Leveraging Feature Extraction to Detect Adversarial Examples

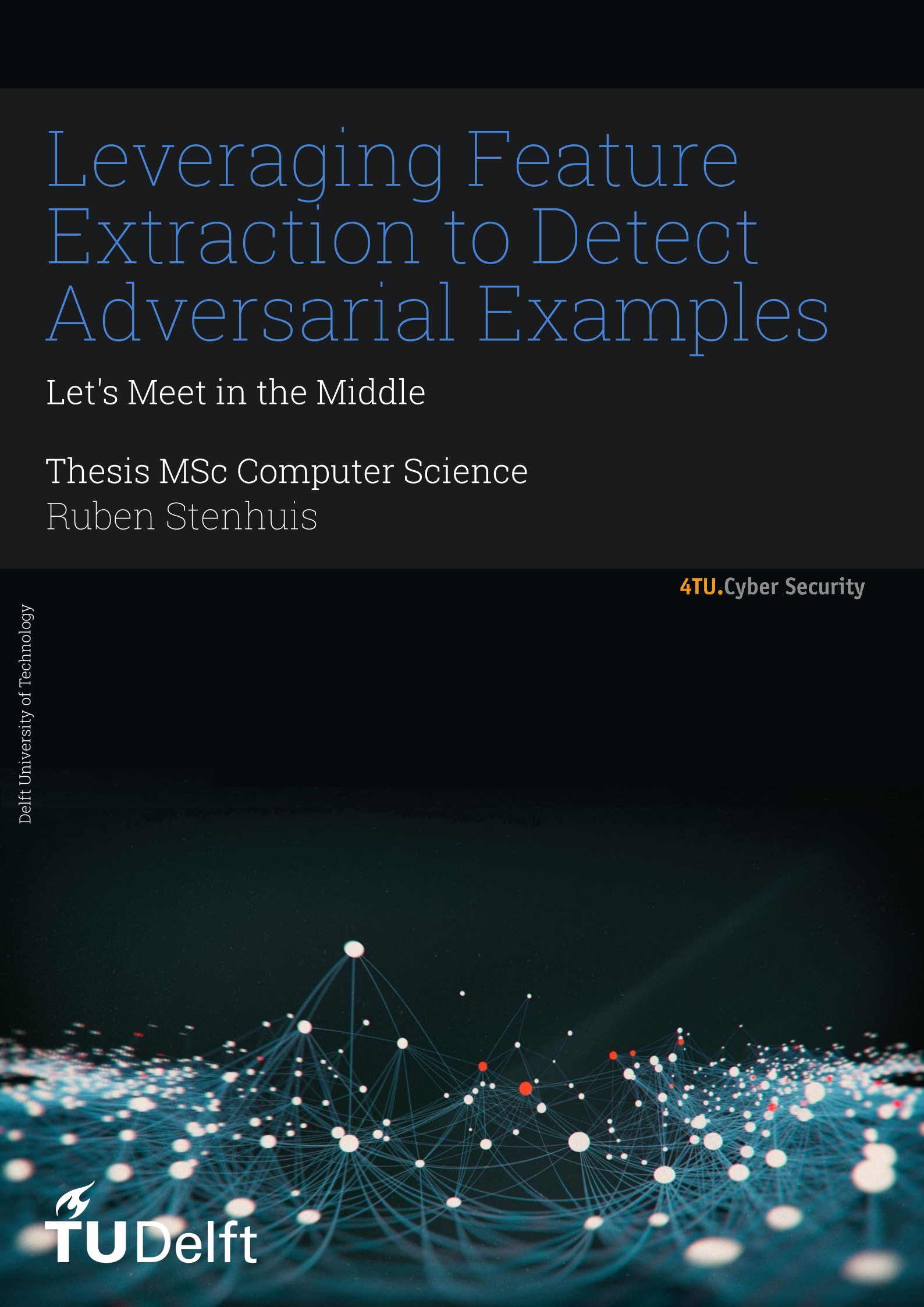
Let's Meet in the Middle

Thesis MSc Computer Science

Ruben Stenhuis

4TU.Cyber Security

Delft University of Technology



TU Delft

Leveraging Feature Extraction to Detect Adversarial Examples

Let's Meet in the Middle

by

Ruben Stenhuis

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday October 3, 2024 at 12:45.

Student number:	4927508	
Project duration:	November 6, 2023 – October 3, 2024	
Thesis committee:	Dr. ir. S. E. Verwer,	TU Delft, thesis advisor
	Dr. K. Liang,	TU Delft, supervisor
	Dr. J.E.A.P. Decouchant,	TU Delft
	D. Liu,	TU Delft, daily supervisor

Cover:	From an IBM blog [Sol19] of Stephanie Solomon
Style:	TU Delft Report Style

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft

4TU.Cyber Security

Preface

Adversarial learning stands as a cornerstone in contemporary machine learning research, offering insights into the vulnerabilities and limitations of neural network models. Through the investigation of adversarial examples and the development of defense mechanisms, researchers strive to enhance the robustness and reliability of machine learning systems.

One prominent issue addressed in this thesis is the presence of bias in neural network models towards benign data, *i.e.* data without adversarial perturbation. In general, biased models can perpetuate and amplify societal biases, leading to inequitable outcomes and reinforcing existing disparities. By identifying and mitigating such bias in machine learning algorithms we can foster fairness and accountability in decision-making processes.

Bias may also be injected at training time to form a backdoored or poisoned model. As frequently stated, 'garbage in equals garbage out': misguided evidence at training time ultimately serves as an unmediated source of bias for machine learning models. Understanding the origins of bias and its propagation in neural networks is essential for developing strategies to combat it effectively.

Through rigorous experimentation and analysis, this thesis endeavors to contribute to the ongoing dialogue surrounding adversarial learning and bias mitigation in machine learning. By addressing these challenges head-on, we aim to pave the way for more robust and equitable artificial intelligence systems.

Acknowledgement: I would like to express my sincere gratitude to my daily supervisor, Dazhuang Liu (TU Delft), who made this project possible for me.

Ruben Stenhuis
Delft, September 2024

Abstract

Previous research has explored the detection of adversarial examples with dimensional reduction and Out-of-Distribution (OOD) recognition. However, these approaches are not effective against white-box adversarial attacks.[1] Moreover, recent OOD methods that utilize hidden units hinder the scalability of the target model.

For that reason, various explanations of adversarial examples are studied to get a better understanding about its properties and anomalies. Furthermore, we discuss the added value of using natural scene statistics and utility functions to improve the relevance of the features for detection. By utilizing the anomalies we identified for adversarial examples in an ensemble, this thesis is the first to propose a robust solution for adaptive and white-box attacks.

Particularly, we address these challenges with MeetSafe. A Gaussian Mixture Model that leverages principal component analysis, feature squeezing, and density estimation to detect adaptive white-box adversaries. Furthermore, our enhanced Local Reachability Density (LRD) algorithm further improves the efficiency of state-of-the-art OOD methods. In particular, the proposed LRD enhances scalability by feature bagging hidden units with large absolute Z-scores. We then show that predictors, including LRD, are far more effective in ensembles like MeetSafe which supports prior conjectures that a range of different heuristics may further constrain adversaries when combined.

Extensive experiments on 14 models show that MeetSafe detects adaptive perturbations with an accuracy of 62% on STL-10, 75% on CIFAR-10, and 99% on MNIST, using either adversarial training or Reverse Cross Entropy (RCE), achieving an improvement of at least 8.1% for each evaluated method by averaging across the three datasets.

[1] Our detector is designed for an adaptive, white-box adversary that has complete knowledge of model, dataset, and defense. This is a stronger assumption than state-of-the-art attacks like AutoAttack [Tra+20].

Contents

1

Introduction

Neural networks paved the way to a diverse field in science. Convolutional Neural Networks (CNN), for example, have significantly improved the performance of computer vision. This made some promising steps towards some essential challenges. Challenges that concerns medical treatment and wellbeing [Lit+17], object detection [Li+22], or facial recognition [Li+22; Zha+03]. In most of these applications, the reliability and robustness of machine learning models are of paramount importance. This robustness is currently lacking for neural networks. A slight and imperceptible **perturbation** is enough to cause state-of-the-art models to misclassify at a high rate [Sze+14].

Perturbations that cause these misclassifications are generated by evasion methods, which involve strategically altering the input data, typically with gradient information, to deceive the neural network. Most evasion methods craft near-optimal adversarial examples, which use the ℓ_p-norm as a part of its objective. By minimizing the example's ℓ_p distance from benign input, it is likely that the crafted sample is imperceptible to humans. Popular evasion methods include C&W [CW17b], DeepFool [MFF16], and the Fast Gradient Sign Method (FGSM) [GSS15].

Adaptive adversarial examples complicate the task of developing robust neural networks even further, as these dynamically adjust the perturbation based on the model's defenses. A common approach to generate adaptive adversarial examples is with existing evasion methods on gradient information of both the defense and target model. Appending the logits of neural networks with the ones of the detector would thereby allow the evasion method to account for a detector [CW17a].

To mitigate this, some defenses intentionally break gradient descent and cause obfuscated gradients [ACW18]. Non-differentiable defenses can't be evaded with iterative evasion methods, and thus its authors often claim robustness. For instance, dropout [Fei+17] has stochastic gradients and RAID [ECW20], that randomly chooses out of a pool of predictors, has nonexistent, shattered gradients. Unfortunately, **Backward Pass Differentiable Approximation** (BPDA) [ACW18] successfully addressed such obfuscated gradients by substituting the non-differential parts with an estimate. Obfuscating the gradients is thus *not* the solution for adaptive adversarial examples.

Previous work proposed many methods to optimize the robustness of machine learning models [SN20] and detect [Ald+22] any **adversarial example**. However, to the best of our knowledge, no previous defense achieved useful performance in an adaptive white-box setting. Existing methods that do, either obfuscate the detector's gradient or render the evaluation of exact gradients too costly [CSG20; Hu+19; Rag+21; Tia+21; ACW18]. Solutions like these may therefore not be guaranteed to be secure.

Certified methods [Wen+18b; RSL18] can also be used for adversarial detection by means of minimum distance decoding [Tra22], but such methods generally operate on small ℓ_p distances. This would be ineffective for semantic examples [GSG20] that manipulate color or shadows to reach substantially large and imperceptible perturbations. As such, Ghiasi *et al.* [GSG20] show that any perturbation on semantic attributes – such as shadows – may be just as effective as contrived noise. Similarly, given the

vastness of an image's semantics, supervised detection may be inadequate for evasion attacks [ZH18].

The increasing adoption of neural networks in all kinds of critical environments necessitates ongoing research and innovative approaches to ensure the security and reliability of neural networks in practical applications. This work aims to explore semi-supervised detection methods and propose new methodologies to enhance the robustness of neural networks against sophisticated and adaptive adversarial examples.

1.1. Problem Statement

Various attempts have been made to detect adversarial examples. However, many were later circumvented by Carlini *et al.* [CW17a; ACW18; AC18; Tra+20] using white-box attacks that adaptively optimize against the detector. One example of such method is based upon Principal Component Analysis (PCA) [HG17]. According to Carlini, this may be evaded by limiting perturbations along low-rank eigenvectors.

Other works [HG17; Ald+22] proposed to combine discrepant detectors with other defenses to constrain the adversary's level of pathology. Indeed, even imperfect detectors may constrain the adversary. The adversary should, in that case, consider and evade each defense. Combining defenses to form a more robust detector has been referred to as the *'meet the defense'* approach. Unfortunately, both papers only stated the idea and did not construct such a detector.

Meanwhile, some defenses [ZH18; Fei+17] do not scale well with the increasing number of parameters when each (hidden) unit of such models is treated as a separate feature. For example, the full covariance matrix Σ, used in I-Defender [ZH18], is scaled as $\mathcal{O}(d^2)$ with respect to the input dimension d. Likewise, a rise in the number of features exponentially degrades the performance of any method that employs the Euclidean distances on these features, such as Feinman *et al.*'s method [Fei+17], due to the curse of dimensionality.

1.1.1. Research Objective

We hypothesize that the linearity of the predictor is one of the reasons that detection methods can be evaded. Other explanations may be because the method is narrow, or overfitting. For one, we explained how PCA has been shown to be too linear. Conversely, non-linear methods [Fei+17; XEQ18] may not cover every perturbation type. As was suggested by Feinmann *et al.* [Fei+17]: multiple methods should be considered for adversarial examples that are 'near'- and 'far' from benign training samples, since Density-based and variance-based methods only cover certain levels of perturbation.

To test the above hypothesis, this thesis proposes a new detection method to resolve white-box security and the scalability challenges by leveraging the Local Reachability Density (LRD) metric [Bre+00] and Gaussian Mixture Models (GMM). LRD is a distance-based method that can determine global outliers using a smoothed k-distance called *reachability*. For LRD, we use two utility functions based on a hidden unit's Z-scores or rate of change under perturbation. This should enhance its memory efficiency, and improve the scalability of the underlying k-Nearest Neighbors (kNN) density estimation. LRD will then be combined with two other detection methods in a Gaussian Mixture Model (GMM) called MeetSafe. Our MeetSafe effectively *meets* detectors of different statistical regularities. MeetSafe can easily be applied on larger models, as the feature space is reduced by only considering the 10 best features according to the utility functions.

We evaluate MeetSafe on models that were trained on CIFAR-10, Tiny-ImageNet, STL-10, and MNIST. Most of our models use adversarial training [GSS15] or Reverse Cross Entropy (RCE) [Pan+18]. We tested MeetSafe under both the grey- and white-box threat model at a True Negative Rate (TNR) of around 90%. Experiments on CIFAR-10 show that MeetSafe attains a 74%+ detection accuracy under the white-box threat model; and a 96%+ detection accuracy against FGSM. The results will also show a 79%+ accuracy for PCA under white-box attacks, contradicting some prior work. Our main contributions can be summarized as follows:

i MeetSafe, a detection algorithm for adaptive adversarial examples.

ii Two utility functions that allow LRD and other detectors to scale based on a unit's Z-scores or rate of change under perturbation.

iii Theoretical discussion of four properties of adversarial examples that induce essential measures for detection methods.

iv Theoretical evaluation of methods using local density with a rational difference equation.

v Extensive empirical evaluations on 4 datasets and 14 models that show effectiveness of whitening (*i.e.* PCA) [HG17] and MeetSafe under adaptive white-box attacks.

1.2. Outline

The thesis will be presented in the following structure. Chapter 2 gives some background on the security game for adversarial learning. The chapter assesses concepts such as robustness, perturbation, convexity, and transferability. Chapter 3 will show further details of our contribution, motivation, and points of interests for CNNs. Chapter 4, explains why and how adversarial examples can be created. It will introduce explanations like the boundary tilting perspective and linear explanation. Chapter 5, discusses our detector in detail. Chapter 6 will contain the methodology and results of our experiments on the GMM and individual detectors. Chapter 7 concludes this work.

2

The Security Game of Adversarial Learners

The thesis begins with an elementary discussion on neural networks. This introduces a couple of concepts that are necessary to understand the underlying vulnerabilities of neural networks. The chapter then continues with a discussion on adversarial learning and different attack types. One can show strong connections to adversarial example generation, which is the main subject of this thesis. The chapter ends with a taxonomy of these attack types.

The basic building block of a neural network is a neuron, which is composed of an activation function such as ReLu or Sigmoid. A neural network is a connected graph of these neurons. This allows the (deep) neural network to recognize complex and non-linear relations of the input features. Something that other AI techniques as Support Vector Machines (SVM) have difficulty with. The main argument for this difficulty is one of dimensionality. Neural networks allow for the adaption of its basis functions, which enables neural networks to be sparser than linear models. This is essential for big data, as increasing the dimensionality with fixed basis functions will result in the **curse of dimensionality** . A downside of neural networks is that the likelihood function is no longer **convex**. This proves to be a major issue when estimating the **robustness** of neural networks.

Another argument can be made on the complexity of neural networks. The dimensionality of its inputs and outputs are frequently determined by the dataset (e.g. amount of classes, features, etc.). The amount of hidden units are left free for tuning, which may adjust the generalization capability of a neural network. However, complex networks may also be more vulnerable, especially on large feature dimensions [GSS15]. In combination with non-convexity, **dimensionality** aggravates the estimation of its robustness.

Different architectures of neural networks are proposed for specific problems. One of these architectures is an autoencoder, which are perfect in modelling sparse nonlinear dynamics [Cha+19]. We will focus on the Resnet-50 [He+16] and Visual Geometric Group-13 (VGG-13) [SZ15] architecture. These architectures are popular for image recognition and often used to test solutions for adversarial learning. Both models will be explained after a short and formal introduction of neural networks.

2.1. Neural Networks

As discussed, neural networks resemble a connected graph of neurons. Outputs of neurons resemble a basis function $f^{(l)}(X|W,B) : \mathbb{R}^m \rightarrow \mathbb{R}^L$ that is controlled by the parameters $W^{(l)}$ and its biases $B^{(l)}$ up to the layer l [Bis06]. the basis function is formed by a series of similar transformations with a nonlinear function $h(\cdot)$ of a linear input, called its activation. When we define the dimension of a layer to be $D^{(l)}$, then the first fully connected layer can be thought of like shown in Eq. 2.1.

$$f_j^{(0)}(X|W^{(0)}, B^{(0)}) = h(\sum^{D^{(0)}} w_{i,j}^{(0)} x_i + b_j^{(0)}) \tag{2.1}$$

Subsequent layers wrap around the output of its previous layer. Meaning that $x_i \in X$ would be substituted by $f_i^{(l-1)}(X|W^{(l-1)}, B^{(l-1)})$. These outputs are referred to as hidden units of the neuron (i.e. the basis function of $f^{(l)}$). Finally, the last layer outputs **logits** of the model. Logits are especially interesting for evasion attacks (Sec. 4.2.1) as it counteracts **vanishing gradients** and a wrong sense of relative importance between labels when compared to the usage of probits.

Normally, the logits are processed by another nonlinear function to give probits as output vector Y. The choice of activation function is determined by the set of properties Y should have. Probits act as probabilities and Softmax ensures this exact property. It ensures two properties:

- **Probability values:** Y satisfies $0 \leq y_i \leq 1$
- **Additivity property:** $\sum^{|Y|} y_i = 1$

For simplicity, logits will be defined to be $Z(X)$ and softmax outputs to be $F(X)$. A summary of this notation can be found in Appendix E.

2.1.1. Convolution & VGG

Classification of images requires some more regularization of the model. The plain model that is introduced could in principle learn any invariance in the image data. Think of rotations or translations. However, this would ignore one obvious correlation among invariances, which is the relative location of the pixels. This will often not change much, or only slightly. At least, its disparity with the change of the raw data is enormous. That is why CNNs are ubiquitous for image recognition tasks.

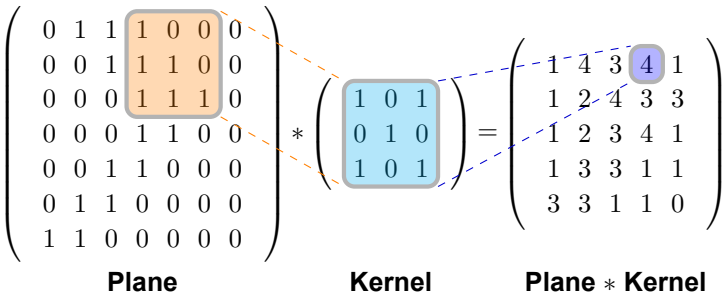


Figure 2.1: The convolution operation

In addition to the fully connected layers (Eq. 2.1), CNN's have at least two other operations: **convolution** and **pooling** . Convolution is illustrated in Fig. 2.1. On the left, the units of one plane are shown. Each convolutional layer may operate on multiple planes or feature maps. The kernel is of a predetermined size and slides over the whole feature maps (omitting its stride). This results in the convolution on the right. The cells in the kernel function are just like neurons. They have their own linear combination of weights and biases.

Pooling also operates with a kernel, but will not have weights. It takes pooling operations (averaging, centering, etc.), the size of the feature map is therefore typically smaller after a pooling layer. In theory, convolution layers provide some weight sharing across the feature map that recognizes patterns regardless of its position. Pooling layers removes small invariance of the feature map.

VGG is a convolutional network that is designed for ImageNet. It improves upon earlier networks with a small kernel size of 3x3 for convolutions and 2x2 for pooling. This enables the network to go deeper on similar input sizes. In our case, the depth of VGG equals ten layers. Followed by one fully connected

layer (original VGG has three). That is because it is used for smaller-sized images, in our case (Chap. 6). For detailed information about the profile of the architecture, we refer to the original paper [SZ15].

2.1.2. Convexity & Resnets

Intuitively, the depth of neural networks have the potential to greatly decrease any under-fitting. After all, more hidden units would increase the complexity of the model. This is not entirely true. To explain this, recall the meaning of convexity. A classifier $g(\cdot)$ of an input space $\mathfrak{I}$ is convex when the following holds (Eq. 2.2):

$$g(tX_1 + (t-1)X_2) \leq tg(X_1) + (t-1)g(X_2) \tag{2.2}$$

where $\forall X_1, X_2 \in \mathfrak{I}$ and $0 \leq t \leq 1$. Non-convex functions can have disjoint local minima, convex functions only have one (if it exists). During the training of neural networks, the model converges to a local minimum of the error function. This is equivalent to maximizing the likelihood function of the neural network. One example of an error function is the cross entropy loss for multi-class classification. These Likelihood/error functions are, just like the error function of the model, non-convex implying the presence of (suboptimal) local maxima/minima, which has some consequences for the model's robustness and convergence.

The authors of Resnet [He+16] called this specific issue with non-convexity a **degradation problem**. They demonstrated that accuracy of very deep neural networks degrade rapidly when the amount of hidden units increased. This is not caused by overfitting, as it is characterized by a high training error; but rather the convergence to local minima.

Residual learning was proposed as a solution. It allows neural networks to reduce to shallow variants with an identity mapping after a set of convolutions. The dimension must remain equal before and after the set of convolutions. The identity mapping $f^{(l)}(X|W,B) + P_l X$ of some sets, called bottlenecks, thus need a projection matrix P_l if it is not equal. Resnets are able to reach much higher depths (up to 152 layers) than VGG with this technique. The interested reader is again referred to the original paper for Resnet's profile [He+16].

2.2. Threat Model

Training on benign data involves minimizing an error function $\mathcal{L}_f(\mathbf{X}|W,B) : \mathbb{R}^{\mathcal{B}\times m} \rightarrow \mathbb{R}$. The objective of the error function changes, however, when an adversarial setting is considered. Then the model's parameters W and B should remain its accuracy under some noise δ. The **empirical risk** $\mathcal{R}$ for dataset $\mathfrak{D}$ would provoke a **min-max problem** , when the error function is minimized [SN20]. Instead of minimizing the desired loss, the model should minimize the loss under the most perturbation. Solving this equation requires a solution to the inner maximization (Eq. 2.3).

$$\mathcal{R}_f^* = \min_{W,B} \frac{1}{|\mathfrak{D}|} \sum_{\mathbf{X}_i \in \mathfrak{D}} \max_{\delta} \mathcal{L}_f(\mathbf{X}_i + \delta|W,B) \tag{2.3}$$

The optimization of this term is NP-hard with deterministic techniques [RSL18, p.4]. Many certification methods have been proposed that relax Eq. 2.3 to a convex hull (Sec. 4.4.3); but this comes at a cost for the model's accuracy or computational efficiency during training. Another approach is to substitute the minimum provided by powerful attacks for approximation. The achieved δ of these attacks is measured with the ℓ_p**-norm** . The norm computes the Minkowski distance $||X - X'||_p$ between the sampled benign image X and resulting **adversarial example** X'. The introduction (Chap. 1) already teased the downside of this metric: it does not simulate the perception of human vision. This is misleading when some invariances of images cause large changes in the pixel values.

2.2.1. Robustness

The ℓ_p-norm is often assumed to represent the severity of an attack [MFF16]. Following this argument, robustness of a model would be the average **perturbation** that is needed to misclassify an image (Eq. 2.4). This kind of usage of the ℓ_p-norm is a misconception. The attacker is far more likely, in practice, to

maximize the perturbation with **high confidence attacks** . A lot of defenses failed to be effective under these and white box attacks [CW17a; ACW18; CW16; AC18; Tra+20; He+17]. We are not even aware of any detector that is secure in a **white box setting**. Security Evaluation Curves may also capture robustness by plotting the model's accuracy along an increasing ℓ_p-norm [BR18].

$$\frac{1}{|\mathfrak{D}|} \sum_{X_i \in \mathfrak{D}} \frac{||X - X'||_p}{||X||_p} \tag{2.4}$$

A related factor to robustness that is not always is covered is the transparency of neural networks. Whilst, it plays a large part in the robustness. Without interpretability, any human intervention against intentional malice is hindered. Decision can thus be manipulated when opaque models were used for decision-making. In fact, following the Chinese Room argument, we can question if neural networks may ever be strong AI with this vulnerability, regardless of how intelligently or human-like the decisions of the model may be. Currently, it seems that neural networks can't replace humans. In Europe, recent regulation enforces **explainable Artificial Intelligence** (XAI) [GF17] . The European Union argues that administrators should be able to reflect on their systems. Especially when the system operates on sensitive data.

Methods that can visualize these boundaries (i.e. white-box models) are therefore of high value. It helps system designers to trust and explain the model's decisions. Machine learning models should thereby be able to detect novelties that do not fit into the training data. Human intervention can then be asked for edge cases [BR18]. The dilemma here is one of the model's accuracy against its robustness. How many human resources or mitigations are we prepared to invest to ensure correct classifications before its benign accuracy is of higher importance.

Takeaway 1.
Knowledge of anomalies is crucial for human intervention, however there are shortcomings for these three points:

- ☞ *Current certified defenses for neural networks are limited to convex relations.*
- ☞ *The ℓ_p-norm does not comprise the bias of human vision.*
- ☞ *White box detection is yet unsolved.*

2.2.2. Taxonomy

This thesis focuses on adversarial examples. Adversarial examples are strongly related with two other categories of adversarial attacks (model extraction and poisoning), depending on the strategy of an adversary. Model Extraction is combined with adversarial examples to evade models in a black box scenario. Also, some poisoning attacks use similar gradient-based techniques to manipulate models at training time. Essential dimensions where these three types of adversarial attacks differ are:

- **Timing:** Poisoning attacks are performed during the training phase of the model by injecting synthetic and malicious data. Of course, the injection may happen earlier. For example, malfunctioning sensors or aggregators in a Wireless Sensor Network (WSN) can alter the dataset. On the other hand, Federated Learning solutions expose the training process directly. Evasion attacks assume a converged model.
- **Knowledge:** Adversarial knowledge can be defined in terms of a space Θ [BR18]. This space can include multiple components to which the adversary has access to. In a white box setting, this includes any information that is interesting for the adversary. For evasion attacks these are often the gradients, defenses protocol, and architecture: $\Theta = \{\nabla_X f(X), \Pi, f(\cdot)\}$. The space of poisoning attacks diverges from this. Weaker assumptions for the adversary are the grey- and black box setting. **Grey box** attacks refer to those in which the attacker has no knowledge about the defenses. Whereas **black box** attacks only assume knowledge of soft labels $F(X)$ and input without information on the model or dataset.
- **Capability:** Adversaries are also influenced by the constraints imposed by the system. For example, adversarial examples must contort itself to evade any security measure in place. The

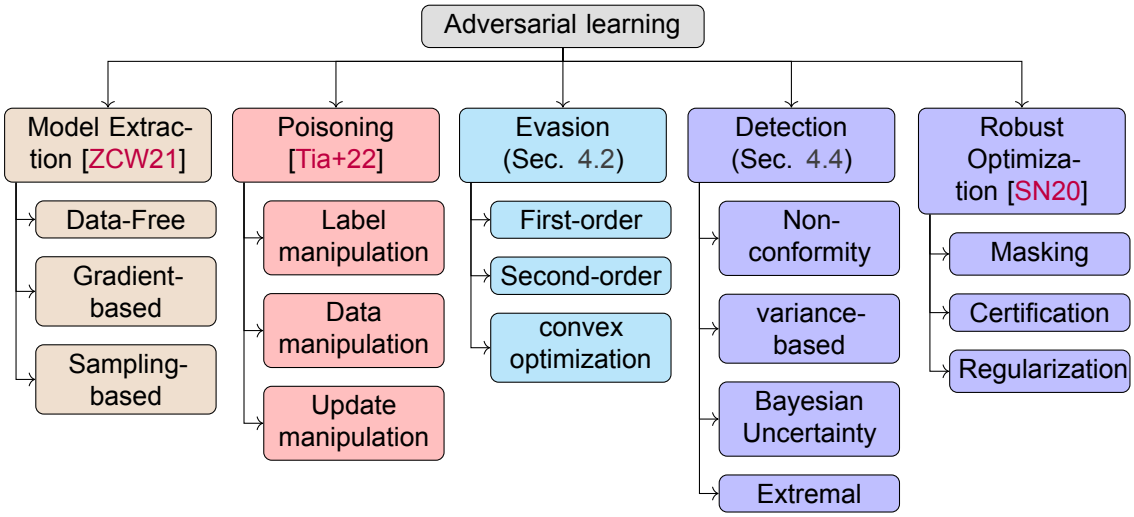


Figure 2.2: Our taxonomy for adversarial learning

question of what and how data is malleable is defined by these constraints. Poisoning attacks are also known as causative attacks, whilst evasion attacks are exploratory [BR18]. They thus distinct in what data is manipulated. Poisoning attacks may manipulate both training and test data. This is not the case with evasion attacks.

- **Intention:** Model Extraction causes a privacy violation. Extraction of variables like model parameters give insight to the training data and decision boundaries via its returned confidence. This aim is in contrast with evasion and poisoning. Those affect the integrity of the model.
- **Specificity:** Attack specificity is about the target that is chosen by the attacker [BR18]. During in vitro evaluation we consider three approaches of targeting an incorrect class: best, worst, and average case [CW17b]. These target the incorrect class that is close, far, or any proximity from the sampled point X respectively.

Fig. 2.2 subdivides the main categories for adversarial learning based on earlier proposed taxonomies [ZCW21; Tia+22; SN20] and our literature review in Chap. 4. In the literature review, different explanations are given for adversarial examples. Each perspective results in an observable characteristic of adversarial examples. To this end, we can categorize detectors by which of these novelties they measure. Before delving into the specifics, the sections below provide a high overview of the different attack types first.

Evasion

The work of Szegedy *et al.* [Sze+14] was the first to show the existence of adversarial examples . A small perturbation is enough to cause misclassifications of neural networks at a high rate. Manipulation of the input towards the adversarial direction is called evasion. The adversarial example can be very close to the benign input, with on average a Euclidean distance of 0.3 for CIFAR models [CW17b].

The intra-boundary distance of different neural network architectures are much smaller than the perturbation [Tra+17]. As a result, examples often transfer to multiple architectures and even other learning algorithms. **Transferability** is particularly relevant in black-box attack scenarios, where the attacker may not have complete knowledge of the target model's architecture and parameters. This property is far less present for CNNs or models that have independent subsets of input features. One explanation for transferability is Model-Agnostic perturbation, which is perturbation between class norms. Each model can be evaded towards this direction [Tra+17].

Adversarial examples don't have to be close to the benign data point. There are other perturbations with different levels of pathology. Some research has shown fooling images that are also unrecognizable by human vision [NYC15; RD18]; physical perturbations with shadows [Zho+22]; and semantic

perturbations [GSG20]. This thesis will focus on near-optimal perturbations and thus follow-up work of Szegedy. Most methods of this kind utilize the first or second-order gradient of a model (Sec. 4.2). These seek to minimize the following objective (Eq. 2.5):

$$\mathcal{A}(\mathbf{X'}, \mathbf{X}) = ||\mathbf{X} - \mathbf{X'}||_p + \lambda(\mathcal{L}_f(\mathbf{X}|W, B) - \mathcal{L}_f(\mathbf{X'}|W, B)) \tag{2.5}$$

where $\lambda \in [0, \infty)$. This objective may have additional constraints, like its specificity and the capability of the attacker.

Although it is not the main focus of this work, the robustness metric (Sec. 5.3.2) may prove useful for **semantic examples** . That is because semantic examples minimize the naturalness of images, related to distribution $p(X)$, (Eq 2.6) instead of the ℓ_p-norm. The paper [GSG20] does this with the addition of a penalty for variance, mean, and ratio of the color channels.

$$\mathcal{A}(\mathbf{X'}, \mathbf{X}) = -p(\mathbf{X}) + \lambda(\mathcal{L}_f(\mathbf{X}|W, B) - \mathcal{L}_f(\mathbf{X'}|W, B)) \tag{2.6}$$

Poisoning

The basic idea of poisoning is to degrade the overall performance of the model for benign input. That assumes a higher level of influence than evasion attacks. The adversary needs to have the necessary capabilities to manipulate data during training. Still, examples of such attacks are not uncommon [BR18]. The data that may be manipulated are its labels, features, or both. The latter is an issue for federated learning algorithms. As it is hard to know which data is malicious when its data isn't independent identically distributed (non-iid) [Tia+22].

The amount of perturbation for dataset samples is determined with gradient-based methods (e.g. line search) [ZCW21]. Evasion attacks are therefore closely related to poisoning. Poisoning problems, however, have an inner-minimization (Eq. 2.7) that is hard to solve with a direct gradient. Rather, the minimization is solved with a system of Karush Kuhn Tucker conditions [BR18]. Although this method is out of scope for this thesis, it shows that poisoning can be reduced to the evasion case.

$$\begin{aligned} \mathcal{A}(\mathbf{D'}, \mathbf{D}) = \quad & -\mathcal{L}_f(\mathbf{X}\,|W', B') \\ \text{s.t.} \quad & \underset{W', B'}{\text{argmin}} \quad \mathcal{L}_f(\mathbf{D} \cup \mathbf{D'}|W', B') \end{aligned} \tag{2.7}$$

Complete robustness against poisoning attacks is not achievable. This issue can partly be seen as an engineering problem. We should be able to assume a certain level of integrity from the testing data. **Misguided evidence** is an unmediated source of bias for machine learning [Tsa+21].

Model Extraction

Evasion and poisoning attacks have a large potential to disrupt the functionality of neural networks . At least, when some information about its gradient is known. Papernot *et al.* [Pap+17] were the first to mimic target models in a black box context. Their strategy is to substitute the model and learn its parameters based on queries. This process can be achieved by traversing the target model with specially crafted inputs designed to elicit informative responses. The traversal of the target model is determined with regular sampling techniques, evasion methods, or data-free searching (e.g. line search) [ZCW21].

The adversary can use any architecture to train a substitute model, given that its generalization performance is enough. The source model can best be adapted to the input-output relation. As such, it would help to utilize a CNN for image recognition. The final objective of the adversary would be to minimize the Kullback-Leibler (KL) divergence (Sec. 6.1.1) of the target $g(\cdot)$ and source model $f(\cdot)$ (Eq. 2.8).

$$\mathcal{A}(f(\cdot), g(\cdot)) = \sum_{X \in \mathfrak{I}} g(X) \log(\frac{g(X)}{f(X)}) \tag{2.8}$$

The confidence of a substitute model can hint towards the training data [FJR15]. The fact that this information leaks through basic interactions with a model is worrisome for the privacy guarantees of the training set. The transferability also adds the possibility of subsequent attacks of the other two kinds. We find evasion the most interesting, as its gradient-based methods seem to be a universally applicable across all three threats.

3

Contribution

This chapter states the requirements, intuition, and motivation of our contribution. It first discusses the relevant features for CNNs. The chapter then reviews the related work and shows the limitations of these methods. We consider semi-supervised detection methods, which includes image purification and estimation of the manifold. The chapter concludes with the contribution of this thesis.

A generalized perspective of adversarial examples is based on the training **manifold** . Many high-dimensional datasets only use a small fraction of the space, which is this manifold. A common intuition of adversarial perturbation is that it pushes benign samples off the manifold. According to Szegedy *et al.* [Sze+14], this places the sample in a blind spot of neural networks, where the model is extrapolating the classification. When this argument is followed it implies three possible situations [Fei+17], where the adversarial example X' is either near the manifold, far from it, or near the decision boundary.

Our goal is to cover all cases so that the detection is more sound. The recurring theme in this work is the idea of **meeting the defense** [Ald+22]. Stating that harmonizing prevention and detection covers both near and far samples most effectively. The potential of samples that lie close to the manifold can be lowered with robust optimization. Like limiting the Lipschitz constant (Sec. 4.4.3) or regularizing the model. For large perturbations, these methods impact the fitting of the model. But, in this case, these examples introduce certain anomalies towards the training set.

Meanwhile, current methods decide on narrow statistical regularities [HG17]. This collapses when the metric becomes a target for the adversary. For example, principal components can be evaded and variance can be gamed. This may not be the case for an ensemble of several strong but imperfect predictors. The same holds for different Points of Interests (POI) in the neural network. For instance, the raw image can be distorted on a block level by a large margin, so that the flattened image can be near the trained set. To counter this and improve the robustness of detectors, our main contribution of this thesis is a detector that measures irregularity on different POI, whilst being assisted by robustness optimization techniques for small perturbations.

3.1. Point of Interests in CNNs

Typical CNNs exist of two main stages: feature extraction and classification (Fig. 3.1). During feature extraction (shown in blue), the image is feature mapped by kernel convolutions and downsampled with pooling. This will result in multiple planes of a lower resolution. The feature maps are then flattened, for the subsequent Fully-Connected (FC) layers that do the classification. The details of these operations were explained earlier (Sec. 2.1).

From this construction, one can determine at least three POI for our detection. On the raw image, block level distortion may be found with an image quality assessment like BRISQUE [MMB12]. The underlying features of BRISQUE quantify the 'naturalness' of the image, which will be elucidated in the

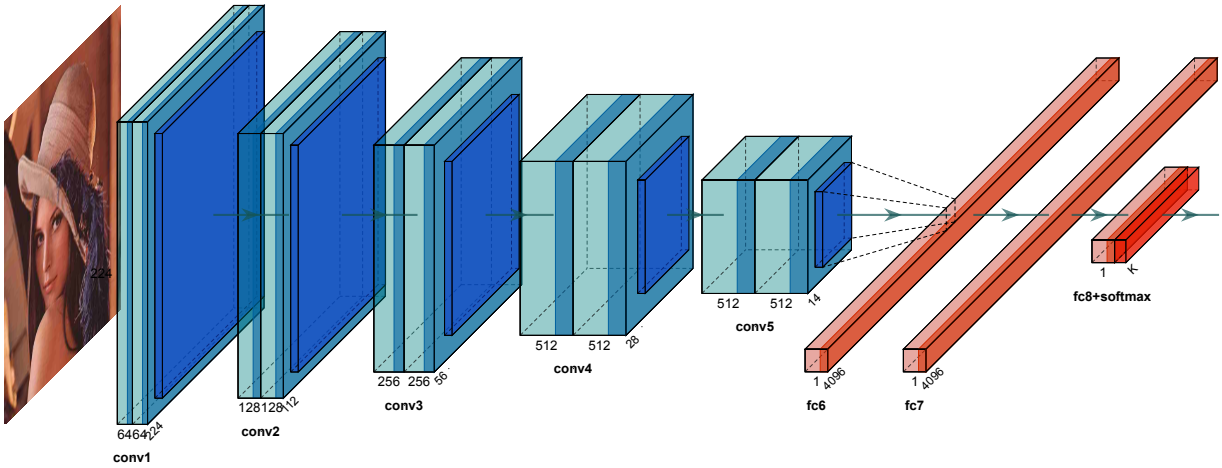


Figure 3.1: Visualization of VGG-13

next chapter (Sec. 4.3.1). The features of the raw image are also called Natural Scene Statistics (NSS) and our first point of interest. Next, the flattened image represents the learned features of the model. The flattened image after convolution excludes small invariances of the raw image and is the direct input of the discriminating network. We will call the utilization of the flattened image Learned Features Analysis (LFA). Lastly, the hidden layers of the model can be analyzed. Goodfellow *et al.* [GSS15] showed that hidden layers can be highly responsive to perturbations, as the activation for some neurons is much higher under adversarial examples. Indeed, one can expect that the adversarial examples are likely to target neurons with higher **gradients** towards the target label when these often optimize on gradient information (Ch. 4).

3.2. Requirements

Most detection techniques easily detect adversarial examples that were generated by the Fast Gradient Sign Method (FGSM) [Ald+22], but this does not mean that the detection is sound. Carlini & Wagner [CW17a] recommend evaluating the models on more powerful attacks. Their suggestions, including this one, lead to the following requirements for our defense technique:

- **R1:** The defense should be robust in a white box or adaptive setting. Security in a grey box setting is not enough, this would replace the problem by one of reverse engineering. Following the famous **Kerchoffs' principle** from the field of cryptography: Security through obscurity is not sustainable.
- **R2:** The first requirement implies that the system should be secure against an universal threat. Meaning that any evasion method generating adversarial perturbation, whether noisy or near-optimal, should be detected.
- **R3:** Its security must scale well with the dimension of the input or output, both in its predictive and computational capabilities. Small datasets, such as MNIST [LeC98], are significantly more distorted under the same amount of ℓ_2 perturbation than others like STL10 [CNL11] and ImageNet [LY15]. As a result, these might need different heuristics to be effective on higher and lower resolutions.
- **R4:** Speed of the network should be maintained and especially at employment. As such, the system's complexity should grow $\mathcal{O}(n)$ with the size of the neural network. The complexity at training time, however, can be seen as a fixed cost and has less of an impact.
- **R5:** The model should remain accurate in classification when any regularization is used. This relates to remedies involving weight decay, L1 or L2 regularization, and robust optimization. These increase the risk of under-fitting.

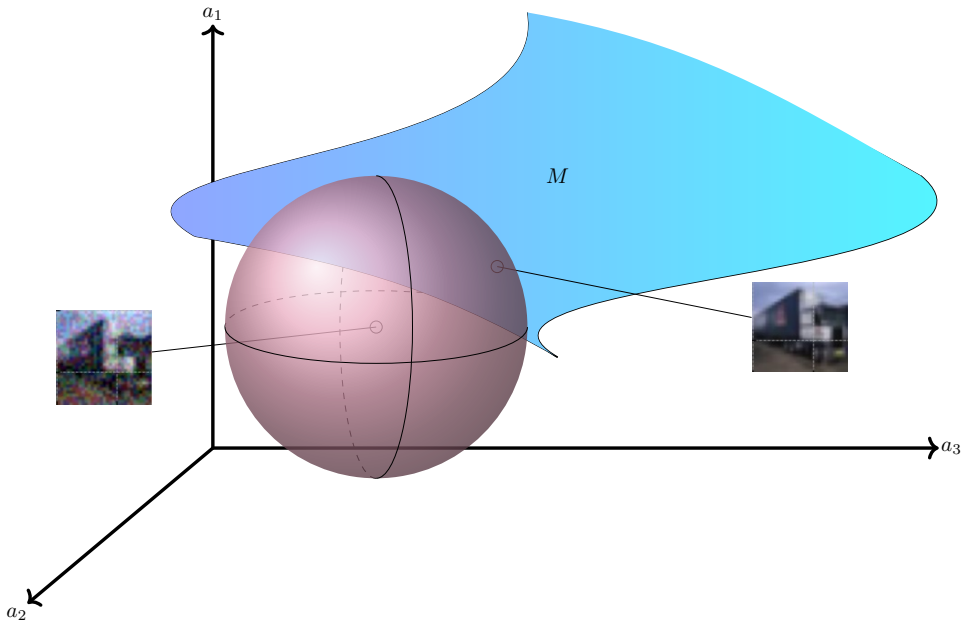


Figure 3.2: Illustration of the submanifold M for $p(X|\hat{y})$ and an ϵ-ball of foreign input.

3.3. Related Work

The literature review is limited in line to the requirements in Sec. 3.2. We are not aware of any work that solicits a detector which is known to be secure in a white box setting. The scope of the related work will therefore only cover work that possibly have some resistance. In particular, it leverages the general characteristics of normal data rather than being narrowly tailored to specific attack patterns, thereby enhancing robustness against a variety of unforeseen threats. These are *semi-supervised detectors with a pronounced accuracy on near-optimal adversarial examples and without gradient obfuscation*. Supervised detectors may outperform under the attack that is used during training; but are not consistent on other incursions [ZH18] and this is validated in our experiments as well (Sec. 6.2.3). To this extent, the following papers were consulted. Divided in two sections for clarity.

3.3.1. Measurement of Intrinsic Properties

Papernot & McDaniel [PM18] improve the interpretability , robustness , and confidence estimates of neural networks with the (Deep) k-Nearest Neighbor (DkNN) algorithm. They leverage the kNN algorithm to calculate the **credibility** metric from the hidden units. Credibility represents the level of support under the training set $\mathfrak{D}$. This is the amount of evidence that is congruent to the new prediction $\hat{y}$. In other words, the metric estimates the conditional probability $p(\hat{y}|\mathfrak{D}, X)$.

Credibility is a characterization of the model's confidence. Other methods to estimate confidence include Bayesian approaches. A simple example is Monte Carlo (MC) dropout [GG16]. Dropout's original use was regularization of neural networks; but was later proposed as a proximate for Bayesian inference of the posterior. To this extent, Gal *et al.* [GG16] illustrated that confidence depends on the log-likelihood of multiple Bayesian approximations at test time. The results of Papernot & McDaniel [PM18] show a low credibility for adversarial examples ($\sim 75\%$+ recovery of Carlini & Wagner attacks); a similarity in the semantics of neighboring adversarial examples; and demonstrate interpretability via analysis of nearest neighbors.

Feinman *et al.* [Fei+17] devise their detector using the assumption that adversarial examples lie off the small manifold of training data. They argue that the identification of low-confidence regions is less effective for data that lie far from the data manifold. Density estimation is used for these situations instead. Kernels on the train points should give an estimate of what the submanifold corresponding to a label $\hat{y}$ is. MC dropout can provide additional information about model confidence not conveyed by

distance metrics. They show a consistent and high ROC-AUC measures across five attack algorithms (AUC of 72%+).

Kernel Density Estimation (KDE) and kNN are both **nonparametric methods**. A fascinating property of these methods is its flexibility. It can be shown that the methods converge to the true density function in its limit to infinite sample sizes. The KDE of Feinman *et al.* [Fei+17] model the conditional probability $p(X|\mathfrak{D}, \hat{y})$. The difference between the two estimators is that kNN fixes density and KDE the volume of the density estimate. KDE is therefore less adaptive than kNN. In regions of high data density, KDE may construct a washed-out structure of training data.

Ma *et al.* [Ma+18] use this argument (i.e. over-smoothing) to show that KDE has some limitations in local adversarial pockets. Measuring the local dimensional structure instead of the kernel density of the data would solve this problem. Dimensionality of the local data is revealed by the growth of the data on each dimension. They use the fact from physics that a m-dimensional ball grows in a polynomial order of m when it is scaled by radius r. Local Intrinsic Dimensionality (LID) estimates the growth of the local volume using this property. One way to do this is with the kNN algorithm (Sec. 4.3.2). Ma *et al.* [Ma+18] show higher ROC-AUC measures for LID (AUC of 82%+).

Lee *et al.* [Lee+18] improved upon LID by estimating the density with the Mahalanobis distance. Their method is specifically designed for detecting OOD samples. The idea is to measure the probability density of test data using a class-conditional Gaussian. They thereby assume the normality of the logits of neural networks. Lee *et al.* [Lee+18] show again higher ROC-AUC measures for Mahalanobis (AUC of 99%+).

Similarly, Zheng & Hong [ZH18] propose I-Defender to estimate the training manifold $p(X|\hat{y})$ accounting latent variables. They assume the manifold to be a distribution of k hidden states. The marginal distributions of **Gaussian Mixture Models** (GMM) (Sec. 5.3.1) can approximate these hidden states of each class, which models $p(X|\hat{y})$. I-Defender uses that estimate to reject new samples. Their results show an adversarial accuracy of 78%+.

A novel algorithm to find the maximum likelihood for GMMs is **Expectation-Maximization** (EM) . The algorithm iterates between the E-step and M-step. The E-step computes the posterior of the model, given the covariance matrices Σ_k and means μ_k. The M-step updates the parameters of the model. A concern that will be discussed shortly is the size of Σ_k. This hyperparameter grows quadratically with the dimension of the input.

Raghuram *et al.* [Rag+21] propose JTLA, which is another generative classifier for detecting adversarial examples. JTLA models and aggregates the class counts of kNN in a Dirichlet distribution. On each layer, this should result in a class-conditional probability of the test sample. The authors propose two techniques to normalize the class-conditional probabilities based on its p-value. The authors show a precision of 23%+ of on adaptive and non-adaptive attacks.

Hendrycks & Gimpel [HG17] show that near-optimal perturbations result in a higher variance of low-rank principal components. They proposed to PCA whitening to detect an abnormal emphasis on lower-ranked principal components compared to clean images, which simply measures the variance of such components. Their results show a perfect classifier on the FGSM attack. Li & Li [LL17] propose another application of PCA that measures extreme neuron activations (Sec. 4.3.4).

3.3.2. Image Purification

The idea of image purification is to move foreign images X' back to the training distribution. That should remove any adversarial noise of the image. Purified images may then be classified by a regular classifier. Song *et al.* [Son+18] propose a denoiser approach for purification. Their method, PixelDefend, seeks the highest likelihood of $\hat{p}(X)$ within the ϵ-ball of X'. Fig. 3.2 illustrates this setting among three activations. The purified image can be determined with optimization techniques like L-BFGS. The authors chose a greedy approach instead, which increases the performance to 3.6 CIFAR10 [KH09] images a second. This setup results in an adversarial accuracy of 81%+ with adversarial training and

Table 3.1: Comparison with prior work. R3 and R5 also depend on the target model.

Detector	Universal (R2)	White Box (R1)	No Singularity	Non-linear	Efficient (R4)
DkNN	✗	✗	✓	✓	✓
KDE & Dropout	✓	✗	✓	✓	✗
LID	✗	✗	✓	✓	✓
Mahalanobis	✗	✗	✓	✓	✓
I-Defender	✓	✗	✗	✓	✗
JTLA	✓	✗	✓	✓	✗
Whitening	✗	✗	✓	✗	✓
Extremal	✗	✗	✓	✗	✓
Feature Squeezing	✗	✗	✓	✓	✓
Pixeldefend	✗	✗	✓	✓	✗
LRD (Ours)	✗	✗	✓	✓	✓
MeetSafe (Ours)	✓	✓	✓	✓	✓

under small perturbations.

Xu, Evans, & Qi [XEQ18] reduced the degrees of freedom with predetermined filters or Feature Squeezers. A diminutive change on features with low variance are discarded due to these filters, which limits the opportunities of adversaries significantly. At least, for near-optimal perturbations. They explore two types of Squeezers: bit-depth, and spatial smoothing (median blur). A detector compares the ℓ_1 distance between the probits $F(X)$ of the squeezed and original image. Their results showed an overall detection rate of 84%+. However, the results on highly perturbed images were notably lower, with a rate of 20%+. Feature squeezing is similar to the work of Liang *et al.* [Lia+18].

3.3.3. Opportunities & Motivation

Tab. 3.1 summarizes the related work based on the discussed requirements. Singularity and non-linearity are of additional interest. Prior work has evaluated multiple linear detection methods [CW17a], of which ones based on PCA. Indeed, analysis of principal components show vast differences between benign and manipulated data [HG17]. Only, the linearity of this behavior seems to make it unfit for the white box setting [CW17a].

A second challenge arises when one scales the target model too wide. Ultimately, treating each (hidden) unit of wide models as a feature challenges the scalability of the detector. For instance, I-Defender [ZH18] faces such issue already for the ResNet-50 [He+16] model. The full covariance matrix Σ of this defense grows $\mathcal{O}(n^2)$ in the input dimension d. I-Defender would thus have around half a million independent variables ($\approx \frac{1000^2+1000}{2}$). As a result, smaller but common datasets like STL-10 [CNL11] and Tiny-ImageNet [LY15] induce singularities in the detector's likelihood function, because the sample covariance matrix is likely to be **positive semidefinite** when there are less training samples than the number of variables.

Besides, some classifiers degrade on wide models because of the curse of dimensionality. A reduction of parameters and detector's complexity might thus be necessary, for example by assuming the matrix Σ_k to be diagonal. Alternatively, one may prune the considered features with feature engineering. To this extent, we indeed show that some units change more than other under perturbation, allowing a heuristic to extract relevant features from only a subset of the units (Appendix B).

Similarly, for the issue of dimensionality, it is at least crucial to maximize the data. The segmentation in the KDE and DkNN method is not preferable. After all, we are not certain of the true class of a given sample. Meaning that inherent misclassifications, that are not caused by any perturbation, will be compared to an ill-suited distribution. The KDE method can thus hypothetically be improved by estimating the density function with all labels. In addition to what was suggested by the authors of the LID heuristic [Ma+18].

A third issue is the coverage of the detectors, this is shown in the table as universal. Most detec-

tors focus their method on a specific perturbation type, making them vulnerable in the white box setting. As an example, Feinman *et al.* [Fei+17] showed that distance based metrics are not effective for small perturbations. Similarly, a squeezer can't filter out substantial changes in the input. Whilst squeezing helps in lowering the relative noise, it will not remove the vulnerability of the classifier. Indeed, their results on major perturbations are not consistent [XEQ18] for this reason. The fourth challenge is the robustness measure and interpretability of neural networks (Sec. 2.2.1). The robustness measure, consisting of the ℓ_p-norm, does currently not resonate with human vision. It is crucial for certified methods to have a metric that does relate better to human perception.

These four shortcomings in current technology are the motivation to propose a model-agnostic detector that measures nonconformity on multiple POIs of a CNN. This detector is an ensemble of different heuristics to reach a higher coverage. We want to test the following hypothesis:

Hypothesis 1:
Detectors fail in white box situation when it is either linear, overfit, or has narrow statistical regularities.

Takeaway 2.
Literature review of the recent advancements in the detection of adversarial examples results in the following conclusions:

- ☞ *Supervised detection, using the possibly manipulated image as feature, fails to be consistent.*
- ☞ *KDE may be over-smooth in regions of high density.*
- ☞ *Feature engineering is necessary on the hidden units*
- ☞ *Current detectors decide on narrow statistical regularities.*

3.4. Contribution

The next chapter will show how the related work fits in the taxonomy of Sec. 2.2.2. The thesis will then turn to the proposed detector, MeetSafe. This method combines the 'perks' from three heuristics in a GMM. Our first intuition was to use Local Outlier Factor (LOF) to measure nonconformity, improving upon the KDE method. That failed and the reason why is later proven in Sec. 6.2. With the lessons learned of this set back, the thesis proposes a new heuristic that outperforms other nonconformity measures. To summarize, this paper contributes:

- **C1:** The thesis discusses four properties of adversarial examples that induce essential measures for detection methods. Each related work is then categorized in the proposed taxonomy (Sec. 2.2.2).
- **C2:** The thesis shows an unexpected limitation of heuristics for local density using a rational difference equation. Which are, among others, LOF and LID (Sec. 6.2.2).
- **C3:** It proposes MeetSafe, a detector that uses three heuristics in a Gaussian Mixture (Chap. 5).
- **C4:** The thesis proposes LRD, a new method on nonconformity estimation. This improves the complexity and robustness of prior methods (Sec. 5.1.2).
- **C5:** The thesis introduces two utility functions that allow LRD and other detectors to scale based on a unit's Z-scores or rate of change under perturbation.
- **C6:** The thesis Empirically validates hypothesis 1 with different models (Resnet-50 and VGG-13). It shows MeetSafe to have an 74%+ accuracy on the white box threat model; and 96%+ accuracy on FGSM perturbations. This is with a True Negative Rate (TNR) around 90%. The thesis will also draw interesting conclusions about PCA, contradicting some prior work when robustness optimization is used. (Chap. 6).

4

Background

Gradients have a crucial role in the process of generating adversarial examples. This chapter will therefore start with a discussion of the gradient field and its use in optimization. This part will comprise essential concepts for gradient-based optimization like the Hessian, Jacobian, and backpropagation. It will then continue with the different methods on adversarial example generation and discuss the anomalies these introduce to the data. The chapter will conclude with a brief review of the research directions in robust optimization. This will give rise to the methods: adversarial training, regularization and certification. Both the theory on any anomalies and robust optimization will prove useful for our detector.

The discussion of Chap. 2 showed universal applicability of gradient-based methods across three types of adversarial attacks. That is perhaps less surprising when one considers its dependence on the loss term $\mathcal{L}_f(\mathbf{X'}|W, B)$. In particular, recall the objective functions of evasion (Eq. 2.5-2.6). The objectives consist of a term measuring human perceptual similarity and another term for the model's confidence. The challenge here is to optimize the model's loss whilst remaining similar. To do this, we often resort to similar procedures as during network training. Maximizing the error function under a certain budget (e.g. ϵ-ball).

The error function is visualized in Fig. 4.1 for two weights of the model. Note that the blue arrows below the error function points in the direction of the greatest increase for the function. Such a vector field is known as the **gradient field** $\nabla\mathcal{L}_f$. A step of ∂W would therefore change the error function by $\partial W^T \cdot \nabla_W \mathcal{L}_f$. The smallest value of $\mathcal{L}_f$ occurs on a stationary point where that gradient $\nabla\mathcal{L}_f$ is of length zero.

Stationary points are on the saddle, minima, and maxima of the function. Finding the optimal or a good feasible solution by solving $\mathcal{L}_f$ directly is rather infeasible. The error function has a broad non-linear dependence on the input and can have, as discussed in Chap. 2, distinct local optima. Stochastic Gradient Descent (SGD) is a more naive approach for network training. It takes a select amount of training samples, a mini-batch, to calculate the new gradient $\nabla_W \mathcal{L}_f$; and then update to update W according to Eq. (4.1) with learning rate η. The algorithm iterates the first step in a randomized fashion.

$$W_{t+1} = W_t - \eta \nabla_W \mathcal{L}_f \tag{4.1}$$

A more applicable optimizer for adversarial generation is the Adam optimizer [KB15; CW17b]. Adam scales a momentum $\hat{m}$ for η up or down according to the amount that a specific parameter has changed. Evaluated against the sum of squared (prior) gradients (Eq. 4.4). The adjustments ensure a balanced convergence among each parameter. For adversarial generation, this promotes diffuse perturbation. The term $\hat{v}$ also allows the algorithm to decay the learning rate η. The update rules for the momentum m, v, and parameters W are as follows:

$$m_{t+1} = \beta_1 m_t + (1 - \beta_1) \cdot \nabla_W \mathcal{L}_f \tag{4.2}$$

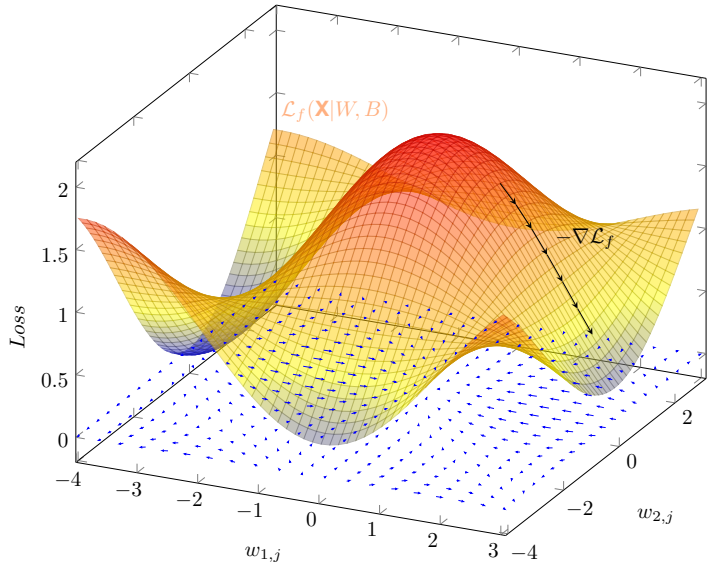


Figure 4.1: The Gradient Descent Algorithm

$$v_{t+1} = \beta_2 v_t + (1-\beta_2)\cdot(\nabla_W \mathcal{L}_f)^2 \tag{4.3}$$

$$W_{t+1} = W_t - \frac{\eta}{\epsilon + \sqrt{\hat{v}_{t+1}}}\hat{m}_{t+1} \tag{4.4}$$

where the hat refers to its normalized counterparts and β_1, β_2 are discount parameters.

Less frequently used, but possibly even stronger, are second-order optimizers. The second derivative of the error function adds additional information about its curvature. Take a saddle point as an example. Saddle points, shown on the diagonal of Fig. 4.1, are by definition suboptimal. With the limited information of $\nabla\mathcal{L}_f$, SGD can get stuck. The gradient of the error function will **vanish** on these points. Sometimes leading to no perturbation at all under attack algorithms that rely on first-order gradients (Sec. 6.2.1).

The second-order gradient of the error function $\nabla\nabla\mathcal{L}_f$ measures the curvature towards each parameter. This matrix is better known as the Hessian matrix H . To demonstrate the meaning of this matrix, we will decompose it in its eigenvectors:

$$H = \sum^{|W|} \lambda_i u_i u_i^T \tag{4.5}$$

where u_i are eigenvectors. The characteristic of H allows distinguishing saddle, maxima, and minima points. A negative eigenvalue λ_i corresponds to a downward curvature in direction u_i. The combination of both positive and negative eigenvalues thus suggests a saddle point at a vanishing first-order gradient. Similarly, a maxima has only negative eigenvalues. Under these situations, H is positive semi-definite. Any local minima would in turn have a positive definite H (Eq. 4.6) . If not, the error function can be improved towards the inconclusive eigenvectors.

$$u_i^T \cdot H \cdot u_i > 0 \tag{4.6}$$

Gradient-based techniques work very well on large dimensions. Each variable provides an extra path on which the state of them can be moved. For neural networks, it is unlikely to end up in poor local minima [RHW86]. Despite this fact, the process is usually restarted with different starting values.

4.1. Exact Gradient Evaluation

The Hessian and gradient can be computed efficiently with the use of partial derivatives respect to components within the network. This approach lowers the naive complexity from $\mathcal{O}(|W|^2)$ to $\mathcal{O}(|W|)$.

It consists of two phases: the forward pass evaluating $\mathcal{L}_f$; and the backward pass that propagates the errors back through the neural network [RHW86]. Because of the backwards calculation, the method is often referred to as **backpropagation** . The process is demonstrated for the first-order gradient below.

The first step in the backward pass is to calculate $\frac{\partial \mathcal{L}_f}{\partial z_i}$ of each logit $z_i \in Z(X)$. Backward steps then utilize the chain rule for partial derivatives to evaluate either the parameter (Eq. 4.8) or previous layer (Eq. 4.9). For Eq. 4.8, we multiply the partial derivative of a logit, or any other activation, with the derivative of the activation function $h'(\cdot)$. That obtains the effect of the activation a_j before $h(\cdot)$ on $\mathcal{L}_f$. Since each activation has a linear contribution with respects to the weight, it is enough to multiply this and the previous unit to get the gradient. Note that this last part equals to one for the bias term. The same idea is repeated in Eq. 4.9, but now it is linearly dependent on the activation $f_j^{(l-1)}$ and is summed over each unit. A sequence of both equations allows evaluating any gradient. The Hessian can, whilst more involved, be calculated with the same rules.

$$\frac{\partial f_j^{(l)}}{\partial a_j} = h'\Big(\sum^{D^{(l-1)}} w_{i,j}^{(l-1)} f_i^{(l-1)} + b_j^{(l-1)}\Big) \tag{4.7}$$

$$\frac{\partial \mathcal{L}_f}{\partial w_{i,j}^{(l-1)}} = \frac{\partial \mathcal{L}_f}{\partial f_j^{(l)}} \frac{\partial f_j^{(l)}}{\partial a_j} f_i^{(l-1)} \tag{4.8}$$

$$\frac{\partial \mathcal{L}_f}{\partial f_j^{(l-1)}} = \sum^{D^{(l)}} \frac{\partial \mathcal{L}_f}{\partial f_i^{(l)}} \frac{\partial f_j^{(l)}}{\partial a_j} w_{i,j} \tag{4.9}$$

Evasion methods often calculate the gradients with backwards propagation, but may start on intermediate components like the logits $Z(X)$ that outputs a vector instead of a single value. That will result in a $|W| \times |Z(X)|$ matrix of partial derivatives for the weights or pixel values. This is the **Jacobian** and includes gradients for each label in this example. One can estimate the distance to the closest decision boundary, knowing the value of $Z(X)$ and the Jacobian (Sec. 4.2.1). When we relate all concepts with each other, then the Hessian can be seen as the Jacobian of the gradient field $\nabla \mathcal{L}_f$.

The chain rule allows a large usability for backpropagation and is certainly not limited by neural networks. **Pytorch** keeps track of the partial derivatives with a computational graph during each forward pass. It includes 'backward' functions for each performed operation within this graph. That way, any model can be computed with gradient-based optimizers. Even though, PGD may not always be the best choice for optimizing parameters. Specifically, the iterative and approximate nature of optimizers may be burdensome. Besides, the complexity of evaluating the Hessian is not practical in most cases. That matrix has $\mathcal{O}(|W|^2)$ ($\mathcal{O}(|W| \cdot |Z(X)|)$ for the Jacobian) components. First-order gradient-based methods are thus ubiquitous; but require multiple iterations and are not guaranteed to converge.

4.2. Generating Adversarial Examples

The gradient of the error function has a crucial role in the process of generating adversarial examples. Almost all techniques that generate near-optimal perturbations use $\nabla_X \mathcal{L}_f$ to determine in which direction the pixel's values should be changed. That is, the gradient with respect to the input data, instead of the model's parameters. After that, the algorithm ensures human imperceptibility with some distance metric as the ℓ_p-norm. An exception to this regularity are algorithms that rely on Constraint- (CP) or Linear Programming (LP). LP and CP can only be effective in convex regions because of the discussed min-max problem of neural networks.

4.2.1. First-Order Gradient-based Attacks

The scope of this thesis is limited to attacks that optimize the ℓ_2 distance in a white box or grey box setting. Attacks, other than the Fast Gradient Sign Method (FGSM), that are an example of other approaches are briefly discussed at the end of this section. We focus on these methods since the ℓ_2 distance gives a rather unrelated measure of perturbation towards the resolution used by the targeted image. Besides, black box evasion methods add more noise to adversarial examples. Our experiments aim to test the worst case under similar amounts of perturbation (Chap. 6).

Algorithm 1 Carlini & Wagner ℓ_2 Algorithm

Require
  $\mathfrak{D}$ Dataset in input space $\mathfrak{I}$
  $F(\cdot)$ Softmax outputs
  $\hat{f}(\cdot)$ Criterion of adversarial examples
Ensure
  $\mathbf{X}'$ Batch of adversarial examples
$X \in \mathfrak{D}$
$\delta \leftarrow \mathbf{0}$
$\mathbf{X}' \leftarrow \emptyset$
$t \leftarrow$ argmax $F(X)$
for epochs **do**
  for iters **do**
    $l \leftarrow c\hat{f}(X) + ||X||_2$
    $\delta \leftarrow$ Adam(δ, l)
    $X \leftarrow X + \delta$
    if argmax $F(X) \neq t$ **then**
      Append $\mathbf{X}'$ with (X, δ)
    end if
  end for
  $c \leftarrow$ Binary_Search$(c, \mathbf{X}')$
end for
return $\mathbf{X}'$

Fast Gradient Sign Method (FGSM)

Goodfellow *et al.* [GSS15] allege that neural networks are too linear around the proximate region of benign input. The activation functions are in fact non-linear, but are designed to behave linearly almost all the time. For instance, a popular activation functions called ReLu exhibits only non-linear behavior near inputs of zero. Whilst, Sigmoid is very linear around small activations.

FGSM uses this fact to perturb the target image X with a ℓ_∞ distance of ϵ as follows (Eq. 4.10). It first applies the sign of the gradient $\nabla\mathcal{L}_f$. The algorithm will then adjust the values of the pixel up or down (uniformly) according to the gradient. Goodfellow *et al.* [GSS15] showed that such ℓ_∞ adjustment around 0.015 is enough to fool unprotected neural networks. This operation is quick because of the linearity assumption, which is the primary reason to still discuss this method even though it optimizes ℓ_∞. Using FGSM, we select our features (Sec. 5.2) and optimize the robustness of the model (Sec. 4.4).

$$X' = X + \epsilon \cdot \mathit{sign}(\nabla_X \mathcal{L}_f) \tag{4.10}$$

Kurakin *et al.* [KGB18] refine this approach with the Basic Iterative Method (BIM), which iterates FGSM according to the projected gradient. Essentially clipping the resulting image to the maximum ℓ_∞ perturbation of ϵ after each iteration.

Carlini & Wagner Attacks

The Carlini & Wagner (C&W) attacks [CW17b] are a follow-up of the first method for adversarial generation called Broyden–Fletcher–Goldfarb–Shanno (BGFS) (Sec. 4.2.2). By utilizing first-order gradients, C&W manages to be faster than the original work and the strongest first-order evasions in our experiments. Both methods optimize an objective function that is solved with an optimizer: Adam [KB15] and BGFS [Fle70]. It directly computes $\mathcal{A}(X', X)$ (Eq. 2.5) with some minor differences. The objective function consists out of the ℓ_p distance and a criterion $\hat{f}(X) : \mathbb{R}^m \to \mathbb{R}$ (Eq. 4.11) guarded by the sensitivity c. The equation demonstrates one of the seven criterions advertised in their paper [CW17b]. Eq. 4.11 measures the ℓ_1 distance towards the highest, untargeted logit $z_i \in Z(X)$.

$$\hat{f}(X) = (\max_{i \neq t}(z_i) - z_t + \kappa)^+ \tag{4.11}$$

Algorithm 2 DeepFool Algorithm

Require
  $\mathfrak{D}$ Dataset in input space $\mathfrak{I}$
  f_i i'th softmax output in $F(X)$
Ensure
  X' An adversarial example
$X \in \mathfrak{D}$
$\delta \leftarrow \mathbf{0}$
$k_0, t \leftarrow$ argmax $F(X)$
while $k_0 \neq t$ **do**
  for $k_i \neq k_0$ **do**
    $w_i \leftarrow \nabla_X f_{k_i} - \nabla_X f_{k_0}$
    $p_i \leftarrow f_{k_i} - f_{k_0}$
  end for
  $l \leftarrow$ argmin $\frac{|p_i|}{||w_i||_2}$
  $\delta \leftarrow \delta + \frac{|p_l|}{||w_l||_2} w_l$ ▷ $||w_l||_2^2$ in their paper, $||w_l||_2$ in their code
end while
return $X + \delta$

The use of logits is related to the fact that logits represent raw, unnormalized predictions. Probits, on the other hand, lost the distance information of the original domain after the contraction towards probability space. Moreover, distillation (Sec. 4.4) aggravates contraction during network training, as it divides the logits with a distillation temperature in the softmax function. In effect, the model exhibits vanishing gradients that are not useful for evasion methods [Pap+16a]. C&W attacks circumvents this defense regardless of how much the softmax layer changes the relative importance.

In addition, Carlini & Wagner [CW17b] propose using high-confidence attacks to test adversarial robustness. Especially because they [CW17b; CW17a; ACW18; CW16; AC18; Tra+20; He+17] showed that most detectors fails to show reasonable robustness under these circumstances. The strength of C&W adversarial examples can be improved in at least three ways. The targeted confidence of the model can be increased with κ. This parameter encourages Adam to find examples X' with a larger distance from the decision boundary.

A second parameter is that may be adjusted is c. This influences the value of such larger distance with respect to the noise δ. A binary search approach on c can be used to find the best balance. The third option is to include the suspicion of employed defences into the function $\hat{f}(X)$ with another sensitivity of c^*.

To ensure the validity of the perturbation δ as an image, certain **Box Constraints** are applied. One box constraint is already discussed, which is the projected gradient. A better approach is to use the tanh-space for clipping (Eq. 4.12). It could smooth out the clipped gradient so that it eliminates the problem of getting stuck in extreme regions. For this reason, the image is projected on the tanh-space leading to an inverse domain of $[-1, 1]$. When we shift that domain accordingly, the values leading to a valid image can be demarcated.

$$X' = \frac{\tanh(X^{clip}) + 1}{2} \tag{4.12}$$

Algorithm 1 shows an overview of the l_2 C&W attack, which is one of the three proposed attacks in their paper [CW17b]. Generally, the method chooses a new starting point at the beginning of each epoch and continues for some iterations. A successful perturbation during the epoch divides the sensitivity parameter c by 2 according to the divide-and-conquer strategy of binary search, as this would suggest that it is possible to find a smaller perturbation.

DeepFool

Deepfool [MFF16] fits a hyperplane on the target model. The hyperplane is an aggregation of the binary, affine classifier $a(X) : \mathbb{R}^m \to \mathbb{R}^2$. The binary case is vulnerable to any perturbation orthogonal to the classifier's boundary. More formally, it is enough to perturb a sample X with the following to reach the boundary of $a(X)$:

$$\frac{a(X)}{||\nabla_X a(X)||_2^2} \cdot \nabla_X a(X) \tag{4.13}$$

where $\nabla_X a(X)$ may be replaced with the weights of the affine operation. A similar function is important in second-order optimization (Sec. 4.2.2) and well known as **Newton's method** , which is an iterative way of finding roots. This method uses Newton's method to find the root $a(X) = 0$. In the next section, we show how to find a stationary point $\nabla_X F(X) = 0$ with second-order optimization.

Now, we use a comparable, but first-order, intuition for any multi-class classifier $A(X) : \mathbb{R}^m \to \mathbb{R}^L$. Geometrically, this classifier constitutes a convex polyhedron, where the classification t also depends on the confidence of other outputs in $A(X)$; since we classify test samples to be the label with the maximum output among $A(X)$. That is, the class t is enclosed in the set $\{a_t | a_t \geq a_i\}$ for any $a_i \in A(X)$. Application of Eq. 4.13 to this relation gives,

$$\frac{|a_t - a_i|}{||\nabla_X a_t - \nabla_X a_i||_2^2} \cdot (\nabla_X a_t - \nabla_X a_i) \tag{4.14}$$

which is again enough to reach the decision boundary with class i. Notice, that this point is *on* the boundary. To make $A(X)$ misclassify the sample, a small overshoot η should be added as a scalar.

Moosavi *et al.* [MFF16] estimate the boundaries of neural networks with the gradients of the probits. This assumes linearity of the boundary. Their method is iterative for this reason, not doing so might lead to insufficient perturbation (Algorithm 2). Though, it will generally halt quickly, after around 1-3 iterations; and whilst it has a higher complexity than the previous methods, it is often much faster than the C&W attacks.

More recently, Croce & Hein [CH20a] made three improvements to DeepFool with their Fast Adaptive Boundary (FAB) Attack. They include (**i**) multiple restarts (**ii**) a dependency on the original sample for each iterative step. The reasoning behind this is that the adversarial example should be biased to the original sample. Projecting the latest iteration in a convex combination of the original sample would minimize the divergence from this starting point. Similarly, FAB adds (**iii**) a Backward step after each iteration towards the original sample. FAB, projected descent, and the Square Attack [CH20b] were used in an ensemble of attacks called Auto Attack, which is currently considered state-of-the-art for the grey box setting.

Semantic Attacks

The downside of using the ℓ_p-norm in RGB space is that it does not fully simulate human perception [GSG20]. A simple example is that invariances like rotations and translations cause large differences in RGB space, but contain identical characteristics for human vision. There is, however, a recent line of work on semantic perturbation that can find perturbation more aligned with human vision [Luo+22; Dua+21; ZLL20; GSG20]. Semantic attacks exploit such invariances to reach extreme perturbations, whilst preserving the structural information of the targeted image. Semantic attacks are thus not restricted to the ℓ_p distance in RGB space, and rather optimize attributes that do simulate human perception. What follows are two examples of semantic attacks.

The shadow Attack [GSG20] is a Projected Gradient Descent (PGD) on three semantic penalties and the average model's cross-entropy loss $\nabla_X \mathcal{L}_f$ over a batch of Gaussian perturbed copies (Eq. 4.15). Penalty $C(\delta)$ restricts the perturbation to each color channel, which prevents extreme color changes. $TV(\delta)$ encourages the perturbation δ to be smooth and uniform across the pixel values by penalizing total variation. Penalty $Dissim(\delta)$ promotes the perturbation to be similar on each color channel. The constants $\lambda_c, \lambda_{tv}, \lambda_s$ are part of the hyperparameters for the shadow attack method.

$$\max_{\delta} \ \nabla_X \mathcal{L}_f - \lambda_c C(\delta) - \lambda_{tv} TV(\delta) - \lambda_s Dissim(\delta) \tag{4.15}$$

PerC [ZLL20] alters C&W's objective function to have the perceptual color distance CIEDE2000 $\Delta E_{00} : \mathbb{R}^{2\times 3} \to \mathbb{R}$ that better aligns with human perception. In particular, ΔE_{00} calculates the weighted ℓ_2 distance in chroma $\Delta C'$, lightness $\Delta L'$, and hue $\Delta H'$ attributes (Eq. 4.16). The weight functions S_L, S_C, S_H, R_T were specifically optimized to human vision. PerC is proposed in two variants. PerC-C&W adds ΔE_{00} into the joint optimization of C&W's objective: $||\Delta E_{00}||_2 + c \cdot \hat{f}(X)$. PerC-AL alternates between the respective left and right term each epoch.

$$\Delta E_{00} = \sqrt{(\frac{\Delta L'}{S_L})^2 + (\frac{\Delta C'}{S_C})^2 + (\frac{\Delta H'}{S_H})^2 + R_T \frac{\Delta C'}{S_C} \frac{\Delta H'}{S_H}} \tag{4.16}$$

Additional Attacks

Additionally, some attacks with a differing threat model will be discussed. These will not be implemented for our experiments, but have strong connections with- or are frequently cited by the other work. In order, we will now discuss a black box, ℓ_0, and an input-agnostic method.

Black box methods have by definition no access to backpropagation. An analogous approach to this algorithm is **numerical differentiation** . It does not utilize any structural information, like backpropagation. That makes this method less accurate and $\mathcal{O}(|X|^2)$; but still interesting for the black box context. Numerical differentiation calculates the gradients with finite (central) differences for each partial derivative. This would thus add a small perturbation ϵ on any pixel value $x_{i,j,k} \in X$ to two forward passes and takes its normalized difference to be the gradient (Eq. 4.17).

$$\frac{\partial \mathcal{L}_f}{\partial x_{i,j,k}} \approx \frac{\mathcal{L}_f(\mathbf{X} + \epsilon) - \mathcal{L}_f(\mathbf{X} - \epsilon)}{2\epsilon} \tag{4.17}$$

Zeroth Order Optimization (ZOO) [Che+17] applies this method on FGSM and C&W. It only considers the confidence scores of multiple queries. They lower the complexity of this numerical differentiation by reducing the dimension of the image with a hierarchical scheme.

Andriushchenko *et al.* [And+20] use random sampling to perturb samples without gradient information. They propose the Square Attack, which iteratively perturbs one squared window under a uniform distribution. Their ℓ_2 attack also 'moves mass' of the intermediate perturbation $X' - X$ from a second squared window. Meaning that the algorithm uses the budget of the second window so that the perturbation is localized and the ℓ_2-norm is further minimized.

Papernot *et al.* [Pap+16b] introduced an attack optimized for ℓ_0 perturbation known as the Jacobian-based Saliency Map Attack (JSMA) . This algorithm builds a saliency map of the input in terms of a pair of pixels (p, q). The saliency map depends on the gradient of the pair towards the target label $\alpha_{p,q}$ and any other label $\beta_{p,q}$. The algorithm then chooses the candidate maximizing Eq. 4.18 for perturbation.

$$(p, q) = (-\alpha_{p,q} \cdot \beta_{p,q}) \cdot (\alpha_{p,q} > 0) \cdot (\beta_{p,q} < 0) \tag{4.18}$$

Moosavi *et al.* [Moo+17] demonstrate a method for Universal Adversarial Perturbation (UAP); perturbations that are input-agnostic capable of causing misclassification across most images in a given dataset. To do this, they accumulate the perturbation of evasion methods over a batch of samples until the perturbation achieves the fooling rate $(1 - \tau)$.

4.2.2. Second-Order Gradient-based Attacks

Second-Order optimization methods leverage the available information with the (estimated) Hessian H_X of the error function. The rationale of these methods is to find a root of the gradient field $\nabla \mathcal{L}_f$ where H_X is positive definite. A simple approach to this problem is to iteratively update an initial point towards the root. Recall that this can be done with Newton's method (Eq. 4.13),

$$X_{t+1} = X_t - H_X^{-1} \cdot \nabla_X \mathcal{L}_f \tag{4.19}$$

notice that this converges faster than gradient descent (Eq. 4.1), as the curvature is now included instead of approached with a learning rate.

Algorithm 3 Broyden–Fletcher–Goldfarb–Shanno Algorithm

Require
  $\mathfrak{D}$ Dataset in input space $\mathfrak{I}$
  B Initial guess of the positive definite Hessian H
  $\mathcal{A}$ Objective to solve
Ensure
  δ Near-Optimal perturbation
$X \in \mathfrak{D}$
$\delta_0 \in \mathbf{0}$
for $n \in \{0..\text{iters}\}$ **do**
  $P \leftarrow -B^{-1} \cdot \nabla_X \mathcal{A}(X, X + \sum^n \delta_i)$
  $\alpha \leftarrow_{\text{[Wol69]}} \text{argmin}\ \mathcal{A}(X, X + \alpha P + \sum^n \delta_i)$ ▷ adhering to the Wolfe conditions
  $\delta_n \leftarrow \alpha P$
  $y \leftarrow \nabla\mathcal{A}(X, X + \sum^n \delta_i) - \nabla\mathcal{A}(X, X + \sum^{n-1} \delta_i)$
  $B \leftarrow_{\text{[Fle70]}} \text{Correction}(\delta_n, y)$ ▷ Update B with the rank two correction
end for
return $\sum^{\text{iters}} \delta_i$

A problem with this approach is that the Hessian H_X is not guaranteed to be positive definite after termination. After all, Newton's method will converge to *any* root of the gradient. Whilst one can know from the Hessian if this is the case, it would still mean that Newton's method may converge to a saddle point or maximum, especially for non-convex functions. It is thus necessary to restrict the Hessian to be positive-definite. Also, computing the hessian is not realistic for large inputs. The inverse operation requires $\mathcal{O}(|X|^3)$ operations, as H is a $|X| \times |X|$ matrix. An approximation of H_X would therefore be more practical.

Broyden, Fletcher, Goldfarb and Shanno independently proposed BGFS [Fle70], which allows for the approximation of a positive definite matrix H^{-1} close to the previous point X_t. They approximate H with an approximation B, which is updated at each step according to the **secant condition** ,

$$B_t \cdot (X_t - X_{t-1}) = \nabla_{X_t}\mathcal{L}_f - \nabla_{X_{t-1}}\mathcal{L}_f \tag{4.20}$$

this condition arises because it is true for any exact hessian. This gives some limited information on what H should be, so we approximate B_t by substituting the left and right-hand side of Eq. 4.20 for B_{t-1} in the vectors u and v of a rank 2 update (Eq. 4.23). Since we also need to impose the secant condition of the updated Hessian, we scale the outer products with α and β respectively.

$$\alpha = (\nabla_{X_t}\mathcal{L}_f - \nabla_{X_{t-1}}\mathcal{L}_f)^T \cdot (X_t - X_{t-1}) \tag{4.21}$$

$$\beta = (X_t - X_{t-1})^T \cdot B_{t-1} \cdot (X_t - X_{t-1}) \tag{4.22}$$

$$B_t = B_{t-1} + \alpha^{-1} u u^T + \beta^{-1} v v^T \tag{4.23}$$

Then a point X_t should be found that also constraints the new matrix B_t to be positive definite. A fast way is to do this is by preserving this property on each update via the **Wolfe Conditions** [Wol69]. These operations lead to Algorithm 3, with some references to the full equations, as these may cause clutter and refrain from the main idea. This and its computational complexity are significant downsides of second-order gradient-based attacks. Even though their convergence benefits from the added curvature approximation.

The work of Szegedy *et al.* [Sze+14] proposed a box constrained BGFS optimization for the adversarial objective (Eq. 2.5). This was the first method to generate adversarial examples. The use of a general optimization methods, like for the BGFS method and C&W attacks, is beneficial. It adds some flexibility in adding new constraints and objectives. For instance, one may add the suspicion of deployed defenses or instead optimize human perceptual similarity.

4.2.3. Convex Optimization

The process of optimizing an objective function, as the C&W and BGFS method do, is called constrained optimization . Here, an objective function is restricted with hard constraints on the variables. These define the feasible region of that function. Till now, all techniques for adversarial example generation optimize a non-linear set of constraints. This section will turn to the linear case. The constraints are thus limited to the form:

$$\sum^{D} a_j x_j \leq b \tag{4.24}$$

for dimension D and constants $a \in \mathbb{R}^D$ and b. Geometrically, this can be seen as a convex polyhedron or polytope for enclosed problems. Each extreme point of that shape is of interest during optimization. Since the linear constraints ensure convexity, it is possible to find a better, feasible solution each step.

A popular method for linear programming is **Simplex**. Simplex systematically finds better extreme points by choosing a new pivot or element to optimize. The method adds this variable to the basis, which represents a new extreme point. This element can only be nonzero on one constraint, the pivot row, to be in the basis. The solution is optimal when none of the elements in the objective function contributes to a better value. More specifically, each element has the coefficient $a_j \leq 0$ given a maximization problem. As an example, consider the basis of the following linear program:

$$\begin{aligned} \text{min} \quad & z = x_1 + 12 s_1 \\ \textbf{s.t.} \quad & 13 x_1 + x_2 + 12 x_3 + s_1 = 5 \\ & x_1 + x_3 + s_2 = 16 \\ & x_1, x_2, x_3 \geq 0 \end{aligned} \tag{4.25}$$

this basis consists of $x_2 = 5$, and $x_3 = 16$ and is optimal.

It can be shown that the worst case complexity of simplex is $\mathcal{O}(2^{|W|})$ in its input. This is the case where each constraint is used in the pivot row, for all possible constraints for a set of parameters $|W|$. Such cases are Klee-Minty cubes [KM72]. Practically, the complexity of the next methods will not meet this instance, as the constraints are independent at each layer of the model or formalize a simple model-agnostic approach.

Convex Restriction

Bastani *et al.* [Bas+16] model the convex relations of neural networks in a linear program. It follows the same logic as the paper of Goodfellow *et al.* [GSS15]. Stating the linearity of nearby inputs. This region is defined by a set of linear activations $A \rightarrow^{h(\cdot)} \{f^{(l)}\}_{l=1}^{N}$ and non-linear units $\{f^{(l)}\}_{l=1}^{N}$ for N layers. For Resnet and VGG the nonlinear function is the ReLu, that is the function $(x)^+$. Networks can be encoded in the constraints:

$$\mathcal{C}_f = (\bigwedge^{|A|} \mathcal{C}_{act}) \wedge (\bigwedge^{|\{f^{(l)}\}_{l=1}^{N}|} \mathcal{C}_{unit}) \wedge \mathcal{C}_{out} \tag{4.26}$$

this gives three components to define. The constraints for units, activations, and the required label. Pooling layers can be encoded similarly to the unit or ReLu function [Bas+16]. The activation's constraints are by definition linear, so no reformulation is necessary here. However, the unit's constraints of the ReLu $z = (a)^+ \in \{f^{(l)}\}_{l=1}^{N}$ handles the max operation in a disjunction:

$$\mathcal{C}_{unit} = (a \leq 0 \wedge z = 0) \vee (a \geq 0 \wedge z = a) \tag{4.27}$$

and the targeted label t is enforced with the $\mathcal{C}_{out}$ constraints:

$$\mathcal{C}_{out} = \bigwedge^{|F(X)|} y_t \geq y_i \tag{4.28}$$

where $F(X)$ are the probits of the neural network. Resnet and VGG can thus be written as these three ($\mathcal{C}_{act}$, $\mathcal{C}_{unit}$, $\mathcal{C}_{out}$) linear sets. The final challenge is to reach convexity, which Simplex requires. Conjunctions form a polyhedron or polytope, as discussed. However, the disjunction is not convex when it

Table 4.1: Performance of each attack on CIFAR10. *N/A* values denotes unreported performances by the original paper.

Evasion	Complexity	Advertised ℓ_2 [CW17b; Bas+16]	Accuracy	Assumes Linearity
FGSM	$\mathcal{O}(\lvert X\rvert + \lvert W\rvert)$	0.83[1]	87.2%	✓
C&W Attacks	$\mathcal{O}(\lvert X\rvert + \lvert W\rvert)$	0.17	100%	✗
DeepFool	$\mathcal{O}(\lvert X\rvert \cdot \lvert F(X)\rvert + \lvert W\rvert)$	0.85	100%	✓
FAB	$\mathcal{O}(\lvert X\rvert \cdot \lvert F(X)\rvert + \lvert W\rvert)$	*N/A*	99%	✓
PerC-AL	$\mathcal{O}(\lvert X\rvert + \lvert W\rvert)$	*N/A*	*N/A*	✗
Shadow Attack	$\mathcal{O}(\lvert X\rvert + \lvert W\rvert)$	*N/A*	*N/A*	✗
L-BFGS	$\mathcal{O}(\lvert X\rvert^2 + \lvert W\rvert)$	*N/A*	*N/A*	✗
Model-agnostic	$\mathcal{O}(\lvert X\rvert)$	*N/A*	*N/A*	✓
LP & CP	$\mathcal{O}(2^{\lvert W\rvert})$	0.61[2]	61.5%	✓

is part of the convex hull. The disjunction of Eq. 4.27 should thus be restricted to be convex.

A simple solution that Bastani *et al.* [Bas+16] propose to eliminate the disjunct that is $False$ for the sampled image X. The resulting constraint models a subset of the original, feasible region. Some solutions are thus lost; but that burden should be minimal when one assumes linearity in the proximate region of X

Model-agnostic Perturbation

Model-agnostic perturbation shifts the benign image towards the intra-class mean of another label [Tra+17]. That is the expected value of a feature mapping $\phi(X)$. For a fixed mapping, the perturbation is most effective when it is orthogonal to the target label t and original label l:

$$\delta_\phi = \frac{\mathbb{E}_t[\phi(X)] - \mathbb{E}_l[\phi(X)]}{2} \tag{4.29}$$

where X is a benign input. That direction must fool accurate classifiers at some point. To put these concepts into perspective, consider the neural network's last basis function $f^{(l-1)}$ to be a feature map to its latent space. So just before the last linear layer [Tra+17]. If the weights of the final layer are aligned with δ_ϕ, i.e. its dot product is not zero, then that direction will certainly fool the model.

However, this assumes that we know $f^{(l-1)}$, whilst we do not. Therefore, one has to test if is if the feature mapping is linear enough (i.e. pseudo-linear) to propagate the perturbation of its input space. Meaning, the orthogonal component of δ_ϕ should remain small for effective perturbation between input-class means (Eq. 4.30). In general, this is true for most classifiers. Tramer *et al.* [Tra+17], showed this for deep neural networks, linear-, and quadratic classifiers. Although, CNNs seem to be lacking this property.

$$f^{(l-1)}(X+\delta) - f^{(l-1)}(X) = \alpha \cdot \delta_{f^{(l-1)}} + \beta \cdot \delta^{\perp}_{f^{(l-1)}} \tag{4.30}$$

The generalization to multi-class classifiers can be made through pairwise differences in means, where one only considers the target t and original label l. Model-agnostic perturbation can then be applied according to the ℓ_2 norm as in Eq. 4.31 with constant ϵ.

$$\delta = \epsilon \cdot (\mathbb{E}_t[X] - \mathbb{E}_l[X]) \tag{4.31}$$

4.2.4. Learning in an Adversarial Environment

Tab. 4.1 shows the performance of each evasion method. The computational complexity of Gradient-based approaches consists of backpropagation $\mathcal{O}(|W|)$, and application of the perturbation $\mathcal{O}(|X|)$. The lowest bound of near-optimal perturbations is thus $\Omega(|X| + |W|)$. Deepfool and BGFS exceed that bound. That is because, these calculate the Jacobian and Hessian respectively. Still, the effect of the increased complexity are misleading. In practice, it shows that Deepfool is faster due to its limited

[1] 0.015 ℓ_∞ on CIFAR10, converted to ℓ_2
[2] 0.011 ℓ_∞ on CIFAR10, converted to ℓ_2

amount of iterations. Even though one iteration takes longer than C&W; but that may change when the number of probits increase. The complexity of Simplex is also deceptive. Resnet and VGG-13 are feed-forward networks, a good pivot rule would be a constraint of the next layer. Feed-forward networks have a uni-directional flow, therefore any layer is independent of subsequent activations.

Not all models are vulnerable to near-optimal perturbations. **Generative models**, that model $p(X, y)$ for input X, can approximate the probability distribution $p(X)$ to find anomalies, which is a property that this thesis will use with the use of GMMs. Other methods are immune to adversarial examples. **Radial Basis Functions** (RBF) lose confidence when X is far from the input space [GSS15]. These models depend on the radial distance to the mean μ_l for each label during training. That makes linear combinations of RBFs hard to fool imperceptibly. The only downside is that the limited complexity of RBF results in a poorer ability to generalize invariances.

Takeaway 3.
Methods for adversarial example generation succeed to generate small perturbations that fool converged models; only it is apparent that their performance is at least dependent on:

- ☞ *The use of logits, contraction of its domain changes the relative importance across the labels.*
- ☞ *The gradient $\nabla \mathcal{L}_f$, which can efficiently be computed with backpropagation, is widely used by evasion attacks. Gradient-based techniques work well on high dimensions and are thus expected to find smaller, effective perturbations on higher resolutions.*
- ☞ *Exact Hessians can add useful curvature information; but is currently not practical to calculate.*
- ☞ *Near-optimal perturbation often hinges on a radial distance to the decision boundary which is small. RBF networks are therefore immune to these perturbations.*

4.3. Properties of Adversarial Examples

The existence of adversarial examples has driven a few explanations for near-optimal perturbations, which are based on some notable anomalies induced by the methods in Sec. 4.2. Some nuance the adversarial strength of such examples (Sec. 4.3.2). Others conjecture the origin of adversarial examples (Sec. 4.3.1). The following sections will present existing theories to perturbation and also some interesting anomalies that can be used for detection.

4.3.1. Szegedy's Low-probability Pockets

Szegedy *et al.* [Sze+14] regarded adversarial examples as blind spots of the neural network, close to the manifold. Normally, the network is expected to assign insignificant probits much further away from input space. For example, under some random noise or invariances. Near-optimal perturbation is an exception to this. These change the model's prediction, even with a small perturbation on training data.

That seems to suggest a dense manifold, where the adversarial examples represent low-probability pockets near almost any benign input. The work of Szegedy *et al.* [Sze+14] measure the possible magnitude of this effect with the **Lipschitz constant**. This constant is a measure of the rate of change of a function. It is defined as the smallest constant $\mathcal{K}$ such that the following inequality holds for all x_1 and x_2 in the domain of the function $g(\cdot)$:

$$||g(x_1) - g(x_2)||_p \leq \mathcal{K}||x_1 - x_2||_p \tag{4.32}$$

where the ℓ_p-distance is used. The Lipschitz constant can give an upper bound for the additive instability of neural networks, and can be expressed independently for each layer. The final constant is the product of the resulting constants. Fig. 4.2 illustrates the segregated calculation of $\mathcal{K}$ for linear layers. The rate of change of a linear transformation depends on the weights W and equal to $||W\vec{r}||_p$ for some unit direction $\vec{r}$. Since the constant $\mathcal{K}$ must be the upper bound of $||W\vec{r}||_p$ one can assume the extreme value $||W||_p$ to be the Lipschitz constant for a linear transformation. Non-linear activations don't change this upper bound. Function $h(\cdot)$ is either contractive (ReLu) or has a reduced slope (Sigmoid), so their rate of change satisfies Eq. 4.32 for an $\mathcal{K} \leq 1$. Convolutional layers require wavelet approximation of

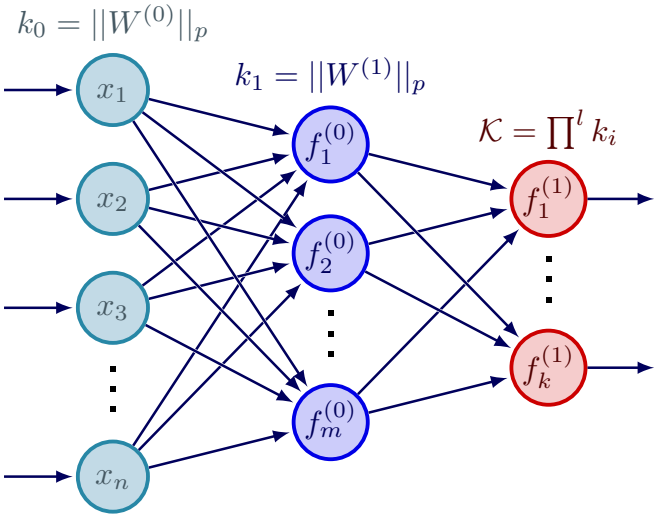


Figure 4.2: The upperbound of the Lipschitz constant

the kernel.

According to Szegedy *et al.* [Sze+14], A high Lipschitz constant enables the manifold to be dense and adversarial examples closer to benign input. Such adversarial directions exploit the Lipschitz continuity to change the labels, even though other features are present. This unintended result is likely not trained upon and nonconform.

This explanation gives some new angles to detecting adversarial examples. The risk of an effective adversarial direction increases when the sample is nonconform and of low-probability (i) or when it provokes large instabilities in the network (ii). For instance, by activating neurons with large weights. We will turn back to the latter metric in Sec. 4.3.4.

Detection methods that measure nonconformity or, in general, the sample's support of the training data are DkNN [PM18], I-Defender [ZH18], KDE [Fei+17], and numerous other works [CSG20; Lee+18; Rag+21; Gro+17; YKR19; LBS19]. Nonconformity was introduced by Papernot *et al.* [PM18] to estimate the lack of support for a label l based on the training data $\mathfrak{D}$ given a set $\Omega_{X,k}$ of k nearest neighbors (Eq. 4.33).

$$\alpha(X, l) = |\{n \mid (n) \in \Omega_{X,k}, \text{ argmax } F(n) \neq l\}| \tag{4.33}$$

However, the definition of nonconformity can be extended to something more closely related to the marginal likelihood of the data (Sec. 5.1.2) and a conditional probability through its empirical p-value [PM18]. Earlier work [SGV99] that Papernot *et al.* cited followed a generalized definition, calling nonconformity supportiveness instead.

An example of a supervised approach is proposed by Akhtar, Monteiro & Falk [AMF18] and measure distortion with no-reference image quality features. To this extent, the features are classified by supervised methods: Random Forests (RF), Support Vector Machines (SVM), and kNN. SVM showed the best performance, it attained at least an 89% accuracy for MNIST and 60% for CIFAR10. A similar work only evaluates no-reference image quality assessment called BRISQUE on an SVM [Khe+20]. In the next chapters, we show that this can be improved by a large margin, whilst being semi-supervised.

4.3.2. Boundary Tilting Perspective

A geometric analysis renders a different perspective towards adversarial examples. When the boundary tilts too much towards the manifold of one label, then the distance of another classification is relatively close [TG16]. In other words, the classifier is said to suffer from adversarial examples iff the mirror image is in the ϵ-ball and off the manifold of the other class. Where the mirror image is the image on

the other side of the boundary, having a similar classification score for another label.

This setting can best be illustrated geographically, see Fig. 4.3. The blue boundary suffers from adversarial examples on the label 'truck', as the sample is close to the boundary. A better boundary would be the red one. In general, one can say that a robust boundary is the **bisecting boundary** $\mathcal{B}$, that lies exactly between the two manifolds. When we assume a linear model then,

$$\frac{f(X_1)}{||W||_2} = -\frac{f(X_2)}{||W||_2} \tag{4.34}$$

is a bisecting boundary, where X_2 is the mirror image of X_1 and W are the weights of a hyperplane $f(\cdot)$. This boundary minimizes the strength of adversarial examples, because the distance to the opposed manifold tends to zero for mirror images.

Now let $\mathcal{C}$ be the boundary of the target model. We can suppose that the boundary $\mathcal{C}$ should be at least a standard deviation σ_l, for label l, away from the centroid to be an accurate classifier. So in the worst case, adversarial examples (mirror images) are on average 2σ from the benign sample. This condition can be reached by tilting on one of the bases. Of course, the severest tilt is the one on components with low variance. Since this allows the boundary to be closer to the manifolds.

Tanay & Griffin [TG16] measure the adversarial strength to be the deviation angle γ_c of $\mathcal{C}$ regarding $\mathcal{B}$. The deviation angle tilts the boundary to the normal of $\mathcal{B}$, called the zenith direction z. Expressing the angle using this component gives a normal of $\mathcal{C}$ (Eq. 4.35). An additional parameter is the ratio r_c. As shown in Fig. 4.3, the tilt might not be centered. If the manifold of the opposite class is down left, then the tilt would be less severe for one class. So some ratio is introduced into the final measure of Tanay & Griffin (Eq. 4.36).

$$c = \cos(\gamma_c) \cdot z + \sin(\gamma_c) \cdot z^{\perp} \tag{4.35}$$

$$s_{\mathcal{C}} = \arctan\left(\frac{\sqrt{\sin^2(\gamma_c) + r_c^2}}{\cos(\gamma_c) \pm r_c}\right) \tag{4.36}$$

The boundary tilting perspective implies some important properties of adversarial examples. Firstly, it shows that the variance of the manifold influences the strength of adversarial examples. The boundary $\mathcal{C}$ can tilt further and mirror images are closer when the variance along the zenith direction is small, without major performance hits of $\mathcal{C}$. Thus, if near-optimal perturbations exist, they must use the components with the smallest variance. Secondly, the existence of near-optimal perturbation conveys a lack of regularization. The boundary $\mathcal{C}$ should be restrained in altering too much.

The first property is of interest for detectors. Small perturbations, such as those induced by DeepFool and C&W, can likely be attributed to their small variance components. The detectors using that measure are feature squeezing [XEQ18], LID [Ma+18], and whitening [HG17] and much more [Tia+21; Hu+19; Lia+18; Son+18]. Feature squeezing removes small variances with so-called squeezers and then measures the change in $F(X)$. The other two extract this information in-place, along the training data, with LID and Principal Component Analysis (PCA). LID [Ma+18] is a metric for the volumetric change within the data. For the statistical setting, this is formulated as the expected number of neighbors. Ma *et al.* [Ma+18] estimate this value to be:

$$-\frac{k}{\sum^{k} \log\left(\frac{r_i(X)}{r_k(X)}\right)} \tag{4.37}$$

where r_k is the k-distance. PCA decomposes the data in terms of principal components. We presume that the reader is familiar with the details of PCA.

4.3.3. Bayesian Uncertainty Perspective

The probits $F(X)$ of a neural network are regularly interpreted as empirical probabilities of the labels, given an input X. It posits some level of confidence of the model. That is partially true, the probits generalize the estimate far outside the training data and are therefore a meaningless measure for epistemic uncertainty [PM18] . That is the confidence, or the absence of confidence, introduced by limited

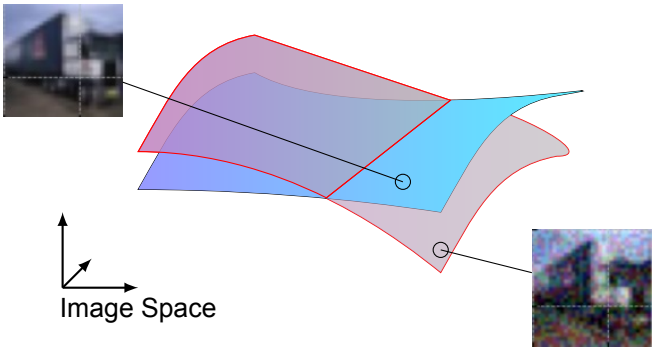


Figure 4.3: The Boundary Tilting Perspective.

training data. Adversarial examples illustrate this misconception very well. These force the output to change, and often with pronounced probits for a wrong class. Supposedly, having a higher confidence than benign samples.

Considering the probit's shortcommings, the instability of neural networks has at least two reasons [PM18]. First, the model is too sensitive when extrapolating unseen data. Second, the model learned shared features across classes and the input includes the ambiguities. As a result, the probits are increased for two or more labels. This thesis did already discuss of how the adversary chooses the target label in the second situation (Sec. 2.2.2). The adversary's specificity is largely based on probits. Deepfool and especially C&W choose the target label utilizing the relative importance according to the logits or probits. The difference in the perturbation's effect between the worst- and best case approach, using the model's confidence, is roughly tenfold [CW17b].

This difference shows the imbalance of local features described by Pang *et al.* [Pan+18]. They argue that predictions are more reliable when that imbalance is minimized. Concretely, this is the case when each non-maximal element in $f_i \in F(X)$ have uniform uncertainty. In theory, the target of choice would be equally far for best, and worst case adversarial examples.

Pang *et al.* [Pan+18] construct non-Maximal Entropy (non-ME) to measure the uniformity (Eq. 4.38). This metric is somewhat related to what the certainty sensitivity penalty [RD18] tries to optimize. With the only difference that non-ME focuses on non-maximal components (i.e. $i \neq l$).

$$H(X) = -\sum_{i \neq l} f_i(X) \log(f_i(X)) \tag{4.38}$$

Interesting about the constriction to local features for labels is its increased interpretability. Other work showed perturbations regulated by certainty sensitivity to be more convincing and reasonable for human vision [PM18; RD18]; the results are also promising for adversarial detection. Optimizing a model's confidence increases perturbations. Besides, it constraints possible perturbations to have a certain amount non-ME, as it would advert a serious anomaly otherwise.

Methods that measure the model's certainty for detection are **Reverse Cross Entropy** (RCE) [Pan+18], dropout [Fei+17], and DkNN [PM18]. Earlier, this thesis described dropout and DkNN (Sec. 3.3). The problem with these two is its performance for large models. Whilst DkNN is much more efficient than a Monte Carlo solution, RCE requires almost no computation. That is because it is a training procedure. RCE trains models only with the regularization method called label smoothing. Pang *et al.* [Pan+18] reverse the target label, encouraging uniformity:

$$Y(X) = \begin{cases} \frac{1}{L-1} & y_i = 0, \\ 0 & \text{otherwise} \end{cases} \tag{4.39}$$

where $Y(X)$ is the target label and L the amount of classes. If the model is then reversed after training by negating the logits, we will obtain a model that returns ordinary predictions; and maximizing Eq. 4.38.

Because of the negation, the models are often referred to as reversed models.

4.3.4. Linear Explanation

The linear explanation of Goodfellow *et al.* [GSS15] shows that solely the high capacity of models is not necessary to produce adversarial examples. Even when a class is well-separated by a linear classifier, a small perturbation is enough to misclassify samples. To see this, consider the adversarial example in Eq. 4.40. The right side adds a major perturbation when the weights W have a high dimension. This effect grows linearly given the dimension. So the average image on ImageNet would have a ℓ_1 perturbation of $\epsilon \cdot 224^2$ when each pixel is changed with ϵ.

$$W^T X' = W^T X + W^T \epsilon \tag{4.40}$$

Goodfellow *et al.* [GSS15] describe this effect as **accidental steganography**. Even though the signal has different features that are similar, the small differences in its amplitude can mislead the model. Adversarial examples occur therefore naturally on higher dimensions.

Looking back to the Lipschitz continuity of Eq. 4.32, one can indeed find that the rate of change for a function $g(\cdot)$ does depend on the change of the input and capacity of the model. High dimensional samples that aim to maximize the Lipschitz constant $\mathcal{K}$ via the gradient have thus much potential. One characteristic that is apparent for this strategy is the excited activations it causes. Li & Li [LL17] came up with an extremal statistics to detect this behavior. They measure the activations on each layer and normalize these by its standard deviation during training.

Takeaway 4.
Near-optimal ℓ_2 perturbations are comprised of a small, worst-case perturbation. The Lipschitz continuity in Eq. 4.32 implies that such examples are highly aligned with the error function's gradient on most dimensions of the input. In practice, this is manifested by the following properties:

- ☞ *The sample is nonconform to the training data and in a so-called blind spot of the model.*
- ☞ *Accurate classifiers can tilt the boundary the most on low variance components. Strong adversarial examples thus differ the most on the smallest principal components.*
- ☞ *Adversarial examples act on regions with a low epistemic certainty.*
- ☞ *Maximizing the Lipschitz continuity causes extremely excited neurons.*

4.4. Robustness Optimization

Defenses of adversarial attacks can be subdivided in two main types [SN20]. The first option is to control the capacity of the neural network during training, also to avoid over-fitting on the training data. Popular regularization techniques can, in theory, already increase the resilience towards strong adversarial examples [TG16]; and this thesis indeed finds that especially strong attacks suffer from regularization techniques (Sec. 6.2).

Robustness optimization is a strategy more specific for adversarial examples. These methods improve the model with derivative-based penalties to smooth out gradients or lower the Lipschitz constant to limit the worst-case potential of adversarial examples. This thesis will briefly discuss these methods in the following sections, as some regulations prove helpful for detecting adversarial examples. It, metaphorically, raises the bar for the adversary, in particular for finding strong perturbations.

The second option is to find statistical outliers on one or more properties described in Sec. 4.3. The idea is to detect **point anomalies**, rather than contextual- or collective ones. Sequential information would be interesting to detect oracle queries for model extraction (e.g. ZOO). Although, it would also have a lot of noise when multiple actors use the model, and similarly rendering it less effective against white box attacks.

An additional option is to obfuscate the gradients at classification. Some uncertainty around perturbations could make the attacks less effective, as the targeted classification cannot be ensured with

small deviations. Whilst this is true, gradient obfuscation often leads to **security by obscurity** and should *not* be used as the main defense [ACW18]. The randomness of stochastic gradients, for example, may even out after multiple iterations. The remainder of this chapter will now focus on robustness optimization. We refer to Silva *et al.* [SN20] for an extended inquiry.

4.4.1. Gradient Obfuscation & Masking

Gradient masking complicates the generation of adversarial examples with manipulated, stochastic, or vanishing gradients. Those gradients are not useful for iterative optimizers. One popular defense uses knowledge distillation [Pap+16a], which originated as a method for imparting knowledge to a second network with limited capacity. It trains two networks on the same dataset, where the shallow network trains on distilled probits $F(X)$ (Eq. 4.41) of the teacher. A hyperparameter called the temperature T provides a way to control the level of uncertainty during the training procedure, as the influence of small variations are reduced. Papernot *et al.* [Pap+16a] argue that the probits will contain more information about similarities between classes, which increases robustness. Later, it has been demonstrated that the defense relies on vanishing gradients [CW17b].

$$F(X) = \frac{e^{\frac{Z(X)}{T}}}{\sum_{z_i \in Z(X)} e^{\frac{z_i}{T}}} \tag{4.41}$$

4.4.2. Regularization

Regularization techniques, described by Silva *et al.* [SN20], apply some penalty term during training. Limiting the rate of change for the model. A common candidate for this term is the gradient, as the instability of neural networks can likely be explained by sensitive gradients (Sec. 4.3.3). The process of adding a gradient to the error function is known as **double backpropagation** [DL92].

Others [YGZ18] optimize the robustness on near-optimal perturbations like DeepFool. Parseval networks [Cis+17] may be closely related with these methods. They also optimize the divergence but then in terms of the Lipschitz constant. This by updating the weights towards Parseval tightness, i.e. converging to a Lipschitz constant of one.

Adversarial training is a stream of research that has been proved successful. Models trained with this technique augment the dataset with adversarial counterparts, leading to an increase in the model's robustness [GSS15]. There is, however, an active discussion on if adversarial training is an absolute defense [SN20; ZH18].

Either way, The work of Goodfellow et al. [GSS15] proposed the use of fast gradient-based attacks during training. Others improved upon this method [Bai+21]. For instance, Tramèr et al. [Tra+18] introduced ensemble adversarial training, which involves adversarial examples from different pretrained models.

4.4.3. Certification

Certified defenses are a promising alternative, as it instead proves the robustness of the model. This comes only at a significant decrement in capacity or computational efficiency. The nonconvexity of the adversarial setting forces an approximation of the actual robustness. For instance, via convex relaxations or interval propagation under a certain ℓ_p distance, even though the ℓ_p distance is not reminiscent of human perception. CLEVER [Wen+18a] does this with cross-lipschitz's extreme value theory. They use sampling techniques to estimate a fixed probability distribution: the Weibull distribution. The Lipschitz continuity can then be calculated based on the estimated parameters. Others propose white box methods instead, that can be optimized to maximally separate feature values [VV22].

4.4.4. Hardness Reduction to Robust Classification

Robust detectors can theoretically be reduced from classifiers up to a factor of two in the distance metric and up to computational constraints [Tra22]. That is, given a detector with an empirical risk of $\mathcal{R}^*_{det}$ for attacks within a ϵ-radii, it has been shown that an inefficient classifier must exist with equal adversarial loss for attacks within a $\frac{\epsilon}{2}$-radii. Moreover, the reverse direction is also true, thereby establishing an

equivalence of detectors and classification systems [Tra22].

A useful construction to demonstrate the reduction utilizes minimum distance decoding that are often used for recovering messages sent over noisy channels. In particular, minimum distance decoding recovers messages by minimizing a certain distance. Generally the Hamming distance. In the case of adversarial examples, we can view its perturbation as the error which we want to recover. Where we do not minimize the Hamming distance, but the verdict of the detector under any ℓ_p-norm.

A popular result of this strategy is the ball-packing argument: if it is possible to detect a perturbation of ϵ perfectly, then it must be possible to recover half of that; since any benign example is at least ϵ away from another one. Choosing the minimal distance would therefore only be just as successful when the adversarial example is closer to the original sample, i.e. smaller than $\frac{\epsilon}{2}$. A direct consequence of the equivalence is the dependence of the detector's performance. We can write:

$$\mathcal{R}^*_{clf} \leq FPR + FNR + \mathcal{R}_{det} \tag{4.42}$$

$$\mathcal{R}_{clf} = \frac{1}{|\mathfrak{D}|} \sum_{\mathbf{X} \in \mathfrak{D}} \mathbb{1}_{\{\text{argmax } F(\mathbf{X}) \neq t\}} \tag{4.43}$$

where FPR and FNR are the detector's false-positive and false-negative rate for a fixed detection threshold, and $\mathcal{R}$ is the defense's standard risk with target label t (Eq. 4.43).

5

MeetSafe Defender

This chapter introduces MeetSafe, a novel detector that combines the Local Reachability Density (LRD) with Feature Squeezing and Principal Component Analysis (PCA). The detector is trained with a Gaussian Mixture Model (GMM) and optimized on reverse models. To decrease the burden of LRD, we propose to use Z-values and BRISQUE. It limits the feature space to the most sensitive ones under perturbation. The chapter concludes with the introduction of a new robustness measure that enjoys the generative property of GMMs.

In the previous chapter, we argued that adversarial examples may exhibit larger differences on low-variance features that are almost vacant within the training data. Also, we have reason to believe that adversarial examples have limited support from the training data. To our knowledge, no prior work has covered both statistical irregularities, and none others achieved useful performance in a whitebox setting.

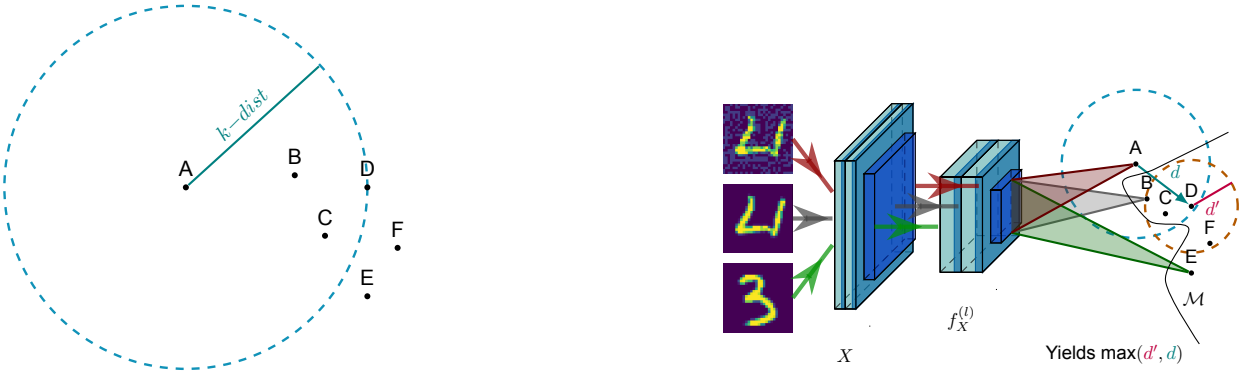
Carlini & Wagner tested tens of defenses that fail their most powerful attack [CW17a]. That is a method that generates near-optimal perturbation. Concerning is the rise of semantic examples. Apparently, the ℓ_p measure is not at all measuring human imperception. Semantic examples optimize conformity with the training data instead.

As mentioned in Chap. 3, there is room for improvement for prior density estimation algorithms like Feinman's [Fei+17]. For instance, other methods like kNN are more adaptive on the kernel; fewer intrinsic properties may be captured with segmentation, as it does not cover all training data. Lastly, large models may need additional feature engineering. To use each hidden unit as a feature would incur infeasible memory requirements for nonparametric methods.

For these reasons, we propose to use LRD with BRISQUE [MMB12] and **Z-scores**, a heuristic that enjoys the flexibility of nonparametric methods with lower memory consumption. LRD estimates density by accounting for the proximity and distribution of neighboring training points. We then combine LRD with variance-based anomaly detection – whitening [HG17] – and feature squeezing [XEQ18], for which an ablation study is given in Sec. 6.2.5. The scores of the three heuristics are learned through Expectation-Maximization (EM) of a GMM we call MeetSafe.

5.1. Towards Conformity

One way to approach point anomaly detection is by fitting a predefined generative model on the training data (Sec. 4.2.4); but this would be less flexible, so our approach uses the distance-based method (i.e. LRD) for large input spaces. Our goal is to find **Hawkins-outliers** [KN98] or global outliers with the use of these local distances.



(a) K-distance visualization

(b) Reachability from node D

Figure 5.1: The LRD metric

5.1.1. Estimating Density with k-Nearest Neighbors

Nonparametric methods model the distribution $p(X)$ with limited assumptions for the underlying distribution. This makes the models flexible. Distribution $p(X)$ can, for instance, be generalized with its volume V and cardinality K of X's proximite region, given enough observations. In contrast to KDE, the kNN method fixes the cardinality and finds the appropriate volume from the data. For a specific sample X, one can then estimate $p(X) : \mathbb{R}^m \to \mathbb{R}$ and $p(y|X) : \mathbb{R}^m \to \mathbb{R}^L$ with:

$$\hat{p}(X) = \frac{K}{|\mathfrak{D}| \cdot V} \tag{5.1}$$

$$\hat{p}(y|X) = \frac{K_y}{K} \tag{5.2}$$

where K_y models the density amongst label y and the dataset $\mathfrak{D}$ is sampled from $p(X)$. The volume is defined by a sphere with as radius the k**-distance**. That volume is the smallest such that the fixed cardinality K is satisfied, as illustrated in Fig. 5.1a. That makes the k-distance of a test sample X sufficient to approximate the likelihood of the data. The estimate would only be shallow with limited information from its neighbors. We may add recursive calls on neighbors to improve the estimate with a higher depth.

By default, each feature has an equal contribution to the k-distance. Notice that BRISQUE features or hidden units maybe affected by unequal standard deviations, as these are not normalized. Some have thus more influence on the k-distance than others. This is not favorable, especially because it was stated earlier that exactly the low variance components are important characteristics of strong adversarial examples. Our method will therefore use the scaled Euclidean distance instead. This normalizes the k-distance with respect to a diagonal covariance matrix Σ. Besides, we will lower the memory burden of the kNN algorithm in Sec. 5.2, after we discuss the details and idea of LRD.

5.1.2. Local Reachability Density (LRD)

The intuition behind LRD comes from the work of Breunig *et al.* [Bre+00]. In that same paper, they propose LOF, a heuristic for finding local outliers. LRD improves upon the k-distance as it smooths out statistical fluctuations in at least two ways. First, the actual distance, called reachability, used to estimate the volume V is capped by the k-distance of the neighbor. Second, The average is taken amongst the k nearest neighbors. The reachability measure of node A and D is demonstrated in Fig. 5.1b, the value depends on the volume of node D and the Euclidean distance of A. Whichever value is bigger equals the reachability from D to A, so that is written as:

$$d(X_1, X_2) = \max\left\{d'_{X_2,k}, \sqrt{(X_1 - X_2)^T \Sigma^{-1} (X_1 - X_2)}\right\} \tag{5.3}$$

where d' is the k-distance. Substituting the reachability from A to its neighbors $\Omega_{A,k}$ in Eq. 5.1, yields its reachability density (Eq. 5.4). Using reachability as a measure to assess the density in the proximate

region of node A has the advantage that only a fixed amount of neighbors needs to be considered.

$$LRD(X) = \frac{|\Omega_{X,k}|}{\sum_{n_i \in \Omega_{X,k}} d(X, n_i)} \tag{5.4}$$

Our method adds one novel improvement called **feature bagging** to both LRD and LOF [LK05]. This enables them to capture higher dimensions. The generalizability of kNN degrades under these circumstances, as the distance between all data points becomes larger and individual features have less of an impact. Bagging is a popular approach to limit this issue. It takes a subset (with random cardinality) of the features for multiple iterations and returns a combined LRD or LOF score. This will therefore also smooth the heuristics, in Sec. 6.2.5 we will show the effect of this smoothing. To make use of the full k-d tree, one takes the union of both branches when a feature is not in the feature bag.

Theoretically, LRD is on average more efficient than the KDE of Feinman *et al.* [Fei+17], whilst it makes no assumption on the Parzen window and estimates the density by averaging its neighbors k-distance (Corollary 5.1.0.1). To improve our detector even further, we show many feature engineering possibilities to lower the memory requirements.

Corollary 5.1.0.1. *When the size of the dataset equals $|\mathfrak{D}| = n$ and a k-distance is used. Then the LRD metric is, on average, computationally bounded by $\mathcal{O}(k \cdot \log(n))$ and KDE [Fei+17] is $\mathcal{O}(n)$.*

Proof. The k-distance can be estimated in $\mathcal{O}(\log(n))$ expected time, with an k-d tree [Ben75, Sec. 4.2]. The reachability distance is the maximum of the k-distance and the Euclidean distance. Meaning, it would depend on the complexity of searching the k-d tree, which is again $\mathcal{O}(\log(n) + c)$ and some constant time c independent of n. LRD then consists of the average of the k reachability distances, which takes $\mathcal{O}(k)$ to calculate. Combining these results gives a complexity of $\mathcal{O}(k \cdot \log(n))$. On the other hand, the KDE of Feinman *et al.* [Fei+17] is $\mathcal{O}(n)$, as it estimates the density for each sample with a predefined kernel. $\mathbb{Q}.\mathbb{E}.\mathbb{D}.$

5.2. Feature Engineering

Before combining the heuristics, we will consider the features of two possible Points Of Interests (POI) for LRD first: the layer after convolution and the pixel values, where we refer to the former as Learned Feature Analysis (LFA). Specifically, we explain how we select its features for both options as without limiting its feature space, LRD would be space inefficient and suffer from sparse data.

On raw pixel values, we advise the use of BRISQUE and Z-scores during feature engineering. BRISQUE fits a Gaussian-like distribution on the raw image while maintaining structural information, which can evaluate the naturalness of an image. Moreover, BRISQUE is 149 times faster than wavelet methods like DIIVINE and performs almost similar on white noise [MMB12].

On hidden layers, we extract a random set of hidden units. Depending on the chosen POI, the Z-scores of FGSM examples X' will be used in order to select the best 10 features of BRISQUE or the best 10 hidden units. We will now explain this feature selection more formally.

5.2.1. Feature Selection

For the hidden units $f^{(l)}$ a pool $\mathcal{P}$ is defined, so that $\mathcal{P} \subseteq_R f^{(l)}$. Members of the pool are chosen randomly at initialization and are preserved during execution. The detector then follows a watching scheme upon $\mathcal{P}$ and its utility (Eq. 5.5-5.6).

The first equation calculates the difference in Z-scores of all features in the pool. It estimates the units that were relevant under perturbation. The second estimates the rate of change of one layer. The first utility is calculated with FGSM after every epoch, this is the fastest evasion method we know, limiting the constraints on scalability or parameter updates. The second utility showed most potential in the final layers, making that our preferred choice (Appendix B). Because of this, we believe the second utility is optional.

$$U_{\mathcal{P}} = \frac{1}{|\mathfrak{D}|} \sum_{X \in \mathfrak{D}} \frac{|\mathcal{P}_X - \mathcal{P}_{X'}|}{\sigma_{\mathcal{P}}} \tag{5.5}$$

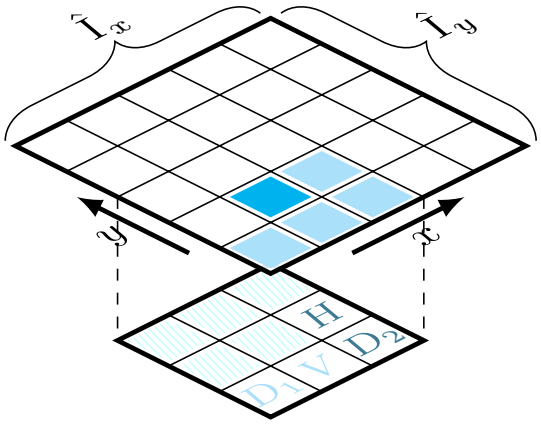


Figure 5.2: The Paired Products of MSCN features.

$$U_{f^{(l)}} = \frac{||\mathbb{E}_{X\sim\mathfrak{D}}[f^{(l)}_{X'}] - \mathbb{E}_{X\sim\mathfrak{D}}[f^{(l)}_{X}]||_2}{||\mathbb{E}_{X\sim\mathfrak{D}}[f^{(l)}_{X}]||_2} \tag{5.6}$$

BRISQUE

Ruderman [Rud94; MMB12] noticed that normalized luminance values are distributed like a Gaussian for natural images. White noise or blur deviates from this Gaussian in its **kurtosis**: heaviness of its tail. BRISQUE [MMB12] is a non-reference method that detect this and other statistical deviations of an image. It utilizes normalized luminance values to fit one Generalized Gaussian Distribution (GGD) and four Asymmetric GGDs (AGGD).

One GGD is fitted on the normalized luminance values and four AGGDs on structural components; the pairwise products of neighboring pixels. The luminance values are better known as Mean Substracted Contrast Normalized (MSCN) coefficients. Meaning that it is calculated with the contrast $\sigma(i,j)$ and mean $\mu(i,j)$ in the spatial domain of the greyscale image $I \in \mathbb{R}^{m\times n}$:

$$\hat{I}(i,j) = \frac{I(i,j) - \mu(i,j)}{\sigma(i,j) + C} \tag{5.7}$$

where the mean and deviation are calculated over a finite kernel with C as some constant. Then by a pairwise product with neighboring pixels, the light blue positions in Fig. 5.2, one obtains the structural components: $H(i,j)$, $V(i,j)$, $D_1(i,j)$, and $D_2(i,j)$. For neighbors on the horizontal, vertical, and diagonal direction respectively. We will now turn to fitting the Gaussians to this data.

The GGD and AGGD are **generalized Gamma distributions**. These may reduce to other density functions such as Gaussians and Laplacian distribtions, depending on a shape parameter α that defines the kurtosis. A value of 1 yields to the Laplacian and 2 to the Gaussian. In case of GGD we aim to estimate the variance σ^2, and shape α of the distribution, the mean is assumed to be zero [MMB12]. To do this, we apply the **Maximum Likelihood Estimation** (MLE) under the observed $\hat{I}(i,j)$ of one sample and calculate the shape α by adopting the generalized Gaussian ratio function (Eq. 5.8) [SL95]. The latter is a first-order moment match of the shape using a lookup table for α, which saves some computation because the gamma function $\Gamma(\cdot)$ is not smooth for the domain of real numbers $\mathbb{R}$. The process results in two features: (α, σ^2).

$$r(\alpha) = \frac{\sigma^2}{(\mathbb{E}_{|X|}[X])^2} = \frac{\Gamma(\alpha^{-1})\cdot\Gamma(3\alpha^{-1})}{(\Gamma(2\alpha^{-1}))^2} \tag{5.8}$$

AGGD divides GGD in a negative part and positive part and fits the exact same distribution on them. AGGD consists of one shape parameter α for two distributions with variance σ_l^2 and σ_r^2. Note that a

difference in the variance changes the skewness of AGGD, it makes AGGD asymmetric. Therefore, Mittal *et al.* [MMB12] measured the skew η, even though it is not one of the model's parameters. The moment matching is rather similar for AGGD. One first estimates the variances with MLE and the shape with a similar ratio function [LSB09]. This results in four features: (η, α, σ_l^2, σ_r^2).

5.3. How to Meet in the Middle

We combine LRD, using hidden units as POI, with variance-based anomaly detection – whitening [HG17] – and feature squeezing [XEQ18]. The scores of the three heuristics are learned through Expectation-Maximization (EM) of a GMM we call MeetSafe.

To detect a sample, we first select the best features based on $U_{\mathcal{P}}$ for LRD and the eigenvectors of the training data. Then we evaluate the three heuristics (denoted as $\mathcal{H}_X$). Assuming normality, a three-dimensional GMM fitted on benign data can classify the sample as malicious when it exceeds the 90'th percentile of

$$-\log \hat{p}(X) = -\log \left\{ \sum^{K} \pi_i \cdot \mathcal{N}(\mathcal{H}_X | \mu_i, \Sigma_i) \right\} \tag{5.9}$$

where π_i is the prior of one latent feature, $\mathcal{N}(\cdot)$ a multivariate Gaussian with variance Σ and mean μ, and $\mathcal{H}_X \in \mathbb{R}^3$ a set with the suspicion of the estimators. The following sections describe GMMs in more detail. We also add a novel idea to measure robustness with MeetSafe. The pseudocode of our method can be found in Appendix D.

5.3.1. Gaussian Mixture Models

An GMM is a linear combination of Gaussians. It will provide some essential flexibility of the density model, it is **multi-modal**. GMMs can be written as a joint distribution of the latent features $\xi_i \in \xi$ and observations X. Each latent feature is assumed to be normally distributed. Something that is favorable for its robustness. Our detector needs such combination because the whole dataset depends on latent components that are not universal. For instance, it needs to model multiple classes with different local features. One Gaussian is said may be responsible for one latent component. So we can say that one Gaussian in the linear combination of Eq. 5.9 models $p(X|\xi_i)$ and π_i is its prior $p(\xi_i)$. These definitions give the opportunity to write the responsibility of ξ_j with use of **Bayes' rule**,

$$\hat{p}(\xi_j | X) = \frac{\pi_j \mathcal{N}(\mathcal{H}_X | \mu_j, \Sigma_j)}{\sum^{K} \pi_i \mathcal{N}(\mathcal{H}_X | \mu_i, \Sigma_i)} \tag{5.10}$$

where the notation is used of Eq. 5.9. This formula will be important during training. Here, $p(\xi_j|X)$ is optimized iteratively until convergence using a powerful algorithm called EM.

Expectation-Maximization

The EM algorithm is a powerful method to optimize the parameters of a GMM. It is a two-step algorithm that iteratively optimizes the parameters of the GMM. The first step is the E-step, where the responsibilities of the latent features are calculated for some fixed parameters Σ_j, μ_j, and π_j. That gives the current estimate of $p(\xi_j|X)$, the probability that the sample X is caused by the latent component ξ_j. The M-step updates the parameters by averaging out the responsibility of each Gaussian as shown in Eq. 5.11-5.13. Where $\gamma(\xi)$ is defined to be the sum of responsibilities $\sum^{K} \hat{p}(\xi_i|\mathcal{H}_X)$ and $\mathcal{H}_{\mathfrak{D}}$ is the family of sets $\{\mathcal{H}_X | X \in \mathfrak{D}\}$.

$$\mu_j = \frac{1}{\gamma(\xi)} \sum_{\mathcal{H}_X \in \mathcal{H}_{\mathfrak{D}}} \hat{p}(\xi_j | \mathcal{H}_X) \cdot \mathcal{H}_X \tag{5.11}$$

$$\Sigma_j = \frac{1}{\gamma(\xi)} \sum_{\mathcal{H}_X \in \mathcal{H}_{\mathfrak{D}}} \hat{p}(\xi_j | \mathcal{H}_X) \cdot (\mathcal{H}_X - \mu_j)(\mathcal{H}_X - \mu_j)^T \tag{5.12}$$

$$\pi_j = \frac{\gamma(\xi)}{|\mathcal{H}_{\mathfrak{D}}|} \tag{5.13}$$

5.3.2. Measuring Robustness

If one accepts the current explanations for adversarial examples (Sec. 4.3), then we can assume that an adversarial example X' does not appear naturally within the training data. The samples fall off the manifold after perturbation, either far or close to the manifold [Fei+17]. Its marginal likelihood is therefore (near) zero. A major advantage of generative models is that it allows the distribution $p(X)$ to be determined.

Following this argument, we may measure the severity of an evasion on the distance to the manifold. Of course, the manifold is not closed form, as some benign noise may deduce to an adversarial perturbation. In any case, it brings up two hypotheses. the sample is on the manifold (H_0) or the sample is off the manifold (H_1). If there is enough support from the training data one can assume H_0 to be tenable.

The generative model gives an estimate of that input space. The hypothesis H_0 can therefore be accepted with a **t-test** below some critical value τ. We will conjecture the robustness of a model to be the sum of $(\tau - p(X))^+$ for such conspicuous samples:

$$\text{argmax}_\delta \frac{1}{|\mathfrak{D}|} \sum_{X \in \mathfrak{D}} (\tau - \hat{p}(X + \delta))^+ \tag{5.14}$$

where δ is the perturbation of an adversarial example. This notation can give better estimates for different datasets. For instance, the MNIST dataset has a rather small input space and the contrast of the samples is large. In essence, the numbers are white with a black background. Samples that do not adhere to the strong contrast are by default unlikely to occur naturally. The estimate $\hat{p}(X + \delta)$ will be much lower for most samples, and so models may reach a higher robustness. This is not necessarily the case for the l_p-based metric. Further experiments may prove this intuition. For this thesis, however, it will be out of scope.

6 Experiments

This chapter reports the results of the experiments. The chapter starts with a description of the methodology used to evaluate the performance of the MeetSafe. It clarifies the choices made for the experiments and the hyperparameters used. The chapter then presents the results of the experiments, including the performance of the model, the sensitivity of the hidden layers, and the performance of the prior methods.

We evaluate MeetSafe and LRD against several evasion methods including FGSM, DeepFool, and C&W. The experiments will demonstrate white- and/or grey-box performance for four datasets: Tiny-ImageNet [LY15], CIFAR-10 [KH09], MNIST [LeC98], and STL-10 [CNL11]. Only for Tiny-ImageNet, we resized the samples to be in $\mathbb{R}^{3\times64\times64}$. The attacks are restricted to an ℓ_2 distance to ensure a fair comparison across datasets and ℓ_∞-based methods. For instance, an ℓ_∞ distance permits more noise for higher resolution images. Consequently, we use the ϵ parameter given by Eq. 6.1, so that the maximum allowed perturbation of FGSM equals that of ℓ_2 methods (δ_{max}); where the image X is given by a $\mathbb{R}^m$ flattened matrix.

$$\epsilon = \sqrt{\frac{||\delta_{\mathsf{max}}||_2^2}{m}} \tag{6.1}$$

Each attack, defense, and target model is re-implemented in Pytorch. Herewith, we evaluated 14 models based on ResNet-50 [He+16] and VGG-13 [SZ15] as shown in Tab. 6.1. Ten of them are trained with robust optimization techniques, utilizing gradient smoothing (RCE) [Pan+18] or adversarial training with FGSM (ℓ_2-radii of 5) (AL) [GSS15]. The convergence graphs of the models are shown in Appendix A.

The experiments also include some related methods that will be compared to MeetSafe and LRD. The baseline for MeetSafe (MS) are KDE with predictive uncertainty (KDE+BU) [Fei+17], I-Defender (I-Def) [ZH18], and the Mahalanobis measure (MAH) [Lee+18]. Additional work that we tested are LID [Ma+18], Whitening (PCA) [HG17], Feature Squeezing (FSQ) [XEQ18], extremal measure (EXM) [LL17], and an earlier application of BRISQUE (SVM) [AMF18]. The method SVM is rather equivalent to the work of kherchouche *et al.* [Khe+20].

6.1. Methodology

For the experiments, we trained the models for 150 epochs on predetermined training sets. During training, a batch size was used of 256, learning rate of 0.01 with momentum 0.9 under a cosine annealing schedule, and $1\mathrm{e}{-4}$ weight decay. The model is also adapted to exhibit required invariances. All training samples are normalized on each channel, randomly flipped horizontally, and cropped within a padding of 4. Adversarially learned models were additionally trained half-on-half on benign and perturbed data.

Table 6.1: The architectures of our models for CIFAR-10. The output sizes and operations of conv_1 may differ on other datasets, as this layer uses an adaptive convolution.

	VGG-13		ResNet-50	
	Output Size	Operations	Output Size	Operations
Conv_1	16×16	$[3 \times 3, 32] \times 2$	33×33	$[2 \times 2, 64]$
Conv_2	8×8	$[3 \times 3, 128] \times 2$	33×33	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
Conv_3	4×4	$[3 \times 3, 256] \times 2$	17×17	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
Conv_4	2×2	$[3 \times 3, 512] \times 2$	9×9	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$
Conv_5	2×2	$[3 \times 3, 512] \times 2$	5×5	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
FC_1	1×1	512-d	1×1	2048-d
Total Parameters	9.4 million		23.5 million	

The detectors were then tested on unseen test images and organized according to the following attack types. First, we test the defense and model on low strength perturbations. That includes FGSM, as it does not directly optimize the ℓ_p distance. Then, we test the defenses on smaller perturbations with DeepFool and C&W. One technique (RCE or AL) is chosen to be tested on white-box attacks.

White-box attacks can be generated by adding the detector's likelihood function (Eq. 5.9) to C&W's objective [CW17a]. In essence, this optimizes a multi-objective gradient with Adam that considers both the gradient of the detector's internals and the confidence of the target model:

$$||X||_p + c \cdot \hat{f}(X) + c' \cdot (-\tau^{-1} \log \hat{p}(X) - 1 + \kappa)^+ \tag{6.2}$$

where τ is a given threshold and c^* a constant that controls the sensitivity towards the detector's gradient, optimized with binary search. The sample is updated with perturbation δ when its aggregate $X + \delta$ fools successfully. For our experiments, we limit the perturbation to a ℓ_2 distance of 5.

White-box attacks follow an all-or-nothing strategy: the batch with adversarial examples are either clean or fully successful. For this reason, we assume that the detector is successful if either the white-box perturbation is detected or classified by the target model. Its true positives are thus in the set $\{X \mid \text{argmax } F(X) \neq k \ \vee \ -\log \hat{p}(X) \leq \tau\}$ for some true class k and threshold τ.

The performance of the defenses is measured using its overall detection accuracy of one test run in both adversarial and benign situations, where the detector classifies at a TNR of 90%+. The adversarial setting may include samples without perturbation when the target model already misclassifies the clean sample. We therefore have an optimal detection accuracy of $1 - \frac{\mathcal{R}_f}{2}$ with standard empirical risk $\mathcal{R}_f$ of the target model.

During test runs we reduced the batch size to 128 (64 for white-box); other hyperparameters, used for the attacks and defenses, were as follows. The magnitude ϵ of FGSM is deduced from Eq. 6.1, DeepFool had an overshoot of 0.02, and C&W executed 5 steps with 500 iterations (10 and 1000 for white-box) under a 0.05 confidence κ. For all defenses, we applied the same feature selection. That took the best 10 in a pool of at most 500 features, given $U_{\mathcal{P}}$. In Appendix B, we demonstrate the added value of such feature selection and give more details about the hyperparameters used. The experiments were conducted on AMD Ryzen 7 7700X and Nvidia RTX 4070 Ti.

The hyperparameters' selection of our detector will be briefly discussed in Sec. 6.2.2. We concluded that an k of 8 for LRD and a GMM with 8 components performed best overall. The related work used its default or recommended parameters by the original work; but there is one exception. The original work

Table 6.2: Accuracies and proportion of stationary points for FGSM of the trained target models on the CIFAR-10, Tiny-ImageNet, STL-10, and MNIST testing set respectively. VGG-13 is only trained on CIFAR-10. More details in-text.

Model	Learn. Rate	Robust Optim.	Evasions ($\ell_2 \leq 5$) $\rightarrow$ Stat. Points (%) $\downarrow$	Model's Top-1 Accuracy $\Uparrow$			
				Benign	**FGSM** Eq. (6.1)	**C&W** $\kappa = 0.05$	**DeepFool** $\eta = 0.02$
ResNet-50	0.01	✗	50.5, 0.0	0.932, 0.581	0.619, 0.013	0.000, 0.000	0.061, 0.188
			13.7, 12.8	0.711, 0.992	0.111, 0.307	0.008, 0.000	0.214, 0.130
ResNet-50	0.01	AL	37.9, 0.0	0.916, 0.559	0.640, 0.176	0.000, 0.003	0.086, 0.149
			2.6, 8.1	0.612, 0.991	0.229, 0.2704	0.200, 0.000	0.221, 0.385
ResNet-50	0.01	RCE	0.0, 0.0	0.885, 0.030	0.485, 0.003	0.000, 0.000	0.106, 0.015
			0.0, 0.0	0.614, 0.990	0.081, 0.085	0.000, 0.000	0.147, 0.228
VGG-13	0.01	✗	22.4	0.928	0.294	0.000	0.096
VGG-13	0.01	AL	2.9	0.885	0.578	0.000	0.158
VGG-13	0.01	RCE	0.0	0.883	0.329	0.000	0.060

of the Akhtar *et al.* [AMF18] applies an SVM on multiple NSS than only BRISQUE. Our experiments only include the features of BRISQUE as we found that this performed similarly, adds less complexity, and is also utilized by LRD.

6.1.1. Kullback-Leibler Divergence

The Kullback-Leibler divergence (KL) [KL51] is a measure of how one probability distribution diverges from a second probability distribution. It is a non-symmetric measure, that is, the KL divergence from g to f is not the same as the KL divergence from f to g. The KL divergence of two distributions g and f is defined as:

$$KL(g||f) = \sum_{X \in \mathfrak{D}} g(X) \log(\frac{g(X)}{f(X)}) \tag{6.3}$$

which will be used to measure the entropy given a mean μ and deviation σ of a feature. We observed that this and the size of the dataset will influence the performance of LID and LOF dramatically. When the density of a distribution gradually resembles a Gaussian and thus the entropy maximized, then Local anomalies are less present for perturbations. That is why we measure Hawkins-outliers with LRD. This is discussed further in Sec. 6.2.2.

6.1.2. Silverman's Rule

A crucial free parameter of a KDE is its bandwidth h. Its value influences the smoothness of the density estimate. The distribution becomes spiky for smaller values of h, which has a poor generalizing ability. An over-smoothed estimate, however, may fail to capture some essential properties of the source distribution.

The bandwidth h of the KDE is determined by Silverman's rule in our experiments. This rule is based on the standard deviation σ of the dataset, the amount of samples n, and dimension d. Silverman [Sil18] showed that the mean integrated shared error is minimized for $1.06\sigma n^{-0.2}$, assuming that the underlying distribution is a uni-modal and uni-variate Gaussian. A generalization for more dimensions is given in Eq. 6.4. This gives us a more refined estimate of h than the fixed h of Feinman *et al.* [Fei+17].

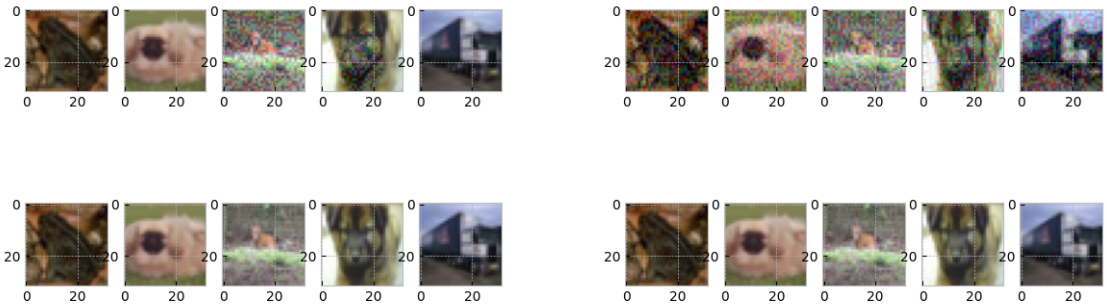
$$h = \sigma \cdot (0.25d + 0.5)^{-(d+4)^{-1}} \cdot n^{-(d+4)^{-1}} \tag{6.4}$$

6.2. Results

The following section will primarily discuss the performances on CIFAR-10. The results on the other datasets and models led to the same conclusions for MeetSafe. In fact, we also find decent accuracies under advanced semantic methods, like the shadow attack [GSG20] and PerC [ZLL20] (Appendix B-C).

6.2.1. Model Performance

The accuracy of the models is shown in Tab. 6.2. The CIFAR-10 ResNet-50 model is able to reach an average cross-entropy of 0.0249 and a test accuracy of 93.2%. That is higher than the results of the ResNet paper [He+16, Tab. 6]. The cross-entropy for robust optimization techniques is notably higher, this increased to 0.47 for adversarial training and 431.1 for RCE. A higher value for RCE was expected,



(a) FGSM on vanilla Resnet-50

(b) FGSM on the reversed Resnet-50

Figure 6.1: Examples of vanishing gradients

as almost each class now adds to the error instead of only the true class. We also observe that the fit and convergence changes dramatically when the amount of classes is increased. Take ImageNet which has 200 classes, where the others have 10, a RCE model trained on ImageNet does only reach an accuracy of 3%. When we test the STL-10 subset, RCE does not show this behavior.

Furthermore, the convergence graphs in Appendix A shows interesting characteristics that seem homogeneous for adversarial training and RCE. Adversarial trained models diverge after convergence on the benign images and especially for larger inputs. As well, the RCE models converge slower than the other techniques. In fact, the Tiny-ImageNet did not fully converge after 150 epochs, whilst the adversarial learned model did.

Robust optimization shows a descent prevention of FGSM for all models and no meaningful prevention against C&W attacks. The high adversarial accuracy of the plain model is somewhat unexpected. However, this has a clear reason. The confidence of this model is higher, which causes vanishing gradients. The results of DeepFool show that the optimized models are often more robust than the plain ones under small perturbations.

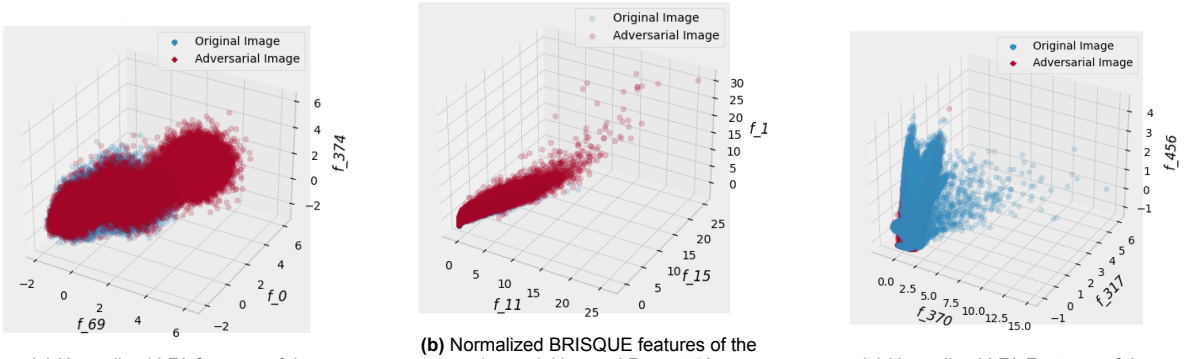
Vanishing Gradients

The effect of vanishing gradients is visible in Fig. 6.1. A look at the confidence score of the plain ResNet-50 would show a unit vector towards the correct class for some images (Tab. 6.2), which means that its gradient is zero. By inspection of the figure, it presents no perturbation for the two images on the left; and indeed the ℓ_2 distance is zero. This is expected, as these images may sit on a stationary point for the current parameters. This is a major drawback of FGSM, but not present for DeepFool and C&W which use gradients of different loss function.

It does give a sense of robustness for the plain model, whilst it is probably not. As such, small noise may push the image off the stationary point, causing an effective FGSM perturbation. Besides, the optimized models predict with a decreased confidence. This is visible on the second figure. More noise is visible here and sometimes even more spread as shown in the two images on the right.

Sensitivity of Hidden Layers

The utilities discussed in Sec. 5.2.1 grow more or less each layer for the CIFAR-10 ResNet. The RCE and plain model exhibit the largest normalized ℓ_2 distance at the fourth bottleneck and the smallest distance at the raw input. Noteworthy is that the utility $U_{f^{(l)}}$ at the first and last bottleneck differ significantly with RCE; as for 5 samples, the 95% t-Confidence Interval (CI) is 0.29 ± 0.003 and 0.51 ± 0.01 respectively.



(a) Normalized LFA features of the adversarial learned Resnet-50

(b) Normalized BRISQUE features of the adversarial learned Resnet-50 (Logarithmic)

(c) Normalized LFA Features of the reversed Resnet-50

Figure 6.2: The distribution of LFA and MSCN features

This supports the unfolding intuition of Bengio *et al.* [Ben+13]. They argue that the deeper layers provide more linear and 'unwrapped' manifolds to work with. Although, we find the behavior of adversarial learned models to be different. There, the first layers seem to be the most sensitive, with the first bottleneck (0.66 ± 0.14) having a higher utility than the fourth (0.48 ± 0.13). That may be because similar perturbations are part of the training data, which gets unfolded by the neural network. Specifically, FGSM samples form points of higher density compared to the general training procedure and assuming the disentanglement of deeper layers [Ben+13, Hypothesis H3(b)], we expect less variance. In any case, detection methods may thus be fine-tuned by utilizing different layers.

Takeaway 5.
The evaluation of the various models yields the following insights:

- ☞ *RCE is sensitive to the amount of labels, as it regulates all labels that are not the target label.*
- ☞ *Robustness optimization is often effective against stronger perturbations.*
- ☞ *A major drawback of FGSM are vanishing gradients*
- ☞ *The last layer is often the best choice to detect perturbation, except when using adversarial training.*

6.2.2. Why LOF and LID fail

Fig. 6.2 shows three scatter plots of LFA and BRISQUE features. The axes are the features that have the highest Z-score for FGSM examples. The chosen features on the hidden layer are rather random each epoch; but this is not true for BRISQUE. Depending on the dataset, the best features are either shape or variance parameters (Appendix B.1.1). For CIFAR10 the best three are: 1, 15, and 11 of the generalized Gamma functions.

All three plots show dissimilarities between adversarial examples. The most left figure is more spread out for adversarial examples, it has a large cluster on the large positive values. The plot in the middle shows similar results, the adversarial examples have sometimes extreme variances; notice that this plot is logarithmic. The red outliers in this plot are further on the original scale.

The right is far less spread, that plot is very dense for adversarial samples. That is expected when we consider the added constraints of RCE. This behavior was actually the purpose of this technique [Pan+18]. It enforces non-maximal activations to be uniform and thus harder to manipulate. The results will show that distance-based metrics perform worse on such features (Appendix B.1).

The Issue of Local Outlier Detection

Local outlier detection, like LOF and LID, might fail on some features of BRISQUE as well. These methods suffer under the gradual change in density of these features. When one measures the KL divergence between the extracted features and a Gaussian, where the Gaussian maximizes the entropy

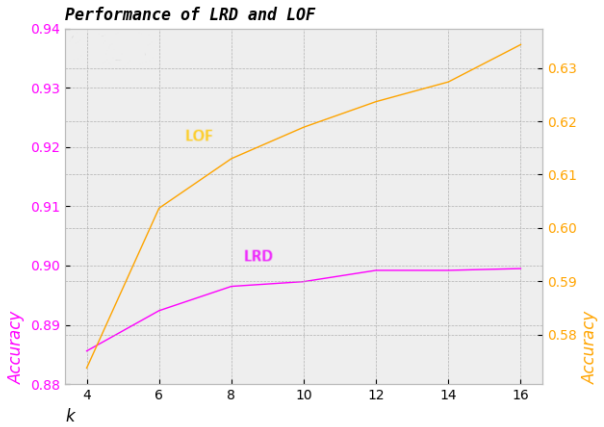


Figure 6.3: k parameter for LOF and LRD on MSCN features

given one mode and variance, then we get a divergence of around 0.9 nats for BRISQUE and 2.0 for LFA. BRISQUE will therefore not be suitable for LOF and LID.

To test this further, we first measured the accuracy of LOF and LRD (LRD detects Hawkings-outliers) on features provided by BRISQUE. The result is shown in Fig. 6.3 for a range of k configurations. The left vertical axis are the accuracies of LRD, and on the right of LOF. These were plotted separately since their distance was too far for showing the shape of the two line plots. Both domains have a width of 0.06 for the accuracy. From the ellbow of the plots we conclude an optimum of $k = 8$ for LOF and LRD; and that value will be chosen for all methods that use kNN. Furthermore, we chose to use $t = 30$ feature bags to smooth out the curve.

Two observations can be made from the line plots. (**i**) The accuracy is higher for LRD with a large margin, that confirms our suspicion of LOF's performance. (**ii**) Also, the value of k has more impact on LOF than for LRD, which also supports the metrics would perform better on more global scope. These two observations are present for LID as well. The second point will now be elaborated.

An extreme example is when the underlying distribution will be a Gaussian. To show that the LOF value of some sample would not change much when displaced under this circumstance, consider now that we write the density as the affiliated Cumulative Density Function (CDF). Then we can determine the expected frequentist parameters regarding Eq. 5.1 (Lemma 6.2.1).

Lemma 6.2.1. *Let $\mathfrak{D}$ be the complete dataset and $f(x,d) = CDF[X-d, X+d]$ be the integral of the cumulative density function of the domain $[X-d, X+d]$. Then the amount of neighbors $|\Omega_{x,k}|$ within radius d is on average $f(X,d) \cdot |\mathfrak{D}|$.*

Proof. The expected value of the region R around X is the integral of the density of that region, which is $f(X,d)$. For N i.i.d observations, the amount of samples that fall in the region is binomially distributed with N trials and $f(X,d)$ chance of success. This has an expected value $\mathbb{E}_{Bin(X|R,N)}[X]$ of $N \cdot R$. $\mathbb{Q}.\mathbb{E}.\mathbb{D}.$

As discussed in Sec. 5.1.1, the cardinality under a certain k-distance is fixed according to the frequentist view and its volume is defined by the sphere of that distance. Lemma 6.2.2 follows from this definition when applied to $f(X,d)$.

Lemma 6.2.2. *k-distance, which is subject to the function $d(X) = f(X,d) - \frac{k}{|\mathfrak{D}|} = 0$, follows the gradient $\nabla f(X,d)$ which is of length zero. That is, some infinitesimal vector $\vec{r}$ where $||\vec{r} \cdot [\frac{\partial f(X,d)}{\partial X}, \frac{\partial f(X,d)}{\partial d}]||_2 = 0$*

Proof. The formula of $d(X)$ follows from Lemma 6.2.1 by reordering the terms to the left side of the equation. Since the density $d(X)$ is constant by definition, it follows that $d(X)$ should be zero. Suppose

Table 6.3: Accuracies on CIFAR10 for rational methods as outlier detection. Evasions have a ℓ_2-radii of 5, best performing in **bold**.

Evasion Attack	LOF-h	LOF	LID-h	LID
FGSM	0.631	**0.633**	0.508	0.534
FGSM (Adv. Learned)	**0.812**	0.646	0.586	0.519
FGSM (RCE)	**0.653**	0.619	0.545	0.527

now that the partial derivatives exist for $f(X, d)$ then the derivative of $d(X) = 0$ can be written as the gradient field $\nabla f(X, d) = 0$. Ignoring the constant term $\frac{k}{|\mathfrak{D}|}$. $\mathbb{Q}.\mathbb{E}.\mathbb{D}.$

Informally, Lemma 6.2.2 causes changes in density to cancel out because the cardinality is homogenous or the volume given by the k-distance (d) is changed. The constraint of function $d(X)$ thus results in an equilibrium. For simplicity, we will consider the one dimensional case for Lemma 6.2.3.

Lemma 6.2.3. *Let $p(X)$ be the probability density function of a region that is strictly increasing (or decreasing) for $\vec{r}$. That means that $\nabla p(X) \cdot \vec{r} > 0$ is true for any X. Then $\vec{r} \cdot \nabla_d d(X)$ reaches its maximum at a point dependent on d.*

Proof. From Lemma 6.2.2 follows an equilibrium, as the density should remain constant. When the input X has one dimension and a region is considered that is strictly increasing or decreasing in X. Then it must be that $\frac{\partial f(X,d)}{\partial X} = p(X + d) - p(X - d)$ equals the difference in density with an updated d to conserve the equilibrium. The maximum change in d is reached when convergence $\frac{\partial^2 f(X,d)}{\partial^2 X} = 0$; the convergence is dependent on d. $\mathbb{Q}.\mathbb{E}.\mathbb{D}.$

These Lemmas can be combined to find the upper bound of LRD's- and LOF's estimate, since the Gaussian has the smoothness properties that are required in Lemma 6.2.2. We conclude this section with Theorem 6.2.4 and a review of the overall accuracy of LID and LOF.

Theorem 6.2.4. *For a Gaussian, the effectiveness of LOF decreases exponentially with d. LRD will degrade linearly with d.*

Proof. Let $p(X)$ be strictly increasing as in Lemma 6.2.3 and part of a normalized Gaussian $\mathcal{N}(0, 1)$. The gradient $\nabla p(X)$ of the Gaussian corresponds to $-X \cdot p(X)$. Substituting this in $\frac{\partial f(X,d)}{\partial X}$ yields the equilibrium:

$$\begin{gathered}(-X - d) \cdot p(X + d) - (-X + d) \cdot p(X - d) = 0 \Rightarrow \\ (-X - d) \cdot \frac{e^{0.5(-d^2 - 2Xd - X^2)}}{\sqrt{2\pi}} - (-X + d) \cdot \frac{e^{0.5(-d^2 + 2Xd - X^2)}}{\sqrt{2\pi}} = 0 \Rightarrow \\ \frac{e^{0.5(-d^2 - X^2)}}{\sqrt{2\pi}} \cdot ((-X - d) \cdot e^{-Xd} - (-X + d) \cdot e^{Xd}) = 0\end{gathered}$$

Simplifying this yields:

$$\begin{gathered}p(X + d) \cdot ((X - d) \cdot e^{2Xd} - X - d) = 0 \Rightarrow \\ X(e^{2Xd} - 1) - d \cdot e^{2Xd} - d = 0 \\ X = -d + \mathcal{O}(e^{-2d^2}) \ \vee \ X = d + \mathcal{O}(e^{-2d^2}).\end{gathered}$$

The limit with d to infinity results thus in a stationary point on d and $-d$. On the other hand, when d goes to zero the stationary point changes to the numerical differentiation of a Gaussian. both cases converge and grow exponential to the above. Now, let LOF and LRD measure a set of random variables from $\mathcal{N}(0, 1)$. Their upper bound is (by rewriting Eq. 5.4):

$$\begin{gathered}LRD(d) = l \cdot d\lambda_l + (1 - l) \cdot d \\ LOF(d) = l \cdot \frac{LRD(d\lambda_l)}{LRD(d)} + (1 - l) \cdot \frac{LRD(d\lambda_h)}{LRD(d)}\end{gathered}$$

Table 6.4: Accuracies of several detection algorithms against grey-, and white-box (GB/WB) adversaries on CIFAR-10, with a ℓ_2-radii of 5. White-box attacks are evaluated on the detector's best performing robust optimization under DeepFool. DeepFool's and C&W's results are dependent on the error rate err_f of the models. A perfect detector has an accuracy of $1 - \frac{err_f}{2}$. Top-3 results are **bolded**.

Evasion Attack	Robust Optim.	POI →	Detection Accuracy ⇑								
			Hidden Units				Scene Statistics				Logits
		WB ↓	**LRD**	**KDE+BU**	**LID**	**EXM**	**LRD**	**SVM**	**PCA**	**FSQ**	**MAH**
FGSM	✗	✗	0.680	0.545	0.508	0.584	**0.698**	**0.725**	**0.722**	0.570	0.596
FGSM	AL	✗	0.852	0.663	0.586	**0.863**	0.815	**0.884**	**0.878**	0.757	0.792
FGSM	RCE	✗	0.644	0.818	0.545	0.726	**0.875**	**0.987**	**0.990**	0.636	0.639
DeepFool	✗	✗	0.546	**0.630**	0.517	0.512	0.490	0.500	0.506	**0.900**	**0.801**
DeepFool	AL	✗	0.516	0.596	0.511	0.499	0.500	0.502	**0.663**	**0.890**	**0.676**
DeepFool	RCE	✗	**0.775**	**0.785**	0.513	0.564	0.498	0.501	0.533	**0.759**	0.756
C&W	AL	✗	0.520	0.561	0.501	0.500	0.502	0.500	**0.758**	**0.744**	**0.635**
C&W	RCE	✗	0.621	**0.707**	0.513	0.558	0.497	0.500	0.582	**0.780**	**0.647**
C&W	Best Perf.	✓	0.516	0.489	0.453	0.450	**0.660**	**0.619**	**0.792**	0.484	0.474

where l is the ratio of neighbors having a lower density; $d\lambda_l$ and $d\lambda_h$ is the average k-distance of the neighbors of lower or higher density. The latter increases exponentially in d. In effect, $LRD(d)$ and $LRD(d\lambda_h)$ differ by similar margins. $\mathbb{Q.E.D.}$

Tab. 6.3 presents the accuracy of LID and LOF for other POIs and confirms Theorem 6.2.4. Generally, the algorithms perform better on the hidden layers when the robustness is optimized. LID, however, achieves an accuracy close to the random classifier; whilst LOF has a reasonable accuracy of 0.812 for one setting. The KL divergence has been measured on the adversarial learned model, of which the results are discussed. LID and LOF show expected results for this model: it has a higher accuracy for LFA features.

6.2.3. Grey Box Detection

We start by examining grey-box attacks. This provides a more comprehensive understanding of our method's performance. Table 6.4 shows LRD besides various other works. Instance-based methods similar to ours are KDE+BU, LID, and MAH. We see that LID does not lead to practical results. On the other hand, LRD reaches an accuracy above 85% for FGSM perturbations, this outperforms similar methods.

Robust optimization is often beneficial. The detection accuracy is frequently higher with one of these methods. Particularly for PCA, its result against DeepFool and FGSM shows a respective difference of 15% and 26%. Moreover, PCA shows strong and similar results as the supervised method SVM on FGSM; the extremal measure, that also uses PCA, is less effective.

Dropout benefits from RCE models, the added entropy to benign examples segregates them from adversarial examples (Appendix B). This method is part of Feinman's KDE [Fei+17]. Feature squeezing is not effective for FGSM perturbations. It achieves the lowest accuracy of 76%.

These results largely change for smaller perturbations. SVM drops from 90%+ accuracy to a random classifier. In fact, almost all methods suffer from smaller perturbations, except for purification measures like feature squeezing. Its situation is reverse for smaller perturbations and does improve in this setting, which suggests that most methods do not have a sufficient scope to cover all evasion methods. This supports the claims of Feinman *et al.* [Fei+17] that a distance-based metric is not enough for cases near the manifold.

6.2.4. White Box Detection

We now turn to experiments in a more challenging scenario, allowing adaptive adversaries. The results are also shown in Table 6.4. We find that most methods can be broken by an adaptive attacker. In particular, the results for KDE+BU, LID, EXM, and MAH showed a true positive rate close to 0% on the

CIFAR-10 and STL-10 datasets. These results do largely agree with prior work [CW17a; ACW18].

Regarding the other methods, we see that especially PCA excels with accuracies of around 80% for CIFAR-10. Surprisingly, PCA was proven as not robust earlier [CW17a]. LRD is, besides PCA, also somewhat resilient against adaptive attacks. Although, it should be noted that BRISQUE does utilize local non-linear operations to estimate generalized gamma functions [MMB12], which are not smooth functions. We can therefore only consider LRD robust on the hidden units.

Some results are worse than in the grey-box setting, that is possible due to the positive confidence value. Hence, the adversarial example is stimulated to be 5% below the detector's threshold, which improves its transferability on models with feature bags or other uncertainties.

Takeaway 6.
Initial experiments on possible heuristics for MeetSafe results in the following:

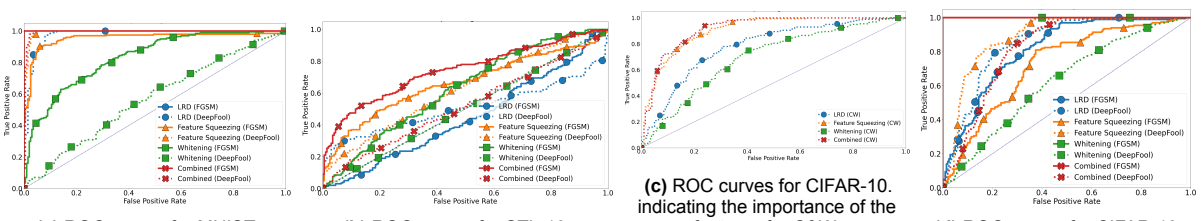
- ☞ *The PCA method of Hendrycks & Gimpel [HG17] works very well with adversarial learning.*
- ☞ *robustness optimization increases the results of detectors and dropout performs especially well with RCE models.*
- ☞ *Most methods are not consistent under different levels of perturbation. SVM varies the most between the variants, this method trains on adversarial examples.*

6.2.5. Ablation Study

We continue to show the fluctuation of three methods (LRD, PCA, and FSQ) across different datasets and compare this to their MeetSafe ensemble. We include LRD since it has the lowest average correlation with PCA and FSQ across all results in Table 4.1. Additionally, LRD showed more resilience against adaptive adversaries compared to similar method. Likewise, PCA and FSQ performed very well on distinct evasion methods.

For the ablation study, the p-values of the method's confidence are used. That is the value which is compared to the threshold. We evaluate the p-values of 10 random FGSM samples on a reverse ResNet-50 under the benign training set. A lower p-value is beneficial for the GMM's generalization, as this assumes normality.
On MNIST, an opposing utility between LRD and whitening techniques becomes clear. Here, LRD has a p-value of near zero ($\leq$ 1e-99), whilst whitening has a value of 6e-6. On the other hand, whitening performs relatively better on CIFAR-10 with a p-value smaller than 1e-80 against 5e-17 for LRD. Whitening and LRD might therefore offset each other's effects against FGSM.

The added value of feature squeezing is apparent for small perturbations. Fig. 6.4 shows the performance of the three detectors for DeepFool and FGSM. It shows a noticeably higher AUROC for feature squeezing on DeepFool. Furthermore, feature squeezing has the smallest p-value of 0.01 for CIFAR-10, followed by LRD with 0.25. Feature squeezing could thus be helpful in the case when the adversarial sample is near the benign input. Indeed, the ROC curves for MeetSafe, which combines all three heuristics, has the highest lowerbound among DeepFool and FGSM. Moreover, experiments on C&W also showed the added value of feature squeezing with a slight benefit LRD (Appendix 6.4).



(a) ROC curves for MNIST. **(b)** ROC curves for STL-10. **(c)** ROC curves for CIFAR-10. indicating the importance of the features for C&W. **(d)** ROC curves for CIFAR-10.

Figure 6.4: ROC curves of MeetSafe's features on a reverse ResNet-50 as target model. The curves show the performance against C&W, DeepFool, and FGSM.

Table 6.5: Accuracies of GMM-based detectors against grey-, and white-box (GB/WB) adversaries with a ℓ_2-radii of 5. DeepFool's and C&W's results are dependent on the error rate err_f of the models, like in Tab. 4.1. The t-CI is based on 5 runs. Top-3 results are **bolded**.

Evasion Attack	Robust Optim.	Dataset / Model →	Detection Accuracy ± [t-CI 95%] ⇑							
			CIFAR-10				STL-10		VGG-13	
		WB ↓	**MS-4**	**MS-8**	**MS-16**	**I-Def**	**MS-8**	**I-Def**	**MS-8**	**I-Def**
FGSM	✗	✗	**0.680**	0.671	**0.672**	0.627	0.520	0.549	**0.808**	0.569
FGSM	AL	✗	**0.876**	0.829	0.853	**0.925**	0.485	0.508	**0.942**	0.508
FGSM	RCE	✗	**0.965**	**0.926**	0.895	0.717	0.737	0.567	**0.956**	0.606
DeepFool	AL	✗	**0.691**	**0.752**	**0.762**	0.520	0.665	0.502	0.686	0.552
DeepFool	RCE	✗	**0.738**	**0.785**	**0.745**	0.571	0.612	0.531	0.657	0.633
C&W	AL	✗	0.746	**0.804**	**0.815**	0.518	0.517	0.498	**0.803**	0.509
C&W	RCE	✗	**0.810**	**0.818**	**0.814**	0.557	0.574	0.518	0.689	0.574
C&W	RCE	✓	**0.762**	**0.745 ± 0.04**	0.689	0.469	0.544	0.475	**0.896**	0.492

We further will extend upon the ablation study by removing each component of MeetSafe systematically. This further establishes the importance of each component within the ensemble. For the ablation, we trained 3 GMMs all with two of the original components (LRD, PCA, FSQ) on a reversed ResNet-50 for the CIFAR-10 dataset. Firstly, when we remove LRD the accuracy decreases by 0.079 for C&W, 0.027 for FGSM, and 0.150 for DeepFool. LRD is thus beneficial for MeetSafe. Similar to the results in Sec. 6.2.3, these results show that LRD does not add much for FGSM perturbations on CIFAR-10. Second, when we remove PCA the accuracy decreases by 0.191 for C&W, 0.234 for FGSM, and 0.145 for DeepFool. Whitening is thus an important component on CIFAR-10. However, Fig. 6.4 shows that this may not be the case for MNIST. Third, when we remove FSQ the accuracy decreases by 0.150 for C&W, 0.101 for FGSM, and -0.040 for DeepFool, which is also in line with the results in Sec. 6.2.3. The scatterplots of Fig. 6.5 also show the relative importance of each component. We observe that either FSQ, PCA, or LRD may be necessary to effectively segregate the data.

Deterministic Feature Bagging

We did also test MeetSafe (MS-8) with deterministic feature bagging. To this extent, the random seeds are set to 144 at the start of evaluating LRD. Without this randomness, MeetSafe has an accuracy of 0.651 for CIFAR-10, 0.990 for MNIST, and 0.516 for STL-10 against an adaptive, white-box adversary.

6.2.6. GMM-based Detection

The prior section established that each of these methods has at least one scenario that particularly favors one over the others. Based on these results, we have chosen PCA, feature squeezing and LRD for our final model. Following that, We test MeetSafe's performance under a certain number of latent components: 4, 8, and 16 (Table 6.5). For grey-box perturbations, we see that only 4 latent variables may be useful for FGSM, but this increases for small perturbations. To balance these accuracies, we think that 8 components is desirable. Comparing the performance of MS-8 to that of I-Defender shows

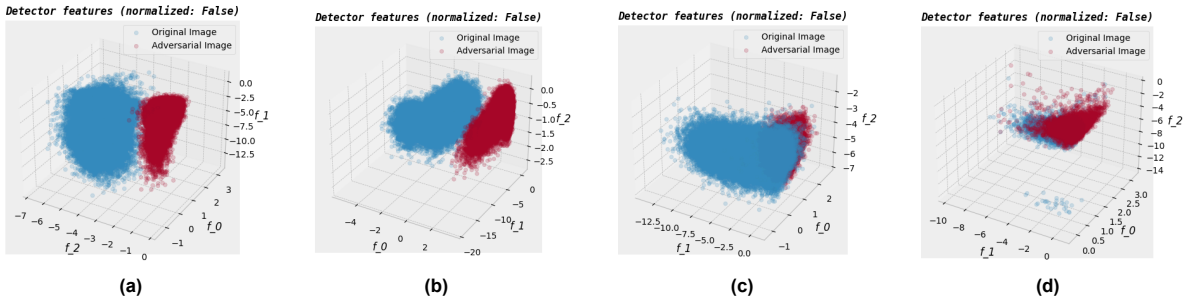


Figure 6.5: Scores of LRD (f_0), feature squeezing (f_1), and whitening (f_2) before and after a perturbation with a ℓ_2-radii of 5. The scores were in the same configuration as used by MeetSafe. (**a**) Scores for a reverse ResNet-50 model on CIFAR-10; evaded with FGSM (**b**) Scores for a reverse ResNet-50 model on MNIST; evaded with FGSM (**c**) Scores for a reverse ResNet-50 model on CIFAR-10; evaded with DeepFool (**d**) Scores for a reverse ResNet-50 model on STL-10; evaded with FGSM.

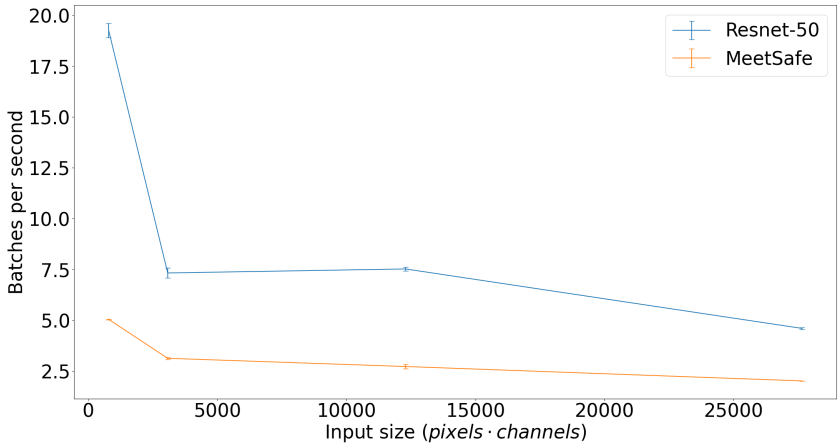


Figure 6.6: Graph that shows the inference times of MeetSafe and a reversed ResNet-50 for a variation of input sizes. The data points are from the datasets: MNIST, CIFAR-10, Tiny-ImageNet, and STL-10. The error bar denotes the t-CI across 5 runs.

similar results on FGSM, but lower accuracies on stronger attacks.

Though, it should be noted that the accuracy for RCE and MeetSafe do not scale well. This is most notable when we compare the efficacy for datasets of higher resolution. Tiny-ImageNet shows accuracies on C&W of at most 0.553 for MeetSafe and 0.511 for I-Defender. STL-10 shows similar results (Table 6.5). However, the effect of dimensionality is not a limitation specific to our method but rather a general issue of defenses against adversarial examples [GSS15; Sha+18].

MS-8 achieves an improvement of at least 8.1% on adaptive attacks and 10.2% on the worst-case results for each evaluated method by averaging across STL-10, MNIST, and CIFAR-10. MeetSafe may therefore be employed universally whilst maintaining a considerable detection accuracy.

6.2.7. Inference Times

Fig. 6.6 shows the time taken for performing inference on MeetSafe and its target model when using a reversed ResNet-50. We test the inference time on all four datasets: MNIST, CIFAR-10, Tiny-ImageNet, and STL-10. These are plotted ad four data points on the x-axis according to its input size: $\dim(\mathbb{R}^{1\times28\times28})$, $\dim(\mathbb{R}^{3\times32\times32})$, $\dim(\mathbb{R}^{3\times64\times64})$, and $\dim(\mathbb{R}^{3\times96\times96})$ respectively. From the figure, we can observe that the discrepancy of ResNet-50 and MeetSafe converges to a factor of about 2.3 when the input size gets larger. The inference time seems therefore reasonable, after training. Especially for larger input sizes.

To challenge the performance of MeetSafe further, we increased the feature size from 10 to 17 and the pool size from 500 to 850. More could not fit in our VRAM. We anticipated that this would lead to increased computation, especially for LRD. The results on CIFAR-10 demonstrate that this increase led to a slower processing rate, with the model running 0.28 batches per second slower than before. We consider this change to be limited, indicating that the computational overhead is also manageable given the increased feature and pool sizes.

Complexity

The worst-case computational complexity of MeetSafe is $\mathcal{O}(4n + \dim(\mathfrak{D}) + k \cdot d)$ where d is the input dimension for LRD and n the target model's parameters. The complexity consists of 3 forward passes for feature squeezing $\mathcal{O}(3n)$, 1 forward pass to acquire the hidden units for LRD $\mathcal{O}(n + k \cdot d)$. For which we refer to Corollary 5.1.0.1 that describes the complexity of LRD. Whitening needs $\mathcal{O}(\dim(\mathfrak{D}) \cdot c)$ to decompose the dataset into c eigenvectors [HMT11, Algorithm 5.1]. Even though the GMM has an exponential complexity in with respect to the input dimension, our application only takes the three heuristics as features, regardless of the dataset size and target model.

7

Concluding Remarks

The purpose of this thesis was to analyze whether the 'meet the defence' approach [Ald+22] of detectors leads to white box robustness, whilst retaining a linear computational complexity. Our literature review presents four possible shortcomings in current technology (Chap. 3). This thesis challenges at least three of them with the hypothesis: *Detectors fail in white box situation when it is either linear, overfit, or has narrow statistical regularities.*

A novel detector, MeetSafe, was proposed, to improve the state-of-the-art. It leverages variance and a new distance-based metric (called LRD) in GMMs; harmonizing prevention and detection techniques to ensure sound detection for both small and larger perturbations. Since not all units of the neural network may be of interest, MeetSafe will perform a fast feature selection based on FGSM perturbations.

The hypothesis is tested based on in-vitro experiments with C&W white box attacks and other grey box methods that generate a near-optimal ℓ_2 perturbation. The results of MeetSafe can be used as support for the hypothesis, when compared to methods that are either linear, trained on adversarial examples, or impose weak constraints on the adversary. We were surprised by a few outcomes, that forced us to revise the hypothesis to the one below. The reason for this will become clear next section.

Hypothesis 1 (Revised):

Detectors fail in white box situation when it is either overfit, limited by the target's robustness, or has narrow statistical regularities.

7.1. Findings & Discussion

The experiments show promising results of our proposed methods: LRD and MeetSafe. LRD alone has a lower complexity than Feinman's KDE [Fei+17]. LRD's theoretical complexity is $\mathcal{O}(k \log(n))$ instead of $\mathcal{O}(n)$ for k neighbors. Additionally, it performs much better on FGSM and white-box perturbations. MeetSafe more effectively detects adversarial examples from a wide range of attacks and datasets. Especially compared to our evaluated methods. MeetSafe achieves accuracies of around 80% with a TNR of at least 90%. We observed only major difficulties for input spaces with a lot of classes; but this was not unique to MeetSafe. The main findings are as follows.

Finding 1: RCE optimization is sensitive to the amount of labels

Input spaces that have a large class cardinality are not beneficial to the robustness of target models. This can be observed by the accuracy of Reverse Cross Entropy (RCE) [Pan+18] on higher dimensions. RCE stimulates neural networks to learn distinct features. A large set of classes, however, would mean that the training is more constrained. Leading to a slow convergence of the network and, in extreme cases, unacceptable accuracies of 3%. Goodfellow *et al.* [GSS15] already argued that adversarial examples come naturally from an increased dimension. We think that this holds true for labels as well. Maybe it has an even higher impact, as there are methods to purify images by reducing the degrees of freedom [XEQ18]. Adversarial training shows instability in another way. It seems to diverge after the

model has learned to classify benign images. Most likely, the adversarial penalty causes the model to diverge.

Recommendation: *limit the number of classes, apply class-incremental learning, and adjust the learning rate η on the applied robustness optimization.*

Finding 2: Robustness optimization is beneficial for detection

Robustness optimization decreases the classification error on small perturbations and are of additional value during detection, as optimized models had higher detection rates for any evasion method. We observed an additional accuracy of around 20% when such an approach was taken. With this result, some earlier work can be put into question. Carlini & Wagner [CW17a] considered PCA broken in the white box setting; but this is actually not true for optimized models. We consider it almost as effective as MeetSafe, and the only method besides MeetSafe to be robust under white box perturbations. However, we find that the performance of PCA primarily stands out on CIFAR10. MeetSafe performs well on more datasets. We therefore conclude that the robustness of the target model is a crucial limiting factor and not the linearity of PCA.

The last layer is often the best choice to detect perturbation, except for adversarial training. Our experiments revealed that the sensitivity to perturbation is highest on the last layer of plain and RCE models. Our feature selection algorithm will thus always take the last layer out of all hidden layers. That supports the work of Bengio *et al.* [Ben+13], they argue that the deeper layers provide more linear and 'unwrapped' manifolds to work with. Still, we find the exact reverse behavior for adversarial training, which suggests that selection on a layer's or hidden unit's utility may be beneficial.

Recommendation: *add robustness optimization and apply any detection where there the ℓ_2 distance is high between benign and adversarial samples.*

Finding 3: Narrow statistical regularities

Feinman *et al.* [Fei+17] simplified adversarial examples to be in one of two scenarios. These are either near or far away from the manifold. Our results showed that one metric can hardly cover both situations. For instance, distance-based metrics are good at detecting high perturbations, but lose a significant accuracy on lower ones. Feature squeezing has similar issues for larger perturbations, and this is also reflected by the performance illustrated in the original paper [XEQ18]. Pixel binning the input can purify the image for perturbations within a certain interval; but will not solve the problem for larger perturbations. Hendrycks & Gimpel [HG17] call such metrics too narrow. They proposed that detectors should instead combine imperfect measures in an ensemble. The improved generalization that we observed for MeetSafe and Feinman *et al.*'s KDE [Fei+17] along multiple datasets supports these claims.

Recommendation: *combine distance-based metrics with variance metrics.*

Finding 4: Local anomalies do not detect adversarial examples

Local anomaly detection fails in most cases, especially for raw image features (i.e. BRISQUE). These methods suffer under the gradual change in the density of these features. Local outliers are not defined by the global view of the data. Rather, these are anomalies relative to their local neighborhoods [Bre+00]. These only work under a large differential entropy. The absence of this in some features results in a better performance for global anomaly detection, like LRD or KDE. This is called Hawkins-outlier detection.

Recommendation: *use Hawkins-outlier detection.*

7.1.1. Limitations

The experiments in this thesis focus on the robustness of MeetSafe and other detectors under near-optimal perturbations. This is one specific type of perturbation out of several others. For instance, other types include physical perturbations [Zho+22], semantic perturbations [GSG20], and fooling perturbations [RD18]. Whilst the ℓ_2 distance of these other methods is not necessarily small, they do assume that the manipulation is constrained to human imperceptibility.

Yet, practical applications might already be vulnerable with human perceptibility. When one follows this argument, it can be concluded that adversarial images can't be mitigated, since an unrestricted, inter-means perturbation is almost always sufficient to misclassify the sample [Tra+17].

Also, this paper only covers the general case; but There are more specialized applications. For instance, the perturbation for language models has to have semantic similarity [ZDS17]. The discussed evasion methods would generate ungrammatical sentences. Similar work is the shadow attack [GSG20], encoder attack [KW13], and semantic segmentation attack [Xie+17]. The performance of MeetSafe is partially unknown for semantic examples. This may also be true for some other architectures. While our experiments were conducted using two widely adopted architectures, we can not conclude similar performance on architectures like transformers, or even other machine learning algorithms.

Similarly, we are aware that there are multiple stochastic defenses published that may achieve higher accuracies. On the other hand, we also have seen significant shortcomings of stochastic defenses and denoising. Athalye et al. [ACW18] were the first to criticize the use of randomness in defenses (stochastic gradients), as these may be easily circumvented with Backward Pass Differentiable Approximation (BPDA). The paper [Tra+20], that was published later, likewise evaded a lot of stochastic defenses. Because the performance of obfuscated gradients often prove wrong or is at least questionable, we did only refer to the apprehension in the introduction. Additionally, we believe some methods like denoisers or JTLA [Rag+21] would not offer a fair comparison given their computational cost. We thus focus on detectors that use local invariants and have quasi-deterministic gradients.

Another limitation of our research is due to resource constraints. We considered I-Defender and Feinman *et al.*'s [Fei+17] KDE not practical in certain situations. For models like ResNet-50, it would cost at least $372.53\ GiB$ to evaluate I-Defender. On the other hand, KDE+BU required a large computational graph in Pytorch during white box testing. That was because of the tens of forwards for dropout. For the same reason, we could only utilize $U_{f^{(l)}}$ for the extremal value. Other methods (LRD, LID, and KDE+BU), require instance-based learning and more memory. Also, we left the evaluation of vision transformers, large language models and other transformers for future work. Such architectures are vastly different from CNNs, so we may not conclude the performance of MeetSafe on transformers. Lastly, we solely relied on BRISQUE image quality features for our experiments. These may not capture all relevant aspects of image quality.

A key property of MeetSafe is the diversity of its components which makes it more robust under larger range of datasets. Other methods may outperform MeetSafe on specific attacks. For example, SVMs achieve higher accuracies on FGSM (Table 6.4); but such improvement is not universal across all attacks. The consistency and robustness against white-box attacks is exactly what makes MeetSafe superior to related work. Even under considerable perturbations, like adversarial examples with a ℓ_2 distance of 5. MeetSafe manages to separate such adversarial examples from benign ones; even though perturbation is often perceptible, as shown in Fig. 7.1.

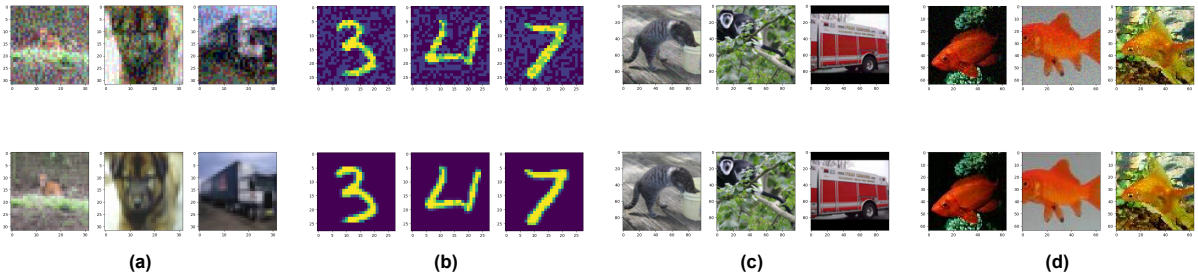


Figure 7.1: FGSM perturbations with an ℓ_2 of 5 and applied on the top row. Images were sampled from (**a**) CIFAR-10, (**b**) MNIST, (**c**) STL-10, and (**d**) ImageNet. The target model was a reverse ResNet-50 model.

Relaxing the Requirements

The findings and limitations of this thesis showed the requirements in Chap. 3 to be presumptuous. For instance, considering the model-agnostic perturbation of Tramer *et al.* [Tra+17] one can conclude that detecting *any* evasion method is an overstatement and not feasible. Instead, we should relax this requirement to human perceivable perturbations (**R2**) and even this is not really possible for larger dimensions.

In addition, We observed that all evaluated defenses against adversarial examples do not scale well. This is especially true when training models with RCE (Finding 1). We can therefore not ensure that the target model remains accurate on larger feature spaces (**R5**). However, it does aid detection on at least CIFAR-10 and MNIST whilst remaining accurate on the sample's classification.

7.2. Conclusion

In conclusion, this thesis has investigated the efficacy of various defense mechanisms against adversarial attacks, focusing particularly on the 'meet the defence' approach and the development of the MeetSafe detector. The thesis identifies potential shortcomings in existing technology and formulating hypotheses to challenge these limitations. The subsequent development of MeetSafe, leveraging variance and novel distance-based metrics in Gaussian Mixture Models (GMMs), represents a significant advancement in the field of adversarial defense mechanisms.

7.2.1. Future Work

Future research should explore the application of robustness optimization and detection techniques across diverse perturbation types, including semantic attacks and fooling perturbations. Additionally, the dimensionality of the classes and its effect on robustness can lead to interesting insights. Lastly, we limited the image quality features to BRISQUE. Other techniques or an ensemble of them might be of added value.

References

[AC18] Anish Athalye and Nicholas Carlini. “On the robustness of the cvpr 2018 white-box adversarial example defenses”. In: *arXiv preprint arXiv:1804.03286* (2018).

[ACW18] Anish Athalye, Nicholas Carlini, and David A. Wagner. “Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples”. In: *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*. Vol. 80. Proceedings of Machine Learning Research. [Accessed 11-07-2024]. PMLR, 2018, pp. 274–283. URL: http://proceedings.mlr.press/v80/athalye18a.html.

[Ald+22] Ahmed Aldahdooh et al. “Adversarial example detection for DNN models: a review and experimental comparison”. In: *Artif. Intell. Rev.* 55.6 (2022), pp. 4403–4462. DOI: 10.1007/S10462-021-10125-W. URL: https://doi.org/10.1007/s10462-021-10125-w.

[AMF18] Zahid Akhtar, João Monteiro, and Tiago H. Falk. “Adversarial Examples Detection Using No-Reference Image Quality Features”. In: *2018 International Carnahan Conference on Security Technology, ICCST 2018, Montreal, QC, Canada, October 22-25, 2018*. IEEE, 2018, pp. 1–5. DOI: 10.1109/CCST.2018.8585591. URL: https://doi.org/10.1109/CCST.2018.8585591.

[And+20] Maksym Andriushchenko et al. “Square attack: a query-efficient black-box adversarial attack via random search”. In: *European Conference on Computer Vision (ECCV)*. Springer. 2020, pp. 484–501.

[Bai+21] Tao Bai et al. “Recent Advances in Adversarial Training for Adversarial Robustness”. In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*. ijcai.org, 2021, pp. 4312–4321. DOI: 10.24963/IJCAI.2021/591. URL: https://doi.org/10.24963/ijcai.2021/591.

[Bas+16] Osbert Bastani et al. “Measuring neural net robustness with constraints”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 29 (2016).

[Ben+13] Yoshua Bengio et al. “Better mixing via deep representations”. In: *International Conference on Machine Learning (ICML)*. PMLR. 2013, pp. 552–560.

[Ben75] Jon Louis Bentley. “Multidimensional binary search trees used for associative searching”. In: *Commun. ACM* 18.9 (Sept. 1975), pp. 509–517. ISSN: 0001-0782. DOI: 10.1145/361002.361007. URL: https://doi.org/10.1145/361002.361007.

[Bis06] C Bishop. “Pattern recognition and machine learning”. In: *Springer* 2 (2006), pp. 35–42.

[BR18] Battista Biggio and Fabio Roli. “Wild patterns: Ten years after the rise of adversarial machine learning”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, pp. 2154–2156.

[Bre+00] Markus M. Breunig et al. “LOF: Identifying Density-Based Local Outliers”. In: *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*. ACM, 2000, pp. 93–104. DOI: 10.1145/342009.335388. URL: https://doi.org/10.1145/342009.335388.

[CH20a] Francesco Croce and Matthias Hein. “Minimally distorted adversarial examples with a fast adaptive boundary attack”. In: *International Conference on Machine Learning (ICML)*. PMLR. 2020, pp. 2196–2205.

[CH20b] Francesco Croce and Matthias Hein. “Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks”. In: *International Conference on Machine Learning (ICML)*. PMLR. 2020, pp. 2206–2216.

[Cha+19] Kathleen Champion et al. “Data-driven discovery of coordinates and governing equations”. In: *Proceedings of the National Academy of Sciences* 116.45 (2019), pp. 22445–22451.

[Che+17] Pin-Yu Chen et al. “Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models”. In: *Proceedings of the 10th ACM workshop on artificial intelligence and security*. 2017, pp. 15–26.

[Cis+17] Moustapha Cisse et al. “Parseval networks: Improving robustness to adversarial examples”. In: *International Conference on Machine Learning (ICML)*. PMLR. 2017, pp. 854–863.

[CNL11] Adam Coates, Andrew Y. Ng, and Honglak Lee. “An Analysis of Single-Layer Networks in Unsupervised Feature Learning”. In: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2011, Fort Lauderdale, USA, April 11-13, 2011*. Vol. 15. JMLR Proceedings. [Accessed 11-07-2024]. JMLR.org, 2011, pp. 215–223. URL: http://proceedings.mlr.press/v15/coates11a/coates11a.pdf.

[CSG20] Gilad Cohen, Guillermo Sapiro, and Raja Giryes. “Detecting adversarial samples using influence functions and nearest neighbors”. In: *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 14453–14462.

[CW16] Nicholas Carlini and David Wagner. “Defensive distillation is not robust to adversarial examples”. In: *arXiv preprint arXiv:1607.04311* (2016).

[CW17a] Nicholas Carlini and David A. Wagner. “Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods”. In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security, AISec@CCS 2017, Dallas, TX, USA, November 3, 2017*. ACM, 2017, pp. 3–14. DOI: 10.1145/3128572.3140444. URL: https://doi.org/10.1145/3128572.3140444.

[CW17b] 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*. IEEE Computer Society, 2017, pp. 39–57. DOI: 10.1109/SP.2017.49. URL: https://doi.org/10.1109/SP.2017.49.

[DL92] Harris Drucker and Yann Le Cun. “Improving generalization performance using double backpropagation”. In: *IEEE transactions on neural networks* 3.6 (1992), pp. 991–997.

[Dua+21] Ranjie Duan et al. “Advdrop: Adversarial attack to dnns by dropping information”. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*. 2021, pp. 7506–7515.

[ECW20] Hasan Ferit Eniser, Maria Christakis, and Valentin Wüstholz. “Raid: Randomized adversarial-input detection for neural networks”. In: *arXiv preprint arXiv:2002.02776* (2020).

[Fei+17] Reuben Feinman et al. “Detecting adversarial samples from artifacts”. In: *arXiv preprint arXiv:1703.00410* (2017).

[FJR15] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. “Model inversion attacks that exploit confidence information and basic countermeasures”. In: *Proceedings of the 22nd ACM SIGSAC conference on computer and communications security*. 2015, pp. 1322–1333.

[Fle70] Roger Fletcher. “A new approach to variable metric algorithms”. In: *The computer journal* 13.3 (1970), pp. 317–322.

[GF17] Bryce Goodman and Seth Flaxman. “European Union regulations on algorithmic decision-making and a “right to explanation””. In: *AI magazine* 38.3 (2017), pp. 50–57.

[GG16] Yarin Gal and Zoubin Ghahramani. “Dropout as a bayesian approximation: Representing model uncertainty in deep learning”. In: *International Conference on Machine Learning (ICML)*. PMLR. 2016, pp. 1050–1059.

[Gro+17] Kathrin Grosse et al. “On the (statistical) detection of adversarial examples”. In: *arXiv preprint arXiv:1702.06280* (2017).

[GSG20] Amin Ghiasi, Ali Shafahi, and Tom Goldstein. “Breaking Certified Defenses: Semantic Adversarial Examples with Spoofed robustness Certificates”. In: *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. [Accessed 11-07-2024]. 2020. URL: https://openreview.net/forum?id=HJxdTxHYvB.

[GSS15] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. “Explaining and Harnessing Adversarial Examples”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. [Accessed 11-07-2024]. 2015. URL: http://arxiv.org/abs/1412.6572.

[He+16] Kaiming He et al. “Deep Residual Learning for Image Recognition”. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90. URL: https://doi.org/10.1109/CVPR.2016.90.

[He+17] Warren He et al. “Adversarial example defense: Ensembles of weak defenses are not strong”. In: *11th USENIX workshop on offensive technologies (WOOT 17)*. 2017.

[HG17] Dan Hendrycks and Kevin Gimpel. “Early Methods for Detecting Adversarial Images”. In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. [Accessed 11-07-2024]. 2017. URL: https://openreview.net/forum?id=B1dexpDug.

[HMT11] Nathan Halko, Per-Gunnar Martinsson, and Joel A Tropp. “Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions”. In: *SIAM review* 53.2 (2011), pp. 217–288.

[Hu+19] Shengyuan Hu et al. “A new defense against adversarial images: Turning a weakness into a strength”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 32 (2019).

[KB15] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. [Accessed 11-07-2024]. 2015. URL: http://arxiv.org/abs/1412.6980.

[KGB18] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. “Adversarial examples in the physical world”. In: *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112.

[KH09] A. Krizhevsky and G. Hinton. “Learning multiple layers of features from tiny images”. In: *Master’s thesis, Department of Computer Science, University of Toronto* (2009). [Accessed 11-07-2024].

[Khe+20] Anouar Kherchouche et al. “Detection of adversarial examples in deep neural networks with natural scene statistics”. In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–7.

[KL51] Solomon Kullback and Richard A Leibler. “On information and sufficiency”. In: *The annals of mathematical statistics* 22.1 (1951), pp. 79–86.

[KM72] Victor Klee and George J Minty. “How good is the simplex algorithm”. In: *Inequalities* 3.3 (1972), pp. 159–175.

[KN98] Edwin M Knox and Raymond T Ng. “Algorithms for mining distancebased outliers in large datasets”. In: *Proceedings of the international conference on very large data bases*. Citeseer. 1998, pp. 392–403.

[KW13] Diederik P Kingma and Max Welling. “Auto-encoding variational bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[LBS19] Yingzhen Li, John Bradshaw, and Yash Sharma. “Are generative classifiers more robust to adversarial attacks?” In: *International Conference on Machine Learning (ICML)*. PMLR. 2019, pp. 3804–3814.

[LeC98] Yann LeCun. *The MNIST database of handwritten digits*. http://yann.lecun.com/exdb/mnist/. 1998.

[Lee+18] Kimin Lee et al. “A simple unified framework for detecting out-of-distribution samples and adversarial attacks”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 31 (2018).

[Li+22] Zewen Li et al. “A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects”. In: *IEEE Transactions on Neural Networks and Learning Systems* 33.12 (2022), pp. 6999–7019. DOI: 10.1109/TNNLS.2021.3084827.

[Lia+18] Bin Liang et al. “Detecting adversarial image examples in deep neural networks with adaptive noise reduction”. In: *IEEE Transactions on Dependable and Secure Computing* 18.1 (2018), pp. 72–85.

[Lit+17] Geert Litjens et al. “A survey on deep learning in medical image analysis”. In: *Medical image analysis* 42 (2017), pp. 60–88.

[LK05] Aleksandar Lazarevic and Vipin Kumar. “Feature bagging for outlier detection”. In: *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*. 2005, pp. 157–166.

[LL17] Xin Li and Fuxin Li. “Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics”. In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 5775–5783. DOI: 10.1109/ICCV.2017.615. URL: https://doi.org/10.1109/ICCV.2017.615.

[LSB09] Nour-Eddine Lasmar, Youssef Stitou, and Yannick Berthoumieu. “Multiscale skewed heavy tailed model for texture analysis”. In: *2009 16th IEEE International Conference on Image Processing (ICIP)*. IEEE. 2009, pp. 2281–2284.

[Luo+22] Cheng Luo et al. “Frequency-driven imperceptible adversarial attack on semantic similarity”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)*. 2022, pp. 15315–15324.

[LY15] Ya Le and Xuan Yang. “Tiny imagenet visual recognition challenge”. In: *CS 231N* 7.7 (2015), p. 3.

[Ma+18] Xingjun Ma et al. “Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality”. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. [Accessed 11-07-2024]. 2018. URL: https://openreview.net/forum?id=B1gJ1L2aW.

[MFF16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. “DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016, pp. 2574–2582.

[MMB12] Anish Mittal, Anush Krishna Moorthy, and Alan Conrad Bovik. “No-reference image quality assessment in the spatial domain”. In: *IEEE Transactions on image processing* 21.12 (2012), pp. 4695–4708.

[Moo+17] Seyed-Mohsen Moosavi-Dezfooli et al. “Universal adversarial perturbations”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1765–1773.

[NYC15] Anh Nguyen, Jason Yosinski, and Jeff Clune. “Deep neural networks are easily fooled: High confidence predictions for unrecognizable images”. In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 427–436.

[Pan+18] Tianyu Pang et al. “Towards robust detection of adversarial examples”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 31 (2018), pp. 4579–4589.

[Pap+16a] Nicolas Papernot et al. “Distillation as a defense to adversarial perturbations against deep neural networks”. In: *2016 IEEE symposium on security and privacy (SP)*. IEEE. 2016, pp. 582–597.

[Pap+16b] Nicolas Papernot et al. “The limitations of deep learning in adversarial settings”. In: *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE. 2016, pp. 372–387.

[Pap+17] Nicolas Papernot et al. “Practical black-box attacks against machine learning”. In: *Proceedings of the 2017 ACM on Asia conference on computer and communications security*. 2017, pp. 506–519.

[PM18] Nicolas Papernot and Patrick McDaniel. “Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning”. In: *arXiv preprint arXiv:1803.04765* (2018).

[Rag+21] Jayaram Raghuram et al. “A general framework for detecting anomalous inputs to dnn classifiers”. In: *International Conference on Machine Learning (ICML)*. [Accessed 11-07-2024]. PMLR. 2021, pp. 8764–8775.

[RD18] Andrew Ross and Finale Doshi-Velez. “Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. “Learning representations by back-propagating errors”. In: *nature* 323.6088 (1986), pp. 533–536.

[RSL18] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. “Certified Defenses against Adversarial Examples”. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. 2018. URL: https://openreview.net/forum?id=Bys4ob-Rb.

[Rud94] Daniel L Ruderman. “The statistics of natural images”. In: *Network: computation in neural systems* 5.4 (1994), p. 517.

[SGV99] Craig Saunders, Alex Gammerman, and Volodya Vovk. “Transduction with Confidence and Credibility”. In: *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*. [Accessed 11-07-2024]. Morgan Kaufmann, 1999, pp. 722–726. URL: http://ijcai.org/Proceedings/99-2/Papers/010.pdf.

[Sha+18] Ali Shafahi et al. “Are adversarial examples inevitable?” In: *arXiv preprint arXiv:1809.02104* (2018).

[Sil18] Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

[SL95] Karnran Sharifi and Alberto Leon-Garcia. “Estimation of shape parameter for generalized Gaussian distributions in subband decompositions of video”. In: *IEEE Trans. Circuits Syst. Video Technol.* 5.1 (1995), pp. 52–56. DOI: 10.1109/76.350779. URL: https://doi.org/10.1109/76.350779.

[SN20] Samuel Henrique Silva and Peyman Najafirad. “Opportunities and challenges in deep learning adversarial robustness: A survey”. In: *arXiv preprint arXiv:2007.00753* (2020).

[Sol19] Stephanie Solomon. *How workforce planning analytics builds stronger businesses*. [Accessed 11-07-2024]. Apr. 2019. URL: https://www.ibm.com/blog/how-workforce-planning-analytics-builds-stronger-businesses/.

[Son+18] Yang Song et al. “PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples”. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. 2018. URL: https://openreview.net/forum?id=rJUYGxbCW.

[SZ15] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. [Accessed 11-07-2024]. 2015. URL: http://arxiv.org/abs/1409.1556.

[Sze+14] Christian Szegedy et al. “Intriguing properties of neural networks”. In: *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. [Accessed 11-07-2024]. 2014. URL: http://arxiv.org/abs/1312.6199.

[TG16] Thomas Tanay and Lewis Griffin. “A boundary tilting persepective on the phenomenon of adversarial examples”. In: *arXiv preprint arXiv:1608.07690* (2016).

[Tia+21] Jinyu Tian et al. “Detecting Adversarial Examples from Sensitivity Inconsistency of Spatial-Transform Domain”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 35.11 (May 2021). [Accessed 11-07-2024], pp. 9877–9885. DOI: 10.1609/aaai.v35i11.17187. URL: https://ojs.aaai.org/index.php/AAAI/article/view/17187.

[Tia+22] Zhiyi Tian et al. “A comprehensive survey on poisoning attacks and countermeasures in machine learning”. In: *ACM Computing Surveys* 55.8 (2022), pp. 1–35.

[Tra+17] Florian Tramèr et al. “The space of transferable adversarial examples”. In: *arXiv preprint arXiv: 1704.03453* (2017).

[Tra+18] Florian Tramèr et al. “Ensemble Adversarial Training: Attacks and Defenses”. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. [Accessed 11-07-2024]. 2018. URL: https://openreview.net/forum?id=rkZvSe-RZ.

[Tra+20] Florian Tramer et al. “On adaptive attacks to adversarial example defenses”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 33 (2020), pp. 1633–1645.

[Tra22] Florian Tramer. “Detecting adversarial examples is (nearly) as hard as classifying them”. In: *International Conference on Machine Learning (ICML)*. PMLR. 2022, pp. 21692–21702.

[Tsa+21] Andreas Tsamados et al. “The ethics of algorithms: key problems and solutions”. In: *Ethics, Governance, and Policies in Artificial Intelligence* (2021), pp. 97–123.

[VV22] Daniël Vos and Sicco Verwer. “Robust optimal classification trees against adversarial examples”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 8. 2022, pp. 8520–8528.

[Wen+18a] Lily Weng et al. “Towards fast computation of certified robustness for relu networks”. In: *International Conference on Machine Learning (ICML)*. [Accessed 11-07-2024]. PMLR. 2018, pp. 5276–5285.

[Wen+18b] Tsui-Wei Weng et al. “Evaluating the Robustness of Neural Networks: An Extreme Value Theory Approach”. In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. 2018. URL: https://openreview.net/forum?id=BkUHlMZ0b.

[Wol69] Philip Wolfe. “Convergence conditions for ascent methods”. In: *SIAM review* 11.2 (1969), pp. 226–235.

[XEQ18] Weilin Xu, David Evans, and Yanjun Qi. “Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks”. In: *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.

[Xie+17] Cihang Xie et al. “Adversarial examples for semantic segmentation and object detection”. In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2017, pp. 1369–1378.

[YGZ18] Ziang Yan, Yiwen Guo, and Changshui Zhang. “Deep defense: Training dnns with improved adversarial robustness”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 31 (2018).

[YKR19] Xuwang Yin, Soheil Kolouri, and Gustavo K Rohde. “Gat: Generative adversarial training for adversarial example detection and robust classification”. In: *arXiv preprint arXiv:1905.11475* (2019).

[ZCW21] Zhanyuan Zhang, Yizheng Chen, and David Wagner. “Seat: similarity encoder by adversarial training for detecting model extraction attack queries”. In: *Proceedings of the 14th ACM Workshop on Artificial Intelligence and Security*. 2021, pp. 37–48.

[ZDS17] Zhengli Zhao, Dheeru Dua, and Sameer Singh. “Generating natural adversarial examples”. In: *arXiv preprint arXiv:1710.11342* (2017).

[ZH18] Zhihao Zheng and Pengyu Hong. “Robust detection of adversarial attacks by modeling the intrinsic properties of deep neural networks”. In: *Advances in Neural Information Processing Systems (NeurIPS)* 31 (2018).

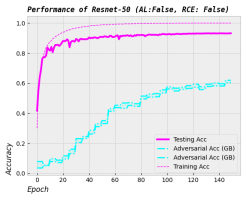
[Zha+03] Wenyi Zhao et al. “Face recognition: A literature survey”. In: *ACM computing surveys (CSUR)* 35.4 (2003), pp. 399–458.

[Zho+22] Yiqi Zhong et al. “Shadows can be dangerous: Stealthy and effective physical-world adversarial attack by natural phenomenon”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022, pp. 15345–15354.

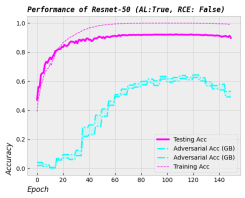
[ZLL20] Zhengyu Zhao, Zhuoran Liu, and Martha Larson. “Towards large yet imperceptible adversarial image perturbations with perceptual color distance”. In: *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 1039–1048.

A

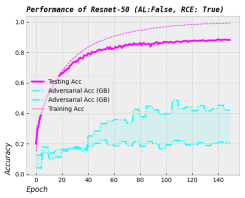
Convergence Graphs of the Trained Models



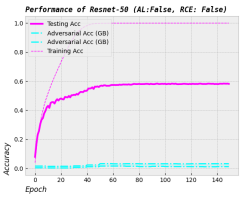
(a) Convergence graph for ResNet-50, trained on CIFAR-10. Showing the training, testing, and Grey Box (GB) adversarial (FGSM, ℓ_2: 2-8) accuracy on a scale between 0 and 1. The hyperparameters are given in-text.



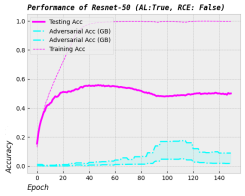
(b) Convergence graph for an adversarial learned ResNet-50, trained on CIFAR-10. Showing the training, testing, and Grey Box (GB) adversarial (FGSM, ℓ_2: 2-8) accuracy on a scale between 0 and 1. The hyperparameters are given in-text.



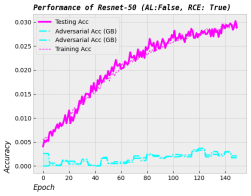
(c) Convergence graph for the reverse ResNet-50, trained on CIFAR-10. Showing the training, testing, and Grey Box (GB) adversarial (FGSM, ℓ_2: 2-8) accuracy on a scale between 0 and 1. The hyperparameters are given in-text.



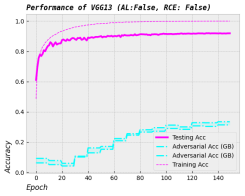
(d) Convergence graph for ResNet-50, trained on Tiny-ImageNet. Showing the training, testing, and Grey Box (GB) adversarial (FGSM, ℓ_2: 2-8) accuracy on a scale between 0 and 1. The hyperparameters are given in-text.



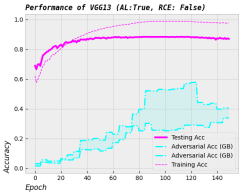
(e) Convergence graph for an adversarial learned ResNet-50, trained on Tiny-ImageNet. Showing the training, testing, and Grey Box (GB) adversarial (FGSM, ℓ_2: 2-8) accuracy on a scale between 0 and 1. The hyperparameters are given in-text.



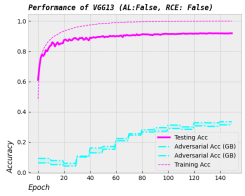
(f) Convergence graph for the reverse ResNet-50, trained on Tiny-ImageNet. Showing the training, testing, and Grey Box (GB) adversarial (FGSM, ℓ_2: 2-8) accuracy on a scale between 0 and 1. The hyperparameters are given in-text.



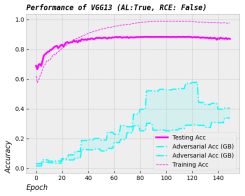
(g) Convergence graph for VGG-13, trained on CIFAR-10. Showing the training, testing, and Grey Box (GB) adversarial (FGSM, ℓ_2: 2-8) accuracy on a scale between 0 and 1. The hyperparameters are given in-text.



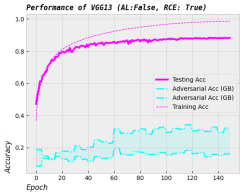
(h) Convergence graph for an adversarial learned VGG-13, trained on CIFAR-10. Showing the training, testing, and Grey Box (GB) adversarial (FGSM, ℓ_2: 2-8) accuracy on a scale between 0 and 1. The hyperparameters are given in-text.



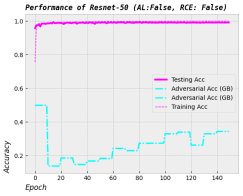
(g) Convergence graph for VGG-13, trained on CIFAR-10. Showing the training, testing, and Grey Box (GB) adversarial (FGSM, ℓ_2: 2-8) accuracy on a scale between 0 and 1. The hyperparameters are given in-text.



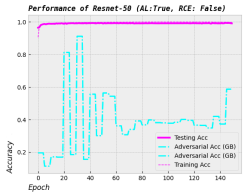
(h) Convergence graph for an adversarial learned VGG-13, trained on CIFAR-10. Showing the training, testing, and Grey Box (GB) adversarial (FGSM, ℓ_2: 2-8) accuracy on a scale between 0 and 1. The hyperparameters are given in-text.



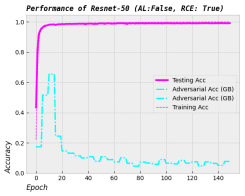
(i) Convergence graph for the reverse VGG-13, trained on CIFAR-10. Showing the training, testing, and Grey Box (GB) adversarial (FGSM, ℓ_2: 2-8) accuracy on a scale between 0 and 1. The hyperparameters are given in-text.



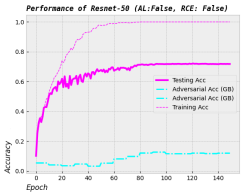
(j) Convergence graph for ResNet-50, trained on MNIST. Showing the training, testing, and Grey Box (GB) adversarial (FGSM, ℓ_2: 5) accuracy on a scale between 0 and 1. The hyperparameters are given in-text.



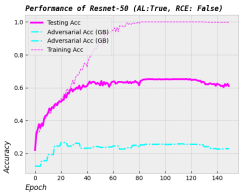
(k) Convergence graph for an adversarial learned ResNet-50, trained on MNIST. Showing the training, testing, and Grey Box (GB) adversarial (FGSM, ℓ_2: 5) accuracy on a scale between 0 and 1. The hyperparameters are given in-text.



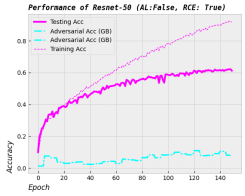
(l) Convergence graph for the reverse ResNet-50, trained on MNIST. Showing the training, testing, and Grey Box (GB) adversarial (FGSM, ℓ_2: 5) accuracy on a scale between 0 and 1. The hyperparameters are given in-text.



(m) Convergence graph for ResNet-50, trained on STL-10. Showing the training, testing, and Grey Box (GB) adversarial (FGSM, ℓ_2: 5) accuracy on a scale between 0 and 1. The hyperparameters are given in-text.



(n) Convergence graph for an adversarial learned ResNet-50, trained on STL-10. Showing the training, testing, and Grey Box (GB) adversarial (FGSM, ℓ_2: 5) accuracy on a scale between 0 and 1. The hyperparameters are given in-text.



(o) Convergence graph for the reverse ResNet-50, trained on STL-10. Showing the training, testing, and Grey Box (GB) adversarial (FGSM, ℓ_2: 5) accuracy on a scale between 0 and 1. The hyperparameters are given in-text.

Figure 2: Convergence graph for each evaluated model. Showing the training, testing, and Grey Box (GB) adversarial (FGSM, ℓ_2-radii between 2-8) accuracy on a scale between 0 and 1. The hyperparameters are given in-text.

B

Feature Extraction and Hyperparameters

This Appendix analyzes the feature engineering methods of Sec. 5.2 in more detail. This will support in-text claims about feature selection with additional empirical results. To this extent, we show the added value of selecting hidden units with larger, absolute Z-scores, higher normalized ℓ_2 scores on the last layers, and the feature separability of BRISQUE features. The Appendix ends with the hyperparameters that were used in our experiments.

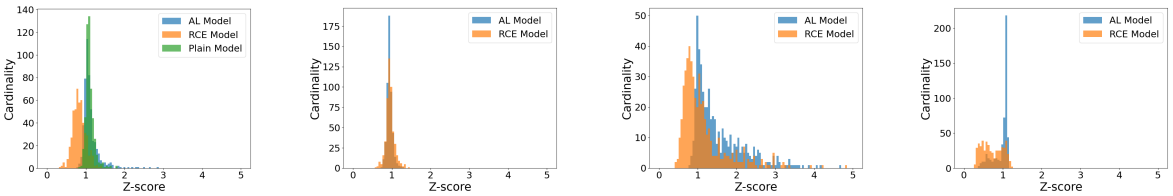


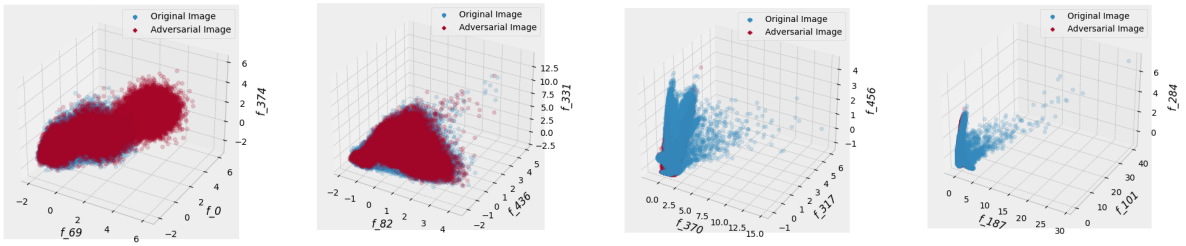
Figure B.1: Absolute Z-scores for FGSM evasions with a ℓ_2-radii of 5. Samples were from a random pool of 500 hidden units, directly after convolution. (**a**) The Z-scores when using a ResNet-50 on CIFAR-10 (**b**) The Z-scores when using a VGG-13 on CIFAR-10 (**c**) The Z-scores when using a ResNet-50 on MNIST (**d**) The Z-scores when using a ResNet-50 on STL-10.

B.1. Z-scores of the Hidden Units and other Features

To demonstrate the added value of our feature selection with utility $U_{\mathcal{P}}$, we plotted 500 Z-scores of nine models in Fig. B.1. The challenge with functions which employ the Euclidean distance lies in the curse of dimensionality. Whilst such functions may perform well in low-dimensional space, a rise in the number of dimensions exponentially increases the volume $V = r^d$ for some radius r, which causes data to be more sparse for similar dataset sizes.

The figure shows a large variance for the hidden units' utility, except for VGG-13. The first and third sub-figure presents a long tail for adversarial trained models. Reaching absolute Z-scores of 5, when the mean is around 1. This suggests that certain units more often show irregularities under perturbation than others. We also see an apparent shift for RCE in these two plots, having lower bins. The uniformity of the perturbation among the hidden units is thus lower.

We can confirm the observations again with the scatterplots in Fig. B.2. The first two plots show the dispersion of best- and worst-case hidden units for CIFAR-10 in terms of its Z-score. The former has far outliers on higher values for unit f_{69}. With this, the three features separate a large portion of the adversarial examples. The units with small Z-scores (Fig. B.2b) do not have such segregation, which demonstrates the diminishing significance of these features.



(a) (b) (c) (d)

Figure B.2: Activations' Z-Score of 3 hidden units before and after an FGSM perturbation, with a ℓ_2-radii of 5. The activations are sampled from a random pool of 500 hidden units, directly after convolution. (**a**) Activation of the hidden units with the highest Z-score for an Adversarial Learned ResNet-50 model on CIFAR-10 (**b**) Activation of the hidden units with the lowest Z-score for an Adversarial Learned ResNet-50 model on CIFAR-10 (**c**) Activation of the hidden units with the highest Z-score for a reverse ResNet-50 model on CIFAR-10 (**d**) Activation of the hidden units with the lowest Z-score for a reverse ResNet-50 model on CIFAR-10.

The last two plots shows a similar picture, but with more restricted perturbations due to the RCE loss. The two distributions are more distinct on dense regions of the benign data. Forming an enclave of adversarial samples on both plots, which is not ideal for density-based measures on benign data.

We did notice that such distribution, as the last plots, is useful for supervised methods. Reaching a detection accuracy of 0.984 and 0.884 with a Support Vector Machine on the units of the latter and former plots respectively. The hyperparameters are listed at the end of this Appendix.

B.1.1. Analyzing BRISQUE Features

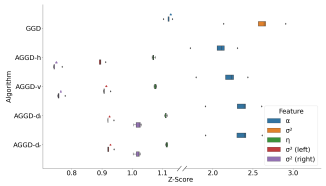
In Fig. B.3 we perform the same analysis on all 17 BRISQUE features. The MSCN variance emerges as a noteworthy indicator of adversarial examples. It consistently shows the largest absolute Z-scores. notable, is the increase of this feature's score for the MNIST dataset to a value 8 to 24, which is atypical for benign images.

Furthermore, the shape parameters of structural components distinguishes larger perturbations. Whereas, we see that relatively smaller perturbations on STL-10 yield stronger anomalies in terms of the skew of the AGGD distributions.

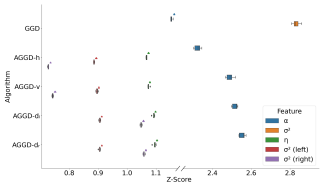
B.1.2. Predictive Uncertainty & Robust Optimization

To measure the importance of dropout in Feinman's KDE [Fei+17], we extract the linear relationship, *i.e.* weights, from the logistic regressor. An adversarial learned ResNet-50 on CIFAR-10 yields the coefficients -1.003 for the KDE estimate and 0.019 for dropout. This increases to -2.319 and 0.4530 for the reverse model. RCE is thus especially beneficial for dropout, as the weight's magnitude grows by a factor of 23.8.

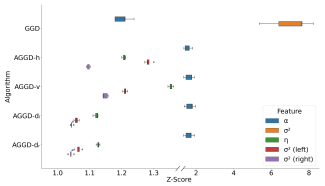
For STL-10, the coefficients change from -1.3488 to -2.0615 for the density estimate and 0.1014 to -0.1608 for dropout. Both datasets indicate the effectiveness of RCE for KDE and dropout, with dropout exhibiting a larger advantage, especially on CIFAR-10.



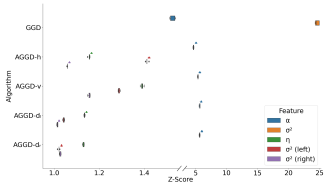
(a) Features extracted from CIFAR-10 training samples with an adversarial learned ResNet-50 as target model.



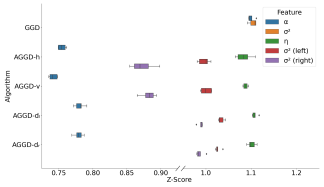
(b) Features extracted from CIFAR-10 training samples with a reverse ResNet-50 as target model.



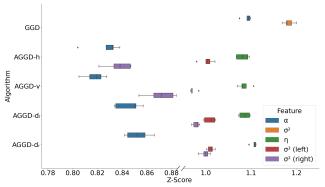
(c) Features extracted from MNIST training samples with an adversarial learned ResNet-50 as target model.



(d) Features extracted from MNIST training samples with a reverse ResNet-50 as target model.

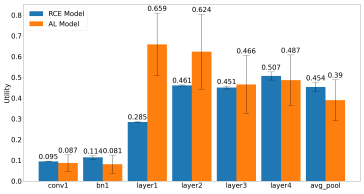


(e) Features extracted from STL-10 training samples with an adversarial learned ResNet-50 as target model.

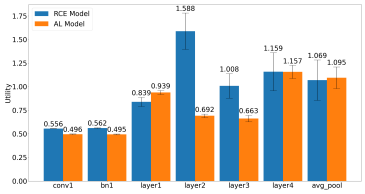


(f) Features extracted from STL-10 training samples with a reverse ResNet-50 as target model.

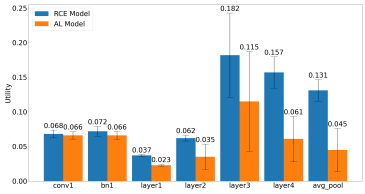
Figure B.3: Absolute Z-scores of all 17 BRISQUE features under a FGSM perturbation with a ℓ_2-radii of 5. BRISQUE includes 2 features from Generalized Gaussian Distribution (GDD) on the MSCN field and 4 for the Asymmetric GGDs (AGGD) on pairwise products of horizontal (h), vertical (v), and diagonal (d_l, d_r) neighbors. The results include 5 measurements of the features. Some colors are intentionally emphasized with a triangle on the top right. The features are in the same order (top-down) as described in Sec. 5.2.



(a) Distances for the reverse and adversarial learned model on CIFAR-10.



(b) Distances for the reverse and adversarial learned model on MNIST.



(c) Distances for the reverse and adversarial learned model on STL-10.

Figure B.4: The average, normalized Euclidean distance between FGSM (ℓ_2 of 5) and benign samples, $U_{f^{(l)}}$ in-text, at each ResNet-50 layer with 'layer#' denoting the bottlenecks. The error bars denote the 95% t-CI based on 5 runs.

B.2. Sensitivity and Utility of the Hidden Layers

Fig. B.4 explores the $U_{f^{(l)}}$ utility in the paper. The barplots show the utility at the outputs of that layer, which tends to increase when the perturbed sample traverses deeper layers; but this may come with fluctuations. For instance, the value's for the STL-10 models seem to decrease on the last layers. The adversarial learned CIFAR-10 model also decreases in utility after the first bottleneck. Still, the final bottleneck remains in the Top-3, suggesting its pivotal role in model performance and feature extraction. Conversely, the initial convolution, denoted as 'conv1', frequently exhibits the least utility.

We also see a major difference in the CI's critical region of the CIFAR-10 models. Specifically, the adversarial learned model displays notable variability on each layer, especially when compared to the RCE model. That is in line with the narrow activations in the scatterplots of Fig. B.2. The RCE models exhibit greater utilities in its deeper layers. Consequently, we anticipate enhanced performance when these hidden units are utilized for detection purposes. Otherwise, adversarial learning would become more intriguing due to the improved model accuracy (Tab. 6.2).

Table B.1: Hyperparameters used by default for each detection method, implementation is present in the supplementary material.

Algorithm	Parameters
Default	Used the best 10 features from a pool of 500, selected as discussed in-text. The Constant of Eq. 5.7 is set to 1. Utilized the cross entropy loss function. All generated adversarial examples are untargeted.
Shadow attack	Used 200 iterations, copy size of $(10, 1, 1, 1)$, batch size of 64, λ_{tv}=0.1, λ_{ch}=20, λ_{dissim}=10, σ=0.05
PerC attack	Used 200 iterations, batch size of 64, α_l=1, α_c=0.5
Logistic Regression	learned for 100000 epochs with learning rate 0.01
Support Vector Machine	learned for 100000 epochs with learning rate 0.1, early stopping and a scale γ of $(N * \sigma)^{-1}$ for the RBF network, N features, and variance σ. An SVM has been trained on BRISQUE features
Local Reachability Density	8 neighbors with a $t = 3$ iterations of feature bagging
Local Outlier Factor	8 neighbors with a $t = 3$ iterations of feature bagging
Local Intrinsic Dimensionality	8 neighbors without feature bagging
Kernel Density Estimation	50 (15 for white-box) iterations of dropout and a bandwidth chosen with Silverman's rule
Gausian Mixture Model	8 latent components with k-means initialization, learned till convergence
I-Defender	Trained 1 GMM with 50 hidden features were selected for CIFAR-10 and MNIST; 5 for STL-10; and 20 (RCE) / 30 (other models) for Tiny-ImageNet. This, with the discussed feature selection (Sec. 5.2)
Feature Squeezing	The squeezers used are: a bit depth reduction to 5 and a median pool with kernel 2×2 and padding 1
Whitening	used 500 low-rank PCA components: MNIST 284-784, CIFAR-10 2500-3000, Larger datasets 10000-10500
Principal Component Analysis (LFA)	used 200 last PCA components, in reduced form, rounded to hundreds: 800-1000 (ResNet-50)
Criterion 1 [Hu+19]	Gaussian noise sampled from $\mathcal{N}(0, 0.001)$.
Mahalanobis	ODIN perturbation ϵ of 0.002.

C

Experiments on Additional Datasets and Models

Table C.1: Accuracies of several detection algorithms against grey-, and white-box (GB/WB) adversaries on STL-10, MNIST, CIFAR-10 respectively top to bottom; with a ℓ_2-radii of 5. C&W attacks are evaluated on the detector's best performing robust optimization (AL/RCE) under DeepFool. DeepFool's and C&W's results are dependent on the error rate err_f of the target model (like in Tab. 4.1): VGG-13 on CIFAR-10 only and ResNet-50 on any dataset. Also, note that PerC-AL requires datasets to be in color. The top-3 results are **bolded**.

				Detection Accuracy ⇑						
Evasion Attack	Robust Optim.	Dataset	WB	**MS-8**	**I-Def**	**LRD (LFA)**	**PCA**	**KDE+BU**	**FSQ**	**MAH**
FGSM	✗	STL-10	✗	0.520	**0.549**	0.511	0.521	**0.542**	0.494	**0.546**
		MNIST		**0.823**	**0.814**	**0.829**	0.612	0.809	0.667	0.797
VGG-13:		CIFAR-10		**0.808**	0.569	0.690	**0.843**	**0.748**	0.594	0.682
FGSM	AL	STL-10	✗	0.485	**0.508**	**0.517**	**0.506**	0.498	0.497	0.505
		MNIST		0.897	0.894	**0.928**	0.614	**0.957**	0.828	**0.915**
VGG-13:		CIFAR-10		**0.942**	0.508	0.489	**0.941**	0.538	**0.676**	0.571
FGSM	RCE	STL-10	✗	**0.737**	0.567	0.528	0.531	0.571	**0.667**	**0.635**
		MNIST		**0.953**	0.935	**0.955**	0.678	**0.975**	0.915	0.659
VGG-13:		CIFAR-10		**0.956**	**0.606**	0.570	**0.949**	0.602	0.593	0.592
DeepFool	✗	STL-10	✗	**0.540**	0.495	0.507	0.500	**0.542**	0.481	**0.556**
		MNIST		**0.951**	0.877	0.726	0.522	0.840	**0.942**	**0.938**
VGG-13:		CIFAR-10		**0.640**	0.568	0.519	0.502	0.601	**0.894**	**0.756**
ResNet-50:		CIFAR-10		**0.729**	0.700	0.546	0.506	0.630	**0.900**	**0.801**
DeepFool	AL	STL-10	✗	**0.665**	0.502	0.509	0.500	0.503	**0.588**	**0.510**
		MNIST		**0.941**	0.880	0.744	0.645	0.829	**0.915**	**0.927**
VGG-13:		CIFAR-10		**0.686**	0.552	0.501	0.560	0.579	**0.857**	**0.650**
DeepFool	RCE	STL-10	✗	**0.612**	0.531	**0.647**	0.500	0.611	**0.634**	0.524
		MNIST		**0.953**	0.934	0.925	0.505	**0.960**	**0.947**	0.945
VGG-13:		CIFAR-10		0.657	0.633	0.546	0.567	**0.850**	**0.759**	**0.698**
C&W	Best perf.	STL-10	✗	0.517	0.518	0.512	0.500	**0.524**	**0.586**	**0.522**
		MNIST		**0.958**	0.924	0.894	0.609	0.866	**0.949**	**0.947**
VGG-13:		CIFAR-10		**0.803**	0.574	0.542	0.564	**0.652**	**0.870**	0.638
C&W	Best perf.	STL-10	✓	**0.619**	0.475	**0.507**	0.456	0.498	**0.547**	0.479
		MNIST		**0.989**	0.953	0.940	0.599	0.912	**0.986**	**0.988**
VGG-13:		CIFAR-10		**0.896**	0.492	0.536	**0.581**	0.521	**0.606**	0.485
PerC-AL[3]	RCE	STL-10	✗	**0.523**	0.503	0.503	**0.526**	0.516	**0.533**	0.505
		MNIST		N/A	N/A	N/A	N/A	N/A	N/A	N/A
VGG-13:		CIFAR-10		**0.815**	0.745	0.771	**0.795**	**0.782**	0.514	0.766
ResNet-50:		CIFAR-10		0.553	0.529	**0.560**	**0.612**	0.525	0.548	**0.667**
Shadow Attack[4]	RCE	STL-10	✗	0.587	0.479	**0.624**	0.450	**0.683**	**0.597**	0.464
		MNIST		**0.927**	**0.937**	0.887	0.453	**0.963**	0.449	0.833
VGG-13:		CIFAR-10		**0.665**	0.477	0.511	**0.698**	0.529	0.504	**0.663**
ResNet-50:		CIFAR-10		**0.653**	0.561	0.596	**0.716**	0.485	0.513	**0.619**

[3] The RCE Model's Top-1 Accuracies under the PerC-AL attack are (Top-Down):
→ 0.569 (STL-10); - (MNIST); 0.317 (VGG-13,CIFAR-10); 0.631 (ResNet-50,CIFAR-10).

[4] The RCE Model's Top-1 Accuracies under Shadow Attack are (Top-Down):
→ 0.124 (STL-10); 0.078 (MNIST); 0.122 (VGG-13,CIFAR-10); 0.137 (ResNet-50,CIFAR-10).

Table C.2: Accuracies of GMM-based detectors against grey-box adversaries on Tiny-ImageNet, STL-10 and CIFAR-10, with a ℓ_2-radii of 5. Demonstrating the efficacy of small-scale models in comparison to datasets containing images of higher resolution. DeepFool's and C&W's results are dependent on the error rate err_f of the models, like in Tab. 4.1. Best results are **bolded**.

Evasion Attack	Model	Dataset	Robust Optim. →	Detection Accuracy ⇑					
				Plain Model		AL Model		RCE Model	
			WB ↓	**MS-8**	**I-Def**	**MS-8**	**I-Def**	**MS-8**	**I-Def**
FGSM	ResNet-50	Tiny-ImageNet	✗	0.954	0.532	0.939	0.533	**0.976**	0.503
C&W	VGG-13	CIFAR-10	✗	0.790	0.593	**0.803**	0.509	0.689	0.574
C&W	ResNet-50	STL-10	✗	0.531	0.499	0.517	0.498	**0.574**	0.518
C&W	ResNet-50	Tiny-ImageNet	✗	**0.553**	0.494	0.523	0.497	0.502	0.511

D

Implementation Details

Algorithm 4 MeetSafe

Require
- $\mathfrak{D}$ Dataset of benign samples
- $\mathcal{P}_{max}$ Maximum amount of features
- K, τ_{90} Latent components and threshold
- $f^{(l)}$ Basis function of the neural network
- $\mathbf{X'}$ Suspicious samples

Ensure
- $M_{\mathbf{X'}}$ MeetSafe classifications

$U_{f^{(l)}} \leftarrow \mathbf{0}$
for $X \in \mathfrak{D}$ and basic block l **do** ▷ Get the utility of each layer (optional)
  $X' \leftarrow X + \epsilon \cdot \textit{sign}(\nabla_X \mathcal{L}_f)$
  $U_{f^{(l)}} \leftarrow Sequential\_Avg(U_{f^{(l)}}, f^{(l)}_{X'}, f^{(l)}_{X})$ ▷ Eq. 5.6
end for

Let $\{l\}$ have the largest $U_{f^{(l)}}$ value and let $\{\mathcal{P}\} \subseteq_R f^{(l)}$.
for $X \in \mathfrak{D}$ **do** ▷ Get the utility of each hidden unit in $\mathcal{P}$
  $X' \leftarrow X + \epsilon \cdot \textit{sign}(\nabla_X \mathcal{L}_f)$
  $\mathcal{P}_X \leftarrow f^{(l)}_X$ where $\{f^{(l)}\} \in \mathcal{P}$
  $\mathcal{P}_{X'} \leftarrow f^{(l)}_{X'}$ where $\{f^{(l)}\} \in \mathcal{P}$
end for
$U_{\mathcal{P}} \leftarrow \frac{1}{|\mathfrak{D}|} \sum_{X \in \mathfrak{D}} \frac{|\mathcal{P}_X - \mathcal{P}_{X'}|}{\Sigma_{\mathcal{P}}}$ ▷ Eq. 5.5
$\mathcal{P} \leftarrow top\_n(\mathcal{P}_{max}, \mathcal{P})$ ▷ Pick the best hidden units

for $X \in \mathfrak{D}$ **do** ▷ Initialize the GMM
  Initialize $\{\mu_i\} \in \mathbb{R}$ via the K-means algorithm and $\{\pi_i, \Sigma_i\} \in \mathbb{R}, \mathbb{R}^{3\times3}$ uniformly at random.
  $\mathcal{H}_X \leftarrow Extract\_Features(\mathfrak{D}, \mathcal{P}, \mathcal{P}_{max}, X)$ ▷ Algorithm 2
end for
$\mu_i, \Sigma_i, \pi_i \leftarrow EM(\mu_i, \Sigma_i, \pi_i, \mathcal{H}_X) \quad i \in [0..K), \ \forall\, X \in \mathfrak{D}$

for $X' \in \mathbf{X'}$ **do** ▷ Classify samples
  $\mathcal{H}_{X'} \leftarrow Extract\_Features(\mathfrak{D}, \mathcal{P}, \mathcal{P}_{max}, X')$
end for
return $-\log\left\{\sum^{K} \pi_i \cdot \mathcal{N}(\mathcal{H}_{X'} | \mu_i, \Sigma_i)\right\} > \tau_{90} \ \ \forall\, X' \in \mathbf{X'}$ ▷ Eq. 5.9

Algorithm 5 MeetSafe's Feature Extraction

Require
- $\mathfrak{D}$ Dataset of benign samples
- $\mathcal{P}$ Set of basis functions
- $\mathcal{P}_{max}$ Maximum amount of features
- X A benign or adversarial sample

Ensure
- $\mathcal{H}_{\mathcal{X}}$ X's Features

if kNN or SVD is not initialized **then** ▷ Prepare heuristics
  $kNN \leftarrow Prepare\_kNN(f_{\mathfrak{D}}^{(l)})$ where $\{f^{(l)}\} \in \mathcal{P}$
  $\mathbf{U_t}, \mathbf{S_t}, {\mathbf{V_t}}^T \leftarrow SVD(\mathfrak{D})$ ▷ Truncated to a predefined range of 500 components
  for $X \in \mathfrak{D}$ **do**
    $X' \leftarrow X + \epsilon \cdot \textit{sign}(\nabla_X \mathcal{L}_f)$
    ${\mathcal{P}^*}_X \leftarrow X\mathbf{V_t}$
    ${\mathcal{P}^*}_{X'} \leftarrow X'\mathbf{V_t}$
  end for
  $U_{\mathcal{P}^*} \leftarrow \frac{1}{|\mathfrak{D}|} \sum_{X \in \mathfrak{D}} \frac{|{\mathcal{P}^*}_X - {\mathcal{P}^*}_{X'}|}{\Sigma_{\mathcal{P}^*}}$ ▷ Eq. 5.5
  $\mathbf{V_t} \leftarrow top\_n(\mathcal{P}_{max}, P^*)$ ▷ Pick the best eigenvectors
end if

$\Omega_{X,k} \leftarrow kNN(f_X^{(l)})$ where $\{f^{(l)}\} \in \mathcal{P}$
$\Sigma \leftarrow Diag(\sigma_{\mathcal{P}})$
$H_0 \leftarrow \frac{1}{LRD(f_X^{(l)}, \Omega_{X,k}, \Sigma)}$ where $\{f^{(l)}\} \in \mathcal{P}$ ▷ Eq. 5.3-5.4

$X_{color}, X_{blur} \leftarrow Reduce\_Bit\_Depth(X),\ Blur(X)$
$H_1 \leftarrow \max\left\{||F(X) - F(X_{color})||_1, ||F(X) - F(X_{blur})||_1\right\}$ ▷ Feature Squeezing [XEQ18]

$H_2 \leftarrow Var(X\mathbf{V_t})$ ▷ Whitening [HG17]
return $\{H_0, H_1, H_2\}$

E

Table of Notations

Table E.1: Table of Notations.

Notation	Description
$\mathbf{X}, X$	A batch of samples, a single sample
$\mathbf{X}', X'$	An adversarial batch, adversarial example
$X(i, j)$	Indexing of an image to row i and column j
$\mathcal{P}$	Pool of features (*e.g.* hidden units or eigenvectors)
$\mathfrak{D}, \mathfrak{I}$	Dataset & input space of benign samples
m, L	input-, output size
$\mathcal{B}$	The chosen batch size
$\mathcal{H}_{\mathfrak{D}}$	Feature set of dataset $\mathfrak{D}$
$U(\cdot), U.$	Utility function or current utility for a given input
$\mathcal{R}^*, \mathcal{R}$	Empirical risk of a classifier with and without adversarial perturbations
$\mathcal{A}(\cdot)$	Adversarial objective given his/her intention, capability, and strategy
θ, Π	Adversarial knowledge defined as the space θ, which may include the implemented defense protocol Π
$\mathcal{K}$	The Lipschitz constant
γ_c, r_c	Deviation angle and ratio of a boundary $\mathcal{C}$ *w.r.t.* the bisecting boundary $\mathcal{B}$
$\mathcal{C}_\phi$	A set of linear constraints that encodes the feature map ϕ
$f^{(l)}(\cdot)$	Basis functions of a deep neural network at layer l
$h(\cdot)$	An activation function, like ReLU, Sigmoid etc.
$f_X^{(l)}$	The activation of hidden units for the classifier $f^{(l)}(X)$, sample X, and layer l. Possibly conditioned with its weights W and bias B
$z_i \in Z(\cdot), y_i \in F(\cdot)$	Logits and probits of a deep neural network
$D^{(l)}$	The dimension of the (hidden) layer l
$\mathcal{L}_f(\cdot)$	Loss function of classifier $f^{(l)}(X)$
$\nabla_X \mathcal{L}_f$	Gradient for the current model *w.r.t.* loss $\mathcal{L}_f(\cdot)$
H_X	Hessian for the variables in X
u_i, λ_i	Eigenvectors, and -values
B	Positive definite estimate of the Hessian
δ, δ_ϕ	Adversarial perturbation generated by an evasion method. δ_ϕ denotes only the effective perturbation towards the inter-class boundary of feature map ϕ
ϵ	scaler for an (adversarial) perturbation to control its ℓ_p-radii
κ	Confidence parameter of the C&W attack, regulating the strength of adversarial samples
$\lambda_{tv}, \lambda_{ch}, \lambda_{dissim}$	Scalars for the shadow attack, regulating the total variation $TV(\delta)$, color channels $C(\delta)$, and similarity on each color channel $Dissim(\delta)$. Given the perturbation δ
ΔE_{00}	CIEDE2000 color distance using chroma $\Delta C'$, lightness $\Delta L'$, and hue $\Delta H'$ attributes
$\mu_{\mathcal{P}}, \sigma^2_{\mathcal{P}}$	Mean and variance given the data in $\mathcal{P}$
π_i	Mixing coefficient for the i'th latent variable of a Gaussian Mixture
$\mathcal{N}(\mu, \Sigma)$	Guassian with mean μ and covariance Σ
$\mathcal{N}(\cdot\|\mu, \Sigma)$	Density function of the Guassian
ξ_i	A latent variable modeled by the Gaussian Mixture
$\gamma(\cdot)$	Sum of responsibilities $\sum^K \hat{p}(\xi_i\|\mathcal{H}_X)$ for K Gaussians
$\hat{p}(\cdot)$	Estimate of the distribution $p(\cdot)$
α	Shape parameter of a GGD (unless stated otherwise)
η	Skewness (GGDs), learning rate, or overshoot (DeepFool)
τ	Threshold or critical value, used to regulate the conspicuousness of a certain sample and for a given metric
$\|\|\cdot\|\|_p, d'_{\cdot,k}, (\cdot)^+$	ℓ_p, k-distance, and ReLU
$\mathbb{1}_A$	Indicator function of A
$\mathbb{E}_f[\cdot]$	The expected value of the function $f(X)$
$\Gamma(\cdot)$	The Gamma function
$\Omega_{X,k}$	Set of nearest neighbors for X
$\mathcal{M}$	A submanifold of sampled data
$\mathcal{O}(\cdot), \Omega(\cdot)$	Big O & Big Omega notation

Index