Applications of Adversarial Robustness Analysis in Machine Learning

PhD Thesis

István Megyeri

Supervisor: Dr. Márk Jelasity

Department of Computer Algorithms and Artificial Intelligence
Doctoral School of Computer Science
Faculty of Science and Informatics
University of Szeged

Szeged
2023

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Let us start with a short story that demonstrates well the topic of the dissertation and also the current status of artificial intelligence research.

Clever Hans was a horse that gained fame in the early 20th century for apparently being able to perform complex arithmetic and other intellectual tasks. The horse was owned by a German mathematics teacher named Wilhelm von Osten, who claimed to have taught him these skills.

Von Osten became a sensation in Germany, with many people flocking to see the amazing horse. However, skeptics suspected that there was more to the horse's abilities than met the eye. A psychologist named Oskar Pfungst investigated the phenomenon and concluded that Clever Hans was not actually performing arithmetic, but was instead responding to subtle cues from his trainer and audience.

Pfungst discovered that von Osten was unwittingly providing the horse with cues, such as body language or slight head movements that told it when the horse had actually found the correct answer. Once this was realized and providing cues was prevented, Clever Hans was no longer able to perform the same feats of arithmetic when his trainer was not present or when he was blindfolded.

The case of Clever Hans became an important milestone in the history of psychology, as it demonstrated the importance of experimental controls and the potential for unconscious cueing to influence the behavior of both humans and animals.

Artificial intelligence has reached a similar milestone. A subfield called deep learning gained significant popularity in the 2010s. While the concept of neural networks and deep learning has existed for several decades, it was during this period that several factors converged, leading to a surge in its popularity.

One of the main catalysts was the availability of large datasets and advancements in computational power, which allowed researchers to train deeper neural networks and process massive amounts of data more efficiently. And the development of specialized hardware, such as graphics processing units (GPUs), accelerated the training of deep learning models.

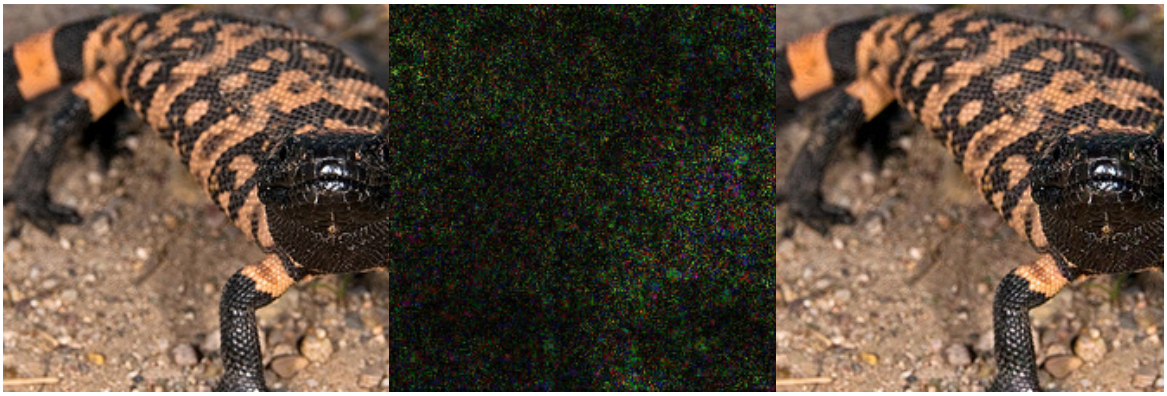

Figure 1.1: *A sample adversarial image. Left: original image (prediction: 'Gila monster'); Middle: adversarial perturbation; Right: adversarial image (DenseNet201 prediction: 'custard apple'). The adversarial modification is invisible to the human eye but it changes the neural network prediction.*

Another crucial factor was the breakthrough in performance achieved by deep learning models in various challenging tasks. Deep learning architectures, such as convolutional neural networks (CNNs) and recurrent neural networks (RNNs), demonstrated superior performance in image recognition, speech recognition, natural language processing, and other domains, often surpassing traditional machine learning approaches.

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) in 2012 played a pivotal role in showcasing the power of deep learning. The winning team, led by Geoffrey Hinton, utilized deep convolutional neural networks and significantly outperformed other methods. This event served as a turning point and drew attention from classical machine learning methods to the capabilities of deep learning.

The success and breakthroughs in deep learning, combined with the increasing availability of open-source tools and libraries, made it more accessible to researchers and practitioners. This accessibility, along with the growing interest from both academia and industry, contributed to the rapid proliferation and popularity of deep learning.

Since then, deep learning has continued to make strides in various fields, and its popularity has only increased. It has become the dominant approach in many areas of artificial intelligence and machine learning, shaping advancements in sectors ranging from healthcare and finance to automotive and entertainment.

A few years after the ImageNet challenge, Szegedy et al. discovered intriguing properties of these highly successful deep neural networks [71]. They found that deep neural networks are discontinuous to a certain extent. They showed that it is possible to cause the network to misclassify an image by applying a certain imperceptible perturbation, which is obtained by maximizing the network's prediction error.

They coined the phrase adversarial examples for these perturbed and misclassified images, which is now widely used in the literature. The presence of adversarial examples was also confirmed by Goodfellow et al. [23]. A sample of an adversarial image shown in fig. 1.1.

Since these seminal papers on this topic appeared, several studies have demonstrated that current deep learning models are highly sensitive to adversarial input where very small adversarially constructed input perturbations can induce incorrect behavior [1, 5, 7, 8, 13, 15, 16, 17, 23, 25, 33, 34, 35, 41, 44, 45, 47, 55, 56, 62, 70, 77, 79, 83, 84, 85, 88]. Further, it is possible to generate perturbations that remain adversarial in the physical world as well i.e. after printing it on a T-shirt [12, 17, 18, 20, 21, 42, 80, 81].

Here, we will focus on the problem of image classification. In it, the sensitivity of the current models to adversarial input indicates that these models are not in complete accord to human perception. Similar to Clever Hans, after this sensitivity was noted the networks were no longer able to solve their given task.

In Chapter 3, we study the adversarial robustness of linear machine learning models where we propose novel insights, that provide an alternative explanation for the adversarial sensitivity of linear models. We will focus on the effect of regularization and dimensionality.

The original formulation of the problem in [71] assumes that we are given a model and a correctly classified example. The attacker wishes to find a minimal perturbation of the example such that the model predicts any wrong label (untargeted attack) or a given desired label (targeted attack). Since then, a large number of methods have been proposed to create better adversarial examples [9, 57].

In Chapter 4, we will propose and study a more general version of this problem, where we are given more than one model and an example. For each model, we will specify whether the given model should correctly classify the example or predict any wrong label or predict a fixed specific label. This formulation permits a wide variety of adversarial constraints on a given model set.

There were also proposed many defense mechanisms [23, 46, 74], many of which were proved to be ineffective in [9] while others remain robust [46, 87]. Among the many defenses against input perturbation, adversarial training was found to be the most effective. In a nutshell, adversarial training means that the model is trained over the adversarially perturbed version of the training data to improve the robustness of the model. Wang et al. provided a recent overview [76].

In Chapter 5, we will study robust out-of-distribution (OOD) detection. Adversarially trained models are relatively robust to adversarial input but they might assign high confidence to OOD samples and this represents a serious vulnerability in a real-world application [65]. Besides this, OOD input is also open to adversarial perturbation, making OOD detection even harder. We will present a systematic design

	chapter 3	chapter 4	chapter 5
ESANN 2019 [50]	•		
ESANN 2020 [52]		•	
IJCNN 2020 [51]		•	
IJCNN 2021 [53]			•
PRAI 2023(submitted for publication) [54]			•

Table 1.1: *The connection between the thesis chapters and publications.*

space that covers most of the popular design choices for the various components of robust classification and OOD detection. This allows us to draw several interesting conclusions based on our empirical results.

1.1 Contributions

The ideas, figures, tables and results included in this thesis were published in scientific papers (listed at the end of the thesis). In table 1.1, we indicate the connection between the chapters and publications. Overall, the author is responsible for the following contributions:

Chapter 2.: In this chapter, we demonstrated that even in the case of simple binary classification problems with linear models, the adversarial problem is real and it strongly depends on regularization and the less obvious properties of high-dimensional spaces. We performed an experimental evaluation and we showed that the optimal regularization strength is very different for adversarial robustness and prediction accuracy, and that the convergence of adversarial robustness is much slower than that of the accuracy metric. Also, in higher dimensions an overly weak regularization setting might result in a significantly harder optimization problem in some cases. The related experimental design, implementation, and analysis of the results were prepared by the author.

Chapter 3.: Here, we introduced an iterative algorithm to find small adversarial perturbations that fool multiple models simultaneously in a given pattern. This problem formulation has many interesting applications, such as the generation of transferable adversarial examples as well as generating a single perturbation such that all the models in a given model set predict specified, different classes. The latter scenario allows us to explore the decision boundaries of the model set from a new perspective. The experimental design, evaluation, as well as the formalism of the problem and the

algorithm, were carried out by the author.

Chapter 4.: The contribution of this chapter is that we defined a design space, where one can systemically analyze the problem of robust OOD detection and robust classification. The main components were identied as the training objectives, detection methods and attack methods for the combination of the robust OOD detection problem and the robust classification problem with the help of a set of score functions. Also, we introduced a strong threat model in which both in-distribution and OOD samples are adversarially perturbed to mislead OOD detection.

Moreover, we performed a thorough empirical evaluation of this framework. We found that adding an adversarial OOD objective to the training method does not harm robust in-distribution accuracy, in fact, a significant improvement can be seen in some cases. This indicates that it is always safe to add such an objective.

We also found that it is impossible to pick a score function for robust OOD detection independently of how the model in question was trained. Instead, we get the best results when training and detection is based on the same score function. In other words, while non-robust OOD detection is more robust to the training procedure, in robust OOD detection it is more important to align the detection method with the training method, that is, to use the same score function in both. Also, a similar statement can be formulated in terms of the OOD detection method and the attack on this detection method. The most successful attack is performed using the same score function as the one used by the detection method.

The unified treatment of the combined problems, implementation and the design of the related experiments were all done by the author.

Chapter 2

Background

This chapter presents the general notations used throughout this thesis. And, it gives a concise summary of the relevant adversarial machine learning concepts such as the kind of adversarial attacks, the network robustness measures, and defenses. This information is required for the reader to understand our contributions and motivations.

2.1 Notations

Let us first introduce some basic notations. We are interested in supervised learning more precisely in image classification task where a set of training instances is given of the form (x, y), $x \in \mathbb{R}^d$, $y \in \mathcal{C}$. The image and the corresponding label from the set of all the labels $\mathcal{C}$ are denoted by x and y respectively. Here, d denotes the dimension of the image and the one-hot encoded vector of the label y is e_y. For n examples (x_i, y_i), $x_i \in \mathbb{R}^d$, $y_i \in \mathcal{C}$, $i = 1, \ldots, n$ and the theoretical distribution $\mathcal{D}$ from which they are drawn from.

To solve the classification problem, we will apply deep neural networks. These are denoted by the function $f : \mathbb{R}^d \to \mathbb{R}^{|\mathcal{C}|}$, which returns the output of a deep multi-layer neural network classifier with parameters θ without a softmax normalization layer that is the so-called logits without any activation. When multiple networks are assumed, f has subscript i.e. f_1,$f_2 \cdots$. The j-th output value of the network on a certain input is refered as $f(x)_j$. In the case of a linear network namely logistic regression, we shall use explicit forms as in eq. (2.1). Here, the set of parameters is $\theta = \{w, b\}$ and σ denotes the activation function.

$$\sigma(w^T x + b) = 1/(1 + \exp(-w^T x + b)) \tag{2.1}$$

We could measure the magnitude of a vector $v \in \mathbb{R}^d$ with respect to the Euclidean norm or infinity norm. These will be defined as in eq. (2.2) and eq. (2.3) respec-

tively.

$$||v||_2 = \sqrt{\sum_{i=1}^{d} v_i^2} \tag{2.2}$$

$$||v||_\infty = \max(|v_1|, \ldots, |v_d|) \tag{2.3}$$

The neural network is typically trained via some gradient-based optimizer such as SGD or Adam [4] to minimize the loss function $\mathcal{L}$. For binary classification, we use the negative log likelihood function eq. (2.4), where α is the regularization coefficient.

$$\min_{w,b} \mathcal{L}(w,b) = -\frac{1}{n}\sum_{i=1}^{n} y_i \cdot \log(\sigma(w^T x_i + b)) + (1-y_i)\cdot\log(1-\sigma(w^T x_i + b)) + \alpha\|w\|_2^2 \tag{2.4}$$

For multi-class tasks we will use categorical crossentropy eq. (2.6), where the softmax activation eq. (2.5) is applied to the network output which transforms the logit vector to a probability distribution. Although any differentiable loss function could be used. Note that, θ holds all the parameters of the deep neural network f.

$$\text{softmax}(l)_j = \frac{\exp(l_j)}{\sum_{i=1}^{|\mathcal{C}|}\exp(l_i)} \tag{2.5}$$

$$\min_{\theta} \mathcal{L}(f) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{|\mathcal{C}|}(e_{y_i})_j \cdot \log \text{softmax}(f(x_i))_j \tag{2.6}$$

2.2 Adversarial attacks and defenses

2.2.1 Adversarial attacks optimization objectives

The adversarial robustness of deep neural networks characterizes the ability of a neural network to maintain its accuracy and performance even when subjected to deliberate adversarial attacks. Adversarial attacks carefully craft perturbations which are applied to input data with the intention of fooling the neural network into making incorrect predictions.

All the attack algorithms find the perturbation by solving an optimization problem. However, we can divide the algorithms into two subcategories based on the way they tackle the amount of perturbation in their optimization problem.

Certain algorithms minimize the size of the perturbation with respect to some norm measures and treat the misclassification as a constraint. Formally speaking, the

optimization problem will have the form as in eq. (2.7).

$$\begin{aligned} &\min_{\delta} ||\delta||_p \\ \text{subject to } &\arg\max_j(f(x+\delta)_j) \neq y \end{aligned} \tag{2.7}$$

In eq. (2.7), the size of the perturbation is minimized while ensuring that the given perturbation leads to misclassification. After solving this optimization problem, we will have such a perturbation that changes the predicted class obtained by the network f. However, the size of the perturbation can be arbitrary, meaning it might be noticeable to humans.

Another popular form of defining the attack objectives is to directly maximize the network's loss function and apply some constraints on the size of the perturbation. The constraints will require an input argument that defines the maximum size of the allowed perturbation. This is commonly defined as epsilon(ϵ), indicating that it should jave a relatively small value. These attacks should have an objective like that in eq. (2.8)

$$\begin{aligned} &\max_{\delta} \mathcal{L}(f(x+\delta), y) \\ &\text{subject to } ||\delta||_p \leq \epsilon \end{aligned} \tag{2.8}$$

The solution of eq. (2.8) will have a perturbation size of at most the predefined ϵ. Therefore, the distortions should meet the requirement of being unnoticeable by humans. However, the misclassification might not be achieved.

Common choices of p include the Euclidean norm($p = 2$) defined in eq. (2.2) and infinite norm($p = \infty$) defined in eq. (2.3).

Surprisingly, these problems are readily solved for most of the currently used networks, which shows the vulnerability of deep neural networks to such adversarial attacks. It has been noticed that even imperceptible perturbations added to the input can lead to significant changes in the network's output. Adversarial attacks exploit this vulnerability and they can have serious consequences in real-world applications, such as misclassifying images, causing security breaches, and manipulating the behavior of autonomous systems.

As a defender, the goal is to find such networks that provide an output consistent within the range of perturbations that are not imperceptible by humans.

2.2.2 Targeted vs untargeted attacks

The attack objectives in section 2.2.1 are untargeted, so the attackers do not specifiy what the wrong label should be, all the labels are accepted which differ from the ground truth. However, in the case of multi-class classification, the mistakes might not all have an equal impact. For example in a self-driving car, confusing a red light and green light is a much more serious issue than recognizing a certain speed limit as a lower or higher one.

For the above mentioned reasons, we might wish to specify the adversarial label that we wish to achieve. The targeted version of eq. (2.7) and eq. (2.8) are in eq. (2.9) and eq. (2.10) respectively.

$$\begin{aligned} &\min_{\delta} ||\delta||_p \\ \text{subject to } &\arg\max_j(f(x+\delta)_j) = t, t \in \mathcal{C} \setminus y \end{aligned} \tag{2.9}$$

$$\begin{aligned} \min_{\delta} &\mathcal{L}(f(x+\delta), t), t \in \mathcal{C} \setminus y \\ &\text{subject to } ||\delta||_p \leq \epsilon \end{aligned} \tag{2.10}$$

2.2.3 The adversary's knowledge of the targeted model

A defender might decide to retain certain information about its model in order to increase the model robustness. The main categories based on the attacker's level of knowledge include:

- White-box attacks: In white-box attacks [7, 8, 15, 23, 45, 47, 56, 70], the attacker has complete knowledge of the target model, including its architecture, parameters, and training data. This allows them to craft highly effective adversarial examples.

- Black-box attacks: In black-box attacks [1, 5, 13, 25, 34, 77, 83, 88], the attacker has limited knowledge about the target model. They may have access only to the inputs and outputs of the model without any knowledge of its internal features. Black-box attacks often involve querying the target model to generate adversarial examples.

- Transfer attacks: Transfer attacks exploit the transferability of adversarial examples [16, 33, 35, 41, 44, 62, 84, 85]. They involve training a substitute model with a similar architecture to the target model and generating adversarial examples on the substitute model, which are then transferred to the target

model.The substitute model's training might also require the collection of training data and associated labels.

- Physical attacks: Physical attacks seek to fool machine learning systems deployed in the real world [12, 17, 18, 20, 21, 42, 80, 81] For example, adversarial stickers or patterns added to objects can cause misclassification in real-time object recognition systems. The main challenge of a physical attack is to find such a modification that fools the network consistently under various conditions such as varying the viewing angle, color or shape variation.

.

2.2.4 A brief review of the employed white-box attacks

Here, we will review and formalize some white-box attacks that are applied in the later chapters.

The fast gradient (sign) method(FGM) was introduced in [23] as a quick and simple method to test the linearity of deep neural networks. FGM is a single-step attack, where the gradient of the loss function with respect to the input data is used to generate the perturbation. The gradient is simply scaled according to the predefined epsilon to leverage the maximum allowed perturbation budget. FGM is part of the loss-based attack algorithms so it will have a similar form to that in eq. (2.8). The L_2 and L_∞ version of the attack are given in eq. (2.11) and eq. (2.12), respectively.

$$\delta = \epsilon \frac{\nabla_x \mathcal{L}(f(x), y)}{||\nabla_x \mathcal{L}(f(x), y)||_2} \tag{2.11}$$

$$\delta = \epsilon \,\text{sign}(\nabla_x \mathcal{L}(f(x), y)) \tag{2.12}$$

The magnitude of the perturbation introduced to the input data is controlled by ϵ. The choice of epsilon affects the perceptibility of the adversarial examples. A smaller epsilon value corresponds to imperceptible perturbations, while a larger epsilon value allows for more noticeable modifications to the input.

Projected Gradient Descent(PGD) [46] is an iterative attack that extends FGSM by performing multiple iterations of gradient calculation and perturbation updates. For each iteration, the perturbation is clipped or projected onto a permissible range defined by the epsilon value, ensuring that the perturbed input remains within the allowed epsilon budget. The L_2 and L_∞ versions of the attack are given in algorithm 1 and algorithm 2, respectively, where α is the step size and m is the number of gradient steps.

PGD can be reinforced by repeating it with a different random initialization when the initial run does not succeed in finding an adversarial perturbation. Similar to

Algorithm 1 L_2 Projected Gradient Descent Attack

Input: example x, label y, model f, perturbation size ϵ, step size α
$\delta_0 \sim \text{Normal}(0, 1)$
$\delta_0 \leftarrow \epsilon \frac{\delta_0}{||\delta_0||_2}$
$i \leftarrow 1$
while $i \leq i_{max}$ **do**
    $\delta_i \leftarrow \delta_{i-1} + \alpha \frac{\nabla_{\delta_{i-1}} \mathcal{L}(f(x+\delta_{i-1}),y)}{||\nabla_{\delta_{i-1}} \mathcal{L}(f(x+\delta_{i-1}),y)||_2}$ ▷ calculate adversarial direction
    $\delta_i \leftarrow \epsilon \frac{\delta_i}{||\delta_i||_2}$ ▷ enforce $\|\delta_i\|_2 \leq \epsilon$
    $i \leftarrow i + 1$
end while
return $\delta_{i_{max}}$ ▷ the perturation

Algorithm 2 L_∞ Projected Gradient Descent Attack

Input: example x, label y, model f, perturbation size ϵ, step size α
$\delta_0 \sim \text{Uniform}(-\epsilon, \epsilon)$
$i \leftarrow 1$
while $i \leq i_{max}$ **do**
    $\delta_i \leftarrow \delta_{i-1} + \alpha \text{sign}(\nabla_{\delta_{i-1}} \mathcal{L}(f(x + \delta_{i-1}), y))$ ▷ calculate adversarial direction
    $\delta_i \leftarrow \text{clip}(\delta_i, -\epsilon, -\epsilon)$ ▷ enforce $\|\delta_i\|_\infty \leq \epsilon$
    $i \leftarrow i + 1$
end while
return $\delta_{i_{max}}$ ▷ the perturation

FGM, PGD also belongs the set of loss-based attacks which have a form similar to that in eq. (2.8).

Next, we will introduce a distance-based attack called DeepFool [57] which follows the form of eq. (2.7). This attack attempts to generate adversarial examples by iteratively perturbing the input data. It seeks to find the closest adversarial class and perturb the input along that direction so as to cause a misclassification. The distance to the decision boundary of the target model is measured by a linear approximation using the gradient. The basic steps of the DeepFool algorithm are stated in algorithm 3, which can be repeated until the maximum iteration is reached or the misclassification is achieved.

Instead of relying on a fixed perturbation budget like that in FGM and PGD, DeepFool focuses on minimizing the distance for each step while ensuring the step is taken in an adversarial direction. As a result, the found perturbation might be perceptible as there is no explicit restriction on the size of perturbation. One advantage of distance-based attacks is that they do not require a predefined epsilon value. Another example of a distance-based attack is the C&W attack [9], but it is not applied in this thesis we suggest that the reader peruse the article.

Algorithm 3 DeepFool Attack

Input: example x, label y, model f
$x_0 \leftarrow x$
$i \leftarrow 1$
while $\arg\max_j f(x_{i-1})_j = y$ **do**
    $k_i \leftarrow \arg\min_{j \in \mathcal{C} \setminus y} \frac{|f(x_{i-1})_j - f(x_{i-1})_y|}{||\nabla_{x_{i-1}} f(x_{i-1})_j - \nabla_{x_{i-1}} f(x_{i-1})_y||_2}$ ▷ find closest adversarial class
    $x_i \leftarrow x_{i-1} + \frac{|f(x_{i-1})_{k_i} - f(x_{i-1})_y| (\nabla_{x_{i-1}} f(x_{i-1})_{k_i} - \nabla_{x_{i-1}} f(x_{i-1})_y)}{||\nabla_{x_{i-1}} f(x_{i-1})_{k_i} - \nabla_{x_{i-1}} f(x_{i-1})_y||_2^2}$ ▷ apply the update
    $i \leftarrow i + 1$
end while
return $x_{i-1} - x$ ▷ the perturation

2.2.5 Measures on neural network robustness

Measuring the adversarial robustness of deep neural networks is challenging due to their nonlinear nature. Therefore it has not yet been standardized and it is an active area of research various metrics have been proposed. Some commonly used metrics include:

- The attack success rate: It measures the percentage of perturbed examples that successfully fool the model, that is they are adversarial. A higher success rate means a less robust network.
- Robust accuracy: This is the accuracy of the model on the perturbed examples. A higher robust accuracy means a greater resilience to adversarial attacks. It is equivalent to one minus the success rate.
- The magnitude of the adversarial perturbation: The size of perturbation is usually measured using some norms which reflect the magnitude of perturbations introduced by adversarial attacks. Some commonly used options are the Euclidean norm and infinity norm and a few less-used options are the zero norm which represents the number of modified pixels, and the absolute norm which measures the sum of the absolute differences. The magnitude is usually aggregated over a set of examples by taking the mean or the median perturbation magnitudes.
- Transferability: This measures the ability of perturbed examples generated on a model to fool another model. A high transferability implies a vulnerability that can be exploited by attackers.

Measuring the network adversarial robustness helps us to evaluate the effectiveness of defense mechanisms and develop robust models that are resilient to adversarial attacks.

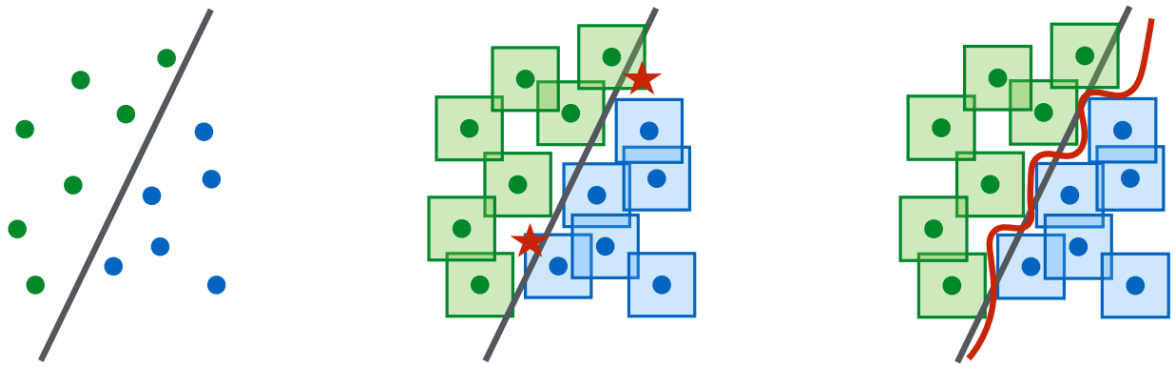

Figure 2.1: *An illustration of a standard(left) and a robust(right, red curve) network decision boundary taken from [46]. In the middle, adversarial examples of a standard network are denoted by red stars. The robust model shows the result of adversarial training where the perturbed images were generated using the infinite norm(squares).*

2.2.6 Adversarial training

Many different defense strategies have been proposed since the seminal paper of Szegedy et al [71], but only a few techniques have survived [73]. Even fewer defenses are scalable to large problems. Currently, the most effective and somewhat scalable methods for improving neural networks resilient to adversarial examples are based on adversarial training.

Adversarial training employs a perturbed input during the training to improve robustness to certain kind of attacks. It was first proposed in [46], where they introduced a modified network loss function like that given in eq. (2.13).

$$\min_{\theta} \mathcal{L}(f) = -\frac{1}{n}\sum_{i=1}^{n} \max_{\delta} \sum_{j=1}^{|\mathcal{C}|} (e_{y_i})_j \cdot \log \text{softmax}(f(x_i + \delta))_j \tag{2.13}$$

The inner maximization is estimated by some attack algorithm such as PGD. It can be also viewed as a min-max game between the network and the attacker, where the network seeks to minimize its error on the perturbed input and the attack algorithm attempts to generate a harmful perturbation during the training phase. An illustration of adversarial training is given in fig. 2.1.

It is worth pointing out that adversarial training is not a foolproof solution and it does not completely eliminate the existence of adversarial examples. It can significantly improve the model's resilience to common adversarial attacks, but determined attackers can still find ways of creating more challenging adversarial examples.

Chapter 3

Adversarial Robustness of Linear Models

Many machine learning models are sensitive to adversarial input, meaning that very small but carefully designed noise added to correctly classified examples may lead to misclassification. The reasons for this are still poorly understood, even in the simple case of linear models. Here, we study linear models and offer a number of novel insights. We focus on the effect of regularization and dimensionality. We show that in very high dimensions adversarial robustness is inherently very low due to some mathematical properties of high-dimensional spaces that have received little attention so far. We also demonstrate that—although regularization may help—adversarial robustness is harder to achieve than high accuracy during the learning process. This is typically overlooked when researchers set optimization meta-parameters.

The chapter: It starts with an introduction and review of the related works, then the role of dimension and regularization is discussed from a robustness point of view. Next, the experimental results are presented and the final thoughts are summarized in Section 3.5.

3.1 Introduction

The high sensitivity of most machine learning models to adversarial examples was pointed out not long ago [23, 71]. A number of methods have been proposed to create better adversarial examples [9, 57] as well as to provide defense mechanisms against these [46, 74].

Here, we focus on the adversarial robustness of linear machine learning models. The theoretical basis of the problem is still lacking. Some results are known e.g. Fawzi et al. [19] offer bounds on robustness for the linear case based on the distance of classes, but their study is orthogonal to ours. Goodfellow at al. [23] suggested that higher-dimensional linear models are more sensitive because the same amount

of noise in each dimension can result in a larger Euclidean distance from the point simply due to the larger number of dimensions, provided the sign of the noise is the same as the sign of the value in the given dimension. However, we argue that the Euclidean distance is of limited interest simply because classes and data points in general will also have larger Euclidean distances from each other in higher dimensions.

In this chapter, we propose novel insights, which provide an alternative explanation to the adversarial sensitivity of linear models. We focus on the effect of regularization and dimensionality. We will show that in very high dimensions adversarial robustness is inherently very low due to the fact that a random hyperplane is very close to any data point. This property which has received little attention so far, is highly counter-intuitive.

We also point out that regularization has a profound effect on adversarial robustness. From the point of view of prediction accuracy and adversarial robustness the amount of regularization required will be different. We should add that the current practice of setting meta-parameters based only on prediction accuracy might result in very high sensitivity to adversarial examples. This is because the convergence of robustness is much slower than that of accuracy and because robustness requires stronger regularization.

We shall also provide a thorough experimental evaluation of our claims where we study the effect of dimensionality, regularization, and the interaction of these two factors. In this evaluation, we will use artificial datasets as well as a subset of the MNIST dataset.

3.2 Linear Models in High Dimensional Spaces

We are given a set of training instances of the form (x, y), $x \in \mathbb{R}^d$, $y \in \{0, 1\}$, and we are looking for a hyperplane $Pl(w) = \{z|\langle w, z\rangle = 0\}$ defined by $w \in \mathbb{R}^d$ such that $Pl(w)$ separates the data points with different labels. This plane is typically found via optimizing a loss function based on the examples and w. Model optimization typically starts with a random initial model, or, equivalently, an initial model that is independent of the optimal model. The following result implies that such a random model will be extremely close to *any* point in expectation, hence, it should also be very close to each instance. This highly unintuitive property implies that a random plane has a very high sensitivity to adversarial examples.

Proposition 1. *Let $w \in \mathbb{R}^d$ define a random plane $Pl(w) = \{z|\langle w, z\rangle = 0\}$. Let w_i $(i = 1, \ldots, d)$ be i.i.d. random variables with $P(w_i = -1) = P(w_i = 1) = 0.5$. Let $dist(\mathbf{1}, Pl(w))$ denote the distance between $Pl(w)$ and the point $\mathbf{1} = (1, \ldots, 1)$. Then we have $\lim_{d\to\infty} \mathbb{E}(dist(\mathbf{1}, Pl(w))) = O(1)$.*

Proof. We have $dist(\mathbf{1}, Pl(w)) = |\langle \mathbf{1}, w \rangle| / \|w\|_2 = \frac{1}{\sqrt{d}} |\sum_{i=1}^{d} w_i|$. Also, we have $\sum_{i=1}^{d} w_i \to \sqrt{d}\mathcal{N}(0, \sigma^2)$ due to the central limit theorem, where $\sigma^2 = 0.25$ is the variance of w_i. The mean of $|\mathcal{N}(0, \sigma^2)|$ is finite and it does not depend on d, so it is $O(1)$; thus $\mathbb{E}(\frac{1}{\sqrt{d}} |\sum_{i=1}^{d} w_i|) \to \frac{1}{\sqrt{d}} \sqrt{d} O(1) = O(1)$, which completes the proof. □

Note that there exists a plane for which the distance from $\mathbf{1}$ is $\sqrt{d} = O(\sqrt{d})$, namely when $w = \mathbf{1}$. However, according to the result above, a random plane is of distance $O(1)$ in expectation. The result is not specific to $\mathbf{1}$ because it is invariant to rotation. The striking consequence is that a random plane will result in a high sensitivity to adversarial examples, *no matter how the classes are positioned*. This means that the optimal plane in terms of distance is very special, regardless of the difficulty of the classification problem, so we suspect that this very special plane is hard to find during optimization. Our experimental results are consistent with this view.

3.3 Linear Models and Regularization

Here, we argue that regularization is closely related to the geometric properties outlined in Section 3.2. Assuming n examples (x_i, y_i), $x_i \in \mathbb{R}^d$, $y_i \in \{0, 1\}$, $i = 1, \ldots, n$, let us now consider logistic regression where the goal is to approximate the data using the logistic function $y \approx \sigma(w^T x + b) = 1/(1 + e^{-w^T x + b})$. This will lead to a linear separator defined by w and b and a logistic probability approximation as a function of the distance from the separating hyperplane.

The loss function typically used to find the best model (that is, w and b) is the negative log likelihood function $L(w, b) = -\sum_{i=1}^{n} y_i \cdot log(\sigma(x_i; w, b)) + (1 - y_i) \cdot log(1 - \sigma(x_i; w, b))$. To handle noisy data, it is customary to add a regularization term to the loss function. Here we focus on the so-called L2 regularization: $L(w, b) + \alpha \|w\|_2^2$, where α is the regularization coefficient.

We would like to study the effect of regularization from the point of view of adversarial robustness. L2 regularization results in preventing the length of w from growing indefinitely. This in turn results in preventing the derivative of the model from growing indefinitely. To see this, consider the derivative $\sigma(a \cdot x)' = a\sigma(a \cdot x)\sigma(1 - a \cdot x)$. Clearly, increasing the length of w will make the logistic curve steeper. Without regularization, the model in practice becomes a step function so the loss function will simply attempt to minimize the number of misclassified examples. With regularization, all the examples will affect the orientation of the separating hyperplane.

This means that if regularization is not strong enough then noisy examples will have too much influence, forcing the hyperplane out of optimal position, which in turn will result in very high adversarial sensitivity according to the proposition provided in Section 3.2. Accordingly, we expect that for optimal robustness one will

have to use quite strong regularization.

3.4 Experimental Results

In order to evaluate the effect of dimensionality and regularization, we carried out a systematic experimental study. Now let us describe the experimental setup and the methodology in detail.

3.4.1 Binary Classification Problems

We will use two binary classification problems that are described below. The first dataset is a subset of the MNIST dataset [39] that includes two classes: 3 and 7 (also used by the authors of [23]). We will refer to this dataset as MNIST-73. It contains about 6000 samples per class. The raw pixel values were normalized to the range $[0, 1]$.

We will also use an artificial dataset called 2-GAUSS. The two classes are defined by the distributions $\mathcal{N}(\mathbf{1}, \Sigma)$ and $\mathcal{N}(\mathbf{0}, \Sigma)$, where $\mathbf{0}$ is the origin and Σ is the diagonal matrix $4\mathbf{I}$, hence the variance is $\sigma^2 = 4$, which is the same for each dimension. Note that the Euclidean distance of the class centers is $\sqrt{d}$, where d is the dimension. Here, we sampled 6000 instances per class.

For each dataset, $100/6 \approx 16.7\%$ of the data was separated to form a test set. In the preprocessing step, the training data values were translated so as to have a zero mean. The mean was estimated over the training set, and the test set was translated as well using this value.

To examine the effect of the dimensionality on adversarial robustness, we will use a range of input dimensions. The dataset 2-GAUSS can naturally be generated in any dimensions. The MNIST-73 examples were scaled using image processing algorithms. The original dimension of the images was 28×28. We performed preliminary tests with different interpolation methods (cubic, linear, nearest-neighbor) that gave similar results. Here, we applied the nearest-neighbor method.

3.4.2 Methodology

Our two main measures of interest are *accuracy* (i.e. the proportion of correctly classified examples) and the distance of the examples from the hyperplane normalized by the dimension $\sqrt{d}$. The latter measure characterizes the sensitivity to adversarial examples; namely the smaller the distance, the higher the sensitivity. Here, we normalize the distance by $\sqrt{d}$ for two reasons. First, it is more meaningful to measure sensitivity *relative* to the distance of the two classes, and the distance of the two classes grows with $\sqrt{d}$. Second, in the case of image data, this also means that we

characterize the sensitivity of each pixel, which is a more natural measure. We will call this measure the *normalized distance*.

We used ADAM [4] as our optimizer with a minibatch size of 32. Since we were interested in the actual optimal model (to avoid artifacts due to early stopping) we ran the algorithm with an extremely small stopping threshold of 10^{-10}. We will also include results with a 10^{-4} stopping threshold that is often used as a default. We can still study the effect of early stopping, since we record the convergence history as well. In our plots, we will indicate the regularization coefficient used in the case of $d = 28 \times 28$, however, for different dimensionalities, the regularization value was scaled proportional to d to make the strength of regularization in different dimensions comparable.

3.4.3 Results

Figure 3.1 shows some of the results of our experiments. The MNIST-73 results indicate that normalized distance and accuracy behave very differently in terms of regularization. Most importantly, one is normally interested in prediction performance, and the meta-parameters optimal for that purpose perform rather badly for adversarial robustness. To optimize the distance, it is good to have a regularization coefficient that is as large as possible, whereas accuracy displays a degrading trend with increased regularization. These observations hold true regardless of the problem dimension. In other words, in each dimension we see that they have almost the same values.

The 2-GAUSS problem behaves slightly differently because no noisy examples are added and because in high dimensions there is a wide linear separation margin between the classes and this grows with d. The optimal values for distance and accuracy are found in almost every case. However, we noticed that for low regularization values the optimizer struggles to find the optimum in high dimensions. For no regularization, even the smaller stopping threshold is insufficient to find the theoretically optimal model. This is because then the loss function is extremely flat. This effect is closely related to the dimensionality d, and the problem is more severe with larger values of d.

Let us also examine the dynamics of convergence during optimization, which is shown in Figure 3.2. Clearly, the convergence of distance is significantly slower than that of accuracy in each case. For the 2-GAUSS problem, this effect is more marked. With $\alpha = 10^{-4}$, due to the wide separation margin and relatively large weights, the loss function practically vanishes and gives only a very weak signal to the optimizer, while the accuracy attains its optimum quite quickly.

With the MNIST-73 dataset we see there is a local optimum for distance when no regularization is applied. This is due to the length of the parameter vector w gradu-

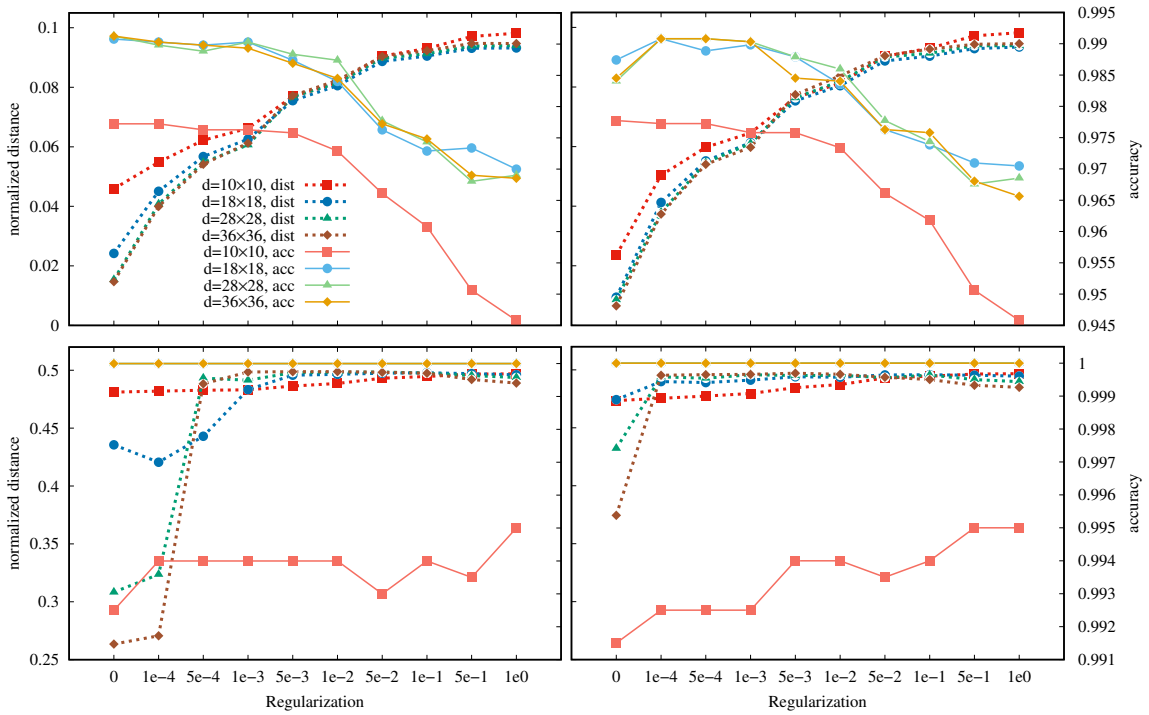

Figure 3.1: *Normalized distance and accuracy as a function of regularization coefficient and dimension for the* MNIST-73 *dataset (top) and the* 2-GAUSS *dataset (bottom), and stopping threshold* 10^{-4} *(left) and* 10^{-10} *(right).*

ally increasing. With the 2-GAUSS dataset we have no noisy examples that could make the model go in the wrong direction as w grows due to the lack of regularization, so this effect is not so marked.

3.5 Conclusions

In this chapter, we demonstrated that even in the case of simple binary classification problems with linear models, the adversarial problem is real and it strongly depends on regularization and the less obvious properties of high-dimensional spaces. We presented an experimental evaluation where we showed that the optimal regularization strength is very different for adversarial robustness and prediction accuracy, and that the convergence of adversarial robustness is much slower than that of the accuracy metric. Also, in higher dimensions an overly weak regularization setting might result in a significantly harder optimization problem in some cases.

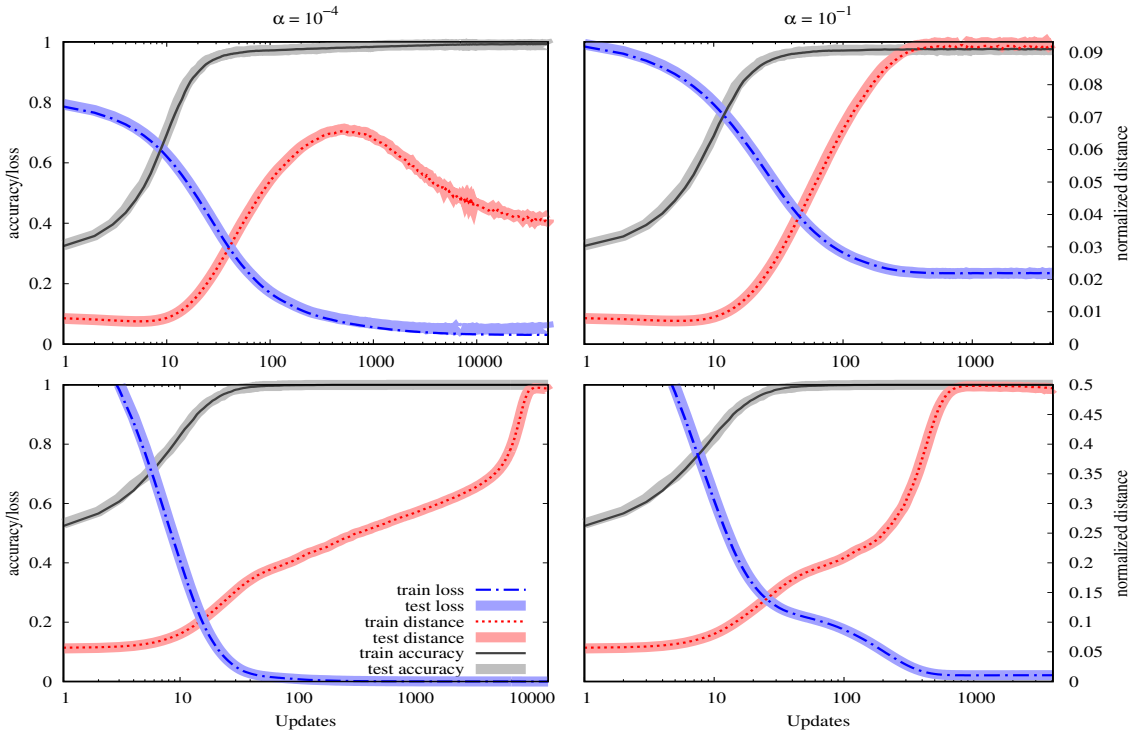

Figure 3.2: *Convergence of normalized distance and accuracy in $d = 28 \times 28$ dimensions for the* MNIST-73 *dataset (top) and the* 2-GAUSS *dataset (bottom), with regularization coefficient $\alpha = 10^{-4}$ (left) and $\alpha = 10^{-1}$ (right).*

Chapter 4

Adversarial attacks on model sets

Machine learning models are vulnerable to very small adversarial input perturbations. In this chapter, we study the question of whether the list of predictions made by a list of models can also be changed arbitrarily by a single small perturbation. Clearly, this is a harder problem since one has to simultaneously mislead several models using the same perturbation, where the target classes assigned to the models might differ. This attack has several applications over models designed by different manufacturers for a similar purpose. One might want a single perturbation that acts differently on each model; like only misleading a subset, or making each model predict a different label. Also, one might want a perturbation that misleads each model the same way and thereby create a transferable perturbation. Current approaches are not applicable for this general problem directly. Here, we propose an algorithm that is able to find a perturbation that satisfies several kinds of attack patterns. For example, all the models could have the same target class, or different random target classes, or target classes designed to be maximally contradicting.

The chapter: It has two parts. Each part starts with an introduction and review of the related works, then the attack algorithm and problem definition are described. Next, the experimental results are presented and the final thoughts are summarized in Section 4.3.

In section 4.1, we introduce an iterative heuristic algorithm inspired by the DeepFool attack. We evaluate our method over the MNIST and CIFAR-10 data sets. We show that it can find feasible multi-model adversarial perturbations, and that the magnitude of these perturbations is similar to the single model case.

In section 4.2, we generalize our algorithm and evaluated using three model sets consisting of publicly available pre-trained ImageNet models of varying capacity and architecture. We demonstrate that, in all the scenarios, our method is able to find visually insignificant perturbations that achieve our target adversarial patterns.

4.1 Model set attacks on MNIST and CIFAR-10

4.1.1 Introduction

Here, we propose and study a more general version of this problem, where we are given more than one model and an example. For each model, we specify whether the given model should correctly classify the example or predict any wrong label or predict a fixed specific label. This way, all the models specify a constraint on the desired perturbation. We assume a white box scenario where all the models are fully known.

This formulation allows for a wide variety of adversarial constraints on a given model set. Here, we focus on two patterns of constraints with interesting applications. In the first case, we wish to find an adversarial example such that all the models predict the same wrong label. This is related to the problem of finding adversarial examples for ensembles of models in the hope that these examples would also fool additional unseen black box models [44, 60, 75]. Known methods rely on the traditional setup where the ensemble is treated as a single model that has to be fooled. Unlike our approach, it always allows for multiple models in the ensemble to predict the correct label or even other inconsistent labels, as long as the ensemble decision is fooled.

The second pattern we study involves fooling a single model from the set, while making sure the rest of the models keep predicting the correct label. To the best of our knowledge, this is a novel scenario. In a sense, this is the inverse of the transferability problem, where we are looking for adversarial perturbations that do *not* transfer to other models, in a well-controlled manner. An interesting application is when an attacker wants to fool the product of a specific manufacturer, while making sure all the other products work correctly.

Our multi-constraint adversarial problem cannot be tackled with existing attack approaches directly. We propose an iterative optimization algorithm inspired by the DeepFool method [57]. We evaluate our method over the MNIST and CIFAR-10 data sets. We show our approach can find feasible multi-model adversarial perturbations, and that the magnitude of these perturbations is similar to the single model case.

4.1.2 Algorithm

Let us first introduce our notations. We assume a set of multi-class models $f_1, \ldots, f_m$ where $f_i : \mathbb{R}^d \to \mathbb{R}^C$. The models have C outputs that correspond to the possible class labels. The classification of a given input x by a model f_i is given by $k_i(x) = \arg\max_j f_{i,j}(x)$, where $f_{i,j}$ is the jth output dimension of f_i. We are looking for adversarial examples such that a given subset of the models is fooled while the rest of the models are not.

Algorithm 4 Multi-model adversarial perturbation

Input: example x, targeted models $\mathcal{F}_t$, protected models $\mathcal{F}_p$
Assumption: $\forall f_j \in \mathcal{F}_t \cup \mathcal{F}_p : k_j(x) = \hat{c}$, where $\hat{c}$ is the correct class of x
$x_0 \leftarrow x$
$i \leftarrow 0$
while $i < i_{max}$ and $[\exists f_j \in \mathcal{F}_t : k_j(x_i) = \hat{c}$ or $\exists f_j \in \mathcal{F}_p : k_j(x_i) \neq \hat{c}]$ **do**
    $r_t \leftarrow$ getStep$(x_i, \{$every class label except $\hat{c}\}, \mathcal{F}_t)$
    $r_p \leftarrow$ getStep$(x_i, \{\hat{c}\}, \mathcal{F}_p)$
    $r \leftarrow r_{\arg\max_{i \in \{t,p\}} \|r_i\|_2}$ ▷ the larger of r_t and r_p
    $x_{i+1} \leftarrow x_i + r$
    $i \leftarrow i + 1$
end while
return x_i ▷ the perturbed input

The basic idea behind the algorithm comes from the DeepFool method [57], where we also implement a heuristic iterative optimization algorithm based on the first order approximations of the decision boundaries. However, unlike DeepFool, we deal with several models targeted simultaneously by several different attack patterns. Our algorithm is shown in Algorithm 6. We assume that we are given an example x that is classified correctly by all the models. The models in $\mathcal{F}_t$ have to be fooled while those in $\mathcal{F}_p$ must not be. The loop runs until this goal is met. Within the loop, we ask for two perturbation steps: one that fools all the models in $\mathcal{F}_t$ and one that makes sure that all of the models in $\mathcal{F}_p$ predict the correct label. We apply the one with the larger norm. The idea is that this way we first solve the harder problem and then gradually satisfy the rest of the constraints.

A single iteration step is computed by Algorithm 5. The goal is to find a perturbation for x such that all the models in $\mathcal{F}$ predict a common label $c^* \in \mathcal{C}$, where $\mathcal{F}$ and $\mathcal{C}$ are parameters of Algorithm 5. In this version we present the untargeted version of the algorithm where this common label is not given as input, it can be arbitrary. The idea behind the algorithm is that for each label and each model we compute one potential targeted step for the iteration like the DeepFool iteration step [57]. We then pick the class label c^* that minimizes the maximal perturbation size over all the models. The maximal perturbation vector corresponding to this class label (where the maximum is taken over the models) is returned.

Note that there are several cases that we do not elaborate on here, for example, when some of the sets are empty. These can be handled in a natural way.

4.1.3 Experiments

We used the MNIST and CIFAR-10 data sets. The MNIST [39] data set consists of grayscale 28×28 images of handwritten digits, from 0 to 9. The CIFAR-10 [37] data

Algorithm 5 getStep

Input: example x, targeted classes $\mathcal{C}$, targeted models $\mathcal{F}$.
for $c \in \mathcal{C}$ **do**
  for $f_i \in \mathcal{F}$ such that $k_i(x) \neq c$ **do** ▷ models predicting other than c
    $\hat{w}_{i,c} \leftarrow \nabla f_{i,c}(x) - \nabla f_{i,k_i(x)}(x)$ ▷ $\approx$ direction to class c
    $w_{i,c} \leftarrow \hat{w}_{i,c} / \|\hat{w}_{i,c}\|_2$ ▷ $\approx$ normalized direction to class c
    $\delta_{i,c} \leftarrow |f_{i,c}(x) - f_{i,k_i(x)}(x)| / \|\hat{w}_{i,c}\|_2$ ▷ $\approx$ distance to class c
  end for
  $m_c = \arg\max_i \delta_{i,c}$ ▷ index of model with maximal distance to c
end for
$c^* \leftarrow \arg\min_c \delta_{m_c,c}$ ▷ class where maximal distance from $k_i(x)$ is minimal
return $\delta_{m_{c^*},c^*} \cdot w_{m_{c^*},c^*}$ ▷ perturbation towards making all $\mathcal{F}$ predict $c^* \in \mathcal{C}$

Table 4.1: *Properties of data sets*

	Training Set	Test Set	#features (d)	Consistently Classified
MNIST	60 000	10 000	784	7860/9180/9443
CIFAR-10	50 000	10 000	3072	4335

set contains 32×32 RGB color images representing 10 classes of objects. The main properties are shown in Table 4.1. The column "Consistently Classified" is explained later on. As preprocessing, the features were normalized in both data sets to the range $[0, 1]$.

We created four model sets to test our multi-model attack method. Three sets were created on MNIST. For each set, we fixed a network structure and used eight different regularization parameters to train eight different weight sets for the network. A fourth set was created on CIFAR-10, where we used one network structure and eight different regularization parameters.

The three networks for the MNIST data set had one hidden layer of sigmoid neurons of size 10, 100 and 1000, respectively, and a softmax output layer. On CIFAR-10 we trained a convolutional network with a shallow LeNet-like architecture. It uses two blocks of two convolutional layers followed by max-pooling, followed by two dense layers. Every layer has ReLU activation except the last one, which has a softmax activation. The dimensions of the first four convolutional layers of 3x3 filters, and the last two dense layers are (32x32x32), (30x30x32), (15x15x64), (13,13,64), 512, and 10. This results in 1,250,858 parameters.

We used ADAM [4] as our optimizer with a minibatch size of 128 and a stopping threshold of 10^{-10}. The eight regularization parameters were different for each network, as seen in Table 4.2. The reason is that we calibrated the range so that the last setting is overly regularized.

The properties of the individual models in the model sets are shown in Figure 4.1

Table 4.2: *Regularization coefficients used to create model set*

	0	1	2	3	4	5	6	7
MNIST 10	0	1e-8	1e-7	1e-6	1e-5	1e-4	1e-3	1e-2
MNIST 100	0	1e-9	1e-8	1e-7	1e-6	1e-5	1e-4	1e-3
MNIST 1000	0	1e-9	1e-8	1e-7	1e-6	1e-5	1e-4	5e-4
CIFAR-10	0	1e-5	1e-4	1e-3	2e-3	3e-3	4e-3	5e-3

(left). We define robustness as the L_2 norm of the untargeted adversarial perturbation found by DeepFool, normalized by $\sqrt{d}$, where d is the input dimension. We normalize with $\sqrt{d}$ because, in the case of image data, this way we characterize the sensitivity of each pixel irrespective of the resolution of the image, which is a more natural measure. Recall, that each input feature has a value in the range $[0, 1]$. We can see that robustness is increasing with regularization in all the cases, as expected.

The last column of Table 4.1 shows the number of test examples that were correctly classified by all the models in the respective model set, in the case of MNIST in the order of the 10, 100 and 1000 neuron hidden layers. Our multi-model attack method was evaluated on these consistent examples only. For each model set, we computed the adversarial perturbation for all these test examples in 9 different scenarios. This includes targeting each of the 8 different models in the model set individually while protecting the rest of the models (that is, $\mathcal{F}_t = \{f_i\}$ and $\mathcal{F}_p = \{f_0, \ldots, f_7\} \setminus \mathcal{F}_t$ for $i = 0, \ldots, 7$) as well as targeting all the models simultaneously (that is, $\mathcal{F}_t = \{f_0, \ldots, f_7\}$ and $\mathcal{F}_p = \{\}$). The number of iterations was limited by $i_{max} = 1000$. We used the models without the softmax activation, as was done in [57].

All the attacks were successful for all the examples, except in the case of MNIST with 10 hidden neurons, where the number of examples on which the attack was not successful ranged from 16 to 278 out of the 7860 consistent examples, which amounts to an 0.2% to 3.5% error rate. The results are shown in Figure 4.1 (right). The required number of iterations of our attack method is rather small. Surprisingly, the CIFAR-10 model set requires much fewer iterations than the MNIST sets despite it containing larger models.

Quite surprisingly, the multi-model perturbations are very similar in size to those of the single model (DeepFool) perturbations shown in the left column. This result was not anticipated, because the models differ only in the applied regularization coefficient, so they are fairly correlated, which would suggest that finding an adversarial example that fools one model but not the others is hard. However, in all the model sets, even for the most robust model (with large regularization) we can easily find an adversarial example with very small perturbation that does not fool the rest

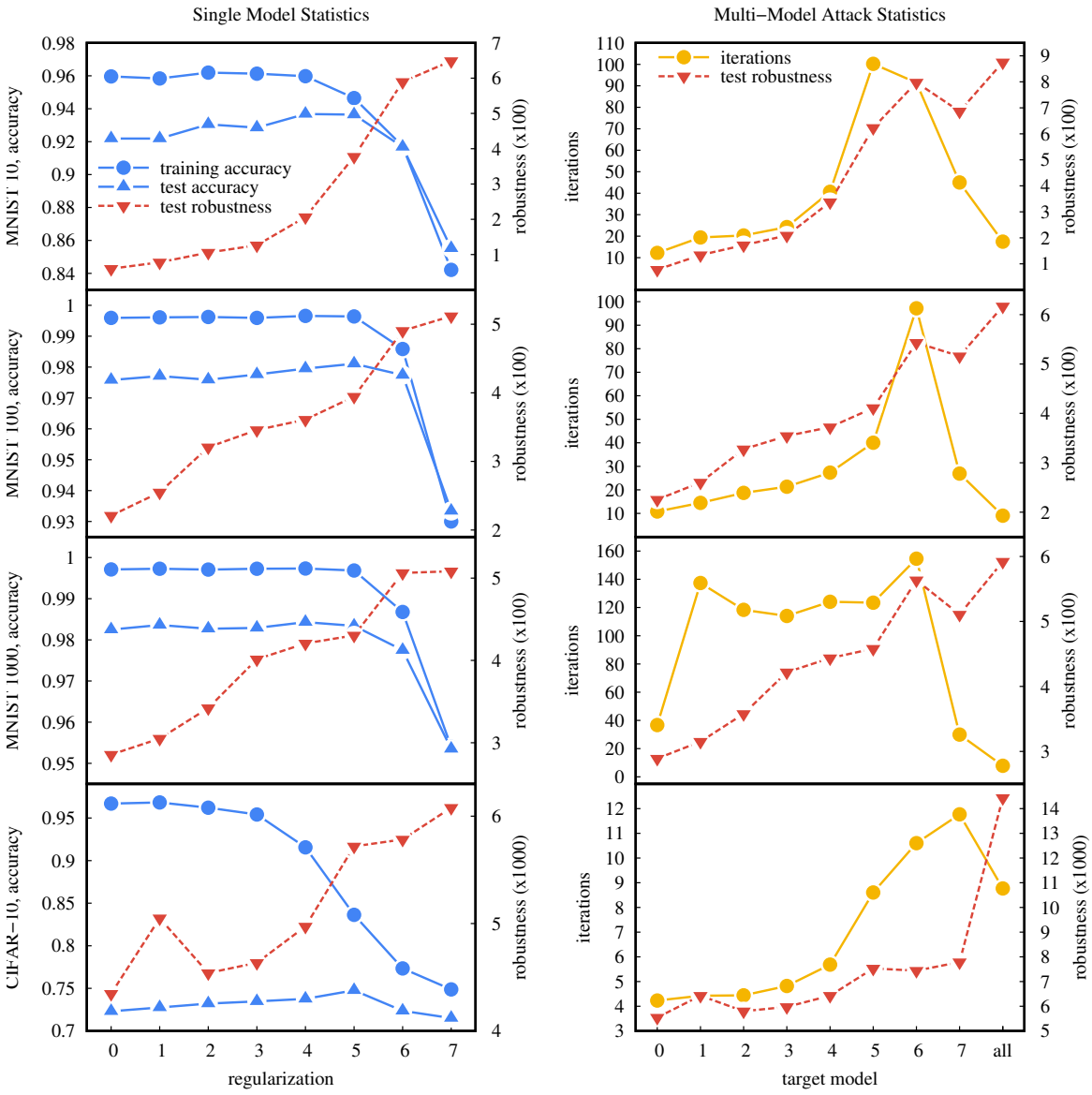

Figure 4.1: *Properties of the individual models (left), and multi-model attack statistics, where the horizontal axis indicates the targeted models $\mathcal{F}_t$ (right).*

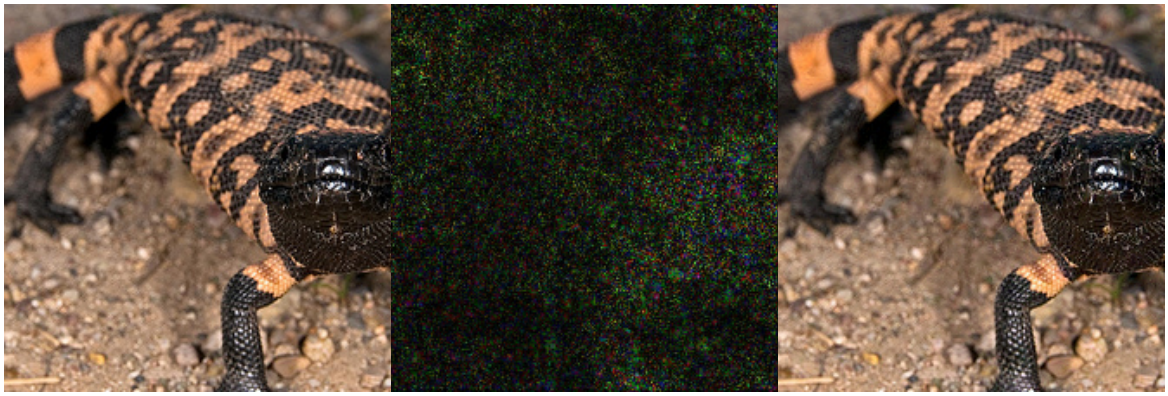

Figure 4.2: *Left: original image (prediction: 'Gila monster'); Middle: multi-model perturbation; Right: adversarial image (predictions: MobileNetV2: 'bison', MobileNet: 'balloon', NASNetMobile: 'pole', DenseNet121: 'acorn', DenseNet169: 'washbasin, hand-basin, washbowl, lavabo, wash-hand basin', DenseNet201: 'custard apple').*

of the models. In the CIFAR-10 dataset, the perturbation found for the "all targeted" case has a somewhat larger magnitude than that for the other attacks. However, it is still an extremely small (normalized) perturbation - amounting to less than 1.5% per input feature.

4.2 Model set attacks on ImageNet

4.2.1 Introduction

In this section, we study a more general version of this problem, where we are given a set of models trained on the same multi-class classification problem, as well as an input example. We assume that our set contains independently trained standalone models. That is, our focus is *not* on model ensembles. We could envisage the models to be products of different manufacturers. Nevertheless, our results are applicable to any set of models. We assume that the models are completely known; in other words, we are concerned with the white box scenario.

Our formulation allows for a wide variety of attacks on a given model set. The attacker may want every model to make the same mistake or (if, for example, hired by one of the manufacturers) to make different mistakes. The scenario where all of the models are made to predict the same wrong label is also related to the problem of finding adversarial examples for ensembles of models in the hope that these examples will also fool other unseen black box models [44, 60, 75]. In our version, however, every single model will make the given wrong prediction; not only the ensemble as a whole, where it might be enough to mislead, for example, the majority of the models (depending on the ensemble decision method).

An attacker may also be a nihilist who wants to achieve chaotic behavior. One way of doing this would be to make every model predict different random labels for the same input. One might think that a single perturbation that satisfies such harsh conditions should be extremely hard to find or might not even exist. Figure 4.2 shows such a perturbation, found by our algorithm. In a recent work, Song et al. investigated random target labels [68] in a multi-label classification setting. However, our setting is more challenging because we have a set of models trained on the same multi-class classification task. For this reason, the intersection of arbitrary classes is expected to be smaller than in a multi-label problem.

We propose a heuristic iterative algorithm to solve our multi-model adversarial problem. The algorithm is inspired by the DeepFool method [58] in that we also guide our search with the help of linear approximations of decision boundaries. We evaluated our method over the ImageNet dataset [63] using three sets of pre-trained models. We selected model sets so that we could evaluate different aspects of model diversity such as architecture and capacity. Model capacity is interesting to consider, since it has been shown that capacity alone could increase the robustness of individual models [46]. The diversity of the model architectures within the set is another important factor. For example, in [60], Pang et al. used the ensemble of diverse models to train a robust ensemble classifier. We expect that the diversity and the capacity of the models play a key role in robustness, and a set that is diverse along both dimensions is harder to attack.

We demonstrate that our algorithm can find feasible adversarial perturbations that fool all the models according to the given pattern in all the scenarios we examined. The adversarial target patterns include predicting the same wrong label, predicting different randomly selected labels, and predicting a set of labels that are designed to be maximally inconsistent. We found that model sets that include models with different architectures have a somewhat higher robustness than sets with similar architectures but with different capacity. However, in each case, the perturbations we found are imperceptible to the human eye.

In section 4.1, we made preliminary steps towards tackling the multi-model problem. Our original contributions here are the following:

1. Formalizing the targeted multi-model adversarial perturbation problem.

2. Proposing and evaluating several novel algorithm variants for solving the multi-model adversarial perturbation problem. These include the application of the pseudo-inverse of the constraint matrix, the application of the aggregated gradient instead of the maximal distance direction, the introduction of a step size limit to reduce the perturbation size, and the introduction of a candidate class list to manage problems with many classes.

3. Evaluating the algorithm over ImageNet using pre-trained publicly available

deep networks.

4. Evaluating several adversarial patterns including the random label assignment and two variants of 'hard' label assignments where the idea is to assign different target labels to different models so that the required perturbation is maximally inconsistent.

The outline of section 4.2 is as follows. In section 4.2.2 we describe our algorithms in detail. In section 4.2.3 we present several designs for creating adversarial target pattern. In section 4.2.4 we present our experimental evaluation.

4.2.2 Attacking Model Sets

Let us first introduce our notations. As mentioned in the Introduction, the problem is defined over a set of models and an example to perturb. The example is given in the form (x, y), where $x \in \mathbb{R}^d$ is the feature vector, and $y \in \mathcal{C}$ is its true label. The set $\mathcal{C}$ contains the class labels and $\mathcal{C}_{adv} = \mathcal{C} \setminus \{y\}$ is the set of possible adversarial target labels for x.

The set of multi-class models is given as $\mathcal{F} = \{f_1, \ldots, f_m\}$, where $f_i : \mathbb{R}^d \to \mathbb{R}^{|\mathcal{C}|}$. That is, the models have one output for every possible class label. We assume that all the models are trained over the same multi-class problem. The classification of a given input x by a model f_i is given by $k_i(x) = \arg\max_j f_{i,j}(x)$, where $f_{i,j}(x)$ is the jth element of the vector $f_i(x)$.

Although not strictly required by the framework, we note that in our implementation we used the unnormalized output (that is, the output without the softmax normalization) as our models f_i. When it is possible, this makes sense, because the normalization simply makes the gradients flatter without changing the ranking of the labels.

The predicted labels of the models are given by $K(x) = (k_1(x), \ldots, k_m(x)) \in \mathcal{C}^m$. In this section, we define the set $\mathcal{C}^m_{adv}$ as the set of all the possible adversarial patterns for the model set $\mathcal{F}$. Note that this way, we require here that all the models are assigned an adversarial label. This is not a severe restriction, one could also allow or even require some models to retain their true labels. An attacker will probably have further specific restrictions on the desired target patterns, so the set of desired adversarial patterns are given as a subset of all the patterns as $\mathcal{P} \subseteq \mathcal{C}^m_{adv}$. Formally, we are looking for a perturbation r^* that has minimal L_2 norm and makes the classifiers predict one of the desired patterns:

$$r^* = \arg\min_r \|r\|_2, \text{ subject to } K(x + r) \in \mathcal{P}. \tag{4.1}$$

The constraint of this optimization problem in (4.1) can be unrolled as a system of

equations for a given pattern $p \in \mathcal{P}$ as

$$\begin{aligned} k_1(x+r) &= p_1 \\ &\vdots \\ k_m(x+r) &= p_m. \end{aligned} \tag{4.2}$$

We can transform this system of equations into a system of inequalities using the fact that the equations require that the adversarial label correspond to the maximal element in the model's output. Accordingly, for the ith equation $k_i(x+r) = p_i$, let $\hat{k}_i(x) = \arg\max_{j \neq p_i} f_{i,j}(x)$ be the label that is the most likely label among the labels other than p_i. This gives us the equivalent system of inequalities

$$\begin{aligned} f_{1,\hat{k}_1(x+r)}(x+r) &\leq f_{1,p_1}(x+r) \\ &\vdots \\ f_{m,\hat{k}_m(x+r)}(x+r) &\leq f_{m,p_m}(x+r), \end{aligned} \tag{4.3}$$

where the equality holds when a point $x+r$ is exactly on the decision boundary between classes p_i and $\hat{k}_i(x+r)$. This can be considered as a generalization of the formalization of the DeepFool [58] algorithm where, instead of a system of inequalities, we have just a single inequality. In other words, in the model set there is only one model.

Due to the nonlinearities of the functions $f_{i,j}(x+r)$, this problem cannot be solved directly. As in the DeepFool algorithm, we substitute these functions with their first order approximations around x, that is, $f_{i,j}(x+r) \approx f_{i,j}(x) + \nabla f_{i,j}(x)^T \cdot r$, which results in the following system of inequalities:

$$\begin{aligned} \nabla(f_{1,\hat{k}_1(x)}(x) - f_{1,p_1}(x))^T \cdot r &\leq f_{1,p_1}(x) - f_{1,\hat{k}_1(x)}(x) \\ &\vdots \\ \nabla(f_{m,\hat{k}_m(x)}(x) - f_{m,p_m}(x))^T \cdot r &\leq f_{m,p_m}(x) - f_{m,\hat{k}_m(x)}(x), \end{aligned} \tag{4.4}$$

which is a set of linear constraints. The objective of the optimization problem is quadratic, which means that we have a quadratic programming (QP) problem.

A naive view would be to solve this problem with an efficient QP solver, and—since the linear constraints only approximate the actual constraints—iterate this procedure until the original set of constraints is satisfied. Using a QP solver, however, introduces certain problems. First, although efficient solvers are known, we might need to run them many times, which might become inefficient. Second, if there is no feasible solution for the linear approximation, the original problem might still have one, so we should create new approximations for each iteration. For these reasons, instead we used three heuristics to approximate the solution of the QP problem. We discuss these later on in more detail.

Algorithm 6 Multi-model adversarial perturbation

Input: example x, models $\mathcal{F}$, adversarial patterns $\mathcal{P}$
$x_0 \leftarrow x$
$i \leftarrow 0$
while $i < i_{max}$ and $K(x_i) \notin \mathcal{P}$ **do**
  for $p_k \in \mathcal{P}$ **do**
    $r_k \leftarrow$ approximateQP(x_i, p_k)
  end for
  $r \leftarrow r_{\arg\min_k \|r_k\|_2}$ ▷ r_k with the smallest norm
  $r \leftarrow \min(\eta/\|r\|_2, 1) \cdot r$ ▷ enforce $\|r\|_2 \leq \eta$
  $x_{i+1} \leftarrow x_i + r$
  $i \leftarrow i + 1$
end while
return x_i ▷ the perturbed input

We are now ready to present the pseudocode of the algorithm, represented as Algorithm 6. We assume that we are given an example x. The models in $\mathcal{F}$ have to predict one of the patterns in $\mathcal{P}$. Recall that if there are more than one pattern in $\mathcal{P}$, satisfying any of these patterns is considered a success. The outer loop runs until this goal is met (or we reach the maximal number of iterations).

The inner loop iterates through the patterns, and calculates the heuristic solution of the QP that is defined by the particular pattern. After calculating these approximate solutions r_k, the one with the smallest norm is selected. If the magnitude of this perturbation is larger than the allowed maximum step size η, then the perturbation is normalized. This step offers some protection to the algorithm in cases where the linear approximation has a large error. This perturbation size restriction technique is partly related to trust region methods. Here, we use a fixed η parameter that controls the size of the trusted region in a static manner. Changing η dynamically could offer further improvements [82].

Heuristics to Solve the Inner QP Problem

Let us now describe three implementations of the method APPROXIMATEQP that is used in the inner loop of Algorithm 6. Let us first normalize the inequalities in (4.4) by dividing both sides by $\|\nabla(f_{i,\hat{k}_i(x)}(x) - f_{i,p_i}(x))\|_2$. Let us introduce a new notation to represent the normalized inequalities. On the left hand side, let

$$w_i = \frac{\nabla(f_{i,\hat{k}_i(x)}(x) - f_{i,p_i}(x))}{\|\nabla(f_{i,\hat{k}_i(x)}(x) - f_{i,p_i}(x))\|_2} \tag{4.5}$$

and on the right hand side, let

$$f'_{i,p_i} = \frac{f_{i,p_i}(x) - f_{i,\hat{k}_i(x)}(x)}{\|\nabla(f_{i,\hat{k}_i(x)}(x) - f_{i,p_i}(x))\|_2}. \tag{4.6}$$

With these notations, equation i becomes $w_i^T \cdot r \leq f'_{i,p_i}$.

Note that the value f'_{i,p_i} represents the gap between the class that is currently predicted by model f_i and the desired adversarial target class p_i. Thus, we know that at the beginning of the attack $f'_{i,p_i} \leq 0$ and the equality holds exactly when the predicted class is already the target class. From now on, all the heuristics below will ignore the models where the target class is already predicted, and the algorithms will work only with those equations that still have a negative right hand side.

MAX Here, the idea is that we first identify the model that has the largest gap between its target class and the current class. We then approximate the minimal perturbation that closes this gap, that is, that 'fixes' this model. The idea is that this way we first solve the hardest model and then gradually adjust for the rest of the constraints. The approximation takes the linear approximation of the model, for which the minimal perturbation can be computed in closed form. The formulation of this heuristic is

$$\begin{aligned} j &= \arg\max_{1 \leq z \leq m} -f'_{z,p_z} \\ r_{max} &= -f'_{j,p_j} w_j. \end{aligned} \tag{4.7}$$

Note that this computation is very similar to that of DeepFool, since here we have a single model to consider (the one with the maximal gap), and on that model we essentially run a DeepFool step.

AVG Here, we compute a direction that takes into account all the models, instead of just choosing the one with the maximal gap. Recall, that the vector w_i represents the direction of the minimal perturbation required to 'fix' model i, and $\|w_i\|_2 = 1$ due to the normalization. We first compute the average of these directions, weighted by the gap values. We then return a perturbation vector that points in this average direction. The length of the returned vector is computed to be the maximum of the gap values. The reason is that this way we are guaranteed not to overshoot, because this step size is the minimal step size that potentially changes all the predictions. If we use a smaller step size, there will be at least one model that will not predict the target label. The formulation is

$$\begin{aligned} w_{avg} &= \textstyle\sum_{z=1}^{m} -f'_{z,p_z} w_z \\ j &= \arg\max_{1 \leq z \leq m} -f'_{z,p_z} \\ r_{avg} &= -f'_{j,p_j} \frac{w_{avg}}{\|w_{avg}\|_2}. \end{aligned} \tag{4.8}$$

PINV When we have only one model, the minimal perturbation is computed as one step of the Newton method, as done by DeepFool and by the MAX heuristic above as well. We could also try to generalize this idea to multiple models by taking the set of inequalities in (4.4) and demanding that equality holds in each inequality. Since the number of variables will be usually larger than the number of equalities, the system will be underdetermined. We can then still solve this set of equalities using the pseudo-inverse technique as applied in [68].

This heuristic will return a perturbation that points towards the intersection of all the decision boundaries, which may or may not be the smallest perturbation among those that satisfy (4.4). In pathological cases, this intersection might be extremely far from the optimal perturbation direction. In fact, there might not even be such an intersection. Still, we include this algorithm for comparison with [68]. Also this algorithm could potentially be fast (ignoring pathological corner cases), since in every iteration we try to solve every constraint.

Consider the matrix $W = (w_1, \cdots, w_m)^T$ and vector $F = (-f'_{1,p_1}, \cdots, -f'_{m,p_m})^T$. The system of equations is then $W \cdot r = F$, so the perturbation returned by the PINV algorithm is given by

$$r_{pinv} = W^+ \cdot F, \tag{4.9}$$

where $W^+ = (W^T W)^{-1} W^T$ (assuming W has full rank) is the pseudo-inverse of W. Note that it is known that r_{pinv} will be the minimum norm solution if there is a solution, and it will represent an approximation with minimal error if there is no solution [22].

Time complexity

The number of iterations of the algorithm depends on many factors and hyperparameters. We evaluate this experimentally later on. Here, we focus on the time complexity of a single iteration. To perform one iteration, the linear approximation in (4.4) needs to be calculated for every $p \in \mathcal{P}$. For each pattern p, we need to compute the prediction and the gradient of each model, which means that we need to propagate one forward pass and one backward pass through every network, if the models are feedforward neural networks. Thus, the cost of one iteration is $2 \cdot m \cdot |\mathcal{P}|$ network propagations. If $\mathcal{P}$ has only one element then the time complexity depends only on m (the number of models). Note that these calculations can be parallelized across the models as well as the patterns, and only the forward pass needs to be performed before the backward pass.

4.2.3 Adversarial Target Patterns

Our algorithm was formulated as a targeted attack, where the acceptable adversarial patterns are given in the pattern set $\mathcal{P}$. This set might contain a single pattern, which is similar to the classical targeted attack. Although it would be possible to formulate a modified algorithm for the untargeted case, one can achieve a similar effect by including several carefully selected patterns in the target set.

In the following, we describe and motivate our four different designs for the pattern set that we will use in our experimental evaluation. We assume that we are given a fixed input x and the patterns are designed as a function of this input. Our patterns include three targeted patterns and an untargeted one.

Some of the designs are based on a ranking of class labels $\mathcal{C}$, based on the output of all the models for a given input x. For a given input, each individual model defines a ranking of the class labels based on the ordering of the values of the corresponding output elements. The highest ranking class is predicted by the model. The idea is that we wish to identify those class labels that are 'similar' to the true class label and hence that are ranked high by most of the models. Further, we are also interested in those labels that are treated as very irrelevant by most of the models.

The ranking of the labels can be defined as a suitable aggregation of the rankings of the individual models. Formally, the ranking of a class label $c \in \mathcal{C}$ for a given example x and a classifier f_i is denoted by $q_{i,x}(c) \in \{0, \ldots, |\mathcal{C}| - 1\}$. In other words, $q_{i,x}(c)$ is the rank of label c in the ranking defined by $f_i(x)$. For the predicted class label we have $q_{i,x}(k_i(x)) = 0$. Note that any ranking aggregation could be used to determine the common ranking. Here we used the Rank Product [6] aggregation method, where the ranking of a label c is defined by the ordering of the geometric mean rankings

$$RP_x(c) = \prod_{j=1}^{m} q_{j,x}(c)^{1/m}, \quad \forall c \in \mathcal{C} \tag{4.10}$$

Random

$\mathcal{P}_{random}$ contains only one element, that is, it defines a targeted attack. This single pattern contains randomly generated adversarial class labels for all of the models. This attack should be hard because the perturbed input should end up in the intersection of the different unrelated classes of the different models.

Reverse

Our goal here is to define a target pattern that is even harder than the random pattern. $\mathcal{P}_{reverse}$ will also contain only a single pattern. To create this pattern, we used the ranking of the class labels described above. From this ranking, we selected the

label that ranks the lowest, namely the label that is the most irrelevant among the model set. We then demand that the pattern contain this label for all the models.

Diverse

As another attempt to define a pattern that is maximally hard, here $\mathcal{P}_{diverse}$ has one pattern that contains labels that rank low and, in addition, that are also inconsistent with each other. We again use the low end of the ranking of the labels but this time we consider the last 10 labels. Next, we compute the adversarial direction for this ten labels for every model, using a single linear approximation (DeepFool) step. We now have 10 directions for all the models. Out of this set, for every model we select the direction such that the set of selected directions over all the models form a maximally diverse set. We measured the diversity of a given candidate pattern with the help of the average cosine similarity of every pair of directions. The pattern with the lowest value is selected. In our evaluation, we performed an exhaustive search to find the most diverse pattern.

Consistent

Here, we define an untargeted attack. This case is exceptional for two reasons. First, we include more than one pattern in $\mathcal{P}_{consistent}$. Second, and most importantly, in this case the pattern set will be dynamic; that is, we will update the set of target patterns in every iteration using the method described here. For clarity of presentation, this step is not included in the pseudocode of Algorithm 6.

The patterns can just target the same adversarial label for every model. In other words, all the models are required to predict the same, adversarial label. We used the top 10 adversarial labels in each cycle. The ranking of the labels is updated in each iteration using the Rank Product method and the (already perturbed) input x_i.

4.2.4 Experiments

We did experiments on models trained over the ImageNet [63] dataset that contains RGB images with varying sizes. The images are labeled with one of 1000 class labels. In the preprocessing stage, a 224×224 image was cropped from the middle of every original image.

Table 4.3 lists the pre-trained models we used along with references. The table contains basic information about the architecture of the networks. To evaluate our multi-model algorithm, we created model sets using these individual models. The mobile set includes the three mobile networks. The members of the mobile set have a similar capacity, that is, they have a similar number of parameters. However,

Table 4.3: *Networks used in the evaluation*

Model	Parameters	Depth	Correctly Classified
MobileNetV2 [64]	3.5M	88	7153
MobileNet [30]	4.2M	88	7257
NASNetMobile [89]	5.3M	-	7832
DenseNet121 [32]	8.1M	121	7729
DenseNet169 [32]	14.3M	169	8095
DenseNet201 [32]	20.2M	201	8331

Table 4.4: *Model sets*

Model set	Correctly Classified
Mobile	53
Dense	74
All	52

they have rather different network architectures. MobileNet uses depthwise separable convolutions and MobileNetV2 uses residual connections as well. NasNetMobile contains non-human-designed blocks. The dense set contains the three variants of DenseNet. These models have a similar architecture but have a different capacity. Lastly, we also experimented with the union of the mobile and dense sets that contain all the six models. We will refer to this set as 'All'.

During our experiments, we evaluated the individual models as well as the three model sets. In our evaluation, we worked with a set of 10,000 randomly selected images taken from the training set for the individual models. The subset was the same for every model. Out of this set of examples, we removed those examples that were misclassified by the respective model. Table 4.3 shows the number of examples that were classified correctly. These examples form the evaluation set for the given model. Due to their larger cost, the model sets were evaluated on a smaller set, that contains 100 randomly selected examples taken from the training set. Here, we also used the same examples for every model set. Table 4.4 shows the number of examples that were classified correctly by all the members of the given set. These examples form the evaluation set for the given model set.

Our main figure of merit is *perturbation size*. We define perturbation size as the L_2 norm of the adversarial perturbation found, normalized by $\sqrt{d}$, where $d = 224 \times 224 \times 3$ is the input dimension. We normalize by $\sqrt{d}$ because in the case of image

data, this way we characterize the average perturbation of each pixel irrespective of the resolution of the image, which is a more natural measure. Recall, that each input feature has a value in the range $[0, 255]$.

The properties of the individual models can be seen in Figure 4.3. Note that for individual models, our algorithm is equivalent to DeepFool when $\eta = \infty$. All the attacks on all the correctly classified examples were successful, irrespective of the value of η. Clearly, the perturbation size increases with model capacity. When setting a smaller η, we get a smaller perturbation, but at the cost of more iterations. Interestingly, the effect of η is much stronger on NASNetMobile than on any other model. In this case, we can get a marked improvement over the DeepFool algorithm. This is interesting, because DeepFool is often used to compute an upper bound on the sensitivity of models, and in this case this upper bound could be improved significantly. In general, setting a smaller η improves the perturbation size in each case to some extent.

Next, we evaluated our multi-model algorithms on the mobile set to test the effect of η, and the different QP solver heuristics. The results are shown in Figure 4.4. We specified $i_{max} = 10,000$. Like before, all the attacks were successful, although the required number of iterations is significantly larger than in the case of the individual models. A lower η results in a smaller perturbation and an increased iteration number.

The PINV and the AVG heuristics need fewer iterations, as we expected. This is because these heuristics take into account all the models in the model set in each iteration. We can see that PINV results in the largest perturbation, again, as expected, although the difference from the alternative heuristics is surprisingly small.

For the other two model sets (dense and all), we performed our experiments with a fixed $\eta = 10$ maximum step size because, based on Figure 4.4, it offers a good compromise between perturbation size and cost. The perturbation size and the number of iterations over the three model sets and the four attack patterns can be seen in Figure 4.5. The three heuristics are shown separately. All the attempted attacks were successful. The perturbation size of every model set for the four patterns is larger than that for the individual models. Also, significantly more iterations are required.

The consistent pattern leads to the smallest perturbation size and the random and reverse patterns give a larger perturbation, as expected. Surprisingly, the diverse pattern is not consistently harder than the reverse pattern. In terms of the number of iterations, there are significant differences among the attack patterns. The attacks that are designed to be hard always require orders of magnitude more iterations than the easiest (untargeted) attack.

It is interesting that the mobile set requires a larger perturbation than the dense set. We expected the opposite, because the attack seemed to be harder if the models

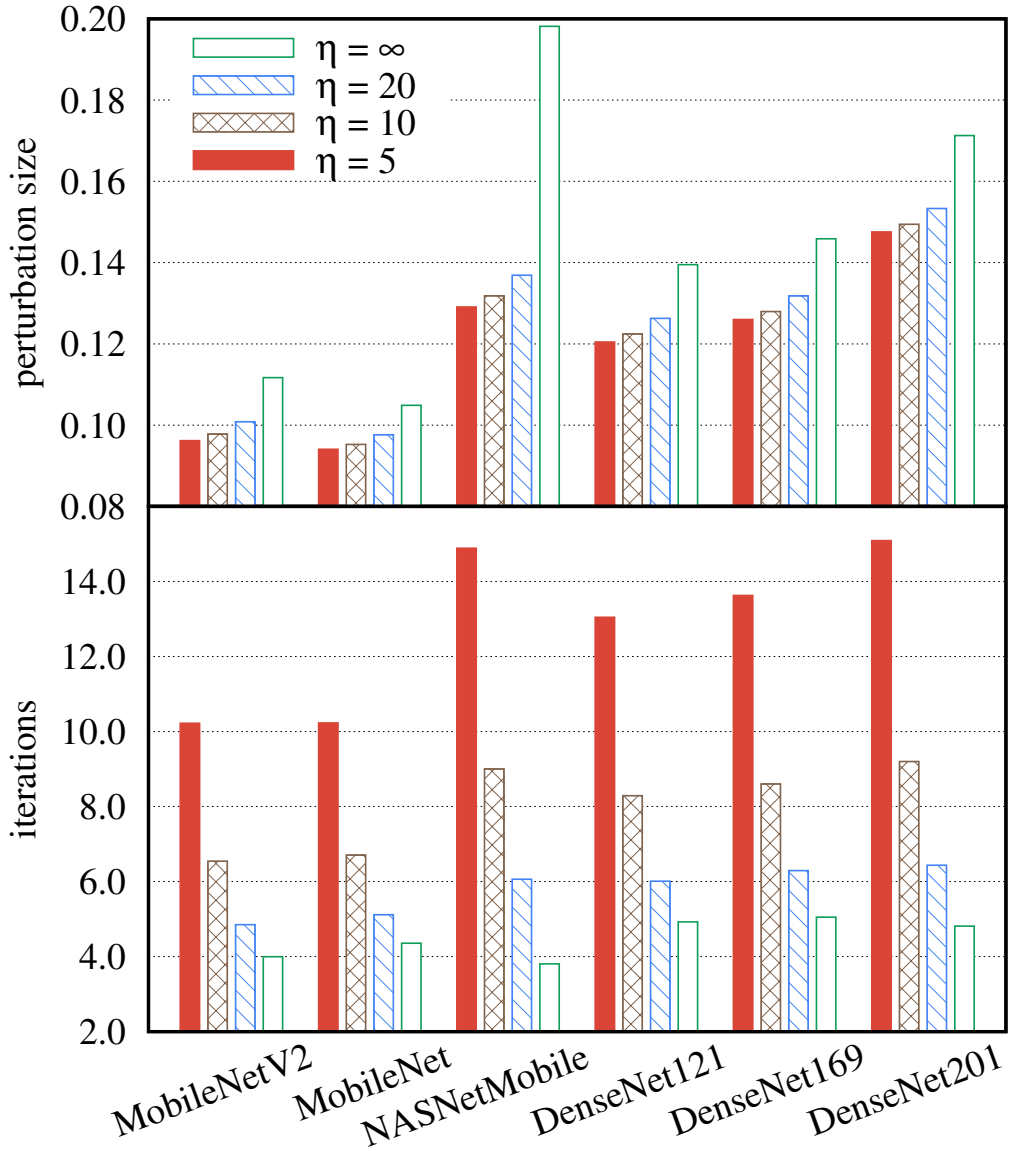

Figure 4.3: *Perturbation size and number of iterations for the individual models as a function of maximum step size (η). The models are shown in the order of increasing capacity.*

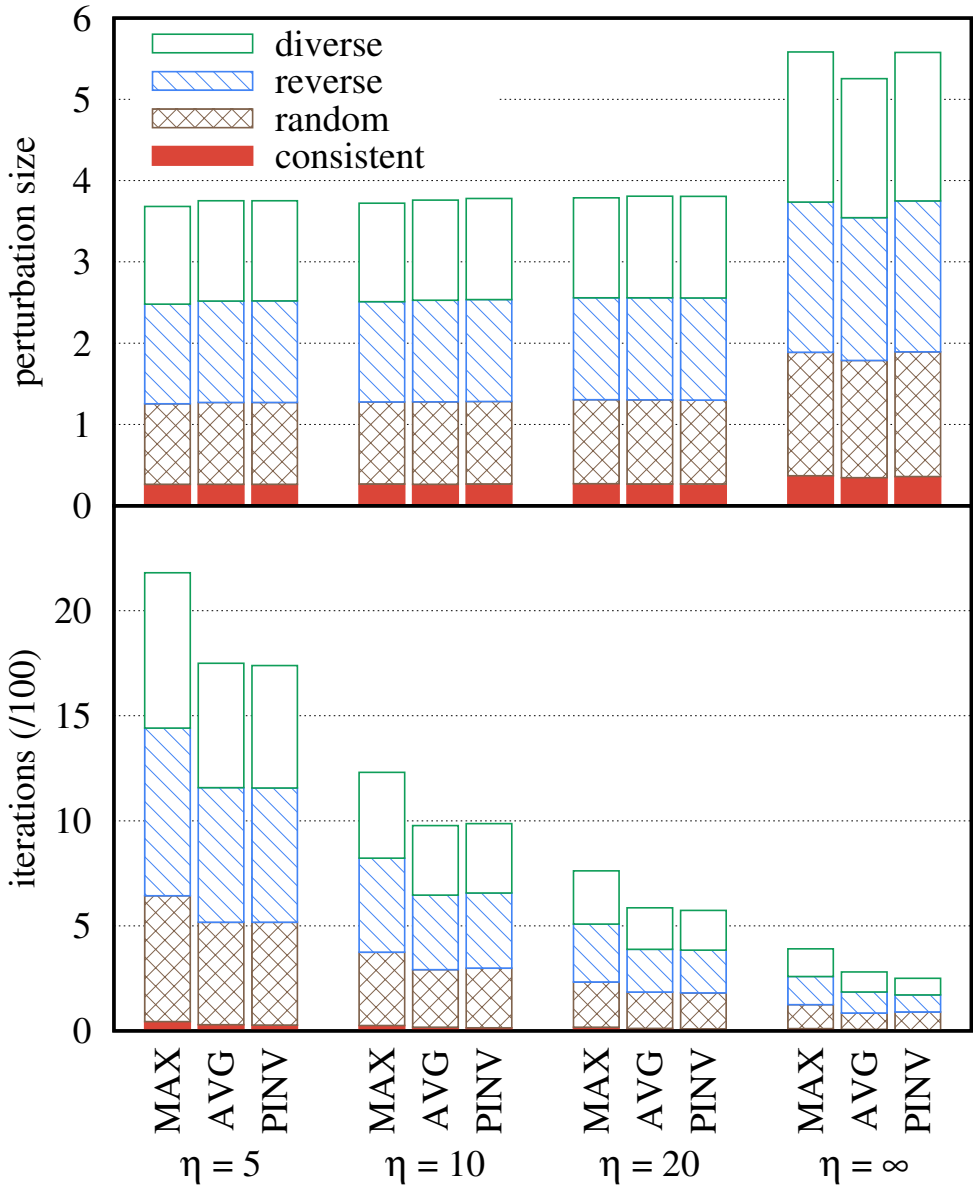

Figure 4.4: *Perturbation size and number of iterations for the mobile set as a function of the maximum step size (η) and QP solver heuristic.*

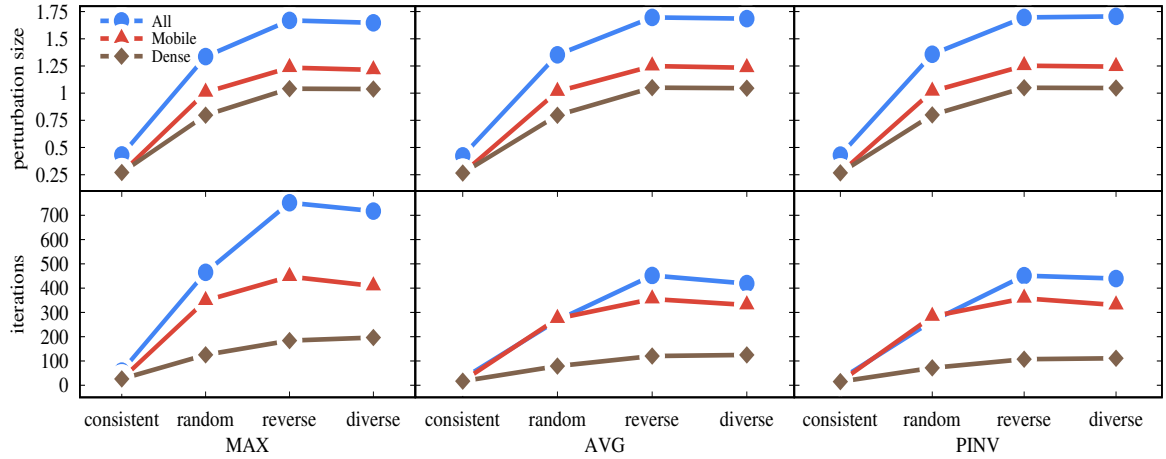

Figure 4.5: *Perturbation size and iterations for all the four attack patterns and the three QP solver heuristics ($\eta = 10$).*

Table 4.5: *Untargeted individual model attack with $\eta = 10$*

Model	Mean iterations
MobileNetV2	6.5
MobileNet	6.7
NASNetMobile	9
DenseNet121	8.3
DenseNet169	8.6
DenseNet201	9.2

in the set are similar, due to the requirement of unrelated classes having a non-empty intersection. Combining all the six models further increases the perturbation size, but it is still less than 1% on average for an input feature. As we will see later, such a perturbation is still imperceptible.

To better illustrate the difference between the number of iterations required to compute the attack for individual models and model sets, we present the average iteration number for these two cases in tables 4.5 and 4.6. We illustrate the untargeted case, that is, the attack pattern was $\mathcal{P}_{consistent}$ for the model set, and a simple untargeted DeepFool attack was used for the individual attacks. From these tables, we see that the required number of iterations appears to be proportional to the number of models in the set.

To illustrate the perturbations that our algorithm creates, we include an example for all the combinations of our three model sets and four attack patterns in figures 4.6, 4.7, 4.8 and 4.9. The examples we include are the ones that have the largest

Table 4.6: *Untargeted model set attack with* $\eta = 10$

Model set	MAX mean iter	AVG mean iter	PINV mean iter
Mobile set	25.9	16.9	14.9
Dense set	25.9	16.6	14.6
All	57.9	26.9	24.6

perturbation among the example inputs. The applied heuristic was MAX, and we set $\eta = 10$. In all the images, the top row contains the original images, the middle row contains the perturbation and the perturbed images are in the bottom row. In the patterns, the order of the models is the same as in Table 4.3.

4.3 Discussion and concluding remarks

In this chapter, we proposed an iterative algorithm to find small adversarial perturbations that fool a given set of models simultaneously in a given pattern. This problem formulation has several applications including the generation of transferable adversarial examples, as well as *non-transferable* examples that target only a specific model and ensure that the other models are safe. Or we might generate a single perturbation such that all the models in a given model set predict specified, different classes.

In section 4.1, we introduced an initial version of the algorithm which applies the first-order approximation of the decision boundaries used in the DeepFool method. We evaluated the algorithm on a number of model sets over MNIST and CIFAR-10 and generated transferable as well as non-transferable examples. We found that the algorithm consistently produces small perturbations in all the cases we examined. Perhaps the most interesting result is that small adversarial perturbations are present even when a non-transferable adversarial example was generated for the most robust model in the set, despite the fact that the models differed only in the regularization coefficient.

In section 4.2, we show a generalized version of the method which has many interesting applications, it is still able to generate transferable adversarial examples as well as generating a single perturbation such that all the models in a given model set predict specified, different classes. The latter scenario allows us to explore the decision boundaries of the model set from a new perspective.

The algorithm can be regarded as a generalization of the DeepFool method to model sets. Also, we improved the DeepFool algorithm itself by adding the step size parameter. We evaluated our algorithm on three model sets using four attack patterns over the ImageNet database. We found that the algorithm produces small

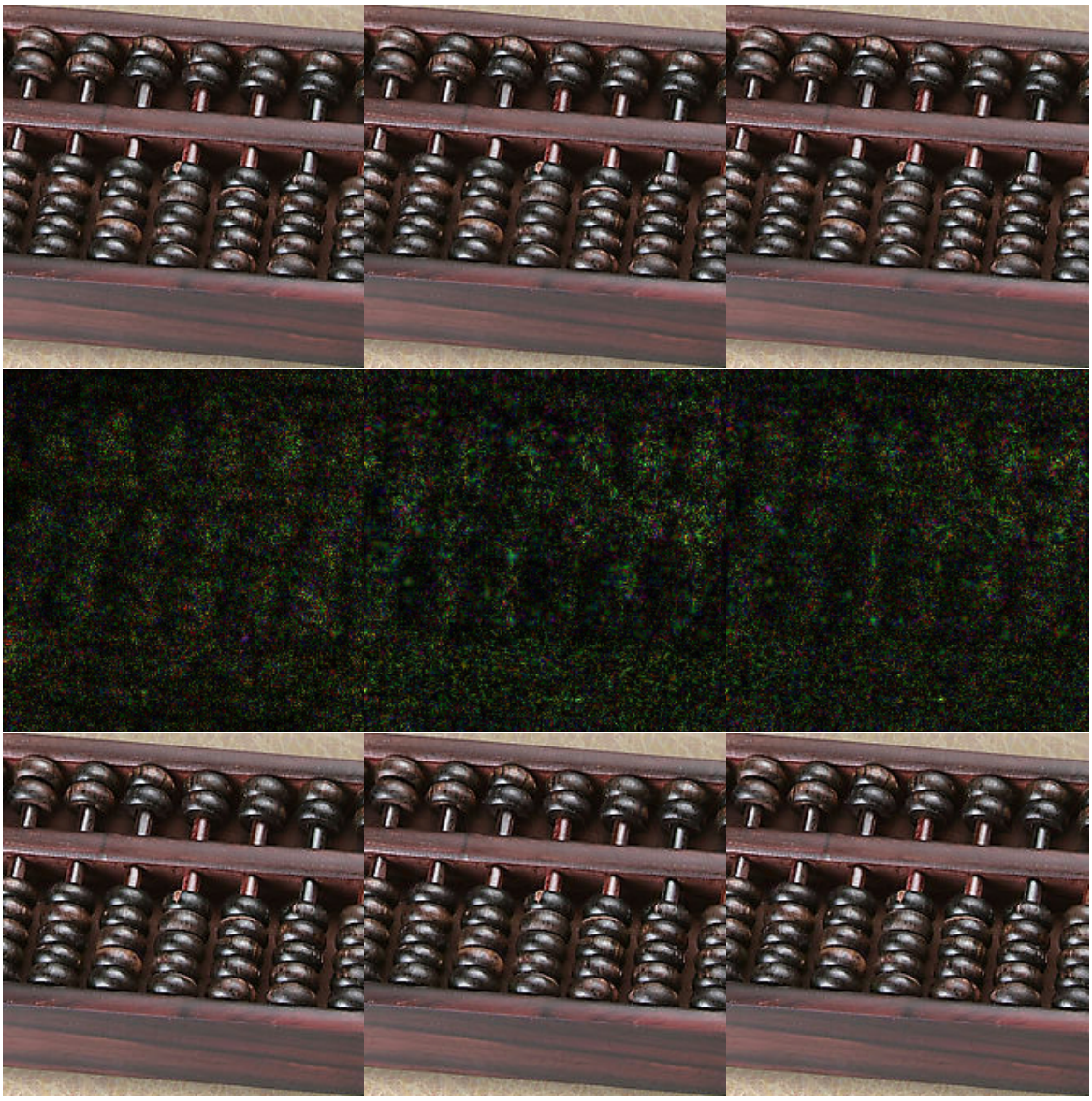

Figure 4.6: *The consistent attack pattern over the mobile set (abacus → dumbbell), dense set (abacus → corn) and all the models (abacus → dumbbell).*

Figure 4.7: *The random attack pattern over the mobile set (crib → [llama, thunder_snake, Norwich_terrier]), dense set (Australian_terrier → [cornet, lycaenid, malinois]), and all the models (abacus → [centipede, Pembroke, Band_Aid, bow-tie, EntleBucher, coyote, poncho]).*

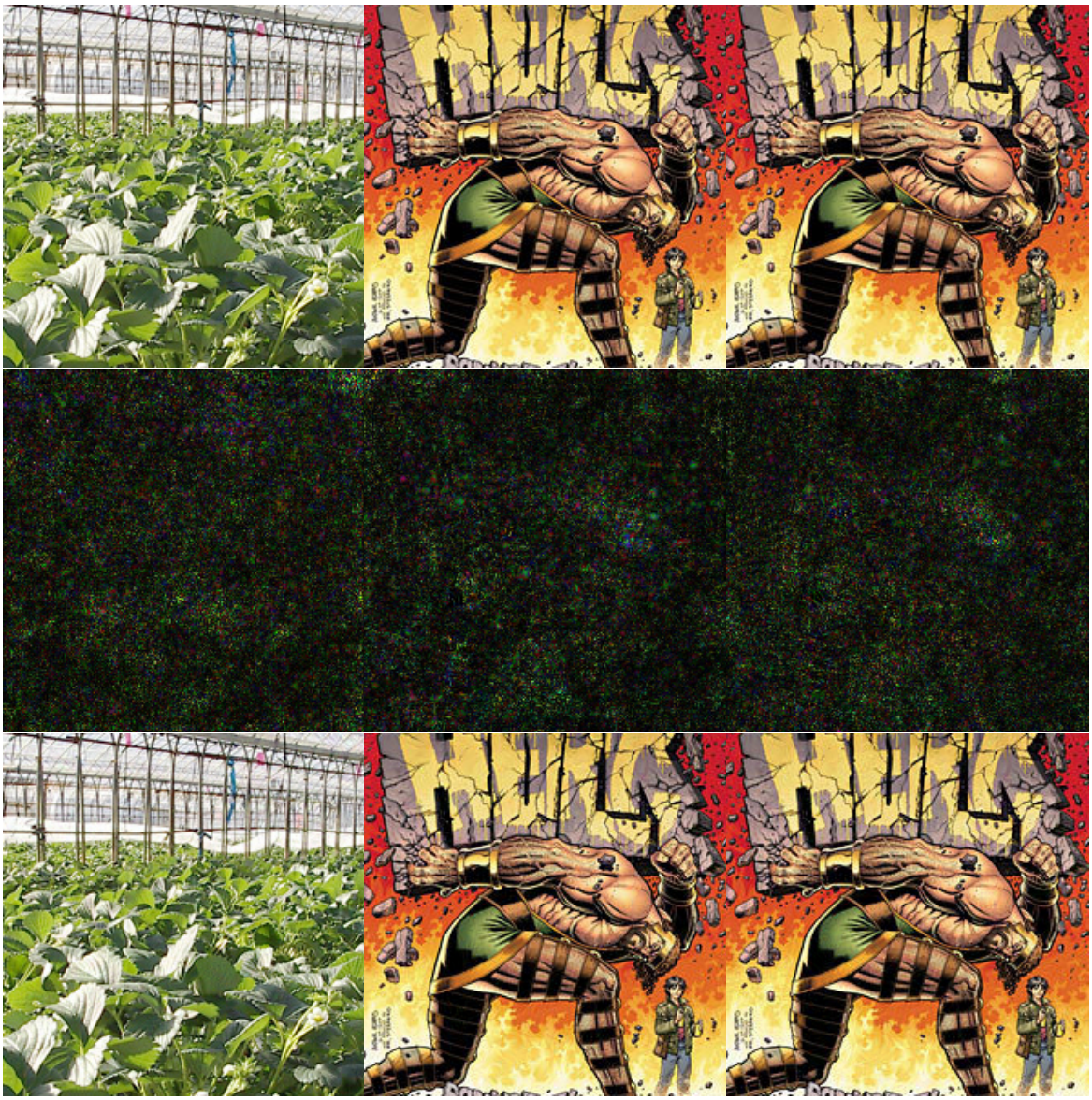

Figure 4.8: *The reverse attack pattern over the mobile set (greenhouse → projector), dense set (comic_book → albatross) and all the models (comic_book → mongoose).*

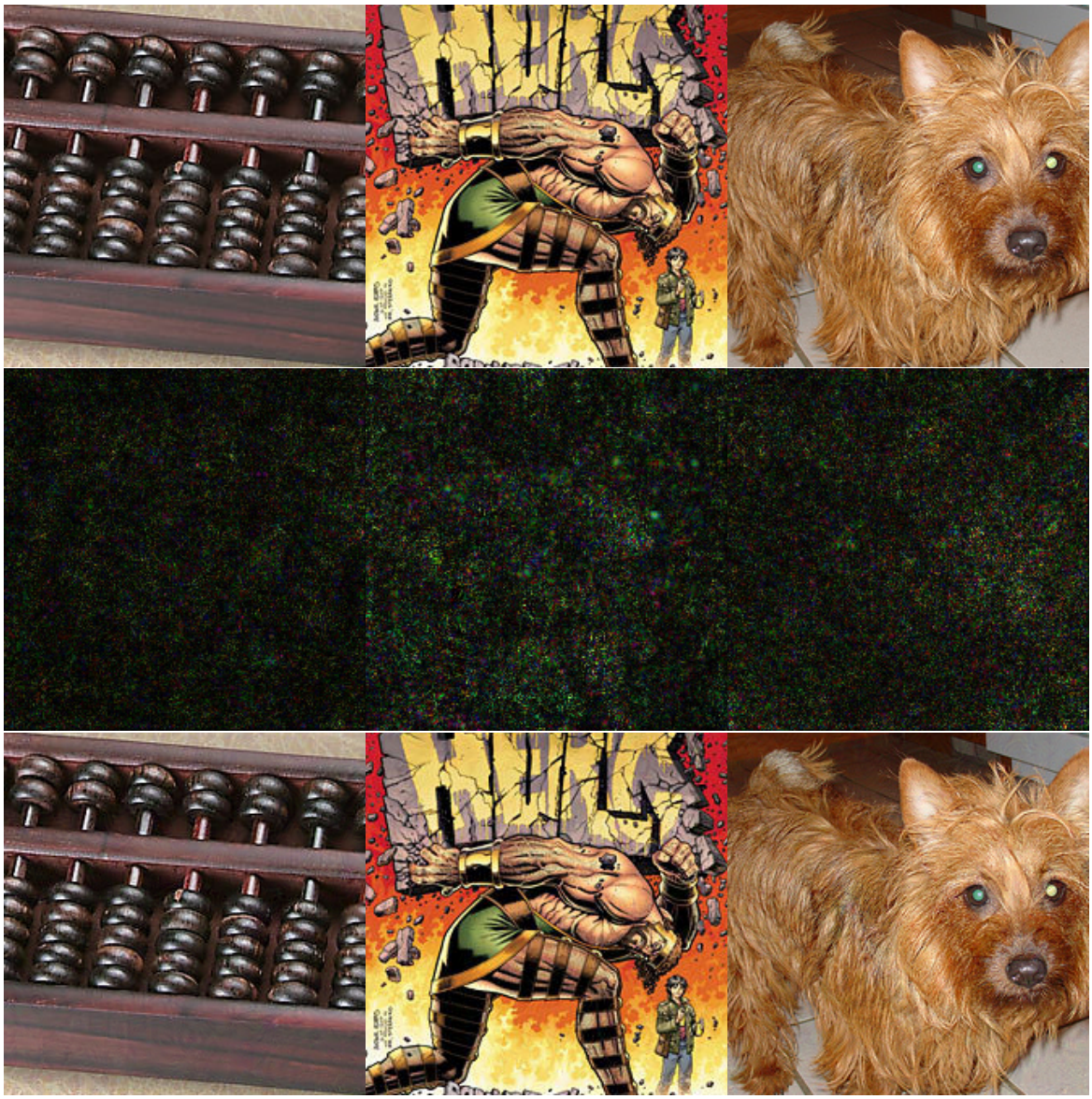

Figure 4.9: *The diverse attack pattern over the mobile set (abacus → [soft-coated-wheaten_terrier, soft-coated_wheaten_terrier, apron]), dense set (comic_book → [sturgeon, black_stork, capuchin]), and all the models (Australian_terrier → [Saluki, borzoi, black_stork, Saluki, gorilla, kuvasz]).*

and successful perturbations reliably in all the attack scenarios we examined. Here, the most interesting result is that imperceptible adversarial perturbations were found even when the labels were selected to make the problem as hard as possible. This was surprising to us, even in the light of the vast literature on adversarial attacks.

The perturbation sizes over the three model sets offered some interesting insights as well. The set with different model architectures (mobile set) needed somewhat larger perturbations, but we expected just the opposite. Increasing the size of the model set increased perturbation size as well. Nevertheless, all the perturbations we found are imperceptible to the human eye.

Chapter 5

Combining Robust Classification and Robust out-of-Distribution Detection

Classification models in machine learning often make over-confident but incorrect predictions on input samples that do not belong to any of the output classes. Such samples are called out-of-distribution (OOD) samples. This problem has received considerable attention, because this represents a vulnerability similar to adversarial input perturbation, where models make incorrect predictions on seemingly in-distribution input samples that contain a very small but adversarial perturbation. We are interested in models that are robust to both OOD samples and adversarially perturbed in-distribution samples. Furthermore, we require that OOD detection be robust to adversarial input perturbation. That is, OOD samples and in-distribution samples should not have adversarial perturbations that makes them appear to be in-distribution and OOD samples, respectively. Several related studies apply an ad-hoc combination of several design choices to achieve similar goals. One can use several functions over the logit or soft-max layer for defining training objectives, OOD detection methods and adversarial attacks. Here, we present a design-space that covers such design choices, as well as a principled way of evaluating the networks. This includes a strong attack scenario where both in-distribution and OOD examples are adversarially perturbed to mislead OOD detection. We draw several interesting conclusions based on our empirical analysis of this design space. Most importantly, we argue that the key factor is not the OOD training or detection method in itself, but rather the application of matching detection and training methods.

The chapter: section 5.1 contains the introduction and section 5.2 summarizes the related works. In section 5.3, we formalize our problem statement that is combining robust classification and robust out of distribution detection. Using our framwerok, in section 5.4 we higlight the main components. The experimental setup is detailed in section 5.5. Result are summarized in sections 5.6 to 5.10 and concluding remarks are in section 5.11.

5.1 Introduction

Recently, robust out-of-distribution (OOD) detection also received considerable attention [2, 27, 65]. Adversarially trained models are relatively robust to adversarial input but they might assign high confidence to OOD samples. In a real-world application, this also represents a serious vulnerability [65]. Besides, OOD input is also open to adversarial perturbation, making OOD detection even harder. Recently, adversarial training on both in-distribution and OOD samples was shown to be able to increase the robustness of OOD detection [2, 65]. However, the proposed algorithms are somewhat ad-hoc, as the underlying design-space for robust training, detection, and attack methods has not been explicitly formalized and explored.

Here, we present a systematic design-space that covers most of the popular design choices for the various components. This includes the possible adversarial training objectives, robust OOD detection methods and adversarial attacks on both in-distributions and OOD samples. This allows us to propose ideal combinations of training and detection methods, and to explore the robustness to the various adversarial attacks.

We draw several interesting conclusions based on our empirical results. Most importantly, we argue that the key factor is not the OOD training or detection method in itself, but rather the application of matching detection and training methods. This observation is important especially when both the in-distribution and OOD samples are adversarially perturbed. Also, it is interesting to note that among the detection methods that we evaluate here, the widely used Maximum Softmax Probability [28] baseline performed the poorest.

This chapter is a significantly revised and extended version of a conference publication [53]. The original contributions of this conference publication were

- the identification of the main components that can be used to systematically build training objectives, OOD detectors and attack methods that cover most algorithms from related work,

- an experimental analysis of this design-space that resulted in novel observations regarding the best combinations of these components,

- an evaluation methodology for measuring the robustness of the models under the *strongest possible attack*, where we measured the robust accuracy over the in-distribution samples attacked based on the loss function, and the robust OOD detection performance after attacking both in-distribution and OOD samples based on the score function used by the OOD detector.

The novel contributions relative to this previous publication include

- the investigation of the popular approach of using a separate trainable class to represent OOD inputs,
- the experimental evaluation of the framework over additional network architectures and specific network instances from related work to illustrate the importance of using a correct evaluation methodology,
- the evaluation of ℓ_2 norm attacks,
- and a detailed analysis of the fine resolution of OOD detection performance as a function of the semantics of the OOD inputs.

5.2 Related work

Outlier, or OOD detection in classification has long been a topic of interest [24, 28, 29, 43]. Here, we focus on works where robustness to adversarial perturbation is a goal as well. Augustin et al. [2] and Sehwag et al. [65] consider the problem of combining robust classification and robust OOD detection, like we do. Sehwag et al. investigated the robustness of multiple OOD detection methods and found that existing OOD detectors are not robust. They proposed the adversarial training of a classifier, in which the OOD samples are considered an extra class. Augustin et al. used a different detection method for OOD samples. They require that the classifier outputs a uniform class distribution on perturbed OOD samples. Our contribution relative to these works is showing that training and detection have to be based on the same criteria in order to get the best performance, as well as proposing a principled evaluation method.

A number of studies make the assumption that adversarially perturbed in-distribution samples and OOD samples (such as noise, for example) should be treated in a similar fashion. In effect, these approaches wish to characterize clean in-distribution samples. Hein et al. [27] attempt to reduce OOD confidence via using an adversarial training objective only for the OOD samples and show that this improves the detection of adversarial in-distribution samples as well. Stutz et al.[69] propose to train the model to predict a uniform distribution for adversarial input and the correct class for clean input and show that this approach improves the detection of OOD samples as well. Lee et al. [40] assume that a pre-trained classifier is given. They wish to detect non-clean examples based on the softmax distribution of this classifier. Here, we assume that adversarially perturbed in-distribution samples must be assigned the correct label of their clean version. This is a significantly harder requirement.

In [10], the authors propose ATOM (adversarial training with informative outlier mining). ATOM also learns to classify clean in-distribution samples. The OOD samples are represented as a dedicated background class, meaning the classifier network

has an additional output class for OOD samples. In contrast to our work, since ATOM does not use adversarial training on in-distribution, the method is unable to classify perturbed in-distribution images. Moreover, the method fails to perform robust OOD detection in the strongest configuration when in-distribution and OOD samples are both attacked. This is because their training objective does not use perturbed in-distribution images for adversarial OOD detection.

In [66, 67], robust training was used on in-distribution as well, combined with three techniques. These included self-supervision, denoising layers, and a reconstruction loss using a decoder on the learned features. The authors claim that this modification will encourage the model to learn more semantic features that help identify OOD samples. The method does not utilize any OOD data during training nor does it use an adaptive attack to evaluate the proposed detection method. However, our results indicate that matching the attacking objective and the detection method is essential for reliable evaluation. In the hardest case of evaluation when both in-distribution and OOD samples were attacked, this method was unable to detect OOD samples, as confirmed by [3].

In [11], robust training is proposed on both in-distribution and OOD examples, similarly to this work. The evaluation method the authors propose is sound and uses adaptive attacks. On OOD samples, the applied loss function requires the model to produce a uniform probability over the output classes. The authors tested two detection methods. The first was based on the maximum softmax probability, and the second was ODIN [43] that is also based on the maximum softmax value. In contrast, our work suggests that the best performance is achieved when the training objective and the detection method are based on the same scoring function.

In [3], the authors propose an adversarially trained discriminator to detect OOD samples. The discriminator is trained using perturbed in-distribution and OOD samples as well as generated samples in the feature space. As a feature extractor, the authors adversarially train a network over the in-distribution. This means that they are also able to perform robust classification. The evaluation is sound and uses adaptive attacks for the detectors on both in-distribution and OOD samples. Based on our experiments robust classification and robust OOD detection can boost each other when both objectives are pursued at the same time. However, Azizmalayeri et al. do not leverage this effect because the pre-trained feature extractor is not trained for OOD detection.

One can also think of our work as an improvement of adversarial training [47] with a more realistic, stronger threat model. Indeed, although not in the focus of the present study, our extended adversarial training approach does improve adversarial accuracy relative to the work of Madry at al. [47]. Maini et al. [48] also extend adversarial training, but instead of using perturbed OOD samples, they apply a set of different perturbations over the in-distribution samples. Stutz et al. [69] showed,

however, that the approach of Maini et al. does not make models robust to attacked OOD samples. Combining these two different ways of extending adversarial training could be a direction for future work.

5.3 Combining our Two Objectives

We have two objectives that we want to achieve simultaneously: robust classification and robust OOD detection. This raises a number of design issues. First of all, one has to design the model in such a way so as to support both classification and OOD detection at the same time. This can be done in many different ways. For example, in feed forward neural networks, we need to decide whether to base OOD detection on the logit layer or on the softmax layer, or whether to extend the classifier with an extra class representing OOD samples or not. Second, we need to define a combined loss function that represents both objectives. This function will likely be different for OOD samples and normal samples.

In this section, we first present a framework in which these decisions can be represented. We then analyze related work and show how the main different design choices fit into this framework. This analysis will reveal that in related work the design decisions are often ad hoc in the sense that the OOD detection method and the training process are inconsistent, resulting in a suboptimal performance overall. When using our framework, the appropriate pairing of the training and detection method is evident.

5.3.1 The Robust OOD Learning Problem

Let us first introduce some basic notations. We are interested in the supervised classification task where each training example $x \in \mathbb{R}^d$ is drawn from an underlying theoretical distribution $\mathcal{D}_{in}$ (that is, $x \sim \mathcal{D}_{in}$). Set $\mathcal{C}$ contains the possible labels, and exactly one label $y \in \mathcal{C}$ is assigned to each training example in the training dataset. Let $\mathcal{K} = |\mathcal{C}|$ denote the number of classes. Since the label is assumed to be a deterministic function of the example, we will also abuse the notation and write $(x, y) \sim \mathcal{D}_{in}$ to indicate the label.

We are also given a set of unlabeled examples drawn from a distribution $\mathcal{D}_{out}$ over $\mathbb{R}^d$ such that $\mathcal{D}_{out}$ and $\mathcal{D}_{in}$ are sufficiently different. From the point of view of the present study, a rigorous mathematical specification of the difference is not necessary. Instead, the key property of $\mathcal{D}_{out}$ that we rely on is that the probability of an example $\hat{x} \sim \mathcal{D}_{out}$ having a correct label within $\mathcal{C}$ is vanishingly small.

Let the function $f_\theta : \mathbb{R}^d \to \mathbb{R}^\mathcal{K}$ denote the output of a feed forward neural network classifier with parameters θ without the softmax normalization layer. In other words, f_θ returns the so-called logit layer.

Robust classification can be formalized as a robust optimization problem [47]. Here, we are given a set of possible input perturbations, for example, $\Delta = \{\delta : ||\delta||_\infty \leq \epsilon\}$. We want to minimize the loss of our classifier, assuming that the input can be perturbed using any perturbation from Δ, that is, we wish to solve the minimization problem $\min_\theta \rho_{in}(\theta)$, where

$$\rho_{in}(\theta) = \mathbb{E}_{(x,y)\sim\mathcal{D}_{in}}[\max_{\delta\in\Delta} \mathcal{L}(e_y, f_\theta(x+\delta))]. \tag{5.1}$$

Here, $\mathcal{L}(e_y, f_\theta(x+\delta))$ is a loss function of example (x, y) and model parameters θ, and e_y is a one-hot encoded vector of the label y. From now on, we will assume that the loss function is the categorical cross-entropy function

$$\mathcal{L}(e_y, f(x)) = -\sum_{i=1}^{\mathcal{K}} (e_y)_i \log \sigma(f(x))_i \tag{5.2}$$

although any differentiable loss function could be used. Here, function σ represents the softmax function, that is,

$$\sigma(z)_i = \frac{\exp(z_i)}{\sum_{j=1}^{\mathcal{K}} \exp(z_j)}. \tag{5.3}$$

The second objective is robustly detecting OOD samples from $\mathcal{D}_{out}$. This needs a different loss function that is defined for the OOD training examples. Let us define a *score function* $s(f_\theta(x))$, such that we expect a low score for OOD samples and a high score for in-distribution input. The score function is defined over the logit layer output, that is, we have $s : \mathbb{R}^{\mathcal{K}} \to \mathbb{R}$. Based on this score, we can define the OOD training objective to be $\min_\theta \rho_{out}(\theta)$, where

$$\rho_{out}(\theta) = \mathbb{E}_{x\sim\mathcal{D}_{out}}[\max_{\delta\in\Delta} s(f_\theta(x+\delta))] \tag{5.4}$$

When training the model on in-distribution and OOD samples simultaneously, the two optimization problems are combined as

$$\min_\theta \rho_{in}(\theta) + \lambda\rho_{out}(\theta) \tag{5.5}$$

where λ is a weight parameter that controls the weight of in-distribution and OOD training examples.

5.3.2 Score Functions

To instantiate the learning problem in eq. (5.5), one needs to select a score function. Here, we present a set of common score functions.

A common baseline method defined over the softmax output is the maximum softmax probability (MSP) proposed in [28] as

$$s^{msp}(f_\theta(x)) = \|\sigma(f_\theta(x))\|_\infty, \tag{5.6}$$

and another score function s^{uni}, defined as the cross-entropy of the softmax output and the uniform distribution, was proposed in [29]

$$s^{uni}(f_\theta(x)) = \mathcal{L}(\mathbf{1}/\mathcal{K}, f_\theta(x)), \tag{5.7}$$

where $\mathbf{1}$ is the vector of all ones. Thus, both of these two score functions are defined over the softmax output. Note that the maximum softmax probability is minimal when the softmax output is uniform, so these score functions are closely related.

Another possibility is to use the logit output to define score functions. One such option is

$$s^{ml}(f_\theta(x)) = \|f_\theta(x))\|_\infty, \tag{5.8}$$

that is, the maximum logit value. Although we have not found s^{ml} in related work, we include this variant in our evaluation because it is related to s^{msp}, the maximum softmax value. The smooth approximation of s^{ml} can be computed using the Log-SumExp function as

$$s^{lse}(f_\theta(x)) = \log \sum_{j=1}^{\mathcal{K}} \exp(f_\theta(x)_j). \tag{5.9}$$

This score function was used in [24].

5.3.3 Score Functions for a Dedicated OOD Class

So far, we have implicitly assumed that there is no dedicated OOD class in the output of the network. However, OOD detection can also be implemented using a $\mathcal{K}+1$-th class in such a way that for OOD inputs we use the label $y = \mathcal{K}+1$.

In this setting, eq. (5.1) is still the same, however, specific score functions need to be defined that focus on the $\mathcal{K}+1$-th class. One such score function is

$$s^{bgp}(f_\theta(x)) = 1 - \sigma(f_\theta(x))_{\mathcal{K}+1}, \tag{5.10}$$

that is, the probability of the OOD class.

Here, we can also base the score function directly on the logit layer as well. One possibility is the logit value itself, as

$$s^{bgl}(f_\theta(x)) = -f_\theta(x)_{\mathcal{K}+1}, \tag{5.11}$$

and a second logit-based score function is the logit margin of the OOD class:

$$s^{bgd}(f_\theta(x)) = -f_\theta(x)_{\mathcal{K}+1} + \max_{i \in 0...\mathcal{K}} f_\theta(x)_i. \tag{5.12}$$

5.4 OOD Detection, Adversarial Attack and Robust Training

Having defined our framework, let us consider the three major components of training and evaluating a robust OOD network: the OOD detection method, the adversarial attack method, and the robust training method. First of all, observe that *all these three components rely on a score function*.

Clearly, during adversarial training, in eq. (5.4) we have to apply a score function that is attacked in the inner-loop maximization step.

During evaluation, again, we need to select a score function that, similarly to the inner loop of eq. (5.4), is used as the target of the attack. After the adversarial perturbation, we need to select a third score function that is used for detection. In the detection step, the score function is computed, and a decision is made based on a suitable threshold.

5.4.1 Correct Evaluation Methodology

In the light of our formulation, to follow the correct methodology, one has to pay attention to using the correct score functions. In particular, we propose the following rules to be followed during the evaluation of a robust OOD network:

1. In the detection method, use the score function that was used during training because that is the score function that was made robust, thus, this score function is expected to provide the best decision after any attack

2. When attacking OOD samples, use the score function that is used for detection, because this way the worst-case performance can be measured

In our experimental evaluation, we will provide empirical evidence supporting these two rules. Quite remarkably, in related work, these rules are not always followed, and as a consequence, the detection performance can be improved using the correct score function, as we shall demonstrate. We now look at examples of training and detection methods from related work, and show whether or not these are suitably matched, in the light of these observations.

5.4.2 Training Objectives in Previous Work

In the first approach we discuss, the goal is to make the model output a uniform distribution when an OOD input is presented [2, 29]. This is implemented using the cross-entropy loss function with the uniform distribution as the true distribution:

$$\rho_{out}^{uni}(\theta) = \mathbb{E}_{x\sim\mathcal{D}_{out}}[\max_{\delta\in\Delta} \mathcal{L}(\mathbf{1}/\mathcal{K}, f_\theta(x+\delta))], \tag{5.13}$$

In [27], instead of the distance from the uniform distribution, the maximum softmax probability was minimized for OOD samples. In this case, the optimum is the same, namely the maximal probability is minimized by the uniform distribution.

We can fit this approach into our framework if we chose the score function s^{uni}. This would suggest, as discussed in section 5.4, that the model trained this way should use s^{uni} also for the detection of OOD examples. Interestingly enough, that practice is not always followed in related work. For example, in both [2] and [29], the detection is implemented using s^{msp} (see section 5.3.2), although Hendrycks et al. mention in the Appendix that s^{uni} might be more promising [29].

The second common representation is interpreting OOD samples as an extra background class [49, 65]. In this approach, the objective is simply given by

$$\rho_{out}^{bg}(\theta) = \mathbb{E}_{x\sim\mathcal{D}_{out}}[\max_{\delta\in\Delta} \mathcal{L}(e_{\mathcal{K}+1}, f_\theta(x+\delta))] \tag{5.14}$$

In our framework, that amounts to the applications of s^{bgp} as score function. This would suggest that the model trained this way should use s^{bgp} (with an appropriate threshold) for the detection of OOD examples. Again, in related work this is not always the case. For example, Sehwag et al. [65] simply use the criteria whether the background class is maximal or not, instead of thresholding s^{bgp}. Also, they do not use s^{bgp} to attack OOD samples for testing robust OOD detection either, another natural choice that we will discuss in the following sections. Instead, they apply a targeted attack to maximize the label of a random in-distribution class.

We also experimented with a slight modification of this approach, where we do not train extra parameters for the background class. Here, we use the same loss as given in eq. (5.14), while freezing the logit of the extra class to be zero, that is, $f_\theta(x)_{\mathcal{K}+1} = 0$. As a result of training, the original classes from 1 to $\mathcal{K}$ can adapt to this constant so that this constant is maximal when the input is OOD. This way, the number of parameters will be the same as in the case of using ρ_{out}^{uni} in eq. (5.13), allowing for a fair comparison between the two approaches.

We will refer to this frozen-extra-class objective as ρ_{out}^{lse}, because, as we argue

below, in this case the score function is effectively s^{lse}. Indeed, using eq. (5.2),

$$\mathcal{L}(e_{\mathcal{K}+1}, f_\theta(x)) = \log \sum_{j=1}^{\mathcal{K}+1} \exp(f_\theta(x)_j). \tag{5.15}$$

Since $f_\theta(x)_{\mathcal{K}+1} = 0$, this function is a monotonous function of the LogSumExp of the in-distribution classes (that is, the classes up to $\mathcal{K}$). This indicates that in fact the decisive factor is whether the in-distribution classes have a small logit value, as opposed to the extra background class (the $(\mathcal{K}+1)$th class) having the maximal value. Again, this would suggest that the model trained this way should use s^{lse} (with an appropriate threshold) for the detection of OOD examples.

As a last note, the three approaches above (ρ_{out}^{uni}, ρ_{out}^{lse} and ρ_{out}^{bg}) are far from being equivalent. The objective represented by ρ_{out}^{lse} represents a larger degree of freedom, because the score function is applied before softmax normalization and because the uniform distribution is not enforced. While ρ_{out}^{uni} and ρ_{out}^{lse} have exactly the same number of parameters, in the case of the ρ_{out}^{bg} objective the parameters of the background class are learnable.

5.5 Experimental setup

We are interested in the effect of the possible combinations of the various training objectives and score functions that we described previously, as well as the effect of other hyper-parameters such as the choice of $\mathcal{D}_{out}$ and network capacity. In order to understand this, we performed a systematic empirical study, in which we combined training objectives and score functions, and used various hyper-parameters and databases for in-distribution and OOD samples. The two databases we used to represent in-distribution samples were MNIST [39] and CIFAR-10 [37].

We first describe those settings that were common to the MNIST and CIFAR-10 experiments in section 5.5.1 and section 5.5.2. We then lay out the settings specific to the MNIST and CIFAR-10 experiments in section 5.5.3 and section 5.5.4, respectively.

We shall apply the PGD algorithm [38] in many different contexts. Let us introduce the notation PGD_b^a, where the superscript a denotes the number of iterations and subscript b denotes the number of trials. We omit the subscript when there is only one run.

5.5.1 Training

For preprocessing, we divided all the input values by 255, thus scaling the data into the range $[0, 1]^d$.

In each setting, we used adversarial training [47] for both in-distribution and OOD samples. The adversary during training was PGD (with database specific parameters defined later) that used the loss function in ρ_{in} for in-distribution samples, and the score function in ρ_{out} for OOD samples, respectively. We held out 1000 in-distribution samples as a validation set. We then selected the best model that was found during training according to the robust accuracy over the validation set, that is, the accuracy over the adversarially perturbed validation samples, as suggested in [61]. The validation samples were attacked using the same adversary as used for training in the case of CIFAR-10. For MNIST, we used a stronger adversary (PGD_5^{100}) because the training adversary (to be described below) was not able to reliably identify a unique maximum validation accuracy.

Throughout our evaluation, the training was performed with a batch size of 100, that consisted of 50 in-distribution examples and 50 OOD examples. For the baseline case when no OOD samples were used, the batch size was 50.

Let us now describe the training objectives that we evaluated. The generic formula for the training objective is given in equation eq. (5.5). This formula contains parameter λ that controls the relative weight of the in-distribution and OOD objective. We evaluated three possible values of λ: $0.1, 0.5$, and 1.0. $\lambda = 0.1$ means that the OOD samples will have the same contribution as any other in-distribution class (recall that each dataset defines 10 classes), essentially treating the OOD samples as a 11th class. Sehwag et al. [65] applied this weighting strategy. $\lambda = 1.0$ means the distributions $\mathcal{D}_{in}$ and $\mathcal{D}_{out}$ have equal importance. Hein et al. used this setting [27].

Apart from parameter λ, we also varied the OOD objective ρ_{out}. In particular, we experimented with two possible OOD objectives presented in eqs. (5.13) and (5.15). In total, the three values of λ and the two possible OOD objectives result in $6 = 3 \cdot 2$ trainings for every combination of dataset and network architecture (to be described below).

5.5.2 Evaluation

We are interested in robust accuracy and robust OOD detection. Robust accuracy was measured as the accuracy against an untargeted PGD adversary [47]. In the attack, PGD maximized the loss used in ρ_{in}. The performance of the non-robust OOD detection was measured using the area under curve (AUC) metric, as usual in related work (for example, [2, 27]). AUC is equal to the probability that a randomly chosen in-distribution sample $x \sim \mathcal{D}_{in}$ gets a larger score than a randomly chosen OOD sample $\hat{x} \sim \mathcal{D}_{out}$. That is, $\text{AUC} = P(s(f_\theta(\hat{x})) \leq s(f_\theta(x)))$. If the AUC is close to one then there exists a threshold that separates the OOD samples well. If the AUC is close to $1/2$ then separation is not possible.

Since AUC is sensitive to imbalanced classes, it was calculated using an equal

number of samples from $\mathcal{D}_{in}$ and $\mathcal{D}_{out}$. More precisely, we used 1000 samples from both distributions for each evaluation.

The robust version of OOD detection was evaluated by measuring the AUC after attacking only the OOD samples, or both the in-distribution and OOD samples. In this robust version, the in-distribution samples are attacked using PGD minimizing the score of a given score function. In the case of the OOD samples, the same score function was maximized by PGD.

During the evaluation of robust OOD detection, we combined the 4 functions described in section 5.3.2, using them for both the attack and the detection method. These represent $4 \cdot 4 = 16$ possible attack-detection combinations. In the case of the ρ_{out}^{bg} objective, in addition to these 4 functions we also tested the 3 functions listed in section 5.3.3, which results in $7 \cdot 7 = 49$ attack-detection combinations.

5.5.3 MNIST-Specific Settings

We trained the same convolutional network that was used in [47]. It has two convolutional layers with 32 and 64 filters, respectively. Each convolutional layer followed by a 2x2 max-pooling layer. After the last pooling, two dense layers are applied with 1024 and 10 neurons. ReLu activation was used in each of the hidden layers.

We used Adam [36] as our optimizer with a learning rate of 10^{-4}. We ran it for 100 epochs. The adversary during adversarial training was PGD^{40} with a step size of $\alpha = 10^{-2}$ and $\epsilon = 0.3$. Recall, that ϵ defines the set of possible input perturbations: $\Delta = \{\delta : ||\delta||_\infty \leq \epsilon\}$.

We used two OOD datasets for training. The first one is the synthetic noise distribution introduced in [27], we will refer to it as $\mathcal{D}_{out}^{SN}$. In a nutshell, half of the inputs are generated uniformly at random and the other half is generated by permuting the pixels of images from the training set. A Gaussian smoothing filter is applied for all the images, followed by a global rescaling into $[0, 1]^d$. The idea here is to preserve as much as possible from the global statistics of the original images while destroying the more complex features. The second OOD dataset was the KMNIST dataset[14].

For the evaluation of OOD detection, we used the test sets of the two OOD datasets used for training and two additional datasets to test how OOD detection generalizes to unseen distributions. The first was the Fashion-MNIST [78] test set and the second was uniform noise within $[0, 1]^d$, we will refer to it as $\mathcal{D}_{out}^{U}$. PGD_{50}^{100} was used for all the attacks (as in in [47]) with a step size of $\alpha = 10^{-2}$ and $\epsilon = 0.3$, except for Fashion-MNIST where we used $\epsilon = 0.1$.

Table 5.1: *DenseNet [31] architectures used for CIFAR-10 experiments.*

Model	Parameters
Wide-DenseNet-BC (L=16, k=60)	1.1M
Wide-DenseNet-BC (L=28, k=60)	2.5M
Wide-DenseNet-BC (L=40, k=60)	4.3M
Wide-DenseNet-BC (L=52, k=60)	6.4M
Wide-DenseNet (L=16, k=60)	2.6M
Wide-DenseNet (L=52, k=60)	41.7M
Wide-ResNet (L=28, k=10)	36.5M
ResNet-50	23.5M

Table 5.2: *Accuracy and ℓ_2 robust accuracy values for CIFAR-10 Ratio [2] models.*

Model	accuracy	robust accuracy($\epsilon = 0.5$)
$\text{Ratio}_{0.25}$	0.9353	0.7198
$\text{Ratio}_{0.5}$	0.9108	0.7487

5.5.4 CIFAR-10-Specific Settings

Here, we describe the architectures and model instances we applied, the training algorithms, and the evaluation methodology.

Architectures

Here, we used a large number of architectures in order to be able to study the effect of network size, and to be able to compare our measurements with related work.

We used several wide variants of DenseNet [31] using bottleneck layers and compression (denoted by Wide-DenseNet-BC), with the parameters listed in table 5.1. We also used a number of Wide-DenseNets with all the bottleneck layers removed and no compression. We refer to this version as Wide-DenseNet.

To allow for a more direct comparison with related work, we include a number of additional network architectures in some of our measurements. In [65] a Wide-Resnet [86] variant was used while in [2] the authors applied the Resnet-50 [26] architecture. We include these networks in our experiments.

We also evaluate two specific network instances: $\text{Ratio}_{0.25}$ and $\text{Ratio}_{0.5}$, used in [2]. These are Resnet-50 networks pretrained on CIFAR-10. As a training OOD dataset, the authors also used the 80 Million Tiny Images dataset [72]. During training a 7-step ℓ_2 PGD attack was applied on the in-distribution with an $\epsilon = 0.25$ and 0.5 for $\text{Ratio}_{0.25}$ and $\text{Ratio}_{0.5}$, respectively. In the inner loop of the robust OOD objective an ℓ_2 PGD was applied but with 20 steps and $\epsilon = 1.0$. Table 5.2 contains the

performance of these models.

We were not able to obtain the network instances used in [65]. Therefore, we trained and evaluated our own instances using the Wide-Resnet-28-10 [86] architecture and the proposed objective function, which was ρ_{out}^{bg} (see eq. (5.14)). As optimizer we used SGD with momentum 0.9 and an initial learning rate of 10^{-1}. We ran this optimizer for 200 epochs with a batch size of 128. The learning rate was divided by 10 at 50% and 75% of the training, as was done in [47]. In addition, we applied weight decay with a coefficient of $5 * 10^{-4}$. We also applied standard augmentation: mirroring and shifting similarly to [47].

Training

For all the DenseNet architectures, the optimizer we used was SGD with momentum 0.9 and an initial learning rate of 10^{-1}. We ran this optimizer for 240 epochs. The learning rate was divided by 10 at 50% and 75% of the training, as was done in [31]. In addition, we applied weight decay with a coefficient of 10^{-4}. We also applied standard augmentation: mirroring and shifting similarly to [47].

In related work, different threat models are used. For example, the ℓ_2 attack with an $\epsilon = 63.75/255$ and $127.5/255$ was used in [2], while a more common ℓ_∞ threat model was applied in [65] with $\epsilon = 8/255$. For a better comparison, we experimented with both threat models. As an ℓ_2 adversary, we used PGD^7 with a step size of $\alpha = 127.5/255/7 * 2.5$ and $\epsilon = 127.5/255$ in both the in-distribution and OOD objectives. Note that the number of steps is the same as in [2] for better comparison.

As an ℓ_∞ adversary we used PGD^{10} with a step size of $\alpha = 2/255$ and with $\epsilon = 8/255$. Similarly to MNIST, we used two OOD datasets for training. Again, the first one is the synthetic noise distribution $\mathcal{D}_{out}^{SN}$ introduced in [27]. The second OOD dataset was the 80 Million Tiny Images dataset [72], we will refer to it as $\mathcal{D}_{out}^{T}$. This dataset was used also in [29] and [2]. This is a good choice because the global statistics and the low-level features of the images are similar to those of the CIFAR-10 images, while the high-level features are different. This means that OOD detection is forced to focus on high-level features, which in turn increases its robustness to adversarial OOD inputs. Note that the CIFAR-10 dataset is actually a subset of the tiny images set, so we removed the CIFAR-10 classes like it was done in [29]. Afterwards, we separated a test set of 1000 samples.

Evaluation

For the evaluation of OOD detection, we used the test sets of the two OOD datasets used for training and two additional datasets to test how OOD detection generalizes to unseen distributions. The first was a set of 10,000 samples from the SVHN [59] test set. The second was uniform noise within $[0, 1]^d$. We will refer to it as $\mathcal{D}_{out}^{U}$. PGD$_{10}^{20}$

Table 5.3: *Robust validation accuracy, clean test accuracy, and robust test accuracy for MNIST experiments*

objective(ρ_{out})	$\mathcal{D}_{out}$	λ	rob.val.acc.	acc.	rob.acc.
None			0.932	0.9895	0.9241
lse	SN	0.1	0.935	0.9893	0.9277
lse	K-MNIST	0.1	0.928	0.9897	0.9141
uni	SN	1.0	0.93	0.9904	0.9144
uni	K-MNIST	0.1	0.928	0.9887	0.9184

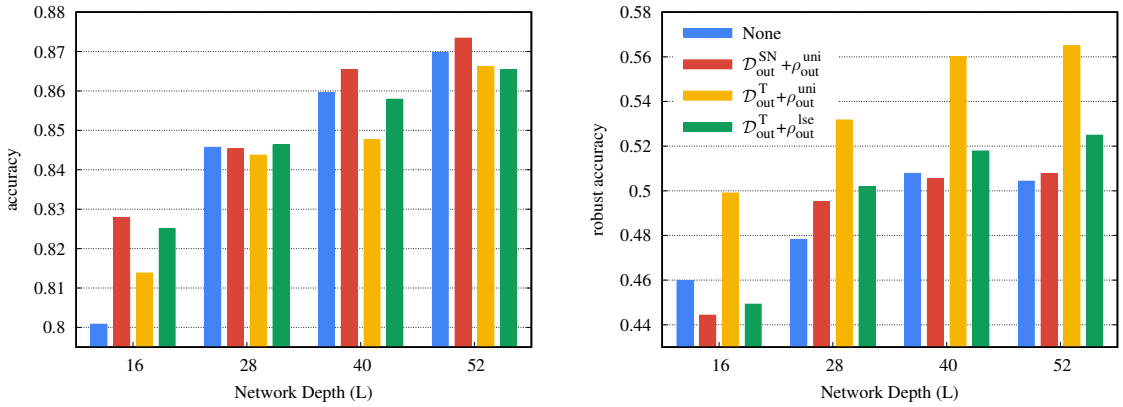

Figure 5.1: *Clean test accuracy and robust test accuracy (PGD_{10}^{20}, $\epsilon = 8/255$) on the CIFAR-10 dataset.*

was used for all the ℓ_∞ attacks (as in [47]) with a step size of $\alpha = 2/255$ and with $\epsilon = 8/255$. For the ℓ_2 attacks we used PGD_{10}^{20} with a step size of $\alpha = 127.5/255/20 * 2.5$ and $\epsilon = 127.5/255$, unless otherwise stated.

5.6 Results on Robust Accuracy

Here, our main observation shall be that *robust accuracy is not reduced (MNIST) or even improved (CIFAR-10) if a robust OOD adversarial objective is added to the adversarial training.*

To see this, let us first focus on the accuracy of the in-distribution samples, with or without adversarial input perturbation. For the MNIST dataset, the results are included in table 5.3. When computing the robust accuracy, the samples were perturbed by PGD_{50}^{100}, with $\epsilon = 0.3$. The indicated λ values are the optimal choices among the possible choices of λ for the given objective according to the robust accuracy on the validation set against a PGD_5^{100} adversary. The value "None" indicates plain adversarial training without an OOD objective.

It is clear that adding the OOD objective does not reduce robust accuracy. We also

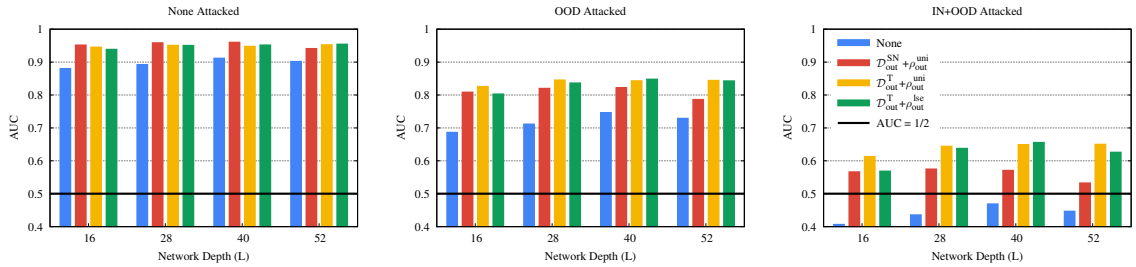

Figure 5.2: *OOD detection AUC over CIFAR-10 under three different kinds of attack scenarios: no attack, only OOD samples are perturbed and both in-distribution and OOD samples are perturbed.*

note that our reported value of 0.9241 is significantly higher than the one reported in [47]. This is because of our better early stopping criterion based on a stronger adversary used over the validation set.

Accuracy values for the CIFAR-10 dataset are shown in fig. 5.1. The OOD datasets $\mathcal{D}_{out}^{SN}$ and $\mathcal{D}_{out}^{T}$ were described in section 5.5.4. We show the results with the best choice of parameter λ according to the robust accuracy on the validation set against a PGD^{10} adversary.

Robust accuracy is significantly improved compared to plain adversarial learning, while clear accuracy remains approximately the same. It is also interesting to note that increasing the network size in general increases robust accuracy, a well-known fact that has been pointed out in [47] as well. The best choice is the uniform OOD objective with the Tiny Images OOD dataset in this case.

Similar observations were reported in [2] for an l_2 threat model (here, we use the l_∞ norm, as described before). However, our results highlight another important factor, namely the choice of the objective function. Although both ρ_{out}^{uni} and ρ_{out}^{lse} improve the robust performance compared to plain adversarial training, ρ_{out}^{uni} is a significantly better choice.

This last observation is further supported by the results in fig. 5.4, that shows our results with additional network architectures. Again, it is clearly visible that robust accuracy improves as OOD data is utilized during the training and the best choice is the objective ρ_{out}^{uni}, with the Tiny dataset with respect to both threat models and all the tested architectures.

A last remarkable observation is that comparing Wide-DenseNet-BC and Wide-DenseNet with 16 layers reveals that removing the bottleneck layers from the DenseNet architecture improves robust accuracy. This is in contrast with normal training where bottleneck layers offer better generalization and hence higher accuracy [31].

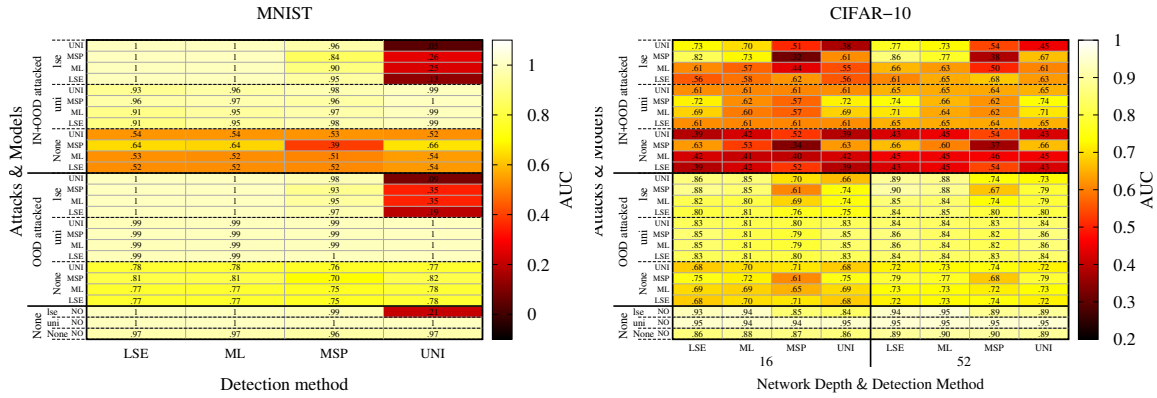

Figure 5.3: *OOD detection AUC over MNIST and CIFAR-10 under three different kinds of attack scenarios: no attack, only OOD samples are perturbed, and both in-distribution and OOD samples are perturbed. These attacks are indicated on the vertical axis. Under each attack, the three different OOD training objectives are indicated: None, ρ_{out}^{uni} and ρ_{out}^{lse}. Under each training objective, 4 different score functions are indicated that are used for attacking samples at detection time. The horizontal axis indicates possible score functions used for detection. The CIFAR-10 plot also includes the smallest and largest network architecture, indicated on the horizontal axis. During training, $\mathcal{D}_{out}^{SN}$ was used on MNIST, and $\mathcal{D}_{out}^{T}$ on CIFAR-10.*

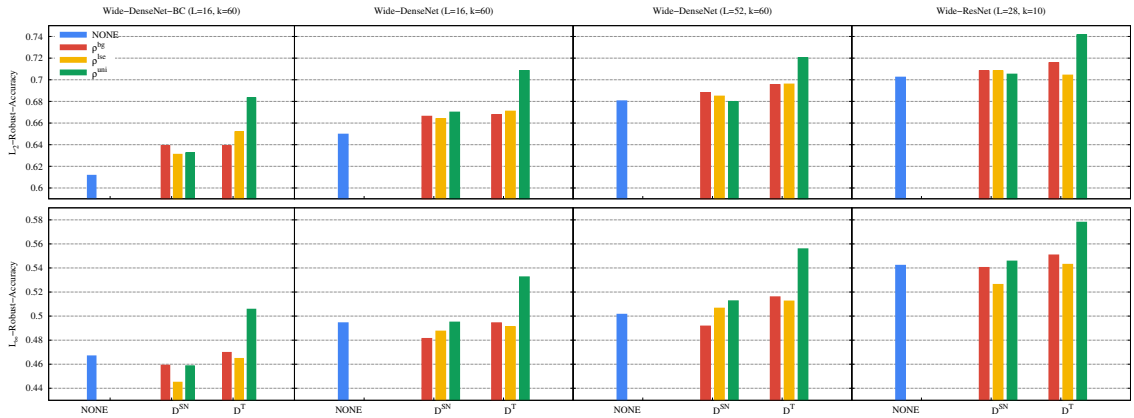

Figure 5.4: *Robust test accuracy (PGD_{10}^{20}, $\epsilon = 127.5/255$ and $\epsilon = 8/255$ for L_2 and L_∞ norms, respectively) on the CIFAR-10 dataset. Removing the bottleneck from the Wide-DenseNet-BC-16 increases performance. Robust accuracy is the best when ρ^{uni} is used as objective and D^T is used as OOD dataset, for both norms and all the investigated architectures.*

5.7 Results on OOD Robustness

Let us now look at how the OOD training objectives affect the ability of the models to detect OOD samples, under different kinds of adversarial attacks applied at detection time. The main point we make in this section is that *the OOD training objective has the strongest positive effect on OOD detection when both in-distribution and OOD inputs are attacked*.

Figure 5.2 illustrates the effect of the three kinds of adversarial attacks: (1) no attack (clean inputs), (2) only the OOD inputs are attacked and (3) both in-distribution and OOD inputs are attacked. The adversary was PGD_{10}^{20} using $\epsilon = 8/255$. The indicated AUC values are averages of AUC values computed over 13 test OOD databases, given by the 10 classes of the SVHN dataset (each treated as a separate OOD database) and the test sets of the databases $\mathcal{D}_{out}^{T}$, $\mathcal{D}_{out}^{SN}$ and $\mathcal{D}_{out}^{U}$.

For a given OOD dataset we calculated the AUC value as a *minimax value*. This value is the *AUC of the best possible detection method assuming the best attack for each detection method*. In more detail, for all the 4 possible score functions used for detection, we computed the minimum AUC value over the 4 possible score functions used for the attack. This way, for all the detection methods we have the best attack (worst case AUC). We then took the maximum of these values, which gives the minimax AUC.

What is clear is that for clean examples the training objectives have a relatively little effect, although not using an OOD objective is consistently the worst option. The harder the attack the larger the relative difference becomes between the model that did not use any OOD objective during training and those that did. Here, the decisive factor appears to be the OOD database used during training: $\mathcal{D}_{out}^{T}$ seems to be the best choice.

The same conclusion is valid also in the case of the MNIST dataset. This is evident from fig. 5.3, where the OOD detection AUC values are illustrated in a finer resolution for both MNIST and CIFAR-10. Here, compared to fig. 5.2, the AUC values are not aggregated using the minimax technique but instead all the $4 \cdot 4 = 16$ combinations of detection and attacking score functions are included individually. The values are still averages over our test OOD datasets. For CIFAR-10 we used the same 13 sets described above. For MNIST, we used 22 datasets, given by 10 classes of Fashion-MNIST, 10 classes of K-MNIST, and the test sets of $\mathcal{D}_{out}^{SN}$ and $\mathcal{D}_{out}^{U}$.

For MNIST as well, clearly, the harder the attack the larger the relative difference becomes between the models that did or did not use any OOD objective during training.

Let us point out here, that our strongest attack is significantly stronger than the attacks normally studied in related work, where in-distribution samples are not perturbed during the OOD detection task. Also, as fig. 5.2 reveals, the AUC barely

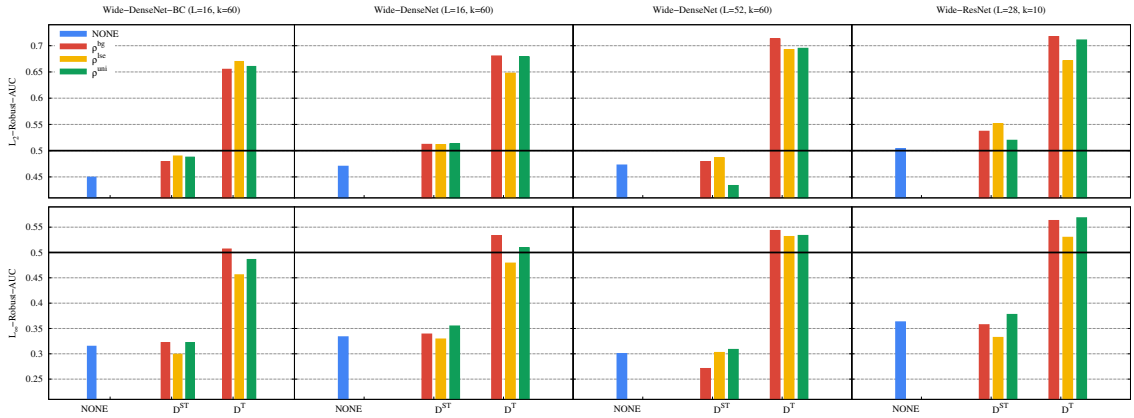

Figure 5.5: *OOD detection minimax AUC calculated over CIFAR-10 and CIFAR-100, SVHN, $\mathcal{D}^T$, $\mathcal{D}^{SN}$ and $\mathcal{D}^U$. We note that both in-distribution and OOD samples are attacked with PGD_{10}^{20} using $\epsilon = 127.5/255$ and $\epsilon = 8/255$ for L_2 and L_∞ norms respectively. Minimax AUC calculation: for all the possible score functions used for detection we computed the minimum AUC value over all the possible score functions used for the attack. We then took the maximum of these values, which gives the minimax AUC.*

improves when increasing the size of the network. It is possible that improvements would appear only with much larger networks.

Finally, in fig. 5.5 we include experimental data with additional models and training objectives. Here, we present only the most adversarial scenario when both in-distribution and OOD samples are attacked. The displayed values are averages over 113 test OOD datasets, thus here, we added the 100 classes of CIFAR-100 to our usual set of OOD datasets. The results clearly show that AUC values are below random guessing (that is, they are less than 0.5) when no OOD data was used for training, which indicates that the attacks are very successful. This is unlike the case of non-robust OOD detection, where methods work acceptably well even without OOD data [28, 43].

It is also clear that the OOD detection performance depends mostly on the architecture (larger models result in better robustness) and the OOD dataset used during training (D^T is the best in all the configurations).

5.8 Background Class Representation vs. Parameter-Free Methods

As described in section 5.3, some score functions require a dedicated class output [65], with the extra parameters represented by the weights of this extra output. One may ask the question whether it is worth modifying the architecture and to train extra weights? So far, we have seen that the training objective ρ_{out}^{uni} performs quite

well, so it is possible that the extra output with extra weights does not make a large difference.

Figure 5.4 compares ρ_{out}^{uni}, ρ_{out}^{bg}, and ρ_{out}^{lse} from the point of view of robust accuracy. The clear winner is ρ_{out}^{uni}. Clearly, due to not having an extra class the adversarial training objective over the OOD samples strengthens the in-distribution performance as well, as we have in previous sections.

The robust minimax AUC is similar across the three investigated objectives, as shown in fig. 5.5. It is an open question whether adding a more sophisticated OOD head with more parameters, perhaps including several layers, would make a significant difference. In any case, the ρ_{out}^{uni} objective is competitive with ρ_{out}^{bg}, and, overall, the uniform loss objective is a better option due to the clearly superior robust accuracy over the in-distribution data.

5.9 Matching Score Functions

In section 5.4.1 we made the point that certain rules about matching score functions need to be followed. Here, we present empirical evidence supporting this.

5.9.1 Attacks on OOD Samples are most Effective when Attacking and Detection Score Functions are the Same

Ideally, we want to use the strongest possible attack when testing the robustness of any machine learning model. However, so far, it has not been clear how the strongest attack should be designed for a given OOD detection mechanism. In our framework based on score functions, it is rather natural to assume that maximizing the score function used for detection is the strongest attack.

This conclusion is supported by the data in fig. 5.3. As we can see, for all the detection methods, the minimal AUC value belongs to the attack method that uses the same score functions, in all the three attack scenarios, for all the training objectives, for both MNIST and CIFAR-10.

5.9.2 OOD Detection and OOD Training Objective should Use the same Score Function

In the framework based on score functions, we can examine the relationship between the training objective and the detection method for OOD samples. A natural hypothesis is that these two components should be based on the same score function in order to enforce the best possible AUC value.

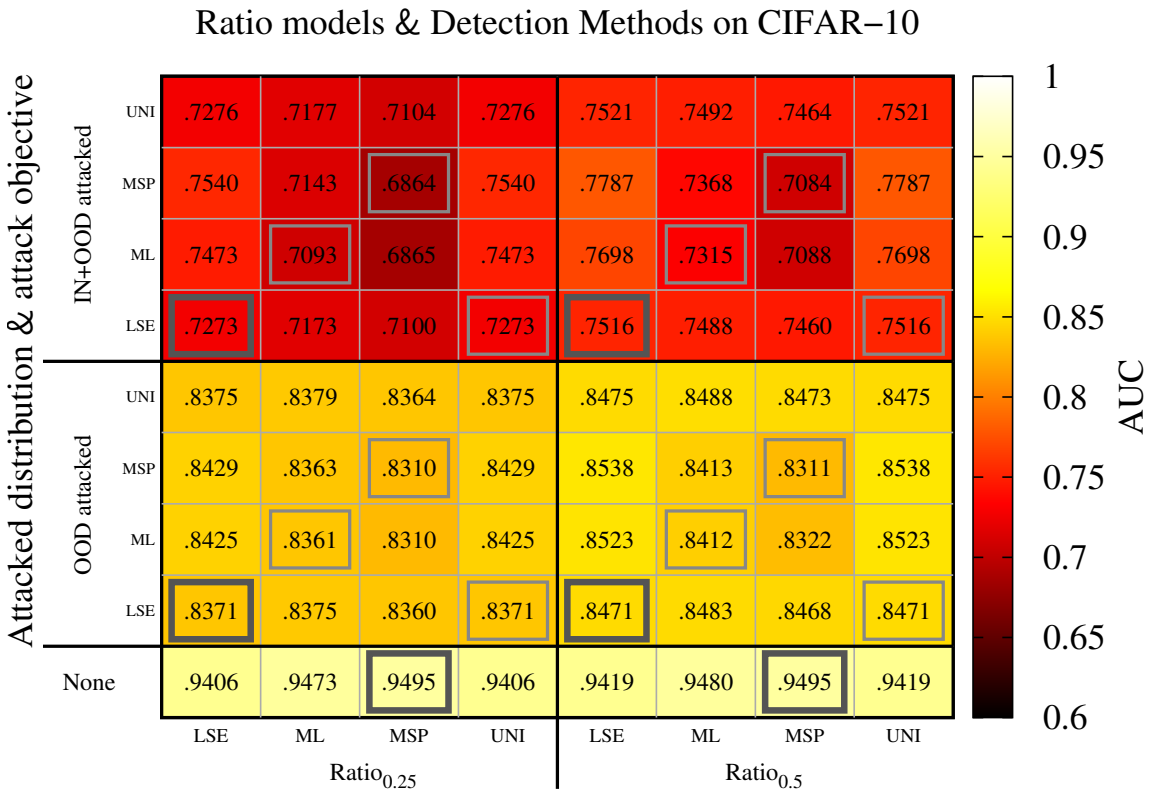

Figure 5.6: *OOD detection AUC for $Ratio_{0.25}$ (left) and $Ratio_{0.5}$ (right). The minimum AUC is marked with a border in each block per column and the minimax AUC is indicated using a thick border. Three different kinds of attack scenarios are shown: no attack, only OOD samples are perturbed, and both in-distribution and OOD samples are perturbed. These attacks are indicated on the vertical axis. Under each attack, 4 different score functions are indicated that are used for attacking samples at detection time. The horizontal axis indicates possible score functions used for detection.*

This conclusion can be verified in fig. 5.3. To see this, recall that we are interested in the minimax AUC value, that is, we want to maximize the minimal AUC value over the possible pairs of score functions used for OOD training and detection. In other words, we assume that the attacker knows the detection method as well as the training method and so she can pick the attack resulting in the minimal AUC. In the table, for all pairs of training and detection score functions four attacks are listed. The minimum of these is always maximal when the training and detection score functions are the same.

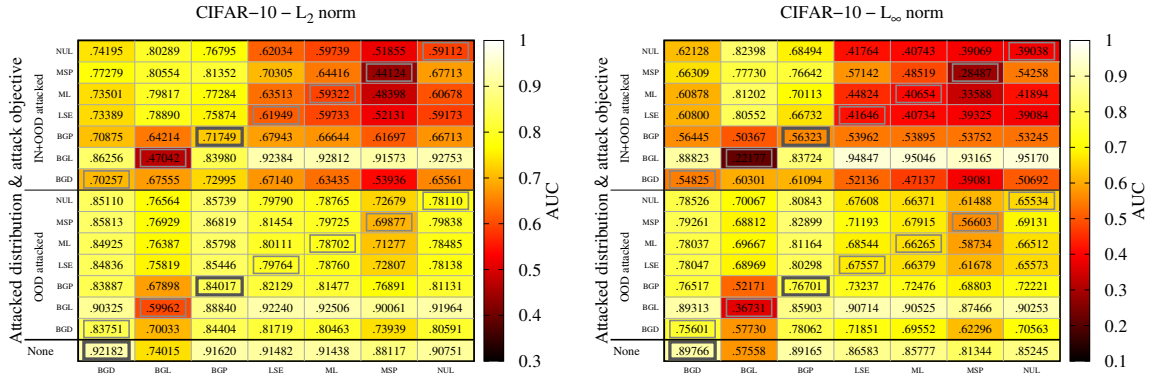

Figure 5.7: *OOD detection AUC for the Wide-Resnet architecture with the ℓ_2 (left) and ℓ_∞ (right) threat models. The training used the s^{bgp} score function and $\mathcal{D}_{out}^{T}$ as OOD dataset. The minimum AUC is marked with a border in each block per column and the minimax AUC is indicated using a thick border. Three different attack scenarios are shown: no attack, only OOD samples are perturbed, and both in-distribution and OOD samples are perturbed. These attacks are indicated on the vertical axis. Under each attack, 7 different score functions are indicated that are used for attacking samples at detection time. The horizontal axis indicates possible score functions used for detection.*

5.9.3 Ratio Models

The detailed OOD evaluation of the Ratio models [2] is shown in fig. 5.6. We can see all the AUC values (not only the aggregated minimax AUC) without attack, with OOD attacked, and with both in-distribution and OOD attacked. For a faithful OOD evaluation, we followed the methodology in [2]. In particular, we used the same OOD datasets, namely SVHN, CIFAR-100, LSUN_CR, Imagenet(CIFAR-10 classes excluded), Uniform Noise, and Synthetic Noise. As an attack, we used PGD_5^{100} in the ℓ_2 threat model with a step size of $\alpha = \epsilon/100{*}2.5$ with $\epsilon = .5$ and $\epsilon = 1$ for in-distribution and OOD, respectively.

After fixing these parameters, we evaluated the combinations of all the detection and attack score functions. For both networks, the results suggest that the attacks are most effective when the attacking and detection score functions are the same, supporting our conclusions in section 5.9.1. There is only one exception for this, namely for the detection method based on the score function s^{uni} the attack based on s^{lse} is slightly more effective, but using s^{uni} has practically the same strength, the difference is negligible. Also, we can see again that the best option is to use the same score function for detection that was used for training, as we have seen in section 5.9.2.

Both ratio models were trained using the s^{uni} score function and evaluated with the s^{msp} detection method [2]. However, our results show that the best performance is achieved when the s^{uni} or s^{lse} detection methods are used that are more in line

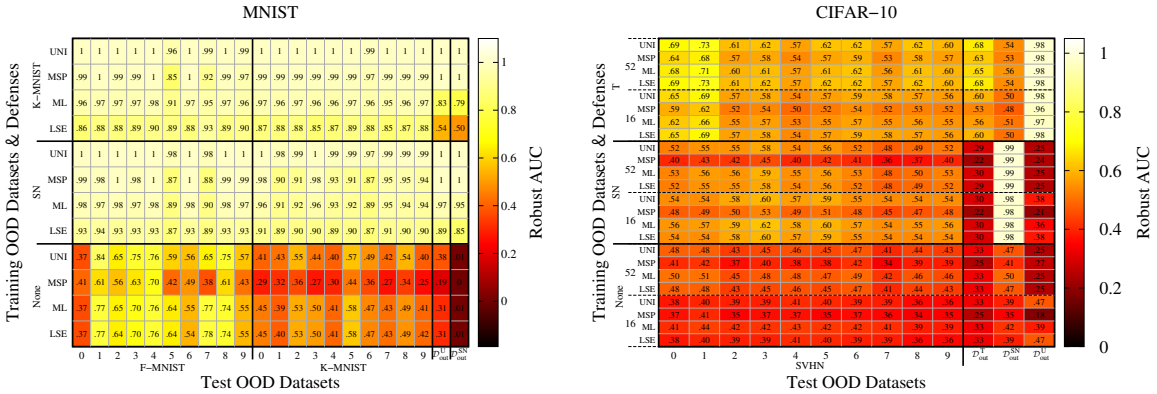

Figure 5.8: *Minimum OOD detection AUC over MNIST and CIFAR-10 under combinations of OOD datasets used during training and detection. The databases used for training are indicated on the vertical axis. The training objectives were ρ_{out}^{uni} in both cases. Under each training OOD dataset, 4 different score functions are indicated that are used for detection. The horizontal axis indicates OOD datasets used for evaluation. The CIFAR-10 plot also includes the smallest and largest network architecture, indicated on the vertical axis.*

with the training objective. The significance of the proper detection method is largest in the most adversarial case, when both the in-distribution and OOD inputs are attacked. It is remarkable that simply replacing the score function with the one used for training the model, (namely, s^{uni}) offers a 4% improvement of the robust AUC value compared to the detection method based on s^{msp}, which was used in [2].

5.9.4 Models with Trainable Background Class

Our results for the case of architectures with a trainable background class are shown in fig. 5.7. Here, we used the same architecture as in [65], to verify that also in this case, the best choice is following our proposed rules for choosing score functions. Indeed, it can be verified based on our results that this is the case. In particular, the best score function to use in all the components is s^{bgp} in the case of both the ℓ_2 and ℓ_∞ threat models.

5.10 Generalization to Unseen OOD Datasets

It is a central question whether robust OOD detection generalizes beyond the OOD dataset that was used during the adversarial training. The current practice for evaluating OOD detection is using as many datasets as possible and aggregating the metrics to have a single performance measure. Although the performance per dataset is also often reported, finer details such as class-wise performance are hard to find.

Here, we provide an evaluation at a finer level of detail, and we look at generalization to each individual class in the datasets involved. In this section, our main conclusion is that *the learned robust models do not generalize well for every possible OOD class, despite having a good average performance*.

Our first set of results related to this problem can be found in fig. 5.8. The values shown in the table correspond to those of the most successful attacks, that is, the table represents the worst case scenario. The most successful attack, as described previously, is the scenario when both in-distribution and OOD samples are adversarially perturbed, based on the same score function that is used for detection. The databases are examined at a fine resolution, that is, we consider all the classes of the unseen databases (see section 5.5.3 and section 5.5.4) as a separate database.

We can see that the models do not generalize equally well to each OOD dataset. On MNIST, the models seem to be less effective in identifying some classes as OOD, for example, Fashion-MNIST classes 5 (Sandal) and 7 (Sneaker). Similar observations can be made regarding the CIFAR-10 models. For example, SVHN classes 0 and 1 are identified as OOD much easier than the other classes.

We can also see that, overall, the $\mathcal{D}_{out}^{T}$ dataset offers the best robust OOD performance over CIFAR-10. However, it is remarkable that training with $\mathcal{D}_{out}^{T}$ does not generalize well to $\mathcal{D}_{out}^{SN}$ (although it does generalize to the uniform distribution $\mathcal{D}_{out}^{U}$) and the performance is not very impressive on the test set of $\mathcal{D}_{out}^{T}$ either. On the other hand, training with $\mathcal{D}_{out}^{SN}$ is radically different: in that case the model clearly overfits $\mathcal{D}_{out}^{SN}$ without any generalization to $\mathcal{D}_{out}^{T}$ or $\mathcal{D}_{out}^{U}$. This suggests that a mixture of multiple OOD datasets might be a better choice for representing OOD samples during training.

5.10.1 Results with CIFAR-100 classes

To examine OOD generalization in an even finer detail, we present more experimental data over additional models and over the CIFAR-100 dataset, class-wise. Recall that here, we treat each class as a separate OOD dataset so we have 100 values for CIFAR-100 representing the OOD detection performance on each class. The minimum AUC values are displayed in fig. 5.9 in fine resolution, where each column represents an OOD dataset (a class) that are grouped according to the CIFAR-100 superclasses.

Like in our previous results, the best OOD detection is achieved when the $\mathcal{D}_{out}^{T}$ dataset is used as the training OOD dataset, for both norms. In general, the conclusions made in section 5.10 so far are supported by the larger, more detailed data. However we can draw some additional conclusions. Namely, we can see great differences across the CIFAR-100 superclasses. We identified four types of these superclasses. We shall review these one by one below.

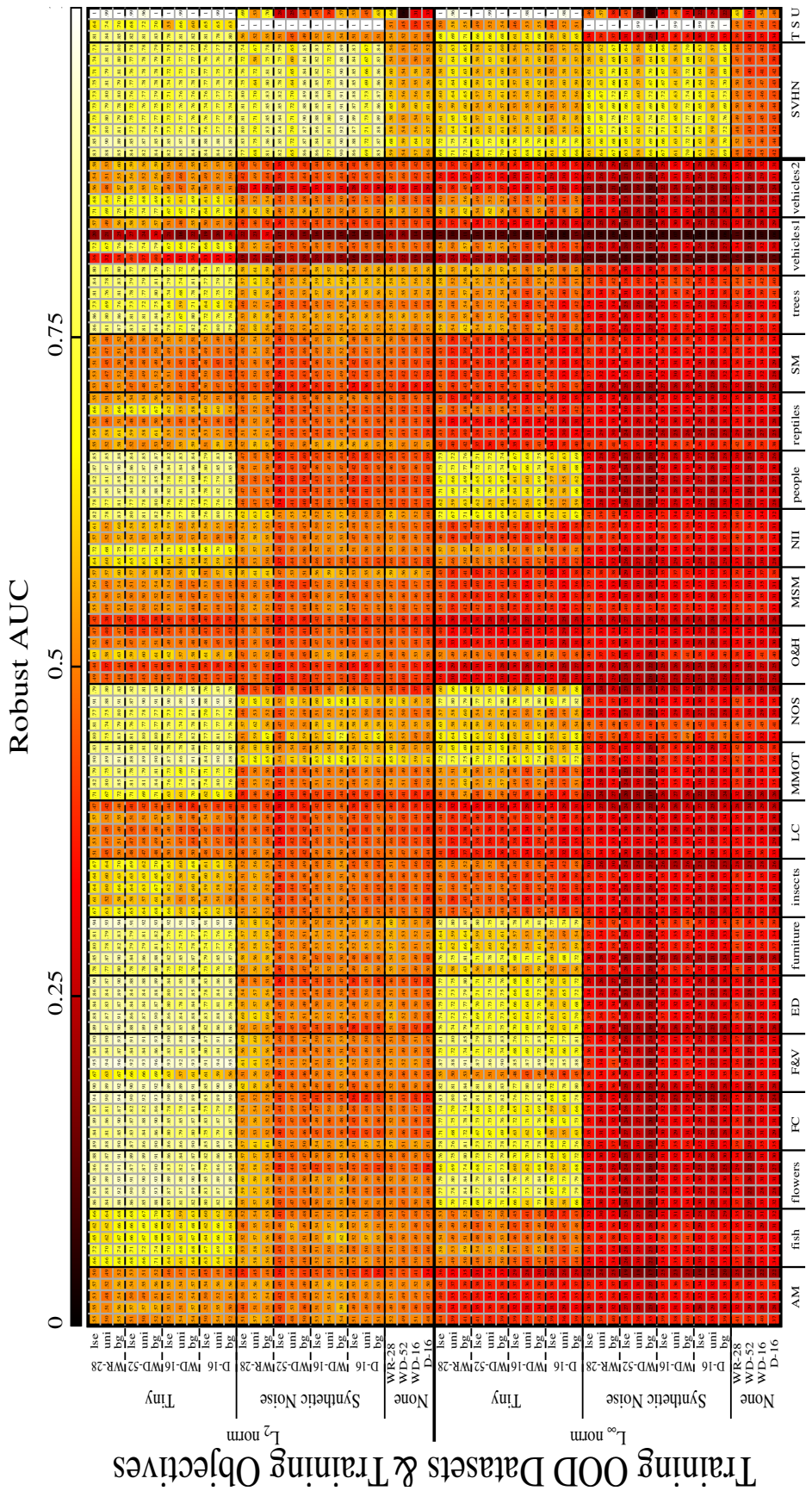

Figure 5.9: *Minimum OOD detection AUC over CIFAR-100 for a combination of norms, training datasets, architectures and training score functions (vertical axis) and test OOD datasets (horizontal axis). Abbreviations: AM: aquatic mammals, FC: food containers, F&V: fruit and vegetables, ED: household electrical devices, furniture: household furniture, LC: large carnivores, MMOT: large man-made outdoor things, NOS: large natural outdoor scenes, O&H: large omnivores and herbivores, MSM: medium-sized mammals, NII: non-insect invertebrates, SM: small mammals; T: Tiny, S: Synthetic Noise, U: uniform noise.*

Figure 5.10: *The probability that a given CIFAR-100 class sample is predicted as a given CIFAR-10 class member. Both ℓ_2 and ℓ_∞ based adversarial training is shown using the WRN-28-10 architecture and the s^{uni} objective, with $\mathcal{D}^T$ as OOD training dataset. Best viewed in combination with fig. 5.9.*

In our discussion, we will also refer to the data shown in fig. 5.10, where the probabilities of the predicted CIFAR-10 classes for each CIFAR-100 class are shown for both norms. (For clarity, here, only the largest WRN-28-10 model is shown, trained with the ρ_{out}^{uni} objective and the Tiny dataset as the OOD dataset.) These prediction probabilities represent useful extra information to explain the observed results.

Problematic Superclasses

Here, we list the superclasses which are not detected as OOD examples. That is, their AUC score remains close to the random guessing baseline (0.5) for the majority of the included classes, with respect to both norms. These superclasses are the following, indicating similar CIFAR-10 classes in parenthesis:

- aquatic mammals (airplane, ship, bird, frog, dog)
- large carnivores (cat, deer, dog, frog, horse)
- large omnivores and herbivores (horse, deer, bird, dog)
- medium sized mammals (cat, dog, frog, airplane, deer)
- reptiles (frog, bird)
- small mammals (bird, deer, frog, dog)

The predicted labels of these classes (fig. 5.10) reveal that they are classified to semantically similar CIFAR-10 classes, which explains why they are not detected as OOD. For example, large carnivores, medium sized mammals, and small mammals are frequently confused with CIFAR-10 mammal classes such as cat, dog and horse. The majority of the large omnivores and herbivores were confused with the horse and deer classes. A counterexample is the aquatic mammals superclass, which is confused with airplanes, ships and birds.

It is interesting to see that these superclasses were unable to improve over the random guessing baseline despite the increased model capacity, and the exhausting search over the detection methods and the training OOD datasets.

Superclasses with an inhomogeneous AUC

Here, we list the superclasses where the OOD detection is inconsistent, despite the semantic similarities within the superclass. The superclasses with the greatest variability are the following, indicating similar CIFAR-10 classes in parenthesis:

- fruit and vegetables (frog, cat, bird)

- household furniture (ship, cat, bird, dog)
- large man-made outdoor things (ship, truck)
- large natural outdoor scenes (ship, frog, deer, airplane, bird)
- non-insect invertebrates (frog, bird)
- people (dog, cat)
- vehicles 1 and 2 (automobile, truck, ship)

Typically, in these superclasses, most classes are detected as OOD while a few others are not. Examples of weakly detected classes (despite the overall good performance within the superclasses) are mushrooms from the fruit and vegetables superclass, and baby from the people superclass. Mushrooms are frequently confused with frogs and babies with dogs.

There are more examples for the case when a given class is detected better than the superclass average. Examples include the chair and wardrobe classes from the household furniture superclass, road from the large man-made outdoor things superclass, plain from the large natural outdoor scenes superclass, worm from non-insect invertebrates superclass, bicycle and motorcycle from the vehicles 1 and law-mower and rocket from vehicles 2 superclasses.

Easily Detectable Superclasses

Here, we list those superclasses that are well detected (AUC ≥ 0.7) indicating the most similar CIFAR-10 classes in parenthesis:

- flowers (frog, dog, bird)
- food containers (bird, frog, dog, cat)
- fruit and vegetables (frog, cat, bird)
- household electrical devices (truck, dog, cat, bird)
- people (cat, dog)

A common property of these superclasses is that in CIFAR-10 there are no semantically similar classes, although some classes do stand out in terms of prediction probability, for example, flowers are most likely identified as the bird CIFAR-10 class.

Superclasses Detectable under ℓ_2 but undetectable under ℓ_∞

The fourth type of superclasses are those that are better detected assuming the ℓ_2 attack model than with ℓ_∞:

- fish (airplane, frog, ship)
- insects (bird, frog)
- large man-made outdoor things (ship, truck)
- trees (truck, airplane, deer)

The fact that such superclasses exist is very interesting because it indicates that the attack model has a nontrivial influence on the generalizability of the adversarial training in the context of OOD detection.

5.10.2 Concluding Remarks on Generalization

We have seen that measuring just an average detection performance over different datasets is insufficient for truly understanding the performance of OOD detection. We identified four types of OOD classes. A closer look at the undetectable and the easily detectable classes showed that superclasses with less semantic similarity are easier to detect as OOD, and vice versa. Quite importantly, this effect seems to be *independent* of the model size and the training OOD dataset size. This suggests that, in contrast to the currently accepted wisdom that emphasizes dataset and model size, semantic similarity is the key factor in the case of OOD datasets. Another interesting finding is the difference between the ℓ_2 and ℓ_∞ threat models from the OOD detection point of view. The ℓ_2 threat model seems to offer better generalization. The combination of the two threat models appears to be a promising future direction.

5.11 Conclusions

We defined a design space, where one can define training objectives, detection methods and attack methods for the combination of the robust OOD detection problem and the robust classification problem with the help of a set of score functions. Also, we introduced a strong threat model in which both in-distribution and OOD samples are adversarially perturbed to mislead OOD detection.

We performed a thorough empirical evaluation of this framework. We found that adding an adversarial OOD objective to the training method does not hurt robust in-distribution accuracy, in fact, a significant improvement can be seen in some cases. This indicates that it is always safe to add such an objective.

We also found that it is impossible to pick a score function for robust OOD detection independently of how the model in question was trained. Instead, we get the best results when training and detection is based on the same score function. In other words, while non-robust OOD detection is more robust to the training procedure, in robust OOD detection it is more important to align the detection method with the training method, that is, to use the same score function in both. Also, a similar statement can be formulated in terms of the OOD detection method and the attack on this detection method. The most successful attack is performed using the same score function as the one used by the detection method.

We also pointed out that a deeper understanding of how OOD detectors generalize to unseen distributions is an interesting direction for future research.

Bibliography

[1] Maksym Andriushchenko, Francesco Croce, Nicolas Flammarion, and Matthias Hein. Square attack: A query-efficient black-box adversarial attack via random search. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXIII*, page 484–501, Berlin, Heidelberg, 2020. Springer-Verlag.

[2] Maximilian Augustin, Alexander Meinke, and Matthias Hein. Adversarial robustness on in- and out-distribution improves explainability. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conf., Glasgow, UK, August 23-28, 2020, Proc., Part XXVI*, volume 12371 of *Lecture Notes in Computer Science*, pages 228–245. Springer, 2020.

[3] Mohammad Azizmalayeri, Arshia Soltani Moakar, Arman Zarei, Reihaneh Zohrabi, Mohammad Taghi Manzuri, and Mohammad Hossein Rohban. Your out-of-distribution detection method is not robust! In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho, editors, *Advances in Neural Information Processing Systems*, 2022.

[4] Jimmy Ba and Diederik Kingma. Adam: A method for stochastic optimization. In *3rd Intl. Conf. on Learning Representations (ICLR)*, 2015.

[5] Yang Bai, Yisen Wang, Yuyuan Zeng, Yong Jiang, and Shu-Tao Xia. Query efficient black-box adversarial attack on deep neural networks. *Pattern Recognition*, 133:109037, 2023.

[6] Rainer Breitling, Patrick Armengaud, Anna Amtmann, and Pawel Herzyk. Rank products: A simple, yet powerful, new method to detect differentially regulated genes in replicated microarray experiments. *FEBS letters*, 573:83–92, 09 2004.

[7] Wieland Brendel, Jonas Rauber, Matthias Kümmerer, Ivan Ustyuzhaninov, and Matthias Bethge. Accurate, reliable and fast robustness evaluation. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett,

editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[8] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symp. on Security and Privacy (SP)*, pages 39–57, 2017.

[9] Nicholas Carlini and David A. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, pages 39–57. IEEE Computer Society, 2017.

[10] Jiefeng Chen, Yixuan Li, Xi Wu, Yingyu Liang, and Somesh Jha. Atom: Robustifying out-of-distribution detection using outlier mining. *In Proceedings of European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, 2021.

[11] Jiefeng Chen, Yixuan Li, Xi Wu, Yingyu Liang, and Somesh Jha. Robust out-of-distribution detection for neural networks. In *The AAAI-22 Workshop on Adversarial Machine Learning and Beyond*, 2022.

[12] Shang-Tse Chen, Cory Cornelius, Jason Martin, and Duen Horng (Polo) Chau. Shapeshifter: Robust physical adversarial attack on faster r-cnn object detector. In Michele Berlingerio, Francesco Bonchi, Thomas Gärtner, Neil Hurley, and Georgiana Ifrim, editors, *Machine Learning and Knowledge Discovery in Databases*, pages 52–68, Cham, 2019. Springer International Publishing.

[13] Shuyu Cheng, Yinpeng Dong, Tianyu Pang, Hang Su, and Jun Zhu. Improving black-box adversarial attacks with a transfer-based prior. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[14] Tarin Clanuwat, Mikel Bober-Irizar, Asanobu Kitamoto, Alex Lamb, Kazuaki Yamamoto, and David Ha. Deep learning for classical Japanese literature. Technical Report cs.CV/1812.01718, arXiv, 2018.

[15] Francesco Croce and Matthias Hein. Minimally distorted adversarial examples with a fast adaptive boundary attack, 2020.

[16] Yinpeng Dong, Shuyu Cheng, Tianyu Pang, Hang Su, and Jun Zhu. Query-efficient black-box adversarial attacks guided by a transfer-based prior. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):9536–9548, 2022.

[17] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 1625–1634. Computer Vision Foundation / IEEE Computer Society, 2018.

[18] Kevin Eykholt, Ivan Evtimov, Earlence Fernandes, Bo Li, Amir Rahmati, Chaowei Xiao, Atul Prakash, Tadayoshi Kohno, and Dawn Song. Robust physical-world attacks on deep learning visual classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[19] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *Machine Learning*, 107(3):481–508, Mar 2018.

[20] Weiwei Feng, Baoyuan Wu, Tianzhu Zhang, Yong Zhang, and Yongdong Zhang. Meta-attack: Class-agnostic and model-agnostic physical adversarial attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7787–7796, October 2021.

[21] Weiwei Feng, Nanqing Xu, Tianzhu Zhang, Baoyuan Wu, and Yongdong Zhang. Robust and generalized physical adversarial attacks via meta-gan. *IEEE Transactions on Information Forensics and Security*, pages 1–1, 2023.

[22] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, third edition, 1996.

[23] Ian J. Goodfellow and Jonathon Shlens Christian Szegedy. Explaining and harnessing adversarial examples. In *3rd Intl. Conf. on Learning Representations (ICLR)*, 2015.

[24] Will Grathwohl, Kuan-Chieh Wang, Joern-Henrik Jacobsen, David Duvenaud, Mohammad Norouzi, and Kevin Swersky. Your classifier is secretly an energy based model and you should treat it like one. In *Int. Conf. on Learning Representations*, 2020.

[25] Chuan Guo, Jacob Gardner, Yurong You, Andrew Gordon Wilson, and Kilian Weinberger. Simple black-box adversarial attacks. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2484–2493. PMLR, 09–15 Jun 2019.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[27] Matthias Hein, Maksym Andriushchenko, and Julian Bitterwolf. Why relu networks yield high-confidence predictions far away from the training data and how to mitigate the problem. In *IEEE Conf. on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 41–50. Computer Vision Foundation / IEEE, 2019.

[28] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *5th Int. Conf. on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conf. Track Proc.* OpenReview.net, 2017.

[29] Dan Hendrycks, Mantas Mazeika, and Thomas Dietterich. Deep anomaly detection with outlier exposure. In *Int. Conf. on Learning Representations*, 2019.

[30] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *ArXiv*, abs/1704.04861, 2017.

[31] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, 2017.

[32] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[33] Qian Huang, Isay Katsman, Horace He, Zeqi Gu, Serge Belongie, and Ser-Nam Lim. Enhancing adversarial example transferability with an intermediate level attack. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[34] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. Black-box adversarial attacks with limited queries and information. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2137–2146. PMLR, 10–15 Jul 2018.

[35] Nathan Inkawhich, Wei Wen, Hai (Helen) Li, and Yiran Chen. Feature space perturbations yield more transferable adversarial examples. In *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.

[36] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014. cite arxiv:1412.6980Comment: Published as a Conf. paper at the 3rd Int. Conf. for Learning Representations, San Diego, 2015.

[37] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[38] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *5th Int. Conf. on Learning Representations, ICLR*, 2017.

[39] Yann Lecun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, November 1998.

[40] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31, pages 7167–7177. Curran Associates, Inc., 2018.

[41] Qizhang Li, Yiwen Guo, and Hao Chen. Practical no-box adversarial attacks against dnns. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 12849–12860. Curran Associates, Inc., 2020.

[42] Yanjie Li, Yiquan Li, Xuelong Dai, Songtao Guo, and Bin Xiao. Physical-world optical adversarial attacks on 3d face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 24699–24708, June 2023.

[43] Shiyu Liang, Yixuan Li, and R. Srikant. Enhancing the reliability of out-of-distribution image detection in neural networks. In *Int. Conf. on Learning Representations*, 2018.

[44] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *Proc. 5th International Conference on Learning Representations (ICLR)*, 2017.

[45] Ye Liu, Yaya Cheng, Lianli Gao, Xianglong Liu, Qilong Zhang, and Jingkuan Song. Practical evaluation of adversarial robustness via adaptive auto attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15105–15114, June 2022.

[46] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th Intl. Conf. on Learning Representations (ICLR)*, 2018.

[47] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Int. Conf. on Learning Representations*, 2018.

[48] Pratyush Maini, Eric Wong, and Zico Kolter. Adversarial robustness against the union of multiple perturbation models. In Hal Daumé III and Aarti Singh, editors, *Proc. of the 37th Int. Conf. on Machine Learning*, volume 119 of *Proc. of Machine Learning Research*, pages 6640–6650. PMLR, 13–18 Jul 2020.

[49] Michael McCoyd and David A. Wagner. Background class defense against adversarial examples. In *2018 IEEE Security and Privacy Workshops, SP Workshops 2018, San Francisco, CA, USA, May 24, 2018*, pages 96–102. IEEE Computer Society, 2018.

[50] István Megyeri, István Hegedűs, and Márk Jelasity. Adversarial robustness of linear models: Regularization and dimensionality. In *Proceedings of the 27th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, Bruges, Belgium, 2019.

[51] István Megyeri, István Hegedűs, and Márk Jelasity. Adversarial robustness of model sets. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2020.

[52] István Megyeri, István Hegedűs, and Márk Jelasity. Attacking model sets with adversarial examples. In *Proceedings of the 28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, Bruges, Belgium, 2020.

[53] István Megyeri, István Hegedűs, and Márk Jelasity. Robust classification combined with robust out-of-distribution detection: An empirical a nalysis. In *International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021.

[54] István Megyeri, István Hegedűs, and Márk Jelasity. Combining robust classification and robust out-of-distribution detection: An empirical analysis. In *Progress in Artificial Intelligence*, 2023.

[55] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[56] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[57] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *The IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, June 2016.

[58] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *The IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2574–2582, June 2016.

[59] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011.

[60] Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In *Proceedings of the 36th International Conference on Machine Learning, (ICML)*, pages 4970–4979, 2019.

[61] Leslie Rice, Eric Wong, and Zico Kolter. Overfitting in adversarially robust deep learning. In Hal Daumé III and Aarti Singh, editors, *Proc. of the 37th Int. Conf. on Machine Learning*, volume 119 of *Proc. of Machine Learning Research*, pages 8093–8104. PMLR, 13–18 Jul 2020.

[62] Luke E. Richards, André Nguyen, Ryan Capps, Steven Forsyth, Cynthia Matuszek, and Edward Raff. Adversarial transfer attacks with unknown data and class overlap. In *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*, AISec '21, page 13–24, New York, NY, USA, 2021. Association for Computing Machinery.

[63] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[64] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, June 2018.

[65] Vikash Sehwag, Arjun Nitin Bhagoji, Liwei Song, Chawin Sitawarin, Daniel Cullina, Mung Chiang, and Prateek Mittal. Analyzing the robustness of open-world machine learning. In *Proc. of the 12th ACM Workshop on Artificial Intelligence and Security*, AISec'19, page 105–116, New York, NY, USA, 2019. Association for Computing Machinery.

[66] Rui Shao, Pramuditha Perera, Pong C Yuen, and Vishal M Patel. Open-set adversarial defense. In *European Conference on Computer Vision*, pages 682–698. Springer, 2020.

[67] Rui Shao, Pramuditha Perera, Pong C. Yuen, and Vishal M. Patel. Open-set adversarial defense with clean-adversarial mutual learning. *Int. J. Comput. Vision*, 130(4):1070–1087, apr 2022.

[68] Q. Song, H. Jin, X. Huang, and X. Hu. Multi-label adversarial perturbations. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1242–1247, Nov 2018.

[69] David Stutz, Matthias Hein, and Bernt Schiele. Confidence-calibrated adversarial training: Generalizing to unseen attacks. In Hal Daumé III and Aarti Singh, editors, *Proc. of the 37th Int. Conf. on Machine Learning*, volume 119 of *Proc. of Machine Learning Research*, pages 9155–9166. PMLR, 13–18 Jul 2020.

[70] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *Int. Conf. on Learning Representations*, 2014.

[71] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *2nd Intl. Conf. on Learning Representations (ICLR)*, 2014.

[72] Antonio Torralba, Rob Fergus, and William T. Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(11):1958–1970, November 2008.

[73] Florian Tramèr, Nicholas Carlini, Wieland Brendel, and Aleksander Mądry. On adaptive attacks to adversarial example defenses. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[74] Florian Tramer, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *6th Intl. Conf. on Learning Representations (ICLR)*, 2018.

[75] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *Proc. 6th International Conference on Learning Representations (ICLR)*, 2018.

[76] Jia Wang, Chengyu Wang, Qiuzhen Lin, Chengwen Luo, Chao Wu, and Jianqiang Li. Adversarial attacks and defenses in deep learning for image recognition: A survey. *Neurocomput.*, 514(C):162–181, dec 2022.

[77] Chen Wu, Ruqing Zhang, Jiafeng Guo, Maarten De Rijke, Yixing Fan, and Xueqi Cheng. Prada: Practical black-box adversarial attacks against neural ranking models. *ACM Trans. Inf. Syst.*, 41(4), apr 2023.

[78] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. Technical Report cs.LG/1708.07747, arXiv, 2017.

[79] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Yuyin Zhou, Lingxi Xie, and Alan Yuille. Adversarial examples for semantic segmentation and object detection. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 1378–1387, 2017.

[80] Kaidi Xu, Gaoyuan Zhang, Sijia Liu, Quanfu Fan, Mengshu Sun, Hongge Chen, Pin-Yu Chen, Yanzhi Wang, and Xue Lin. Adversarial t-shirt! evading person detectors in a physical world. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 665–681, Cham, 2020. Springer International Publishing.

[81] Xiao Yang, Chang Liu, Longlong Xu, Yikai Wang, Yinpeng Dong, Ning Chen, Hang Su, and Jun Zhu. Towards effective adversarial textured 3d meshes on physical face recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4119–4128, June 2023.

[82] Z. Yao, A. Gholami, P. Xu, K. Keutzer, and M. W. Mahoney. Trust region based adversarial attack on neural networks. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11342–11351, June 2019.

[83] Fei Yin, Yong Zhang, Baoyuan Wu, Yan Feng, Jingyi Zhang, Yanbo Fan, and Yujiu Yang. Generalizable black-box adversarial attack with meta learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–13, 2023.

[84] Zhenqin Yin, Yue Zhuo, and Zhiqiang Ge. Transfer adversarial attacks across industrial intelligent systems. *Reliability Engineering & System Safety*, 237:109299, 2023.

[85] Zheng Yuan, Jie Zhang, and Shiguang Shan. Adaptive image transformations for transfer-based adversarial attack. In Shai Avidan, Gabriel Brostow, Moustapha Cissé, Giovanni Maria Farinella, and Tal Hassner, editors, *Computer Vision – ECCV 2022*, pages 1–17, Cham, 2022. Springer Nature Switzerland.

[86] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *CoRR*, abs/1605.07146, 2016.

[87] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric P. Xing, Laurent El Ghaoui, and Michael I. Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, 2019.

[88] Jie Zhang, Bo Li, Jianghe Xu, Shuang Wu, Shouhong Ding, Lei Zhang, and Chao Wu. Towards efficient data free black-box adversarial attack. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15115–15125, June 2022.

[89] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8697–8710, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society.

Summary

The PhD thesis presents methods that contribute to advancing the research field of adversarial machine learning.

The dissertation consists of three major parts. In chapter 3, we analyzed the robustness of linear models from regularization and dimensionality points of view. chapter 4 presents attack algorithms that are able to generate such perturbation which can mislead multiple models simultaneously. In chapter 5, we analyze defense methods for the problem of robust classification and robust out-of-distribution detection.

Adversarial Robustness of Linear Models

Many machine learning models are sensitive to adversarial input, meaning that very small but carefully designed noise added to correctly classified examples may lead to misclassification. The reasons for this are still poorly understood, even in the simple case of linear models. In chapter 3, we study linear models and offer a number of novel insights. We focus on the effect of regularization and dimensionality. We show that in very high dimensions adversarial robustness is inherently very low due to some mathematical properties of high-dimensional spaces that have received little attention so far. We also demonstrate that—although regularization may help—adversarial robustness is harder to achieve than high accuracy during the learning process. This is typically overlooked when researchers set optimization meta-parameters.

Adversarial attacks on model sets

Machine learning models are vulnerable to very small adversarial input perturbations. In chapter 4, we study the question of whether the list of predictions made by a list of models can also be changed arbitrarily by a single small perturbation. Clearly, this is a harder problem since one has to simultaneously mislead several models using the same perturbation, where the target classes assigned to the models might differ.

This attack has several applications over models designed by different manufacturers for a similar purpose. One might want a single perturbation that acts differently on each model; like only misleading a subset, or making each model predict a different label. Also, one might want a perturbation that misleads each model the same way and thereby create a transferable perturbation. Current approaches are not applicable for this general problem directly. In chapter 4, we propose an algorithm that is able to find a perturbation that satisfies several kinds of attack patterns. For example, all the models could have the same target class, or different random target classes, or target classes designed to be maximally contradicting.

Combining Robust Classification and Robust out-of-Distribution Detection

Classification models in machine learning often make over-confident but incorrect predictions on input samples that do not belong to any of the output classes. Such samples are called out-of-distribution (OOD) samples. This problem has received considerable attention, because this represents a vulnerability similar to adversarial input perturbation, where models make incorrect predictions on seemingly in-distribution input samples that contain a very small but adversarial perturbation. In chapter 5,we are interested in models that are robust to both OOD samples and adversarially perturbed in-distribution samples. Furthermore, we require that OOD detection be robust to adversarial input perturbation. That is, OOD samples and in-distribution samples should not have adversarial perturbations that makes them appear to be in-distribution and OOD samples, respectively. Several related studies apply an ad-hoc combination of several design choices to achieve similar goals. One can use several functions over the logit or soft-max layer for defining training objectives, OOD detection methods and adversarial attacks. In chapter 5, we present a design-space that covers such design choices, as well as a principled way of evaluating the networks. This includes a strong attack scenario where both in-distribution and OOD examples are adversarially perturbed to mislead OOD detection. We draw several interesting conclusions based on our empirical analysis of this design space. Most importantly, we argue that the key factor is not the OOD training or detection method in itself, but rather the application of matching detection and training methods.

Összefoglalás

A PhD értekezés olyan módszereket mutat be, amelyek hozzájárulnak az ellenséges gépi tanulás kutatási területének előrehaladásához.

A disszertáció három fő részből áll. A 3. fejezetben lineáris modellek robosztusságát elemezzük a regularizáció és a dimenzionalitás szempontjából. A 4. fejezetben olyan támadási algoritmusokat mutat be, amelyek képesek olyan perturbációt generálni, amelyek egyszerre több modellt is félrevezethetnek. A 5. fejezetben védekezési módszereket vizsgálunk a robosztus osztályozás és a robosztus outlier detektálás problémájára.

Lineáris modellek ellenséges robusztussága

Számos gépi tanuló modell érzékeny az ellenséges bemenetre, ami azt jelenti, hogy a helyesen osztályozott példákhoz hozzáadott nagyon kicsi, de gondosan megtervezett zaj téves osztályozáshoz vezethet. Ennek okai még mindig tisztázatlanok, még az egyszerű lineáris modellek esetében is. A 3. fejezetben a lineáris modelleket vizsgáljuk, és számos új meglátást kínálunk. A regularizáció és a dimenzionalitás hatására összpontosítunk. Megmutatjuk, hogy nagyon nagy dimenziókban az ellenséges robusztusság eredendően alacsony a nagydimenziós terek néhány olyan matematikai tulajdonsága miatt, amelyek eddig kevés figyelmet kaptak. Azt is megmutatjuk, hogy - bár a regularizáció segíthet - az ellenséges robusztusságot nehezebb elérni, mint a nagy pontosságot a tanulási folyamat során. Ezt jellemzően a kutatók figyelmen kívül hagyják, amikor optimalizációs metaparamétereket állítanak be.

Ellenséges támadások modellhalmazok ellen

A gépi tanuló modellek sérülékenyek a nagyon kis bemeneti zavarokkal szemben. A 4. fejezetben azt a kérdést vizsgáljuk, hogy a modellek listája által készített előrejelzések listája is tetszőlegesen megváltoztatható-e egyetlen kis perturbációval. Ez nyilvánvalóan nehezebb probléma, mivel egyidejűleg kell több modellt félrevezetni ugyanazzal a perturbációval, ahol a modellekhez rendelt célosztályok eltérhetnek.

Ennek a támadásnak többféle alkalmazása is elképzelhető a különböző gyártók által hasonló célra tervezett modellek esetében. Lehet, hogy egyetlen olyan perturbációt szeretnénk, amely minden modellre másképp hat; például csak egy részhalmazt vezethetünk félre, vagy minden modell más-más címkét jósolhat. Az is előfordulhat, hogy olyan perturbációra van szükség, amely minden modellt ugyanúgy vezet félre, és ezáltal egy hordozható perturbációt hoz létre. A jelenlegi megközelítések nem alkalmazhatók közvetlenül erre az általános problémára. A 4. fejezetben egy olyan algoritmust javasolunk, amely képes olyan perturbációt találni, amely többféle támadási mintát is kielégít. Például az összes modellnek lehet ugyanaz a célosztálya, vagy különböző véletlenszerű célosztályok, vagy olyan célosztályok, amelyeket úgy terveztek, hogy hogy maximálisan ellentmondásosak legyenek.

A robusztus osztályozás és a robusztus outlier detekció kombinálása

A gépi tanulásban alkalmazott osztályozási modellek gyakran túlságosan magabiztos, de helytelen előrejelzéseket adnak olyan bemeneti mintákra, amelyek nem tartoznak egyik kimeneti osztályba sem. Az ilyen mintákat eloszláson kívüli (outlier) mintáknak nevezzük. Ez a probléma jelentős figyelmet kapott, mivel az ellenséges bemeneti perturbációhoz hasonló sebezhetőséget jelent, amely során a modellek hibás előrejelzéseket tesznek a látszólag eloszláson belüli bemeneti mintákra, amelyek nagyon kicsi, de ellenséges perturbációt tartalmaznak. Az 5. fejezetben olyan modellek iránt érdeklődünk, amelyek mind az outlier mintákra, mind az ellenségesen perturbált eloszláson belüli mintákra robusztusak. Továbbá megköveteljük, hogy az outlier felismerés robosztus legyen az ellenséges bemeneti perturbációval szemben. Vagyis az outlier minták és az eloszláson belüli minták esetén sem lehetnek olyan ellenséges hatású perturbációk, amelyek miatt azok eloszláson belüli, illetve outlier mintáknak tűnnek. Számos kapcsolódó tanulmány több tervezési lehetőség ad-hoc kombinációját alkalmazza hasonló célok elérése érdekében. A logit vagy softmax réteg felett több függvényt is használhatunk a képzési célok, az outlier felismerési módszerek és az ellenséges támadások meghatározására. Az 5. fejezetben bemutatunk egy olyan tervezési teret, amely tartalmazza ezen választási lehetőségeket, valamint a hálózatok kiértékelésének elvi módját adja meg. Ez magában foglal egy erős támadási forgatókönyvet, ahol mind az eloszláson belüli, mind az outlier példákat ellenséges módon megzavarják, hogy félrevezessék az outlier észlelést. Ennek a tervezési térnek az empirikus elemzése alapján számos érdekes következtetést vonunk le. A legfontosabb tanulság, hogy a kulcsfontosságú nem az outlier képzési vagy -felismerési módszer önmagában, hanem inkább a megfelelő felismerési és képzési módszerek alkalmazása.

Publications

Journal publications

[1] **István Megyeri**, István Hegedűs, and Márk Jelasity Combining Robust Classification and Robust out-of-Distribution Detection: An Empirical Analysis. In *Progress in Artificial Intelligence*(submitted for publication), 2023.

Full papers in conference proceedings

[2] **István Megyeri**, István Hegedűs, and Márk Jelasity Adversarial Robustness of Linear Models: Regularization and Dimensionality. In *Proceedings of the 27th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2019.

[3] **István Megyeri**, István Hegedűs, and Márk Jelasity Attacking Model Sets with Adversarial Examples. In *Proceedings of the 28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2020.

[4] **István Megyeri**, István Hegedűs, and Márk Jelasity Adversarial Robustness of Model Sets. In *Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2020.

[5] **István Megyeri**, István Hegedűs, and Márk Jelasity Robust Classification Combined with Robust out-of-Distribution Detection: An Empirical Analysis. In *Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN)*, IEEE, 2021.

Further related publications

[6] Gergely Pap and **István Megyeri** Translational Robustness of Neural Networks Trained for Transcription Factor Binding Site Classification. In *Proceedings of the 14th International Conference on Agents and Artificial Intelligence*, 2019.

[7] Tibor Csendes, Nándor Balogh, Balázs Bánhelyi, Dániel Zombori, Richárd Tóth, and **István Megyeri** Adversarial Example Free Zones for Specific Inputs and Neural Networks. In *Proceedings of the 11th International Conference on Applied Informatics (ICAI)*, 2020.

[8] Dániel Zombori, Balázs Bánhelyi, Tibor Csendes, **István Megyeri**, and Márk Jelasity Fooling a Complete Neural Network Verifier. In *The 9th International Conference on Learning Representations (ICLR)*, 2021.

[9] Ammar Al-Najjar and **István Megyeri** PCA improves the adversarial robustness of neural networks. In *Proceedings of the 30th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN)*, 2022.

Further publications

[10] **István Megyeri**, János Csirik, and Zoltán Majó-Petri Utasszámlálás a városi közösségi közlekedésben: mire lehet alkalmas több adat és a „free WiFi”?. In *Közlekedéstudományi Konferencia Győr 2019 Conference on Transport Sciences: Alternatív-Autonóm-Kooperatív-Komparatív Mobilitás*, 2019.

[11] **István Megyeri**, Melinda Katona, and László Nyúl A Novel Approach to Detect Outer Retinal Tubulation Using U-Net in SD-OCT Images. In *15th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*, 2019.

[12] Mohammed Mohammed Amin and **István Megyeri** Improving keyword spotting with limited training data using non-sequential data augmentation. In *The 12th Conference of PhD Students in Computer Science (CSCS)*, 2020.

[13] András Bánhalmi, Vilmos Bilicki, **István Megyeri**, Zoltán Majó-Petri, and János Csirik Extracting Information from Wi-Fi Traffic on Public Transport. In *International Journal of Transport Development and Integration*, 2021.

Acknowledgments

First of all, I would like to thank my supervisor, Dr Márk Jelasity, for directing and supporting my research. He showed infinite patience during my studies and taught me how science works.

I would also like to thank my colleagues and friends who helped me to realize the results presented here and to enjoy the period of my studies. Special thanks to Dr. István Hegedűs for his constant support during my studies. He was always open to discussing whatever new ideas I had.

Next, I wish to thank my wife, son, and family for their constant love, patience, and support throughout my education. It wouldn't have been possible without them.

This study was also supported by the National Research, Development and Innovation Office of Hungary through the Artificial Intelligence National Excellence Program(grant 2018-1.2.1-NKP-2018-00008), by the Hungarian Ministry of Human Capacities (grant 20391-3/2018/FEKUSTRAT), by grant TUDFO/47138-1/2019-ITM of the Ministry for Innovation and Technology, Hungary, and by the project "Integrated program for training new generation of scientists in the fields of computer science", no EFOP-3.6.3-VEKOP-16-2017-0002, funded by the European Union and co-funded by the European Social Fund, by the Ministry of Innovation and Technology NRDI Office within the framework of the Artificial Intelligence National Laboratory Program and the Artificial Intelligence National Excellence Program (grant 2018-1.2.1-NKP-2018-00008), as well as grant NKFIH-1279-2/2020, by the European Union project RRF-2.3.1-21-2022-00004 within the framework of the Artificial Intelligence National Laboratory, and project no. TKP2021-NVA-09, implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme.