarial examples (Section 2.2.1) and how they are generated using attack algorithms and how these attack algorithms can be used to evaluate the robustness of models (Section 2.2.2). We also discussed robust training methods for improving the robustness of DNNs, distinguishing between AT (Section 2.3), utilizing adversarial examples and MM (Section 2.4) that aim to maximize margins during training in the input space. Similarly, we discuss how we approximate margins in DNNs (Section 2.4.1) and provide a detailed discussion on how adversarial examples are generated utilizing this approximation. Moreover, we show how this approximation of the margin is utilized in MM loss functions for maximizing margins in the input space (Section 2.4.2).

Additionally, we explored how improvements in robustness affect the generalization ability of models (Sections 2.5.1) and how the link between margins and generalization prediction in previous studies (Section 2.5.2) has led to the development of constrained input margins that can effectively predict generalization by focusing on margins in directions that explain the most variance.

These concepts will be revisited in Chapter 3, where we will implement the Taylor approximation alongside the margin maximization approaches. We will also utilize the FAB approach and look at the trade-off between robustness and generalization of models. In Chapter 4, we will adapt the constrained margin approach which have shown to be predictive of generalization in post-hoc analysis, to the MM approaches to determine if we can improve the generalization performance

of DNNs during training.

Chapter 3

Margin maximization for improving robustness

1 Thessalonians 5:18: "Give thanks in all circumstances; for this is the will of God in Christ Jesus for you."

3.1 Introduction

This chapter explores the role of margin maximization in improving adversarial robustness of DNNs, before maximizing constrained margins, focusing on margin maximization techniques from Elsayed et al.'s method (referred to from here onwards as "Elsayed's method") and DyART, as discussed in Chapter 2. The goal is to identify which elements contribute the most to improved robustness, particularly against strong adversarial attacks like AutoAttack and PGD. We start by developing baseline models, followed by the systematic integration of margin-related components from both methods. Through these experiments, we gain insight into how these components impact performance and robustness, enhancing our understanding of effective strategies to increase adversarial robustness.

3.2 Approach

As we discussed in Section 2.4.2, two promising margin maximization techniques are the MM loss function of Elsayed et al. and the DyART method of Xu et al. Comparing these two methods, it is evident from Section 2.4.2 that DyART can be considered a more complex version of Elsayed's approach that introduces several improvements:

1. DyART introduces an *exponential loss function* which penalizes small margins more.
2. A *burn-in phase* is added to DyART, ensuring that only correctly classified samples contribute to the loss function.
3. DyART utilizes *FAB*, as described in Section 2.4.1 to approximate the margins to the *soft decision boundary*.
4. DyART incorporates a *closed-form solution for calculating the gradients* of $\mathcal{L}_{MM}$ with respect to the model parameters, allowing updates based on how the margin changes, rather than treating the denominator from Equation 2.8 as a constant.

Our approach is to progressively introduce each of these improvements individually to Elsayed's loss function to isolate and identify design choices that lead to improved adversarial robustness. In this section, we elaborate on the different models we train and the expected behavior of each design choice. We provide more extensive implementation and HP details in the following section. We first describe our baseline models in Section 3.2.1. Following this, we describe the different elements that we introduce to Elsayed's method in Section 3.2.2. Finally, in Section 3.2.3 we describe how we evaluate the performance of each model.

3.2.1 Baseline models

We start by implementing three baseline models: 1) a standard model trained with vanilla cross-entropy loss, 2) a model trained using Elsayed's MM loss function, and 3) a model trained with Xu's DyART MM loss function. This allows us to compare each change we make to the original methods, as well as compare to a 'normally' trained model.

We make some slight changes to the original implementations to suit our needs. In terms of

Elsayed's method, we opt to only approximate the margin to the highest predicted non-label class, instead of calculating the margin for every class and using an aggregation operator. This significantly reduces the computational burden. Additionally, as mentioned in Section 2.4.2, Elsayed et al. also experimented with increasing the margins at the hidden layers. We only apply his approach in the input space in order to make it comparable to the DyART method.

For the DyART method of Xu et al. we make some slight changes. They implemented stochastic weight averaging [48] for improved performance. We omit this to better isolate the effect of each loss function individually. In addition to this, they also train on an additional 10M synthetically generated images to further improve their robustness results. We omit this for the same reason.

3.2.2 Additions to Elsayed

In this section we elaborate on each of the elements we add to Elsayed's MM method.

Elsayed's MM with burn-in

Our approach begins with the *Burn-in Phase*, where we enhance the standard MM approach of Elsayed et al. by incorporating an initial burn-in period. During this phase, the model undergoes natural training for several epochs using standard cross-entropy loss before any MM techniques are applied. This helps the model establish a solid baseline with relatively high clean accuracy and a higher proportion of correctly classified samples.

Elsayed's MM with burn-in and exponential Loss.

Following the burn-in phase, we experiment with the Exponential Loss Function, as defined in Equation 2.16. This function is designed to operate on correctly classified samples, which requires a burn-in period, and encourages the model to maintain a margin of at least γ. Because this loss function is exponential, it penalizes samples with smaller margins more heavily than those with larger margins. Incorporating this into Elsayed's loss function might be beneficial in improving the adversarial robustness due to the way smaller margins are managed and penalized.

Elsayed's MM with FAB

Next, we replace Elsayed's Taylor approximation (Equation 2.8) with FAB, with the aim of achieving a more precise margin approximation. FAB accurately calculates the closest boundary points, offering a significant improvement over the original first-order Taylor approximation method. This, however, also requires adapting FAB to function similarly to our adapted Elsayed et al. to only consider the highest predicted class instead of all the classes. This provides some insight into the importance of a much more accurate approximation of the margin, asking whether it is worth the added computational cost and whether this is crucial for increasing the adversarial robustness of DNNs.

Elsayed's MM with FAB and soft boundary

Finally, we integrate the Soft Boundary technique with our adapted FAB to balance the trade-off between computational cost and the accuracy of the margin estimation. Comparing the results of using FAB on its own versus the soft boundary will shed more light on the importance of a more accurate approximation of the margin and whether reducing the compute by utilizing their soft boundary approach can be beneficial.

Table 3.1 aims to clarify the components added to Elsayed's MM function in the experiments. By systematically testing the listed components in a carefully considered order: beginning with the Burn-in Phase, followed by the Exponential Loss Function, then Improved Margin Approximation with FAB, and finally the integration of the Soft Boundary, we aim to build a comprehensive understanding of how each element contributes to the overall robustness of the model.

Elsayed's MM	Burn-in	Exp Function	FAB	Soft Boundary	Gradients
with no additions	×	×	×	×	×
with Burn-in	✓	×	×	×	×
with Burn-in and Exp. Loss	✓	✓	×	×	×
with FAB	×	×	✓	×	×
with FAB and Soft Boundary	✓	×	✓	✓	×
DyART	✓	✓	✓	✓	✓

Table 3.1: Comparison of components added to Elsayed's MM Function

3.2.3 Evaluating robustness

For each model, we evaluate its adversarial robustness in several ways. More specifically, we rely on adversarial attack algorithms and focus on three specific attacks and two key use cases. First, we use standard PGD as used by Madry et al. [17] to generate adversarial examples for each epoch during training using the validation set. These examples serve as an additional validation set, where we calculate the accuracy and robust loss of our model. This approach helps mitigate robust overfitting –where prolonged training leads to increased robust loss despite decreasing (clean) training loss [49] – by applying early stopping on the adversarial validation set.

The second use case involves evaluating the adversarial robustness of the final models. We rely on two different attack algorithms: APGD [18] – an improved version of the standard PGD attack [17], and finally AutoAttack [18]. While AutoAttack serves as a strong benchmark, it includes APGD as one of its gradient-based attacks. By separately analyzing using the APGD attack, we gain a more nuanced view of how these models respond to different gradient-based attacks, given that it is not as strong an attack as the overall measure provided by AutoAttack. This combination provides sufficient information to evaluate the adversarial robustness of our models and gain insights into the contributions of the different components.

3.3 Experimental setup

In this section, we elaborate on our implementation and HP details for the experiments outlined in the previous section. First, in Section 3.3.1, we describe the dataset and architecture we consider, along with the standard training HPs. Following this, in Section 3.3.2, we elaborate on how HPs were tuned for each model individually.

3.3.1 Architecture and standard training parameters

In this section, we explain our standard training setup, including the dataset, architecture, and training procedure we use. We use CIFAR10 [50] for all the experiments. This is a popular dataset to investigate adversarial robustness [9], [10], [24], [37]. This is because even though the dataset is considered relatively simple, building a robust CIFAR10 classifier is still an open

problem [24]. Furthermore, we specifically focus on robustness to l_∞ attacks with a maximum perturbation bound of $\delta = \frac{8}{255}$, which is also a popular evaluation case to investigate on CIFAR10 [37].

In terms of model architecture, Xu et al. and Elsayed et al. conducted experiments using larger CNN-based architectures, such as Wide Residual Networks and ResNet-18s. However, these are very large models that are expensive to train. Instead, to accommodate our computational budget, we rely on the smaller but well known VGG-16 architecture [51] for all experiments.

We train our models using stochastic gradient descent with a momentum of 0.9, employing cosine annealing for the learning rate without restarts and a minimum learning rate decay of 0.001 [52]. Cosine annealing allows us to set an initial learning rate and define a target learning rate to which it should decay or increase. This approach enables us to maintain constant learning rates during training or to implement learning rate schedules with specific patterns, such as increasing or decreasing the learning rates from some initial learning rate to some specified target value. The models are trained for 200 epochs, and we use batch normalization on all layers. Xu et al. also experimented with group normalization [53]. We explored the use of group normalization compared to batch normalization during initial experiments and found that batch normalization offers better performance.

For model selection and to mitigate robust overfitting [49], we dynamically create an additional adversarial validation set after each training epoch consisting of 1,024 samples generated from the original validation set using a 20-step PGD attack [17] with a perturbation bound of $\delta = \frac{8}{255}$. Early stopping is performed based on the model's performance on this adversarial validation set, and the model with the highest adversarial validation accuracy was selected.

3.3.2 Hyperparameter optimization

In this section, we expand on how HPs were chosen for each model. We list each model and describe the HP optimization process. From each sweep, we select two models: the model with the highest clean validation accuracy and the model with the highest APGD validation robust accuracy. We report on the results of these HP optimizations in the following section. We demonstrate the HP sweep protocol for one set of models in Appendix A.1. A similar process is followed for the other optimized models, even though the specific HPs searched over differ

from experiment to experiment, as described. The found HPs for these models are reported in Appendix A.1, Table A.3.

Baseline VGG16

We performed extensive HP sweeps to identify the best settings for our VGG-16 model. Initially, we explored various batch sizes (64, 128, 256, 512), learning rates (0.1, 0.01, 0.001), and weight decay values (0.001, 0.0005, 0.0002) to determine the optimal configuration. The model with the highest validation accuracy used a batch size of 128, a learning rate of 0.1, and a weight decay of 0.0005. This model achieves a very respectable 89.24% accuracy on the CIFAR10 test set. We also use these HPs as a starting point for further experiments.

DyART

To find optimal HPs for the VGG16 model using the DyART training method, we conduct the HP sweep in two separate phases. First, we conducted a sweep to optimize the burn-in period. This phase involved experimenting with several learning rates (0.1, 0.01, 0.001) and batch sizes (64, 128, 256, 512, 1024). Since the DyART training method is computationally expensive (due to the expensive margin-measuring method), we limited the burn-in phase to 25 epochs of standard training followed by only 3 epochs using the DyART loss function to assess hyperparameter effectiveness after initialization. (This is enough as the first two epochs would show signs of catastrophic forgetting [54].) This approach ensures that the HPs not only perform well during the burn-in phase but are also useful for DyART. For the HPs specific to the DyART MM loss, we use the exact same HPs as used by Xu et al. [10], which are $\alpha = 3$, $\gamma = \frac{16}{255}$, $\lambda = 1,000$, $\beta = 5$, and $\delta = \frac{8}{255}$. For the second phase, we selected the top 10 models' HPs based on their adversarial validation accuracy from the initial sweep. We then proceeded to retrain these models for 25 epochs of natural training followed by 200 epochs with the DyART loss function.

Elsayed's margin maximization loss

For the MM loss function of Elsayed et al., we conducted a full sweep to determine the optimal learning rate (0.1, 0.01, 0.001) and batch size (128, 256, 512). In addition, it was necessary to also search over values of λ (10, 15, 20, 25), as the scale of the margin loss is different than when using Xu's method and therefore requires a different weighting with cross-entropy [1]

Additions to Elsayed

For each component that we add to Elsayed's loss function (as described in Section 3.2), we start by using the HPs that yielded the highest robust validation accuracy for the vanilla implementation of Elsayed's margin maximization loss. However, we also search over different learning rates for each addition to ensure performance, as the learning rates employed for the original models may not be ideal when integrating the new components. This is with the exception of the addition of Xu's exponential margin loss, which necessitates changing λ as it changes the scale of the margin loss. In the following section we present results using both $\lambda = 25$ (as used for Elsayed's method) and $\lambda = 1,000$ (as used for Xu's method).

3.4 Results

This section presents the outcome of the experiments designed to optimize adversarial robustness for the VGG-16 model using the approaches detailed in Section 3.2 and the setup defined in Section 3.3. As outlined in Section 3.3.2, we conducted extensive HP sweeps across various configurations, including different learning rates, batch sizes, and regularization parameters, to identify the optimal settings for both DyART and MM loss functions. The results of these sweeps are summarized in Table 3.2. We first discuss the observations for each result in Table 3.2 individually, before considering the results as a whole in the following section.

[1] We could also not rely on the λ values chosen by Elsayed et al., as they included the margins of hidden layers which also changes the scale.

Table 3.2: Performance of VGG-16 models evaluated under a perturbation bound of 8/255. 'Clean Accuracy' includes both validation and test accuracies, while 'Robust Accuracy' includes results from AutoAttack and APGD. For each configuration we report the results for models with the highest clean accuracy and the highest APGD accuracy on the test set.

Model	**Robust Accuracy**		**Clean Accuracy**	
	AutoAttack	**APGD**	**Val**	**Test**
VGG-16 baseline				
Baseline	0	0	90.08	89.24
Elsayed's MM function				
Best clean accuracy	0	13.17	88.64	87.62
Best APGD accuracy	0	14.29	88.70	87.35
Elsayed's MM with Burn-in				
Best clean accuracy	0	13.30	88.12	87.00
Best APGD accuracy	0	27.64	87.34	86.44
Elsayed's MM with burn-in and exponential loss $\lambda = 25$				
Best clean accuracy	0	9.22	87.84	87.07
Best APGD accuracy	0	26.78	86.30	85.15
Elsayed's MM with burn-in and exponential loss $\lambda = 1,000$				
Best clean accuracy	0	47.38	88.32	87.51
Best APGD accuracy	0.13	49.26	84.58	83.60
Elsayed's MM with FAB				
Best clean accuracy	0	4.46	88.54	87.47
Best APGD accuracy	0	5.40	85.98	85.24
Elsayed's MM with FAB and soft boundary				
Best clean accuracy	0	2.80	87.38	86.86
Best APGD accuracy	0	25.70	86.32	85.26
DyART				
Best clean accuracy	0.01	18.77	85.60	84.89
Best APGD accuracy	33.24	33.52	74.72	72.81

VGG-16 baseline

The VGG-16 baseline shows no robustness under strong adversarial attacks, with an APGD and AutoAttack accuracy of 0%. This is to be expected, as the model has not undergone any training to withstand adversarial attacks.

Elsayed's MM function

Introducing Elsayed's MM function drastically improves the model's robustness, with the Best APGD accuracy model achieving 14.29% accuracy on APGD, while maintaining a relatively high clean accuracy of 87.35%. This initial comparison shows the effectiveness of Elsayed's MM function in significantly enhancing adversarial robustness compared to the baseline without a substantial drop in clean accuracy. However, we do not observe any gains in accuracy when using AutoAttack.

Elsayed's MM with burn-in

The burn-in phase to Elsayed's MM function further improves robustness. The model with the best APGD accuracy reaches 27.64%, almost doubling the robustness compared to the MM-only approach. The clean accuracy sees a slight decrease, but this trade-off can be justified by the considerable gain in robustness. This indicates that first allowing the model to learn from clean examples before attempting to maximize the margins is the better strategy for achieving robustness. That said, against the stronger adversary of AutoAttack, we still observe an accuracy of 0%.

Elsayed's MM with burn-in and exponential loss

The exponential loss with $\lambda = 25$ leads to mixed results. While the best APGD model still performs well with an APGD of 26.78%, this is slightly lower than with burn-in alone. However, when λ is increased to $1,000$, there is a significant boost in robustness, with the APGD jumping to 49.26%, a substantial improvement over previous configurations. This illustrates that the λ parameter must be carefully calibrated to strike an optimal balance between standard cross-

entropy loss and the MM loss. We also find a 0.13% robust accuracy on AutoAttack using the larger value for λ, albeit at the cost of a noticeable reduction in clean accuracy, highlighting the increased trade-off as robustness improves (recall Section 2.5.1). This points to the notion that the exponential function is effective (to some extent) in mitigating the 'conflicting dynamics' issue identified by Xu et al. (recall Section 2.4.2).

Elsayed's MM with FAB

Utilizing FAB to estimate the margin in Elsayed's MM function results in a decrease in APGD, with the best model achieving only 5.40% accuracy. Even when a soft boundary is added, the maximum APGD accuracy achieved is 25.70%, which is lower than the robustness obtained with the exponential loss approach (which uses the first-order Taylor approximation). This indicates that FAB, while an alternative, might not be as effective in this context when compared to the Taylor approximation. This is especially striking if one considers that FAB is much more computationally expensive.

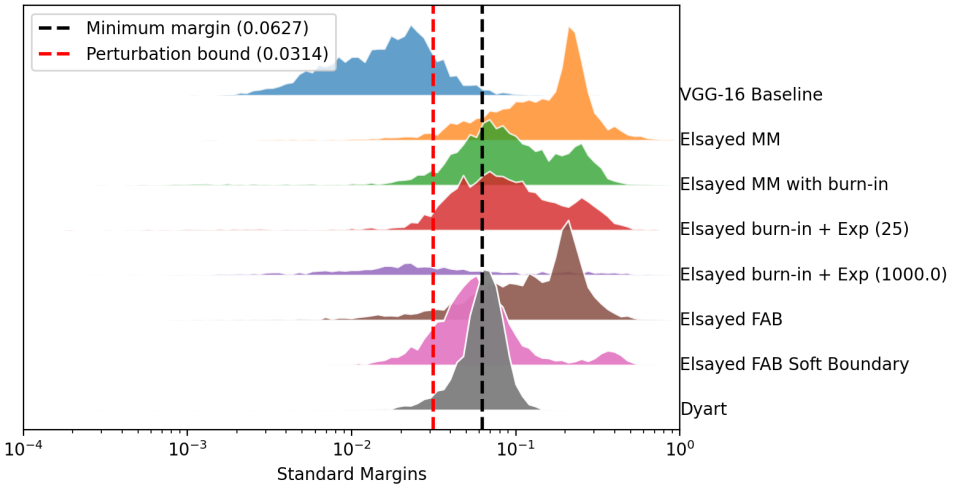
DyART

DyART shows different behavior compared to variations of Elsayed's method. While it has a APGD of 33.52%, this does not surpass the APGD score of the 'Elsayed's MM with Burn-in and Exponential Loss $\lambda = 1,000$' method. However, we do observe a respectable 33.24% accuracy when considering the stronger AutoAttack adversary, which is a substantial improvement on all of the other methods. We suspect that this stems from DyART's more accurate calculation of the margin gradients. That said, we note that DyART also has a more aggressive trade-off between robustness and clean accuracy.

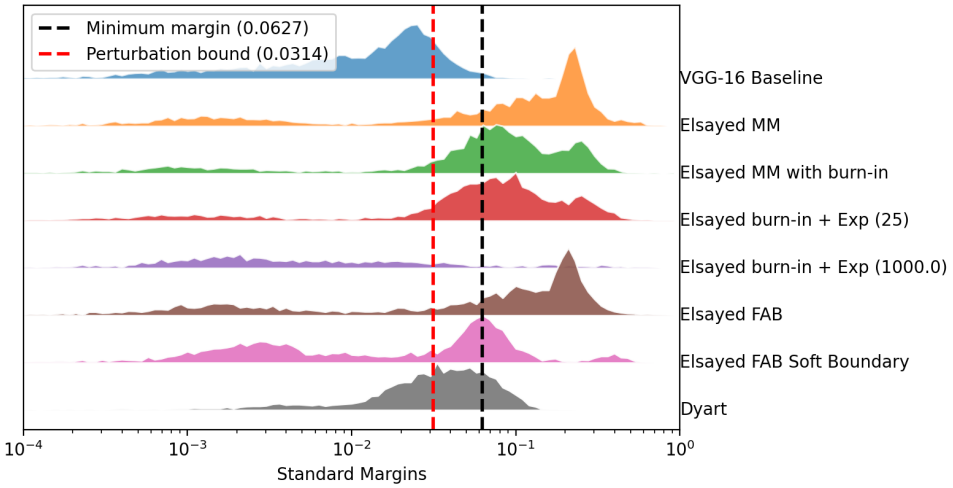
3.4.1 Distribution analysis

To better understand the results, we visualize the final margin distributions for the best APGD models in Table 3.2. We generate margins using the train and test sets, separately. For these plots, we calculate the standard margin (using FAB) for each of these models for 5000 training/test samples using 100 bins. Additionally, we draw vertical lines at the minimum margin

defined for these experiments ($\gamma = \frac{16}{255}$), but also the defined perturbation bound of $\delta = \frac{8}{255}$. Figure 3.1, shows the results.



(a) Train set margins for models with the highest APGD accuracy.



(b) Test set margins for models with the highest APGD accuracy

Figure 3.1: Comparison of margin distributions for models achieving the highest APGD accuracy maximizing standard margins.

From the distribution plots in Figure 3.1, we observe that most MM functions successfully maximize the standard margins on the train set, resulting in median margins larger than both the baseline model and the defined threshold γ.

Another key observation is that these models exhibit larger margins on the train set compared to the test set, indicating robust overfitting.

For the baseline VGG-16 model, the margin distributions on both the train and test sets lie predominantly to the left of δ with a small number of margins larger than δ. This suggests that this model should have had some robustness against these perturbations not aligning with its poor performance in Table 4.2, which shows a robust accuracy of 0%. This highlights that margins smaller than γ make the model more susceptible to perturbations as we see that most samples had margins smaller than γ.

For Elsayed's MM and its variations with burn-in, we observe a high concentration of margins exceeding both γ and the perturbation bound on the train set. However, on the test set, these margins often fall below the perturbation bound. Recall from Section 2.2.1 that the perturbation bound defines the strength of an attack, with its norm representing the maximum distance a sample can be perturbed. Thus, a model with margins exceeding the perturbation bound should theoretically be robust against such attacks. Nevertheless, the test set results show that while these models have some margins exceeding γ, they still have many samples with margins smaller than the defined perturbation which aligns closely with their APGD accuracy. While the APGD accuracy is rather low for Elsayed's MM (14.29%), the margins seems predominantly to the right of the red line in Figure 3.1b, this suggests that APGD is sometimes able to fool the model when FAB is not, i.e. FAB is not accurately estimating the margin for some samples.

Notably, Elsayed's MM with burn-in using the exponential weighting ($\lambda = 1,000$) achieves the highest APGD accuracy (49.26%). However, its margin distribution is more dispersed, lacking the concentration observed in other models. This suggests that its performance may not directly correlate with margin size, implying potential overfitting to the APGD attack.

Furthermore, for Elsayed's MM with FAB Soft Boundary, the train set shows a higher concentration of large margins compared to the test set, where many samples fall below γ.

For DyART, we observe a shorter tail in its test set margin distribution, aligning well with its objective of penalizing smaller margins. Additionally, the margins cluster tightly around γ. Although DyART does not achieve margins as large as those produced by other techniques, it outperforms them against AutoAttack. This highlights that having a higher concentration of margins larger than γ, rather than merely maximizing margin size, is crucial for improving robustness.

3.4.2 Discussion

When considering the results of the previous section as a whole, we noted that all configurations with Elsayed's MM function surpassed the VGG-16 baseline in terms of robustness. This clearly demonstrated the effectiveness of these techniques in defending against adversarial attacks while also highlighting the trade-off with clean accuracy. Furthermore, these models unfortunately did not achieve any robustness improvements when evaluated using AutoAttack.

It was also evident that some of the additions made by Xu et al. are more effective at improving robustness than others. More specifically, we noted that incorporating the exponential loss function achieved a 34.97% increase in robust accuracy evaluated using APGD over the standard approach by Elsayed et al. Likewise, we found that incorporating a burn-in period of standard training was also very beneficial for improving robust accuracy. Conversely, and quite surprisingly, we found that a more accurate and more computationally expensive approximation of the margin with FAB did not improve the adversarial robustness of these models.

When comparing the variations of Elsayed's. method to DyART, we found that the addition of the margin gradients led to a substantial increase in AutoAttack robust accuracy, but simultaneously, a larger trade-off between the robustness and clean accuracy of the models.

3.5 Conclusions

While both Elsayed et al.'s and Xu et al.'s margin maximization techniques improve the adversarial robustness of deep neural networks in the input space, the specific elements contributing to this improvement was unknown. This chapter aimed to address this by providing a detailed overview of the two techniques and isolating the individual elements of the techniques, evaluating their performance using current SOTA adversarial attack algorithms.

We have identified certain key elements that contribute to adversarial robustness in DNNs, as well as certain design choices that do not have a large effect. More specifically, we find that:

1. A period of standard training before introducing margin maximization (known as a burn-in period) is a simple yet effective trick to increase adversarial robustness.

2. Penalizing samples with smaller margins more is very beneficial, such as through the use of an exponential loss function.

3. Using a more accurate estimation of the margin does not necessarily lead to improved robustness.

4. An accurate calculation of the gradient of the margin with regard to the model parameters greatly improves adversarial robustness.

In conclusion, this chapter provided insights into the elements that positively contribute towards designing more robust image classification models. While the studied techniques initially seem complex, it has been shown that they can be decomposed into a number of fairly simple elements, each of which can be analyzed individually. From the observed results, it is expected that future developments that target more accurate and efficient gradient estimations will be more beneficial than techniques aiming to produce more accurate estimations of the margin itself. Next, in Chapter 4, we explore the same questions, but using constrained margins rather than standard margins.

Chapter 4

Large Constrained Margins

Colossians 3:17: "And whatever you do, whether in word or deed, do it all in the name of the Lord Jesus, giving thanks to God the Father through him

---

4.1 Introduction

In this chapter, we aim to enhance the generalization performance of DNNs by incorporating constrained margins (Equation 2.19) into the existing margin maximization techniques, outlined in Section 2.4.2. The primary goal is to assess whether these margin maximization functions can effectively increase constrained margins and, in turn, improve test accuracy, thereby enhancing generalization performance.

We first introduce the general approach used throughout our study (Section 4.2, followed by a discussion of the normalization of constrained margins in Section 4.3. This normalization is required to improve the efficiency of HP sweeps and ensure comparability with standard margins. In Section 4.4, we apply constrained margin maximization in Section 4.5 to gain deeper insights into their contributions to margin maximization for improving generalization. As in Chapter 3, we first use a constant Hessian during optimization but then consider the effect of backpropogating the Hessian as well in Section 4.6. Finally, in Section 4.7, we examine the

relationship between constrained margins and adversarial robustness.

4.2 Approach

In this section, we focus on maximizing constrained margins effectively by adapting constrained margins to the MM loss functions and then progressively applying constrained margins to components from Chapter 3, which have been shown to improve the robustness of DNNs. We aim to establish whether these MM approaches and components that maximize standard margins for improved robustness will lead to improved generalization when maximizing constrained margins. We outline the reasoning behind our decisions and describe the approach used to ensure optimal model performance. In Section 4.2.1, we describe our baseline models used in this Section. Following this, we introduce the components from Chapter 3, which will be used to maximize constrained margins.

4.2.1 Baseline models

We have two sets of baseline models for comparison: 1) The VGG-16 baseline, established in Chapter 3, which is a standard model trained using vanilla cross-entropy loss, and 2) The MM models discussed in Section 3.2.2, which focus on increasing standard margins. At this stage, we do not include FAB or soft-boundary variants of Elsayed's MM due to their computational complexity. These baseline models allow us to analyze the trade-off between clean accuracy and robust accuracy when employing constrained margins versus standard margins. To facilitate this comparison, we employ the ℓ_∞ norm for approximating constrained margins. Additionally, by comparing these models to a naturally trained model, we can assess whether constrained margins can effectively serve as a regularizer.

4.2.2 Adapting MM for constrained margins

We select the following three variants from Chapter 3: Elsayed's MM, Elsayed's MM with burn-in and Elsayed's MM with burn-in and the exponential loss function, and adapt them for constrained margin maximization. In order to maximize constrained margins, we need to

utilize the constrained margin approximation from Equation 2.19 using the l_∞ norm. This will allow us to approximate the margins in directions that explain most of the variance. The current MM techniques from Section 2.4.2 require two inputs: the minimum margin γ and the approximation of the margin R. Therefore, adapting the MM techniques for maximizing constrained margins requires substituting the first-order Taylor approximation from Equation 2.8 with that of constrained margin approximation in Equation 2.19. We now elaborate on the elements we add to Elsayed's MM method, adapting the approach to specifically maximize constrained margins.

Additions to Elsayed's MM function using constrained margins

In this phase, we simply adapt Elsayed's standard MM approach to maximize constrained margins instead of standard margins. Additionally, we incorporate the burn-in phase into Elsayed's MM function and implement Xu's exponential loss function, adapted for constrained margins. By incorporating constrained margins into these components, we aim to enhance the model's generalization ability penalizing smaller constrained margins.

To optimize these additions, we use the same protocol as followed in Section 3.2.2 to identify the set of HPs that effectively maximize the constrained margins for each component. We elaborate more on it later in Section 4.4.1. From Section 2.5.2, we know that the number of PCs is a HP that allows us to consider more directions explaining the variance but beyond a certain point the additional components captures less meaningful variations in the data. Moreover, the margins will vary greatly for different PCs as selecting less principal components implies measuring the margin in a smaller dimensional search space, as such we expect the size of the margins to vary greatly as the number of principal components are changed. This adds several complexities during the HP tuning process, as sweeps over different PCs would require their own unique γ value which is not desirable. We provide a solution to this in the following section.

4.3 Normalizing margins

As mentioned in Section 4.2.2, the number of PCs is a HP that affects the predictive performance of constrained margins, and we also suspect different margin sizes for different numbers of PCs.

To verify this, we calculate the mean Taylor approximated margin for our trained baseline VGG-16 model for 1 to 100 PCs using $5,000$ samples. This is shown in Figure 4.1.

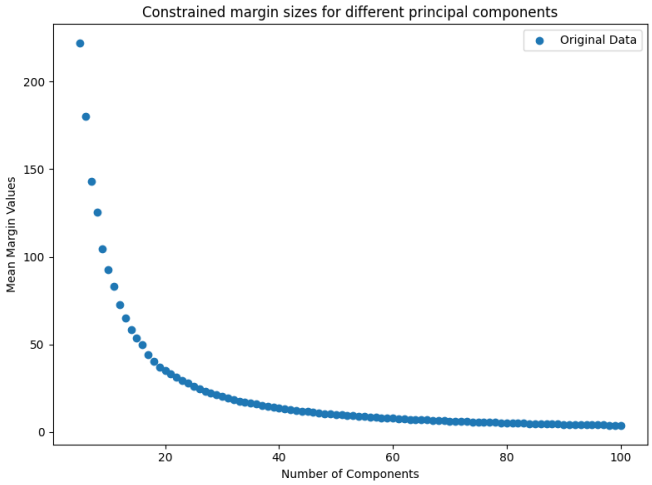


Figure 4.1: Average constrained margin sizes across varying numbers of principal components (PCs), calculated for the baseline VGG-16 model using its training data.

It is evident from Figure 4.1, that fewer PCs generate larger constrained margins, and more PCs generate smaller margins. The change in margin values can be explained by the fact that a smaller number of PCs captures the core structure of the data, focusing on the most important directions of variance, while adding more PCs incorporates noise and less relevant features, reducing the overall margin size as the subspace expands.

This is problematic as this increases the number of HPs to search over and also requires conducting sweeps for each addition to find the optimal γ value for each set of PCs. This is computationally very expensive and not an efficient process. Due to this, we aim to normalize the constrained margin such that the size is invariant to the number of PCs used and similar to those of standard margins.

4.3.1 Goal

Our key objective is to normalize constrained margins to ensure consistency across different numbers of PCs. This normalization aligns the mean sizes of constrained margins with those of standard margins, enabling direct comparisons between the two approaches. By doing so, we can leverage a known γ value from robustness studies as a starting point for experiments,

simplifying HP selection. Additionally, this approach reduces the complexity of HP optimization by allowing the use of a single γ HP across varying PCs.

4.3.2 Experimental setup

In this section, we elaborate on the implementation detail for normalizing constrained margins. We utilize the model obtained in Section 3.3.2, which was trained on the CIFAR-10 dataset and was trained naturally using only cross-entropy loss. To normalize constrained margins, we plan to fit a curve that models the relationship between the mean margin and the number of PCs. Specifically, we select a function that can fit the curve of the mean constrained margin size, similar to that shown in Figure 4.1. This function will enable us to normalize the input mean margin value by modeling the relationship between the mean constrained margin and the number of PCs. To obtain the curve we need to fit, we first calculate and save the PCs on the CIFAR-10 training dataset. We then calculate the constrained margin for PCs 1 to 100 using $5,000$ samples from the dataset and record the mean margin values (as shown in Figure 4.1).

Now that we have the data, we need to establish a family of equations that can fit the curve of mean constrained margins against the number of principal components. This process involves trial and error to identify the appropriate equation that best represents this relationship. To fit the data, we apply Non-linear Least Squares optimization, which allows us to determine the optimal parameters for the function that minimizes the error. Once we have a function that models the relationship, we are able to normalize the constrained margins. The mean of the constrained margin can now be scaled to match the mean of standard margins, to produce scaled, normalized constrained margins. We verify the accuracy of this normalization by comparing the distributions of both the constrained and standard margins.

4.3.3 Results

We evaluate the normalization function's fit and then compare the constrained margin distribution to standard margins, applying a scaling factor to align the distributions for further analysis.

To fit the normalization function, we combined reciprocal functions with a logarithmic term as

described in Equation 4.1.

$$y = a + \frac{b}{x} + \frac{c}{x^2} + \frac{d}{x^3} + e \cdot \log(x) \tag{4.1}$$

The values of a, b, c, d, and e were optimized using non-linear least squares, yielding values of $a = -19.910, b = 720.276, c = 4054.033, d = -8763.092$, and $e = 3.551$ as shown in Figure 4.2. This fitted function provided a strong approximation for the mean margin values across different PCs.

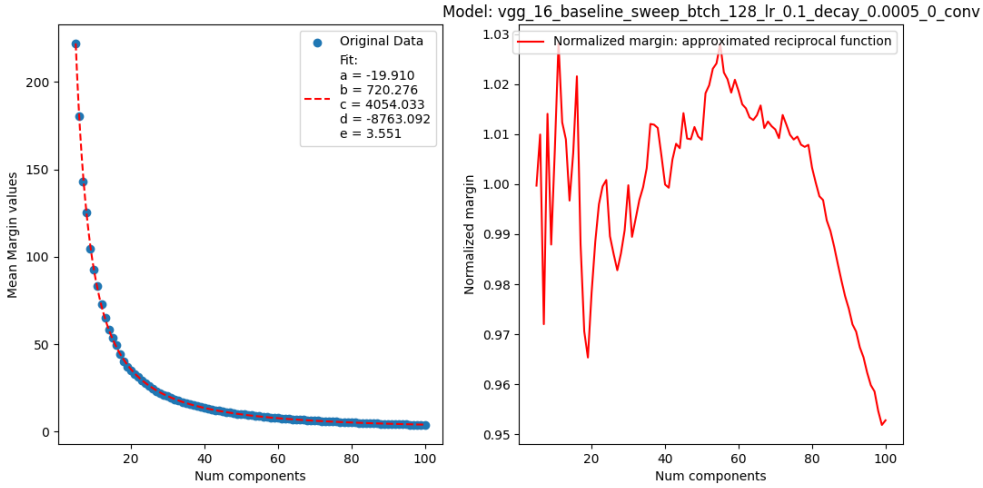


Figure 4.2: Left: The reciprocal function (Equation 4.1) modeling the relationship between mean constrained margins and the number of principal components (PCs). Blue dots represent the original data points, while the red dashed line shows the fitted curve with parameters a=19.910, b=720.276, c=4054.033, and d=3.551. **Right**: The normalized margin approximation as a function of the number of PCs, with y-axis spanning only a narrow range (0.95-1.03), indicating relatively small deviations from unity. (Note the scale.)

Table 4.1 summarizes the mean and median values for both the constrained margins and standard margins. We observe that the normalized constrained margins are significantly larger than standard margins, necessitating a scaling factor to align the distributions.

Table 4.1: Comparison of margin metrics: constrained, normalized, scaled, and standard.

Margin Type	**Mean**	**Median**
Constrained	65.117	59.180
Constrained (normalized)	0.712	0.643
Constrained (normalized + scaled)	0.057	0.051
Standard	0.057	0.053

To align the scales, we divide the normalized constrained mean with the mean of the standard margins to obtain a scaling factor of 12.5, which provides mean constrained margin values that match those of the standard margins (0.057), as shown in Table 4.1. This adjustment allows us to use $\gamma = \frac{16}{255}$, aligning constrained margins with standard margins, and enabling fair comparisons in subsequent analyzes. Figure 4.3 presents the margin distribution for the results in Table 4.1.

4.4 Constrained margin maximization

This section aims to maximize constrained margins using Elsayed's MM functions, as defined in Section 4.2.2. This involves incorporating the constrained margin normalization from Section 4.3 to determine whether constrained margins can be used as a regularizer to improve generalization. We provide the experimental setup in Section 4.4.1 and discuss our results in Section 4.4.2.

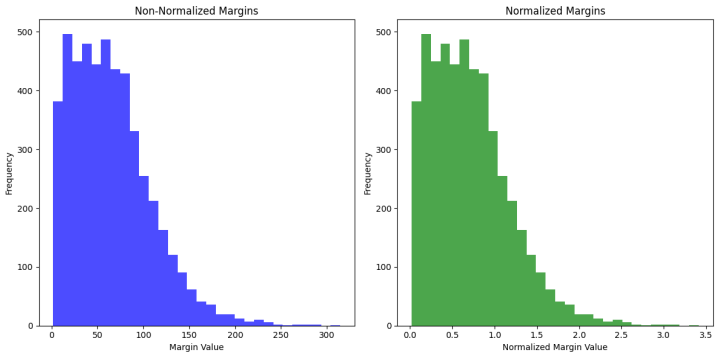
4.4.1 Experimental setup

Here, we elaborate on our implementation. First, we describe the dataset and architecture we use, along with the standard training HPs. Following this, we elaborate on how HPs were tuned for each model individually.

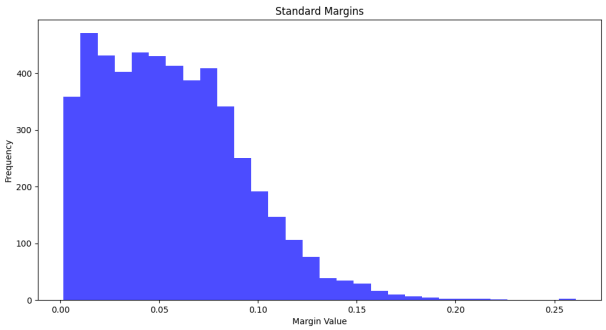
Standard training parameters

In this section, we explain our standard training setup, including the dataset, architecture, and training procedure that we use. We use the CIFAR-10 dataset [50] for all of our experiments. Furthermore, we specifically focus on using the l_∞ norm for the margin calculations in the experiments and not the L_2 norm as used by Mouton et al.. This was done to maintain consistency with the robustness experiments, allowing the results incorporating constrained margins to be directly comparable to those using standard margins. Similarly to Chapter 3, to accommodate our computational budget, we rely on the smaller but well known VGG-16 architecture [51] for all experiments.

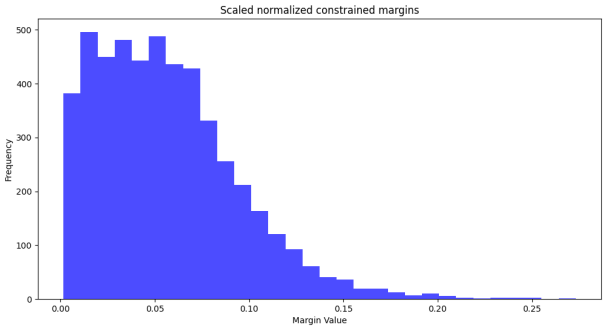
Recall the standard training parameters as outlined in Section 3.3.1. Similarly, for these experiments we train our models using stochastic gradient descent with momentum of 0.9, employing



(a) Constrained margins: non-normalized (left) and normalized (right)



(b) Standard margins



(c) Normalized and scaled constrained margins

Figure 4.3: Margin distributions for VGG-16 baseline model, 30 bins. Constrained margins calculated using 10 PCs.

cosine annealing for the learning rate without restarts and a minimum learning rate decay of 0.001 [52]. We also apply weight decay of 0.0005 on all experiments as used in our VGG-16 baseline model (as identified in Section 3.3.2). These models are also trained for 200 epochs, using batch normalization on all layers. Early stopping is performed based on the model's performance on the validation set, and the model with the highest validation accuracy was selected. Moreover, for the margin calculations, we store the per-batch margin values for both standard and constrained margins during training, and compute the mean and median margin per epoch using these metrics to select the model with the largest margin. After training, we then calculate the constrained margins for all our models on the test set to evaluate the effectiveness of constrained margin maximization. All experiments are conducted on a single NVIDIA A30 GPU to maintain consistency and avoid floating-point discrepancies across multiple devices, ensuring reproducibility.

Hyperparameter optimization

In this section, we expand on how HPs were chosen for the models that incorporate the additions in Section 4.2 using constrained margins and describe the HP optimization process. From each sweep, we select the model with the highest clean validation accuracy.

Additions to MM functions using constrained margins. We conducted several sweeps to define the optimal HPs for the MM functions using constrained margins described in Section 4.2.2. For these experiments, we identified four crucial HPs: learning rate, λ, γ, and number of PCs. For the learning rate, we sweep over various learning rates (0.1, 0.01, 0.001).

Furthermore, we aim to define a range of realistic PCs values for the sweeps to limit computation. We consider the work of Mouton et al., which demonstrated that increasing the number of PCs beyond a certain threshold leads to a negative correlation with the generalization ability of models. Moreover, Mouton et al. used 5 components for generalization prediction on CIFAR-10 models with a VGG-like architecture. However, initial experimentation shows that 5 components is too limited a search space for our needs. Instead, we search over a range of $10, 20, 50$ and 100 PCs, as these values were shown to maintain correlation with generalization by Mouton et al.. Additionally, they observed that this correlation rapidly decreases past 100 PC.

Recall Section 4.3 that constrained margins now have similar margin characteristics to those of standard margins. This enables us to utilize a well-known baseline, $\gamma = \frac{16}{255}$ as used for the experiments in Chapter 3. However, due to the unknown behavior of constrained margins, we also explore larger values of γ ($\frac{32}{255}$, 0.3 and 0.5). This enables us to evaluate the potential benefits of increasing this value.

As we normalize our margins to align with the scale of standard margins, we can use values for λ (specifically 10, 15, 20, 25) as that used for Elsayed's MM functions (without the exponential loss function), similar to that employed in Section 3.3.2. Additionally, we apply a significantly larger value of $\lambda = 50$ to account for a broader range of potential outcomes. This is with the exception of the addition of Xu's exponential margin loss where we will use $\lambda = 25$ (as used for Elsayed's method) and $\lambda = 1000$ (as used for Xu's method) in Table 3.2. Additionally, we explore smaller values and sweep over $\lambda = 0.5$, 1, and 2 for Xu's exponential function to reduce the weight of the MM component for further insights.

4.4.2 Results

The results of these sweeps are summarized in Table 4.2 indicating the mean normalized and scaled constrained margin values for the models as well as their mean standard margins. The HPs for the models presented in Table 4.2 are detailed in Appendix A.2, Table A.3. For the constrained margins in Table 4.2, we report the scaled-normalized constrained margin values.

Did increased constrained margins improve generalization?

We can see that in all cases we achieved increased constrained margin values compared to the baseline model and also achieved mean margin values larger than the defined minimum margin γ. Despite this, we did not obtain improved test accuracy, i.e. generalization performance. Specifically, Elsayed's MM function increased constrained margins compared to the baseline but resulted in lower validation and test accuracy. Adding a burn-in phase led to even higher constrained margins relative to Elsayed's MM alone, though generalization performance decreased further. Moreover, with the addition of Xu's exponential loss function, constrained margins increased successfully unless λ is very large as the model with the highest validation accuracy was achieved during the burn-in period. As mentioned in Section 4.4.1, we conducted further

Table 4.2: Performance of VGG-16 models maximizing constrained margins. 'Clean Accuracy' includes both validation and test accuracies, while 'Mean Margins' include the mean margins calculated for both constrained and standard margins. For each configuration we report the results for models with the highest clean accuracy and the largest mean constrained margin. ('"') indicates that the model with the largest mean constrained margin is the same as the model with the best clean accuracy.

Model	**Mean Margins**		**Clean Accuracy**	
	Constrained	**Standard**	**Val**	**Test**
VGG-16 Baseline				
Baseline	0.057	0.057	90.08	89.24
Elsayed's MM function				
Best Clean Accuracy	0.212	0.244	88.92	87.76
Largest CM	0.315	0.363	88.58	87.90
Elsayed's MM with Burn-in				
Best Clean Accuracy	0.292	0.292	89.36	87.68
Largest CM	0.378	0.423	88.68	87.35
Elsayed's MM with Burn-in and Exponential Loss $\lambda = 0.5$				
Best Clean Accuracy	0.046	0.053	88.54	86.79
Largest CM	0.039	0.085	86.52	84.98
Elsayed's MM with Burn-in and Exponential Loss $\lambda = 25$				
Best Clean Accuracy	0.011	0.012	87.64	86.28
Largest CM	"	"	"	"
Elsayed's MM with Burn-in and Exponential Loss $\lambda = 1000$				
Best Clean Accuracy	-	-	86.90	85.43
Largest CM	0.012	0.013	85.24	84.15

sweeps with smaller λ values using Xu's exponential loss function. These sweeps revealed that $\lambda = 0.5$ achieved the highest validation accuracy for both the models. This further illustrates that constrained margin maximization could not improve the generalization performance.

To conclude, we successfully maximized constrained margins, however this did not result in improved generalization performance. Higher penalization in the MM loss function, particularly with the exponential loss, corresponded to an even greater decline in generalization performance.

Comparing the effect of standard and constrained margins on accuracy

We can see from Table 4.2 that we achieved larger standard margins compared to the baseline model. We also see that maximizing constrained margins resulted in even larger standard margins which is unexpected as we are explicitly maximizing constrained margins. To further understand this phenomenon, let us compare the validation and test performance for the models when maximizing constrained margins and when maximizing standard margins. We use the best clean accuracy values from Table 3.2 in the previous chapter and the models with the highest clean accuracy in Table 4.2.

Table 4.3: Comparison of clean validation and test accuracies between constrained margin (CM) maximization and standard margin (SM) maximization for the VGG-16 model. Results are presented for models with the highest clean accuracy under each configuration.

Configuration	**CM Maximization**		**SM Maximization**	
	Val Acc	**Test Acc**	**Val Acc**	**Test Acc**
Elsayed's MM Function	88.92	**87.76**	88.64	87.62
Elsayed's MM + Burn-in	89.36	**87.68**	88.12	87.00
MM + Burn-in + Exp Loss ($\lambda = 25$)	87.64	86.28	87.84	**87.07**
MM + Burn-in + Exp Loss ($\lambda = 1000$)	86.90	85.43	88.32	**87.51**

As shown in Table 4.3, models trained with Elsayed's MM function, both with and without burn-in, demonstrate better generalization performance when maximizing constrained margins compared to standard margins. However, when incorporating Xu's exponential loss, standard margin maximization achieves improved generalization for both λ values.

Despite this, the more robust configuration in Table 3.2, which utilized both burn-in and Xu's exponential loss with $\lambda = 1000$, resulted in both constrained and standard margins falling below

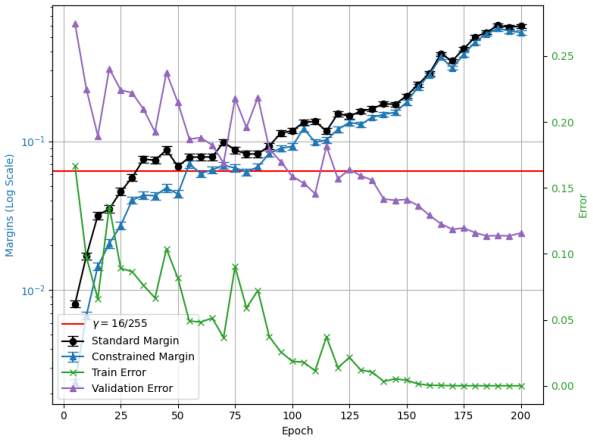
the defined value of γ. This suggests that, even though the constrained and standard margins increased during training, processes that heavily prioritize robustness do not establish a clear link between constrained margins and improved robustness. We will explore this further in Section 4.7.

Notably, using a large λ for constrained margin maximization results in a more pronounced trade-off, reducing generalization performance. This also suggests that the process of maximizing margins inadvertently enhances standard margins, implying a degree of correlation between the two margin approximation approaches. This is not expected because, as we described in Section 2.5.1, standard margins focus on more robust features, while constrained margins specifically look in directions that explain most of the variance, i.e., more spurious features that have been shown to correlate with generalization. Thus, because of the trade-off and the features learned by models that aim to increase robustness and generalization, we would expect increasing constrained margins and not increasing standard margins.

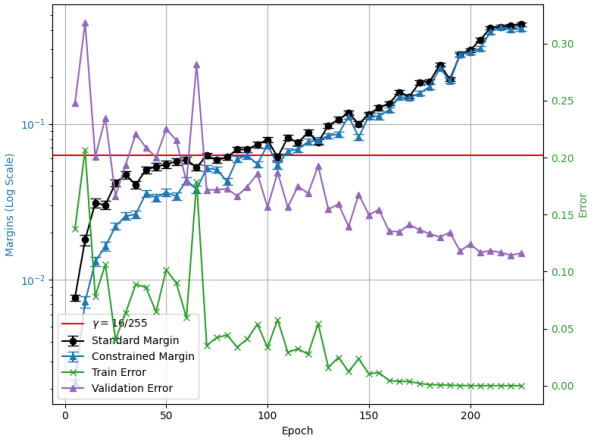
Margins during training

From the results in Table 4.2, even though the constrained margins are successfully maximized, we see that the mean margin values are much higher than the defined $\gamma = \frac{16}{255}$. This is unexpected, as the MM functions explained in Section 2.4.2, are assigned a loss of zero with margins above γ. We show in Figure 4.4 how the mean and standard margins increased during training for the models with the highest clean accuracy in Table 4.2, using a logarithmic scale to capture both large and small values effectively.

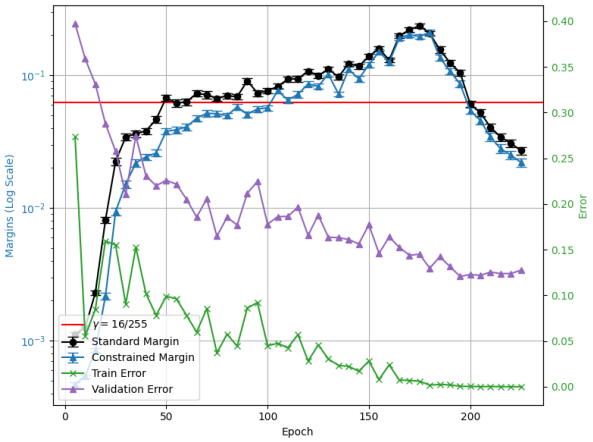
Figure 4.4 shows a notable increase in constrained margins, and we observe that the mean exceeds γ, across all models. This is surprising since the equations in Section 2.4.2 do not target samples with margins larger than $\gamma = \frac{16}{255}$. The HPs used for the MM functions effectively increased the constrained margins, while also reducing loss values, indicating successful model convergence and learning behavior. Similarly, we see that standard margins closely track constrained margins, once again suggesting that there is some correlation between standard and constrained margins and that maximizing margins in the directions explaining most of the variance do not correlate with improved generalization. We explore the possible reasons for this in Section 4.5



(a) Elsayed's MM function



(b) Elsayed's MM function with burn-in



(c) Elsayed's MM function with burn-in and exponential loss function for $\lambda = 0.5$

Figure 4.4: Comparison of constrained and standard margins under different configurations maximizing constrained margins. **Left**: Training and validation errors, along with the standard margin. **Right**: Margins during training, with γ highlighted for reference.

4.5 Implications of using regularizers

The results from Section 4.4.2 were unexpected and pose the question of whether the large increases in constrained margins are caused by the margin maximization method or due to other choices of our training protocol. Specifically, we suspect that regularizers such as weight decay or the learning rate scheduler can contribute to the observed increase in the mean margin beyond the defined γ. In this Section we aim to gain insights on these questions and uncertainties.

4.5.1 Experimental setup

We investigate the cause of the increase in margin values by focusing on the learning rate schedulers and regularizers implemented in our models. Apart from early stopping, the only regularizer used is weight decay. In order to evaluate the effect of regularizers or learning rate schedulers on our models, we isolate the effects of these regularizers by training a model with Elsayed's MM function in three training scenarios:

1. Using our cosine annealing scheduler but removing weight decay (Figure 4.5b).
2. A constant decay learning rate with weight decay (Figure 4.5a).
3. A constant decay learning rate without weight decay (Figure 4.5c).

For these experiments, we select HPs from the model with the highest clean accuracy and then perform a sweep over various learning rates to ensure optimal performance.

4.5.2 Results

This section presents the results of the best performing models on the HPs found during the sweep, focusing on the effects of different learning rate schedulers and the removal of weight decay.

When considering the mean margin, we observe that only the model trained with weight decay achieves a mean constrained margin above gamma. For the learning rate schedule we observe that these schedulers do not aid in maximizing margins, as we can see in Figures 4.5b and

4.5c, the absence of weight decay results in mean margins below the defined γ. Moreover, when considering validation performance, we see that the combined validation loss ($\mathcal{L}_{CE} + \mathcal{L}_{MM}$) reduces i.e. the model is effectively learning.

Figure 4.5 illustrates that weight decay is essential for achieving larger margins, as models trained without it struggle to meet the minimum margin and show lower validation accuracy (below 82%). It might be beneficial to conduct additional experiments in which we obtain a new naturally trained baseline model when not using weight decay to obtain the best possible performance in this scenario. Additionally, while constant learning rate decay supports margin growth, it requires longer training times compared to schedules like cosine annealing as a constant learning rate decay requires more time to achieve smaller learning rates, especially in the absence of weight decay. These results highlight the importance of carefully tuning HPs, particularly weight decay and learning rate schedules, to optimize both margin size and generalization performance.

4.6 Adding Hessian backpropagation

Based on the background provided in Section 2.5.1, you would expect that increasing constrained margins would result in decreasing standard margins or vice versa due to the robustness accuracy trade-off. We found in the sections above that there is not that great a difference between maximizing constrained margins and maximizing standard margins. This might be due to the fact that with Elsayed's method, the denominator of the Taylor approximation is treated as a constant during backpropagation (as mentioned in Section 2.4.2). Since the core difference between calculating standard input margins and constrained margins is in the denominator (recall Equation 2.19), it is natural to assume that additional gradient information might be necessary to effectively increase the constrained margin and not the standard input margin.

4.6.1 Experimental setup

Calculating the gradient of the denominator involves calculating the second-order derivatives, i.e. a Hessian. To do this, we include the denominator calculation in the computational graph in the first-order Taylor approximation. During backpropagation, this enables the calculation

of second-order derivatives ensuring that gradient information is updated based on how the gradient changes. Calculating the Hessian is computationally very expensive, and realistically not a viable option due to the increased training times. However, as we already have a set of HPs that yield good results, we can use these HPs and only add a few additional searches for learning rate. This should provide better maximization of the constrained margins. For the experiments we select the best HPs for the models in Table 4.2 for each addition, and then conduct learning rate sweeps along with the calculation of the Hessian.

4.6.2 Results

This section presents the outcome of the experiments designed to maximize constrained margins when calculating the Hessian of the MM loss functions during backpropagation. Table 4.4, shows the results when calculating the Hessian of the denominator in the first-order Taylor approximation.

Table 4.4 shows that constrained margins are significantly larger when the Hessian is incorporated into the calculations. The burn-in technique further amplifies these constrained margins compared to the results in Table 4.2. Unlike the findings in Table 4.2, where the denominator is treated as a constant and maximizing constrained margins leads to improvements in both constrained and standard margins, incorporating the Hessian allowed us to maximize constrained margins effectively without impacting standard margins. However, this improvement in constrained margins came at the cost of a further reduction in clean accuracy across all configurations.

For models using exponential loss, the highest clean accuracy was obtained during burn-in, before the full application of the MM function, as indicated by the "-" entries. This emphasizes that constrained MM could not improve on the generalization performance achieved with standard cross-entropy loss during burn-in. Furthermore, for the models with the largest constrained margins, increasing λ successfully increased the constrained margins to exceed the defined γ, which was not the case in Table 4.2, and this time without affecting the standard margins. Despite this, we observed a large trade-off, with a 12.82% reduction in generalization performance.

In summary, incorporating Hessian calculations leads to substantially higher constrained margins compared to the results from Table 4.2, does not affect constrained margins, but results in a

Table 4.4: Performance of VGG-16 models maximizing constrained margins when calculating the Hessian of the approximation. 'Clean Accuracy' includes both validation and test accuracies, while 'Mean Margins' includes the mean margins calculated for both constrained and standard margins. For each configuration we report the results for models with the highest clean accuracy and the largest mean constrained margin. (‘”‘) indicates that the model with the largest mean constrained margin is the same as the model with the best clean accuracy. ”-” means that the best model was found during burn-in when no MM was applied.

Model	**Mean Margins**		**Clean Accuracy**	
	Constrained	**Standard**	**Val**	**Test**
VGG-16 Baseline				
Baseline	0.057	0.057	90.08	89.24
Elsayed's MM function				
Best Clean Accuracy	0.003	0.008	86.46	85.35
Largest CM	”	”	”	”
Elsayed's MM with Burn-in				
Best Clean Accuracy	2.508	0.219	85.74	84.88
Largest CM	3.030	0.309	85.06	84.85
Elsayed's MM with Burn-in and Exponential Loss $\lambda = 0.5$				
Best Clean Accuracy	-	-	86.26	84.77
Largest CM	1.574	0.129	84.22	82.58
Elsayed's MM with Burn-in and Exponential Loss $\lambda = 25$				
Best Clean Accuracy	-	-	85.76	85.06
Largest CM	0.181	0.006	77.34	76.47
Elsayed's MM with Burn-in and Exponential Loss $\lambda = 1000$				
Best Clean Accuracy	-	-	86.42	85.32
Largest CM	0.190	0.007	76.06	77.22

larger trade-off of with test accuracy. This emphasizes that maximizing constrained margins during training does not result in improved generalization performance. This points to the question regarding the effect constrained margins have on robustness. We discuss this further in Section 4.7.

4.7 Constrained margin and robustness

Since we could successfully increase the constrained margins during training, but could not improve the generalization performance of these models, we aim to establish whether we achieved improved adversarial robustness when maximizing constrained margins.

4.7.1 Experimental setup

For these experiments we simply utilize the trained models from Tables 4.2 and 4.4, for both the best clean accuracy and largest constrained margin for each configuration. We then evaluate the robust accuracy of these models on AutoAttack and APGD using a perturbation bound of $\delta = \frac{8}{255}$, precisely as done in Chapter 3. This would provide further insight into the effect of maximizing constrained margins.

4.7.2 Results

In this section we show the adversarial robustness of the models which maximized constrained margins. Table 4.5, shows the results when calculating the robust accuracy of the models in Tables 4.2 and 4.4.

From Table 4.5, we compare models trained with and without the Hessian approximation during backpropagation. None of the additions demonstrated improvements in robust accuracy against AutoAttack. While some models trained with burn-in achieved slightly better robust accuracy against the APGD attack, these gains were inconsistent and came at the cost of reduced clean accuracy. Adding the exponential loss function also failed to improve robustness.

Models trained without the Hessian outperformed those trained with the Hessian in both robust

Table 4.5: Performance of VGG-16 models on robust and clean accuracy metrics. 'Clean Accuracy' includes both validation and test accuracies. For each configuration, we report the results for models with the highest clean accuracy and the largest robust accuracy. (‘”‘) indicates that the model with the largest robust accuracy is the same as the model with the best clean accuracy.

Model	**Robust Accuracy**		**Clean Accuracy**	
	AutoAttack	**APGD**	**Val**	**Test**
Elsayed's MM function				
Best Clean Accuracy	0	23.13	88.92	87.76
Largest Robust Accuracy	0	5.04	88.58	87.90
Best Hessian Clean Accuracy	0	1.79	86.46	85.35
Largest Hessian CM	”	”	”	”
Elsayed's MM with Burn-in				
Best Clean Accuracy	0	14.78	89.36	87.68
Largest Robust Accuracy	0	11.70	88.68	87.35
Best Hessian Clean Accuracy	0	3.73	85.74	84.88
Largest Hessian CM	0	1.00	85.06	84.85
Elsayed's MM with Burn-in and Exponential Loss $\lambda = 0.5$				
Best Clean Accuracy	0	0.03	88.54	86.79
Largest Robust Accuracy	0	0.28	86.52	84.98
Best Hessian Clean Accuracy	0	1.39	86.26	84.77
Largest Hessian CM	0	14.17	84.22	82.58
Elsayed's MM with Burn-in and Exponential Loss $\lambda = 25$				
Best Clean Accuracy	0	0	87.64	86.28
Largest Robust Accuracy	”	”	”	”
Best Hessian Clean Accuracy	0	5.27	85.76	85.06
Largest Hessian CM	0	1.00	77.34	76.47
Elsayed's MM with Burn-in and Exponential Loss $\lambda = 1000$				
Best Clean Accuracy	0	3.15	86.90	85.43
Largest Robust Accuracy	0	0	85.24	84.15
Best Hessian Clean Accuracy	0	2.07	86.42	85.32
Largest Hessian CM	0	0	76.06	77.22

accuracy and generalization. As noted in Section 4.6.2, Hessian-based training successfully maximized constrained margins without affecting standard margins. However, Table 4.5 shows that these models experienced the largest trade-offs in clean accuracy and also exhibited reduced robustness compared to models trained without the Hessian, which maximized both constrained and standard margins. This suggests that focusing on constrained margins leads to reduced robustness *and* generalization performance.

We now aim to compare the robustness performance of constrained margin maximization against standard margin maximization approaches from Chapter 3. For this comparison, we focus on the results of constrained margin maximization calculated using the Hessian as this method exclusively maximizes constrained margins. This allows for a direct comparison between pure constrained margin maximization and standard margin maximization. Furthermore, in both cases, we compare the models that achieve the best APGD scores from Table 3.2 and Table 4.5, specifically focusing on the models where Hessian was used for constrained margin calculations.

Table 4.6: Comparison of robustness performance between constrained margin maximization (using the Hessian) and standard margin maximization approaches on VGG-16 models. The Hessian-based results represent pure constrained margin maximization, allowing for a direct comparison against standard margin maximization approaches from Chapter 3. The table displays the test accuracy and robustness of these models measured against AutoAttack and APGD with a perturbation bound of $\delta = \frac{8}{255}$.

Approach	**Variant**	**Clean Acc.**	**APGD**	**AutoAttack**
Standard MM	Base Elsayed's MM	**87.35**	**14.29**	0.00
	MM with Burn-in	**86.44**	**27.64**	0.00
	MM + Burn-in + Exp Loss (λ=25)	**85.15**	**26.78**	0.00
	MM + Burn-in + Exp Loss (λ=1000)	83.60	**49.26**	0.13
Constrained MM	Base Elsayed's MM	85.35	1.79	0.00
	MM with Burn-in	84.88	3.73	0.00
	MM + Burn-in + Exp Loss (λ=25)	85.06	5.27	0.00
	MM + Burn-in + Exp Loss (λ=1000)	**85.32**	2.07	0.00

From Table 4.6, it is clear that standard margin maximization consistently outperforms constrained margin maximization in nearly all configurations. Standard MM not only achieves better robust accuracy, with the best APGD performance of 49.26% in the "MM + Burn-in + Exp Loss (λ=1000)" variant, but also maintains higher clean accuracy. In contrast, constrained MM approaches show significantly reduced robustness, with the best APGD performance only reaching 5.27%. Moreover, the constrained margin models also exhibit lower generalization per-

formance, further emphasizing the trade-off with reduced clean accuracy. These results suggest that standard margins are more effective for both robustness and generalization.

4.8 Discussion

This study investigated whether maximizing constrained margins in deep neural networks could improve their generalization performance. The results reveal that despite successfully increasing constrained margins, this approach did not lead to the anticipated improvements in generalization, and subsequent robustness testing also showed no enhancement in adversarial robustness.

4.8.1 Generalization performance

Our primary finding is that maximizing constrained margins does not improve the generalization performance of DNNs. We find that:

1. Consistent reductions in the validation and test accuracies across all margin maximization configurations compared to the baseline model.
2. Models achieving the largest constrained margins typically showed the worst generalization performance.
3. Even with burn-in periods and various exponential loss parameters (λ), we could not overcome this trade-off between margin size and model performance.

This result is particularly interesting, as it challenges the intuitive expectation that larger constrained margins should lead to better generalization.

4.8.2 Role of weight decay

From our experiments aiming to investigate the cause of the large increase in the constrained margin, we discovered a crucial role of weight decay in the margin maximization process. The experimental results highlighted several key insights:

1. **Essential role of weight decay**: Weight decay proved fundamental for successful margin maximization, with models trained without it failing to achieve acceptable validation accuracy (below 82%). The presence of weight decay significantly impacted both the achievable margin values and the model's generalization capability.

2. **Learning rate interactions**: The effectiveness of margin maximization showed strong dependence on the learning rate schedule. While constant learning rate decay supported margin growth, it required longer training periods compared to cosine annealing schedules, particularly when weight decay was absent.

3. **Margin growth patterns**: Models with weight decay demonstrated more stable and larger margin values, suggesting that this regularization technique plays a vital role in shaping the model's decision boundary during training.

The impact of regularization strategies on margin maximization revealed additional insights:

- **Weight decay and margin values**: Models trained with weight decay consistently achieved higher margin values, indicating that this form of regularization is essential for effective margin maximization while maintaining model stability.

- **Training without weight decay**: The absence of weight decay led to significant challenges in meeting minimum margin requirements, with validation accuracies falling below 82%. This suggests that weight decay plays a crucial role in balancing margin growth with model generalization.

4.8.3 Hessian calculations and model performance

The addition of Hessian calculations for the Taylor approximation yielded notable results:

1. While successfully increasing constrained margins without increasing standard margins, this approach led to further degradation in test accuracy.

2. Models using exponential loss functions with Hessian calculations achieved their best performance during the burn-in phase, suggesting that the margin maximization process itself was detrimental to model performance.

4.8.4 Constrained margins and their effect on robustness

After observing that constrained margin maximization did not improve generalization, we investigated whether it might enhance adversarial robustness. The results were equally disappointing:

1. We had 0% across all configurations against AutoAttack.
2. APGD showed minimal robustness, with even the best models achieving only 23.13% accuracy. We also achieved worse robustness when compared to maximizing standard margins.
3. Larger constrained margins often corresponded to worse robustness metrics.

4.9 Conclusion

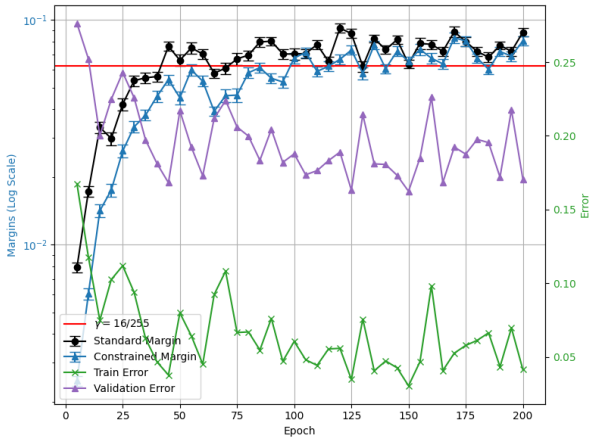
In this study, we investigated whether maximizing constrained margins could improve the generalization performance of deep neural networks. Through extensive experimentation, we found that, while we could successfully increase constrained margins, this did not translate into improved generalization or robustness. Our investigation revealed several key findings that contribute to the understanding of margin-based approaches in deep learning.

First, we demonstrated that weight decay plays a crucial role in the margin maximization process. Models trained with weight decay achieved significantly higher margin values compared to those without weight decay, highlighting the importance of regularization in shaping the decision boundary. However, even with optimal weight decay settings, the increased margins did not lead to better generalization performance.

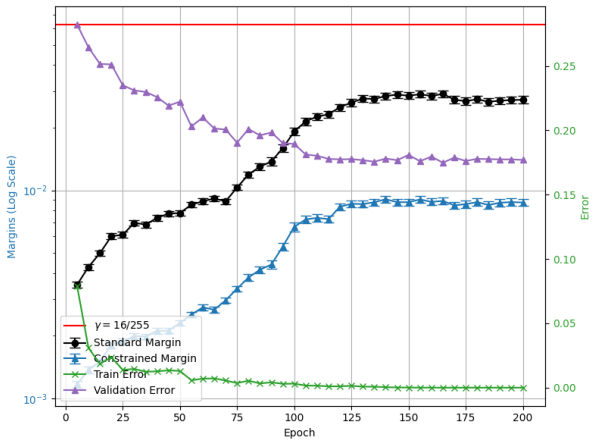
Second, our attempts to enhance the margin maximization process through Hessian calculations of the Taylor approximation, while successful in increasing constrained margins, resulted in decreased test accuracy.

Third, our investigation of adversarial robustness showed that larger constrained margins do not necessarily translate into improved model robustness. Therefore, increasing constrained margins neither improves generalization nor adversarial robustness.

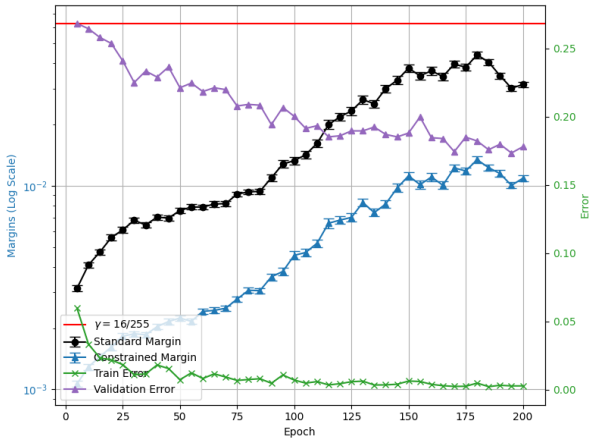
These findings indicate that the relationship between constrained margins and generalization in deep neural networks is more nuanced than theorized. Although the concept of constrained margins is intuitively appealing, our results suggest that directly maximizing constrained margins does not improve the generalization ability of DNNs.



(a) Constant decay learning rate with weight decay



(b) Cosine annealing without weight decay



(c) Constant decay learning rate without weight decay

Figure 4.5: Comparison of constrained and standard margins during training for Elsayed's MM function across different learning rate schedules and the use of regularizers, such as weight decay.

Chapter 5

Conclusion

Hebrews 13:8: "Jesus Christ is the same yesterday, today, and forever."

5.1 Introduction

The goal of this study, as outlined in Chapter 1, was to determine whether constrained margins, which have shown to be predictive of generalization in post-hoc models, could be used as an effective regularizer for enhancing the generalization ability of DNNs. While previous research has shown a correlation between margin measurements and generalization, no methods have been developed to leverage margins, traditionally used to enhance adversarial robustness, to improve the generalization performance of DNNs during training. This chapter evaluates the extent to which these objectives were achieved, discusses key findings, and explores their implications along with potential directions for future research.

5.2 Summary

In Chapter 2, we provided a comprehensive background on adversarial robustness in DNNs and examined the primary techniques used to enhance robustness against adversarial attacks.

We introduced fundamental concepts, including adversarial examples, adversarial attacks, and the methods we employed to evaluate model robustness. Specifically, we discussed adversarial training with a focus on PGD and APGD methods and explored alternative approaches to increase robustness through margin maximization techniques.

Additionally, in Chapter 2, we included an in-depth explanation of margin approximation and how maximizing these margins during training could potentially impact robustness. We discussed the inherent trade-off between generalization and robustness based on current literature, as well as the relationship between margin size, robustness, and generalization.

Our approach aimed to leverage constrained margins, shown to be predictive of generalization, to enhance generalization during training. To achieve this, we examined and adapted key components from the work of Xu et al. and Elsayed et al., identifying which elements contribute to improved adversarial robustness in Chapter 3.

The experimental results from Chapter 3 revealed significant gains in adversarial robustness when maximizing standard margins, especially when incorporating strategies such as burn-in periods and the exponential loss function proposed by Xu et al. Interestingly, we found that while more precise margin approximations did not improve adversarial robustness, accurately calculating the gradient of the margin with respect to model parameters led to substantial robustness gains.

In Chapter 4, our objective was to apply constrained margins to these robustness-enhancing components to determine if maximizing constrained margins could boost generalization. However, our findings indicated that maximizing constrained margins during training did not improve generalization performance in DNNs and, in fact, resulted in reduced test accuracy. Furthermore, constrained margins did not improve adversarial robustness. Consequently, we conclude that constrained margins are not effective as a regularizer for DNNs.

5.3 Key findings

In this section we discuss the key findings of this study and the implications thereof. Specifically, standard margins and their effect on robustness in Section 5.3.1, we then look at the key findings when maximizing constrained margins, and its effect on generalization, Section 5.3.2, and on

robustness Section 5.3.3.

5.3.1 Standard margin maximization and robustness

Here we discuss the key findings from Chapter 3.

- We have shown that incorporating a period of natural training before utilizing MM loss functions greatly improves the adversarial robustness of DNNs. This suggests burn-in is a powerful and easy approach for robustness gains.
- We have shown that modifying Elsayed's well known loss function with an exponential term and adding a period of natural training provides a significant boost in the robustness of DNNs.
- We have shown that more accurate estimations of the margins is not beneficial for improving the robustness of DNNs. We tested this by replacing the first-order Taylor approximation with the more computationally expensive FAB for margin estimation, which yielded no robustness gains over the standard Taylor approximation.
- We have shown that calculating the gradient of the MM loss with respect to model parameters greatly improves the robustness of DNNs on strong attacks like AutoAttack.

5.3.2 Constrained margin maximization and generalizaton

Here, we discuss the the findings from Chapter 4, related to constrained margins and its effect on generalization.

- We proposed a novel normalization approach for constrained margins, which effectively reduces the number of hyperparameters required and allows for clearer comparisons between results achieved with constrained margins and those with standard margins.
- Our study revealed that maximizing constrained margins does not improve the generalization performance of DNNs, as we observed consistent reductions in validation and test accuracies across all configurations relative to the baseline model. Furthermore, when

comparing stopping criteria, the models with the largest constrained margins had poorer generalization performance.

- Our experiments highlighted the crucial role of weight decay in the margin maximization process: it proved essential for successful margin maximization.
- The effectiveness of margin maximization was strongly dependent on the learning rate schedule. While constant decay allowed for margin growth, it requires longer training times than cosine annealing, particularly in the absence of weight decay.
- We observed that adding Hessian calculations in the Taylor approximation during back-propagation increased constrained margins but further decreased the test accuracy. Models using exponential loss functions with Hessian calculations performed best during the burn-in phase, suggesting that the margin maximization process itself was detrimental to generalization.

5.3.3 Constrained margin maximization and robustness

In this section, we list our findings regarding the effect of constrained margins on robustness.

- A secondary investigation into robustness revealed that maximizing constrained margins did not enhance adversarial robustness. All models which maximized constrained margins exhibited 0% accuracy against AutoAttack, and similarly results against APGD showed limited robustness. Furthermore, we found that the best models achieved lower robust accuracies than the models trained with standard margin maximization.
- Larger constrained margins are often correlated with poorer robustness metrics, underscoring that constrained margins do not enhance adversarial robustness in DNN.
- Moreover, we found that standard margins yield both better generalization performance and robustness compared to hessian constrained margin maximization (the hessian resulted in only maximizing constrained margins and not standard margins).

5.4 Future work

Potential future directions include:

- Throughout this study, we have relied on the well known VGG16 architecture and the CIFAR10 dataset. It would be beneficial to explore whether our findings still hold on larger models (e.g. ResNets and Wide-ResNet) and datasets (e.g. ImageNet). This would illuminate the role that model capacity and dataset complexity plays in robustness.

- Given our findings surrounding the margin maximizing effects of weight decay, we wish to gain a more nuanced understanding of which hyperparameter choices such as the optimizer influence robustness and how.

- It is clear that calculating the margin gradients during margin maximization training drastically improve robustness. However, calculating these gradients require an accurate approximation of the margin (e.g. using FAB). It would be beneficial to develop closed form gradient expressions when using more approximate methods (e.g. the Taylor approximation) due to the great computational requirements of more exact methods.

- To further explore the interplay between standard and constrained margins, and how they relate to each other in different training scenarios, for example, using different hyperparameter choices (optimizers, regularizers, etc.) during both natural and robust training.

- Investigating the impact of margin maximization on the loss landscape by examining how maximizing the margin in early training affects the optimization process. Maximizing the margin early in training may direct the model toward regions of the loss surface that are less conducive to generalization, pushing it away from more favorable minima. The extent of burn-in as a hyperparameter could be crucial. Future research could explore training the network with a standard loss function to convergence before introducing margin awareness as a weight refinement step, potentially enhancing both generalization and robustness.

- Investigating the relationship between weight decay and constrained margin maximization. This study observes that weight decay strongly correlates with margin size. Future work could explore how this observation applies to other popular regularization methods, such as dropout.

- A more thorough analysis of the impact of dataset complexity, model architecture, and additional hyperparameters on constrained margin performance.

5.5 Why constrained margins reduce generalization performance

We know that constrained margins are predictive of generalization for post-hoc models. However, we found that maximizing these margins during training, does not result in improved generalization performance. Therefore, constrained margins do not act as a regularizer for DNNs. This leads us to the question of why they are predictive of generalization but reduce generalization (and robustness) performance when maximized during training.

It is not clear why this is the case but we can speculate that this is related to spurious and non-spurious features, as discussed in Section 2.5.2. When generalization improves, constrained margins – that are measured in the most important directions in the data – are a good indicator of performance, as spurious features do not influence the metric. Therefore, constrained margins allow us to ignore spurious features and margins in directions that utilize spurious features. However, when constrained margins are maximized, we artificially increase margins in specific directions: this could overfit the data, resulting in less accurate models, and poorer test accuracy.

As to the reduction in robustness when maximizing constrained margins, we similarly speculate that since constrained margins ignore 'spurious features', these directions can be utilized to create adversarial samples, also resulting in poorer robust accuracy.

5.6 Conclusion

In this study we have gained a better understanding of which components and hyperparameter choices of margin maximization methods contribute to increased margins. We evaluated these in terms of both adversarial robustness and generalization. Our primary goal was to investigate whether maximizing constrained margins leads to improved generalization performance. We find that constrained margins are not an effective regularizer, and the reasons why remain speculative. That said, we believe that we have gained a much better understanding of which DNN characteristics contribute to both generalization and robustness.

Bibliography

[1] D. Tsipras, S. Santurkar, L. Engstrom, A. Turner, and A. Madry, "Robustness may be at odds with accuracy," in *International Conference on Learning Representations*, 2019.

[2] D. Su, H. Zhang, H. Chen, J. Yi, P.-Y. Chen, and Y. Gao, "Is robustness the cost of accuracy?–a comprehensive study on the robustness of 18 deep image classification models," in *Proceedings of the European conference on computer vision)*, 2018.

[3] A. Raghunathan*, S. M. Xie*, F. Yang, J. Duchi, and P. Liang, "Adversarial training can hurt generalization," in *ICML 2019 Workshop on Identifying and Understanding Deep Learning Phenomena*, 2019.

[4] C. Szegedy, W. Zaremba, I. Sutskever, *et al.*, "Intriguing properties of neural networks," in *International Conference on Learning Representations*, 2014.

[5] W. Zhao, S. Alwidian, and Q. H. Mahmoud, "Adversarial training methods for deep learning: A systematic review," *Algorithms*, 2022.

[6] A. Raghunathan, S. M. Xie, F. Yang, J. C. Duchi, and P. Liang, "Understanding and mitigating the tradeoff between robustness and accuracy," in *International Conference on Machine Learning*, 2020.

[7] D. Su, H. Zhang, H. Chen, J. Yi, P.-Y. Chen, and Y. Gao, "Is robustness the cost of accuracy?–A comprehensive study on the robustness of 18 deep image classification models," in *European Conference on Computer Vision*, 2018.

[8] A. Raghunathan, S. M. Xie, F. Yang, J. Duchi, and P. Liang, "Adversarial training can hurt generalization," in *ICML Workshop on Identifying and Understanding Deep Learning Phenomena*, 2019.

[9] G. F. Elsayed, D. Krishnan, H. Mobahi, K. Regan, and S. Bengio, "Large margin deep networks for classification," in *Neural Information Processing Systems*, 2018.

[10] Y. Xu, Y. Sun, M. Goldblum, T. Goldstein, and F. Huang, "Exploring and exploiting decision boundary dynamics for adversarial robustness," in *International Conference on Learning Representations*, 2023.

[11] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[12] I. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *International Conference on Learning Representations*, 2015.

[13] F. Chollet, *Deep Learning with Python*. Manning Publications, 2017.

[14] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015.

[15] M. Belkin, D. Hsu, S. Ma, and S. Mandal, "Reconciling modern machine-learning practice and the classical bias–variance trade-off," *Proceedings of the National Academy of Sciences*, 2019.

[16] B. Neyshabur, R. Tomioka, and N. Srebro, "In search of the real inductive bias: On the role of implicit regularization in deep learning.," in *International Conference on Learning Representations*, 2015.

[17] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *International Conference on Learning Representations*, 2018.

[18] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *International Conference on Machine Learning*, 2020.

[19] F. Croce and M. Hein, "Minimally distorted adversarial examples with a fast adaptive boundary attack," in *International Conference on Machine Learning*, 2020.

[20] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *Conference on computer vision and pattern recognition*, 2016.

[21] Z. Wang, T. Pang, C. Du, M. Lin, W. Liu, and S. Yan, "Better diffusion models further improve adversarial training," in *International Conference on Machine Learning*, 2023.

[22] Y. Bai, M. Zhou, V. M. Patel, and S. Sojoudi, "MixedNUTS: Training-free accuracy-robustness balance via nonlinearly mixed classifiers," *Transactions on Machine Learning Research*, 2024.

[23] S.-A. Rebuffi, S. Gowal, D. A. Calian, F. Stimberg, O. Wiles, and T. Mann, "Data augmentation can improve robustness," in *Neural Information Processing Systems*, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., 2021.

[24] B. R. Bartoldson, J. Diffenderfer, K. Parasyris, and B. Kailkhura, "Adversarial robustness limits via scaling-law and human-alignment studies," in *International Conference on Machine Learning*, 2024.

[25] C. Wei and T. Ma, "Improved sample complexities for deep neural networks and robust classification via an all-layer margin," in *International Conference on Learning Representations*, 2019.

[26] B. Neyshabur, S. Bhojanapalli, and N. Srebro, "A pac-bayesian approach to spectrally-normalized margin bounds for neural networks," in *International Conference on Learning Representations*, 2018.

[27] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky, "Spectrally-normalized margin bounds for neural networks," *Advances in neural information processing systems*, 2017.

[28] V. N. Vapnik, "An overview of statistical learning theory," *IEEE transactions on neural networks*, 1999.

[29] C. Cortes and V. Vapnik, "Support-vector networks mach learn 20," 1995.

[30] M. W. Theunissen, C. Mouton, and M. H. Davel, "The missing margin: How sample corruption affects distance to the boundary in anns," in *Southern African Conference for Artificial Intelligence Research*, 2022.

[31] Y. Guo and C. Zhang, "Recent advances in large margin learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[32] Y. Jiang, D. Krishnan, H. Mobahi, and S. Bengio, "Predicting the generalization gap in deep networks with margin distributions," in *International Conference on Learning Representations*, 2018.

[33] C. Mouton, M. W. Theunissen, and M. H. Davel, "Input margins can predict generalization too," *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[34] Y. Yang, R. Khanna, Y. Yu, *et al.*, "Boundary thickness and robustness in learning models," *Advances in Neural Information Processing Systems*, 2020.

[35] R. Yousefzadeh and D. P. O'Leary, "Deep learning interpretation: Flip points and homotopy methods," in *Mathematical and Scientific Machine Learning*, 2020.

[36] G. W. Ding, Y. Sharma, K. Y. C. Lui, and R. Huang, "MMA training: Direct input space margin maximization through adversarial training," in *International Conference on Learning Representations*, 2020.

[37] F. Croce, M. Andriushchenko, V. Sehwag, *et al.*, "Robustbench: A standardized adversarial robustness benchmark," in *Neural Information Processing Systems*, 2021.

[38] H. Zhang, Y. Yu, J. Jiao, E. Xing, L. El Ghaoui, and M. Jordan, "Theoretically principled trade-off between robustness and accuracy," in *International Conference on Machine Learning*, 2019.

[39] J. Zhang, J. Zhu, G. Niu, B. Han, M. Sugiyama, and M. Kankanhalli, "Geometry-aware instance-reweighted adversarial training," in *International Conference on Learning Representations*, 2021.

[40] Q. Wang, F. Liu, B. Han, *et al.*, "Probabilistic margins for instance reweighting in adversarial training," in *Neural Information Processing Systems*, 2021.

[41] D. Wu, S.-T. Xia, and Y. Wang, "Adversarial weight perturbation helps robust generalization," *Neural Information Processing Systems*, 2020.

[42] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *IEEE Symposium on Security and Privacy*, 2017.

[43] A. Ilyas, S. Santurkar, D. Tsipras, L. Engstrom, B. Tran, and A. Madry, "Adversarial examples are not bugs, they are features," in *Neural Information Processing Systems*, 2019.

[44] M. Andriushchenko, F. Croce, N. Flammarion, and M. Hein, "Square attack: A query-efficient black-box adversarial attack via random search," in *European conference on computer vision*, 2020.

[45] A. Rozsa, M. Günther, and T. E. Boult, "Are accuracy and robustness correlated," in *IEEE International Conference on Machine Learning and Applications*, 2016.

[46] J. Gilmer, L. Metz, F. Faghri, *et al.*, "Adversarial spheres," *arXiv preprint arXiv:1801.02774*, 2018.

[47] Y. Jiang, P. Foret, S. Yak, *et al.*, "NeurIPS 2020 competition: Predicting generalization in deep learning," *arXiv preprint arXiv:2012.07976*, 2020.

[48] P. Izmailov, D. Podoprikhin, T. Garipov, D. P. Vetrov, and A. G. Wilson, "Averaging weights leads to wider optima and better generalization," in *Conference on Uncertainty in Artificial Intelligence*, 2018.

[49] E. Wong, L. Rice, and Z. Kolter, "Overfitting in adversarially robust deep learning," in *International Conference on Machine Learning*, 2020.

[50] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.

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

[52] I. Loshchilov and F. Hutter, "SGDR: Stochastic gradient descent with warm restarts," in *International Conference on Learning Representations*, 2017.

[53] Y. Wu and K. He, "Group normalization," in *European conference on computer vision*, 2018.

[54] B. Pfülb and A. Gepperth, "A comprehensive, application-oriented study of catastrophic forgetting in DNNs," in *International Conference on Learning Representations*, 2019.

Appendix A

Supplemental content

A.1 Appendix: Chapter 3

Here we show the additional standard margin results referenced in Section 3.3.2. We describe the HP sweep protocol followed when adding burn-in to Elsayed's MM function. The process used in the model selection can be transferred to the other implementations as well. First, we had to establish the ideal set of HPs for Elsayed's margin maximization loss that yielded the highest adversarial validation accuracy. This resulted in training 36 models, each with different learning rates, λ values and batch sizes. From the 36 models we selected the model with the highest adversarial validation accuracy (Figure A.1). Figure A.2, shows the train, validation and adversarial validation accuracy, and the loss of the model during training.

Following this, we apply the optimized HPs from Elsayed's MM function to Elsayed's MM with burn-in, adding an additional learning rate sweeps. For Figure A.4, we applied early stopping on the adversarial validation set to save the model for a given number of epochs, with the highest adversarial validation accuracy. For each of the saved models, with their respective learning rates we calculate the robustness of the models against the APGD-CE attack and select the models with the best robust accuracy against APGD.

We then select the models with the highest clean accuracy and report this models results on the APGD attack. We wanted to ensure that the robust accuracy of these models improved during training on the adversarial validation set (Figure A.4 and similarly that the adversarial

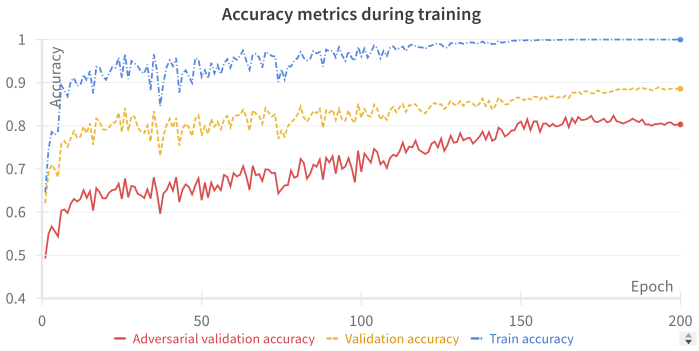


Figure A.1: Train, validation and adversarial validation accuracy during training.

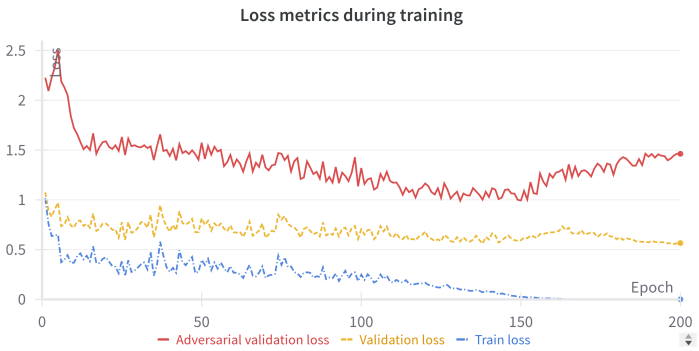


Figure A.2: Train, validation and adversarial validation loss during training.

Figure A.3: Best performing model from Elsayed's MM sweep results

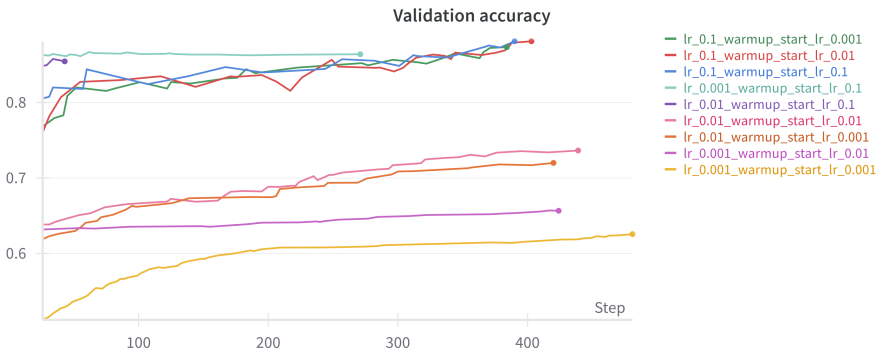


Figure A.4: Validation accuracy of Elsayed's MM with burn-in for various learning rates.

loss decreases (Figure A.5.

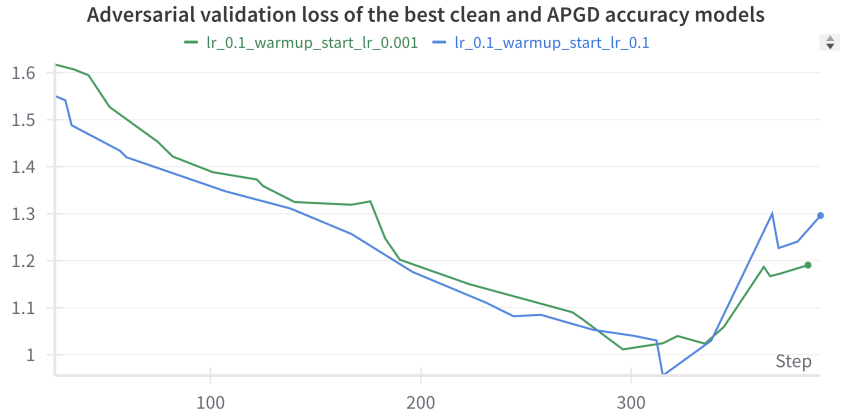


Figure A.5: Adversarial Validation loss of the best APGD and clean accuracy models of Elsayed's MM with burn-in.

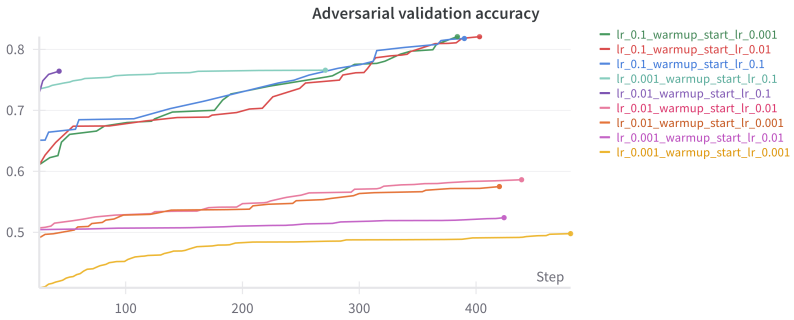


Figure A.6: Adversarial validation accuracy of Elsayed's MM with burn-in for various learning rates.

A.2 Appendix: Chapter 4

Here we show the additional standard margin results referenced in Section 4.4.2.

Table A.1: Hyperparameters for the VGG-16 models evaluated under a perturbation bound of 8/255. The table includes the hyperparameters used for the models with the highest clean accuracy and the highest APGD accuracy on the test set.

Model	Learning Rate	Warmup LR	Batch Size	Weight Decay	λ
VGG-16 Baseline					
Baseline	0.1	-	128	0.0005	-
Elsayed's MM function					
Best Clean Accuracy	0.1	-	128	0.0005	20
Best APGD Accuracy	0.1	-	128	0.0005	10
Elsayed's MM with Burn-in					
Best Clean Accuracy	0.1	0.1	128	0.0005	25
Best APGD Accuracy	0.1	0.001	128	0.0005	25
Elsayed's MM with Burn-in and Exponential Loss $\lambda = 25$					
Best Clean Accuracy	0.1	0.1	128	0.0005	25
Best APGD Accuracy	0.001	0.1	128	0.0005	25
Elsayed's MM with Burn-in and Exponential Loss $\lambda = 1000$					
Best Clean Accuracy	0.1	0.01	128	0.0005	1000
Best APGD Accuracy	0.01	0.1	128	0.0005	1000
Elsayed's MM with FAB					
Best Clean Accuracy	0.1	-	128	0.0005	25
Best APGD Accuracy	0.05	-	128	0.0005	25
Elsayed's MM with FAB and Soft Boundary					
Best Clean Accuracy	0.1	0.1	128	0.0005	25
Best APGD Accuracy	0.001	0.1	128	0.0005	25
DyART					
Best Clean Accuracy	0.001	0.1	128	0.0005	1000
Best APGD Accuracy	0.1	0.1	256	0.0005	1000

Table A.2: Hyperparameters for the VGG-16 models aimed at maximizing constrained margins. The table includes the hyperparameters used for the models with the highest clean accuracy and the largest constrained margin.

Model	**LR**	**Warmup LR**	**Batch Size**	**Weight Decay**	λ	**PCs**	γ
VGG-16 Baseline							
Baseline	0.1	-	128	0.0005	-		
Elsayed's MM function							
Best Clean Accuracy	0.1	-	128	0.0005	10	100	$16/255$
Largest CM	0.1	-	128	0.0005	10	50	$16/255$
Elsayed's MM with Burn-in							
Best Clean Accuracy	0.1	0.1	128	0.0005	20	100	$16/255$
Largest CM	0.1	0.1	128	0.0005	10	10	$16/255$
Elsayed's MM with Burn-in and Exponential Loss $\lambda = 0.5$							
Best Clean Accuracy	0.1	0.01	128	0.0005	0.5	10	$16/255$
Largest CM	0.001	0.1	128	0.0005	0.5	50	$16/255$
Elsayed's MM with Burn-in and Exponential Loss $\lambda = 25$							
Best Clean Accuracy	0.1	0.1	128	0.0005	25	100	$16/255$
Largest CM	"	"	"	"	"	"	
Elsayed's MM with Burn-in and Exponential Loss $\lambda = 1000$							
Best Clean Accuracy	0.001	0.1	128	0.0005	1000	100	$32/255$
Largest CM	0.1	0.01	128	0.0005	1000	50	$16/255$

Table A.3: Hyperparameters for the VGG-16 models aimed at maximizing constrained margins using the Hessian of the approximation. The table includes the hyperparameters used for the models with the highest clean accuracy and the largest constrained margin.

Model	**LR**	**Warmup LR**	**Batch Size**	**Weight Decay**	λ	**PCs**	γ
VGG-16 Baseline							
Baseline	0.1	-	128	0.0005	-		
Elsayed's MM function							
Best Clean Accuracy	0.1	-	128	0.0005	10	100	$^{16}/_{255}$
Largest CM	0.1	-	128	0.0005	10	100	$^{16}/_{255}$
Elsayed's MM with Burn-in							
Best Clean Accuracy	0.1	0.01	128	0.0005	10	10	$^{16}/_{255}$
Largest CM	0.1	0.1	128	0.0005	10	10	$^{32}/_{255}$
Elsayed's MM with Burn-in and Exponential Loss $\lambda = 0.5$							
Best Clean Accuracy	0.001	0.1	128	0.0005	0.5	50	$^{16}/_{255}$
Largest CM	0.1	0.1	128	0.0005	0.5	50	$^{16}/_{255}$
Elsayed's MM with Burn-in and Exponential Loss $\lambda = 25$							
Best Clean Accuracy	0.001	0.1	128	0.0005	25	50	$^{16}/_{255}$
Largest CM	0.1	0.01	128	0.0005	25	10	$^{16}/_{255}$
Elsayed's MM with Burn-in and Exponential Loss $\lambda = 1000$							
Best Clean Accuracy	0.001	0.1	128	0.0005	1000	10	$^{16}/_{255}$
Largest CM	0.1	0.01	128	0.0005	1000	10	$^{16}/_{255}$