INAUGURAL-DISSERTATION

submitted to the

Combined Faculty of Mathematics, Engineering and Natural Sciences

of the

Ruprecht-Karls-University

Heidelberg

for the degree of

Doctor of Natural Sciences

Put forward by

M.Sc. Peter Lorenz

Born in: Bad Radkersburg, Styria, Austria

Oral examination:

Analysis of Adversarial Examples

Advisor: Prof. Dr. Ullrich Köthe

Abstract

The rise of artificial intelligence (AI) has significantly impacted the field of computer vision (CV). In particular, deep learning (DL) has advanced the development of algorithms that comprehend visual data. In specific tasks, DL exhibits human capabilities and is impacting our everyday lives such as virtual assistants, entertainment or web searches. Despite of the success of visual algorithms, in this thesis we study the threat adversarial examples, which are input manipulation to let to misclassifcation.

The human vision system is not impaired and can classify the correct image, while for a DL classifier one pixel change is enough for misclassification. This is a misalignment between the human and CV system. Therefore, we start this work by presenting the concept of an classification model to understand how these models can be tricked by the threat model – adversarial examples.

Then, we analyze the adversarial examples in the Fourier domain, because after this transformation they can be better identified for detection. To that end, we assess different adversarial attacks on various classification models and datasets deviating from the standard benchmarks

As a complementary approach, we developed an anti-pattern utilizing a frame-like patch (prompt) on the input image to counteract the input manipulation. Instead of merely identifying and discarding adversarial inputs, this prompt neutralizes adversarial perturbations during testing.

As another detection method, we expanded the use of a characteristics of multi-dimensional data – the local intrinsic dimensionality (LID) to differentiate between benign and attacked images, improving detection rates of adversarial examples.

Recent advances in diffusion models (DMs) have significantly improved the robustness of adversarial models. Although DMs are well-known for their generative abilities, it remains unclear whether adversarial examples are part of the learned distribution of the DM. To address this gap, we propose a methodology that aims to determine whether adversarial examples are within the distribution of the learned manifold of the DM. We present an exploration of transforming adversarial images using the DM, which can reveal the attacked images.

Zusammenfassung

Der Aufstieg der künstlichen Intelligenz (KI) hat den Bereich der Computer Vision (CV) erheblich beeinflusst. Insbesondere das Deep Learning (DL) hat die Entwicklung von Algorithmen zum Verstehen visueller Daten vorangetrieben. Bei bestimmten Aufgaben zeigt DL (über)menschliche Fähigkeiten und wirkt sich auf unser tägliches Leben aus, z.B. bei virtuellen Assistenten, in der Unterhaltungsbranche oder bei der Websuche.

Trotz des Erfolgs der visuellen Algorithmen untersuchen wir in dieser Arbeit die Bedrohung: Feindliche Beispiele, die das Bild manipulieren, um eine bewusste Fehlklassifizierung zu ermöglichen. Das menschliche Sehsystem ist im Falle von feindlichen Beispielen nicht beeinträchtigt und kann das Bild richtig wahrnehmen, während für einen DL-Klassifikator eine Pixeländerung für eine Fehlklassifizierung ausreicht. Dies ist eine Unstimmigkeit zwischen dem menschlichen und dem maschinellen Sehen. Daher beginnen wir diese Arbeit mit der Vorstellung des Konzepts eines Klassifizierungsmodells, um zu verstehen, wie diese Modelle überlistet werden können.

Anschließend analysieren wir die feindliche Beispiele in der Fourier-Domäne, da sie nach dieser Transformation für die Erkennung besser identifiziert werden können. Zu diesem Zweck bewerten wir verschiedene Angriffe auf verschiedene Klassifizierungsmodelle und Datensätze, welche von den Standardevaluierungen abweichen.

Als weiteren Ansatz haben wir ein Anti-Muster entwickelt, das einen rahmenähnlichen Überlagerung (Prompt) auf dem Eingabebild verwendet, um der Manipulation der Eingabe entgegenzuwirken. Dieser Prompt soll feindliche Eingaben neutralisieren.

Als weitere Erkennungsmethode haben wir unter der Verwendung einer Charakterististik von mehrdimensionale Daten - der lokalen intrinsischen Dimensionalität (LID) - erweitert, um zwischen gutartigen und angegriffenen Bildern zu unterscheiden, was zur Verbesserung der Erkennungsraten für feindliche Beispiele führt.

Neue Fortschritte in Diffusionsmodellen (DMs) haben die Robustheit gegen Angriffsmodellen erheblich verbessert. Obwohl DMs bekannt sind für ihre generativen Fähigkeiten, ist unklar, ob angegriffenen Bilder Teil der gelernten Verteilung sind. Um diese Lücke zu schließen, schlagen wir eine Methodik vor, um zu bestimmen, ob feindliche Beispiele innerhalb der gelernten Verteilung von DM liegen.

Acknowledgements

I would like to thank my advisor Prof. Dr. Ullrich Köthe and my co-advisor Prof. Dr.-Ing. Janis Keuper for their support and tutelage during my journey. I am also grateful to Prof. Dr.-Ing. Margret Keuper for her valuable feedback and suggestions that have helped me improve my work. At this point, I also want to thank Asst. Prof. Dr. Sijia Liu and Dr. Pin-Yu Chen for the internship opportunity.
I would like to thank my colleagues and friends for their support and camaraderie, especially Ricard, but also Aochuan, $2\times$ Dominik, Jens, Matthias, and Paula. I am grateful for the stimulating discussions, collaborations, and friendships that have made my time so fruitful.
Finally, I would like to thank my family for their unwavering love and support. Their encouragement and belief in me have been a constant source of strength throughout this journey.

– Peter

Contents

Glossary of Terms

Artifical intelligence (AI) simulates human intelligence processes using machines, particularly computer systems. It involves creating algorithms for classification, analysis, prediction, and learning from data. AI machines process real-time data and perform problem-solving operations.

Computer vision (CV) is a subcategory of AI that enables computers and systems to derive meaningful information from digital images, videos, and other visual inputs, and take actions or make recommendations based on that information.

Machine learning (ML) is a discipline of artificial intelligence that enables systems to learn and improve from experience without being explicitly programmed.

Neural networks (NN) also known as artificial neural networks (ANNs), are computational models inspired by the human brain's biological neural networks. They consist of interconnected nodes (artificial neurons) organized in layers. During training, the network adjusts connection weights to learn and improve.

Deep learning (DL) is a subfield of machine learning that involves the use of neural networks with multiple layers (deep neural networks). The term "deep" refers to the depth of the network, which is characterized by having many hidden layers between the input and output layers. Deep learning has proven effective in learning hierarchical representations and patterns from complex data.

Adversarial example is an input (e.g. a manipulated image) designed to cause a machine learning model to make a wrong prediction. It is generated from a clean example by adding a small perturbation, imperceptible for humans, but sensitive enough for the model to change its prediction.

Adversarial machine learning focuses on attacks against machine learning algorithms and the corresponding defenses. An adversarial attack generates examples to deceive deep learning models.

Whitebox (WB) scenarios grant attackers complete access to the target model, its architecture and parameters. For instance, the Projected Gradient Descent (PGD) attack is one prominent gradient-based attack method.

Blackbox (BB) is a scenario where an attacker has limited access to the target model and can only observe the outputs without insight into its internal mechanisms.

Adversarial defense aims to protect a deep learning model by reducing the attack surface against adversarial attacks. For instance, adversarial training, in which a network is hardened against adversarial examples.

Adversarial robustness in machine learning refers to a model's capability to maintain performance and accuracy despite being exposed to intentionally crafted adversarial inputs.

Classification in machine learning refers to a supervised learning task where the goal is to assign input data points to predefined categories or classes. This process entails training a model on labeled data to learn patterns and relationships, enabling it to predict outcomes for new, unseen instances.

Detection in computer vision is the process of identifying and locating objects or patterns within images or videos. It involves training models to recognize specific classes of objects and then predicting their presence and positions in new, unseen data.

Identification in computer vision refers to the process of identifying and locating objects or patterns within images or videos. It entails training models to recognize specific classes of objects and subsequently predicting their presence and positions in new, unseen data.

On the other hand, identification goes beyond mere detection. It involves not only locating objects or regions within an image but also assigning specific labels or categories to those objects.

Diffusion models (DMs) belong to a category of generative models employed in machine learning. Their purpose is to generate new data based on the training data they have encountered. Specifically, diffusion models learn a process that generates probabilities, often represented by either joint rotations or positions. These models find applications in various domains, including text-to-video synthesis, image-to-image translation, image search, and reverse engineering.

Explainability in machine learning pertains to a model's capacity to offer a clear and understandable explanation for its predictions or decisions. It encompasses the extent to which humans can interpret the model's internal workings and decision-making process. Explainability is crucial for building trust, identifying biases, rectifying errors, and enhancing model performance.

Interpretability refers to a model's ability to provide insight into its internal processes. It involves explaining how the model processes input data, learns from it, and makes predictions or decisions based on that learning. Interpretability allows us to assess the model's inner workings and ensure it relies on relevant features. The key distinction lies in the focus: explainability emphasizes the model's output, while interpretability delves into its internal mechanisms.

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Computer Vision and Artificial Intelligence

Computer vision (CV), situated within the broader domain of artificial intelligence (AI), empowers machines – ranging from computers and mobile phones to cameras – to discern meaningful insights from diverse visual digital content such as images and videos. This ability enables machines to take informed actions based on their visual perception. While AI imparts cognitive capabilities to systems, CV specifically equips them with visual discernment, aiming to develop algorithms that enable artificial entities to understand intricate details in visual data, akin to human vision, thus inferring contextual information about their surroundings.

When discussing human vision, the primary focus naturally turns to the eyes, the sensory organs responsible for detecting light and transmitting signals containing visual information along the optic nerve to the brain. The brain, serving as the central processing unit, then analyzes and retains the input content. It is essential to recognize that this constitutes a continuous process marked by a gradual and steady learning trajectory.

CV operates in a manner that closely mirrors human vision, albeit with a notably truncated history. Consequently, the artificial solutions within this domain are constrained by temporal limitations. To address this inherent constraint, CV systems are endowed with a suite of tools encompassing cameras, data, and algorithms, supplanting the biological components of retinas, optic nerves, and the visual cortex. This substitution empowers these systems to significantly expedite the learning process. The fig. 1.1 clarifies the visual representation of the schematic progression of human vision, faced with its artificial equivalent within the realm of CV.

Continuously emerging and evolving technologies are consistently adapting to address contemporary and impending challenges. Among these, machine learning – particularly deep learning (DL) – has emerged as a predominant force in numerous CV endeavors, en-

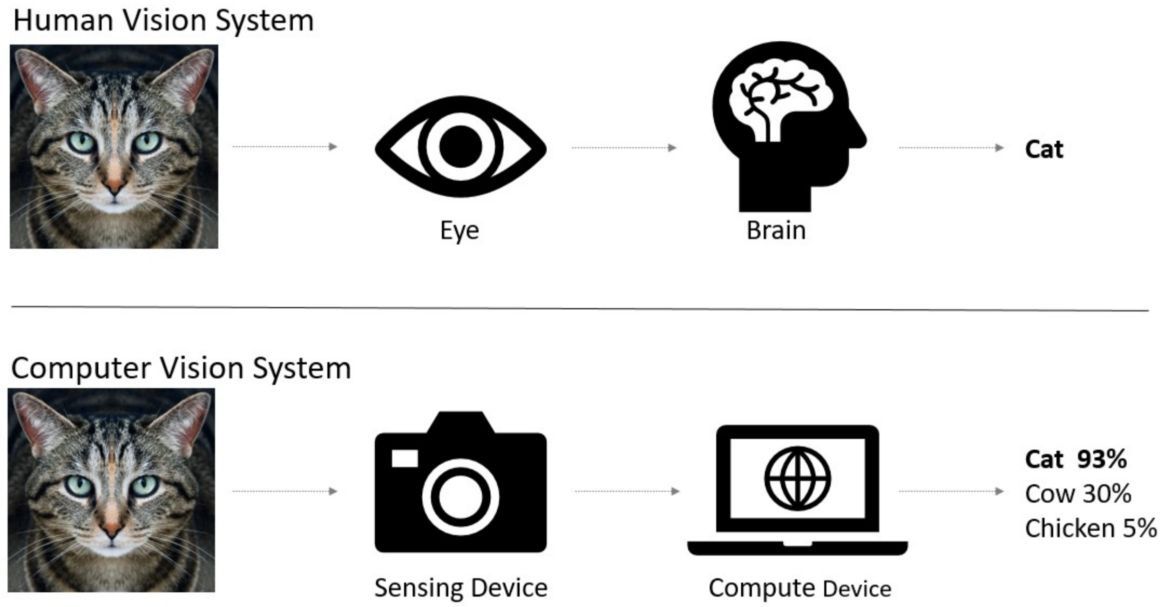


Figure 1.1. Schematic representation of the human vision and computer vision (CV) pipelines. Nature has always served as a source of inspiration for humanity. In the CV community, scientists draw inspiration from the various elements of human vision to construct an artificial equivalent. A machine can calculate the probabilities of the object on the image, while for humans they know from experience.

compassing tasks such as image classification, object detection, image generation, and semantic segmentation. The rationale behind its remarkable achievements lies in the fact that deep learning methodologies embrace self-contained, iterative learning processes that discern specific input patterns, in stark contrast to traditional hard-coded programming. Consequently, the present-day emphasis on automated feature extraction has led to significant breakthroughs, leading influence not solely within the realm of vision, but extending to various other domains as well.

The term “feature extraction” alludes to the process undertaken by an artificial neural network, wherein it is tasked with deriving task-specific indicators from provided data while adhering to the constraints associated with a particular problem. In historical context, the discovery of these indicators or traits has been a laborious endeavor, necessitating manual intervention from machine learning (ML) practitioners. Although DL approaches have eclipsed preceding methodologies in this domain, they are not devoid of shortcomings, which warrant consideration during their application. Notably, they often grapple with data insufficiency due to their dependence on extensive volumes of labeled data. To mitigate this limitation, techniques like data augmentation, algorithms demanding less data, or structures with inherent biases have been employed.

Nonetheless, this remains an ongoing research challenge that continues to engage the attention of researchers. As previously mentioned, DL has outperformed conventional manually designed CV algorithms in overall efficiency across nearly all types of data, thereby enabling data science teams to redirect their efforts toward more substantive tasks such as spam, surveillance, automatic face recognition, or autonomous driving. Some of these tasks rely on safety critical application and the question arises, if these ML systems are robust and trustworthy. For example an autonomous car does not recognize a stop car and continues driving. Besides robustifying existing DL models, images/videos can be authentic generated.

1.2 Classification

Classification is a fundamental concept in machine learning (ML), referring to the process of categorizing or labeling data into distinct classes or categories based on their characteristics or features. The primary goal of classification is to train a model that can automatically assign new, unseen data points to one of these predefined classes.

In a precise definition, we will establish the model (also known as the hypothesis function) denoted as $f_\theta : \mathcal{X} \rightarrow \mathbb{R}^k$. This function acts as a mapping from the input space to the output space, represented as a k-dimensional vector. It is important to note that k represents the number of classes considered in the prediction task. It is worth mentioning that the output is in the logit space, indicating that the values are real numbers that can be positive or negative. The $\boldsymbol{\theta}$ vector encompasses all the parameters that determine the structure of this model, such as convolutional filters or fully-connected layers.

In the next step, we introduce a loss function denoted as $\ell : \mathbb{R}^k \times \mathcal{Z}_+ \rightarrow \mathbb{R}_+$. This function maps the model's predictions and the true labels, producing a non-negative numerical value. The interpretation of this loss function is as follows: the first parameter represents the model's output in the form of logits, which can be positive or negative values, while the second parameter indicates the true class index. In other words, it represents a number between 1 and k that identifies the accurate label's index. Therefore, the symbolic expression

$$\ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y) \tag{1.1}$$

where $\boldsymbol{x} \in \mathcal{X}$ represents the input and $y \in \mathcal{Z}$ denotes the true class, serves as a representation of the loss incurred by the classifier in its predictions concerning $\boldsymbol{x}$, given the presumption that the true class is y.

The cross entropy loss, often referred to as the softmax loss, is the most widely employed loss function in deep learning. It quantifies the disparity between two probability distributions

$$\ell\left(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y\right) = \log\left(\sum_{j=1}^{k} \exp\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})_j\right)\right) - f_{\boldsymbol{\theta}}(\boldsymbol{x})_y, \tag{1.2}$$

where $f_{\boldsymbol{\theta}}(\boldsymbol{x})_j$ denotes the j-th elements of the vector $f_{\boldsymbol{\theta}}(\boldsymbol{x})$. In assessing the effectiveness of a classification model with probability outputs ranging from 0 to 1, the cross entropy loss is employed. As the predicted probability deviates further from the true label, this loss metric increases. The structure of this loss function is derived from the commonly used softmax activation. It involves defining the softmax operator $\sigma : \mathbb{R}^k \to \mathbb{R}^k$, which operates on a vector $\boldsymbol{z}$ as follows:

$$\sigma(\boldsymbol{z})_i = \frac{\exp(\boldsymbol{z}_i)}{\sum_{j=1}^{k} \exp(\boldsymbol{z}_j)}. \tag{1.3}$$

This operator transforms the class logits generated by the model into a probability distribution. When training a neural network, the primary objective is typically to maximize the likelihood of accurately predicting the true class label. However, because probabilities can become extremely small, it is more customary to maximize the natural logarithm of the probability associated with the true class label, as expressed by the formula:

$$\log \sigma\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)_y = \log\left(\frac{\exp\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})_y\right)}{\sum_{j=1}^{k} \exp\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})_j\right)}\right) = f_{\boldsymbol{\theta}}(\boldsymbol{x})_y - \log\left(\sum_{j=1}^{k} \exp\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})_j\right)\right). \tag{1.4}$$

Since the usual practice is to minimize loss rather than maximize probability, the negative form of this quantity is utilized as the loss function.

1.3 Threat Model: Adversarial Examples

Machine learning (ML) leaves the labs and immerse in everyday life. This premise provides malicious attackers with opportunities to exploit various cyber attacks in order to compromise the ML software and its operations. In this section, we introduce the threat model: adversarial examples, which minimal manipulate an image for misclassification.

As in computer vision (CV) captures the real world, we endeavor to to implement ML systems not merely on virtual domains, but also in real systems. It becomes crucial that we evaluate not only the systems' functionality under typical circumstances but also their capacity to consistently demonstrate genuine robustness and reliability. Adversarial robustness has become a prominent area of interest when it comes to the concepts of resilience and trustworthiness. This field concentrates on creating classifiers that can endure input disturbances during testing, even in the presence of adversaries aiming to deceive the classifier. Therefore, in this section, we introduce the treat model "adversarial examples" and how to create them to fool a classifier. Then, we briefly introduce a selection of whitebox adversarial attacks, which has been proposed from the year 2014 until today.

Creating an adversarial example. To manipulate an image and deceive the classifier into misclassifying it, we can create an adversarial example. In the common approach to training a classifier, the parameters $\boldsymbol{\theta}$ are optimized to minimize the average loss over a training set $\{\boldsymbol{x}_i \in \mathcal{X}, y_i \in \mathbb{Z}\}$, which can be formulated as the optimization problem:

$$\underset{\boldsymbol{\theta}}{\text{minimize}} \frac{1}{m} \sum_{i=1}^{m} \ell\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right), y_i\right).$$

This optimization problem is typically solved using (stochastic) gradient descent. For a minibatch $\mathcal{B} \subseteq \{1, \ldots, m\}$, the gradient $\nabla_{\boldsymbol{\theta}} \ell\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right), y_i\right)$ is computed with respect to the parameters, and the parameters are adjusted in the negative direction of the gradient:

$$\boldsymbol{\theta} := \boldsymbol{\theta} - \frac{\alpha}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla_{\boldsymbol{\theta}} \ell\left(f_{\boldsymbol{\theta}}\left(\boldsymbol{x}_i\right), y_i\right),$$

where α is the step size. This process is repeated for different minibatches until the parameters converge.

The gradient is a key term of interest as it quantifies how small adjustments to the parameters affect the loss function. In deep neural networks, the gradient is efficiently computed using backpropagation. The beauty of automatic differentiation, which underlies backpropagation, is that we can also compute the gradient of the loss with respect to the input image itself. This gradient reveals how small changes to the image impact the loss function.

To construct an adversarial example, our objective is to adjust the image to maximize the loss instead of minimizing it as in parameter optimization. We aim to solve the optimization problem:

$$\underset{\hat{\boldsymbol{x}}}{\text{maximize}}\ \ell\left(f_{\boldsymbol{\theta}}(\hat{\boldsymbol{x}}), y\right),$$

where $\hat{\boldsymbol{x}}$ represents the adversarial example. However, we cannot optimize arbitrarily over $\hat{\boldsymbol{x}}$ since some images are not the target class. Thus, we need to ensure that the adversarial example, denoted as $\hat{\boldsymbol{x}}$, remains close to the original input $\boldsymbol{x}$. Conventionally, we achieve this by optimizing over the perturbation added to $\boldsymbol{x}$, denoted as $\boldsymbol{\delta}$, and then optimizing over $\boldsymbol{\delta}$:

$$\underset{\boldsymbol{\delta} \in \Delta}{\text{maximize}}\ \ell\left(f_{\boldsymbol{\theta}}(\boldsymbol{x} + \boldsymbol{\delta}), y\right).$$

Here, Δ represents the set of allowable perturbations. Determining the "correct" set of allowable perturbations is challenging, as we ideally want Δ to include variations that humans perceive as visually similar to the original input $\boldsymbol{x}$. These variations can range from slight amounts of noise to rotations, translations, scalings, or even complete changes in non-target regions of the image.

In summary, to create an adversarial example, we optimize the perturbation to the original image to maximize the loss while ensuring the manipulated image remains visually similar to the original. The set of allowable perturbations should encompass variations that preserve visual similarity. Needless to say, it is not possible to give a mathematically rigorous definition of all the perturbations that should be allowed, but the philosophy behind

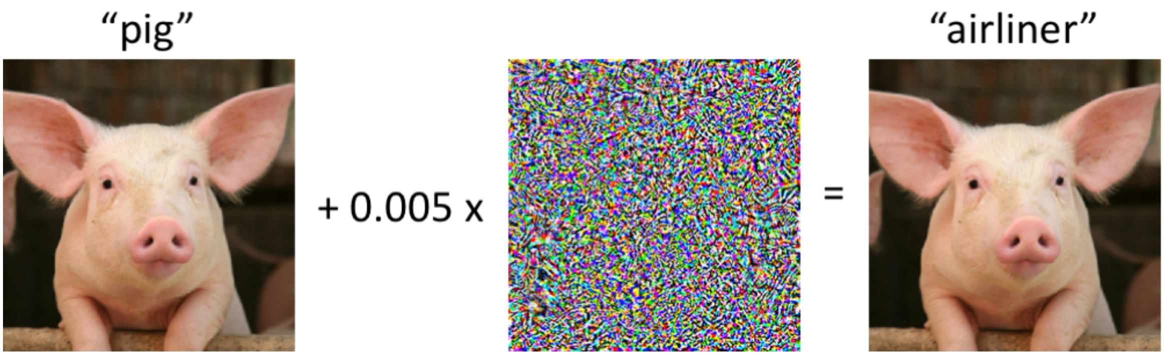


Figure 1.2. Example of adversarial perturbation used to evade classifiers [46]. The perturbation $\boldsymbol{\delta}$ is added to the "pig" image. In this case, the perturbation is just magnified for illustration. Figure from Madry and Schmidt [93].

adversarial examples is that we can consider some subset of the possible space of allowed perturbations, such that by any "reasonable" definition, the actual semantic content of the image could not change under this perturbation. A commonly used perturbation set, although not the only valid choice, is the L^∞ ball, defined as the set:

$$\Delta : \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\|_\infty \leq \epsilon\}, \tag{1.5}$$

where the L^∞ norm of a vector $\boldsymbol{z}$ is given by:

$$\|\boldsymbol{\delta}\|_\infty = \max_i |\boldsymbol{z}_i|. \tag{1.6}$$

In this case, we allow the perturbation to have a magnitude between $[-\epsilon, \epsilon]$ for each of its components. Additionally, we need to ensure that $\boldsymbol{x} + \boldsymbol{\delta}$ remains bounded between $[0, 1]$ to ensure it remains a valid image.

Later on, we will discuss the rationale behind considering the L^∞ ball or norm-balls in general as perturbation sets. For now, we can say that the advantage of the L^∞ ball is that, for small values of ϵ, the perturbations it creates are visually indistinguishable from the original image. This property provides a "necessarily-but-definitely-not-close-to-sufficient" condition for considering a classifier robust to perturbations. It is worth noting that deep networks are susceptible to being easily deceived by such manipulations.

There are methods to target the functionality of the machine learning model itself, which essentially entails identifying inputs that cause the model to behave unexpectedly and inaccurately. Such inputs are referred to as "adversarial examples". This section aims to provide a comprehensive explanation of adversarial examples and their operational principles.

Adversarial examples are specially crafted inputs designed to deceive neural networks, causing them to misclassify the given input. These inputs are visually similar to the original images, yet they cause the network to fail in recognizing their true contents. One prominent type of attack is the fast gradient sign method (FGSM) [46], which is a whitebox attack where the attacker has complete access to the targeted model.

An example of an adversarial image is shown in fig. 1.2. In this case, the attacker introduces small perturbations (distortions) to an original image, labeled as a "pig". As a result, the model confidently misclassifies the image as a gibbon. The process of adding these perturbations can be further explained mathematically.

Linear models. More advanced models, such as deep neural networks, are often employed to capture the non-linear patterns present in the data and improve robustness against adversarial examples. When considering adversarial examples, assuming a linear hypothesis function can be a useful simplification. By assuming linearity, the problem of defending against adversarial examples can be formulated as a linear optimization problem, which is easier to analyze and solve. However, it is important to note that real-world data often contains complex, non-linear relationships. In such cases, using a linear hypothesis function may not be sufficient to effectively defend against adversarial attacks. However, for the multi-class setting $f_{\boldsymbol{\theta}} : \mathcal{R}^n \to \mathcal{R}^k$, we consider a classifier of the form

$$f_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}, \tag{1.7}$$

where $\boldsymbol{\theta} = \left\{\boldsymbol{W} \in \mathcal{R}^{k\times n}, \boldsymbol{b} \in \mathcal{R}^k\right\}$. Before delving into the multi-class case, we will briefly examine a binary classifier with a slightly different structure. This approach will make it easier to illustrate and describe many of the underlying ideas.

By substituting this hypothesis into our robust optimization framework and considering the case where the perturbation set $\boldsymbol{\delta}$ is a norm ball $\boldsymbol{\delta} = \{\boldsymbol{\delta} : \|\boldsymbol{\delta}\| \leq \epsilon\}$ (where the specific norm is not specified, it could be L^∞, L^2, etc.), we arrive at the following min-max problem:

$$\underset{\boldsymbol{W},\boldsymbol{b}}{\text{minimize}} \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x},y\in\mathcal{D}} \max_{\|\boldsymbol{\delta}\|\leq\epsilon} \ell(\boldsymbol{W}(\boldsymbol{x}+\boldsymbol{\delta}) + \boldsymbol{b}, y). \tag{1.8}$$

We will highlight the key point: under this formulation, we can solve the inner maximization exactly for binary optimization and provide a relatively tight upper bound for multi-class classification. Additionally, because the resulting minimization problem remains convex in $\boldsymbol{\theta}$ (even after maximizing over $\boldsymbol{\delta}$), the resulting robust training procedure can be optimally solved. As a result, we can achieve the globally optimal robust classifier, at least for binary classification. However, understanding the linear case provides important insights into the theory and practice of adversarial robustness, and also provides connections to more commonly-studied methods in machine learning such as support vector machines.

Adversarial robustness and training. In this section, we provide a more rigorous analysis of the challenge posed by adversarial attacks on deep learning classifiers, involving the construction of adversarial examples. Additionally, we explore the challenge of training or modifying existing classifiers in a way that enhances their resilience against such attacks.

To begin, we can consider more formally the traditional notion of risk as it is used in machine learning. The risk of a classifier is it's expected loss under the true distribution of samples, i.e.

$$R(f_{\boldsymbol{\theta}}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}}\left[\ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}), y)\right], \tag{1.9}$$

where $\mathcal{D}$ denotes the true distribution over samples. In practice, of course, we do not know the underlying distribution of the actual data, so we approximate this quantity by considering a finite set of samples drawn draw i.i.d. from $\mathcal{D}$,

$$D = \{(\boldsymbol{x}_i, y_i)\} \sim \mathcal{D}, i = 1, \ldots, m \tag{1.10}$$

and we then consider the empirical risk on some training set denoted $\mathcal{D}_{\text{train}}$ (or possibly some regularized version of this objective)

$$\underset{\boldsymbol{\theta}}{\text{minimize}}\, \hat{R}(f_{\boldsymbol{\theta}}, D_{\text{test}}). \tag{1.11}$$

Once the parameters $\boldsymbol{\theta}$ have been chosen based upon the training set D_{train}, this data set can no longer give us an unbiased estimated of the risk of the resulting classifier, and so frequently an alternative data set D_{test} (also contains points sampled i.i.d. from the true underlying distribution $\mathcal{D}$), and we use $\hat{R}(f_{\boldsymbol{\theta}}, D_{\text{test}})$ as a proxy to estimate the true risk $R(f_{\boldsymbol{\theta}})$.

Adversarial attacks. Improving the robustness and generalization ability of neural networks are fundamental problems in machine learning and specifically in computer vision (e.g. [89, 125]). Thereby, several different aspects of robustness and generalization issues are addressed, from simple distribution shifts between training and test data distributions over a network's robustness to severe image corruptions to adversarial examples.

Let us divide the attack methods between blackbox (BB) and WB. The direct access to the model gradient is unrealistic in many real-world applications, where we need to perform attacks in the blackbox manner such as a query search. WB attacks have access to the gradient. Except for the Square attack in AutoAttack (AA), we consult whitebox attacks.

Furthermore, there are targeted attacks, which aims to receive a certain class for an input. Non-targeted adversarial attacks only aims at misclassification. The susceptability of convolutional neural networks to distribution shifts [53, 72, 103, 106, 116] concerns input domain shifts by for example considering corrupted, noisy or blurred data, as well as small changes in the input induced by adversarial attacks. These are targeted and optimized such as to cause mis-classifications [10, 131] - and thereby reveal the model's failure modes. In fact, most current CNN models can easily be fooled by adversarial attacks such as [46, 99, 131]. The Fast Gradient Sign Method (FGSM) [46] has been proposed in 2014 and was followed by more sophisticated methods like Projected Gradient Descent (PGD) [92], DeepFool [99], Carlini and Wagner [14] or Decoupling Direction and Norm [114]. In 2020, [30] launched a benchmark website[1] with the goal to provide a standardized benchmark for adversarial robustness. The dominating adversarial attack method is AutoAttack [33], which is an ensemble of four attacks: two variations of the PGD [92] attack with cross-entropy loss (APGD-CE) and difference of logits ratio loss (APGD-t), the targeted version of the FAB attack [32], and the blackbox Square attack [4]. The AutoAttack benchmark provides several modes. The standard mode executes the four attack methods consecutively. Only if one attack fails, the failed samples are passed to the next attack method.

In table 1.1, we list the attack methods with their properties. We use the untargeted version of the attack if available because we want to degrade the accuracy most. For examining the experiments, such as table 2.2, table 2.12 and fig. 2.1, it is recommended to group the attacks into the "gradient-based" and the "optimal-boundary" attacks. To the first group belongs FGSM, BIM, PGD, and partly AA). To the second group belongs DF and C&W.

Table 1.1. Overview of the selected attacks methods with their properties.

Attacks	ϵ	**Norm**	**Untargeted**	**Whitebox**
FGSM	8/255	L^∞	yes	yes
BIM	8/255	L^∞	yes	yes
PGD	8/255	L^∞	yes	yes
AA	8/255	L^∞	both	both
DF	none	L^2	yes	yes
CW	none	L^2	no	yes

- **Fast Gradient Method (FGSM)**: The FGSM [46] uses the gradients of a given model to create adversarial examples. In other words, it embodies a whitebox attack, requiring full access to the model's architecture and weights. The process involves maximizing the model's loss with respect to the input image through gradient ascent, resulting in the creation of an adversarial image $\boldsymbol{x}_{adv}$:

$$\boldsymbol{x}_{adv} = \boldsymbol{x} + \epsilon \cdot \text{sign}(\nabla_{\boldsymbol{x}} \boldsymbol{J}(\boldsymbol{x}, y)), \tag{1.12}$$

  where $\boldsymbol{x}$ is the original input, ϵ is the perturbation magnitude, $\nabla_{\boldsymbol{x}} \boldsymbol{J}(\boldsymbol{x}, y)$ is the gradient of the loss $\boldsymbol{J}$ with respect to the input $\boldsymbol{x}$, and y is the true label.

- **Basic Iterative Method (BIM)**: The BIM [138] is an iterative version of FGSM. After each iteration the pixel values need to be clipped to ensure the generated adversarial

[1] robustbench.github.io

examples are still within the range of both the ϵ ball (i.e. $[\boldsymbol{x}-\epsilon, \boldsymbol{x}+\epsilon]$) and the input space (i.e. $[0, 255]$ for the pixel values). The formulation is expressed as follows:

$$\begin{aligned} \boldsymbol{x}_{adv}^{0} &= \boldsymbol{x}, \\ \boldsymbol{x}_{adv}^{N+1} &= \mathrm{CLIP}_{\boldsymbol{x},\epsilon}\{\boldsymbol{x}_{adv}^{N} - \alpha \mathrm{sign}(\nabla_{\boldsymbol{x}} \boldsymbol{J}(\boldsymbol{x}, y))\}, \end{aligned} \tag{1.13}$$

where N denotes the number of iterations, $\mathrm{CLIP}_{\boldsymbol{x}}(\cdot, \epsilon)$ ensures that the perturbation stays within the ϵ-ball around $\boldsymbol{x}$, $\nabla_{\boldsymbol{x}} \boldsymbol{J}(\boldsymbol{x}_{adv}^{(t)}, y)$ is the gradient of the loss.

- **Projected Gradient Descent (PGD)**: PGD [92] is a prominent attack method. Opposed to FGSM, PGD is iterative and adds random initialization of the perturbations in each iteration. The optimized perturbations are then projected onto the ϵ ball to maintain similarity between the original and attacked images in terms of L^2 or L^∞ norm.

$$\boldsymbol{x}_{adv}^{(0)} = \boldsymbol{x}, \quad \boldsymbol{x}_{adv}^{(t+1)} = \mathrm{CLIP}_{\epsilon}\{\boldsymbol{x}_{adv}^{(t)} + \alpha \cdot \mathrm{sign}(\nabla_{\boldsymbol{x}} \boldsymbol{J}(\boldsymbol{x}_{adv}^{(t)}, y))\}, \tag{1.14}$$

where t is the iteration index, α is the step size, and $\nabla_{\boldsymbol{x}} \boldsymbol{J}(\boldsymbol{x}_{adv}^{(t)}, y)$ is the gradient of the loss with respect to the perturbed input at iteration t. In addition, random initialization and restarts are adopted to further strengthen the attack. An enhanced version of PGD, known as AutoPGD [33], presents a variant of PGD with automatic step size tuning and a refined objective function. AutoPGD has demonstrated superior effectiveness compared to PGD under similar attack budgets. RobustBench evaluates on the standard mode, executing the four attack methods consecutively. The failed attacked samples are handed over to the next attack method, to ensure a higher attack rate.

- **DeepFool (DF)**: DF [99] is considered one of the more sophisticated attacks and has been widely studied in the field of adversarial machine learning. This is a non-targeted method that is able to find the minimal amount of perturbations possible, which mislead the model using an iterative linearization approach [99]. The main idea is to find the closest distance from the input sample to the model decision boundary. At each iteration, DeepFool computes the gradient of the network's output with respect to the input image:

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}', \boldsymbol{\theta}). \tag{1.15}$$

Then, it finds the minimum-norm direction that maximally changes the network's output class. This can be done by solving the following optimization problem:

$$\min_{\boldsymbol{\theta}} ||\boldsymbol{\theta}||_2 \text{ subject to } f(\boldsymbol{x}+\boldsymbol{\delta}; \boldsymbol{\theta}) \neq f(\boldsymbol{x}', \boldsymbol{\theta}). \tag{1.16}$$

The solution to this problem gives the direction in which the perturbation should be updated. DeepFool updates the perturbation by taking a small step in this direction:

$$\boldsymbol{\delta}_{adv} = \boldsymbol{\delta} + \frac{r}{||\nabla_{\boldsymbol{x}} f(\boldsymbol{x}'; \boldsymbol{\theta})||_2} \nabla_{\boldsymbol{x}} f(\boldsymbol{x}'; \boldsymbol{\theta}) \tag{1.17}$$

where r is a small positive constant that controls the step size. The algorithm continues iterating until the image is misclassified or a maximum number of iterations is reached.

- **Carlini&Wagner (C&W)**: The attack method C&W [14] is considered one of the most effective adversarial attacks, as it can generate adversarial examples that are hard to detect and can fool a wide range of machine learning models. The attack can be formulated as an optimization problem that minimizes the perturbation while maximizing the model's misclassification rate or minimizing its confidence in the correct classification. This attack can be customized to target different types of norms, such as L^∞ and L^2 norms, which measure the magnitude of the perturbation in different ways. In our benchmarks, we use the L^2 distance, which is most common by using this attack. The optimization problem can be formalized as follows:

$$\min \left\| \frac{1}{2}(\tanh(\boldsymbol{x}_{adv}) + 1) - \boldsymbol{x} \right\| + c \cdot f\left(\frac{1}{2}(\tanh(\boldsymbol{x}_{adv}) + 1)\right) \tag{1.18}$$

with

$$f(\boldsymbol{x}) = \max(Z(\boldsymbol{x})_{true} - \max_{i \neq true}\{Z(\boldsymbol{x})_i\}, 0), \tag{1.19}$$

where $Z(\boldsymbol{x})$ is the softmax classification result vector. The initial value for c is $c = 10^{-3}$, a binary search is performed to find the smallest c, s.t. $f(\boldsymbol{x}_{adv}) \leq 0$.

1.4 Contribution

This thesis provides the following main contributions:

1. We highlight the important weakness of adversarial training, since almost all existent methods are based on vast amounts and depending on the image size it becomes computational expensive. A Fourier analysis strategy is proposed to cope with such a limitation, by allowing to detect adversarial examples in the Fourier domain. Moreover, we leverage the research fields to higher resolution datasets and we are able to detect the proposed attack by RobustBench by higher resolutions more easily.

2. Currently, there are many techniques that can detect adversarial examples. We introduce a method with small changes on the local intrinsic dimensionality to enhance

detection of adversarial examples on smaller and larger image sizes. An analysis of the extracted LID features and their theoretical properties allows us to redefine an LID-based feature using unfolded local growth rate estimates that are significantly more discriminative than the aggregated LID measure.

3. We adversarial train a visual prompting for the first time, which will be added at test-time to the input image. Adversarial training will be usually used to harden a classifier. Since visual prompting is just a padding on the image, it does not rely on the heavy data usage.

4. In this chapter, we highlight success of generative diffusion models to generate images which are very close to the real distribution. We study, if adversarial examples are close to this distribution and therefore transform the adversarial and benign samples respectively through the diffusion model. Our experiments demonstrate strong evidence a noteworthy ability to effectively distinguish between different types of attacks, indicating its capacity not only to detect the presence of an attack in an image but also to identify the specific nature of the attack.

1.5 Publications

This dissertation has led to the following scientific peer-reviewed publications:

1. Lorenz P, Harder P, Straßel D, Keuper M, Keuper J. Detecting AutoAttack Perturbations in the Frequency Domain. In International Conference on Machine Learning (ICML) - Workshop on Adversarial Machine Learning. 2021. (Poster)

2. Lorenz P, Strassel D, Keuper M, Keuper J. Is RobustBench/AutoAttack a suitable Benchmark for Adversarial Robustness? In the Association for the Advancement of Artificial Intelligence (AAAI) Workshop on Adversarial Machine Learning and Beyond. 2022. (Poster)

3. Lorenz P, Keuper M, Keuper J. Unfolding Local Growth Rate Estimates for (Almost) Perfect Adversarial Detection. International Conference on Computer Vision Theory and Applications (VISAPP). 2023. (Oral)

4. Chen* A, Lorenz* P, Yao Y, Chen PY, Liu S. Visual prompting for adversarial robustness. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2023. (Oral, recognition top 3%)

5. Lorenz P, Ricard D, Keuper J. Adversarial Examples are Misaligned in Diffusion Model Manifolds. In IEEE International Joint Conference on Neural Networks (IJCNN). 2024.

1.6 Thesis Organization

This thesis consists of several main parts, excluding the introduction and conclusion. A concise summary of the chapters is provided below.

Chapter 2. In the last few years, RobustBench is a standardized adversarial robustness benchmark using AutoAttack that reflects the robustness of the considered classification models. Instead of hardening a network, we detect adversarial attacks during inference, rejecting manipulated inputs based on a rather simple and fast analysis in the frequency domain. We argue that the alternation of data by AutoAttack unrealistically strong, resulting in close to perfect detection rates of adversarial samples even by simple detection algorithms while other attack methods are much harder to detect and achieve similar success rates. Second, results on low resolution data sets do not generalize well to higher resolution images as gradient based attacks appear to become even more detectable with increasing resolutions. This work was previously published at ICML 2021 and AAAI 2022 workshops.

Chapter 3. We propose a simple and light-weight adversarial examples detector, which leverages recent findings on the relation between networks' local intrinsic dimensionality (LID) and adversarial attacks. Based on a re-interpretation of the LID measure and several simple adaptations, we surpass the state-of-the-art LID detection by a significant margin and reach almost perfect results in terms of F1-score for several networks and datasets on several attack methods. This work was previously published at VISAPP 2023.

Chapter 4. Visual prompting is a paradigm shift in the field of computer vision. You label only a few small areas of an object in a few images, and the model almost immediately detects the whole object in all of your images. In most cases, the model's predictions are not 100% accurate the first time around, but you can easily label a few more small areas, re-run the model, and check your results. We are the first who attempt to counteract adversarial examples via visual prompts. Compared to conventional adversarial defenses, visual prompting allows us to design universal (i.e., data-agnostic) input prompting templates, which have plug-and-play capabilities at test time to achieve desired model performance without introducing much computation overhead. Although VP has been successfully applied to improving model generalization, it remains elusive whether and how it can be used to defend against adversarial attacks. In this work, we leverage visual prompting to improve adversarial robustness of a fixed, pre-trained model at test time. This work was previously published at ICASSP 2023 and recognized as top 3% oral paper.

Chapter 5. In this chapter, we explore how diffusion models (DMs) demonstrate robustness by distinguishing adversarial examples outside their learned manifold. While DMs are known for their generative abilities, this study specifically examines their use in detecting anomalies caused by adversarial attacks, rather than enhancing adversarial robustness in image classifiers. We systematically investigate the impact of adversarial examples and evaluate the efficacy on widely used classifier datasets. The results highlight DMs' significant ability to effectively differentiate between normal and attacked images, offering strong evidence that adversarial instances deviate from the DMs' learned manifold. This work was previously published at IJCNN 2024.

Chapter 6. Last but not least, we conclude this thesis with a final discussion summarizing the most interesting findings regarding adversarial. We also draw some conclusions on what might be promising future directions.

Chapter 2

Detecting Adversarial Images in the Fourier-Domain

Despite the success of Convolutional Neural Networks (CNN) in many computer vision and image analysis tasks, CNNs remain vulnerable against so-called adversarial attacks:

Small, crafted perturbations in the input images can lead to false predictions. Instead of hardening a network, e.g. by adversarial training, we aim to detect attacks during inference and reject manipulated inputs. In this context, we evaluate a practical adversarial example detector termed SpectralDefense (SD) for CNNs. Utilizing the Fourier domain representations of input images and feature maps, we use two methods to distinguish benign test samples from adversarial images on non-standard benchmarks: Our first method, $\text{SD}_{\text{BlackBox}}$ (or short SD_{BB}), employs the magnitude spectrum of the input images to detect an adversarial attack. This simple can successfully detect adversarial perturbations of four out of six commonly used attack methods without having access to the attacked network. The second method, $\text{SD}_{\text{WhiteBox}}$ (or short SD_{WB}) additionally extracts the magnitude of Fourier coefficients of feature-maps at different activation layers of the network. This extension further improves adversarial detection rates on different attack methods and architectures and is highly transferable between neural architectures and datasets.

This detection method is a simple alternative to adversarial training as suggested by RobustBench [30], which has become a widely recognized benchmark for the adversarial robustness of image classification networks. In its most commonly reported sub-task, RobustBench evaluates and ranks the adversarial robustness of trained neural networks on CIFAR-10 under AutoAttack [33] with L^{∞} perturbations limited to $\epsilon = 8/255$. With SpectralDefense, we can show its effectiveness on datasets with higher image dimensions or number of classes and show that AutoAttack is not a stronger attack under the lens of the Fourier transformation.

2.1 BACKGROUND

In this section, we provide an overview of the countermeasures against adversarial examples. Additionally, we introduce related literature and some of its most prominent work.

2.1.1 Preliminaries

Convolutional neural networks have significantly increased the accuracy in many computer vision and image analysis tasks, especially in the field of image classification [68, 115]. To date, the predictions of CNNs can easily be fooled, as shown by Goodfellow et al. [46]. Only small changes in an image, sometimes merely one pixel, can force a CNN based classifier to into a misclassification. These perturbed images are called "adversarial examples" [13, 13] and have drawn a lot of attention recent years. One reason is the possible lack of security they might induce to practical use cases. For instance in image spam filters for emails, where spam is embedded in an image [105]. Another example is if the attacker uploads videos on a platform, i.e. Dropbox, where the content violates the rules of the platform. Though various countermeasures have already been suggested in the literature, new defense mechanisms have often been broken quickly again [17]. Adversarial defenses are mainly divided into two categories. The first refers to various training techniques for improved model robustness even under adversarial attacks. Yet, to successfully defend against orchestrated attacks, vast amounts of training data are necessary. Methods belonging to the first category are for example using JPEG compression [83], which causes artifacts and is equal to adding noise to images. There are more sophisticated training paradigms that employ adversarial examples during training to harden networks (e.g. [46,92]). The second option is to learn to detect adversarial images and reject them. Several detection methods based on PCA [54,76] or other statistical properties [73] have been introduced, but often they have been found to be only effective on simple problems like the MNIST dataset [17].

Other approaches use a separate neural network to distinguish between adversarial and non-adversarial images [96]. However, Carlini and Wagner [17] show that by extending the attack, this additional neural network can be fooled as well. Recently, Tramér et al. [135] showed that many defenses are not able to counteract against so-called adaptive attacks, which are aware of the underlying defense method.

In this work, we evaluate two detection methods, called SpectralDefense [52], that utilize the Fourier domain representation of an image or its feature maps. These methods are able to decide whether the input is benign or an adversarial image. Employing the Fourier spectrum to extract features imperceptible to human eyes has shown to be successful before, for example, to detect Deepfakes [40, 63]. As the adversarial perturbations depend on and interact with the image content, they are usually hard to grasp at a local pixel level. We argue that a global, space-invariant representation such as the Fourier spectrum facilitates to discern of such subtle but systematic modifications.

The first SpectralDefense (SD) method $SD_{BlackBox}$ employs the magnitude Fourier spectrum (MFS) of an input image to detect an adversarial example. Unlike almost all existing detection methods, $SD_{BlackBox}$ does not need any access to the underlying network, it only depends on the input images. Our second Spectral Defense - $SD_{WhiteBox}$ extends this approach to the Fourier spectra of feature maps of the network and has, therefore, insights into the architecture and weights.

We evaluate the proposed methods on the commonly used datasets CIFAR-10, CIFAR-100 [67], CelebaHQ-4 [84] and ImageNet [26] with VGG-16 [124] and a WRN 28-10 [160] target architectures in combination with a wide range of state-of-the-art attacks: FGSM [46], BIM [70], PGD [92], AA [33], Deepfool [99] and C&W [14]. We compare our approaches to the well-established local intrinsic dimensionality (LID) and M-D detectors.

2.1.2 Related Work

Improving the robustness and generalization ability of neural networks are fundamental problems in machine learning and specifically in computer vision (e.g. [89, 125]). Thereby, several different aspects of robustness and generalization issues are addressed, from simple distribution shifts between training and test data distributions over a network's robustness to severe image corruptions to adversarial examples. We will shortly introduce adversarial attacks and its defend methods, i.e. adversarial training and detection.

Adversarial attacks. Let us divide the attack methods between blackbox and whitebox. The direct access to the model gradient is unrealistic in many real-world applications, where we need to perform attacks in the blackbox manner such as a query search. Whitebox attacks have access to the gradient. Except for the Square attack in Autoattack, we consult whitebox attacks.

The susceptability of convolutional neural networks to distribution shifts [53, 72, 103, 106, 116] concerns input domain shifts by for example considering corrupted, noisy or blurred data, as well as small changes in the input induced by adversarial attacks. These are targeted and optimized such as to cause mis-classifications [10, 131] - and thereby reveal the model's failure modes. In fact, most current CNN models can easily be fooled by adversarial attacks such as [46, 99, 131]. The Fast Gradient Sign Method (FGSM) [46] has been proposed in 2014 and was followed by more sophisticated methods like Projected Gradient Descent (PGD) [92], DeepFool [99], Carlini and Wagner [14] or Decoupling Direction and Norm [114]. In 2020, [30] launched a benchmark website[1] with the goal to provide a standardized benchmark for adversarial robustness. The dominating adversarial attack method is AutoAttack [33], which is an ensemble of four attacks: two variations of the PGD [92] attack with cross-entropy loss (APGD-CE) and difference of logits ratio loss (APGD-t), the targeted version of the FAB attack [32], and the blackbox Squares attack [4]. The AutoAttack benchmark provides several modes. The standard mode executes the four attack methods consecutively. Only if one attack fails, the failed samples are passed to the next attack method.

Adversarial training. Robust models can be built by extending the training data with adversarial examples, yielding an adversarial training scheme. This idea of adversarial training (AT) can be backtracked to FGSM [46] in 2015. An adversarial example is in this case a subtly changed image causing a machine learning model to misclassify it. Consequently, the achieved robustness by AT depends on the strength and type of the adversarial examples used. For example, training on Goodfellow's FGSM, which is a fast and non-iterative algorithm, only provides robustness against non-iterative attacks, but needs for example early stopping [111, 145] to provide robustness against for example PGD [92] attacks. Consequently, [135] propose training on multi-step PGD adversaries, achieving state-of-the-art (SOTA) robustness against L^∞ attacks on MNIST and CIFAR-10 datasets. The impractical computational complexity of AT makes this hardly affordable for large-scale problems such as ImageNet.

Adversarial detection. Many recent publications have concentrated on adversarial attack detection address to distinguish adversarial from natural images.

Hendrycks and Gimpel [54] showed that adversarial examples have higher weights for larger principal components of the images' decomposition and use this finding to train a detector. Similarly, [76] and [9] also employ a principal component analysis (PCA) approach. Based on the outputs of the neural networks' final layer, [44] defines two metrics: 1) the kernel density estimation and 2) the Bayesian neural network uncertainty to identify adversarial perturbations. [81] proposed a method to detect adversarial examples by leveraging steganalysis[2] and estimating the probability of modifications caused by adversarial attacks. Apart from the statistical analysis of the input images, adding a second neural network to decide whether an image is an adversarial example is another possibility. [96] proposed such a model that is trained on outputs of multiple intermediate layers. Our proposed detection approach is different from the above - it leverages an images or its feature map's frequency

[1] robustbench.github.io

[2] Steganography is the practice of concealing a message within an image.

decomposition to discriminate between benign and adversarial images.

Supervised approaches. The local intrinsic dimensionality (LID) [91] and the mahalanobis distance (M-D) [73] are two strong and popular detectors:

- **Mahalanobis distance detector:** Lee et al. proposed a simple yet effective method for detecting any adversarial samples. The confidence score is defined using the Mahalanobis distance w.r.t. the nearest class-conditional distribution, where its parameters are chosen as empirical class means and tied empirical covariance of training samples. Lee et al. computed the empirical mean and covariance for each training sample. Then, Lee et al. calculated the M-D distance between a test sample and its nearest class-conditional Gaussian.

- **Local Intrinsic Dimension (LID) detector:** LID is a general-purpose metric that measures the distance from an input to its neighbors. Ma et al. used the LID as a characteristic of adversarial subspaces and identified attacks using this measure. Specifically, they propose to numerically approximate the LID for each image and layer of benign and adversarial examples and train a logistic regression model to discriminate between both.

Fourier analysis of adversarial attacks. Durall et al. [137] analyzed the Fourier representation of generated images and pointed out that CNN based generative models are not able to reproduce specific frequency ranges in the Fourier domain. The investigated trade-offs between Gaussian data augmentation and adversarial training [158] take a Fourier perspective on adversarial detection and observed that adversarial examples are not only a high-frequency phenomenon. In their very recent paper, Ma et al. [90] assumed that internal responses of Deep Neural Network (DNN) follow the generalized Gaussian distribution, both for benign and adversarial examples (but with different parameters). They extract the feature maps at each layer in the classification network and calculate the Benford-Fourier coefficients for all of these representations. This concurrent approach is similar to the white-box detector, but builds upon a more complicated representation.

2.2 CONTRIBUTIONS

In this context, the simple SD_{BlackBox} is on-par with these detectors on images attacked by FGSM, BIM, PGD and AA but less successful on DF and C&W, while the SD_{WhiteBox} approach shows superior results in most evaluated scenarios. In detail, this paper provides:

- an investigation of the systematic changes in the frequency representation of images altered by adversarial attacks.

- an in-depth evaluation of $\mathrm{SD}_{\mathrm{BlackBox}}$, a simple detector that only uses the input images, without any need of access to the network, employing the magnitude of Fourier coefficients.

- an in-depth evaluation of $\mathrm{SD}_{\mathrm{WhiteBox}}$, a more complex method that uses the magnitude Fourier spectrum of feature maps and further improves detection performance as well as transferability of learned detectors with respect to datasets, architectures, and attacks.

- an in depth investigation showing that adversarial samples generated by AutoAttack with L^{∞}, $\epsilon = 8/255$ are modifying test images to the extent that these manipulations can be so easily detected, which in conclusion, leads to our argument that this common benchmark configuration should be abandoned.

This work consolidates our previous conference and workshop contributions published in [52], [85] and [87] on Spectral Defenses and their analysis.

2.3 METHOD

In this section, we first summarize Fourier analysis as a tool to investigate the properties and behavior of adversarial attacks. We provide an analysis of adversarial samples generated by different attack methods in the Fourier domain. Last, we present and discuss the proposed methods for adversarial attack detection.

2.3.1 Problem Definition

With the proposed method "SpectralDefense", we address the detection of adversarial examples in the frequency space, in particular the prominent AutoAttack from RobustBench, a standardized adversarial examples benchmark. Since, current adversarial attack methods are usually desgined on the CIFAR-10 dataset, we investigate on other datasets with various scales and number of classes. Given an adversarial attacked image $\mathbf{x} + \boldsymbol{\delta} \in \mathcal{R}^{H \times W \times 3}$, and a pre-trained classifier $f(\boldsymbol{\theta})$, we want to filter detect the attacked images out before the reach and fool the prep-trained classifier. Our goal is to show based on comprehensive experiments, that the proposed attack AutoAttack is not outperforming the other attacks especially on higher image resolutions. Note that H and W are the height and width of the data, respectively.

2.3.2 Definition of the Fourier Transform

The Fourier transformation decomposes a function into its spatial and temporal frequency. A signal sampled at equidistant points is known as discrete Fourier transformation (DFT). The DFT of a signal with length N can be computed efficiently with the Fast Fourier Transformation (FFT) in $\mathcal{O}(N \log N)$ [28] time. For a discrete 2D signal, like color image channels or single CNN feature maps – $\in [0,1]^{N \times N}$ – the 2D discrete Fourier transform is defined as

$$\mathcal{F}(\boldsymbol{X})(l,k) = \sum_{n,m=0}^{N} e^{-2\pi i \frac{lm+kn}{N}} \boldsymbol{X}_{m,n}, \tag{2.1}$$

for $l,k = 0,\dots N-1$, with complex valued Fourier coefficients $\mathcal{F}(\boldsymbol{X})(l,k)$. In the following, we will only utilize the magnitudes of Fourier coefficients

$$|\mathcal{F}(\boldsymbol{X})(l,k)| = \sqrt{\mathrm{Re}\left(\mathcal{F}(\boldsymbol{X})(l,k)\right)^2 + \mathrm{Im}\left(\mathcal{F}(\boldsymbol{X})(l,k)\right)^2} \tag{2.2}$$

and show that this is sufficient to detect adversarial perturbations with high accuracy.

2.3.3 Analysis of Adversarial Samples in Frequency Space

Adversarial examples are not always visible in the spatial domain on the first sight, but is significant in the Fourier domain. Therefore, we investigated each attack and additionally Gaussian noise in fig. 2.1. The spectrum difference between and example and its attacked counterpart shows differences in the Fourier domain for each attack.

The 1$^{\text{st}}$ column shows example of the original image followed by examples for the different attack methods for a perturbation size of $\epsilon = 8/255$. The 2$^{\text{nd}}$ column contains the 2D Fourier spectrum of the (adversarial) examples. The 3$^{\text{rd}}$ column shows the average mean power spectrum of the difference from 1000 examples and adversarial examples in the spatial domain. The 4$^{\text{th}}$ column displays the mean of the difference from 1000 examples and adversarial examples in the spatial domain. The last column shows the spectrum difference between the normal example and the corresponding adversarial example. Both, DF and C&W, have a noticeably smaller spectrum difference as the other attacks.

2.3.4 Detecting Adversarial Samples in the Frequency Domain

Based on the observation in the previous section, we propose a detection method, which is based on the frequency-domain features originally introduced in [52], which we revise in the next subsections. We explicitly propose two types of detectors in contrast to the original [52]. First, a more general $\text{SD}_{\text{BlackBox}}$ based detector, which has zero knowledge about the target network. Second, a $\text{SD}_{\text{WhiteBox}}$ detector which has access to the feature maps of the target network, allowing it to observe the network response to input images. To detect perturbations, we found that the Fourier power spectrum provides sufficient information in both cases. Finally, we neglect the phase-based features, which are also suggested in [52].

Blackbox detection - Fourier features of input images. Figure 2.1 and 2.2 gives a brief visualization of the analysis of the changes in successfully perturbed images from AutoAttack: While different attacks show distinct but randomly located change patterns in the spatial domain (which makes them hard to detect), adversarial samples show strong, well-localized signals in the frequency domain.
Hence, we extract and concatenate the 2D power spectrum of each color channel (see eq. (2.2) and fig. 2.1 as feature representations of input images and use simple classifiers like Random Forests (RF) and Logistic Regression (LR) to learn to detect perturbed input images.

Whitebox detection - Fourier features of feature-maps. In the whitebox (WB) case, we apply the same method as in the blackbox approach, but extend the inputs to the feature map responses of the target network to test samples. Since this extension will drastically increase the feature space for larger target networks, we select only a subset of the available feature maps. Note that the optimal selection of feature maps depends on the topology of the target network. See table 2.5 in section 2.5.1 for details on our selection for CIFAR-10 and for an overview of all feature maps of all networks see section 2.5.4.

2.3.5 Measuring Adversarial Detection

The AutoAttack benchmark [33] uses a “Robust Accuracy” measure to compare different methods (see table 2.1 for details). However, our approach does not fit this evaluation scheme, since we are aiming to reject adversarial test samples instead of hardening the networks. Hence, we propose two different metrics: The *adversarial succes rate (ASR)* in eq. (2.3) is calculated as

$$\mathrm{ASR} = \frac{\#\ \text{perturbed samples}}{\#\ \text{all samples}} \tag{2.3}$$

the fraction of successfully perturbed test images and provides a basis of the ability of attacks to fool unprotected target networks. We measure the performance of our defense by the *adversarial success rate under detection (ASRD)* in eq. (2.4). Here, we compute the ratio of successful attacks under defense

$$\mathrm{ASRD} = \frac{\#\ \text{undetected perturbations}}{\#\ \text{all samples}} = \mathrm{FNR} \times \mathrm{ASR}, \tag{2.4}$$

where FNR is the false negative rate of the applied detection algorithm. The lower the ASRD rate, the more perturbed examples are conquered.

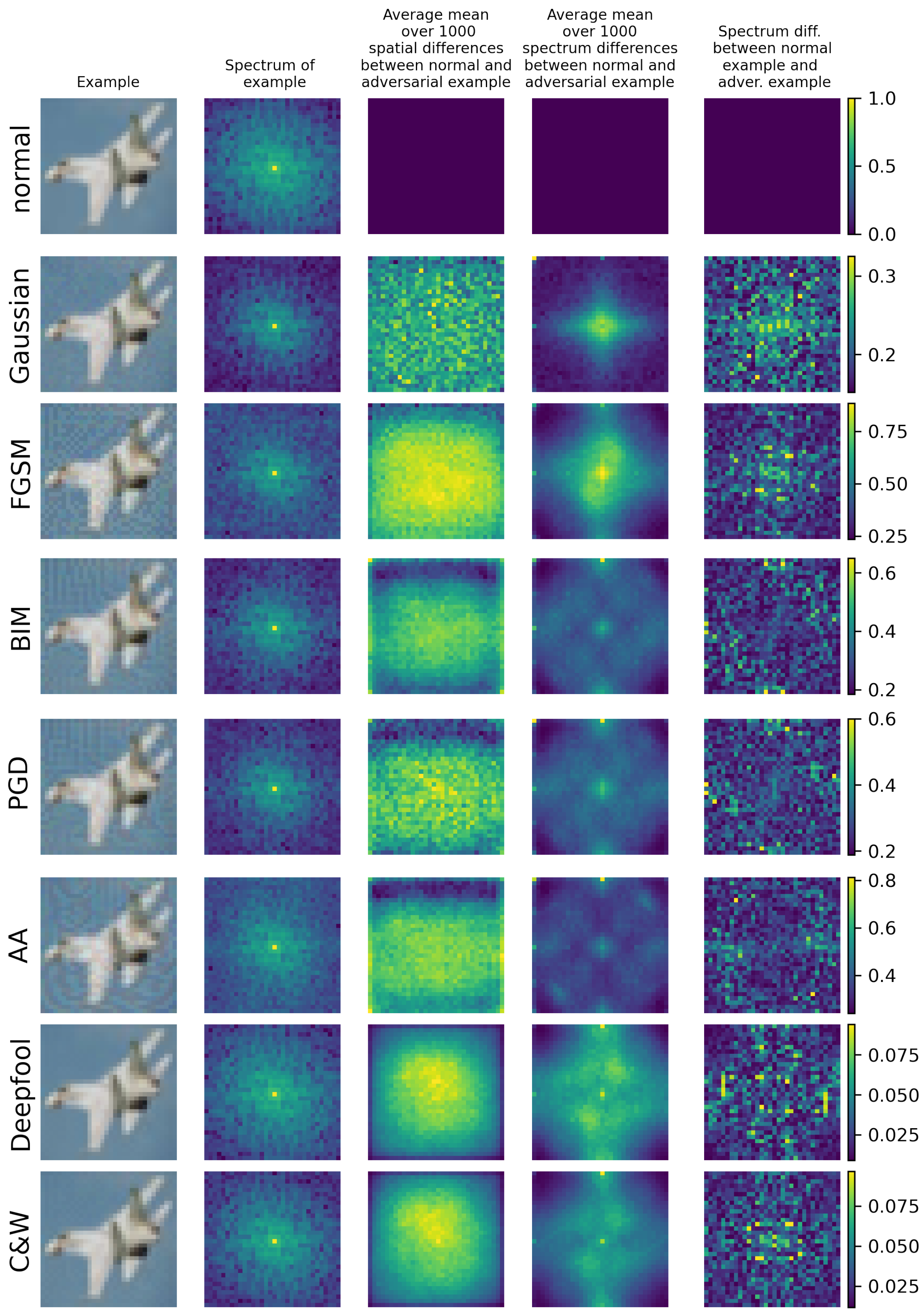


Figure 2.1. Differences between spatial and frequency domain (see eq. (2.1)) are shown. The Fourier power spectrum is plotted logarithmically.

Table 2.1. RobustBench: The top-5 entries of CIFAR-10 leaderboard for L^{∞} in June 2021.

Rank	Method	Standard Accuracy	Robust Accuracy	Extra Data	Architecture	Date
1	Fixing Data Augmentation to Improve Adversarial Robustness	92.23%	66.56%	✓	WRN 70-16	Mar 2021
2	Uncovering the Limits of Adversarial Training against Norm-Bounded Adversarial Examples	91.10%	65.87%	✓	WRN 70-16	Oct 2020
3	Fixing Data Augmentation to Improve Adversarial Robustness	88.50%	64.58%	✘	WRN 106-16	Mar 2021
4	Fixing Data Augmentation to Improve Adversarial Robustness	88.54%	64.20%	✘	WRN 70-16	Mar 2021
5	Uncovering the Limits of Adversarial Training against Norm-Bounded Adversarial Examples	89.48%	62.76%	✓	WRN 28-10	Oct 2020

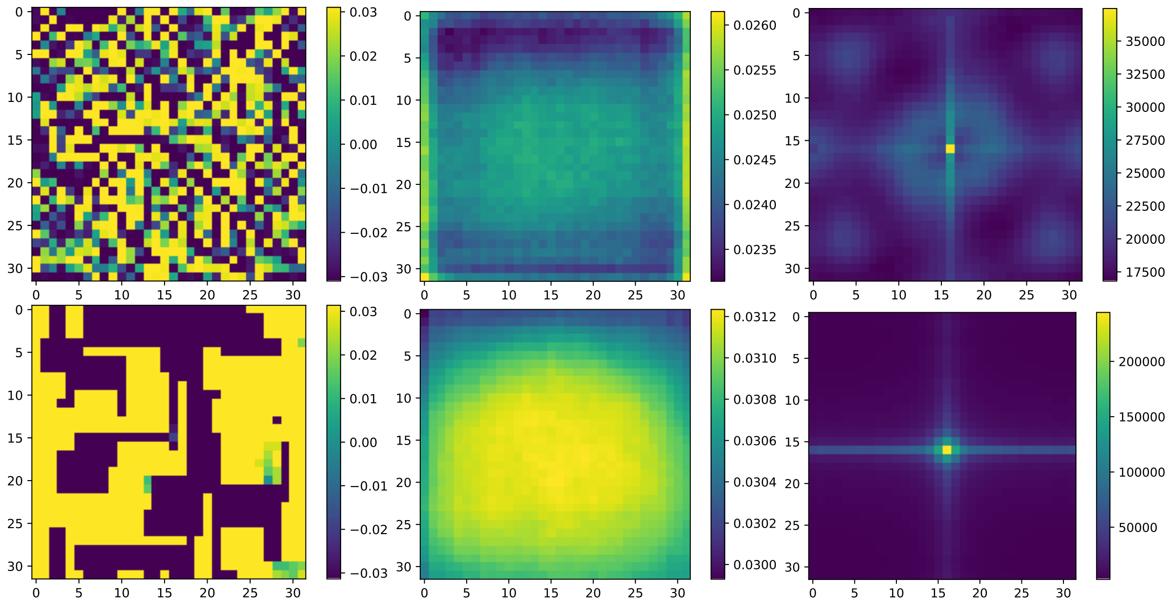


Figure 2.2. Visualization of AutoAttack perturbations on a ResNet18 for CIFAR-10. The top row: APGD-CE L^{∞} attack, bottom row: Squares L^{∞} attack. Left column shows the spacial difference between a random test image from CIFAR-10 and its perturbation. The center column depicts the mean of spacial differences over 1000 perturbed images. Right column: accumulated magnitudes of the spectral differences over the same 1000 images. While there are no obvious clues that can be obtained from the spacial domain, the frequency representation of perturbations show significant and systematic changes which can be exploited to detect attacks.

2.4 EXPERIMENTS

In this section, we present experimental results evaluating the effectiveness of the proposed method. We first give a detailed description of the experimental setup and of the provided datasets. Then, we present the results, and we discuss the possible interpretation. We distinguish between the blackbox and whitebox defense method.

2.4.1 Experimental Setup

In this section, we explain our experimental setup and fig. 2.3 gives an overview: First of all, we train a model on different datasets such as CIFAR-10, ImageNet and CelebaHQ-4, so that we have different image size and class numbers. Then, we can select the benign data and apply different attacks on it. From the attacked images, either we extract the features (DFT) from the images directly or from the feature maps the neural networks. Based on these extracted layer features, we compare our $\mathrm{SD}_{\mathrm{BlackBox}}$ and $\mathrm{SD}_{\mathrm{WhiteBox}}$ defenses. Based on these features we train a binary classifier. For comparison, we choose LR and RF as binary classifier. For all experiments (except we clarify it), we use 2000 samples and split it into 80:20 for train and test set.

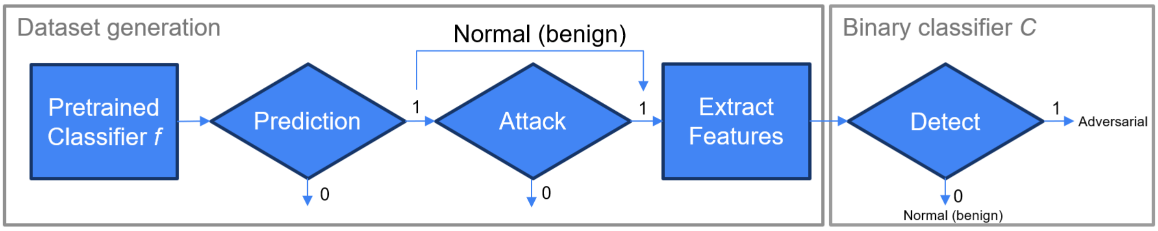


Figure 2.3. Overview of the experimental setup. We have a pre-trained classifier and generate benign dataset by just selecting the correct predicted images. These benign are attacked and features are extracted to train a binary classifier. We define class 0 as benign data and class 1 as the attacked counterparts.

2.4.2 Datasets

Since most of the successful methods ranked on Robustbench are based on a WRN 28-10 [146, 160] architecture, we also conduct our evaluations on a vanilla WRN 28-10, if not noted otherwise. We use the following datasets without applying adversarial examples or other methods to increase the robustness during training.

- **CIFAR-10.** We train on the plain CIFAR-10 training set to a test accuracy of 96% on a WRN 28-10 and apply the different attacks on the test set. We also have benchmarks on VGG-16 [124], which reports a test-accuracy of 72%.

- **CIFAR-100.** The procedure is similar to CIFAR-10 dataset. We train on the CIFAR-100 training set to a test-accuracy of 81% on a WRN 51-2 apply the attacks on the test set. We also have benchmarks on the VGG-16 architecture, which reports a test-accuracy of 83%. The main difference to the CIFAR-10 dataset is that CIFAR-100 has 100 classes instead of just 10.

- **ImageNet.** For the benchmarks on ImageNet we used the pre-trained WRN 51-2 [160] from the PyTorch library. As test set, we apply the official validation set from ImageNet. The accuracy of this pre-trained model is about 77%.

- **ImageNet32 (64 and 128.).** This dataset [26] (and its variants 64×64 / 128×128px) has the exact same classes and images as the original ImageNet with the only difference that the images are down sampled and trained on a WRN 28-10. Moreover, a lower resolution of the images makes the classification task more difficult. The test-accuracy of the ImageNet is for size 32px 60%, for size 64px 69% and for size 128px 86%.

- **CelebaHQ32 (64 and 128.).** This dataset [84] provides images of celebrities faces in HQ quality (1024×1024px) whereas we down sampled it to 32, 64 and 128 pixels width and height. In addition, we only selected the attributes "Brown Hair", "Blonde Hair", "Black Hair" and "Gray Hair" to train a wide residual networks (WRN) 28-10 with an accuracy of 91%. Due to the 4 classes, we call the dataset CelebaHQ-4 throughout this work. The data is unbalanced, where the class "Gray Hair" has least samples.

2.4.3 Evaluation of Attack Success Rates

In this section, we will explain the measurement ASR. There is a relation between perturbation size and how successful an attack can be applied.

Optimal perturbation size for FGSM, BIM, PGD and AutoAttack. The ASR for each method are reported in table 2.12, along with the used perturbation size[3] $\epsilon = 8/255$. We choose the perturbation sizes small enough, not to visually distort the images, but large enough to be able to attack the network successfully. The perturbation size is smaller than in many other publications, for example, [81, 90] this makes detection more difficult but is also a more realistic case. In fig. 2.4 the influence of the perturbation size ϵ on the rate of success is depicted for FGSM, BIM, PGD, AA attacks and the $\text{SD}_{\text{BlackBox}}$ detection method. If the epsilon ϵ is too small, the attacks are only successful on a few samples and it is hard to detect them.

[3] Perturbed images would round the adversarial changes to the next of 256 available bins in commonly used 8-bit per channel image encodings.

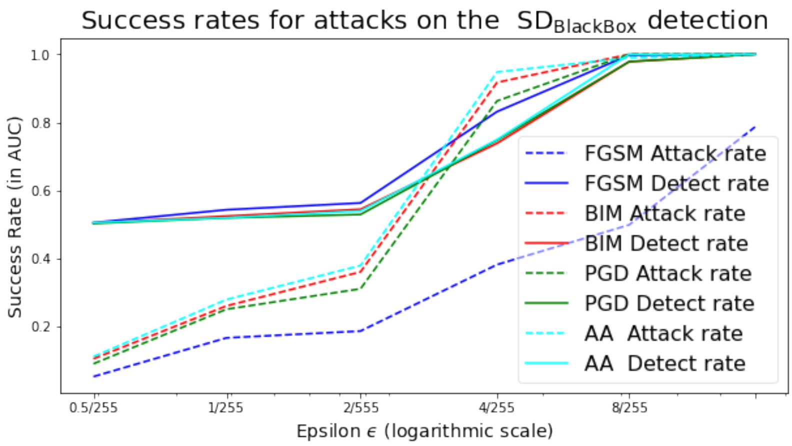
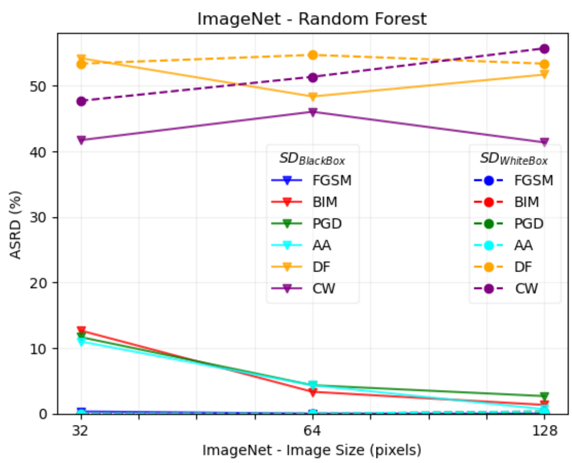
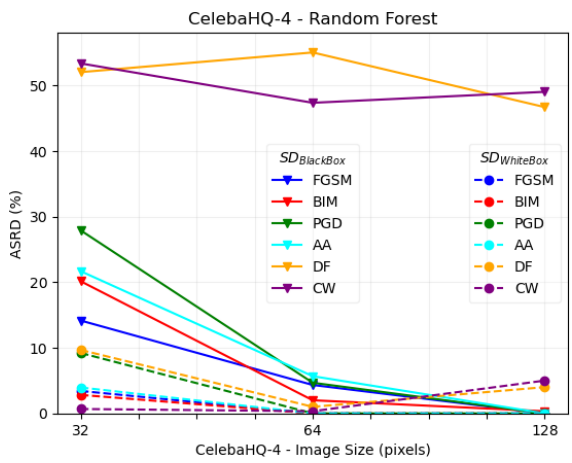


Figure 2.4. The rate of successfully attacked examples by FGSM, BIM, PGD, and AA attacks together with the detection rate (in AUC) of the $\text{SD}_{\text{BlackBox}}$ detector depending on epsilon ϵ. The attacks are applied on the CIFAR-10 test set with their default hyperparameters in foolbox [109].

Success rates depending on image resolution. As shown in fig. 2.5a and 2.5b, we compare attack rates over the three image sizes ($s = \{32, 64, 128\}$) on the datasets CelebaHQ and ImageNet. The attacks FGSM, BIM, PGD and AA are sensitive to the image size. The used detector has better results when the image size is increased. In contrast, DF and C&W keep their attack strength over all image sizes s. Figure 2.7b and 2.7a show a subset of whitebox and blackbox ASRD results for all attack methods on datasets with a resolution of 32×32. The full ASRD evaluation on all datasets is listed in table table 2.12 of the appendix. In both cases, AutoAttack has very low ASRD rates, not only compared to other methods but also in absolute values. In most cases, the probability of successful AA attacks is marginally low.



(a) On ImageNet.



(b) On CelebaHQ-4.

Figure 2.5. ASRD with Random Forest classifiers on increasing resolutions.

2.4.4 Results

Comparative results with other state-of-the-art detectors are reported in table 2.2, we present the area-under-curve (AUC) score, false negative rate (FNR) and adversarial success rate under detection (ASRD), on CIFAR-10 with the WRN 28-10 architecture. This architecture is very often used for adversarial training as listed in RobustBench [30].

We compare our detection methods to two popular open-source detectors, local intrinsic dimensionality (LID) and mahalanobis distance (M-D) detection, as described in section 2.1.2. The hyper-parameters for the LID methods are the batch size and the number of neighbors, as suggested in [91] the batch size is set as 100, and the number of neighbors to 20.

For M-D we use the whole CIFAR-10 training set to calculate the mean and covariance. We choose the magnitude individually for each attack method, 0.0002 for FGSM, 0.00005 for BIM and DF, 0.05 for AA, and 0.00001 for C&W.

For the gradient-based attacks (BIM and AA), blackbox and whitebox on the magnitude Fourier spectrum (MFS) yields promising results. Opposed to blackbox, whitebox is able to improve the result of the DF attack by using the features from the neural network. Note that our blackbox is the only blackbox approach in table 2.2, while the defenses use layer features maps.

On BIM attacked images even the $\mathrm{SD}_{\mathrm{BlackBox}}$ performs better than LID and M-D detectors. The LID detector is also outperformed for BIM and PGD by the blackbox method. On the DF attacked images, the LID detector is able to outperform in overall by using RF classifier. In the appendix table 2.15, we made another experiment with more samples, trained the binary classifier with samples from the training set, and evaluated on samples on the test set and we could see a better performance on the DF attacked samples.

Table 2.2. Comparison of several known detectors with LID [91], M-D [73] (see section 2.1.2) on CIFAR-10 and WRN 28-10 architecture. The full ASRD evaluation on all datasets is listed in the table 2.12 of the appendix.

CIFAR-10 [67] on **WRN 28-10** [160]				
Defenses	**Attacks**	**AUC**	**FNR**	**ASRD**
$SD_{BlackBox}$	FGSM	99.95	0.00	**0.00**
	BIM	99.44	0.00	**0.00**
	PGD	99.47	0.00	**0.00**
	AA	99.79	0.00	**0.00**
	DF	76.04	33.75	33.75
	CW	60.17	41.25	41.25
$SD_{WhiteBox}$	FGSM	100.00	0.00	**0.00**
	BIM	99.87	0.00	**0.00**
	PGD	99.87	0.25	0.25
	AA	99.98	0.00	**0.00**
	DF	94.77	4.75	**4.75**
	CW	95.82	2.75	**2.75**
LID [91]	FGSM	98.90	7.50	3.57
	BIM	98.61	7.89	7.89
	PGD	97.31	10.00	10.00
	AA	99.88	4.21	4.21
	DF	86.74	25.53	25.53
	CW	79.97	26.05	26.05
M-D [73]	FGSM	99.34	2.63	1.25
	BIM	99.61	3.42	3.42
	PGD	99.66	3.42	3.42
	AA	100.00	0.00	0.00
	DF	96.18	6.05	6.05
	CW	96.54	6.05	6.05

2.5 ABLATION STUDY

In this section, we show additonal results to the experiment section. First, we study the blackbox defense depending on the frequencies in the Fourier domain Then we study also the influence of the number of classes for the whitebox defense and also show the importance of the layer selection for the detection quality. Second, we study the AutoAttack's transferability on the epsilon sizes but data also datasets. Lastly, expand our ablation study on transferability on different attacks.

2.5.1 Black- / WhiteBox Defense Properties

In this section, an ablation study of the $\mathrm{SD}_{\mathrm{BlackBox}}$ gives insights into the investigated frequencies from the attack BIM, AA, and DF. The number of classes affects the detection. Finally, an analysis is shown on AA and DF from each ReLU's feature maps on two architecture, WRN 28-10 [160] and VGG-16 [124].

Influence of the frequency. An interesting question is which frequencies are affected by an adversarial effect. Therefore, we analyze at which frequencies are attacked by BIM and DF on CIFAR-10. As shown in table 2.3, we observe for both methods that by only looking at the lowest or highest 25% frequencies the performance is low. When we only consider the mid-frequency bands, we achieve a very good result. [66, 158] already state that adversarial examples are not a high-frequency issue, but rather a mid-frequency issue.

Influence of the number of classes. The number of classes of a dataset affect the detection results. More classes can lower the detection effectiveness. In table 2.4, we trained several WRN 28-10 models on the ImageNet32 dataset with different number of classes: 50, 100 and 250 are evaluated respectively. For the attack DF, the AUC values decreases, while the ASRD value increases by the number of classes. The detection results of AA is confident over the number of classes.

Table 2.3. $SD_{BlackBox}$ detection results (AUC in %) by selected frequencies for AA and DF attacks, using CIFAR-10 on WRN 28-10 net with $\epsilon = 8/255$.

CIFAR-10 [67] on **WRN 28-10** [160]					
from / to		8	16	24	32
BIM	1	53.8	92.8	98.4	97.8
	8		98.0	**98.7**	98.4
	16			98.5	98.4
	24				59.0
AA	1	68.5	92.8	99.7	99.7
	8		**99.9**	**99.9**	**99.9**
	16			**99.9**	**100.0**
	24				76.8
DF	1	50.3	57.0	61.8	62.9
	8		60.9	**64.2**	62.9
	16			62.6	61.9
	24				50.7

Table 2.4. Detection results of our $SD_{WhiteBox}$ detector. ImageNet32 is trained by different number of classes. The more classes the classifier is trained on, the more difficult to detect for our $SD_{WhiteBox}$. We also refer to the appendix fig. 2.7b, which shows similar results.

WB defense on ImageNet32 trained on **WRN 28-10**								
Classes	**Attacks**	**$SD_{WhiteBox}$**		**LID**		**M-D**		
		AUC	**ASRD**	**AUC**	**ASRD**	**AUC**	**ASRD**	
50	AA	100.0	0.00	100.0	0.00	100.0	1.24	
	DF	**61.2**	39.0	53.4	44.9	**56.8**	46.4	
100	AA	100.0	0.00	100.0	0.00	99.94	0.00	
	DF	56.78	46.8	**56.9**	35.91	55.9	52.9	
250	AA	100.0	0.00	100.0	0.00	100.0	0.31	
	DF	49.37	49.5	55.7	36.5	56.3	44.6	

Layer comparison. A deep neural network is basically a composition of input, hidden and output layers. whitebox defenses have access to the hidden layers and extract the feature maps when a normal image or an attacked image is used as an input. The selection of features is important for a successful SD_{WhiteBox} detector. Therefore, we made two experiments: one for the CIFAR-10vgg [124], the second on the WRN 28-10 [160] as shown in table 2.5, where we also distinguish between VGG and WRN architecture. It is noticeable that a lower accuracy of the WRN 28-10 architecture leads to lower quality of the feature, which are needed for the detection. The VGG-16 architecture indeed does not show that high accuracy but this does not influence the extracted features. We noted the layer number, which is always a ReLU function for the VGG-16 architecture. For the WRN 28-10 architecture, we always take the features map from the second ReLU function in each wide residual block. The SD_{WhiteBox} performs on all architectures very high for the last two extracted layers. A more comprehensive layer analysis for WRN 28-10 is located in the appendix in table 2.10 for the magnitude.

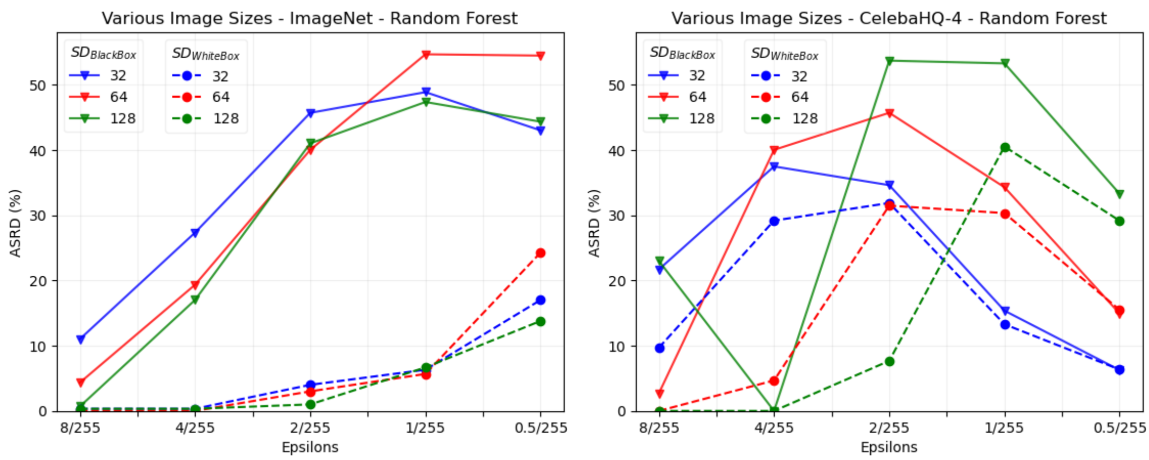
Table 2.5. Detection rate as AUC (in %) using a LR classifier from the SD_{WhiteBox} extracted features from VGG-16 [124] and WRN 28-10 [160] network architectures activation layers on CIFAR-10. The dimension of tensor per layer is also stated.

	SD_{WhiteBox}					
Layer	**WRN28-10**, 96% and 87% ACC				**VGG-16**, 83% ACC	
	AA	**DF**	**AA**	**DF**	**AA**	**DF**
0	99.62	77.21	88.64	57.56	98.52	57.38
1	99.95	80.33	92.59	59.01	98.30	59.06
2	99.87	79.09	96.59	63.60	95.69	53.55
3	99.95	81.41	97.28	64.74	95.62	54.92
4	99.98	81.83	97.87	68.73	98.36	58.82
5	99.94	78.30	97.74	70.86	99.91	66.71
6	99.91	78.17	97.91	69.84	99.94	74.41
7	99.96	78.15	97.21	70.40	**99.98**	82.43
8	99.97	78.33	98.28	71.01	99.96	89.20
9	99.94	76.51	99.40	74.29	99.96	89.20
10	99.94	82.57	99.58	77.37	99.96	92.50
11	**100.0**	92.32	**99.97**	**85.76**	99.95	92.79
12	**100.0**	**94.50**	99.79	82.93	99.96	**93.59**

2.5.2 AutoAttack's Transferability

In this section, we study AutoAttack (AA) in more detail and the capability of attack and dataset transfer. AutoAttack is detectable over different sizes of epsilons with our method. The transferability of between AA, BIM and PGD is high. We also show the transferability between different datasets and resolutions. Finally, we also show the transferability of AA on different epsilon size in combination with different datasets and image resolutions. We also evaluate AA on a more complex and higher resolution dataset, ImageNet.

AutoAttack for different choices of epsilons ϵ. AutoAttack (AA) is the latest attack method. We also can report the best perturbation size is $\epsilon = 8/255$. One might argue that the low ASRD rates of AA might be caused by too high choice of ϵ. Hence, we repeat the full set of AutoAttack experiments for a full range of different ϵ-values. Figures 2.6a and 2.6b show a subset of these evaluation for ImageNet and CelebaHQ-4 on different ϵ, image resolutions as well as SD_{BlackBox} and SD_{WhiteBox} detectors with Random Forests (Comprehensive evaluation results in table 2.18 of the appendix.).



(a) On ImageNet. **(b)** On CelebaHQ.

Figure 2.6. ASRD of AA with random forest for a range of different ϵ and image sizes in pixels.

Examining AA based on WRN 50-2 features. Table 2.6 shows the results for the *individual* mode for $\epsilon = 8/255$ and L^∞-perturbations as well as a comparison to other detection methods. Here, we used 1500 samples from each dataset, CIFAR-10 and ImageNet, for our evaluation. The pre-trained model for ImageNet is from the PyTorch library. AA *standard* is an ensemble of attacks and applies the attacks in a certain order. The first attack, apgd-ce, is most promising one to detect, which is an advantage when using the *standard* mode. A comprehensive evaluation of the *standard* mode is in the appendix table 2.14. Additionally, in table 2.11, we analyzed ReLU layers for our approaches. We take the same feature maps as for AA standard.

Table 2.6. AUC score comparison of detection methods [52]. The attacks from AutoAttack *individual* mode are applied. $\mathrm{SD}_{\mathrm{BlackBox}}$ and $\mathrm{SD}_{\mathrm{WhiteBox}}$ use random forest classifier as in table 2.2 in the section 2.4.4.

AUT Score on each Attack Method from AutoAttack					
		Attack			
Dataset	**Detector**	**APGD-CE**	**APGD-t**	**FAB-t**	**Square**
CIFAR-10 [67] on **WRN 28-10** [160]					
	$\mathrm{SD}_{\mathrm{BlackBox}}$	99.90	99.95	73.36	86.77
	$\mathrm{SD}_{\mathrm{WhiteBox}}$	**100.00**	**100.00**	81.13	**98.12**
	LID [91]	96.99	99.94	93.80	95.56
	M-D [73]	100.00	99.43	**96.59**	97.59
ImageNet [115] on **WRN 51-2** [160]					
	$\mathrm{SD}_{\mathrm{BlackBox}}$	96.97	95.16	50.23	90.94
	$\mathrm{SD}_{\mathrm{WhiteBox}}$	**99.97**	99.97	49.60	**99.05**
	LID [91]	99.68	83.94	55.57	67.73
	M-D [73]	99.89	**99.99**	**99.12**	98.95

2.5.3 Transferability of other Attack Methods

The transferability within attacks or even datasets is very interesting for real world applications. The attack methods and datasets might be unknown and thus it is a desired feature that a detector trained on one attack method performs well for a different attack. Furthermore, we also show the transferability from the epsilon size $8/255 \leftrightarrow 1/255$ and $1/255 \leftrightarrow 8/255$ on AA. Similar to datasets, where a high transferability of different datasets can increase tremendously effectiveness of an attack.

Attack transferability. Attack transfers on $\mathrm{SD}_{\mathrm{BlackBox}}$ and $\mathrm{SD}_{\mathrm{WhiteBox}}$. In table 2.7, we compare three datasets with the same image size, but different dataset complexity by the classes. We can report a high transferability between the gradient-based attacks BIM, PGD

Table 2.7. Attack transfer from different attacks on the datasets CIFAR-10, ImageNet32 and CelebaHQ32-4 trained on WRN 28-10 [160].

Attack Transfer	**CIFAR-10** [67]			**ImageNet32** [26]			**CelebaHQ32-4** [85]		
	AUC	**FNR**	**ASRD**	**AUC**	**FNR**	**ASRD**	**AUC**	**FNR**	**ASRD**
				SD_{BlackBox}					
BIM → PGD	98.57	0.00	**0.00**	86.73	14.33	14.33	85.24	31.33	31.33
BIM → AA	99.29	0.00	**0.00**	85.17	9.67	9.67	82.95	19.67	19.67
PGD → AA	99.31	0.00	**0.00**	85.13	9.33	9.30	82.47	23.00	23.00
				SD_{WhiteBox}					
BIM → PGD	98.29	1.33	1.33	100.0	**0.00**	0.00	95.53	13.67	13.66
BIM → AA	99.33	0.00	**0.00**	99.98	0.33	0.33	96.78	5.00	5.00
PGD → AA	99.52	0.00	**0.00**	99.99	0.33	0.33	95.52	6.00	6.00

and AA. Both SD_{BlackBox} and SD_{WhiteBox} show high transferability between these attacks. The AUC values stay confident over 90% on CIFAR-10 and ImageNet32 SD_{WhiteBox}.

Dataset transferability. The dataset transferability shows strong results across all attack methods in table 2.8. We compare with the image sizes 32×32 and 64×64 px. Compared to table 2.7 the AUC-score decreased far more but still the ASRD depicts high transferability.

Table 2.9. SD_{BlackBox} dataset transfer across different datasets using random forest classifier.

RF	**AUC**	**FNR**	**ASRD**	**AUC**	**FNR**	**ASRD**
From → To	CIFAR-10 → Imagenet32			CIFAR-10 → CelebaHQ32-4		
FGSM	**88.14**	0.00	0.00	47.01	0.00	0.00
BIM	61.55	1.67	1.67	47.06	21.00	21.00
PGD	62.06	2.33	2.33	47.31	19.67	19.67
AA	66.64	1.00	1.00	45.86	14.33	14.33
DF	49.58	2.00	2.00	49.98	2.33	2.33
CW	50.54	10.33	10.33	50.23	32.00	32.00
From → To	ImageNet32 → CelebaHQ32			ImageNet64 → CelebaHQ64-4		
FGSM	57.26	0.67	0.63	**75.20**	0.00	0.00
BIM	52.67	3.00	3.00	62.91	0.00	0.00
PGD	54.08	1.67	1.67	65.27	0.00	0.00
AA	54.39	2.00	2.00	63.67	0.00	0.00
DF	51.18	47.00	47.00	48.82	48.00	48.00
CW	48.54	50.00	50.00	48.42	46.33	46.33

Table 2.8. $SD_{BlackBox}$ dataset transfer across different datasets using logistic regression classifier.

LR	**AUC**	**FNR**	**ASRD**	**AUC**	**FNR**	**ASRD**
From → To	CIFAR-10 → ImageNet32			CIFAR-10 → CelebaHQ32-4		
FGSM	**93.41**	0.00	0.00	53.83	0.00	0.00
BIM	68.66	0.00	0.00	58.25	0.00	0.00
PGD	68.22	0.00	0.00	66.14	0.00	0.00
AA	72.87	0.00	0.00	55.69	0.00	0.00
DF	50.18	1.67	1.67	50.48	0.00	0.00
CW	50.12	2.33	2.33	50.54	0.00	0.00
From → To	ImageNet32 → CelebaHQ32-4			ImageNet64 → CelebaHQ64-4		
FGSM	58.82	4.00	3.75	**73.42**	0.67	0.59
BIM	56.95	0.33	0.33	67.31	0.00	0.00
PGD	56.12	0.33	0.33	65.44	0.33	0.33
AA	56.85	1.67	1.67	68.46	0.33	0.33
DF	50.44	18.00	18.00	50.33	34.33	34.33
CW	50.21	35.00	35.00	50.16	4.00	4.00

2.5.4 Analyzing on all Layers on all Architectures and Datasets

In this section, we analyze all layers on all architectures and dataset. Moreover, we distinguish between logistic regression (LR) and random forest (RF). The top 10% of the AUC or ASRD values per attack are highlighted in the colors green (AUC) and blue (ASRD). The layers across gradient-based attacks are bold faced. See Section 2.4 for details of the experimental setup.

Table 2.10. SD_{WhiteBox} investigation of each ReLU layer of the WRN 28-10 trained on CIFAR-10. The top 10% of the AUC or ASRD values per attack are highlighted in the colors green (AUC) and blue (ASRD). For the LR classifier, we select layer 4 for for attacks FGSM - AA and layer 11 for DF and C&W.

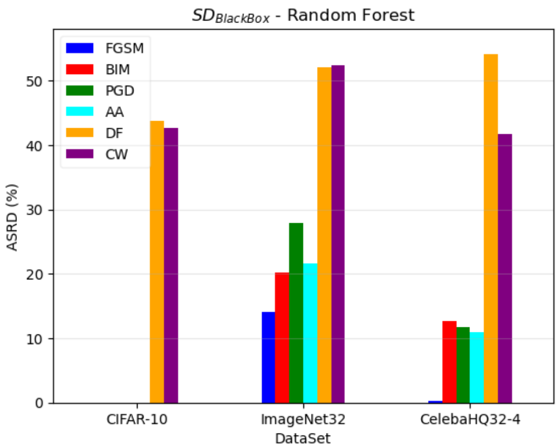
CIFAR-10 [67] on **WRN 28-10** [160]												
Layers	**FGSM** [46]		**BIM** [70]		**PGD** [92]		**AA** [33]		**DF** [99]		**CW** [14]	
	AUC	**ASRD**	**AUC**	**ASRD**	**AUC**	**ASRD**	**AUC**	**ASRD**	**AUC**	**ASRD**	**AUC**	**ASRD**
SD_{WhiteBox} - Logistic Regression												
0	**100.00**	**0.00**	99.89	1.40	99.55	1.15	99.91	0.67	79.82	29.33	65.23	40.67
1	**100.00**	**0.00**	99.99	0.35	99.72	0.87	99.89	**0.00**	84.17	28.33	72.05	34.33
2	**100.00**	**0.00**	**100.00**	**0.00**	99.84	0.29	99.98	0.33	84.23	29.67	70.95	36.33
3	**100.00**	**0.00**	99.99	0.35	**99.92**	0.29	**100.00**	**0.00**	86.25	26.00	77.06	31.00
4	**100.00**	**0.00**	**100.00**	**0.00**	99.86	**0.00**	99.99	**0.00**	86.76	25.33	76.12	33.00
5	**100.00**	**0.00**	99.99	0.35	99.86	0.58	99.99	0.33	84.64	30.00	72.37	37.00
6	**100.00**	**0.00**	99.94	0.70	99.87	0.29	**100.00**	**0.00**	82.60	32.67	70.96	37.00
7	**100.00**	**0.00**	99.91	1.05	99.78	1.73	**100.00**	**0.00**	82.49	30.00	70.37	38.33
8	**100.00**	0.20	99.96	0.70	**99.92**	0.58	**100.00**	**0.00**	82.07	30.33	71.17	37.67
9	99.94	**0.00**	99.88	1.05	99.55	1.73	**100.00**	**0.00**	81.93	29.00	72.27	34.00
10	99.80	0.63	99.72	0.70	99.79	2.31	**100.00**	**0.00**	85.30	24.67	77.74	33.33
11	99.74	**0.00**	99.93	1.75	99.81	2.89	**100.00**	**0.00**	**93.52**	18.33	**89.54**	**18.00**
12	99.23	1.40	98.82	4.21	98.90	3.18	**100.00**	**0.00**	**92.96**	**14.33**	**88.11**	**18.67**
SD_{WhiteBox} - Random Forest												
0	99.98	**0.00**	99.80	**0.00**	99.44	**0.29**	99.69	0.33	77.62	34.67	62.39	44.33
1	**99.99**	**0.00**	99.91	**0.00**	99.73	**0.29**	99.89	**0.00**	80.91	32.33	67.95	39.00
2	99.97	**0.00**	99.87	**0.00**	99.57	**0.29**	99.83	**0.00**	79.12	34.33	65.35	43.00
3	**100.00**	**0.00**	**99.98**	**0.00**	**99.72**	**0.00**	**99.95**	**0.00**	81.53	29.33	71.86	36.67
4	**99.98**	**0.00**	**99.96**	**0.00**	**99.75**	0.58	99.91	**0.00**	81.50	31.67	69.03	38.00
5	99.83	0.60	99.62	1.05	98.97	2.31	99.81	1.00	78.30	32.00	60.61	38.33
6	99.80	1.00	99.50	1.05	98.52	2.31	99.82	0.33	77.96	31.67	61.68	44.33
7	99.79	0.80	99.58	1.05	98.55	3.18	99.92	0.67	78.00	33.33	62.86	33.67
8	99.82	**0.00**	99.34	2.46	98.54	3.47	99.86	**0.00**	78.03	33.67	63.41	42.00
9	99.60	0.40	99.02	5.96	97.73	9.53	99.82	2.33	77.10	37.33	66.55	34.00
10	99.02	0.63	99.34	3.86	98.88	5.20	99.95	1.33	82.53	31.33	82.06	18.00
11	99.33	0.40	99.54	1.75	99.25	2.89	**100.00**	**0.00**	**92.13**	**9.33**	**89.50**	**13.00**
12	99.56	0.60	99.65	2.11	99.20	2.89	**100.00**	**0.00**	**94.57**	**5.00**	**93.90**	**5.33**

Table 2.11. $SD_{WhiteBox}$ investigation of each ReLU layer of the WRN 51-2 [160] trained on ImageNet [115]. The top 10% of the AUC or ASRD values per attack are highlighted in the colors green (AUC) and blue (ASRD). For the LR classifier, we select layer 13 for all attacks.

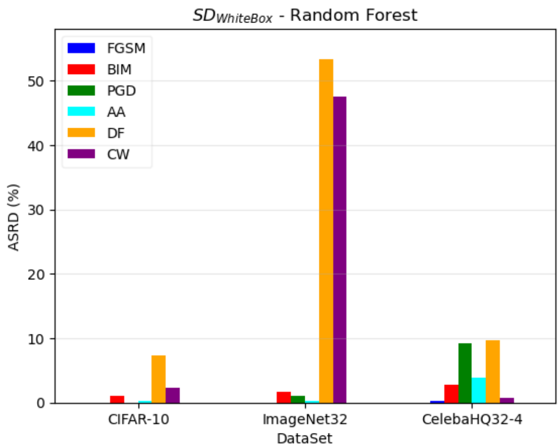
ImageNet [115] on **WRN 50-2** [160]												
Layers	**FGSM** [46]		**BIM** [70]		**PGD** [92]		**AA** [33]		**DF** [99]		**CW** [14]	
	AUC	**ASRD**	**AUC**	**ASRD**	**AUC**	**ASRD**	**AUC**	**ASRD**	**AUC**	**ASRD**	**AUC**	**ASRD**
$SD_{WhiteBox}$ - Logistic Regression												
0	99.31	1.52	95.37	9.12	93.85	10.53	98.37	2.81	50.22	47.02	51.04	43.86
1	**100.00**	**0.00**	99.86	1.40	99.78	2.11	99.98	**0.00**	52.47	53.33	55.63	51.93
2	**100.00**	**0.00**	99.87	1.05	99.63	1.75	**100.00**	**0.00**	53.80	47.72	56.41	44.56
3	**100.00**	**0.00**	99.85	**0.35**	99.77	1.05	**100.00**	**0.00**	56.74	43.16	60.02	41.05
4	99.97	0.30	99.98	0.70	99.95	1.40	**100.00**	**0.00**	**63.85**	37.19	**68.11**	34.04
5	99.98	0.30	99.98	0.70	**99.98**	1.05	**100.00**	**0.00**	63.74	40.70	67.63	35.79
6	**100.00**	**0.00**	99.98	0.70	**99.98**	0.70	**100.00**	**0.00**	63.23	42.81	67.51	35.09
7	**100.00**	**0.00**	99.98	0.70	99.97	1.40	**100.00**	**0.00**	63.38	41.05	67.32	36.49
8	**100.00**	**0.00**	99.96	0.70	99.96	0.70	**100.00**	**0.00**	59.03	**36.84**	63.78	36.14
9	**100.00**	**0.00**	99.96	0.70	99.92	1.05	**100.00**	**0.00**	58.90	37.19	63.85	**33.68**
10	**100.00**	**0.00**	99.96	1.05	99.92	1.40	**100.00**	**0.00**	59.07	**36.84**	63.01	35.44
11	**100.00**	**0.00**	99.96	1.05	99.91	1.40	**100.00**	**0.00**	59.06	40.00	63.28	35.44
12	**100.00**	**0.00**	99.97	1.05	99.92	1.40	**100.00**	**0.00**	59.08	39.30	63.01	36.84
13	**100.00**	**0.00**	**99.99**	0.70	99.96	1.05	**100.00**	**0.00**	58.87	39.30	63.58	35.09
14	99.97	**0.00**	99.86	1.75	99.85	1.40	**100.00**	**0.00**	56.85	40.70	60.44	44.56
15	99.95	**0.00**	99.91	1.05	**99.98**	**0.35**	**100.00**	**0.00**	57.22	41.75	60.12	43.51
16	97.23	8.54	97.56	9.12	98.92	6.67	**100.00**	**0.00**	59.76	42.11	62.53	41.05
$SD_{WhiteBox}$ - Random Forest												
0	99.13	0.30	93.68	2.11	94.44	1.75	97.51	**0.00**	49.78	52.98	50.53	48.07
1	98.52	3.05	96.60	4.91	93.04	8.77	98.94	**0.00**	49.84	55.44	50.57	50.88
2	99.41	1.83	96.90	3.51	95.04	6.32	99.45	0.35	48.42	53.68	49.28	51.93
3	99.41	1.22	95.21	8.77	92.53	10.18	99.18	0.70	48.90	51.93	51.03	51.58
4	99.54	0.61	99.01	1.75	98.75	1.75	99.87	0.35	51.14	51.23	53.02	43.51
5	99.72	0.61	99.06	1.75	99.01	1.40	99.95	0.35	50.08	51.93	51.31	52.28
6	99.79	1.22	99.40	2.46	98.89	1.75	99.92	0.35	51.02	52.28	49.48	50.53
7	99.80	1.22	99.36	2.46	99.10	**0.70**	99.96	0.35	48.22	55.09	50.17	51.23
8	**99.99**	**0.00**	99.88	1.05	99.66	2.46	**100.00**	**0.00**	49.28	52.63	56.84	45.96
9	**99.99**	0.30	99.93	**0.35**	**99.78**	1.75	**100.00**	**0.00**	51.19	43.16	54.60	46.32
10	**99.99**	0.61	99.92	**0.35**	99.55	1.40	**100.00**	**0.00**	53.25	49.82	53.07	46.67
11	**99.99**	0.91	99.92	0.70	99.50	2.46	**100.00**	**0.00**	50.55	49.82	52.16	47.37
12	99.98	0.61	**99.94**	0.70	99.59	1.75	**100.00**	**0.00**	52.83	47.02	52.65	46.32
13	99.97	1.22	99.86	**0.35**	99.59	2.81	**100.00**	**0.00**	51.83	45.61	50.50	48.42
14	99.97	**0.00**	99.42	2.46	99.22	4.91	**100.00**	**0.00**	52.95	48.07	52.95	51.58
15	99.93	0.30	99.70	2.46	99.76	2.46	**100.00**	**0.00**	52.81	49.47	54.69	45.61
16	91.60	13.42	98.25	4.56	99.22	1.75	**100.00**	**0.00**	**57.18**	**39.65**	**60.49**	**43.16**

2.5.5 Comprehensive Study on Vanilla Classification Models

In this section, we show our results of our methods $SD_{BlackBox}$ (or short SD_{BB}) and $SD_{WhiteBox}$ (or short SD_{WB}) on different models and datasets. In table 2.12 is a comprehensive study on WRN 28-10 of our methods. The $SD_{BlackBox}$ is effective on the attacks FGSM, BIM, PGD and AA. While the $SD_{WhiteBox}$ performs better on DF and C&W on smaller resolutions. Additionally, we plotted these insights in fig. 2.7a and fig. 2.7b. In table 2.15, we show that the number of classes have an impact of detection performance from the adaptive attack DF. Lastly, we expand our study on the VGG-16 [124] trained on CIFAR-10 and CIFAR-100 and the pre-trained model WRN 50-2 [160] trained on ImageNet and show that our defenses are model independent.



(a) $SD_{BlackBox}$.



(b) $SD_{WhiteBox}$.

Figure 2.7. ASRD comparison using a random forest classifier on different 32×32 datasets. The lower the ASRD value, the less adversarial examples have successfully fooled the classifier. $SD_{WhiteBox}$ detector shows in general better ASRD results as $SD_{BlackBox}$ detector. $SD_{WhiteBox}$ detector [52]: AutoAttack are so easy to detect. ImageNet32 has the most classes and therefore is harder to detect for DF and C&W.

Table 2.12. Results of the proposed detectors on different attacks with the hyper-parameter ϵ (default in most publications is $\epsilon = 8/255$) and test sets. ASR=Attack Success Rate, ASRD=Attack Success Rate under Detection. A logistic regression and random forest classifier obtains $SD_{BlackBox}$ and $SD_{WhiteBox}$ results on all datasets. AUC and the false negative rate (FNR) are used to report the detection performance. See Section 2.4 for details of the experimental setup.

Dataset	Attack	ASR	$SD_{BlackBox}$						$SD_{WhiteBox}$					
			AUC		FNR		ASRD		AUC		FNR		ASRD	
			LR	RF	LR	RF	LR	RF	LR	RF	LR	RF	LR	RF
WRN 28-10 . Selected Layers for FGSM - AA: Layer 3, 4; DF - CW: Last layer with ReLU activation function. (see table 2.5)														
CIFAR-10	FGSM	50.88±0.0	99.57±0.0	99.79±0.0	1.18±0.0	0.00±0.0	0.6±0.0	0.00±0.0	100.0±0.0	100.0±0.0	0.00±0.0	0.00±0.0	0.00±0.0	0.00±0.0
	BIM	100.0±0.0	96.99±0.0	99.67±0.0	7.33±0.0	0.00±0.0	7.33±0.0	0.00±0.0	100.0±0.0	99.99±0.0	0.26±0.0	0.00±0.0	0.26±0.0	0.00±0.0
	PGD	100.0±0.0	97.43±0.1	99.57±0.0	9.33±0.5	0.00±0.0	9.33±0.5	0.00±0.0	99.99±0.0	99.96±0.0	0.26±0.0	0.26±0.0	0.26±0.0	0.26±0.0
	AA	100.0±0.0	96.66±0.2	99.31±0.0	7.67±2.2	0.00±0.0	7.67±2.2	0.00±0.0	100.0±0.0	100.0±0.0	0.00±0.0	0.00±0.0	0.00±0.0	0.00±0.0
	DF	100.0±0.0	59.00±0.4	62.87±1.3	47.33±2.3	43.67±12.	47.33±2.3	43.67±12.	93.66±1.8	94.82±2.5	10.33±26.	7.33±7.4	10.33±26.	7.33±9.4
	CW	100.0±0.0	51.41±0.1	50.31±0.0	54.33±4.2	42.67±25.	54.33±4.2	42.67±25.	91.48±4.8	96.44±0.8	11.67±22.	2.33±0.5	11.67±22.	2.33±0.5
CIFAR-100	FGSM	84.50±0.0	98.74±0.0	99.97±0.0	6.67±0.0	0.00±0.0	5.64±0.0	0.00±0.0	100.0±0.0	100.0±0.0	0.00±0.0	0.00±0.0	0.00±0.0	0.00±0.0
	BIM	99.95±0.0	95.55±0.0	98.88±0.0	10.33±0.0	0.00±0.0	10.32±0.0	0.00±0.0	99.92±0.0	99.49±0.0	0.79±0.0	0.26±0.0	0.79±0.0	0.26±0.0
	PGD	99.85±0.0	95.65±0.5	99.48±0.0	10.0±6.8	0.00±0.0	9.98±6.8	0.00±0.0	99.97±0.0	99.68±0.0	1.32±0.1	0.79±0.0	1.32±0.1	0.79±0.0
	AA	100.0±0.0	97.6±0.8	99.8±0.0	7.33±2.5	0.67±0.1	7.33±2.5	0.67±0.1	99.98±0.0	99.73±0.0	0.26±0.0	0.53±0.2	0.26±0.0	0.53±0.2
	DF	100.0±0.0	50.00±0.0	49.98±0.0	49.67±4.9	46.00±7.7	49.67±4.9	46.00±7.7	78.51±1.0	76.1±0.3	32.0±4.2	23.67±6.4	32.0±4.2	23.67±6.4
	CW	100.0±0.0	50.07±0.0	49.92±0.0	49.67±3.3	10.33±0.8	49.67±3.3	10.33±0.8	72.74±1.5	70.08±0.2	34.33±1.6	30.00±2.7	34.33±1.6	30.00±2.7
CelebaHQ32-4	FGSM	98.24±0.0	77.65±4.1	78.18±2.2	31.71±0.9	24.39±1.6	31.15±0.8	23.96±1.5	99.74±0.1	97.88±0.1	2.79±0.4	3.48±1.0	2.74±0.3	3.42±0.9
	BIM	100.0±0.0	69.89±2.7	73.66±0.2	39.59±29.0	27.65±0.0	39.59±29.0	27.65±0.0	99.19±0.0	98.4±0.0	2.73±1.6	2.82±0.5	2.73±1.6	2.82±0.5
	PGD	99.93±0.0	65.19±1.7	72.18±2.9	43.34±18.7	29.01±0.5	43.31±18.6	28.99±0.5	98.7±0.1	95.09±0.8	6.48±1.6	9.22±1.5	6.48±1.6	9.21±1.5
	AA	100.0±0.0	77.05±1.2	80.6±0.2	31.67±3.9	21.67±3.7	31.67±3.9	21.67±3.7	99.74±0.0	98.67±0.1	1.05±0.1	3.95±0.6	1.05±0.1	3.95±0.6
	DF	100.0±0.0	59.05±7.6	60.47±0.0	51.05±0.3	52.00±18.	39.67±31.	52.00±18.	86.25±8.4	93.13±0.7	23.33±27.	9.67±1.1	23.33±27.	9.67±1.1
	CW	100.0±0.0	55.76±4.5	57.95±0.1	50.23±0.0	52.33±1.0	44.33±17.	52.33±1.0	86.43±12.	98.67±0.5	19.33±30.	0.67±0.5	19.33±30.	0.67±0.5
CelebaHQ64-4	FGSM	100.0±0.0	90.67±1.1	86.29±0.8	18.09±2.5	9.56±1.3	18.09±2.5	9.56±1.3	100.0±0.0	100.0±0.0	0.00±0.0	0.00±0.0	0.00±0.0	0.00±0.0
	BIM	100.0±0.0	86.46±0.2	84.86±2.1	17.75±20.4	10.58±5.6	17.75±20.4	10.58±5.6	100.0±0.0	100.0±0.0	0.00±0.1	0.00±0.0	0.00±0.1	0.00±0.0
	PGD	100.0±0.0	79.91±1.0	78.59±2.0	24.91±14.5	19.11±3.0	24.91±14.5	19.11±3.0	100.0±0.0	99.96±0.0	0.00±0.1	0.00±0.0	0.00±0.1	0.00±0.0
	AA	100.0±0.0	91.47±1.4	92.87±0.6	14.33±7.2	5.67±1.0	14.33±7.2	5.67±1.0	99.63±0.0	100.0±0.0	2.67±0.2	0.00±0.0	2.67±0.2	0.00±0.0
	DF	100.0±0.0	50.60±0.0	49.84±0.0	54.00±2.3	55.00±9.0	54.00±2.3	55.00±9.0	87.11±4.4	95.30±0.2	15.67±4.2	1.00±0.6	15.67±4.2	1.00±0.6
	CW	100.0±0.0	50.60±0.0	49.53±0.1	50.00±1.8	47.33±7.2	50.00±1.8	47.33±7.2	84.77±5.2	96.21±0.0	19.33±3.9	0.33±0.8	19.33±3.9	0.33±0.8
CelebaHQ128-4	FGSM	95.74±0.0	99.91±0.0	99.97±0.0	2.00±0.1	0.00±0.0	1.91±0.1	0.00±0.0	99.99±0.0	100.0±0.0	0.67±0.0	0.00±0.0	0.64±0.0	0.00±0.0
	BIM	99.95±0.0	99.23±0.1	99.98±0.0	2.00±0.1	0.33±0.0	2.00±0.1	0.33±0.0	99.65±0.0	100.0±0.0	1.33±0.0	0.00±0.0	1.33±0.0	0.00±0.0
	PGD	99.76±0.0	99.78±0.1	99.96±0.0	1.33±2.3	0.00±0.1	1.33±2.2	0.00±0.1	99.97±0.0	100.0±0.0	1.33±0.1	0.00±0.0	1.33±0.1	0.00±0.0
	AA	100.0±0.0	98.08±0.0	99.64±0.0	3.00±3.9	0.00±0.1	3.00±3.9	0.00±0.1	99.89±0.0	100.0±0.0	1.33±0.2	0.00±0.0	1.33±0.2	0.00±0.0
	DF	100.0±0.0	55.96±1.9	54.48±1.7	44.33±16	46.67±3.3	44.33±16	46.67±3.3	75.50±5.8	90.93±0.3	30.00±15.	4.00±0.0	30.00±15.	4.00±0.0
	CW	100.0±0.0	50.82±0.0	50.35±0.0	47.33±7.6	49.00±1.9	47.33±7.6	49.00±1.9	75.75±3.7	88.95±0.6	29.67±4.3	5.00±0.3	29.67±4.3	5.00±0.3
Imagenet32	FGSM	94.35±0.3	95.97±0.0	97.94±0.1	6.65±0.4	1.02±0.2	6.27±0.3	0.96±0.2	100.0±0.0	100.0±0.0	0.00±0.0	0.00±0.0	0.00±0.0	0.00±0.0
	BIM	100.0±0.0	69.67±2.8	80.31±6.2	35.38±2.5	14.62±2.8	35.38±2.5	14.62±2.8	100.0±0.0	99.99±0.0	0.00±0.0	0.00±0.0	0.00±0.0	0.00±0.0
	PGD	100.0±0.0	69.94±3.9	79.46±4.0	34.1±0.3	15.13±1.2	34.1±0.3	15.13±1.2	100.0±0.0	99.99±0.0	0.00±0.0	0.00±0.0	0.00±0.0	0.00±0.0
	AA	100.0±0.0	80.23±0.5	87.95±2.0	23.8±1.1	9.62±3.1	23.8±1.1	9.62±3.1	100.0±0.0	99.93±0.0	0.00±0.0	0.76±0.1	0.00±0.0	0.76±0.1
	DF	100.0±0.0	50.00±0.0	50.02±0.0	53.31±0.1	54.17±0.8	53.31±0.1	54.17±0.8	57.77±0.7	53.44±4.6	47.67±2.3	53.33±18.	47.67±2.3	53.33±18.
	CW	100.0±0.0	50.10±0.0	50.01±0.0	0.28±516	41.67±5.4	0.28±516	41.67±5.4	56.79±0.9	52.82±2.9	42.33±7.5	47.67±3.6	42.33±7.5	47.67±3.6
ImageNet64	FGSM	100.0±0.0	93.78±0.0	98.26±0.1	12.00±0.9	0.00±0.1	12.00±0.9	0.00±0.1	100.0±0.0	99.99±0.0	0.00±0.0	0.00±0.0	0.00±0.0	0.00±0.0
	BIM	100.0±0.0	80.00±0.6	89.64±1.0	26.33±2.7	3.33±2.3	26.33±2.7	3.33±2.3	99.91±0.0	99.92±0.0	0.33±0.0	0.00±0.0	0.33±0.0	0.00±0.0
	PGD	100.0±0.0	80.70±0.1	89.11±0.3	25.00±4.5	4.33±1.2	25.00±4.5	4.33±1.2	99.96±0.0	99.84±0.0	0.33±0.0	0.00±0.0	0.33±0.0	0.00±0.0
	AA	100.0±0.0	82.20±0.6	89.50±0.2	21.33±4.7	4.33±0.0	21.33±4.7	4.33±0.0	100.0±0.0	99.99±0.0	0.00±0.0	0.00±0.0	0.00±0.0	0.00±0.0
	DF	100.0±0.0	50.00±0.0	50.03±0.0	51.33±6.2	48.33±1.4	51.33±6.2	48.33±1.4	58.17±0.2	48.74±6.9	43.67±0.8	54.67±29.	43.67±0.8	54.67±29.
	CW	100.0±0.0	50.16±0.0	50.01±0.0	50.99±30.	46.00±0.2	50.99±30.	46.00±0.2	57.01±0.1	49.71±6.9	42.0±10.3	51.33±6.8	42.0±10.3	51.33±6.8
ImageNet128	FGSM	100.0±0.0	94.89±0.8	98.99±0.0	10.00±4.9	0.00±0.0	10.00±4.9	0.00±0.0	100.0±0.0	100.0±0.0	0.00±0.0	0.00±0.1	0.00±0.0	0.00±0.1
	BIM	100.0±0.0	90.24±1.3	98.91±0.0	20.33±10.3	1.33±0.1	20.33±10.3	1.33±0.1	100.0±0.0	99.97±0.0	0.00±0.0	0.33±0.0	0.00±0.0	0.33±0.0
	PGD	100.0±0.0	89.71±0.6	98.38±0.0	19.00±4.8	2.67±0.1	19.00±4.8	2.67±0.1	100.0±0.0	99.96±0.0	0.00±0.0	0.00±0.0	0.00±0.0	0.00±0.0
	AA	100.0±0.0	82.40±1.4	93.26±0.2	18.67±2.5	0.67±0.0	18.67±2.5	0.67±0.0	99.9±0.0	99.88±0.0	0.00±0.0	0.33±0.1	0.00±0.0	0.33±0.1
	DF	100.0±0.0	50.00±0.0	50.00±0.0	50.33±19.	51.67±0.9	50.33±19.	51.67±0.9	56.08±1.0	48.61±0.0	41.00±9.4	52.33±0.1	41.00±9.4	52.33±0.1
	CW	100.0±0.0	50.16±0.0	50.00±0.0	25.0±33.	41.33±4.3	25.0±33.	41.33±4.3	57.52±2.1	50.15±0.1	39.33±6.0	55.67±6.0	39.33±6.0	55.67±6.0

Table 2.13. Analogous to table 2.12, we evaluated our proposed detectors on the by using a VGG-16 model. See Section 2.4 for details of the experimental setup.

VGG-16 [124]. Selected layers for FGSM - AA [52] are the feature maps of the last 6 and for C&W and DF [52] the last ReLU activation function(s).

Dataset	Attack	ASR	$SD_{BlackBox}$						$SD_{WhiteBox}$					
			AUC		FNR		ASRD		AUC		FNR		ASRD	
			LR	RF	LR	RF	LR	RF	LR	RF	LR	RF	LR	RF
CIFAR-10 [67]	FGSM	92.6±0.3	99.57±0.0	99.79±0.0	1.18±0.0	0.0±0.0	1.09±0.0	0.0±0.0	100.0±0.0	99.83±0.0	0.0±0.0	0.78±0.0	0.0±0.0	0.72±0.0
	BIM	100.0±0.0	96.99±0.0	99.67±0.0	7.33±0.0	0.0±0.0	7.33±0.0	0.0±0.0	99.93±0.0	99.52±0.0	0.67±0.0	1.67±0.0	0.67±0.0	1.67±0.0
	PGD	100.0±0.0	97.43±0.1	99.57±0.0	9.33±0.5	0.0±0.0	9.33±0.5	0.0±0.0	99.94±0.0	98.97±0.0	0.33±0.1	2.67±0.1	0.33±0.1	2.67±0.1
	AA	100.0±0.0	96.66±0.2	99.31±0.0	7.67±2.2	0.0±0.0	7.67±2.2	0.0±0.0	99.94±0.0	99.2±0.0	0.33±0.0	0.33±0.5	0.33±0.0	0.33±0.5
	DF	100.0±0.0	72.52±0.0	77.03±0.0	36.67±0.9	34.0±1.0	36.67±0.9	34.0±1.0	93.19±0.0	94.52±0.0	14.67±0.0	5.33±0.5	14.67±0.0	5.33±0.5
	CW	100.0±0.0	57.34±0.1	60.77±0.4	45.0±0.5	44.0±2.5	45.0±0.5	44.0±2.5	88.28±0.1	94.29±0.0	19.0±0.1	5.67±0.2	19.0±0.1	5.67±0.2
CIFAR-100 [67]	FGSM	72.27±0.0	98.24±0.2	99.66±0.0	5.50±0.5	0.00±0.0	3.97±0.3	0.00±0.0	98.46±0.1	99.70±0.0	4.00±0.5	0.00±0.0	2.89±0.3	0.00±0.0
	BIM	98.19±0.0	94.24±1.8	98.78±0.4	11.25±1.4	0.00±0.0	11.05±1.4	0.00±0.0	94.50±2.0	98.87±0.4	10.50±2.1	0.00±0.0	10.31±2.0	0.00±0.0
	PGD	98.00±0.0	95.25±1.7	99.22±0.1	10.00±1.1	0.00±0.0	9.80±1.0	0.00±0.0	96.22±2.3	99.05±0.1	9.75±3.6	0.00±0.0	9.56±3.4	0.00±0.0
	AA	100.0±0.0	95.74±0.0	99.32±0.0	10.00±0.5	0.00±0.0	10.00±0.5	0.00±0.0	95.62±0.0	99.73±0.0	9.75±0.5	0.00±0.0	9.75±0.5	0.00±0.0
	DF	100.0±0.0	55.89±0.0	67.71±0.0	53.00±6.1	38.50±1.5	53.00±6.1	38.50±1.5	71.66±0.8	83.56±3.9	36.00±0.4	22.33±7.2	36.00±0.4	22.33±7.2
	CW	100.0±0.0	52.14±0.0	57.55±0.6	54.25±5.5	39.75±4.8	54.25±5.5	39.75±4.8	72.32±1.0	84.80±1.9	37.67±5.5	17.33±2.7	37.67±5.5	17.33±2.7

Table 2.14. Analogous to table 2.12, we evaluated our proposed detectors on the ImageNet dataset. See Section 2.4 for details of the experimental setup.

PyTorch Library: **WRN 50-2** [160]. Selected Layers for FGSM - AA: Layer 8, 9; DF - CW: Last layer with ReLU activation function. (see table 2.14)

Dataset	Attack	ASR	$SD_{BlackBox}$						$SD_{WhiteBox}$					
			AUC		FNR		ASRD		AUC		FNR		ASRD	
			LR	RF	LR	RF	LR	RF	LR	RF	LR	RF	LR	RF
ImageNet [69]	FGSM	86.94±0.3	93.91±0.2	98.74±0.0	10.67±1.4	0.00±0.0	9.28±1.0	0.00±0.0	99.99±0.0	99.96±0.0	0.00±0.0	0.00±0.3	0.00±0.0	0.00±0.2
	BIM	99.95±0.0	81.93±0.2	93.37±0.5	22.0±1.0	1.00±0.0	21.99±1.0	1.00±0.0	99.99±0.0	99.94±0.0	0.00±0.0	0.26±0.2	0.00±0.0	0.26±0.2
	PGD	100.0±0.0	91.23±0.1	98.43±0.1	12.00±1.7	0.00±0.1	12.00±1.7	0.00±0.1	99.88±0.0	99.23±0.0	0.79±0.1	2.63±1.0	0.79±0.1	2.63±1.0
	AA	100.0±0.0	90.82±0.0	97.22±0.0	16.67±2.2	0.33±0.0	16.67±2.2	0.33±0.0	100.0±0.0	100.0±0.0	0.00±0.0	0.00±0.0	0.00±0.0	0.00±0.0
	DF	100.0±0.0	50.10±0.0	49.27±0.2	67.33±20.2	55.0±8.2	67.33±20.2	55.0±8.2	61.62±1.5	58.13±0.7	35.26±4.0	40.53±3.0	35.26±4.0	40.53±3.0
	CW	100.0±0.0	50.31±0.0	49.31±1.0	62.33±10.2	49.33±0.5	62.33±10.2	49.33±0.5	62.52±2.2	60.76±0.5	38.68±2.3	39.74±1.0	38.68±2.3	39.74±1.0

Table 2.15. This table shows the difference of WRN and VGG features in regards of number of classes from the dataset. Same setup as in table 2.12. We use all possible train samples and test samples for CIFAR-10 and CIFAR-100 and for ImageNet we take the equivalent amount (44000 from training set and 8000 from the test set). On the DeepFool (DF) attack, we are able to show an improvement of the ASRD value for comparison: table 2.12 and table 2.14. VGG features are not effected by the number of classes. For the WRN 28-10 with CIFAR-100, it also shows that it is more difficult with the WRN 28-10 features and just little performance improvement on DF. See section 2.4 for details of the experimental setup.

Dataset	Attack	ASR	$SD_{BlackBox}$						$SD_{WhiteBox}$					
			AUC		FNR		ASRD		AUC		FNR		ASRD	
			LR	RF	LR	RF	LR	RF	LR	RF	LR	RF	LR	RF
VGG-16 [124]														
CIFAR-10 [67]	AA	100.0	99.38	99.74	2.81	0.04	2.81	0.04	100.0	99.98	0.04	0.67	0.04	0.67
	DF	100.0	60.07	63.33	45.98	37.90	45.98	37.90	95.60	97.70	7.99	1.01	7.99	1.01
CIFAR-100 [67]	AA	100.0	96.70	99.44	6.99	0.03	6.99	0.03	99.99	99.84	0.33	0.15	0.33	0.15
	DF	100.0	62.88	70.21	46.30	39.24	46.30	39.24	76.90	90.76	15.45	1.80	15.45	1.80
WRN 28-10 [160]														
CIFAR-10 [67]	AA	100.0	97.69	99.74	5.98	0.03	5.98	0.03	100.0	99.86	0.23	1.13	0.23	1.13
	DF	100.0	61.74	66.23	45.47	37.47	45.47	37.47	81.97	90.79	21.31	1.14	21.31	1.14
CIFAR-100 [67]	AA	100.0	98.37	99.65	4.42	0.06	4.42	0.06	99.99	99.74	0.15	0.61	0.15	0.61
	DF	100.0	59.98	64.64	48.43	39.44	48.43	39.44	76.92	79.84	24.16	23.54	24.16	**23.54**
WRN 51-2 [160]														
ImageNet [69]	AA	100.0	94.76	97.43	6.70	0.29	6.70	0.29	100.0	100.0	0.05	0.10	0.05	0.10
	DF	100.0	50.01	49.88	99.98	52.62	99.98	52.62	66.74	63.43	35.44	37.79	35.44	**37.79**

2.5.6 Comprehensive Study on Adversarial Trained Classification Models

In this section, we evaluated our defenses, SD_{BlackBox} and SD_{WhiteBox}, on an adversarial-trained model [47] from RobustBench. In Table 2.16, we show at first our results on the same features maps as selected in table 2.12. Then, we adapted the selection of features for the robust model. The selection of the feature maps shows a significant improvement of the detection results. See Section 2.4 for details of the experimental setup.

Table 2.16. Same setup as in table 2.12, but different trained WRN 28-10. The model weights [47] are downloaded from RobustBench, which is adversarial trained. Different feature maps are taken for the whitebox defense method.

Dataset	Attack	ASR	SD_{BlackBox}						SD_{WhiteBox}					
			AUC		FNR		ASRD		AUC		FNR		ASRD	
			LR	RF	LR	RF	LR	RF	LR	RF	LR	RF	LR	RF
RobustBench: **WRN 28-10** same activation functions as for CIFAR-10 for WRN 28-10 in table 2.12.														
CIFAR-10 [67]	FGSM	66.73±0.1	92.51±0.4	98.37±0.1	13.33±3.3	0.00±0.0	8.90±1.5	0.00±0.0	75.08±0.5	60.88±0.2	25.67±15.	41.33±12.	17.13±6.8	41.33±12.
	BIM	79.92±0.0	89.54±0.8	97.96±0.5	14.00±0.5	0.33±0.0	11.19±0.3	0.26±0.0	73.25±0.4	60.52±0.0	30.67±0.5	43.00±0.5	24.51±0.3	43.00±0.5
	PGD	74.22±0.0	89.51±0.4	97.02±0.6	18.00±3.4	0.33±0.0	13.36±1.9	0.24±0.0	72.57±0.9	60.87±0.3	33.00±0.3	43.67±6.7	24.49±0.1	43.67±6.7
	AA	27.42±0.0	90.02±0.3	97.94±0.0	17.33±0.2	0.33±0.1	4.75±0.0	0.09±0.0	61.92±0.0	49.78±6.0	43.67±4.3	51.67±6.4	11.97±0.3	51.67±6.4
	DF	100.0±0.0	62.83±1.0	68.50±0.8	47.67±4.2	46.33±9.0	47.67±4.2	46.33±9.0	71.95±0.2	66.19±0.1	30.00±5.7	19.67±4.3	30.00±5.7	19.67±4.3
	CW	100.0±0.0	58.68±0.6	65.87±0.1	45.33±2.1	45.67±3.1	45.33±2.1	45.67±3.1	71.06±0.3	65.64±0.6	29.33±3.1	22.00±2.4	29.33±3.1	22.00±2.4
RobustBench: **WRN 28-10** FGSM - AA: first activation function. DF and CW: last activation functions.														
CIFAR-10 [67]	FGSM	66.73±0.1	92.49±0.5	98.67±0.2	13.00±4.0	0.00±0.0	8.67±1.8	0.00±0.0	93.81±0.3	91.00±0.3	13.33±10.	11.67±1.3	8.90±4.9	**7.79±0.6**
	BIM	79.92±0.0	89.54±1.0	97.80±0.4	13.67±7.6	0.33±0.0	10.93±4.9	0.26±0.0	90.31±0.6	83.15±4.2	22.33±15.	24.67±14.	17.85±9.3	19.72±8.8
	PGD	74.22±0.0	89.52±0.7	97.18±0.5	18.00±3.1	0.33±0.1	13.36±1.7	0.24±0.0	91.51±1.9	86.66±1.0	15.00±2.1	18.00±8.9	11.13±1.2	13.36±5.0
	AA	27.42±0.0	90.02±0.3	97.94±0.0	17.33±0.2	0.33±0.1	4.75±0.0	0.09±0.0	86.11±1.3	83.03±1.1	24.67±2.1	25.00±3.3	6.76±0.2	**6.86±0.2**
	DF	100.0±0.0	62.83±1.0	68.50±0.8	47.67±4.2	46.33±9.0	47.67±4.2	46.33±9.0	71.95±0.2	66.19±0.1	30.00±5.7	19.67±4.3	30.00±5.7	19.67±4.3
	CW	100.0±0.0	58.68±0.6	65.87±0.1	45.33±2.1	45.67±3.1	45.33±2.1	45.67±3.1	71.06±0.3	65.64±0.6	29.33±3.1	22.00±2.4	29.33±3.1	22.00±2.4

2.5.7 AutoAttack: Hyperparameter and Datasets

In this section, we compare the performance of AutoAttack (AA) applied to different epsilons on all datasets trained on WRN 28-10. In table 2.17, we transferred from larger epsilon to a smaller epsilon ($8/255 \rightarrow 1/255$) and vice versa. The transferability from smaller to larger epsilon reveal strong ASRD values on SD_{WhiteBox}.

Table 2.17. AutoAttack ϵ-transfer on WRN 28-10 [160]. We transfer from the epsilon (ϵ) size $8/255 \leftrightarrow 1/255$ and $1/255 \leftrightarrow 8/255$. $1/255 \leftrightarrow 8/255$ (smaller to larger epsilon) shows higher transferability.

ϵ-Transfer		8/255 →1/255						1/255 →8/255					
		AUC		FNR		ASRD		AUC		FNR		ASRD	
		LR	RF	LR	RF	LR	RF	LR	RF	LR	RF	LR	RF
CIFAR-10 [67]	SD_{BB}	57.78	67.45	90.67	89.33	90.67	89.33	60.80	63.06	98.33	92.67	98.33	92.67
	SD_{WB}	98.30	88.63	0.00	1.67	0.00	1.67	97.20	88.75	2.33	6.00	2.33	6.00
ImageNet32 [26]	SD_{BB}	52.04	52.62	70.00	63.67	70.00	63.67	91.49	92.85	98.00	85.00	98.00	85.00
	SD_{WB}	74.72	63.53	18.00	34.33	18.00	34.33	99.17	99.76	0.00	0.00	0.00	0.00
CelebaHQ32-4 [87]	SD_{BB}	50.58	51.86	70.07	73.81	70.07	73.81	52.08	56.49	92.18	89.46	92.18	89.46
	SD_{WB}	61.52	62.74	36.67	45.67	36.67	45.67	88.80	69.26	4.67	32.67	4.67	32.67
ImageNet64 [87]	SD_{BB}	52.07	52.98	75.33	66.67	75.33	66.67	82.11	85.85	99.33	99.00	99.33	99.00
	SD_{WB}	88.08	66.43	4.33	32.00	4.33	32.00	99.83	99.85	0.00	0.00	0.00	0.00
CelebaHQ64-4 [87]	SD_{BB}	51.33	52.88	89.00	78.33	89.00	78.33	56.97	64.67	97.67	99.71	97.67	99.71
	SD_{WB}	76.78	58.90	18.33	46.33	18.33	46.33	99.49	96.01	0.00	1.67	0.00	1.67
ImageNet128 [87]	SD_{BB}	51.64	51.72	78.00	66.67	78.00	66.67	73.70	70.29	99.33	98.00	99.33	98.00
	SD_{WB}	84.17	62.40	8.33	30.67	8.33	30.67	95.17	97.26	0.00	0.00	0.00	0.00
CelebaHQ128-4 [87]	SD_{BB}	51.72	52.84	91.67	87.67	91.67	87.67	58.97	66.33	99.33	99.67	99.33	99.67
	SD_{WB}	89.50	60.56	5.67	44.33	5.67	44.33	99.93	98.45	0.00	0.33	0.00	0.33

Table 2.18. Various datasets are attacked by AutoAttack but with different epsilons for the perturbation [87]. The ASR falls for different datasets. Another visualization of this table can be found in fig. 2.6a and fig. 2.6b in section 2.5.2. See section 2.4 for details of the experimental setup.

AutoAttack (AA) [33] on **WRN 28-10** [160]														
			$SD_{BlackBox}$						**$SD_{WhiteBox}$**					
Dataset	**Epsilon ϵ**	**ASR**	**AUC**		**FNR**		**ASRD**		**AUC**		**FNR**		**ASRD**	
			LR	**RF**	**LR**	**RF**	**LR**	**RF**	**LR**	**RF**	**LR**	**RF**	**LR**	**RF**
CIFAR-10 [67]	8/255	100.0	96.58	99.38	7.00	0.00	7.00	0.00	99.90	99.85	2.00	0.33	2.00	0.33
	4/255	100.0	90.74	96.63	15.67	0.33	15.67	0.33	97.45	95.85	7.33	2.67	7.33	2.67
	2/255	94.41	76.39	90.10	31.67	10.33	29.90	9.75	91.99	86.08	14.00	16.00	13.22	15.11
	1/255	56.39	61.75	75.05	44.00	26.33	24.81	14.85	75.66	64.38	30.33	33.33	17.10	18.79
	0.5/255	23.14	53.47	54.18	55.52	10.95	47.33	9.56	59.64	53.80	40.67	51.00	9.41	11.80
CIFAR-100 [67]	8/255	100.0	97.52	99.78	8.67	0.33	8.67	0.33	99.69	99.70	2.00	0.00	2.00	0.00
	4/255	99.90	90.80	97.32	17.33	1.33	17.31	1.33	98.02	98.06	9.00	4.67	8.99	4.67
	2/255	97.28	78.33	90.26	31.33	9.33	30.48	9.08	91.51	91.70	15.67	12.00	15.24	11.67
	1/255	73.65	64.89	76.07	36.67	23.33	27.01	17.18	82.42	80.03	25.00	19.67	18.41	14.49
	0.5/255	38.97	55.13	61.72	51.33	36.33	20.00	14.16	68.27	63.13	39.33	37.00	15.33	14.42
ImageNet32 [26]	8/255	100.0	77.80	85.39	29.33	11.00	29.33	11.00	99.99	100.0	0.00	0.33	0.00	0.33
	4/255	99.95	65.67	66.55	37.00	27.33	36.98	27.32	99.92	99.91	0.67	0.33	0.67	0.33
	2/255	100.0	56.61	59.55	42.67	45.67	42.67	45.67	99.70	99.03	3.67	4.00	3.67	4.00
	1/255	99.67	52.04	50.86	47.67	49.00	47.51	48.84	95.17	96.45	12.33	6.33	12.29	6.31
	0.5/255	92.78	50.55	49.60	45.00	46.33	41.75	42.98	86.83	84.81	20.00	18.33	18.56	17.01
ImageNet64 [26]	8/255	100.0	82.20	89.50	21.33	4.33	21.33	4.33	100.0	99.99	0.00	0.00	0.00	0.00
	4/255	100.0	67.44	74.63	33.00	19.33	33.00	19.33	99.79	99.92	1.33	0.00	1.33	0.00
	2/255	100.0	58.75	59.79	39.00	40.00	39.00	40.00	99.65	99.03	2.00	3.00	2.00	3.00
	1/255	99.95	53.98	52.90	52.00	54.67	51.97	54.64	95.69	95.76	12.67	5.67	12.66	5.67
	0.5/255	98.40	51.04	49.25	54.67	55.33	53.80	54.44	77.30	78.21	37.00	24.67	36.41	24.28
ImageNet128 [26]	8/255	100.0	82.40	93.26	18.67	18.67	18.67	0.67	99.90	99.88	0.00	0.33	0.00	0.33
	4/255	100.0	65.76	76.18	42.33	42.33	42.33	17.00	99.37	99.56	1.67	0.33	1.67	0.33
	2/255	98.47	55.72	59.02	44.33	44.33	44.33	41.00	97.49	97.56	6.33	1.00	6.33	1.00
	1/255	100.0	51.85	54.08	54.00	54.00	54.00	47.33	89.97	88.16	15.00	6.67	15.00	6.67
	0.5/255	100.0	50.55	51.03	53.00	53.00	52.19	44.31	76.53	74.87	25.00	14.00	24.62	13.79
CelebaHQ32-4 [84]	8/255	100.0	77.05	80.60	31.67	21.67	31.67	21.67	94.86	96.33	11.33	9.67	11.33	9.67
	4/255	99.43	58.42	57.43	43.33	37.67	43.08	37.46	78.05	78.09	27.33	29.33	27.17	29.16
	2/255	68.26	52.02	51.33	49.00	50.67	33.45	34.59	61.60	60.26	40.00	46.67	27.30	31.86
	1/255	27.70	50.17	49.17	57.82	55.44	16.02	15.36	52.35	51.65	52.38	47.96	14.51	13.28
	0.5/255	10.91	49.99	51.04	40.17	57.26	4.38	6.25	50.14	45.62	43.59	58.12	4.76	6.34
CelebaHQ64-4 [84]	8/255	100.0	91.47	92.87	14.33	2.67	14.33	2.67	99.63	100.0	2.67	0.00	2.67	0.00
	4/255	100.0	68.72	62.38	35.67	40.00	35.67	40.00	96.40	98.85	10.33	4.67	10.33	4.67
	2/255	99.31	54.54	51.28	43.33	46.00	43.03	45.68	81.43	84.53	28.33	31.67	28.13	31.45
	1/255	69.94	51.28	49.65	54.00	49.00	37.77	34.27	61.66	61.65	47.00	43.33	32.87	30.31
	0.5/255	28.14	50.24	50.64	53.33	53.00	15.01	14.91	53.13	51.90	46.00	55.00	12.94	15.48
CelebaHQ128-4 [84]	8/255	100.0	98.08	99.64	24.67	23.00	24.67	23.00	99.89	100.0	5.00	0.00	5.00	0.00
	4/255	100.0	77.98	79.24	3.00	0.00	3.00	0.00	98.62	99.98	1.33	0.00	1.33	0.00
	2/255	100.0	58.48	52.65	45.33	53.67	45.33	53.67	91.93	96.33	18.67	7.67	18.67	7.67
	1/255	98.02	52.20	49.23	48.67	54.33	47.71	53.25	67.28	62.30	37.67	41.33	36.92	40.51
	0.5/255	61.98	50.77	51.01	48.67	53.67	30.17	33.26	55.28	53.45	47.67	47.00	29.55	29.13

2.6 LIMITATIONS

We have benchmarked a lightweight blackbox and whitebox SpectralDefense algorithm to detect adversarial examples and datasets outside of the standardized benchmark procedure. The algorithm is capable to detect most common attack methods and especially AutoAttack (AA). As shown in fig. 2.5a and 2.5b, we compare the ASRD over the three image size ($s = \{32, 64, 128\}$) on the datasets CelebaHQ and ImageNet.

The attacks FGSM, BIM, PGD and AA are not sensitive to the image size. One limitation is that DF and C&W keep their attack strength over all image sizes s. Again, AA does not show sufficient results for using adversarial detection robustness. The results of our empirical evaluations show strong evidence that the widely used AutoAttack scheme for bench-marking the adversarial robustness of image classifier models on low-resolution data might not be a suitable setup in order to generalize the obtained results to estimate the robustness in practical vision applications. Even for lower choices of the ϵ-parameter, AutoAttack still appears to modify target images beyond reasonable class boundaries.

A potential issue, the resolution of the benchmark images should not be neglected. In terms of resolution as well as in the number of classes and training images, CIFAR-10 is a conveniently sized dataset for the very expensive state-of-the-art adversarial training approaches. However, our experiments suggest that these results might not generalize to problems that are more complex. In light of our results, we argue that too strong adversarial benchmarks like the current setting of RobustBench might hamper the development of otherwise practically relevant methods towards more model robustness. For the future, we would like to exploit our understanding of the Fourier spectrum gathered from our analysis to design an unsupervised detector SpectralDefense.

2.7 SUMMARY

We have presented an extensive evaluation on the blackbox and whitebox adversarial examples defense called SpectralDefense. The main motivation behind SpectralDefense is to understand adversarial examples through the lens of the Fourier transform. We analyze SpectralDefense and compare it with other important variants to reveal practical trade-offs using different datasets with varying image sizes, particularly different scales of the same datasets.

On large and competitive datasets SpectralDefense approach shows strong results along to current state-of-the-art approaches and can resist gradient obfuscation [5]. The results of our empirical evaluations show strong evidence that the widely used AutoAttack scheme for bench-marking the adversarial robustness of image classifier models on low-resolution data might not be a suitable setup in order to generalize the obtained results to estimate the robustness in practical vision applications. Even for lower choices of the ϵ-parameter, AutoAttack still appears to modify target images beyond reasonable class boundaries. Additionally, the resolution of the benchmark images should not be neglected. We hope our experiments will encourage researchers to engage their attention towards the more challenging dataset.

Chapter 3

Enhancing Adversarial Detection through Local Growth Rate Analysis

Convolutional neural networks (CNN) define the state-of-the-art solution on many perceptual tasks. However, current CNN approaches largely remain vulnerable against adversarial perturbations of the input that have been crafted specifically to fool the system while being quasi-imperceptible to the human eye. In recent years, various approaches have been proposed to defend CNNs against such attacks, for example by model hardening or by adding explicit defense mechanisms. Thereby, a small "detector" is included in the network and trained on the binary classification task of distinguishing benign data from data containing adversarial perturbations. In this work, we propose a simple and light-weight detector, which leverages recent findings on the relation between networks' local intrinsic dimensionality (LID) and adversarial attacks. Based on a re-interpretation of the LID measure and several simple adaptations, we improve the adversarial detection and reach almost perfect results in terms of F1-score for several networks and datasets.

3.1 BACKGROUND

In this section, we briefly compare adversarial training with adversarial detection and showcase a few methods. Afterward, we define the concept of Local Intrinsic Dimensionality (LID), which forms the basis for our analysis.

3.1.1 Preliminaries

Deep Neural Networks (DNNs) are highly expressive models that have achieved state-of-the-art performance on a wide range of complex problems, such as in image classification. However, studies have found that DNN's can easily be compromised by adversarial examples [32, 33, 46, 92]. Applying these intentional perturbations to network inputs, chances of potential attackers to fool target networks into making incorrect predictions at test time are very high [16]. Hence, this undesirable property of deep networks has become a major security concern in real-world applications of DNNs, such as self-driving cars and identity recognition [42, 122].

Recent research on adversarial counter measures can be grouped into two main approach angles: adversarial training and adversarial detection. While the first group of methods aims to "harden" the robustness of networks by augmenting the training data with adversarial examples, the later group tries to detect and reject adversarial inputs.

3.1.2 Related Work

In the following, we first briefly review the related work on adversarial attacks and provide details on the established attack approaches that we base our evaluation on. Then, we summarize approaches to network hardening by adversarial training. Last, we revise the literature on adversarial detection.

Adversarial attacks. Convolutional neural networks are known to be susceptible to adversarial attacks, i.e. to (usually small) perturbation of the input images that are optimized to flip the network's decision. Several such attacks have been proposed in the past and we base our experimental evaluation on the following subset of most widely used attacks.
In section 1.3, we introduced some common adversarial attacks. Most of them are whitebox attacks (compare table 1.1), which means they have full knowledge of the target model in order to achieve highest attack strengths.

Adversarial training (AT). AT denotes the concept of using adversarial examples to augment the training data of a neural network. Ideally, this procedure should lead to a better and denser coverage of the latent space and thus to in increased model robustness. FGSM [46] adversarial training offers the advantage of rather fast adversarial training data generation. Yet, models tend to overfit to the specific attack such that additional tricks like early stopping [111, 145] have to be employed. Training on multi-step adversaries generalizes more easily, yet is hardly affordable for large-scale problems such as ImageNet due to its computation costs.

Adversarial detection. Adversarial detection aims to distinguish adversarial examples from benign examples and is thus a low computational replacement to expensive adversarial training strategy. In test scenarios, adversarial attacks can be rejected and cause to faulty classifications.

Given a trained DNN on a benign dataset for the origin task, many existing methods [44, 52, 73, 85, 91] train a binary classifier on top of some hidden-layer embeddings of the given network as the adversarial detector. The strategy is motivated by the observation that adversarial examples have very different distribution from natural examples on intermediate-layer features. So a detector can be built upon some statistics of the distribution, i.e., kernel density (KD) [44], M-D [73] distance, or LID [91].

Spectral defense (SD) approaches [52, 85, 87] aim to detect adversarial images by their frequency spectra in the input or feature map representation.

Complementary, [153] propose to train a variational autoencoder following the principle of the class distanglement. They argue that the reconstructions of adversarial images are characteristically different and can more easily be detected using for example KD, M-D and LID.

Local Intrinsic Dimensionality (LID). LID is a measure that represents the average distance from a point to its neighbors in a learned representation space [3, 56] and thereby approximates the intrinsic dimensionality of the representation space via maximum likelihood estimation.

Let $\mathcal{B}$ be a mini-batch of N benign examples and let $d(\boldsymbol{x}, \boldsymbol{y})$ be the Euclidean distance between the sample $\boldsymbol{x}$ and its i-th nearest neighbor in $\mathcal{B}$ as shown in fig. 3.1. Then, the LID can be approximated by

$$\mathrm{LID}(\boldsymbol{x}) = -\left(\frac{1}{k}\sum_{i=1}^{k}\log\frac{d_i(\boldsymbol{x})}{d_k(\boldsymbol{x})}\right)^{-1}, \tag{3.1}$$

where k is a hyper-parameter that controls the number of nearest neighbors to consider, and d is the employed distance metric. Ma et al. [91] propose to use LID to characterize properties of adversarial examples, i.e. they argue that the average distance of samples to their neighbors in the learned latent space of a classifier is characteristic for adversarial and benign samples. Specifically, they evaluate LID for the j-dimensional latent representations of a neural network $f(\boldsymbol{x})$ of a sample $\boldsymbol{x}$ use the L^2 distance

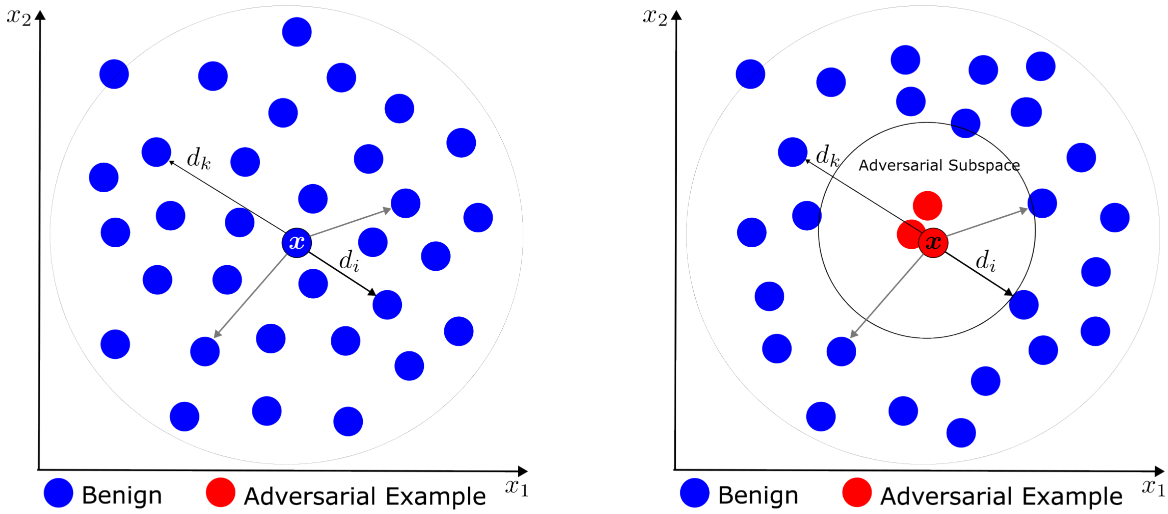
$$d_{\mathfrak{l}}(\boldsymbol{x}, \boldsymbol{y}) = \|f_{\mathfrak{l}}^{1..j}(\boldsymbol{x}) - f_{\mathfrak{l}}^{1..j}(\boldsymbol{y})\|_2 \tag{3.2}$$

for all $\mathfrak{l} \in \mathfrak{L}$ feature maps. They compute a vector of LID values for each sample:

$$\overrightarrow{\mathrm{LID}}(\boldsymbol{x}) = \{\mathrm{LID}_{d_{\mathfrak{l}}}(\boldsymbol{x})\}_{\mathfrak{l}}^{n}. \tag{3.3}$$

Finally, they compute the $\overrightarrow{\mathrm{LID}}(\boldsymbol{x})$ over the training data and adversarial examples generated on the training data, and train a logistic regression classifier[1] to detect adversarial samples.

[1]We are grateful to the authors for releasing their complete source code.



(a) Selected sample $\boldsymbol{x}$ is benign.

(b) Selected sample $\boldsymbol{x}$ is adversrial.

Figure 3.1. Concept of the LID method. An adversarial subspace exists, if the distances are higher as usual from a random selected sample $\boldsymbol{x}$ to k nearest neighbors.

3.2 CONTRIBUTION

In this work, we restrict our investigation to the detection of adversarial images exposed to convolutional neural networks (CNN). We introduce a novel whitebox detector, showing a close-to-perfect detection performance on widely used benchmark settings. Our method is built on the notion that adversarial samples are forming distinct sub-spaces, not only in the input domain but most dominantly in the feature spaces of neural networks [131]. Hence, several prior works have attempted to find quantitative measures for the characterization and identification of such adversarial regions. We investigate the properties of the commonly used local intrinsic dimensionality (LID) and show that a robust identification of adversarial sub-spaces requires (i) an unfolded local representation and (ii) a non-linear separation of these manifolds. We utilize these insights to formulate our novel multiLID descriptor. Extensive experimental evaluations of the proposed approach show that multiLID allows a reliable identification of adversarial samples generated by state-of-the art attacks on CNNs. In summary, our contributions are:

- an analysis of the widely used LID method, which is able to seperate benign from attacked ones.
- novel re-formulation of an unfolded, non-linear multiLID descriptor which allows to improve the detection of adversarial input images in CNN architectures.
- in-depth evaluation of our approach on common benchmark architectures and datasets, showing the superior performance of the proposed method.

3.3 METHOD

We start this section introducing the problem definition. Then, we revise the LID method and suggest our modifications and introduce "multiLID" as novel approach.

3.3.1 Problem Definition

To average the nearest neighbors [7] (see eq. (3.1)) is general accepted in the context of LID. Therefore, our research question is: Can the detection accuracy of adversarial examples be improved, if not averaging over the nearest neighbors?

Thus, we modify the LID and introduce the multiLID as detection method. Adversarial examples detection can be mathematically defined as follows: Let D be a dataset consisting of N pairs $(\boldsymbol{x}_i, y_i)_{i=1}^{N}$, where $\boldsymbol{x}_i$ represents an input image and y_i is the corresponding ground truth label. Let $f : \mathbb{R}^m \to \mathbb{R}^c$ denote a trained neural network model, where m is the input dimensionality (e.g., image dimensions) and c is the number of possible classes.

An adversarial example $\boldsymbol{x}_{\text{adv}}$ for a given input $\boldsymbol{x}$ is generated by applying a perturbation $\boldsymbol{\delta}$ to $\boldsymbol{x}$, such that $\boldsymbol{x}_{\text{adv}} = \boldsymbol{x} + \boldsymbol{\delta}$, where $\boldsymbol{\delta}$ is constrained within a certain perturbation budget (e.g., L^p norm bound). The goal of adversarial examples detection is to identify whether a given input $\boldsymbol{x}_{\text{adv}}$ is an adversarial example or a benign sample, given the model f. Mathematically, an adversarial examples classifier C can be defined as a function $\mathbb{R}^m \to 0, 1$, where: $C(\boldsymbol{x}) = 1$ if $\boldsymbol{x}$ is classified as an adversarial example. $C(\boldsymbol{x}) = 0$ if $\boldsymbol{x}$ is classified as a benign sample.

The goal is to compare the performance of two classifers, LID and multiLID, in accurately identifying adversarial examples. The performance of both classifiers are compared using different metrics exhibits better accuracy in detecting adversarial examples.

3.3.2 Revisiting Local Intrinsic Dimensinality - multiLID

The LID method for adversarial example detection as proposed in [91] was motivated by the MLE estimate for the intrinsic dimension as proposed by [3]. We refer to this original formulation to motivate our proposed multiLID. Let us denote $\mathbb{R}^m, d$ a continuous domain with non-negative distance function d. The continuous intrinsic dimensionality aims to measure the local intrinsic dimensionality of $\mathbb{R}^m$ in terms of the distribution of inter point distances. Thus, we consider for a fixed point $\boldsymbol{x}$ the distribution of distances as a random variable **D** on $[o, +\infty)$ with probability density function f_D and cumulative density function F_D. For samples $\boldsymbol{x}$ drawn from continuous probability distributions, the intrinsic dimensionality is then defined as in [3]:

Definition 1. *Instrinsic Dimensionality (ID). Given a sample $\boldsymbol{x} \in \mathbb{R}^m$, let D be a random variable denoting the distance from $\boldsymbol{x}$ to other data samples. If the cumulative distribution $F(d)$ of* **D** *is positive and continuously differentiable at distance $d > 0$, the ID of $\boldsymbol{x}$ at distance d is given by:*

$$\mathrm{ID}_{\mathbf{D}}(d) \triangleq lim_{\epsilon \to 0} \frac{\log F_{\mathbf{D}}((1+\epsilon)d) - \log F_{\mathbf{D}}(d)}{\log(1+\epsilon)} \tag{3.4}$$

In practice, we are given a fixed number n of samples of x such that we can compute their distances to x in ascending order $d_1 \leq d_2 \leq \cdots \leq d_{n-1}$ with maximum distance w between any two samples. As shown in [3], the log-likelihood of $\mathrm{ID}_{\mathbf{D}}(d)$ for $\boldsymbol{x}$ is then given as

$$n\log\frac{F_{\mathbf{D},w}(w)}{w} + n\log\mathrm{ID}_{\mathbf{D}} + (\mathrm{ID}_{\mathbf{D}} - 1)\sum_{i=1}^{n-1}\log\frac{d_i}{w}. \tag{3.5}$$

The maximum likelihood estimate is then given as

$$\widehat{\mathrm{ID}}_{\mathbf{D}} = -\left(\frac{1}{n}\sum_{i=0}^{n-1}\log\frac{d_i}{w}\right)^{-1} \qquad \text{with} \tag{3.6}$$

$$\widehat{\mathrm{ID}}_{\mathbf{D}} \sim \mathcal{N}\left(\mathrm{ID}_{\mathbf{D}}, \frac{\mathrm{ID}_{\mathbf{D}}^2}{n}\right), \tag{3.7}$$

i.e. the estimate is drawn from a normal distribution with mean $\mathrm{ID}_{\mathbf{D}}$ and its variance decreases linearly with increasing number of samples while it increases quadratically with $\mathrm{ID}_{\mathbf{D}}$. The *local* ID is then an estimate of the ID based on the local neighborhood of $\boldsymbol{x}$, for example based on its k nearest neighbors. This corresponds to equation (3.1). This local approximation has the advantage of allowing for an efficient computation even on a per batch basis as done in [91]. It has the disadvantage that is does not consider the strong variations in variances $\mathrm{ID}_{\mathbf{D}}^2/n$, i.e. the estimates might become arbitrarily poor for large ID if the number of samples is limited. This becomes even more severe as [2] showed that latent representations with large ID are particularly vulnerable to adversarial attacks.
In fig. 3.2, we evaluate the distribution of LID estimates computed for benign and adversarial examples of different attacks on the latent feature representation of a classifier network (see section 3.4). We make the following two observations: (i) the distribution has a rather long tail and is not uni-modal, i.e. we are likely to face rather strong variations in the ID for different latent sub-spaces, (ii) the LID estimates for adversarial examples have the tendency to be higher than the ones for benign examples, (iii) the LID is more informative for some attacks and less informative on others. As a first conclusion, we expect the discrimination between adversarial examples and benign ones to be particularly hard when the tail of the distribution is concerned, i.e. for those benign points with rather large LID that can only be measured at very low confidence according to equation (3.6). Secondly, we expect linear

separation methods based on LID such as suggested by [91] to be unnecessarily weak and third, we expect the choice of the considered layers to have a rather strong influence on the expressiveness of LID for adversarial detection.

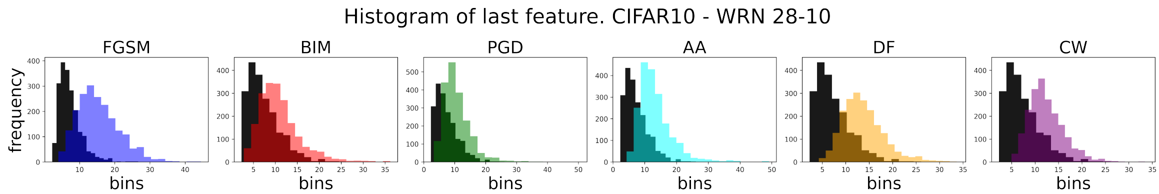


Figure 3.2. Visualization of the LID features from the benign set of samples (black) and different adversarial attacks of 100 samples. The network is WRN 28-10 trained on CIFAR-10 and LID is evaluated on the feature map after the last ReLU activation.

As a remedy, we propose several rather simple improvements:

- We propose to unfold the aggregated LID estimates in equation (3.1) and rather consider the normalized log distances between a sample and its neighbors separately in a feature vector, which we denote multiLID.
- We argue that the deep network layers considered to compute LID or multiLID have to be carefully chosen. An arbitrary choice might yield poor results.
- Instead of using a logistic regression classifier, highly non-linear classifiers such as a random forest should increase LID based discrimination between adversarial and benign samples.

Let us analyze the implications of the LID unfolding in more detail. As argued for example in [91] before, the empirically computed LID can be interpreted as an estimate of the local growth rate similarly to previous generalized expansion models [58, 65]. Thereby, the idea is to deduce the expansion dimension from the volume growth around a sample and the growth rate is estimated by considering probability mass in increasing distances from the sample. Such expansion models, like the LID, are estimated within a local neighborhood around each sample and therefore provide a local view of the data dimensionality [91]. The local ID estimation in eq. (3.1) can be seen as a statistical interpretation of a growth rate estimate. Please refer to [56, 57] for more details.
In practical settings, this statistical estimate not only depends on the considered neighborhood size. In fact, LID is usually evaluated on a mini-batch basis, i.e. the k nearest neighbors are determined within a random sample of points in the latent space. While this setting is necessarily relatively noisy, it offers a larger coverage of the space while considering only few neighbors in every LID evaluation. Specifically, the relative growth rate is aggregated

over potentially large distances within the latent space, when executing the summation in eq. (3.1). We argue that this summation step integrates over potentially very discriminative information since it mixes local information about the growth rate in the direct proximity with more distantly computed growth rates. Therefore, we propose to "unfold" this growth rate estimation. Instead of the aggregated (semi) local ID, we propose to compute for every sample $\boldsymbol{x}$ a feature vector, denoted multiLID, with length k as

$$\overrightarrow{\text{multiLID}_d(\boldsymbol{x})}[i] = -\left(\log \frac{d_i(\boldsymbol{x})}{d_k(\boldsymbol{x})}\right)^{-1}. \tag{3.8}$$

where d is measured using the Euclidean distance. Figure 3.3 visualizes multiLID for 100 benign CIFAR-10 samples and samples that have been perturbed using FGSM. It can easily be seen that there are several characteristic profiles in the multiLID that would be integrated into very similar LID estimates while being discriminative when all k growth ratio samples are considered as a vector. The multiLID defense facilitates to leverage the different characteristic growth rate profiles.

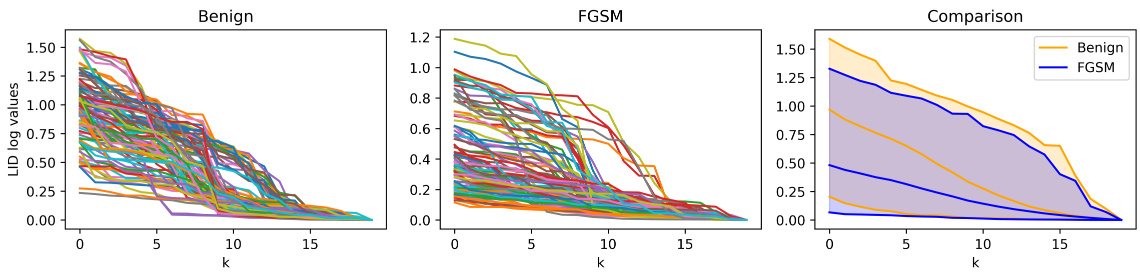


Figure 3.3. Visualization of the LID features from the benign and FGSM set of 100 samples over each k. The network is WRN 28-10 trained on CIFAR-10. The feature values for the nearest neighbors (low values on the x-axis) are significantly higher for the benign dataset. The LID log values are inversely proportional to the distance as shown in eq. (3.8) and fig. 3.1. The plot on the right illustrates mean and standard deviation of the two sets of profiles.

3.4 EXPERIMENTS

In this section, we present experimental results evaluating the effectiveness of the proposed method. We first give a detailed description of the experimental setup. Then, we present our datasets which are attacked by different whitebox attacks. Finally, we compare to other detection methods.

3.4.1 Experimental Setup

To validate our proposed multiLID, we conduct extensive experiments on CIFAR-10, CIFAR-100, and ImageNet. Following the same setup as depicted in fig. 2.3: We train two different models, a WRN 28-10 (wide resnet) [146, 160] and a VGG-16 model [124] on the different datasets. While we use test samples from the original datasets as benign samples, we generate adversarial samples using a variety of adversarial attacks. From benign and adversarial data, we extract the feature maps for different layers, at the output of the ReLU activations. We use a random subset of 2000 samples of this data for each attack method and extract the multiLID features from the feature maps. From this random subset we take a train-test split of 80:20, i.e. we have a training set of 3200 samples (1600 benign, 1600 attacked images) and a balanced test set of 400 images for each attack. This setting is common practice as used in [73, 87, 91]. All experiments were conducted on 3 Nvidia A100 40GB GPUs for ImageNet and 3 Nvidia Titan with 12GB for CIFAR-10 and CIFAR-100.

Layer feature selection per architecture. Following eq. (3.3), for the WRN 28-10 and WRN 51-2, we focus on the ReLU activation layers, whereas in each residual block, we take the last one. This results in 13 activations layer for WRN 28-10 and 17 for WRN 51-2 to compute multiLID representations. This is different from the setting proposed in [153], who propose to use the outputs of the three convolutional blocks. In [91] only simpler network architectures have been considered and the feature maps at the output of every layer are considered to compute LID. For the VGG-16 architecture, according to [52], we take the features of all activation layers, which are again 13 layers in total.

Minibatch size in LID estimation. As motivated in [91], we estimate the multiLID values using a default minibatch size $|\mathcal{B}|$ of 100 with k selected as of 20% of mini batch size [91]. As discussed above and theoretically argued before in [3] the MLE estimator of LID suffers on such small samples, yet, already provides reasonable results when used for adversarial detection [91]. Our proposed multiLID can perform very well in this computationally affordable setting across all datasets.

3.4.2 Datasets

Many of the adversarial training methods ranked on RobustBench[2] are based on the WRN 28-10 [146, 160] architecture. Therefore, we also conduct our evaluation on a baseline WRN 28-10 and train it with benign examples.

- **CIFAR-10:** The WRN 28-10 reaches a test accuracy of 96% and the VGG-16 model

[2] `https://robustbench.github.io`

reaches 72% top-1 accuracy [87] on the test set. We then apply the different attacks on the test set.

- **CIFAR-100:** The procedure is equal to CIFAR-10 dataset. We report a test-accuracy for WRN 28-10 of 83% (VGG-16 reaches 81%) [87] .

- **ImageNet:** The PyTorch library provides a pre-trained WRN 51-2 [160] for ImageNet. As test set, we use the official validation set from ImageNet and reach a validation accuracy of 80%.

3.4.3 Attack Methods

We generate test data from six most commonly used adversarial attacks: FGSM, BIM, PGD(-L^∞), C&W(-L^2), DF(-L^2) and AA, as explained in section 3.1.2. For FGSM, BIM, PGD(-L^∞), and AA, we use the commonly employed perturbation size of $\epsilon = 8/255$, DF is limited to 20 iterations and C&W to 1000 iterations.

3.4.4 Results

In this section, we report our final results of our multiLID method and compare it to competing methods. In table 3.1, we compare the results of the original LID [91] to the results of our proposed multiLID method for both model types, the Wide-ResNets (WRN) and VGG-16 models on the three datasets CIFAR-10, CIFAR-100, and ImageNet. For LID and the proposed multiLID, we extract features from exactly the same layers in the network to facilitate direct comparison. While LID already achieves overall good results the proposed multiLID can even perfectly discriminate between benign and adversarial images on these data in terms of AUC as well as F1 score.

In table 3.2, we further compare the AUC and F1 score, for CIFAR-10 trained on WRN 28-10 to a set of most widely used adversarial defense methods. First, we list the results from [153] for the defenses kernel density (KD), LID and M-D as baselines. According to [153], KD does not show strong results across the attacks, LID and M-D yield a better average performance in their setting. For completeness, we also report the results CD-VAE [153] by showing $R(\boldsymbol{x})$ (which is the reconstruction of a sample x through a β variational auto encoder (β-VAE)). Encoding in such a well-conditioned latent space can help adversarial detection, yet is also time consuming and requires task specific training of the β-VAE.

Our results, when reproducing LID on the same network layers as [153], are reported in the second block of table 3.2. While we can not exactly reproduce the numbers from [153], the resulting AUC and F1-scores are in the same order of magnitude and slightly better in some

Table 3.1. Results. Comparison of the original LID method with our proposed multiLID on different datasets. We report the AUC and F1 score as mean and variance over three evaluations with randomly drawn test samples.

Attacks	CIFAR-10				CIFAR-100				ImageNet	
	WRN 28-10		VGG16		WRN 28-10		VGG16		WRN 50-2	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1
original LID [91]										
FGSM	95.89 ± 0.07	89.46 ± 0.01	87.76 ± 0.21	78.58 ± 0.15	97.71 ± 0.82	92.71 ± 2.17	77.20 ± 1.48	70.08 ± 0.31	71.40 ± 9.07	65.96 ± 4.34
BIM	86.68 ± 0.50	78.50 ± 0.46	87.73 ± 0.68	78.53 ± 0.52	95.53 ± 1.02	87.75 ± 0.93	81.31 ± 3.62	73.73 ± 6.66	94.02 ± 0.27	86.58 ± 0.50
PGD	88.92 ± 0.86	80.11 ± 1.77	84.78 ± 0.68	74.93 ± 1.86	97.76 ± 0.06	91.40 ± 0.09	84.75 ± 1.59	78.04 ± 2.47	95.81 ± 1.00	88.54 ± 3.08
AA	96.49 ± 0.32	90.78 ± 0.22	95.25 ± 0.49	87.35 ± 2.30	99.18 ± 0.03	94.74 ± 0.85	87.02 ± 0.83	78.81 ± 0.23	99.87 ± 0.00	98.01 ± 0.06
DF	94.40 ± 0.07	86.08 ± 3.15	85.93 ± 0.26	75.88 ± 0.74	57.02 ± 0.39	52.59 ± 1.44	54.39 ± 0.03	52.93 ± 0.32	54.62 ± 0.07	49.04 ± 2.06
CW	92.83 ± 0.21	84.33 ± 1.50	83.34 ± 0.47	74.51 ± 0.53	55.07 ± 0.31	53.74 ± 4.31	61.47 ± 0.86	61.99 ± 2.12	54.45 ± 0.05	50.36 ± 3.86
multiLID + improved layer setting + RF or short: **multiLID (ours)**										
FGSM	96.98 ± 0.20	91.19 ± 0.95	90.78 ± 0.38	82.34 ± 1.37	98.55 ± 0.28	94.80 ± 1.32	83.00 ± 0.84	76.89 ± 0.02	79.25 ± 2.75	72.98 ± 0.67
BIM	96.10 ± 0.39	89.93 ± 1.24	94.50 ± 0.26	88.14 ± 0.43	97.88 ± 0.07	91.72 ± 0.09	82.96 ± 0.94	75.42 ± 1.39	94.48 ± 0.18	86.92 ± 0.93
PGD	97.69 ± 0.12	92.81 ± 0.37	92.35 ± 1.46	83.52 ± 3.46	98.76 ± 0.05	94.74 ± 1.14	88.39 ± 0.06	81.42 ± 0.44	96.39 ± 0.11	90.21 ± 0.49
AA	99.45 ± 0.03	96.88 ± 0.00	98.77 ± 0.08	94.85 ± 1.42	99.85 ± 0.00	98.33 ± 0.01	91.25 ± 0.33	83.48 ± 0.17	99.90 ± 0.00	98.83 ± 0.04
DF	97.51 ± 0.12	94.04 ± 0.26	89.37 ± 2.42	84.32 ± 0.83	74.75 ± 0.16	70.18 ± 0.57	73.78 ± 1.12	70.04 ± 0.09	52.93 ± 1.01	52.77 ± 1.89
CW	97.92 ± 0.01	96.00 ± 0.11	89.75 ± 0.21	85.27 ± 0.85	70.10 ± 1.33	67.77 ± 0.85	76.15 ± 0.20	71.59 ± 0.20	53.37 ± 0.01	52.04 ± 0.17

cases. In this setting, LID performs slightly worse than the competing methods M-D and $\mathrm{SD}_{\mathrm{BlackBox}}$ and $\mathrm{SD}_{\mathrm{WhiteBox}}$ [52].
We ablate on our different changes towards the full multiLID in the third block. When replacing LID by the unfolded features as in eq. (3.8) we already achieve results above 98% F1 score in all settings. Defending against BIM is hardest. The next line ablates on the employed feature maps. When replacing the convolutional features used in [153][3] by the last ReLU outputs in every block, we observe a boost in performance even on the plain LID features. Combining these two lead to almost perfect results on the CIFAR-10 dataset. Results for other datasets are in table 3.3. F1-scores and AUC scores classifer results are lifted, on this feature basis, using a random forest classifier instead of the logistic regression. We refer to this setting as multiLID in all other tables including table 3.1.

3.5 ABLATION STUDY

In this section we give insights on the different factors affecting our approach. We investigate the importance of the activation maps the features are extracted from as well as the number of multiLID features that are needed to reach good classification performance. An ablation on the number of considered neighbors as well as on the attack strength in terms of ϵ is also provided.

[3]Assumption of CD-VAE LID layers taken from `https://github.com/kai-wen-yang/CD-VAE/blob/a33b5070d5d936396d51c8c2e7dedd62351ee5b2/detection/models/wide_resnet.py#L86`.

Table 3.2. Comparison of multiLID with the state-of-the-art on CIFAR-10.

Defenses	FGSM		BIM		PGD		CW	
	TNR	**AUC**	**TNR**	**AUC**	**TNR**	**AUC**	**TNR**	**AUC**
CIFAR-10 on WRN 28-10								
Results reported by [153]								
KD	42.38	85.74	74.54	94.82	73.12	94.59	73.33	94.75
KD (R($\boldsymbol{x}$))	57.10	89.69	96.79	99.27	96.56	99.30	94.67	98.73
LID	69.05	93.60	77.73	95.20	71.52	93.19	74.98	94.32
LID (R($\boldsymbol{x}$))	92.60	98.59	86.42	97.29	87.54	97.57	76.42	95.10
M-D	94.91	98.69	88.33	97.66	77.23	95.38	86.30	97.36
M-D (R($\boldsymbol{x}$))	99.68	99.36	98.92	99.74	99.13	99.79	98.94	99.68
Competing Methods								
M-D [91]	97.37	99.34	98.16	99.61	97.37	99.66	91.58	96.54
$SD_{BlackBox}$ [52]	95.79	99.87	92.63	99.83	92.11	99.29	53.68	63.23
$SD_{WhiteBox}$ [52]	99.47	100.00	96.32	99.99	95.79	99.97	84.47	96.89
LID, settings from [153]	87.25	84.82	85.02	81.07	81.61	89.00	85.89	90.48
Ours								
multiLID, settings from [86] + LR	85.89	95.02	83.21	95.56	93.93	98.00	91.07	97.05
multiLID, settings from [86] + RF	87.50	94.01	85.89	96.74	94.64	98.96	92.14	97.15
LID, improved layer setting	90.18	96.62	93.21	98.18	85.89	90.48	87.50	93.36
multiLID + improved layer setting + LR	90.18	96.62	81.43	93.83	86.61	96.44	93.21	98.18
multiLID + improved layer setting + RF	90.54	96.33	85.71	94.75	94.64	98.96	92.14	97.15

3.5.1 Impact of non-linear Classification

In this section, we compare the methods from the last two lines of table 3.2 in more detail and for all three datasets. The results are reported in table 3.1. While the simple LR classifier already achieves very high AUC and F1 scores on multiLID for all attacks and datasets, RF can further push the performance.

3.5.2 Feature Importance

The feature importance (variable importance) of the random forest describes the relevant features for the detection. In fig. 3.4, we plot the feature importance for the aggregated LID features of WRN 28-10 trained on a CIFAR-10 dataset. The feature importance represents the importance of the selected ReLU layers (see [73]) in increasing order. The last features/layers shows higher importance. For the attack FGSM the 3rd and last feature can be very relevant.

Table 3.3. Results of using multiLID. Comparison of LR and RF classifier on different datasets. Comparison to table 3.1 which uses LR. The minibatch size is $|\mathcal{B}| = 100$ and the number of neighbors $k = 20$ according to section 3.4.1.

Attacks	CIFAR-10				CIFAR-100				ImageNet	
	WRN 28-10		VGG16		WRN 28-10		VGG16		WRN 50-2	
	AUC	F1	AUC	F1	AUC	F1	AUC	F1	AUC	F1
multiLID + LR (ours)										
FGSM	97.64 ± 0.16	91.58 ± 1.40	93.24 ± 0.05	86.39 ± 1.23	99.10 ± 0.05	95.06 ± 0.33	85.32 ± 0.02	78.41 ± 0.84	80.65 ± 3.30	73.15 ± 3.08
BIM	95.44 ± 0.36	90.40 ± 0.28	93.59 ± 0.88	87.02 ± 1.27	98.29 ± 0.04	92.84 ± 0.14	84.76 ± 0.72	77.76 ± 3.38	96.78 ± 0.02	90.54 ± 0.32
PGD	96.60 ± 0.36	91.87 ± 1.34	90.91 ± 0.95	82.81 ± 0.56	98.89 ± 0.00	95.26 ± 0.35	89.10 ± 0.18	82.24 ± 0.09	97.44 ± 0.06	92.28 ± 0.01
AA	99.00 ± 0.06	95.74 ± 0.09	98.25 ± 0.24	93.70 ± 0.87	99.89 ± 0.00	98.27 ± 0.04	91.27 ± 0.71	83.66 ± 0.74	99.95 ± 0.00	98.76 ± 0.04
DF	97.93 ± 0.03	94.06 ± 0.04	89.54 ± 1.02	84.34 ± 1.02	75.99 ± 0.73	70.77 ± 1.79	70.76 ± 4.48	67.15 ± 2.55	54.49 ± 0.18	53.06 ± 0.76
CW	97.86 ± 0.03	94.88 ± 0.23	89.79 ± 0.79	84.34 ± 1.21	71.01 ± 1.23	66.03 ± 0.09	70.79 ± 0.90	67.75 ± 2.29	54.51 ± 0.97	53.87 ± 0.66
multiLID + RF (ours)										
FGSM	96.84 ± 0.14	90.90 ± 0.70	91.04 ± 0.24	82.97 ± 1.32	98.47 ± 0.31	94.31 ± 2.28	82.56 ± 0.77	76.61 ± 1.21	79.52 ± 2.09	72.85 ± 0.78
BIM	96.11 ± 0.28	89.61 ± 1.77	94.55 ± 0.33	88.37 ± 0.99	97.74 ± 0.12	91.48 ± 0.14	83.14 ± 0.29	74.94 ± 0.13	94.33 ± 0.26	87.43 ± 2.04
PGD	97.50 ± 0.11	92.39 ± 0.13	92.18 ± 1.07	84.00 ± 2.62	98.77 ± 0.06	94.82 ± 1.08	88.36 ± 0.09	81.40 ± 0.50	96.55 ± 0.19	90.79 ± 0.21
AA	99.54 ± 0.01	96.96 ± 0.05	98.61 ± 0.02	94.74 ± 1.41	99.87 ± 0.00	98.20 ± 0.02	91.25 ± 0.36	83.76 ± 0.87	99.88 ± 0.01	99.08 ± 0.01
DF	97.47 ± 0.21	94.20 ± 0.12	89.04 ± 1.61	84.43 ± 1.11	74.85 ± 0.58	70.11 ± 1.71	73.44 ± 0.66	69.09 ± 0.96	53.41 ± 2.63	52.41 ± 1.36
CW	97.91 ± 0.04	95.76 ± 0.20	89.67 ± 0.52	85.32 ± 0.49	69.80 ± 0.65	66.61 ± 0.03	76.12 ± 0.07	72.66 ± 0.38	53.80 ± 2.39	52.26 ± 3.19

3.5.3 Investigation of the multiLID Features

Following the eq. (3.1), all neighbors k are used for the classification. This time, we investigate the performance of the binary classifier logistic regression over the full multiLID features. For example, in fig. 3.4 we consider 13 layers and the aggregated ID features for each. Thus, the number of multiLID features per sample can be calculated as #layers $\times$ k which yields 260 features for $k = 20$. In fig. 3.5, we visualize the AUC according to the length of the LID feature vectors, when successively more features are used according to their random forest feature importance. On ImageNet, it can be seen that DF and C&W need the full length of these LID feature vectors to achieve the highest AUC scores. The observation, that the attacks DF and C&W are more effectively are also reported in [87]. Using a non-linear classifier on these very discriminant features, we can even achieve perfect F1 scores (see section 3.5.1).

3.5.4 Impact of the Number of Neighbors

We train the LID with the APGD-CE attack from the AutoAttack benchmark with different epsilons (L^∞ and L^2). In fig. 3.6, we compare RF and LR on different norms. Random Forest succeeds on all epsilon sizes[4] on both norms. On smaller perturbation sizes the LR classifier AUC score fall. On the optimal perturbation size ($L^\infty : \epsilon = 8/255$ and $L^2 : \epsilon = 0.5$) the LR shows its best AUC scores. The RF classifier gives us outstanding results over

[4] Perturbed images would round the adversarial changes to the next of 256 available bins in commonly used 8-bit per channel image encodings.

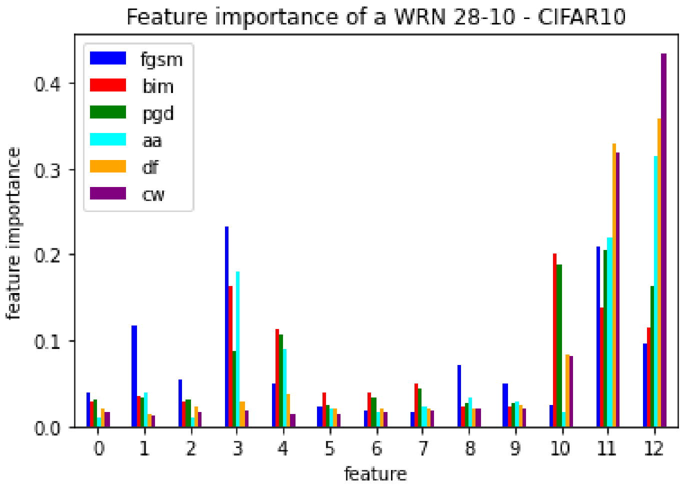


Figure 3.4. Feature importance. Increasing order according to the activation function layers (feature) from WRN 28-10 trained on CIFAR-10. The most relevant features are in the last ReLU layers.

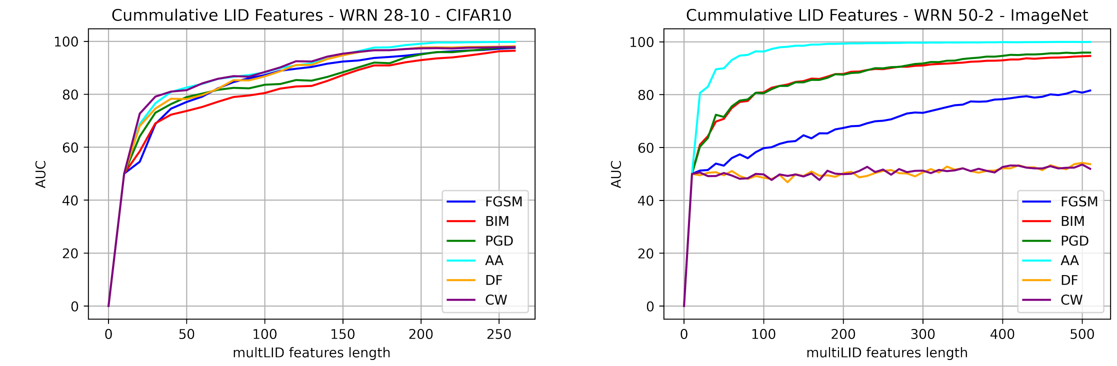
the LR. Moreover, to save computation time, $k = 3$ neighbors would be enough for high accuracy.

3.5.5 Impact of the Number of Neighbors and Attack Strength ϵ.

We train LID and multiLID with the APGD-CE attack from the AutoAttack benchmark for different perturbation magnitudes, i.e. using different epsilons (L^∞ and L^2). On smaller perturbation sizes the logistic regression (LR) classifier AUC scores are dropping, which is to be expected. On the most commonly used perturbation sizes ($L^\infty : \epsilon = 8/255$ and $L^2 : \epsilon = 0.5$) LID shows its best AUC scores. The multiLID classifier provides superior results over LID in all cases. Moreover, to save computation time for multiLID, $k = 10$ neighbors would be enough for high accuracy adversarial detection.

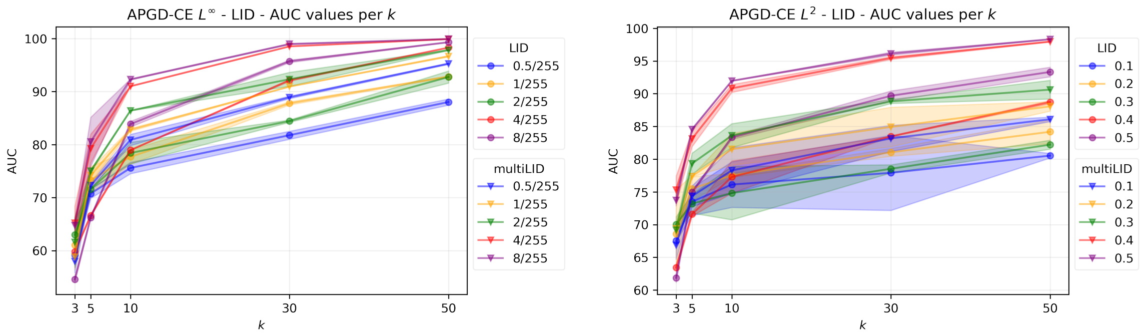
3.5.6 Attack Transferability

In this section, we evaluate the attack transferability of our models, for LID in table 3.4 and multiLID in table 3.4. In case of real world applications, the attack methods might be unknown and thus it is a desired feature that a detector trained on one attack method performs well for a different attack. We evaluate in both directions. The random forest (RF) classifier shows significantly higher transferability on both LID and multiLID. The attack tuples (PGD $\leftrightarrow$ BIM), (PGD $\leftrightarrow$ AA), (AA $\leftrightarrow$ BIM), and (DF $\leftrightarrow$ CW) yield very high bidirectional attack transferability. However, the experiments also show that not all combinations can be transferred successfully, e.g. (FGSM $\leftrightarrow$ CW) in ImageNet. This leaves



(a) Cummulative of all attacks on CIFAR-10. **(b)** Cummulative of all attacks on ImageNet.

Figure 3.5. Cummulative features used for the LR classifier. The x-axis describes the length of the used feature vectors. The y-axis reports the AUC reached by using the most important features out of the full vector, sorted by RF feature importance.



(a) The attack APGD-CE L^∞ evaluated on different epsilons and neighbors.

(b) The attack APGD-CE L^2 evaluated on different epsilons and neighbors.

Figure 3.6. Ablation study of LID and multiLID detection rates by using different k on the APGD-CE (L^2, L^∞) attack and different epsilon sizes.

room for further research.

Table 3.4. Attack transfer - LID. Rows with the target μ give the average transfer rates from one attack to all others. Random forest (RF) in table 3.5 shows higher accuracy (ACC) for the attack transfer as RF.

LID											
Attacks		**CIFAR-10**				**CIFAR-100**				**ImageNet**	
		WRN 28-10		**VGG16**		**WRN 28-10**		**VGG16**		**WRN 50-2**	
from	**to**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**
Logistic regression (LR)											
FGSM	**BIM**	62.3 ± 102.5	55.7 ± 35.5	62.9 ± 118.6	57.3 ± 33.1	66.8 ± 191.5	62.3 ± 107.5	79.2 ± 0.8	64.4 ± 3.9	76.3 ± 0.2	62.1 ± 1.2
FGSM	**PGD**	78.3 ± 65.7	61.2 ± 208.8	86.9 ± 0.4	77.7 ± 0.5	95.6 ± 1.0	87.2 ± 1.8	66.1 ± 273.0	61.1 ± 217.0	60.6 ± 265.3	58.3 ± 137.2
FGSM	**AA**	78.9 ± 68.6	60.8 ± 229.0	86.9 ± 0.3	79.5 ± 0.5	96.5 ± 0.2	88.5 ± 0.7	66.6 ± 321.8	61.7 ± 268.3	62.0 ± 319.2	59.4 ± 176.1
FGSM	**DF**	79.4 ± 45.2	60.4 ± 191.4	86.4 ± 0.0	74.0 ± 1.3	87.4 ± 2.2	74.3 ± 2.2	66.5 ± 284.8	60.8 ± 204.7	61.7 ± 280.9	58.4 ± 136.7
FGSM	**CW**	87.8 ± 0.9	79.4 ± 0.0	67.5 ± 70.9	55.0 ± 50.0	67.2 ± 99.5	54.8 ± 46.1	72.5 ± 227.6	61.7 ± 270.7	92.8 ± 0.3	82.9 ± 0.4
FGSM	μ	69.5 ± 82.7	60.4 ± 36.3	77.5 ± 121.1	69.1 ± 113.1	78.2 ± 142.0	70.0 ± 134.9	76.3 ± 122.6	65.6 ± 107.3	77.5 ± 79.8	66.8 ± 73.4
BIM	**FGSM**	85.5 ± 0.8	77.8 ± 0.2	66.3 ± 43.3	54.6 ± 41.7	66.5 ± 72.4	54.7 ± 43.6	72.0 ± 192.2	60.8 ± 233.3	94.0 ± 0.2	86.6 ± 1.0
BIM	**PGD**	81.0 ± 0.6	74.0 ± 0.4	78.8 ± 0.9	72.0 ± 1.0	92.0 ± 1.7	84.6 ± 2.6	78.8 ± 0.1	69.2 ± 0.4	75.9 ± 0.4	66.8 ± 0.1
BIM	**AA**	81.3 ± 0.6	72.9 ± 1.4	85.0 ± 0.2	75.5 ± 0.5	95.6 ± 0.4	87.9 ± 1.1	80.5 ± 0.0	71.6 ± 0.1	78.3 ± 0.4	69.0 ± 2.0
BIM	**DF**	81.6 ± 0.1	73.5 ± 0.6	87.6 ± 0.8	79.5 ± 0.5	95.5 ± 0.3	87.3 ± 1.6	79.2 ± 0.9	71.4 ± 0.5	77.2 ± 0.8	68.7 ± 0.3
BIM	**CW**	82.7 ± 0.1	71.2 ± 0.4	86.7 ± 0.4	75.4 ± 0.4	84.1 ± 0.3	71.8 ± 0.5	80.7 ± 1.4	67.1 ± 0.1	78.2 ± 1.1	63.2 ± 0.3
BIM	μ	76.8 ± 61.8	66.9 ± 63.9	81.3 ± 0.8	73.3 ± 0.9	84.1 ± 0.3	75.4 ± 1.0	84.2 ± 0.6	76.1 ± 0.7	82.5 ± 0.7	69.7 ± 0.4
PGD	**FGSM**	80.7 ± 0.2	72.6 ± 0.4	83.6 ± 0.7	76.1 ± 2.6	80.5 ± 0.4	73.8 ± 0.8	93.8 ± 0.1	85.3 ± 0.1	82.9 ± 0.5	74.5 ± 0.1
PGD	**BIM**	79.8 ± 0.1	72.6 ± 0.5	84.2 ± 0.4	77.4 ± 1.2	81.1 ± 0.2	74.6 ± 1.8	94.0 ± 0.1	85.2 ± 0.2	86.3 ± 0.2	77.5 ± 0.2
PGD	**AA**	90.1 ± 10.2	78.8 ± 10.5	91.0 ± 4.2	80.3 ± 15.4	94.8 ± 4.8	87.6 ± 9.1	54.5 ± 0.2	52.1 ± 0.1	52.1 ± 0.1	50.7 ± 0.3
PGD	**DF**	94.2 ± 2.9	84.1 ± 8.8	97.7 ± 0.1	91.9 ± 1.1	99.3 ± 0.0	94.8 ± 0.2	52.4 ± 1.1	49.2 ± 0.3	50.9 ± 0.2	48.9 ± 0.1
PGD	**CW**	93.6 ± 5.8	77.2 ± 48.2	95.2 ± 0.9	87.3 ± 1.8	99.3 ± 0.0	94.9 ± 0.2	52.1 ± 1.0	49.7 ± 0.1	51.2 ± 0.2	49.7 ± 0.3
PGD	μ	84.3 ± 0.4	76.5 ± 0.8	85.1 ± 0.2	77.5 ± 0.8	76.5 ± 3.9	69.9 ± 7.1	78.9 ± 0.9	73.8 ± 2.1	78.3 ± 1.6	71.7 ± 10.1
AA	**FGSM**	93.4 ± 2.7	71.2 ± 15.6	94.4 ± 1.2	80.8 ± 8.5	97.0 ± 0.1	87.4 ± 2.1	52.2 ± 0.9	50.0 ± 0.0	50.6 ± 0.0	49.9 ± 0.1
AA	**BIM**	93.3 ± 1.2	78.5 ± 0.8	40.9 ± 114.9	48.2 ± 40.9	29.2 ± 128.7	41.1 ± 42.8	20.8 ± 63.1	37.1 ± 12.8	52.5 ± 0.0	51.5 ± 0.2
AA	**PGD**	87.7 ± 2.5	72.8 ± 4.2	39.3 ± 3.5	42.9 ± 3.4	30.2 ± 5.3	38.3 ± 9.7	20.0 ± 3.6	33.9 ± 4.4	54.1 ± 0.6	52.6 ± 0.2
AA	**DF**	77.7 ± 0.9	70.3 ± 0.0	79.4 ± 0.5	72.9 ± 0.6	83.5 ± 0.4	76.4 ± 0.2	53.3 ± 0.2	50.5 ± 0.2	54.7 ± 1.1	52.0 ± 0.4
AA	**CW**	70.6 ± 0.3	64.2 ± 1.0	83.0 ± 2.4	75.9 ± 3.1	86.3 ± 0.2	79.1 ± 0.4	52.8 ± 0.7	50.2 ± 0.6	56.8 ± 0.4	51.3 ± 1.5
AA	μ	77.5 ± 1.0	67.9 ± 5.3	47.3 ± 61.6	51.3 ± 19.5	46.2 ± 3.1	48.1 ± 4.4	69.7 ± 0.6	64.4 ± 0.3	69.9 ± 0.8	64.1 ± 1.3
DF	**FGSM**	69.5 ± 0.3	64.0 ± 0.7	80.3 ± 1.3	72.6 ± 4.0	83.7 ± 1.0	77.4 ± 0.2	52.5 ± 0.5	50.6 ± 1.6	58.8 ± 0.7	51.8 ± 1.6
DF	**BIM**	71.9 ± 0.8	62.2 ± 0.6	80.3 ± 4.8	71.9 ± 4.5	80.6 ± 1.8	71.6 ± 2.7	53.0 ± 0.2	49.6 ± 1.1	54.3 ± 0.1	49.6 ± 1.2
DF	**PGD**	68.4 ± 2.4	64.1 ± 3.4	72.5 ± 1.0	65.4 ± 0.1	74.4 ± 0.6	68.6 ± 0.1	79.3 ± 12.4	70.0 ± 11.2	55.6 ± 0.2	54.1 ± 0.7
DF	**AA**	61.9 ± 2.4	58.7 ± 2.3	73.7 ± 11.5	66.1 ± 2.8	82.9 ± 7.1	71.1 ± 2.7	77.5 ± 10.9	65.8 ± 1.8	52.5 ± 0.3	52.0 ± 0.3
DF	**CW**	70.1 ± 10.0	64.5 ± 5.7	78.4 ± 2.4	71.5 ± 1.8	94.2 ± 5.8	77.5 ± 6.0	51.5 ± 0.1	51.1 ± 0.9	51.3 ± 0.0	50.9 ± 0.0
DF	μ	69.0 ± 0.8	63.3 ± 1.6	68.0 ± 1.5	61.0 ± 2.0	70.0 ± 3.3	64.4 ± 3.1	69.7 ± 6.5	62.7 ± 2.0	69.1 ± 3.7	63.1 ± 2.9
CW	**FGSM**	56.5 ± 1.1	51.3 ± 0.4	93.8 ± 1.3	87.0 ± 2.0	99.9 ± 0.0	95.8 ± 0.2	54.2 ± 0.1	51.6 ± 0.1	54.1 ± 0.1	50.9 ± 0.1
CW	**BIM**	59.0 ± 2.3	51.2 ± 0.3	91.8 ± 2.7	82.2 ± 2.6	100.0 ± 0.0	97.0 ± 0.2	54.1 ± 0.0	51.0 ± 0.1	53.9 ± 0.0	51.3 ± 0.1
CW	**PGD**	52.7 ± 1.0	49.8 ± 0.2	89.5 ± 1.8	62.9 ± 3.4	88.0 ± 1.8	63.6 ± 1.2	53.4 ± 0.0	50.2 ± 0.0	53.1 ± 0.0	50.0 ± 0.0
CW	**AA**	57.4 ± 3.1	54.8 ± 1.1	92.6 ± 0.6	77.6 ± 0.7	94.9 ± 0.0	78.2 ± 1.0	99.8 ± 0.0	79.8 ± 0.4	54.2 ± 0.0	52.5 ± 0.1
CW	**DF**	56.2 ± 0.0	54.7 ± 0.9	92.5 ± 2.2	75.6 ± 2.0	94.5 ± 0.9	77.4 ± 2.6	99.8 ± 0.0	79.1 ± 2.7	54.4 ± 0.1	53.0 ± 0.5
CW	μ	71.7 ± 0.5	67.3 ± 0.6	71.8 ± 1.0	66.5 ± 0.7	67.3 ± 0.9	55.3 ± 1.0	79.8 ± 0.8	68.6 ± 0.7	79.5 ± 0.7	68.0 ± 1.7

Table 3.5. Attack transfer LID. Rows with the target μ give the average transfer rates from one attack to all others. Random forest (RF) shows higher accuracy (ACC) for the attack transfer as LR in table 3.4.

LID											
Attacks		CIFAR-10				CIFAR-100				ImageNet	
		WRN 28-10		VGG16		WRN 28-10		VGG16		WRN 50-2	
from	**to**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**
Random forest (RF)											
FGSM	**BIM**	62.3 ± 166.8	57.9 ± 79.1	61.0 ± 220.7	55.7 ± 100.7	68.5 ± 243.0	62.4 ± 198.1	79.6 ± 0.2	66.5 ± 2.0	75.7 ± 0.6	64.9 ± 7.3
FGSM	**PGD**	54.4 ± 498.7	54.1 ± 238.9	90.0 ± 1.1	82.3 ± 3.9	94.8 ± 1.4	88.0 ± 2.0	64.4 ± 133.6	59.8 ± 61.7	59.1 ± 130.7	56.0 ± 49.3
FGSM	**AA**	55.6 ± 535.7	55.5 ± 355.3	87.3 ± 0.1	79.6 ± 0.2	95.6 ± 0.9	89.2 ± 1.0	66.3 ± 216.2	61.1 ± 141.7	61.1 ± 224.0	57.4 ± 115.8
FGSM	**DF**	58.3 ± 499.3	57.9 ± 234.0	86.3 ± 1.4	73.2 ± 0.4	88.7 ± 5.8	77.7 ± 5.2	69.3 ± 199.5	62.0 ± 104.3	62.8 ± 248.7	57.0 ± 91.6
FGSM	**CW**	86.5 ± 0.9	78.3 ± 0.5	62.5 ± 102.5	54.7 ± 44.2	66.6 ± 100.0	56.2 ± 76.1	71.5 ± 276.0	63.3 ± 352.0	93.4 ± 0.3	87.3 ± 0.1
FGSM	μ	69.4 ± 126.3	61.5 ± 77.4	72.5 ± 153.1	68.0 ± 71.2	73.2 ± 195.4	68.5 ± 122.8	73.1 ± 191.0	65.6 ± 87.1	76.1 ± 96.0	68.0 ± 94.6
BIM	**FGSM**	84.5 ± 0.4	77.3 ± 0.4	53.9 ± 231.5	55.5 ± 61.6	59.2 ± 230.5	56.8 ± 93.4	66.1 ± 418.1	62.9 ± 335.4	94.6 ± 0.1	89.2 ± 0.8
BIM	**PGD**	79.4 ± 0.1	70.5 ± 0.5	75.6 ± 0.6	67.6 ± 0.4	88.2 ± 3.5	79.5 ± 0.5	76.1 ± 0.0	67.3 ± 0.9	73.4 ± 0.2	65.5 ± 0.1
BIM	**AA**	75.4 ± 0.5	64.9 ± 0.1	90.3 ± 0.5	82.0 ± 0.7	97.6 ± 0.2	91.5 ± 0.1	77.1 ± 2.4	66.8 ± 5.1	74.8 ± 1.3	62.8 ± 1.4
BIM	**DF**	74.9 ± 0.4	66.2 ± 0.1	91.9 ± 1.2	84.3 ± 1.2	97.4 ± 0.1	89.8 ± 0.4	75.0 ± 3.1	65.5 ± 3.7	73.5 ± 0.2	63.3 ± 0.4
BIM	**CW**	79.3 ± 0.1	64.5 ± 0.7	90.9 ± 1.2	81.5 ± 0.3	88.4 ± 0.3	78.3 ± 0.3	80.2 ± 1.0	65.3 ± 0.8	77.7 ± 0.2	60.7 ± 0.9
BIM	μ	71.7 ± 176.1	68.3 ± 98.3	78.5 ± 0.9	70.1 ± 0.5	83.0 ± 1.0	73.6 ± 1.5	82.5 ± 1.0	73.8 ± 1.2	83.3 ± 0.6	70.0 ± 0.6
PGD	**FGSM**	78.8 ± 0.7	71.7 ± 0.8	81.9 ± 0.7	75.4 ± 0.8	78.9 ± 0.7	72.8 ± 2.2	91.5 ± 0.2	82.6 ± 0.3	83.3 ± 0.7	76.8 ± 0.1
PGD	**BIM**	76.4 ± 0.8	71.8 ± 0.7	79.8 ± 0.1	74.2 ± 0.7	77.5 ± 0.0	72.2 ± 0.2	88.1 ± 3.6	82.1 ± 1.6	85.5 ± 0.0	79.5 ± 0.2
PGD	**AA**	80.7 ± 11.2	68.1 ± 1.2	79.0 ± 11.8	67.1 ± 3.4	83.3 ± 3.7	70.4 ± 0.4	53.4 ± 0.1	52.6 ± 0.0	51.7 ± 0.2	51.2 ± 0.1
PGD	**DF**	89.7 ± 0.8	73.7 ± 7.4	97.7 ± 0.0	92.3 ± 0.4	99.1 ± 0.0	94.6 ± 0.3	52.1 ± 2.5	48.7 ± 0.7	50.5 ± 0.0	48.8 ± 0.1
PGD	**CW**	90.5 ± 2.7	67.9 ± 38.9	95.2 ± 0.4	88.7 ± 0.3	99.2 ± 0.0	95.2 ± 0.6	49.9 ± 0.8	49.0 ± 0.2	49.0 ± 0.2	48.7 ± 0.1
PGD	μ	82.9 ± 0.6	75.9 ± 0.8	81.5 ± 0.9	75.9 ± 0.7	69.6 ± 5.4	61.9 ± 1.0	77.8 ± 0.7	71.6 ± 1.8	76.8 ± 0.8	69.9 ± 8.0
AA	**FGSM**	89.0 ± 12.1	58.6 ± 11.9	94.8 ± 0.5	86.3 ± 0.4	97.5 ± 0.1	91.6 ± 0.2	49.7 ± 0.5	49.2 ± 0.0	48.8 ± 0.6	49.3 ± 0.1
AA	**BIM**	70.3 ± 9.3	64.0 ± 7.3	20.1 ± 7.3	34.8 ± 0.1	14.9 ± 0.5	30.6 ± 4.1	10.0 ± 3.5	29.1 ± 2.1	58.7 ± 0.7	56.3 ± 1.3
AA	**PGD**	59.4 ± 5.8	57.3 ± 7.2	22.9 ± 16.7	32.9 ± 4.2	19.1 ± 17.7	29.9 ± 4.6	14.2 ± 25.7	27.5 ± 2.7	60.7 ± 1.9	58.0 ± 2.2
AA	**DF**	73.0 ± 0.7	66.8 ± 0.2	77.7 ± 0.8	71.4 ± 0.7	78.3 ± 0.9	71.3 ± 1.3	55.7 ± 0.2	53.5 ± 0.2	56.6 ± 0.3	54.7 ± 1.5
AA	**CW**	70.7 ± 1.9	63.0 ± 3.9	85.3 ± 1.8	78.5 ± 3.7	89.4 ± 0.6	80.0 ± 0.7	53.1 ± 0.0	51.4 ± 1.2	57.9 ± 0.4	53.9 ± 0.3
AA	μ	76.0 ± 2.8	67.0 ± 2.5	34.8 ± 4.3	42.9 ± 3.0	35.3 ± 13.6	41.1 ± 4.2	68.3 ± 0.6	63.5 ± 0.8	71.3 ± 1.0	65.4 ± 2.0
DF	**FGSM**	70.5 ± 1.3	59.8 ± 0.4	80.5 ± 3.8	72.4 ± 3.8	88.4 ± 0.8	80.2 ± 0.8	53.7 ± 2.0	52.1 ± 1.2	57.8 ± 0.2	54.2 ± 0.9
DF	**BIM**	70.3 ± 0.9	60.5 ± 0.2	82.4 ± 5.2	73.6 ± 6.0	85.7 ± 1.6	77.8 ± 2.7	52.8 ± 0.0	50.1 ± 0.7	56.8 ± 0.3	53.0 ± 0.9
DF	**PGD**	58.8 ± 0.6	56.1 ± 0.4	51.4 ± 2.1	49.9 ± 3.5	56.0 ± 2.4	54.2 ± 2.5	46.6 ± 3.7	47.1 ± 0.7	60.9 ± 2.0	57.4 ± 0.7
DF	**AA**	58.4 ± 2.4	56.7 ± 1.0	62.2 ± 3.4	60.1 ± 1.3	67.8 ± 3.0	63.4 ± 1.9	56.2 ± 7.0	55.0 ± 9.5	59.0 ± 0.1	56.5 ± 0.1
DF	**CW**	59.4 ± 5.9	55.9 ± 2.0	62.1 ± 0.3	58.5 ± 6.0	56.3 ± 7.1	52.9 ± 5.7	50.6 ± 0.1	50.4 ± 1.2	50.5 ± 0.1	50.5 ± 0.1
DF	μ	70.2 ± 1.6	63.7 ± 1.4	69.6 ± 1.6	63.0 ± 2.1	54.7 ± 2.2	52.9 ± 1.6	60.7 ± 3.2	58.3 ± 2.8	55.8 ± 2.7	53.6 ± 3.0
CW	**FGSM**	54.7 ± 3.1	52.8 ± 0.5	92.9 ± 1.2	85.8 ± 1.4	99.4 ± 0.1	94.6 ± 0.1	53.7 ± 0.1	51.4 ± 0.1	53.0 ± 0.0	50.8 ± 0.3
CW	**BIM**	55.3 ± 5.2	52.0 ± 0.8	91.3 ± 1.7	82.7 ± 3.9	99.7 ± 0.0	96.0 ± 0.1	53.6 ± 0.2	51.0 ± 0.0	53.1 ± 0.3	51.4 ± 0.2
CW	**PGD**	48.3 ± 1.9	49.8 ± 0.0	85.5 ± 4.0	64.9 ± 5.3	88.5 ± 7.5	73.0 ± 1.4	51.3 ± 0.2	50.3 ± 0.0	52.0 ± 1.9	50.2 ± 0.0
CW	**AA**	54.1 ± 2.4	53.0 ± 1.0	71.7 ± 4.9	65.7 ± 2.8	73.5 ± 27.9	66.0 ± 6.0	67.6 ± 140.4	62.3 ± 90.2	53.7 ± 1.8	52.7 ± 0.3
CW	**DF**	52.3 ± 1.5	51.9 ± 1.3	72.1 ± 0.3	65.5 ± 1.8	79.6 ± 4.6	68.6 ± 1.0	80.8 ± 124.5	68.3 ± 44.4	53.7 ± 0.7	53.0 ± 1.1
CW	μ	70.7 ± 0.9	67.1 ± 0.5	70.6 ± 1.5	66.6 ± 1.0	65.1 ± 3.1	57.6 ± 1.4	64.1 ± 35.5	59.9 ± 20.1	67.7 ± 26.3	61.4 ± 9.9

Table 3.6. Attack transfer multiLID. Rows with the target μ give the average transfer rates from one attack to all others. The full multiLID with Random forest (RF) in table 3.7 shows significantly better accuracy (ACC) for the attack transfer as LR.

multiLID											
Attacks		**CIFAR-10**				**CIFAR-100**				**ImageNet**	
		WRN 28-10		**VGG16**		**WRN 28-10**		**VGG16**		**WRN 50-2**	
from	**to**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**
Logistic regression (LR)											
FGSM	**BIM**	74.9 ± 62.2	59.5 ± 164.0	90.4 ± 1.6	76.5 ± 33.9	91.0 ± 28.8	76.1 ± 249.5	83.0 ± 4.8	71.7 ± 10.6	79.1 ± 1.5	68.0 ± 5.0
FGSM	**PGD**	42.4 ± 1244.3	61.3 ± 292.9	45.2 ± 1322.9	64.1 ± 373.6	69.2 ± 1808.0	78.7 ± 410.9	32.5 ± 1507.4	58.9 ± 248.6	30.3 ± 1279.1	55.3 ± 197.6
FGSM	**AA**	93.7 ± 2.5	86.0 ± 4.5	50.2 ± 976.9	63.3 ± 334.6	70.0 ± 1672.5	79.7 ± 438.1	88.0 ± 0.5	80.0 ± 0.0	83.7 ± 4.9	76.1 ± 0.0
FGSM	**DF**	69.1 ± 1144.7	72.9 ± 286.9	70.7 ± 959.5	71.3 ± 219.2	70.6 ± 1203.8	75.0 ± 311.5	61.5 ± 1432.3	68.9 ± 222.8	57.9 ± 1367.7	65.1 ± 160.8
FGSM	**CW**	87.0 ± 1.9	79.6 ± 0.9	57.0 ± 587.2	54.7 ± 43.6	87.1 ± 4.5	69.2 ± 10.8	66.4 ± 1584.1	77.6 ± 383.0	97.1 ± 0.7	93.7 ± 0.3
FGSM	μ	83.7 ± 19.8	70.4 ± 92.6	43.9 ± 1432.3	63.6 ± 304.7	77.1 ± 531.5	77.0 ± 155.5	66.0 ± 1221.6	70.7 ± 240.2	78.9 ± 435.7	75.0 ± 87.7
BIM	**FGSM**	86.0 ± 2.2	77.8 ± 3.8	48.1 ± 554.0	52.2 ± 9.4	86.1 ± 1.5	64.0 ± 0.5	66.6 ± 1507.0	76.8 ± 360.1	97.6 ± 0.1	93.7 ± 0.3
BIM	**PGD**	78.3 ± 0.6	68.8 ± 0.2	75.5 ± 1.1	66.8 ± 1.7	89.5 ± 0.6	82.0 ± 0.4	78.1 ± 1.6	68.9 ± 0.6	78.3 ± 1.1	68.0 ± 2.9
BIM	**AA**	81.0 ± 0.6	71.8 ± 0.7	89.9 ± 0.1	82.1 ± 2.3	98.1 ± 0.7	91.2 ± 0.7	70.6 ± 4.4	63.7 ± 0.9	70.7 ± 7.2	63.0 ± 11.5
BIM	**DF**	82.5 ± 2.7	73.9 ± 3.0	92.6 ± 1.4	84.8 ± 0.7	97.8 ± 0.6	89.7 ± 2.3	69.4 ± 0.3	63.9 ± 1.1	69.4 ± 1.4	62.1 ± 5.2
BIM	**CW**	83.6 ± 1.8	70.6 ± 1.3	91.8 ± 0.8	85.6 ± 0.9	88.7 ± 0.7	81.0 ± 2.5	74.6 ± 0.2	63.5 ± 1.1	74.6 ± 1.8	61.9 ± 2.7
BIM	μ	76.9 ± 412.9	72.9 ± 74.8	80.0 ± 1.0	70.9 ± 1.2	82.1 ± 2.6	74.4 ± 3.2	82.4 ± 1.3	74.9 ± 2.5	82.7 ± 1.1	72.5 ± 1.7
PGD	**FGSM**	81.9 ± 0.3	73.4 ± 0.7	78.7 ± 0.1	71.5 ± 0.2	73.0 ± 0.5	67.3 ± 1.1	91.0 ± 1.5	83.7 ± 3.8	89.5 ± 1.3	83.0 ± 1.4
PGD	**BIM**	80.8 ± 0.7	73.2 ± 0.0	78.2 ± 3.4	70.7 ± 0.0	72.4 ± 1.8	66.9 ± 1.8	90.4 ± 2.2	83.4 ± 1.9	89.5 ± 1.2	82.0 ± 1.4
PGD	**AA**	81.5 ± 1.0	65.2 ± 2.1	78.4 ± 2.9	63.4 ± 9.7	84.9 ± 0.9	68.6 ± 2.6	61.6 ± 0.4	54.7 ± 0.5	57.5 ± 1.2	52.3 ± 0.3
PGD	**DF**	77.5 ± 0.6	62.9 ± 4.1	98.8 ± 0.0	94.1 ± 0.4	99.9 ± 0.0	95.8 ± 0.5	43.6 ± 0.5	48.2 ± 0.0	45.2 ± 0.5	48.2 ± 1.0
PGD	**CW**	72.6 ± 5.6	57.6 ± 1.0	97.8 ± 0.1	92.3 ± 0.0	99.9 ± 0.0	97.3 ± 0.3	42.3 ± 0.8	48.4 ± 0.2	44.7 ± 0.9	48.3 ± 0.1
PGD	μ	82.8 ± 0.7	75.8 ± 1.5	82.2 ± 1.9	75.2 ± 1.0	72.8 ± 1.3	60.8 ± 3.0	73.0 ± 0.3	69.8 ± 1.2	71.4 ± 1.5	68.8 ± 0.3
AA	**FGSM**	68.1 ± 0.9	52.4 ± 1.0	97.4 ± 0.2	91.6 ± 0.1	98.5 ± 0.0	94.8 ± 0.7	41.6 ± 0.9	48.9 ± 0.1	43.8 ± 0.7	48.3 ± 0.6
AA	**BIM**	86.8 ± 2.6	79.6 ± 3.7	43.8 ± 12.1	39.2 ± 1.4	37.8 ± 0.1	34.3 ± 1.2	35.7 ± 0.6	32.9 ± 0.5	69.4 ± 1.3	63.0 ± 0.5
AA	**PGD**	83.3 ± 4.3	75.1 ± 0.9	38.7 ± 3.0	36.5 ± 0.2	33.6 ± 5.1	32.6 ± 1.4	30.6 ± 7.4	30.0 ± 1.1	75.2 ± 1.1	68.0 ± 0.3
AA	**DF**	77.4 ± 2.8	70.1 ± 3.6	80.1 ± 3.7	72.5 ± 3.1	81.0 ± 3.9	73.1 ± 4.9	64.6 ± 0.8	57.9 ± 1.8	63.6 ± 0.3	56.1 ± 1.6
AA	**CW**	77.6 ± 0.9	69.9 ± 0.0	87.8 ± 0.3	78.8 ± 0.3	89.4 ± 2.2	80.6 ± 6.8	58.5 ± 3.9	53.9 ± 2.5	60.5 ± 1.1	55.6 ± 0.9
AA	μ	69.9 ± 0.5	67.2 ± 0.5	54.7 ± 3.3	49.8 ± 1.5	52.3 ± 4.2	48.4 ± 0.8	73.3 ± 2.3	65.9 ± 3.0	74.8 ± 1.7	67.7 ± 2.1
DF	**FGSM**	75.2 ± 3.3	64.6 ± 4.9	83.4 ± 0.7	74.7 ± 0.1	88.0 ± 1.0	79.9 ± 0.4	57.6 ± 0.3	52.4 ± 0.2	61.6 ± 0.1	55.3 ± 0.3
DF	**BIM**	74.8 ± 7.1	65.2 ± 11.4	83.1 ± 2.6	74.7 ± 1.6	87.4 ± 0.4	78.8 ± 0.5	55.1 ± 1.4	50.6 ± 0.4	57.1 ± 0.4	52.4 ± 0.1
DF	**PGD**	74.2 ± 0.3	68.1 ± 0.1	67.2 ± 2.8	64.2 ± 0.2	68.4 ± 7.4	64.5 ± 7.3	65.5 ± 0.8	63.4 ± 0.3	67.9 ± 0.7	61.9 ± 0.7
DF	**AA**	73.5 ± 0.7	66.7 ± 0.3	72.8 ± 0.7	67.7 ± 0.9	78.0 ± 1.7	72.2 ± 3.2	71.4 ± 5.6	65.1 ± 2.5	68.7 ± 1.4	62.9 ± 5.4
DF	**CW**	70.4 ± 0.0	63.8 ± 0.6	69.2 ± 5.0	63.4 ± 14.9	75.8 ± 17.0	69.2 ± 19.4	51.6 ± 0.3	51.2 ± 0.0	52.0 ± 0.0	51.4 ± 0.0
DF	μ	73.2 ± 1.1	65.4 ± 1.2	71.5 ± 2.4	64.3 ± 2.8	68.6 ± 2.4	64.4 ± 1.7	72.9 ± 2.0	66.9 ± 2.5	63.8 ± 4.5	59.8 ± 7.0
CW	**FGSM**	59.8 ± 1.0	54.5 ± 2.1	93.1 ± 1.5	85.4 ± 5.1	99.8 ± 0.0	92.1 ± 7.5	54.4 ± 0.4	52.4 ± 0.1	54.2 ± 0.4	51.7 ± 0.2
CW	**BIM**	58.7 ± 0.6	53.5 ± 0.4	92.6 ± 3.6	84.3 ± 5.0	99.8 ± 0.0	93.4 ± 0.2	54.3 ± 0.1	51.9 ± 0.1	54.0 ± 0.6	51.6 ± 0.1
CW	**PGD**	55.8 ± 0.5	50.2 ± 0.1	91.6 ± 0.2	75.7 ± 5.7	93.3 ± 0.1	77.3 ± 1.6	53.8 ± 0.1	50.7 ± 0.0	53.7 ± 0.6	50.4 ± 0.0
CW	**AA**	57.5 ± 2.7	55.8 ± 1.5	88.1 ± 3.4	72.2 ± 3.2	85.4 ± 9.9	73.0 ± 0.0	97.0 ± 0.6	75.1 ± 10.1	53.6 ± 0.2	52.2 ± 0.2
CW	**DF**	58.6 ± 4.9	56.3 ± 3.5	87.4 ± 1.7	71.5 ± 5.0	83.4 ± 9.5	72.0 ± 8.6	96.1 ± 2.4	75.8 ± 1.5	53.6 ± 0.0	52.6 ± 1.3
CW	μ	72.2 ± 0.7	67.2 ± 3.0	71.9 ± 1.0	66.9 ± 1.2	69.6 ± 0.3	60.8 ± 1.5	76.3 ± 3.4	65.6 ± 3.0	75.8 ± 3.7	65.7 ± 4.0

Table 3.7. Attack transfer multiLID. Rows with the target μ give the average transfer rates from one attack to all others. The full multiLID with Random forest (RF) shows significantly better accuracy (ACC) for the attack transfer as LR in table 3.6.

multiLID											
Attacks		**CIFAR-10**				**CIFAR-100**				**ImageNet**	
		WRN 28-10		**VGG16**		**WRN 28-10**		**VGG16**		**WRN 50-2**	
from	**to**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**
Random forest (RF)											
FGSM	**BIM**	62.2 ± 249.6	56.4 ± 131.6	86.5 ± 1.8	70.8 ± 23.6	82.2 ± 336.8	76.6 ± 257.7	84.4 ± 2.0	72.6 ± 1.4	81.2 ± 1.3	68.5 ± 4.4
FGSM	**PGD**	60.6 ± 321.0	56.3 ± 154.8	68.8 ± 389.8	66.2 ± 317.8	83.3 ± 347.0	80.7 ± 316.8	63.6 ± 124.9	54.7 ± 2.2	58.6 ± 130.2	52.2 ± 3.7
FGSM	**AA**	84.8 ± 0.1	73.0 ± 3.5	69.5 ± 301.0	65.2 ± 250.2	85.5 ± 362.8	83.3 ± 324.8	84.2 ± 1.5	57.6 ± 3.5	84.8 ± 0.4	57.2 ± 0.9
FGSM	**DF**	78.0 ± 263.0	69.2 ± 100.4	77.7 ± 116.9	61.2 ± 2.4	80.1 ± 289.1	68.8 ± 77.4	80.5 ± 288.1	62.4 ± 44.3	78.7 ± 385.7	60.5 ± 49.6
FGSM	**CW**	86.6 ± 1.2	78.7 ± 1.7	61.3 ± 190.5	49.7 ± 0.1	84.0 ± 5.4	55.3 ± 0.3	79.9 ± 397.6	74.5 ± 299.3	97.7 ± 0.0	95.4 ± 0.1
FGSM	μ	79.3 ± 118.3	69.0 ± 83.8	67.0 ± 262.6	62.0 ± 159.1	81.8 ± 133.2	67.3 ± 116.6	79.0 ± 268.5	64.4 ± 54.8	81.9 ± 118.9	70.7 ± 60.3
BIM	**FGSM**	84.5 ± 3.5	77.9 ± 1.2	49.7 ± 186.9	49.0 ± 0.2	78.5 ± 2.1	53.8 ± 3.1	76.3 ± 567.8	69.5 ± 221.4	97.2 ± 0.1	94.5 ± 0.2
BIM	**PGD**	77.6 ± 0.0	67.5 ± 0.1	74.9 ± 1.4	65.7 ± 0.9	88.8 ± 0.5	81.0 ± 1.4	80.9 ± 1.6	72.2 ± 0.2	81.4 ± 1.6	69.5 ± 0.9
BIM	**AA**	69.7 ± 2.8	57.1 ± 1.3	91.1 ± 2.0	83.4 ± 4.8	97.4 ± 0.8	92.3 ± 1.8	64.3 ± 12.2	52.8 ± 0.3	64.7 ± 2.4	52.4 ± 0.9
BIM	**DF**	67.5 ± 2.9	57.1 ± 2.3	93.5 ± 0.4	86.2 ± 0.4	97.0 ± 0.9	91.3 ± 1.5	58.3 ± 7.0	51.8 ± 0.2	58.8 ± 4.8	51.9 ± 2.3
BIM	**CW**	79.7 ± 0.4	57.4 ± 2.4	91.8 ± 0.8	83.7 ± 1.2	88.3 ± 4.6	79.4 ± 4.1	79.9 ± 0.9	54.4 ± 0.6	79.6 ± 1.7	54.0 ± 0.6
BIM	μ	77.2 ± 152.1	69.0 ± 45.2	80.7 ± 1.0	71.2 ± 0.7	77.4 ± 4.1	67.6 ± 1.8	75.0 ± 3.2	67.7 ± 1.4	83.9 ± 1.7	65.8 ± 1.8
PGD	**FGSM**	81.2 ± 0.2	72.7 ± 0.0	71.6 ± 1.6	63.7 ± 3.0	66.7 ± 1.6	57.6 ± 2.6	82.3 ± 4.5	75.8 ± 25.4	89.7 ± 0.5	83.1 ± 2.5
PGD	**BIM**	79.8 ± 0.6	72.1 ± 0.7	69.8 ± 0.0	58.1 ± 9.1	64.9 ± 2.8	53.2 ± 5.4	79.0 ± 13.8	67.2 ± 47.7	89.1 ± 1.4	82.8 ± 1.9
PGD	**AA**	80.1 ± 1.1	65.1 ± 1.5	76.6 ± 0.5	64.1 ± 0.7	84.3 ± 0.1	70.1 ± 1.5	54.9 ± 0.7	52.3 ± 0.0	52.8 ± 0.2	51.3 ± 0.2
PGD	**DF**	81.6 ± 6.1	56.8 ± 7.9	98.8 ± 0.1	95.0 ± 0.8	99.9 ± 0.0	97.0 ± 0.1	44.2 ± 4.8	48.2 ± 0.1	43.3 ± 1.0	47.8 ± 0.8
PGD	**CW**	83.0 ± 0.2	52.6 ± 1.3	97.4 ± 0.2	90.8 ± 0.5	99.9 ± 0.0	97.9 ± 0.0	44.6 ± 4.2	48.6 ± 0.0	44.2 ± 0.9	48.5 ± 0.5
PGD	μ	78.3 ± 1.7	70.6 ± 6.7	76.5 ± 3.7	66.7 ± 13.0	69.7 ± 0.6	60.6 ± 0.8	73.6 ± 2.4	69.0 ± 1.9	73.8 ± 1.1	67.7 ± 0.5
AA	**FGSM**	76.4 ± 4.0	50.3 ± 0.0	97.0 ± 0.3	88.3 ± 0.0	98.5 ± 0.1	92.2 ± 0.8	45.8 ± 2.4	49.5 ± 0.0	44.9 ± 6.1	49.2 ± 0.1
AA	**BIM**	76.3 ± 3.3	69.5 ± 8.4	19.3 ± 21.3	34.4 ± 1.1	15.5 ± 11.8	32.5 ± 0.3	13.6 ± 12.8	31.7 ± 0.9	69.3 ± 0.5	63.7 ± 0.6
AA	**PGD**	70.7 ± 4.7	66.6 ± 7.5	21.0 ± 5.0	32.7 ± 2.2	17.8 ± 12.6	30.1 ± 2.5	15.0 ± 4.2	29.6 ± 3.8	73.1 ± 0.7	67.7 ± 1.1
AA	**DF**	76.0 ± 3.3	69.6 ± 1.0	78.5 ± 2.8	71.0 ± 3.0	78.1 ± 2.8	70.2 ± 2.8	61.1 ± 1.2	57.6 ± 1.1	60.7 ± 0.4	57.2 ± 2.7
AA	**CW**	71.6 ± 2.4	63.9 ± 1.3	86.1 ± 1.2	79.1 ± 0.8	89.2 ± 1.0	81.0 ± 1.7	53.9 ± 0.3	51.5 ± 1.0	58.2 ± 2.0	54.7 ± 0.6
AA	μ	72.5 ± 2.6	65.9 ± 0.2	38.8 ± 9.9	46.4 ± 2.3	39.5 ± 5.5	45.3 ± 3.4	70.9 ± 2.1	65.1 ± 2.1	71.8 ± 1.4	66.0 ± 1.1
DF	**FGSM**	70.0 ± 1.1	60.3 ± 2.9	80.5 ± 2.1	71.9 ± 2.4	87.5 ± 0.9	78.8 ± 2.6	54.6 ± 0.6	51.4 ± 0.3	58.8 ± 0.6	54.5 ± 0.7
DF	**BIM**	65.9 ± 0.7	60.0 ± 3.3	81.4 ± 2.7	72.8 ± 5.5	86.7 ± 0.6	78.3 ± 1.0	52.4 ± 0.6	50.5 ± 0.2	56.5 ± 1.1	53.1 ± 0.0
DF	**PGD**	66.0 ± 1.5	61.3 ± 1.0	58.5 ± 2.9	53.9 ± 6.8	63.5 ± 4.1	57.1 ± 2.8	55.0 ± 2.6	50.0 ± 1.5	75.0 ± 0.4	69.1 ± 0.3
DF	**AA**	66.6 ± 7.5	63.1 ± 1.9	65.2 ± 7.3	61.2 ± 9.4	70.3 ± 2.7	66.1 ± 3.3	62.2 ± 10.5	59.0 ± 13.3	70.7 ± 0.5	65.4 ± 0.6
DF	**CW**	55.3 ± 3.7	52.8 ± 3.6	59.5 ± 1.3	55.8 ± 1.9	50.7 ± 11.7	48.8 ± 5.8	50.1 ± 0.3	49.6 ± 0.1	50.5 ± 0.2	50.6 ± 0.0
DF	μ	70.3 ± 1.1	63.4 ± 1.8	68.6 ± 1.1	62.9 ± 2.0	63.6 ± 2.3	58.3 ± 2.5	67.0 ± 5.7	63.0 ± 5.7	53.2 ± 3.4	51.5 ± 2.3
CW	**FGSM**	54.6 ± 0.6	51.6 ± 0.0	94.1 ± 0.1	87.3 ± 1.8	99.5 ± 0.0	93.4 ± 1.6	53.9 ± 0.1	52.5 ± 0.0	53.4 ± 0.5	51.5 ± 0.2
CW	**BIM**	55.2 ± 1.0	51.5 ± 0.3	91.8 ± 0.6	82.9 ± 1.3	99.6 ± 0.0	94.2 ± 1.6	53.6 ± 0.3	51.7 ± 0.4	53.7 ± 0.4	51.2 ± 0.2
CW	**PGD**	51.4 ± 2.6	49.9 ± 0.1	85.7 ± 0.7	63.4 ± 5.8	89.9 ± 0.4	71.9 ± 0.1	51.5 ± 1.1	50.2 ± 0.0	51.7 ± 0.1	50.3 ± 0.0
CW	**AA**	52.8 ± 5.4	51.5 ± 0.5	74.3 ± 2.6	67.1 ± 2.1	78.6 ± 15.6	69.5 ± 6.9	75.1 ± 227.1	66.4 ± 84.9	53.6 ± 0.2	52.0 ± 0.9
CW	**DF**	53.4 ± 1.1	53.0 ± 1.8	76.8 ± 7.9	67.7 ± 0.8	80.8 ± 21.8	71.2 ± 15.1	87.5 ± 63.7	72.5 ± 5.4	50.9 ± 0.8	50.8 ± 1.5
CW	μ	71.1 ± 0.3	67.3 ± 0.7	70.8 ± 0.5	66.3 ± 0.8	66.0 ± 1.0	57.1 ± 1.2	66.9 ± 50.2	61.3 ± 19.1	69.9 ± 19.0	63.0 ± 4.9

3.6 LIMITATIONS

While our method allows to achieve imporoved results in the considered test scenario and for the given datasets, we do not claim to have solved the actual problem. We use the evaluation setting as proposed in previous works (e.g. [91]) where each attack method is evaluated separately and with constant attack parameters. For a deployment in real-world scenarios, the robustness of a detector under potential disguise mechanisms needs to be verified. An extended study on the transferability of our method from one attack to the other can be found in section 3.5.6. It shows first promising resulting in this respect but also leaves room for further improvement.

3.7 SUMMARY

In this work, we revisit the MLE estimate of the local intrinsic dimensionality (LID) which has been used in previous works on adversarial detection. An analysis of the extracted LID features and their theoretical properties allows us to redefine an LID-based feature using unfolded local growth rate estimates that are significantly more discriminative than the aggregated LID measure.

We show improved results across different attacks and variety of epsilons. Moreover, we modify the LID so that the log features are trained by the Logistic Regression which lead to improved results as well. We believe that our approach have a potential to apply to many other related machine learning tasks.

Chapter 4

Visual Prompting for Adversarial Robustness

In this chapter, we leverage visual prompting (VP) to improve adversarial robustness of a fixed, pre-trained model at test time. Compared to conventional adversarial defenses, VP allows us to design universal (i.e., data-agnostic) input prompting templates, which have plug-and-play capabilities at test time to achieve desired model performance without introducing much computation overhead.

Although VP has been successfully applied to improving model generalization, it remains elusive whether and how it can be used to defend against adversarial attacks. We investigate this problem and show that the vanilla VP approach is not effective in adversarial defense since a universal input prompt lacks the capacity for robust learning against sample-specific adversarial perturbations.

To circumvent it, we propose a new VP method, termed Class-wise Adversarial Visual Prompting (C-AVP), to generate class-wise visual prompts so as to not only leverage the strengths of ensemble prompts but also optimize their interrelations to improve model robustness. Our experiments show that C-AVP outperforms the conventional VP method, with 2.1$\times$ standard accuracy gain and 2$\times$ robust accuracy gain. Compared to classical test-time defenses, C-AVP also yields a 42$\times$ inference time speedup.

4.1 BACKGROUND

In this section, we explain the advantages and the already applied fields of visual prompting (alias model reprogramming) and importance for adversarial robustness. Additionally, we introduce related literature and some of its most prominent work.

4.1.1 Preliminaries

Machine learning (ML) models can easily be manipulated (by an adversary) to output drastically different classifications. Thereby, model robustification against adversarial attacks is now a major focus of research. Yet, a large volume of existing works focused on training recipes and/or model architectures to gain robustness. Adversarial training (AT) [92], one of the most effective defense, adopted min-max optimization to minimize the worst-case training loss induced by adversarial attacks. Extended from AT, various defense methods were proposed, ranging from supervised learning to semi-supervised learning, and further to unsupervised learning [18, 24, 43, 108, 121, 144, 150, 161, 163, 168].

Although the design for robust training has made tremendous success in improving model robustness [5, 33], it typically takes an intensive computation cost with poor defense scalability to a fixed, pre-trained ML model. Towards circumventing this difficulty, the problem of test-time defense arises; see the seminal work in Croce et al. [31]. Test-time defense alters either a test-time input example or a small portion of the pre-trained model. Examples include input (anti-adversarial) purification [1, 94, 159] and model refinement by augmenting the pre-trained model with auxiliary components [45, 64, 118]. However, these defense techniques inevitably raise the inference time and hamper the test-time efficiency [31]. Inspired by that, our work will advance the test-time defense technology by leveraging the idea of visual prompting (VP) [8], also known as model reprogramming [23, 41, 136, 162].

Generally speaking, as shown in fig. 4.1, VP [8] creates a universal (i.e., data-agnostic) input prompting template (in terms of input perturbations) in order to improve the generalization ability of a pre-trained model when incorporating such a visual prompt into test-time examples.

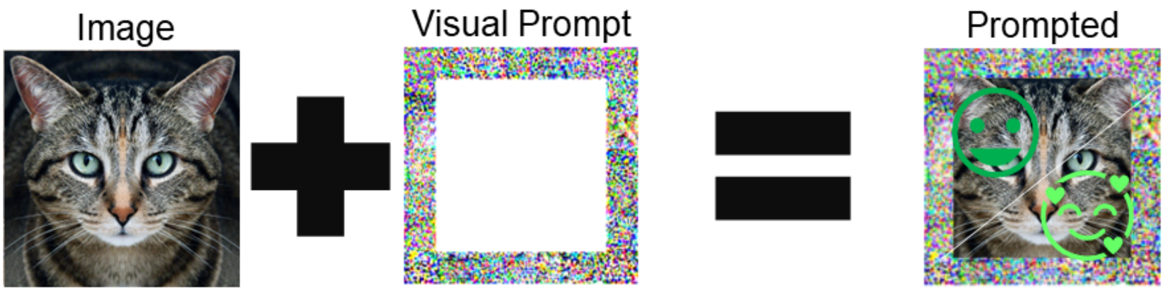


Figure 4.1. Concept of visual prompt (VP). At test-time a visual prompt is added to the input image, which improve the generalization ability of a pre-trained model.

It enjoys the same idea as model reprogramming [23, 41, 136, 162] or unadversarial example [117], which optimizes a universal perturbation pattern to maneuver (i.e., reprogram) the functionality of a pre-trained model towards the desired criterion, e.g., cross-domain transfer learning [136], out-of-distribution generalization [117], and fairness [162].

4.1.2 Related Work

Visual prompting. Originated from the idea of in-context learning or prompting in natural language processing (NLP) [12, 77, 107, 166], VP was first proposed in Bahng et al. [8] for vision models. Before formalizing VP in Bahng et al. [8], the underlying prompting technique has also been devised in computer vision with different naming. For example, VP is closely related to adversarial reprogramming or model reprogramming [23, 41, 100, 136, 152, 169], which focused on altering the functionality of a fixed, pre-trained model across domains by augmenting test-time examples with an additional (universal) input perturbation pattern. Unadversarial learning also enjoys a similar idea to VP. In [117], unadversarial examples that perturb original ones using 'prompting' templates were introduced to improve out-of-distribution generalization. Yet, the problem of VP for adversarial defense is under-explored.

Adversarial defense. The lack of adversarial robustness is a weakness of ML models. Adversarial defense, such as adversarial detection [45, 48, 95, 96, 143, 155] and robust training [11, 24, 43, 118, 144, 163], is a current research focus. In particular, adversarial training (AT) [92] is the most widely-used defense strategy and has inspired many recent advances in adversarial defense [5, 33, 64, 97, 139, 157]. However, these AT-type defenses (with the goal of robustness-enhanced model training) are computationally intensive due to min-max optimization over model parameters. To reduce the computation overhead of robust training, the problem of test-time defense arises [31], which aims to robustify a given model via lightweight unadversarial input perturbations (a.k.a input purification) [123, 159] or minor modifications to the fixed model [25, 167]. In different kinds of test-time defenses, the most relevant work to ours is anti-adversarial perturbation [1]. However, it remains elusive whether or not VP could be designed as an effective solution to adversarial defense. We will investigate this problem, which we call adversarial visual prompting (AVP) in this work. Compared to conventional test-time defense methods, AVP significantly reduces the inference time overhead since visual prompts can be designed offline over training data and have the plug-and-play capability applied to any testing data.

4.2 CONTRIBUTIONS

We summarize our contributions as below.

- We formulate and investigate the problem of AVP for the first time and empirically show the conventional data-agnostic VP design is incapable of gaining adversarial robustness.

- We propose a new VP method, termed class-wise AVP (C-AVP), which produces multiple, class-wise visual prompts with explicit optimization on their couplings to gain better adversarial robustness.

- We provide insightful experiments to demonstrate the pros and cons of VP in adversarial defense.

4.3 METHOD

We start this section introducing the problem definition and discuss to utilize visual prompting for adversarial robustness. This work represents the first approach to improve the robustness against adversarial examples with visual prompting. After examining a universal prompt against adversarial examples we leveraged the robustness by using creating a visual prompt for each class.

4.3.1 Problem Defintion

In this section, we will begin by providing a brief background on VP, and then introduce the problem of our interest – adversarial visual prompting (AVP) – which aims at generating visual prompts to improve adversarial robustness of a pre-trained, fixed model. Through a warm-up example, we will empirically show that the conventional design of VP is difficult to apply to the paradigm of AVP.

Visual prompting. We describe the problem setup of VP following Bahng et al. [8,41,136, 162]. Specifically, let $\mathcal{D}_{\text{tr}}$ denote a training set for supervised learning, where $(\mathbf{x}, y) \in \mathcal{D}_{\text{tr}}$ signifies a training sample with feature $\mathbf{x}$ and label y. And let $\boldsymbol{\delta}$ be a visual prompt to be designed. The prompted input is then given by $\mathbf{x} + \boldsymbol{\delta}$ with respect to (w.r.t.) $\mathbf{x}$. Different from the problem of adversarial attack generation that optimizes $\boldsymbol{\delta}$ for erroneous prediction, VP drives $\boldsymbol{\delta}$ to minimize the performance loss ℓ of a pre-trained model $\boldsymbol{\theta}$. This leads to

$$
\begin{array}{ll}
\underset{\boldsymbol{\delta}}{\text{minimize}} & \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}_{\text{tr}}}[\ell(\mathbf{x}+\boldsymbol{\delta}; y, \boldsymbol{\theta})] \\
\text{subject to} & \boldsymbol{\delta} \in \mathcal{C},
\end{array}
\tag{4.1}
$$

where ℓ denotes prediction error given the training data $(\mathbf{x}, y)$ and base model $\boldsymbol{\theta}$, and $\mathcal{C}$ is a perturbation constraint. Following Bahng et al. [8,41,136], $\mathcal{C}$ restricts $\boldsymbol{\delta}$ to let $\mathbf{x} + \boldsymbol{\delta} \in [0, 1]$ for any $\mathbf{x}$. Projected gradient descent (PGD) [92,117] can then be applied to solving problem (4.1). In the evaluation, $\boldsymbol{\delta}$ is integrated into test data to improve the prediction ability of $\boldsymbol{\theta}$.

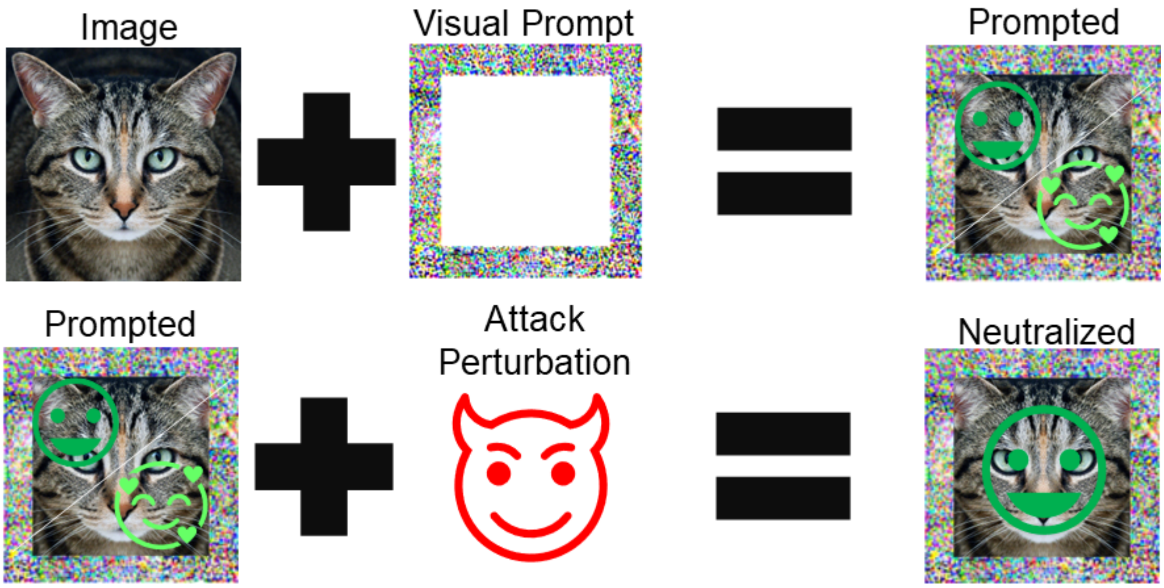


Figure 4.2. Concept of the U-AVP. The (U-AVP) consists of two terms: (Top) The first term affects the visual prompt to improve the prediction ability. (Bottom) The second terms compromises, if the prompted image is attacked, but the visual prompt is trained to neutralize the attack perturbation.

4.3.2 Adversarial Visual Prompting

Inspired by the usefulness of VP to improve model generalization [8, 136], we ask:

(AVP problem) Can VP (4.1) be extended to robustify $\boldsymbol{\theta}$ against adversarial attacks?

At the first glance, the AVP problem seems trivial if we specify the performance loss ℓ as the adversarial training loss [92, 163]:

$$\ell_{\text{adv}}(\mathbf{x} + \boldsymbol{\delta}; y, \boldsymbol{\theta}) = \underset{\mathbf{x}': \|\mathbf{x}' - \mathbf{x}\|_\infty \leq \epsilon}{\text{maximize}} \; \ell(\mathbf{x}' + \boldsymbol{\delta}; y, \boldsymbol{\theta}), \tag{4.2}$$

where $\mathbf{x}'$ denotes the adversarial input that lies in the ℓ_∞-norm ball centered at $\mathbf{x}$ with radius $\epsilon > 0$.

Recall from (4.1) that the conventional VP requests $\boldsymbol{\delta}$ to be universal across training data. Thus, we term universal AVP (U-AVP) the following problem by integrating (4.1) with (4.2) and compare the following equation with fig. 4.2:

$$\begin{array}{ll} \underset{\boldsymbol{\delta}:\, \boldsymbol{\delta} \in \mathcal{C}}{\text{minimize}} & \lambda \mathbb{E}_{(\mathbf{x}, y) \in \mathcal{D}_{\text{tr}}}[\ell(\mathbf{x} + \boldsymbol{\delta}; y, \boldsymbol{\theta})] + \\ & \mathbb{E}_{(\mathbf{x}, y) \in \mathcal{D}_{\text{tr}}}[\ell_{\text{adv}}(\mathbf{x} + \boldsymbol{\delta}; y, \boldsymbol{\theta})] \end{array} \tag{U-AVP}$$

where $\lambda > 0$ is a regularization parameter to strike a balance between generalization and adversarial robustness [163]. The problem (U-AVP) can be effectively solved using a standard min-max optimization method, which involves two alternating optimization routines:

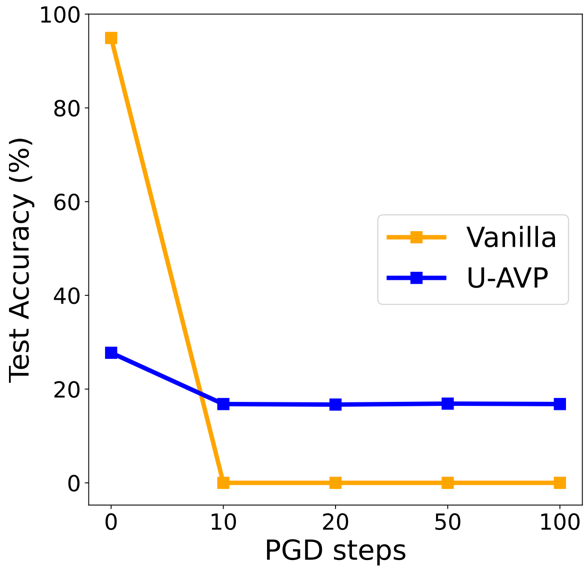


Figure 4.3. Example of designing U-AVP for adversarial defense on (CIFAR-10, ResNet18), measured by robust accuracy against PGD attacks [92] of different steps. The robust accuracy of 0 steps is the standard accuracy.

inner maximization and outer minimization. The former generates adversarial examples as AT, and the latter produces the visual prompt $\boldsymbol{\delta}$ like (4.1). At test time, the effectiveness of $\boldsymbol{\delta}$ is measured from two aspects: (1) standard accuracy, i.e., the accuracy of $\boldsymbol{\delta}$-integrated benign examples, and (2) robust accuracy, i.e., the accuracy of $\boldsymbol{\delta}$-integrated adversarial examples (against the victim model $\boldsymbol{\theta}$). Despite the succinctness of (U-AVP), fig. 4.3 shows its ineffectiveness to defend against adversarial attacks. Compared to the vanilla VP (4.1), it also suffers a significant standard accuracy drop (over 50% in fig. 4.3 corresponding to 0 PGD attack steps) and robust accuracy is only enhanced by a small margin (around 18% against PGD attacks). The negative results in fig. 4.3 are not quite surprising since a data-agnostic input prompt $\boldsymbol{\delta}$ has limited learning capacity to enable adversarial defense. Thus, it is non-trivial to tackle the problem of AVP.

4.4 EXPERIMENTS

We start this section to explain the experiment setup, then we explain the U-AVP algorithm, and afterwards our improvements.

4.4.1 Experiment Setup

We conduct experiments on CIFAR-10 with a pre-trained ResNet18 of testing accuracy of 94.92% on the standard test dataset. We use PGD-10 (i.e., PGD attack with 10 steps [92]) to generate adversarial examples with $\epsilon = 8/255$ during visual prompts training, and with a cosine learning rate scheduler starting at 0.1. Throughout experiments, we choose $\lambda = 1$ in (U-AVP), and $\tau = 0.1$ and $\gamma = 3$ in (C-AVP). The width of a visual prompt is set to 8 (see fig. 4.5 for the visualization).

4.4.2 Universal Adversarial Visual Prompt

In order to clarify the U-AVP, we describe the algorithm 1 in detail in eq. (U-AVP). First the visual prompt is initialized. Since this is a supervised approach, the normal and the adversarial counterpart are added to a batch. These batches are used in optimization loss function, minimized by stochastic gradient descent. The minimum between the normal image and adversarial counterpart should be found.

4.4.3 Class-wise Adversarial Visual Prompt

We explain that a simple utilizaton of the U-AVP is not effective. Then, we expand the U-AVP by different terms to take the information of different classes into account.

No free lunch for class-wise visual prompts. A direct extension of (U-AVP) is to introduce multiple adversarial visual prompts, each of which corresponds to one class in the training set $\mathcal{D}_{\text{tr}}$. If we split $\mathcal{D}_{\text{tr}}$ into class-wise training sets $\{\mathcal{D}_{\text{tr}}^{(i)}\}_{i=1}^{N}$ (for N classes) and introduce class-wise visual prompts $\{\boldsymbol{\delta}^{(i)}\}$, then the direct C-AVP extension from (U-AVP) becomes

$$\underset{\{\boldsymbol{\delta}^{(i)} \in \mathcal{C}\}_{i \in [N]}}{\text{minimize}} \quad \frac{1}{N} \sum_{i=1}^{N} \Big\{ \lambda \mathbb{E}_{(\mathbf{x},y) \in \mathcal{D}_{\text{tr}}^{(i)}} [\ell(\mathbf{x} + \boldsymbol{\delta}^{(i)}; y, \boldsymbol{\theta})] + \mathbb{E}_{(\mathbf{x},y) \in \mathcal{D}_{\text{tr}}^{(i)}} [\ell_{\text{adv}}(\mathbf{x} + \boldsymbol{\delta}^{(i)}; y, \boldsymbol{\theta})] \Big\} \tag{C-AVP-v0}$$

where $[N]$ denotes the set of class labels $\{1, 2, \ldots, N\}$. It is worth noting that C-AVP-v0 is decomposed over class labels. Although the class-wise separability facilitates numerical optimization, it introduces challenges (C1)-(C2) when applying class-wise visual prompts for adversarial defense.

• (C1) Test-time prompt selection: After acquiring the visual prompts $\{\boldsymbol{\delta}^{(i)}\}$ from (C-AVP-v0), it remains unclear how a class-wise prompt should be selected for application to a test-time example $\mathbf{x}_{\text{test}}$. An intuitive way is to use the inference pipeline of $\boldsymbol{\theta}$ by aligning its

Algorithm 1 The expectation $\mathbb{E}$ [6] is calculated using stochastic gradient descent (SGD) in the context of universal visual adversarial prompting for test-time defense, as described in the cited work. This process can be viewed as a game between the visual prompt and the adversarial example. The first term calculates the expectation value per sample of the visual prompt, while the second term maximizes the expectation value of the adversarial example.

Require: A pre-trained classification model f, clean training data $\mathcal{D}_{\text{tr}} = \{(\mathbf{x}, y)\}$, batch size $\mathcal{B}$, λ controls the visual prompt term. The visual prompt **x** with the learnable parameters $\boldsymbol{\theta}$ and the mask $\boldsymbol{m}$.

Ensure: Defense Perturbation - minimize$_{\boldsymbol{\delta}}$ $\lambda \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}_{\text{tr}}}[\ell_{\text{CE}}(f(\mathbf{x} + \mathbf{m} \odot \boldsymbol{\delta}); y)] + \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}_{\text{tr}}} \underset{\|\boldsymbol{\delta}_{\text{adv}}\|_\infty \leq \epsilon}{\text{maximize}}[\ell_{\text{CE}}(f(\mathbf{x} + \mathbf{m} \odot \boldsymbol{\delta} + \boldsymbol{\delta}_{\text{adv}}); y))]$ ▷ $\boldsymbol{\delta}(\mathbf{x}, \boldsymbol{\theta})$

repeat
  Initialization: initial value of visual prompt $\boldsymbol{\delta} = \mathbf{0}$ (or random initialization)
  for each $\mathcal{B}$ in $\{(\mathbf{x}, y)\}$: **do**
    $\mathcal{B}' = []$
    for each $\mathbf{x}, y$ in $\mathcal{B}$: **do**
      $\boldsymbol{\delta}^*_{adv} \leftarrow \arg\max_{\|\boldsymbol{\delta}_{\text{adv}}\| \leq \epsilon} \ell_{\text{CE}}(f(\mathbf{x} + \mathbf{m} \odot \boldsymbol{\delta} + \boldsymbol{\delta}_{\text{adv}}); y)$ ▷ Adv. generation given $\boldsymbol{\delta}$
      $\mathcal{B}'$.append$((\mathbf{x} + \boldsymbol{\delta}^*_{\text{adv}}, y))$
    end for
    Given $\mathcal{B}$ and $\mathcal{B}'$, update visual prompt $\boldsymbol{\delta}$ by SGD: ▷ Visual Prompt

$$\begin{aligned}\boldsymbol{\delta} \leftarrow \boldsymbol{\delta} - \eta_1 \Big[\frac{\lambda}{|\mathcal{B}|} \sum_{\mathbf{x}, y \in \mathcal{B}} \nabla_{\boldsymbol{\delta}} \ell_{CE}(f(\text{CLIP}(\mathbf{x} + \mathbf{m} \odot \boldsymbol{\delta})); y) + \\ \frac{1}{|\mathcal{B}|} \sum_{\mathbf{x}', y \in \mathcal{B}'} \nabla_{\boldsymbol{\delta}} \ell_{CE}(f(\text{CLIP}(\mathbf{x}' + \mathbf{m} \odot \boldsymbol{\delta})); y) \Big]\end{aligned} \tag{U-AVP}$$

  end for
until training converged
Output $\boldsymbol{\delta}$ ▷ U-AVP

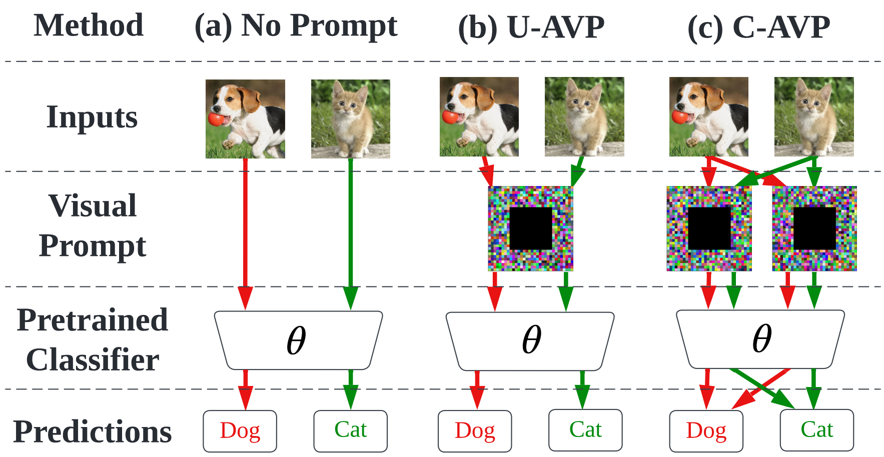


Figure 4.4. Overview of C-AVP over two classes (red and green) vs. eq. (U-AVP) and the prompt-free learning pipeline.

top-1 prediction with the prompt selection. That is, the selected prompt $\boldsymbol{\delta}$ and the predicted class i^* are determined by

$$\boldsymbol{\delta} = \boldsymbol{\delta}^*, \; i^* = \underset{i \in [N]}{\arg\max} \, f_i(\mathbf{x}_{\text{test}} + \boldsymbol{\delta}^{(i)}; \boldsymbol{\theta}), \tag{4.3}$$

where $f_i(\mathbf{x}; \boldsymbol{\theta})$ denotes the ith-class prediction confidence. However, the seemingly correct rule (4.3) leads to a large prompt selection error (thus poor prediction accuracy) due to (C2).

• (C2) Backdoor effect of class mis-matched prompts: Given $\boldsymbol{\delta}^{(i)}$ from (C-AVP-v0), if the test-time example $\mathbf{x}_{\text{test}}$ is drawn from class i, the visual prompt $\boldsymbol{\delta}^{(i)}$ then helps prediction. However, if $\mathbf{x}_{\text{test}}$ is not originated from class i, then $\boldsymbol{\delta}^{(i)}$ could serve as a backdoor attack trigger [51] with the targeted backdoor label i for the 'prompted input' $\mathbf{x}_{\text{test}} + \boldsymbol{\delta}^{(i)}$. Since the backdoor attack is also input-agnostic, the class-discriminative ability of $\mathbf{x}_{\text{test}} + \boldsymbol{\delta}^{(i)}$ enabled by $\boldsymbol{\delta}^{(i)}$ could result in incorrect prediction towards the target class i for $\mathbf{x}_{\text{test}}$.

Joint prompts optimization for C-AVP. The failure of C-AVP-v0 inspires us to rethink the value of class-wise separability. As illustrated in challenges (C1)-(C2), the compatibility with the test-time prompt selection rule and the interrelationship between class-wise visual prompts should be taken into account. To this end, we develop a series of new AVP principles below. Figure 4.4 provides a schematic overview of C-AVP and its comparison with U-AVP and the predictor without VP.

First, to bake the prompt selection rule (4.3) into C-AVP, we enforce the correct prompt selection, i.e., under the condition that $f_y(\mathbf{x}+\boldsymbol{\delta}^{(y)}; \boldsymbol{\theta}) > \max_{k:k\neq y} f_k(\mathbf{x}+\boldsymbol{\delta}^{(k)}; \boldsymbol{\theta})$ for $(\mathbf{x}, y) \in \mathcal{D}^{(y)}$. The above can be cast as a CW-type loss [14]:

$$\ell_{\mathrm{C-AVP},1}(\{\boldsymbol{\delta}^{(i)}\};\mathcal{D}_{\mathrm{tr}},\boldsymbol{\theta}) = \\ \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}_{\mathrm{tr}}} \max\{\max_{k\neq y} f_k(\mathbf{x}+\boldsymbol{\delta}^{(k)};\boldsymbol{\theta}) - f_y(\mathbf{x}+\boldsymbol{\delta}^{(y)};\boldsymbol{\theta}), -\tau\}, \tag{4.4}$$

where $\tau > 0$ is a confidence threshold. The rationale behind (4.4) is that given a data sample $(\mathbf{x}, y)$, the minimum value of $\ell_{\mathrm{C-AVP},1}$ is achieved at $-\tau$, indicating the desired condition with the confidence level τ. Compared with (C-AVP-v0), another key characteristic of $\ell_{\mathrm{C-AVP},1}$ is its non-splitting over class-wise prompts $\{\boldsymbol{\delta}^{(i)}\}$, which benefits the joint optimization of these prompts.

Second, to mitigate the backdoor effect of mis-matched prompts, we propose additional two losses, noted by $\ell_{\mathrm{C-AVP},2}$ and $\ell_{\mathrm{C-AVP},3}$, to penalize the data-prompt mismatches. Specifically, $\ell_{\mathrm{C-AVP},2}$ penalizes the backdoor-alike targeted prediction accuracy of a class-wise visual prompt when applied to mismatched training data. For the prompt $\boldsymbol{\delta}^{(i)}$, this leads to

$$\ell_{\mathrm{C-AVP},2}(\{\boldsymbol{\delta}^{(i)}\};\mathcal{D}_{\mathrm{tr}},\boldsymbol{\theta}) = \\ \frac{1}{N}\sum_{i=1}^{N} \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}_{\mathrm{tr}}^{(-i)}} \max\{f_i(\mathbf{x}+\boldsymbol{\delta}^{(i)};\boldsymbol{\theta}) - f_y(\mathbf{x}+\boldsymbol{\delta}^{(i)};\boldsymbol{\theta}), -\tau\}, \tag{4.5}$$

where $\mathcal{D}_{\mathrm{tr}}^{(-i)}$ denotes the training data set by excluding $\mathcal{D}_{\mathrm{tr}}^{(i)}$. The class i-associated prompt $\boldsymbol{\delta}^{(i)}$ should not behave as a backdoor trigger to non-i classes' data. Likewise, if the prompt is applied to the correct data class, then the prediction confidence should surpass that of a mismatched case. This leads to

$$\ell_{\mathrm{C-AVP},3}(\{\boldsymbol{\delta}^{(i)}\};\mathcal{D}_{\mathrm{tr}},\boldsymbol{\theta}) = \\ \mathbb{E}_{(\mathbf{x},y)\in\mathcal{D}_{\mathrm{tr}}} \max\{\max_{k\neq y} f_y(\mathbf{x}+\boldsymbol{\delta}^{(k)};\boldsymbol{\theta}) - f_y(\mathbf{x}+\boldsymbol{\delta}^{(y)};\boldsymbol{\theta}), -\tau\}. \tag{4.6}$$

Let $\ell_{\mathrm{C-AVP},0}(\{\boldsymbol{\delta}^{(i)}\};\mathcal{D}_{\mathrm{tr}},\boldsymbol{\theta})$ denote the objective function of (C-AVP-v0). Integrated with $\ell_{\mathrm{C-AVP},q}(\{\boldsymbol{\delta}^{(i)}\};\mathcal{D}_{\mathrm{tr}},\boldsymbol{\theta})$ for $q \in \{1,2,3\}$, the desired class-wise AVP design is cast as

$$\begin{array}{ll} \underset{\{\boldsymbol{\delta}^{(i)}\in\mathcal{C}\}_{i\in[N]}}{\text{minimize}} & \ell_{\mathrm{C-AVP},0}(\{\boldsymbol{\delta}^{(i)}\};\mathcal{D}_{\mathrm{tr}},\boldsymbol{\theta}) + \\ & \gamma\sum_{q=1}^{3}\ell_{\mathrm{C-AVP},q}(\{\boldsymbol{\delta}^{(i)}\};\mathcal{D}_{\mathrm{tr}},\boldsymbol{\theta}), \end{array} \tag{C-AVP}$$

where $\gamma > 0$ is a parameter for class-wise prompting penalties.

4.4.4 C-AVP outperforms conventional Visual Prompting

Tab. 4.1 demonstrates the effectiveness of proposed C-AVP approach vs. U-AVP (the direct extension of VP to adversarial defense) and the C-AVP-v0 method in the task of robustify a normally-trained ResNet18 on CIFAR-10. For comparison, we also report the standard accuracy of the pre-trained model and the vanilla VP solution given by (4.1). As we can see, C-AVP outperforms U-AVP and C-AVP-v0 in both standard accuracy and robust accuracy. We also observe that compared to the pretrained model and the vanilla VP, the robustness-induced VP variants bring in an evident standard accuracy drop as the cost of robustness.

Table 4.1. VP performance comparison in terms of standard (std) accuracy (ACC) and robust accuracy against PGD attacks with $\epsilon = 8/255$ and multiple PGD steps on (CIFAR-10, ResNet18).

Evaluation metrics (%)	**Std ACC**	**Robust ACC vs PGD w/ step #** 10	20	50	100
Pre-trained	**94.92**	0	0	0	0
Vanilla VP	94.48	0	0	0	0
U-AVP	27.75	16.9	16.81	16.81	16.7
C-AVP-v0	19.69	13.91	13.63	13.6	13.58
C-AVP (ours)	57.57	**34.75**	**34.62**	**34.51**	**33.63**

4.4.5 Prompting regularization Effect in (C-AVP)

Tab. 4.2 shows different settings of prompting regularizations used in C-AVP, where ‘Si’ represents a certain loss configuration. As we can see, the use of $\ell_{\mathrm{C-AVP},2}$ contributes most to the performance of learned visual prompts (see S3). This is not surprising, since we design $\ell_{\mathrm{C-AVP},2}$ for mitigating the backdoor effect of class-wise prompts, which is the main source of prompting selection error. We also note that $\ell_{\mathrm{C-AVP},1}$ is the second most important regularization. This is because such a regularization is accompanied by the prompt selection rule (4.3). Tab. 4.2 also indicates that the combination of $\ell_{\mathrm{C-AVP},1}$ and $\ell_{\mathrm{C-AVP},2}$ is a possible computationally lighter alternative to (C-AVP).

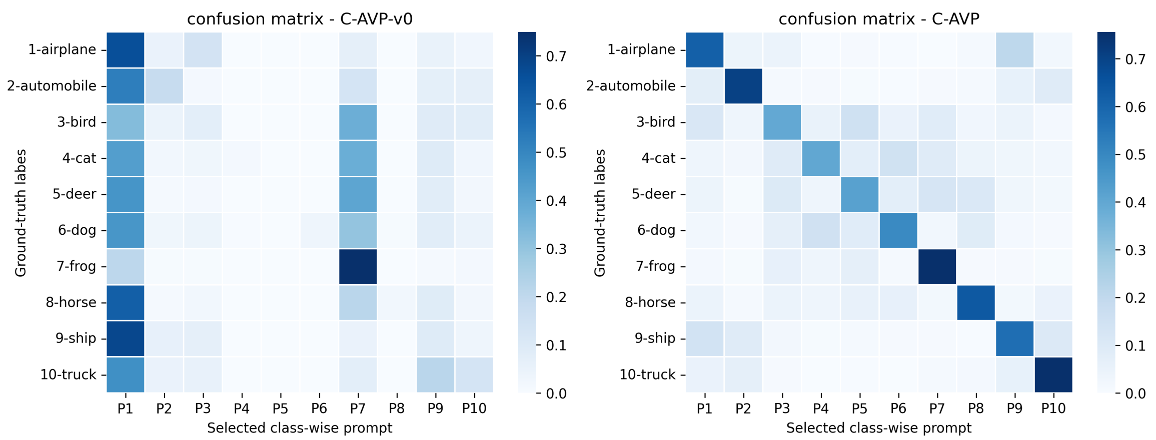
Figure 4.5. C-AVP visualization. One image is chosen from each CIFAR-10 class with the corresponding C-AVP.

Table 4.2. Sensitivity analysis of prompting regularization in C-AVP on (CIFAR-10, ResNet18).

Setting	$\ell_{\mathrm{C-AVP},1}$	$\ell_{\mathrm{C-AVP},2}$	$\ell_{\mathrm{C-AVP},3}$	Std ACC (%)	PGD-10 ACC (%)
S1	✘	✘	✘	19.69	13.91
S2	✓	✘	✘	22.72	13.01
S3	✘	✓	✘	40.01	25.40
S4	✘	✘	✓	17.44	11.78
S5	✓	✓	✘	57.03	32.39
S6	✓	✘	✓	26.02	15.80
S7	✓	✓	✓	**57.57**	**34.75**

4.4.6 Class-wise Prediction Error Analysis

Figure 4.6 shows a comparison of the classification confusion matrix. Each row corresponds to testing samples from one class, and each column corresponds to the prompt ('P') selection across 10 image classes. As we can see, our proposal outperforms C-AVP-v0 since the former's higher main diagonal entries indicate less prompt selection error than the latter.



(a) C-AVP-v0. **(b)** C-AVP.

Figure 4.6. The test-time predictions of C-AVP-v0 vs. C-AVP on (CIFAR-10, ResNet18).

Comparisons with other test-time defenses. In Tab. 4.3, we compare our proposed C-AVP with three test-time defense methods selected from Croce et al. [31]. Note that all methods are applied to robustifying a fixed, standardly pre-trained ResNet18. Following Croce et al. [31], we divide the considered defenses into different categories, relying on their defense principles (i.e., IP or MA) and needed test-time operations (i.e., IA, AN, and R). As we can see, our method C-AVP falls into the IP category but requires no involved test-time operations. This leads to the least inference overhead. Although there exists a performance gap with the test-time defense baselines, we hope that our work could pave a way to study the pros and cons of visual prompting in adversarial robustness.

Table 4.3. Comparison of C-AVP with other SOTA test-time defenses. Per the benchmark in Croce et al. [31], the involved test-time operations in these defenses include: IP (input purification), MA (model adaption), IA (iterative algorithm), AN (auxiliary network), and R (randomness). And inference time (IT), standard accuracy (SA), and robust accuracy (RA) against PGD-10 are used as performance metrics.

Method	IP	MA	IA	AN	R	IT	SA (%)	RA (%)
[123]	✓	✘	✓	✘	✘	518 ×	85.9	0.4
[159]	✓	✘	✓	✓	✓	176 ×	91.1	40.3
[25]	✘	✓	✓	✓	✘	59 ×	56.1	50.6
C-AVP	✓	✘	✘	✘	✘	**1.4** ×	57.6	34.8

4.5 LIMITATIONS

While visual prompting can be effective in improving adversarial robustness, it also has some limitations. Here are a few limitations to consider:

- Single attack defense limitation: The visual prompt may only defend against the attack it is trained on, i.e. PGD. The effectiveness on other attacks depends on transferability.
- Number of classes limitation: This work only shows the result of on simple CIFAR-10 dataset which has only 10 classes. It is a simple dataset because the classes are not hierarchical and has a low complexity from the image size and number of objects per image perspective.

4.6 SUMMARY

In this work, we develop a novel VP method, i.e., C-AVP, to improve the adversarial robustness of a fixed model at test time. Compared to existing VP methods, this is the first work to peer into how VP could be in adversarial defense. We show the direct integration of VP into robust learning is not an effective adversarial defense at test time for a fixed model. To address this problem, we propose C-AVP to create ensemble visual prompts and jointly optimize their interrelations for robustness enhancement. We empirically show that our proposal significantly reduces the inference overhead compared to classical adversarial defenses which typically call for computationally-intensive test-time defense operations.

Chapter 5

Manifold Mismatch: Misalignment of Adversarial Examples with the Learned Space of the Diffusion Model

In recent years, diffusion models (DMs) have drawn significant attention for their success in approximating data distributions, yielding state-of-the-art generative results. Nevertheless, the versatility of these models extends beyond their generative capabilities to encompass various vision applications, such as image inpainting, segmentation, adversarial robustness, among others. This study is dedicated to the investigation of adversarial attacks through the lens of diffusion models. However, our objective does not involve enhancing the adversarial robustness of image classifiers. Instead, our focus lies in utilizing the diffusion model to detect and analyze the anomalies introduced by these attacks on images. To that end, we systematically examine the alignment of the distributions of adversarial examples when subjected to the process of transformation using diffusion models. The efficacy of this approach is assessed across CIFAR-10 and ImageNet datasets, including varying image sizes in the latter. The results demonstrate a notable capacity to discriminate effectively between benign and attacked images, providing compelling evidence that adversarial instances do not align with the learned manifold of the DMs.

5.1 BACKGROUND

In this section, we address the challenge of identifying adversarial images. We provide a comprehensive overview of the related work, with a particular focus on diffusion model-based generated images.

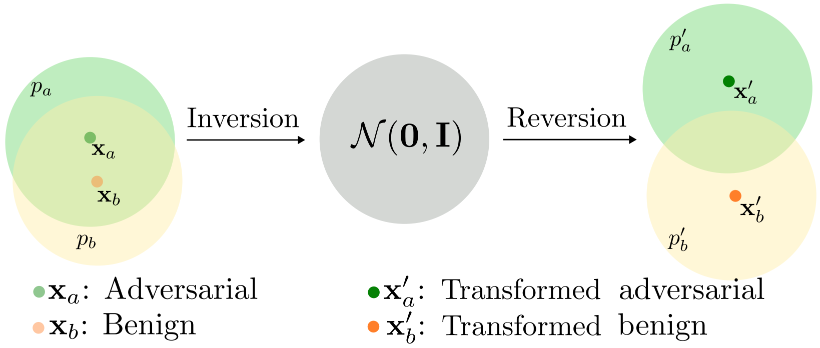


Figure 5.1. Illustration of the difference between an adversarial and a benign sample when subjected to the transformation process. $p_a(\boldsymbol{x})$ represents the distribution of adversarial images, while $p_b(\boldsymbol{x})$ represents the distribution of benign images. Using the inversion and reversion process of a purple DDIM [127], $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ become $\boldsymbol{x}'_a$ and $\boldsymbol{x}'_b$, respectively. These transformed counterparts now belong to distinct distributions, namely $p'_a(\boldsymbol{x})$ and $p'_b(\boldsymbol{x})$, characterized by a significantly reduced overlap. Therefore, this results in a distinct representation of adversarial samples compared to benign samples.

5.1.1 Prelimineries

A longstanding problem of deep learning (DL) is the vulnerability to adversarial examples [46, 130]. These instances are maliciously crafted by introducing imperceptible perturbations to natural examples, inducing in this way erroneous predictions in DL models, such as misclassifications.

Given the potential security threats posed by the lack of adversarial robustness in real-world applications, substantial efforts have been dedicated to developing defenses against adversarial examples. Various strategies have been explored, including addressing obfuscated gradients [5], adversarial training (AT) [92, 132, 163], image denoising [119, 128], and certified defenses [27, 108, 144].

The exploration of adversarial attacks on larger image dimensions is progressing rapidly, as evidenced by recent studies [22, 132, 165]. This progress is particularly focused on enhancing transferability to broaden their scope of application. In contrast, defenses [6, 135] targeting adversarial examples [31, 36] concentrate on adjusting input or hyperparameters during test-time. However, these methods are restricted by computational demands, limiting their effectiveness to smaller image sizes (resolutions), such as CIFAR-10 [67]. Common defense methods are certifiable robustness [148], randomized smoothing [164], and adversarial training [36]. Consequently, supervised learning defense methods are more successful on larger image sizes but only show a proof-of-concept of the detection capabilities rather than being a defense.

Diffusion models (DMs) have emerged as a powerful family of generative models, with

Denoising Diffusion Probabilistic Models (DDPMs) [55, 126] standing as pioneers in this field. DDPMs have set up a new paradigm in image generation, showcasing a robust capability to produce high-quality images [112]. Another important model is the Denoising Diffusion Implicit Model (DDIM) [127], known for speeding up the generation and improving sample quality. Recently, DMs have found applications within the adversarial attack domain. In particular, Nie et al. [102] developed a DM-based method to purify the input images, i.e., removal of noise or adversarial perturbations, from adversarial examples before entering a classifier. Expanding upon this, a recent study by Chen et al. [21] introduces a generative classifier constructed from a pre-trained DM, achieving a high level of robust accuracy against norm-bounded adversarial perturbations.

However, the aforementioned approaches concentrate on perfectly purified images. In this paper, we investigate the impact of adversarial examples after transforming the inverse and reverse processes. As opposed to purification methods, the transformation does not have to be perfect. The hypothesis behind this is that transformed adversarial images yield a certain pattern (fingerprint) resulting from a shift in the manifold learned by the DM on the benign distribution. A similar effect has been observed in Generative Adversarial Networks (GANs) [120]. Our objective is to undertake a similar investigation for DMs, requiring the data to transform a pre-trained DM (see fig. 5.1). The input image, represented by $\mathbf{x}$, undergoes an inversion process, mapping it to the noise vector $\mathbf{x}_T$ in the noise space $\mathcal{N}(0, \mathbf{I})$. This mapping is utilized as an initialization for the subsequent process known as reverse, wherein the denoising of the image occurs from the latent space back to its original image domain. This transformation offers a reliable pipeline for differentiating between attacked and benign images. By training a simple binary off-the-shelf classifier on the transformed samples, it becomes possible to detect adversarial examples with ease. To evaluate the effectiveness of the detector, we take the ImageNet [34] and various WB and BB attacks, each characterized by different hyperparameters.

5.1.2 Related Work

In this subsection, we are going to adversarial attacks, defenses, and then the contribution of DMs to adversarial robustness. n this section, we introduce adversarial attacks and defenses, and finally, we outline the contribution of DMs to adversarial robustness.

Adversarial Attacks

Convolutional neural networks are known to be susceptible to adversarial attacks, i.e. small perturbations of the input images that are optimized to fool the network's decision. In section 1.3, we have discussed most common whitebox attacks. We expand attack methods by a variante of the PGD attack and DM-based and some blackbox attacks. The most relevant ones are presented below.

Masked PGD [151] is a variation of the PGD attack, in which perturbations are confined to a specific area in the image rather than impacting the entire image. With a simple mask, a patch region can be defined to attack. As shown in eq. (5.1), only pixels inside the patch region $[x, y, h, w]$ will be modified by the PGD:

$$\boldsymbol{x}_{adv}^{(t+1)} = \mathrm{Clip}_{\boldsymbol{x}}\left(\boldsymbol{x}_{adv}^{(t)} + \alpha \cdot \mathrm{sign}(\nabla_{\boldsymbol{x}} \boldsymbol{J}(\boldsymbol{x}_{adv}^{(t)}, y, \boldsymbol{\theta})[patch]), \epsilon\right). \tag{5.1}$$

In this context, the term patch denotes the region specified as $[x : x + h, y : y + w]$ with the provided values of $[x, y, h, w]$. Masked PGD is capable of targeting object detectors and video classifiers by extracting gradients from their respective loss functions.

DiffAttack [22] is the first adversarial attack based on DMs [55]. Unlike traditional adversarial attacks that directly manipulate pixel values, DiffAttack focuses on creating human-insensitive perturbations embedded with semantic clues, making them difficult to detect. DiffAttack crafts perturbations in the latent space of DMs, whose properties achieve very high imperceptibility and transferability. DiffAttack leverages the DDIM inversion process [101], where the clean image is mapped back into the diffusion latent space by reversing the sampling process. The image in the latent space is directly perturbed. To create a final attacked image, the latent space must be transformed back into an image. Image editing approaches [29, 98] propose the image latent can gradually shift to the target semantic space during the iterative denoising process.

Natural Evolution Strategy (NES) [60] is a method used in blackbox adversarial attacks on machine learning models. It involves estimating the gradient by averaging the confidence scores of randomly sampled nearby points and then using projected gradient descent to perturb an image of the target class until it is sufficiently close to the original image. NES can be applied to the embedding space, which accelerates the search process for adversarial examples. This approach has been shown to efficiently generate perturbations for a target model, making it effective in compromising the integrity of machine learning models.

Bandits [61] is a technique employed for generating adversarial examples within a blackbox setting, where only limited information about the target model is accessible. This approach harnesses bandit optimization, a form of online optimization featuring bandit feedback, to effectively generate adversarial examples with fewer queries and higher success rates compared to existing methods. The Bandits attack seamlessly integrates gradient priors, which are both data-dependent and time-dependent priors, to improve the efficiency and efficacy of blackbox attacks. Through the incorporation of bandit optimization and gradient priors, this methodology seeks to optimize the generation of adversarial examples while minimizing the requisite number of queries to compromise the target model. The Bandits attack has demonstrated promising outcomes boosting the performance of blackbox adversarial attacks, underscoring its significance as an area of research in adversarial machine learning.

Adversarial Defenses

In recent years, a variety of strategies have emerged to defend against adversarial attacks. Initially, defenses focused on supervised methods for detecting adversarial examples. Then, adversarial training gained popularity as another supervised learning approach. Later, more robust defenses have been developed to counter adaptive attacks [135]. More recently, there has been a shift towards exploring defenses that are adaptive at test-time, as highlighted in the study by Croce et al. [31]. However, adversarial defenses are always one step behind adversarial attacks, since adversarial attacks have a strong ability to transfer effectively across different datasets and models.

Adversarial Detection presents a computationally efficient alternative to adversarial training, focusing on distinguishing adversarial examples from benign ones to mitigate misclassifications. One notable approach is SpectralDefense (SD) [52], which analyzes the frequency domain representation of input images and feature maps to identify adversarial attacks. By leveraging the magnitude spectrum and phase of Fourier coefficients, this method achieves high detection rates. Another one is multiLID [86] which is an improvement of the LID (Local Intrinsic Dimensionality) [88, 91] in terms of detection rates. CD-VAE (Class-Dependent Variational Auto-Encoder) [153] offers an alternative approach by training a variational auto-encoder to extract class-dependent information from images, enhancing adversarial detection. CD-VAE consistently outperforms traditional approaches, including Kernel Density (KD) [15], LID [91], and Mahalanobis distance (M-D) [73], providing valuable insights into the realm of adversarial attack detection.

Adversarial Training (AT) [92, 110, 163] might be the most effective method, which trains neural networks using adversarial augmented data. Noteworthy benchmark leaderboards, such as ARES-Bench[1] and RobustBench[2], are actively tracking advancements in this area. There is a need to address the trade-off between accuracy and resilience against adversarial examples [78]. Despite their popularity, these models often demonstrate robustness primar-

[1] ml.cs.tsinghua.edu.cn/ares-bench
[2] robustbench.github.io

ily to specific attacks they are trained against, exhibiting limited generalization ability to unforeseen threats, as highlighted in [71, 134].

Defenses at Test-time differ from the aforementioned static defense methods (detection and adversarial training) as their inputs and parameters adapt during inference. To adapt the defense parameters, Croce et al. [31] evaluate optimization-based methods. The study reveals considerable difficulty in defending at test-time, with observed accuracy drops of up to 0%. The evaluation compromises: I) Obfuscated gradients [5] where BPDA (Backward Pass Differentiable Approximation) can be used to attack non-differentiable preprocessing-based defenses. II) Randomness: The inclusion of randomized elements, such as Expectation over Transformation (EoT) [6], increases the cost for attackers. This is significant because attacks usually presume a global perspective on the input image.

Transferability of Adversarial Examples [49] across different model architectures or training datasets is the ability that makes these attacks so effective. The transferability property of adversarial examples makes blackbox attacks a powerful methodology, even in cases where the attacker has limited knowledge of the victim network. Furthermore, it is an area of active research that continuously seeks to improve transferability via new methodologies such as mitigating attention shift [37], translation invariant attacks [38], tune variance [140], more fine-grained perturbations through diffusion models like DiffAttack [22], and direction tuning [156]. In contrast, defense strategies focus on adapting during test-time. This adaptation is necessitated by the computational complexity involved in dealing with smaller images. In comparison, there are only a few defenses to aim to mitigate transferability, and if they focus on scaled datasets, i.e. [133] or examined in [31]. At this end, attackers have an easier role because they only have to lead to misclassification to be successful, whereas defenders also need to keep up the correct prediction.

Diffusion Models for Adversarial Robustness

DMs have been applied within the domain of adversarial robustness, demonstrating their efficacy and versatility in addressing challenges related to the security and resilience of systems against adversarial attacks. The DiffPure approach [102] utilizes DMs to purify adversarial perturbations. This purification process involves the addition of Gaussian noises to input images, followed by the denoising of the images. Recently, Yang et al. [154] claim that Diffpure is still not that protective against unseen threats. One potential explanation for this issue is the continued emphasis on discriminative classifiers, which may not capture the underlying structure of the data distribution. DMs have more accurate score estimation in the whole data space, where they explore a DM itself as a robust classifier. Moreover, DMs can also contribute to improving the certified robustness in conjunction with randomized smoothing [148]. Besides, the utilization of data generated by DMs has demonstrated an improvement in the performance of adversarial training [110, 142].

5.2 CONTRIBUTION

Our main contributions can be summarized as follows:

- We utilize the diffusion model transformation process applied to both adversarial and benign samples, enabling the discrimination between attacked and benign samples. This investigation includes a broad spectrum of image sizes.
- The employed classifier demonstrates the ability to effectively distinguish between multiple types of attacks. This implies a discerning capacity to identify not only whether an image has been subjected to an attack, but also the specific nature of the attack itself.
- We explore the transferability of the detector by assessing its performance on other transformed images.

5.3 METHOD

In this paper, we conduct a thorough investigation of adversarial examples passing through a pre-trained DM, specifically of DDIM. It is important to understand that the initial noise vector is replaced by the image to reconstruct the latent space as Wang et al. [141] already have shown in their work for deepfake detection. This initialization should be enough to get reconstructed images from an unconditional DM. Moreover, the transformation of adversarial and benign samples should be different, although there are just tiny pixel changes crafted by an attack method. Our research question compromises these effects of small changes in the initial distribution and consequently on the final images.

This section is organized as follows: We begin with reviewing DDPMs, and the inversion and reconstruction process of the DDIM [127]. Then, we present details of adversarial image detection including the training procedure. The conceptual framework of our proposal is illustrated in algorithm 2.

5.3.1 Problem Definition

In the case of a DM, we can think of the input data as a manifold in a high-dimensional space, where each data point is a point on the manifold. The diffusion model learns to map this manifold to a lower-dimensional latent space, where each point in the latent space corresponds to a point on the manifold.

Now, let's consider an adversarial example, which is a data point that has been deliberately modified to cause misclassification. In the context of a diffusion model, we can think of an adversarial example as a point on the manifold that is close to the original data point, but not necessarily on the same manifold.

The question is, are adversarial examples outliers or inliers in the latent space of the diffusion model?

An outlier, in the context of a diffusion model, is a point in the latent space that is far away from the manifold learned by the model. In other words, an outlier is a point that the model has not learned to generate, and is likely to be a point that the model cannot generate accurately.

An inlier, on the other hand, is a point in the latent space that is close to the manifold learned by the model. In other words, an inlier is a point that the model has learned to generate, and is likely to be a point that the model can generate accurately.

5.3.2 Prelimaries

In the following, we use the notations from DDIM [127] because we use this architecture throughout this paper and is also able to use the pre-trained weights from the DDPM architecture [55]: We note that in [55], a diffusion hyperparameter β_t is first introduced, and then relevant variables $\alpha_t := 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{t=1}^{T} \alpha_t$ are defined. From the DDIM paper we use the notation α_t to represent $\bar{\alpha}_t$ and also define

$$\beta_t = 1 - \frac{\alpha_t}{\alpha_{t-1}}. \tag{5.2}$$

Denoising Diffusion Probabilistic Model (DDPM) is initially proposed in [126], inspired by non-equilibrium thermodynamics. This innovation has marked a significant advancement in image generation, yielding noteworthy results [35, 55, 101, 113].

DDPMs define a Markov chain of diffusion steps, progressively introducing Gaussian noise to the data. This iterative process continues until the data transforms, ultimately converging into an isotropic Gaussian distribution. This defines the forward process of DMs as:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}\left(\boldsymbol{x}_t; \sqrt{\frac{\alpha_t}{\alpha_{t-1}}}\boldsymbol{x}_{t-1}, (1 - \frac{\alpha_t}{\alpha_{t-1}})\mathbf{I}\right), \tag{5.3}$$

in which $\boldsymbol{x}_t$ denotes the noisy image at the t-th step and let $\alpha_1, \ldots, \alpha_T \in (0, 1]^T$ be a predetermined decreasing schedule, where T represents the total number of steps. An essential property conferred by the Markov chain is the direct derivation of x_t from x_0, as follows:

$$q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}\left(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_0, (1 - \alpha_t)\mathbf{I}\right), \tag{5.4}$$

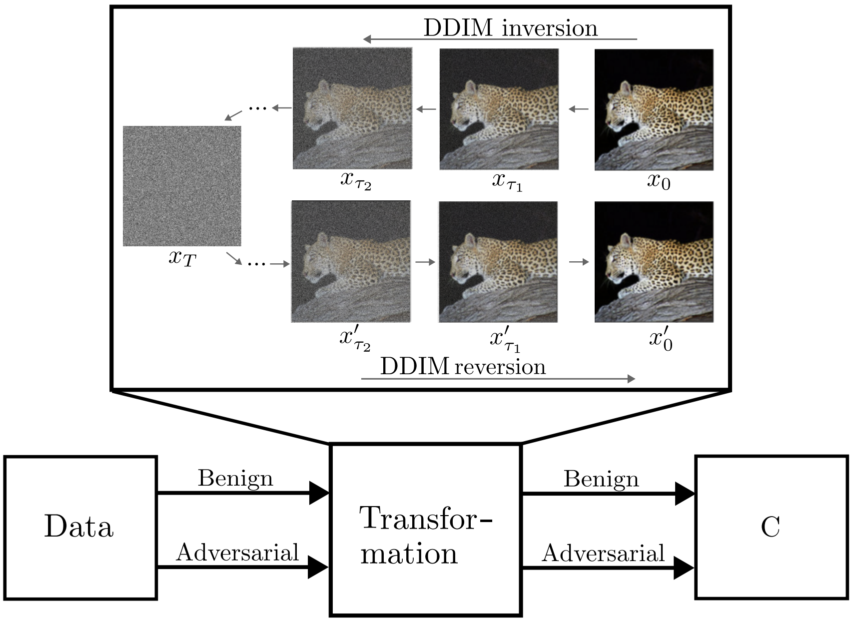


Figure 5.2. Illustration from the data generation over the transformation through a pre-trained DM to train a binary classifier C. Adversarial and benign samples are separately transformed. The transformation implies that the input image $\mathbf{x}_0$ is first gradually inverted into a noise image $\mathbf{x}_T$ using DDIM inversion [127], and then it is denoised step by step until the transformed $\mathbf{x}'_0$ is obtained, as illustrated in eq. (5.11).

Next, the models are trained to reverse this diffusion process, enabling the generation of samples from the noise—a process termed the reverse process. The reverse process in [55] is also defined as a Markov chain:

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}(\boldsymbol{x}_{t-1}; \boldsymbol{\mu}_\theta(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_\theta(\boldsymbol{x}, t)). \tag{5.5}$$

DMs employ a network $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ to approximate the real distribution $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$. The primary goal of optimization is to achieve a sampling and denoising process as outlined below:

$$L_{\text{simple}}(\boldsymbol{\theta}) = \mathbb{E}_{t,\boldsymbol{x}_0,\boldsymbol{\epsilon}}\left[\left\|\boldsymbol{\epsilon} - \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\sqrt{\alpha_t}\boldsymbol{x}_0 + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}, t)\right\|_2^2\right]. \tag{5.6}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$.

Denoising Diffusion Implicit Model (DDIM) [127] proposes a method for accelerating the iterative process without the Markov hypothesis. The modified reverse process in DDIM is defined as follows:

$$\boldsymbol{x}_{t-1} = \underbrace{\sqrt{\alpha_{t-1}}\left(\frac{\boldsymbol{x}_t - \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)}{\sqrt{\alpha_t}}\right)}_{\text{predicted } \boldsymbol{x}_0} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2}\cdot\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)}_{\text{direction pointing to } \boldsymbol{x}_t} + \underbrace{\sigma_t\boldsymbol{\epsilon}_t}_{\text{random noise}}. \tag{5.7}$$

In the given equation, when $\sigma_t = 0$, the term involving $\sigma_t\boldsymbol{\epsilon}_t$ becomes zero. In this case, the reverse process becomes deterministic (backward process) because the term involving noise, $\sigma_t\,\epsilon_t$, is zero. In a deterministic process, each input uniquely determines the corresponding output. Therefore, when $\sigma_t = 0$, the reverse process is fully determined by the given formula, and there is no randomness introduced during the backward process. Furthermore, when T is large enough (e.g. $T = 1000$), eq. (5.7) can be seen as Euler integration for solving Ordinary Differential Equations (ODEs):

$$\frac{\boldsymbol{x}_{t-\Delta t}}{\sqrt{\alpha_{t-\Delta t}}} = \frac{\boldsymbol{x}_t}{\sqrt{\alpha_t}} + \left(\sqrt{\frac{1-\alpha_{t-\Delta t}}{\alpha_{t-\Delta t}}} - \sqrt{\frac{1-\alpha_t}{\alpha_t}}\right)\boldsymbol{\epsilon_\theta}(\boldsymbol{x}_t, t). \tag{5.8}$$

Suppose $\sigma = \sqrt{1-\alpha}/\sqrt{\alpha}$, $\bar{\boldsymbol{x}} = \boldsymbol{x}/\sqrt{\alpha}$, the corresponding ODE becomes:

$$d\bar{\boldsymbol{x}}(t) = \boldsymbol{\epsilon_\theta}\left(\frac{\bar{\boldsymbol{x}}(t)}{\sqrt{\sigma^2+1}}, t\right) d\sigma(t). \tag{5.9}$$

Then, the inversion process (from $\boldsymbol{x}_t$ to $\boldsymbol{x}_{t+1}$) is then later reversed:

$$\frac{\boldsymbol{x}_{t+1}}{\sqrt{\alpha_t+1}} = \frac{\boldsymbol{x}_t}{\sqrt{\alpha_t}} + \left(\sqrt{\frac{1-\alpha_{t+1}}{\alpha_{t+1}}} - \sqrt{\frac{1-\alpha_t}{\alpha_t}}\right)\boldsymbol{\epsilon_\theta}(\boldsymbol{x}_t, t). \tag{5.10}$$

This procedure aims to acquire the corresponding noisy sample $\boldsymbol{x}_T$ for an input image $\boldsymbol{x}_0$. Nevertheless, performing step by step inversion or sampling is notably time-consuming. To speed up the DM sampling, DDIM [127] permits us to sample a subset of S steps $\tau_1, \ldots, \tau_S$, so that the neighboring $\boldsymbol{x}_t$ and $\boldsymbol{x}_{t+1}$ become $\boldsymbol{x}_{\tau_t}$ and $\boldsymbol{x}_{\tau+1}$, respectively, in eq. (5.7) and eq. (5.10).

5.3.3 Method Details

In this paper, we conduct a thorough investigation of adversarial examples passing through a pre-trained DM, specifically a DDIM. After the transformation (inversion and reversion), we examine the impact that the images have undergone, and our core assumption is that samples from the diffusion generation space $p_a(\boldsymbol{x})$ are slightly different reversed as from $p_b(\boldsymbol{x})$.

Given an input image $\boldsymbol{x}_0$, our objective is to classify whether it is adversarial or natural (benign). To achieve this, we utilize a pre-trained diffusion model, specifically a DDIM, and apply the inversion process, gradually introducing Gaussian noise (refer to eq. (5.10)). After T steps, $\boldsymbol{x}_0$ transforms into $\boldsymbol{x}_T$, which now belongs to an isotropic Gaussian noise distribution. Subsequently, we apply the reverse process (refer to 5.7) to convert the noisy image, resulting in a recovered version $\boldsymbol{x}$. The overall transformation is defined as:

$$\boldsymbol{x}'_0 = \mathbf{R}(\mathbf{I}(\boldsymbol{x}_0)), \tag{5.11}$$

where $\mathbf{I}(\cdot)$ represents the inversion process, and $\mathbf{R}(\cdot)$ denotes the reverse process.

To train a binary classifier differentiating between adversarial and benign samples, we apply this transformation to both types of samples. The outcomes are then used to train the binary classifier using binary cross-entropy loss, formulated as:

$$\ell(y, y') = -\sum_{i=1}^{N} \left(y_i \log(y'_i) + (1 - y_i) \log(y'_i) \right), \tag{5.12}$$

where N is mini-batch size, y is the ground-truth label, and y' is the corresponding prediction by the detector.

5.4 EXPERIMENTS

In this section, we begin by introducing the datasets, metrics, and the training procedure. Following this, we present and discuss an extensive collection of experiments.

5.4.1 Datasets

For our experiments, we create several datasets: Adversarial datasets corresponding to an attack denoted as $\mathcal{X}_\text{a}$ and a benign dataset denoted as $\mathcal{X}_\text{b}$. The adversarial datasets are crafted with a batch size of 50, giving particular attention to blackbox attacks due to their enhanced performance with larger batch sizes [129]. Specifically, we only take images where the applied attacks are successful. To create the datasets, we systematically gather 10,000 benign datasets and 10,000 datasets subjected to adversarial attacks, ensuring a complete absence of overlap between the two. Ultimately, we partition the datasets into training 80%, validation 10%, and test 10% sets.

CIFAR-10 [67]: We employ the CIFAR-10 dataset as our low-resolution dataset (size 32×32 pixels). The reverse process is performed using the DDIM CIFAR-10 L-hybrid model, accessible at the DDIM repository[3].

ImageNet [34]: We utilize the ImageNet dataset as our foundational dataset. To ensure a class-balanced representation, we curate a dataset by extracting 100 samples from each of the 100 classes, resulting in a total of 10,000 samples. The image sizes are chosen to align with the pre-trained unconditional diffusion models, specifically for sizes 256×256[4], and 512×512 pixels[5].

[3]github.com/openai/improved-diffusion, CIFAR-10 L-hybrid
[4]github.com/openai/guided-diffusion
[5]huggingface.co/lowlevelware/512x512_diffusion_unconditional

Compressed ImageNet [149]: In the context of the DiffAttack [22], we leverage the Compressed ImageNet dataset [165], as this attack has been optimized and previously assessed in other studies for adversarial robustness. The attack involves downsampling the input image from 256×256 to 224×224 pixels. To meet the required image size of 256×256 pixels for DM reverse steps, we employ zero-padding to the left and bottom of the image.

5.4.2 Evaluation Metrics

To comprehensively evaluate the efficacy of our approach, we employ a diverse set of standard metrics commonly utilized in detection scenarios. These metrics serve as quantitative measures, offering insights into the robustness and accuracy of the proposed method. In particular, we use the following metrics: area-under-curve (AUC), average precision (AP), true negative rate (TNR), false negative rate (FNR), true positive rate (TPR), false positive rate (FPR), precision, recall, and F1.

5.4.3 Training Procedure

In algorithm 2, we outline the training procedure for the adversarial detector. The initial step involves generating both adversarial and benign datasets (see in section 5.4.1). Then, we apply the transformation, using DM, to all data samples. Note that this procedure necessitates the use of specific dimensions, such as 32×32, 256×256, and 512×512 pixels, as the pre-trained DM is designed to process images of these sizes. The resulting transformed images are used for training the classifier C, employing either ResNet-50, originally pre-trained on ImageNet, or ResNet18[6], originally pre-trained on CIFAR-10. Throughout the training of the classifier, we maintain the respective image sizes, except for the transformed images of 256×256 pixels, which are randomly cropped to the dimensions of 224×224 to align with previous work. We pre-process the data accordingly: I) During training, the images fed into the network are randomly cropped and horizontally flipped with a probability of 0.5. II) During testing, the images are center-cropped.

5.4.4 Results and Discussion

This subsection offers a comprehensive analysis of the results from the proposed approach. We delve into a detailed discussion of the performance evaluation, with a specific focus on addressing a fundamental question: *Can the proposed methodology effectively distinguish between instances classified as under attack and those labeled as benign across various image resolutions?*

[6]huggingface.co/edadaltocg/resnet18_cifar10

Algorithm 2 Training of the adversarial detector.

Require: Benign dataset $\mathcal{X}_\text{b}$, Adversarial dataset $\mathcal{X}_\text{a}$

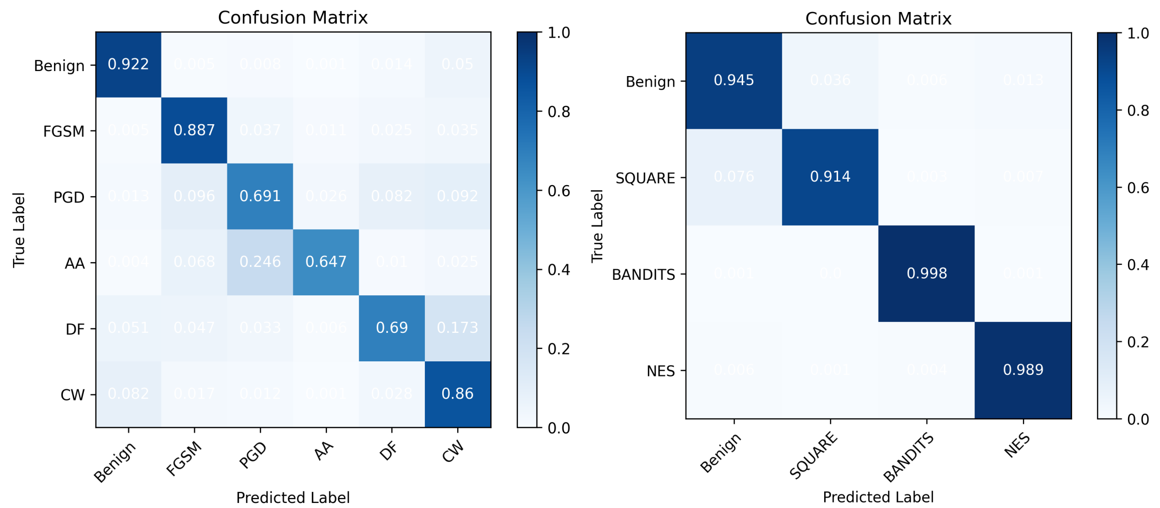
Ensure: Trained classifier C

1: Generate transformed dataset $\mathcal{X}_\text{tf}$ using the pre-trained diffusion model
2: Split $\mathcal{X}_\text{tf}$ into training set $\mathcal{X}_\text{train}$ and test set $\mathcal{X}_\text{test}$
3: Initialize ResNet-50 model C
4: Train C using $\mathcal{X}_\text{train}$
5: Evaluate C using $\mathcal{X}_\text{test}$
6: **return** C

In table 5.1, we conduct a comparative analysis of our method against various adversarial defenses, encompassing supervised learning (i.e. adversarial training), and input purification as principal components. Our method aligns with the supervised learning model category, which usually is determined as the first proof-of-concept. For our evaluations, we employ AA and PGD attacks, considering their prominence in related work and the ongoing debate on their efficacy in assessing the robustness of DM-based defenses [74]. Supervised learning (SL) methods typically serve as an initial benchmark to assess the capabilities of a method for a specific learning task, often yielding superior results compared to alternative approaches. Consequently, different evaluation metrics are employed, leading to nuanced comparisons. However, it is essential to note that other methods may have distinct focuses, such as evaluating robustness during test-time [20], resilience against unseen threats [71], or flexible to adaptive attacks [31]. Due to the divergent focuses, evaluations are often constrained to low-resolution datasets, such as CIFAR-10.

Turning our attention to a higher resolution dataset, ImageNet, in table 5.2 we observe that, apart from SL methods, the defense performance decreases when concentrating on static defense mechanisms without accounting for unseen threats. Notably, while multiLID [86] and SD [85] represent straightforward defense strategies, CD-VAE [153] emerges as a more intricate method by using the common information per class extracted from a GAN.

Furthermore, we provide an extensive examination of the ImageNet dataset, exploring various adversarial attacks. In this analysis, we aim to assess the generalizability of the employed method presented in table 5.3. This involves assessing its performance across various attacks and varying the hyperparameter ϵ sizes when applicable. Additionally, we consider DiffAttack, a novel distance metric-based attack meticulously optimized for the compressed ImageNet dataset and known for its invisible perturbations. Continuing the evaluation, we extend the same rigorous procedure to blackbox attacks, to gain insights into the method's resilience across various adversarial scenarios. Remarkably, our approach consistently demonstrates promising detection outcomes, showing its efficacy in mitigating



(a) Identification of with-box attacks.

(b) Identification of blackbox attacks.

Figure 5.3. Identification. The classifier for whitebox attacks has trouble distinguishing PGD and AA, but also DF. Benign examples can be clearly distinguished from attacked ones. The classifier for blackbox attacks can clearly distinguish the data transformations of each attack method.

diverse types of attacks.

This variation of methods, attacks, and datasets ensures a comprehensive evaluation of method performance across different scales and contexts. Nevertheless, we observe that adversarial examples, characterized by subtle pixel changes, impact the DM transformation (see 5.5.3). This influence extends to the DM's reverse process, leading to the emergence of identifiable and learnable patterns.

5.5 ABLATION STUDY

In this section, we present an ablation study to evaluate the pattern capabilities derived from the transformation (inverse and reverse steps of DDIM). Our study focuses on investigating the following questions: *I) How many reverse steps are required for uncovering the adversarial examples? II) Can these adversarial examples be identified as a unique fingerprint, and what are the transferability properties?*

5.5.1 Impact of the Diffusion Reverse Steps

In DDPM architecture, the reverse process is notorious for its computational demands. By default, we set the reverse steps to 1000 steps. However, in this subsection, we analyze the impact of changing the number of reverse steps on the model's performance. Results shown in table 5.4 indicate that reducing the number of steps leads to a decline in accuracy. TNR and FPR are inversely proportional. On the other hand, if we set $T = 2000$, doubling the number of reverse steps only yields a marginal improvement in accuracy. This implies that the standard parameter ($T = 1000$) for reverse steps is appropriate.

Table 5.4. The relationship between denoising steps T and model accuracy. $T = 1000$ is the default value.

Attack	Steps	ϵ	ACC	AP	TNR	FNR	TPR	FPR
PGD	2000	1/255	96.1	99.53	95.9	3.7	96.3	4.1
	1000	1/255	93.15	98.32	91.3	5.0	95.0	8.7
	750	1/255	88.75	96.41	92.2	14.7	85.3	7.8
	500	1/255	80.3	89.88	82.8	22.2	77.8	17.2
	250	1/255	52.95	47.51	7.7	1.8	98.2	92.3

5.5.2 Identification and Transferability Capability Evaluation

In this section, our focus centers on investigating the transferability capabilities inherent in our method. To that end, we pose the following questions: *Can the discerning identification of each attack be reliably accomplished through the utilization of a multilabel classifier? Furthermore, does the transferability of our approach persist when confronted with unfamiliar data originating from other attacks?*

In our pursuit of addressing the identification question, we embark on an exploration of the efficacy of our approach applied to the ImageNet dataset. As shown in fig. 5.3a, we carefully create and analyze the confusion matrix, focusing specifically on whitebox attacks. The results of this identification process show significantly high accuracy scores for benign data. However, noticeable performance degradation is observed when dealing with PGD, AA, and DF attacks. Expanding our study to blackbox attacks, the confusion matrix shown in fig. 5.3b reveals identification results with very high accuracy scores. This thorough analysis not only explores how well our method works on ImageNet but also shows its strength when facing various types of attacks.

Lastly, we investigate the transferability capabilities of the binary classifier. Adversarial examples have the transferability property, which makes them more strong. A unique fingerprint would show that after the transformation, the transferability capabilities would be hampered.

Additionally, we analyze the transferability capabilities from one attack to another, while also augmenting the training dataset with diverse attacks. In fig. 5.4, the detector is trained on FGSM, PGD, AA (1/255), DF, and CW datasets, smaller epsilon sizes transfer better to

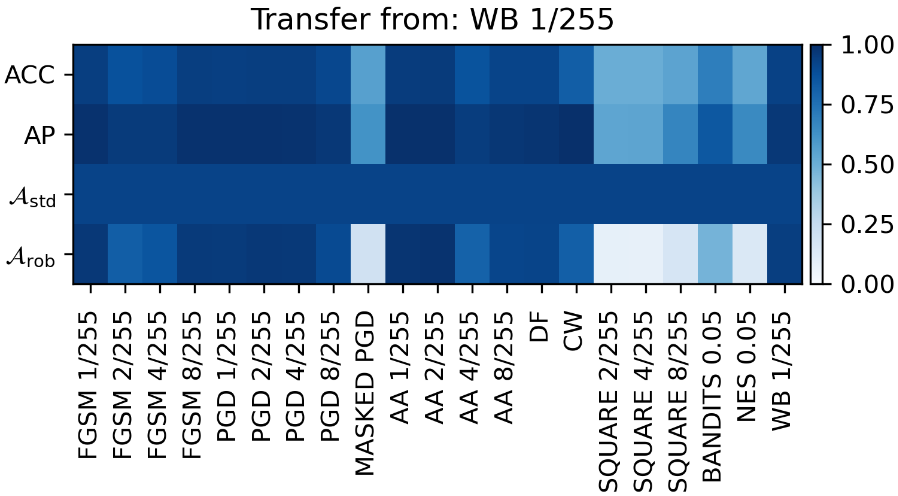


Figure 5.4. Transferability of a binary classifier trained on whitebox attacks ($\epsilon = 1/255$; without Masked PGD) and tested on all other datasets plotted on the x-axis.

larger ones. This augmentation helps to cover more attacks. The transferability to the same attacks but with a larger ϵ size is high. In contrast, the transferability towards unforeseen attacks is very low. Theoretically, the detector would rely on a huge amount of data but could be still bypassed because this detector is static.

Impact of higher Image Resolutions

In table 5.5, we present a comparative analysis of image resolutions, specifically 224 and 512 pixels. To the best of our knowledge, our proposed method is pioneering, presenting the initial results in adversarial detection for an image resolution of 512 pixels. The variability in detection accuracy across different image resolutions seems to be minimal. Nevertheless, it is noticeable that detection accuracy tends to increase with higher image resolution.

Table 5.1. CIFAR-10. The table displays the classification accuracy of our method along with a comparison to various types of adversarial defenses, including Adversarial Training (AT), Input Purification (IP), Supervised Learning (SL), and Auxiliary Network (AN). Attack methods are AA and PGD with $\epsilon = 4/255$; and AA* and PGD* with $\epsilon = 8/255$.

Method	**Type**	**Attack**	**ACC**	**AUC**	**TNR**	$\mathcal{A}_{\textbf{std}}$	$\mathcal{A}_{\textbf{rob}}$
AT-DDPM-ℓ_∞ [110]	AT	AA	76.08	-	-	88.87	63.28
AT-EDM-ℓ_∞ [142]	AT		82.13	-	-	93.36	70.91
RaWideResNet-70-16 [104]	AT		82.17	-	-	93.27	71.07
Visual Prompting [20]	AT		46.16	-	-	57.57	34.75
RDC [21]	IP		83.23	-	-	93.16	73.24
DiffPure [102]							
WRN-28-10	IP		79.83	-	-	89.02	70.64
WRN-70-16	IP		80.68	-	-	90.07	71.29
HEDGE [147]	IP	AA	79.62	-	-	89.16	70.07
	IP	PGD	79.11	-	-	89.16	69.04
AID Purifier [59]	IP, AN		70.42	-	-	88.28	52.56
Mao et.al. [94]			64.21	-	-	60.67	67.79
multiLID [86]	SL, AN	AA	96.43	99.37	94.11	99.86	**99.93**
	SL, AN	PGD	93.93	97.94	92.86	92.86	95.22
SD_{WB} [85]	SL, AN		97.25	**99.93**	95.14	99.54	95.08
SD_{BB} [85]	SL, AN		95.53	99.74	90.99	**100**	91.01
CD-VAE [153]							
KD (R(x))	SL, AN		-	99.30	96.56	-	-
LID (R(x))	SL, AN		-	97.57	87.54	-	-
M-D (R(x))	SL, AN		-	99.79	**99.13**	-	-
Ours	SL, AN	AA	**97.40**	99.73	97.51	97.54	97.32
Ours	SL, AN	PGD	95.40	99.03	96.11	96.08	94.81

Table 5.2. ImageNet with 224 pixels. The table displays the classification accuracy of our method along with a comparison to various types of adversarial defenses, including Adversarial Training (AT), Input Purification (IP), Supervised Learning (SL), and Auxiliary Network (AN). Attack methods are AA and PGD with $\epsilon = 4/255$; and AA* and PGD* with $\epsilon = 8/255$. We refer to other epsilon sizes of our proposed method on the other table 5.3.

Method	**Type**	**Attack**	**ACC**	**AUC**	**TNR**	$\mathcal{A}_{\textbf{std}}$	$\mathcal{A}_{\textbf{rob}}$
Swin-L [80]	AT	AA	69.24	-	-	78.92	59.56
Conv-Next-L [80]	AT		68.25	-	-	78.02	58.48
DiffPure [102]							
ResNet-50	IP	PGD	54.36	-	-	67.79	40.93
WRN-50-2	IP		57.78	-	-	71.16	44.39
DeiT-S	IP		58.41	-	-	73.63	43.18
multiLID [86]	SL, AN	AA*	99.46	99.98	99.29	98.91	99.64
multiLID [86]	SL, AN	PGD*	89.29	95.45	89.46	97.93	89.11
SD_{WB} [85]	SL, AN	AA	97.12	99.71	96.75	97.51	96.75
SD_{BB} [85]	SL, AN		83.27	83.27	59.25	91.04	59.25
CD-VAE [153]							
KD ($R(\boldsymbol{x})$)	SL, AN	PGD	-	**100**	96.56	-	-
LID ($R(\boldsymbol{x})$)	SL, AN		-	97.38	87.54	-	-
M-D ($R(\boldsymbol{x})$)	SL, AN		-	99.77	99.13	-	-
Ours	SL, AN	AA	95.75	99.63	94.0	94.0	97.5
Ours	SL, AN	PGD	**99.1**	99.96	**99.7**	**99.7**	**98.54**

Table 5.3. The table presents the classification accuracy on the ImageNet dataset. We use the pre-trained ResNet-18 model on ImageNet. *Compressed Imagenet [149]. The attacks are generated on a batch size of 50.

Attack	ϵ	**AUC**	**ACC**	**AP**	$\mathcal{A}_{\text{std}}$	$\mathcal{A}_{\text{rob}}$	**TNR**	**FNR**	**TPR**	**FPR**	**Prec**	**Rec**	**F1**
							Whitebox attacks						
FGSM	1/255	99.3	96.3	99.48	95.7	96.9	95.7	3.1	96.9	4.3	95.75	96.9	96.32
FGSM	2/255	99.96	99.15	99.97	99.0	99.3	99.0	0.7	99.3	1.0	99.0	99.3	99.15
FGSM	4/255	99.88	98.65	99.87	98.7	98.6	98.7	1.4	98.6	1.3	98.7	98.6	98.65
FGSM	8/255	100	99.95	100	100	99.9	100	0.1	99.9	0.0	100	99.9	99.95
PGD	1/255	99.09	93.15	98.32	91.3	95.0	91.3	5.0	95.0	8.7	91.61	95.0	93.27
PGD	2/255	99.84	98.2	99.92	98.7	97.7	98.7	2.3	97.7	1.3	98.69	97.7	98.19
PGD	4/255	99.96	99.1	99.97	99.7	98.5	99.7	1.5	98.5	0.3	99.7	98.5	99.09
PGD	8/255	100	99.75	100	99.8	99.7	99.8	0.3	99.7	0.2	99.8	99.7	99.75
Masked PGD	1	100	99.75	99.84	99.8	99.7	99.8	0.3	99.7	0.2	99.8	99.7	99.75
AA	1/255	99.69	96.3	99.55	96.2	96.4	96.2	3.6	96.4	3.8	96.21	96.4	96.3
AA	2/255	99.95	98.65	99.95	99.2	98.1	99.2	1.9	98.1	0.8	99.19	98.1	98.64
AA	4/255	99.63	95.75	99.58	94.0	97.5	94.0	2.5	97.5	6.0	94.2	97.5	95.82
AA	8/255	99.93	98.65	99.93	98.9	98.4	98.9	1.6	98.4	1.1	98.89	98.4	98.65
DF	-	98.5	93.3	98.59	93.0	93.6	93.0	6.4	93.6	7.0	93.04	93.6	93.32
CW	-	95.93	88.85	95.84	87.6	90.1	87.6	9.9	90.1	12.4	87.9	90.1	88.99
DiffAttack*	-	100	99.9	100	99.8	100	99.8	0.0	100	0.2	99.8	100	99.9
							Blackbox attacks						
Square	2/255	98.5	93.3	98.59	93.0	93.6	93.0	6.4	93.6	7.0	93.04	93.6	93.32
Square	4/255	98.67	96.0	98.92	97.3	94.7	97.3	5.3	94.7	2.7	97.23	94.7	95.95
Square	8/255	99.82	98.4	99.84	99.2	97.6	99.2	2.4	97.6	0.8	99.19	97.6	98.39
Bandits	0.05	100	99.65	100	99.3	100	99.3	0.0	100	0.7	99.3	100	99.65
NES	0.05	99.86	98.1	99.85	97.9	98.3	97.9	1.7	98.3	2.1	97.91	98.3	98.1

Table 5.5. Evaluation of various image resolutions on the ImageNet dataset.

Attack	**Size**	ϵ	**AUC**	**ACC**	**AP**	$\mathcal{A}_{\text{std}}$	$\mathcal{A}_{\text{rob}}$	**TNR**	**FNR**	**TPR**	**FPR**	**Prec**	**Rec**	**F1**
PGD	224×224	1/255	99.09	93.15	98.32	91.3	95.0	91.3	5.0	95.0	8.7	91.61	95.0	93.27
	512×512	1/255	99.81	97.95	99.8	97.4	98.5	97.4	1.5	98.5	2.6	97.43	98.5	97.96

5.5.3 Diffusion Model Transformations and Fourier Transformations

In this section, we investigate the reconstructed images by the Fourier transformation, i.e. fig. 5.5 and fig. 5.6. Adversarial examples detection in supervised manners in the Fourier domain is already heuristic proven in [85].

We transform every sample from the spatial domain to the 1D frequency domain, reducing it to a 1D Power Spectrum. This method is formed by a Discrete Fourier Transform followed by an azimuthally average. The transformation can be substantially optimized by employing the Fast Fourier Transform. Notice that after applying the transformation, we use only the power spectrum. A small shift after the first transformation can be recognized, as shown in fig. 5.6a. Therefore, we apply the inverse and reverse process several times and analyze it after each transformation with the FFT analysis. To this end, all FFT spectrums almost totally overlap after applying the recursive transformation process several times.

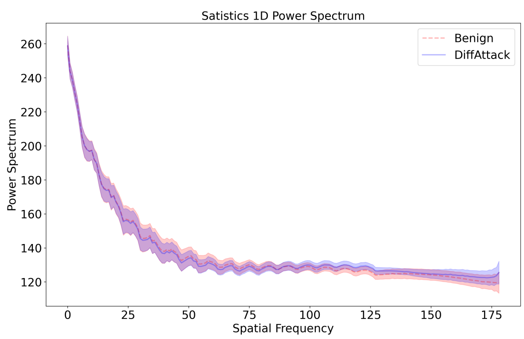
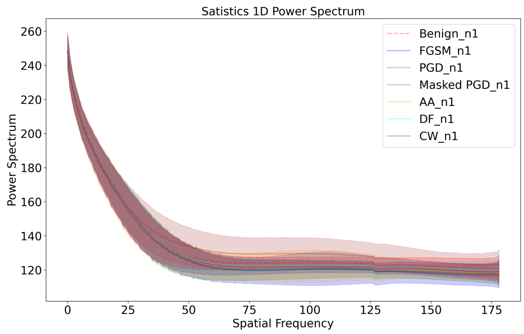
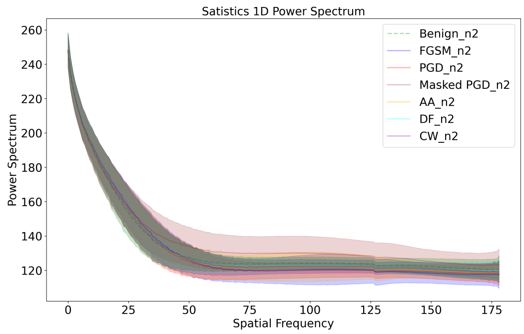


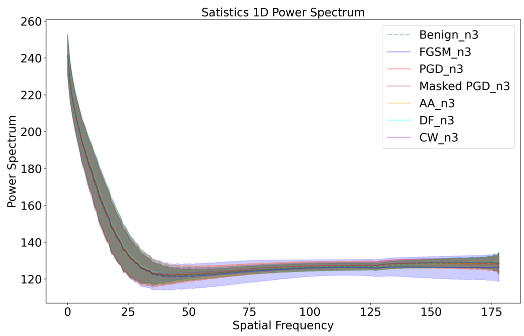
Figure 5.5. 1D power spectrum statistics from each sub-data for DiffAttack [22] on ImageNet-Compressed dataset [149].



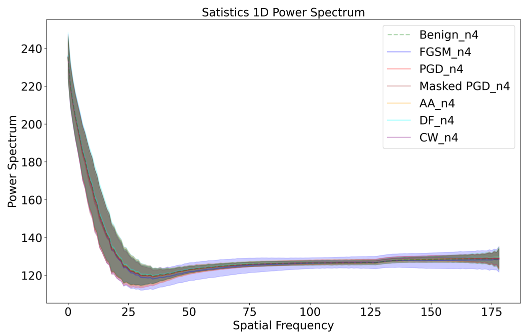
(a) n: 1.



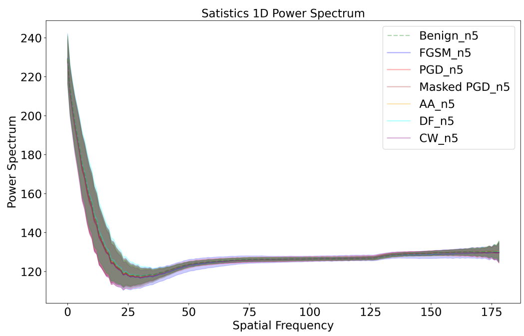
(b) n: 2.



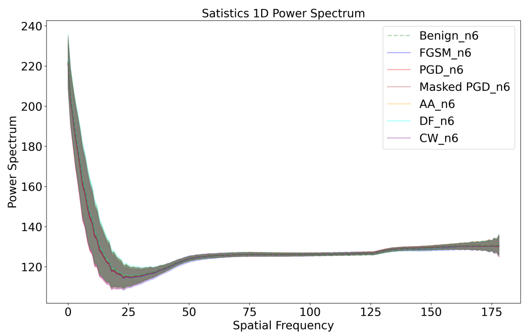
(c) n: 3.



(d) n: 4.



(e) n: 5.



(f) n: 6.

Figure 5.6. 1D power spectrum statistics from each sub-data ImageNet set for each attack method. The more often the transformation is recursively applied, the more the power-spectrum is overlapping.

5.6 LIMITATIONS

The limitations of our investigation relies mainly on the Diffusion Model's (DM) architecture. Hence, we can sum up the limitations in the following two points:

- Interpretability: Understanding the inner workings of DMs subject of current research [19, 75]. This lack of interpretability might make it difficult to trust or fine-tune the model for specific applications.
- Trade-off between speed and quality: DMs might need to trade off between the speed of generating results and the quality of those results. Adjusting parameters for faster generation may lead to a compromise in image quality and therefore harder to find the adversarial patterns.
- Defense mechanism: The objective of this study is to demonstrate the misalignment in the learned diffusion model's manifold. While we can identify adversarial examples, this alone does not suffice to classify it as a robust defense mechanism. The reason being that this approach lacks adaptability during test-time when faced with changes introduced by an attacker.

5.7 SUMMARY

In this work, we introduce an innovative approach that utilizes diffusion models to transform both adversarial and benign examples, subsequently employing these transformations to train a classifier. Our method unveils compelling evidence suggesting that adversarial examples do not belong to the learned manifold of the diffusion model (DM), highlighting its potential to uncover adversarial examples. We provide empirical evidence for this hypothesis and show that our proposed transformation acts as a reliable tool for uncovering adversarial perturbed samples and their fingerprints.

The evaluation of our proposed defenses involves the utilization of attack methods based on Projected Gradient Descent (PGD) or AutoAttack (AA), among others, as well as black-box attacks on the challenging ImageNet dataset up to an image size of 512×512 pixels. Various experiments show the effectiveness and the generalization of the method across different attacks and image sizes. Nonetheless, it is important to acknowledge its role as a complementary defense rather than a standalone solution. This is due to its limitations in countering adaptive defenses capable of dynamic adjustments during test-time, as well as the constraint in the transferability to unseen threats, similar to purification approaches.

We believe that our study contributes valuable insights into the fields of adversarial robustness of DMs and also explores the learned manifolds of DMs. The evaluation of our proposed defenses involves the utilization of attack methods based on Projected Gradient Descent (PGD) or AutoAttack (AA), among others, as well as blackbox attacks on the challenging ImageNet dataset up to an image size of 512×512 pixels. Various experiments show the effectiveness and the generalization of the method across different attacks and image sizes.

Chapter 6

Conclusion

This thesis proposes several approaches to investigate the threat “adversarial examples” in computer vision. In this final chapter, we summarize the main findings, and propose some interesting directions for future work.

6.1 Summary

This thesis starts with explaining deep learning as a crucial part of modern methods used in various computer vision projects. The widespread use of this technology is because it can independently learn unique features from a specific training dataset, specifically within the given problem area, as seen in tasks like image classification.

Our contribution focuses on analyzing adversarial example, in particular, we investigate different detection methods to distinguish between benign and adversarial attacked images. Originally, Goodfellow et al. [46] developed an adversarial attack (called FGSM) to fool image classifiers. Later, several whitebox attacks followed, such as BIM, PGD, DeepFool, C&W, and AutoAttack to list some of them.

Driven by the motivation to understand this problem, in our 2nd chapter, we analyze adversarial attacks through the lens of the Fourier domain. We decide to use whitebox attacks, which have knowledge about the attacked classifier and would be the strongest attack scenario. The results of our empirical evaluations show strong evidence that the widely used AutoAttack scheme for benchmarking the adversarial robustness of image classifier models on low-resolution data might not be a suitable setup in order to generalize the obtained results to estimate the robustness in practical vision applications. Even for lower choices of the perturbations size, AutoAttack still appears to modify target images beyond reasonable class boundaries. Additionally, the resolution of the benchmark images should not be neglected. On higher resolutions, AutoAttack detection is even more promising. Since AutoAttack is an ensemble of four attacks and the first attack is based on the gradient-based

PGD, which attacks most images, because only the unsuccessful attacked images are handed to the next attack method. On the other hand, DeepFool and C&W detection becomes more difficult to detect. The larger image size leverages the attack to find a more minimal perturbation. Moreover, the complexity of the datasets, such as a high number of classes, makes the detection through the Fourier lens more difficult.

In chapter 3, we introduce a simple light-weight detector, which leverages recent findings on the relation between networks's local intrinsic dimensionality (LID) and adversarial attacks. Based on the re-interpretation of the LID measure and several simple adaptations, we surpass the state-of-the-art on adversarial detection by significant margin and reach almost perfect detection results. Hence, we revisit the MLE estimate of the local intrinsic dimensionality which has been used in previous works on adversarial detection. An analysis of the extracted LID features and their theoretical properties allows us to redefine an LID-based feature using unfolded local growth rate estimates that are significantly more discriminative than the aggregated LID measure. We have shown outstanding results across different attacks and variety of epsilons. Moreover, we modified the LID so that the log features are trained by the logistic regression which lead to tremendous results as well. We believe that our approach have a potential to apply to many other related machine learning tasks. At this end, we show the transfer capabilities by comparing multiLID features trained on logistic regression and random forest on different attack methods.

In chapter 4, we leverage the idea to design an anti-pattern, which should be able to neutralize the adversarial examples across the dataset, since an attack minimal perturbate an image. This idea is derived from the field of natural language processing (NLP), where researchers in that field use the term "prompting". In computer vision, it means that small parts of an image is covered by an prompt and results in higher classification accuracy. We present as first visual prompting to improve adversarial robustness of a fixed, pre-trained model at test time. Compared to conventional adversarial defenses, visual prompting allows us to design universal i.e., data-agnostic) input prompting templates, which have plug-and-play capabilities at test time to achieve desired model performance without introducing much computation overhead. Although visual prompting has been successfully applied to improving model generalization, it remains elusive whether and how it can be used to defend against adversarial attacks. We explore this issue and demonstrate that the standard vanilla VP approach lacks effectiveness in countering adversarial attacks, primarily due to the limitations of a universal input prompt when facing targeted adversarial perturbations unique to each sample. To address this limitation, we introduce a novel visual prompt technique called Class-wise Adversarial Visual Prompting (C-AVP). This method involves generating class-specific visual prompts, allowing us to harness the advantages of ensemble prompts while also optimizing their interactions to enhance the model's robustness. Our experimental results clearly indicate the superiority of C-AVP over the traditional VP method. We observe a significant improvement in standard accuracy by a factor of 2.1 and a robust accuracy improvement of 2 times. In addition, when compared to traditional test-time defense mechanisms, C-AVP offers a remarkable 42-fold acceleration in inference time.

Finally in chapter 5, we have conducted an in-depth investigation into the effective distinction between adversarial examples and benign samples, presenting convincing evidence that adversarial instances do not belong to the learned manifold of the diffusion model (DM). Our novel approach involves utilizing the DM to transform both adversarial and benign examples, and then leveraging these transformations to train a classifier. The underlying hypothesis is that adversarial samples exist outside the DM's learned data manifold, regardless of the specific attack mechanism. To validate our hypothesis, we employed an evaluation framework that incorporates various attack methods, including Projected Gradient Descent (PGD) and AutoAttack (AA), as well as blackbox attacks on the challenging ImageNet dataset. Our experiments demonstrate that our approach achieves remarkable detection accuracy, outperforming traditional defection algorithms. However, it is important to note that our approach is not without limitations. Specifically, it lacks the essential properties of adaptive defenses that dynamically adapt during testing, making it unsuitable as a standalone defense mechanism. Moreover, we observe limitations in transferability to unseen threats, similar to purification approaches. Despite these limitations, our paper contributes valuable insights into the realms of adversarial robustness of DMs and the exploration of their learned manifolds. While our approach has limitations, it provides valuable insights adversarial examples do not belong of the DM's learned manifolds, paving the way for future research in this area.

6.2 Future Work

Throughout this thesis, we have stressed the importance of the trustworthiness of deep learning classification models For the threat, adversarial examples, we can apply the following scheme: We generate real and malicious samples (adversarial examples or synthetic generated data), extract features of both data and train a classifier. This approach is supervised and is only effective on known malicious methods. First we discuss, the future work of adversarial robustness and then of importance of the detection of synthetic data.

In this section, we discuss the future work for each detection method regarding adversarial examples. In the section 1.3, we describe our selection of adversarial attacks. Note that adversarial attacks are diverse: Adversarial attacks come in many forms, including but not limited to, whitebox, blackbox, transferable, targeted, and non-targeted attacks to mention few of the properties. Each type of attack targets different vulnerabilities in machine learning models, making it difficult to design a single detector that can effectively counter all of them. In the following, we discuss our investigations and direct to probable future work:

- **SpectralDefense**: With this detection method, we have analyzed adversarial examples in the Fourier domain, especially AutoAttack (AA) from RobustBench. We could show that most AA is optimized towards CIFAR-10 and on other datasets the Fourier analysis depicts even better learnable features for binary classification. However, there are still open questions about attacks which would generate adversarial examples in different frequencies. In our experiments, we could see that, adversarial examples are a mid frequencies problem. This is a result of the mechanics of CNNs in regards to image data. CNNs are very good at learning the low frequencies of a dataset, while high frequencies are considered as less informative for CNNs. It seems that adversarial attacks tend to place adversarial examples on the transition from low to high frequencies. Jia et al. [62] explored the effectiveness of adversarial examples in low, mid and high bands. Low bands outperforming in their experiments best, but these perturbations are more visible in spatial domain. SpectralDefense needs to be adaptive to detect adversarial examples at different frequencies at test time to close the attack surface. Event though, on larger image sizes SpectalDefense becomes ineffectively on DeepFool and C&W attack and further research could be done towards this direction.

- **Visual prompting**: Visual prompting is successful to improve detection systems or downstream pre-trained models to other tasks. In our work, we investigated if we can defeat adversarial examples with visual prompting at test-time. We tested our experiments only on a small-scale dataset CIFAR-10 with 32×32 pixels. The biggest obstacle was that the visual prompt's parameter size is very limited to find the trade-off between defeating the adversarial example and keep clean accuracy. It would be also interesting if on larger images and therefore a larger prompt size might improve the results. We do not believe that replacing class-wise prompts with a universal prompt and adding an additional learnable layer, which introduces more parameters, is beneficial. This is because adversarial training involves training the entire model, which inherently has greater capacity to learn. Moreover, there could be another option to potentially further improve the clean accuracy. Recently, in the field of backdoor defenses, Liu et al. [82] are able to introduce a defense callsed Non-Adversarial Backdoor (NAB), where they are able to outperform previous defenses in regards to less suffering on the clean accuracy. Backdoors are poisoned data which are used during training a machine learning model and later during inference are triggered by certain samples to cause misclassification. The benign data does not affect the model's accuracy, while the backdoor triggers the classifier's prediction a wrong label. It could be an interesting direction to adapt this approach for our method. We fine-tune the pre-trained classifier by utilizing NAB for adversarial examples and later fine-tune the visual prompt for hopefully achieving higher clean accuracy.

- **multiLID**: This detection method is the modification of the local intrinsic dimensionality investigated by Xingjun [91], which led to detection improvement as long as we train our detector for each adversarial attack separately. The transferability to other attacks is still expandable. Our attempts to train a detector by using samples from other attacks failed by the loss of the overall detection accuracy. This is still an open issue for fundamental research, e.g. one rejected paper from Liu et al. [79] suggests to a new building block for DNNs to treat each adversarial examples category differently.

- **Diffusion Model's Manifold**: Diffusion models (DMs) have gained notable attention due to their effective approximation of data distributions, leading to advanced generative outcomes. Adversarial attacks are known for deceiving classifiers and changes the class predictions. The question at hand is whether images subjected to adversarial attacks also fit within the learned manifold of the DM. Our experiments show that adversarial attacked images do not belong to the learned manifold. Since generative models relies on a huge amount of data, these data can be acquired either to use previous generative models to generate a dataset or crawl the data from the world wide web. We observe a potential risk when utilizing crawled images for training a DM that are vulnerable to attacks on the World Wide Web. Such attacks could impact the Diffusion model's manifold. Therefore, we direct to new methodologies to robustify the diffusion model's manifold against adversarial examples.

When we reflect our detection methods, we were training on a specific type of adversarial examples in supervised manners and our detector sorts out or neutralizes the input before reaching the classifier. This means that our detectors were mostly successful against the known threat.

In terms, of new unseen attacks, our detector methods suffer heavily by the accuracy. We encountered by testing the transferability of to other attack methods, but still does not lead to convincing results. These approaches heavily rely on supervised manners. Therefore, we think that larger detection models and more data would be needed to cover more cases. At the end, they might not be successful against unforeseen attack methods.

Recently, Nie et al. [102] introduced a new defense paradigm – "DiffPure" – that purifies images with a DM. Instead of detecting and sorting out malicious input, all input images are purified by the reverse process of diffusion models. DiffPure can handle many unforeseen attack methods and we think that could be one possible direction for future adversarial examples defenses. This method needs to find the trade-off between purification and keeping original image information. We also think that this research direction could be more robust against the transferability of different kind of current known attacks and could be an alternative to adversarial training. It remains unclear whether the whitebox attacks commonly used for adversarial training are the most effective for evaluating DiffPure's robustness. We think that new adversarial attack design needs to be investigated to elaborate on DiffPure's robustness. However, due to the nature of the underlying architecture of DMs, this method lacks of inference speed. The adversarial examples attack independence motivates for future work in this direction.

In this thesis, we have neglected a realistic scenario in the wild is to have patch-wise adversarial attacks instead of manipulating an image on a global viewpoint, i.e. [50], where only a quadratic area of an image is adversarial manipulated. A patch could be printed on a t-shirt and therefore a person wearing this t-shirt is invisible for a CCTV camera in a private area. It would be interesting to assess our multiLID on patch-wise attacked images, since we only have used global whitebox attacks so far. In addition, patch-wise attacks have been shown to be effective on other architectures beyond traditional CNNs, such as vision transformers (ViT) [39]. Due to the nature ViT the input will be patchified and a global attack will be destroyed, while CNNs always take the whole image as once as an input. On the other hand, the adversarial patch is not necessarily destroyed.

At this end, analyzing adversarial examples is complex, as it involves analyzing intricate defense algorithms and security properties. Additionally, the transferability property of adversarial examples enhances the effectiveness of weaker blackbox adversarial examples, further complicating the assessment of defense mechanisms. If a model is robust against one adversarial attack, it does not mean, it is robust against others. In recent years, research has revealed that a shift in defense paradigms, i.e. diffusion models, can enhance state-of-the-art defense mechanisms against known attack methods. This shift also prompts the development of new attack methods.

Bibliography

[1] Alfarra, M., Pérez, J. C., et al. Combating adversaries with anti-adversaries. In *AAAI* (2022).

[2] Amsaleg, L., Bailey, J., Barbe, A., Erfani, S. M., Furon, T., Houle, M. E., Radovanović, M., and Nguyen, X. V. High intrinsic dimensionality facilitates adversarial attack: Theoretical evidence. *IEEE Transactions on Information Forensics and Security 16* (2021), 854–865.

[3] Amsaleg, L., Chelly, O., Furon, T., Girard, S., Houle, M. E., Kawarabayashi, K.-i., and Nett, M. Estimating local intrinsic dimensionality. In *SIGKDD* (New York, NY, USA, 2015), Association for Computing Machinery, p. 29–38.

[4] Andriushchenko, M., Croce, F., Flammarion, N., and Hein, M. Square attack: a query-efficient black-box adversarial attack via random search. In *ECCV* (2020).

[5] Athalye, A., Carlini, N., et al. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *ICML* (2018).

[6] Athalye, A., Engstrom, L., Ilyas, A., and Kwok, K. Synthesizing robust adversarial examples. In *ICML* (2018), vol. 80, pp. 284–293.

[7] Aumüller, M., and Ceccarello, M. The role of local dimensionality measures in benchmarking nearest neighbor search. *Information Systems* (2021).

[8] Bahng, H., Jahanian, A., Sankaranarayanan, S., and Isola, P. Visual prompting: Modifying pixel space to adapt pre-trained models. *arXiv:2203.17274* (2022).

[9] Bhagoji, A., Cullina, D., and Mittal, P. Dimensionality reduction as a defense against evasion attacks on machine learning classifiers. *arXiv abs/1704.02654* (2017).

[10] Biggio, B., and Roli, F. Wild patterns: Ten years after the rise of adversarial machine learning. *Pattern Recognition 84* (2018), 317–331.

[11] Boopathy, A., et al. Proper network interpretability helps adversarial robustness in classification. In *ICML* (2020).

[12] Brown, T., Mann, B., et al. Language models are few-shot learners. *NeurIPS* (2020).

[13] Bulusu, S., Kailkhura, B., Li, B., Varshney, P., and Song, D. Anomalous instance detection in deep learning: A survey. *EEE Symposium on Security and Privacy* (3 2020).

[14] Carlini, N., et al. Towards evaluating the robustness of neural networks. In *IEEE Symposium on S&P* (2017).

[15] Carlini, N., and Wagner, D. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security* (2017), ACM, pp. 3–14.

[16] Carlini, N., and Wagner, D. Magnet and "efficient defenses against adversarial attacks" are not robust to adversarial examples, 2017.

[17] Carlini, N., and Wagner, D. A. Adversarial examples are not easily detected: Bypassing ten detection methods. *AISec* (2017).

[18] Carmon, Y., Raghunathan, A., Schmidt, L., Liang, P., and Duchi, J. C. Unlabeled data improves adversarial robustness. *NeurIPS* (2019).

[19] Chefer, H., Lang, O., Geva, M., Polosukhin, V., Shocher, A., Irani, M., Mosseri, I., and Wolf, L. The hidden language of diffusion models. *arXiv preprint arXiv:2306.00966* (2023).

[20] Chen, A., Lorenz, P., Yao, Y., Chen, P.-Y., and Liu, S. Visual prompting for adversarial robustness. In *ICASSP* (2023), IEEE, pp. 1–5.

[21] Chen, H., Dong, Y., Wang, Z., Yang, X., Duan, C., Su, H., and Zhu, J. Robust classification via a single diffusion model. *ICLR* (2024).

[22] Chen, J., Chen, H., Chen, K., Zhang, Y., Zou, Z., and Shi, Z. Diffusion models for imperceptible and transferable adversarial attack. *ICLR* (2023).

[23] Chen, P.-Y. Model reprogramming: Resource-efficient cross-domain machine learning. *arXiv:2202.10629* (2022).

[24] Chen, T., Liu, S., et al. Adversarial robustness: From self-supervised pre-training to fine-tuning. In *CVPR* (2020).

[25] Chen, Z., Li, Q., et al. Towards robust neural networks via close-loop control. *ICLR* (2021).

[26] Chrabaszcz, P., Loshchilov, I., and Hutter, F. A downsampled variant of imagenet as an alternative to the cifar datasets, 2017.

[27] Cohen, J., Rosenfeld, E., and Kolter, Z. Certified adversarial robustness via randomized smoothing. In *ICML* (2019), PMLR, pp. 1310–1320.

[28] Cooley, J., and Tukey, J. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation 19* (1965), 297–301.

[29] Couairon, G., Verbeek, J., Schwenk, H., and Cord, M. Iclr. *arXiv preprint arXiv:2210.11427* (2022).

[30] Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P., and Hein, M. Robustbench: a standardized adversarial robustness benchmark. *NeurIPS Datasets and Benchmarks* (2020).

[31] Croce, F., et al. Evaluating the adversarial robustness of adaptive test-time defenses. *ICML* (2022).

[32] Croce, F., and Hein, M. Minimally distorted adversarial examples with a fast adaptive boundary attack. In *ICML* (2020).

[33] Croce, F., and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML* (2020), PMLR.

[34] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *CVPR* (2009), Ieee, pp. 248–255.

[35] Dhariwal, P., and Nichol, A. Diffusion models beat gans on image synthesis. *NeurIPS 34* (2021), 8780–8794.

[36] Dong, Y., Fu, Q.-A., Yang, X., Pang, T., Su, H., Xiao, Z., and Zhu, J. Benchmarking adversarial robustness on image classification. In *CVPR* (2020), pp. 321–323.

[37] Dong, Y., Pang, T., Su, H., and Zhu, J. Evading defenses to transferable adversarial examples by mitigating attention shift. *CVPR* (2018).

[38] Dong, Y., Pang, T., Su, H., and Zhu, J. Evading defenses to transferable adversarial examples by translation-invariant attacks. In *CVPR* (2019), pp. 4312–4321.

[39] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR* (2021).

[40] Durall, R., Keuper, M., and Keuper, J. Watch your up-convolution: Cnn based generative deep neural networks are failing to reproduce spectral distributions. *CVPR* (2020), 7887–7896.

[41] Elsayed, G. F., Goodfellow, I., et al. Adversarial reprogramming of neural networks. *ICLR* (2018).

[42] Evtimov, I., Eykholt, K., Fernandes, E., Kohno, T., Li, B., Prakash, A., Rahmati, A., and Song, D. Robust physical-world attacks on deep learning models. *arXiv: Cryptography and Security* (2017).

[43] Fan, L., Liu, S., et al. When does contrastive learning preserve adversarial robustness from pretraining to finetuning? *NeurIPS* (2021).

[44] Feinman, R., Curtin, R. R., Shintre, S., and Gardner, A. B. Detecting adversarial samples from artifacts. *arXiv abs/1703.00410* (2017).

[45] Gong, Y., et al. Reverse engineering of imperceptible adversarial image perturbations. *ICLR* (2022).

[46] Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572* (2014).

[47] Gowal, S., Rebuffi, S.-A., Wiles, O., Stimberg, F., Calian, D. A., and Mann, T. A. Improving robustness using generated data. *NeurIPS 34* (2021), 4218–4233.

[48] Grosse, K., Manoharan, P., et al. On the (statistical) detection of adversarial examples. *arXiv:1702.06280* (2017).

[49] Gu, J., Jia, X., de Jorge, P., Yu, W., Liu, X., Ma, A., Xun, Y., Hu, A., Khakzar, A., Li, Z., et al. A survey on transferability of adversarial examples across deep neural networks. *arXiv preprint arXiv:2310.17626* (2023).

[50] Gu, J., Tresp, V., and Qin, Y. Are vision transformers robust to patch-wise perturbations? *ECCV* (2021).

[51] Gu, T., Dolan-Gavitt, B., et al. Badnets: Identifying vulnerabilities in the machine learning model supply chain. *arXiv:1708.06733* (2017).

[52] Harder, P., Pfreundt, F.-J., Keuper, M., and Keuper, J. Spectraldefense: Detecting adversarial attacks on cnns in the fourier domain. In *IJCNN* (2021).

[53] Hendrycks, D., and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. *ICLR* (2019).

[54] Hendrycks, D., and Gimpel, K. Early methods for detecting adversarial images. *ICLR* (2016).

[55] Ho, J., Jain, A., and Abbeel, P. Denoising diffusion probabilistic models. *NeurIPS 33* (2020), 6840–6851.

[56] Houle, M. E. Local intrinsic dimensionality i: An extreme-value-theoretic foundation for similarity applications. In *Similarity Search and Applications* (Cham, 2017), C. Beecks, F. Borutta, P. Kröger, and T. Seidl, Eds., Springer International Publishing, pp. 64–79.

[57] Houle, M. E. Local intrinsic dimensionality ii: Multivariate analysis and distributional support. In *Similarity Search and Applications* (Cham, 2017), C. Beecks, F. Borutta, P. Kröger, and T. Seidl, Eds., Springer International Publishing, pp. 80–95.

[58] Houle, M. E., Kashima, H., and Nett, M. Generalized expansion dimension. In *IEEE 12th International Conference on Data Mining Workshops* (2012), pp. 587–594.

[59] Hwang, D., Lee, E., and Rhee, W. Aid-purifier: A light auxiliary network for boosting adversarial defense. *Neurocomputing 541* (2023), 126251.

[60] Ilyas, A., Engstrom, L., Athalye, A., and Lin, J. Black-box adversarial attacks with limited queries and information. *ICML* (2018).

[61] Ilyas, A., Engstrom, L., and Madry, A. Prior convictions: Black-box adversarial attacks with bandits and priors. *ICLR* (2018).

[62] Jia, S., Ma, C., Yao, T., Yin, B., Ding, S., and Yang, X. Exploring frequency adversarial attacks for face forgery detection. In *CVPR* (2022), pp. 4103–4112.

[63] Jung, S., and Keuper, M. Spectral distribution aware image generation. In *AAAI* (2021).

[64] Kang, Q., et al. Stable neural ode with lyapunov-stable equilibrium points for defending against adversarial attacks. *NeurIPS* (2021).

[65] Karger, D. R., and Ruhl, M. Finding nearest neighbors in growth-restricted metrics. In *ACM Symposium on Theory of Computing* (2002).

[66] Korkmaz, E. Adversarially trained neural policies in the fourier domain. In *ICML* (2021).

[67] Krizhevsky, A., Hinton, G., et al. Learning multiple layers of features from tiny images. *arXiv* (2009).

[68] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *NeurIPS* (2012), F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., vol. 25, Curran Associates, Inc.

[69] Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *NeurIPS 25* (2012), 1097–1105.

[70] Kurakin, A., Goodfellow, I., et al. Adversarial examples in the physical world. *ICLR* (2016).

[71] Laidlaw, C., Singla, S., and Feizi, S. Perceptual adversarial robustness: Defense against unseen threat models. *arXiv preprint arXiv:2006.12655* (2020).

[72] Lakshminarayanan, B., Pritzel, A., and Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS* (2017).

[73] Lee, K., Lee, K., Lee, H., and Shin, J. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. *NeurIPS 31* (2018).

[74] Lee, M., and Kim, D. Robust evaluation of diffusion-based adversarial purification. *ICCV* (2023).

[75] Lee, S., Lee, G., Kim, H., Kim, J., and Uh, Y. Diffusion models with grouped latents for interpretable latent space. In *ICML 2023 Workshop on Structured Probabilistic Inference {\&} Generative Modeling* (2023).

[76] Li, X., and Li, F. Adversarial examples detection in deep networks with convolutional filter statistics. In *ICCV* (2017), pp. 5775–5783.

[77] Li, X. L., and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. *IJCNP-AACL* (2021).

[78] Lin, J., Njilla, L. L., and Xiong, K. Secure machine learning against adversarial samples at test time. *EURASIP Journal on Information Security 2022*, 1 (2022), 1.

[79] Liu, A., Tang, S., Liu, X., Chen, X., Huang, L., Tu, Z., Song, D., and Tao, D. Towards defending multiple adversarial perturbations via gated batch normalization, 2021.

[80] Liu, C., Dong, Y., Xiang, W., Yang, X., Su, H., Zhu, J., Chen, Y., He, Y., Xue, H., and Zheng, S. A comprehensive study on robustness of image classification models: Benchmarking and rethinking. *IJCV* (2023).

[81] Liu, J., Zhang, W., Zhang, Y., Hou, D., Liu, Y., Zha, H., and Yu, N. Detection based defense against adversarial examples from the steganalysis point of view. *CVPR* (2018), 4820–4829.

[82] Liu, M., Sangiovanni-Vincentelli, A., and Yue, X. Beating backdoor attack at its own game. In *CVPR* (2023).

[83] Liu, Z., Liu, Q., Liu, T., Wang, Y., and Wen, W. Feature distillation: Dnn-oriented jpeg compression against adversarial examples. *CVPR* (2019), 860–868.

[84] Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *ICCV* (December 2015).

[85] Lorenz, P., Harder, P., Straßel, D., Keuper, M., and Keuper, J. Detecting autoattack perturbations in the frequency domain. In *ICML Workshop on Adversarial Machine Learning* (2021).

[86] Lorenz, P., Keuper, M., and Keuper, J. Unfolding local growth rate estimates for (almost) perfect adversarial detection. *VisAPP* (2022).

[87] Lorenz, P., Strassel, D., Keuper, M., and Keuper, J. Is robustbench/autoattack a suitable benchmark for adversarial robustness? In *The AAAI Workshop on Adversarial Machine Learning and Beyond* (2022).

[88] Lu, P.-H., Chen, P.-Y., and Yu, C.-M. On the limitation of local intrinsic dimensionality for characterizing the subspaces of adversarial examples. *ICLRw* (2018).

[89] Lundberg, S. M., and Lee, S.-I. A unified approach to interpreting model predictions. In *NeurIPS* (2017), pp. 4765–4774.

[90] Ma, C., Wu, B., Xu, S., Fan, Y., Zhang, Y., Zhang, X., and Li, Z. Effective and robust detection of adversarial examples via benford-fourier coefficients. *Mach. Intell. Res. abs/2005.05552* (april 2022).

[91] Ma, X., Li, B., Wang, Y., Erfani, S. M., Wijewickrema, S., Schoenebeck, G., Song, D., Houle, M. E., and Bailey, J. Characterizing adversarial subspaces using local intrinsic dimensionality. *ICLR* (2018).

[92] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *ICLR* (2018).

[93] Madry, A., and Schmidt, L. A brief introduction to adversarial examples. `https://gradientscience.org/intro_adversarial`. Accessed: 2023-09-14.

[94] Mao, C., Chiquier, M., et al. Adversarial attacks are reversible with natural supervision. In *ICCV* (2021).

[95] Meng, D., and Chen, H. Magnet: a two-pronged defense against adversarial examples. *Conference on Computer and Communications Security* (2017).

[96] Metzen, J. H., Genewein, T., et al. On detecting adversarial perturbations. *ICLR* (2017).

[97] Mohapatra, J., Ko, C.-Y., et al. Rethinking randomized smoothing for adversarial robustness. *arXiv:2003.01249* (2020).

[98] Mokady, R., Hertz, A., Aberman, K., Pritch, Y., and Cohen-Or, D. Null-text inversion for editing real images using guided diffusion models. In *CVPR* (2023), pp. 6038–6047.

[99] Moosavi-Dezfooli, S.-M., Fawzi, A., and Frossard, P. Deepfool: a simple and accurate method to fool deep neural networks. In *CVPR* (2016), pp. 2574–2582.

[100] Neekhara, P., Hussain, S., et al. Cross-modal adversarial reprogramming. In *WACV* (2022).

[101] Nichol, A. Q., and Dhariwal, P. Improved denoising diffusion probabilistic models. In *ICML* (2021), PMLR, pp. 8162–8171.

[102] Nie, W., Guo, B., Huang, Y., Xiao, C., Vahdat, A., and Anandkumar, A. Diffusion models for adversarial purification. *ICML* (2022).

[103] Ovadia, Y., Fertig, E., Ren, J., Nado, Z., Sculley, D., Nowozin, S., Dillon, J. V., Lakshminarayanan, B., and Snoek, J. Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift. In *NeurIPS* (2019).

[104] PENG, S., XU, W., CORNELIUS, C., HULL, M., LI, K., DUGGAL, R., PHUTE, M., MARTIN, J., AND CHAU, D. H. Robust principles: Architectural design principles for adversarially robust cnns. *BMVC* (2023).

[105] PHUNG, A., AND STAMP, M. Universal adversarial perturbations and image spam classifiers, 2021.

[106] QUIÑONERO-CANDELA, J., SUGIYAMA, M., LAWRENCE, N. D., AND SCHWAIGHOFER, A. *Dataset shift in machine learning*. Mit Press, 2009.

[107] RADFORD, A., KIM, J. W., HALLACY, C., RAMESH, A., GOH, G., AGARWAL, S., SASTRY, G., ASKELL, A., MISHKIN, P., CLARK, J., ET AL. Learning transferable visual models from natural language supervision. In *ICML* (2021), PMLR, pp. 8748–8763.

[108] RAGHUNATHAN, A., STEINHARDT, J., ET AL. Certified defenses against adversarial examples. *ICLR* (2018).

[109] RAUBER, J., BRENDEL, W., AND BETHGE, M. Foolbox: A python toolbox to benchmark the robustness of machine learning models. *arXiv abs/1707.04131* (2018).

[110] REBUFFI, S.-A., GOWAL, S., CALIAN, D. A., STIMBERG, F., WILES, O., AND MANN, T. Fixing data augmentation to improve adversarial robustness, 2021.

[111] RICE, L., WONG, E., AND KOLTER, J. Z. Overfitting in adversarially robust deep learning. In *ICML* (2020), PMLR, pp. 8093–8104.

[112] RICKER, J., DAMM, S., HOLZ, T., AND FISCHER, A. Towards the detection of diffusion model deepfakes. *arXiv preprint arXiv:2210.14571* (2022).

[113] ROMBACH, R., BLATTMANN, A., LORENZ, D., ESSER, P., AND OMMER, B. High-resolution image synthesis with latent diffusion models. In *CVPR* (2022), pp. 10684–10695.

[114] RONY, J., HAFEMANN, L. G., OLIVEIRA, L. S., AYED, I. B., SABOURIN, R., AND GRANGER, E. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In *CVPR* (2019).

[115] RUSSAKOVSKY, O., DENG, J., SU, H., KRAUSE, J., SATHEESH, S., MA, S., HUANG, Z., KARPATHY, A., KHOSLA, A., BERNSTEIN, M., BERG, A. C., AND FEI-FEI, L. ImageNet Large Scale Visual Recognition Challenge. *IJCV 115*, 3 (2015), 211–252.

[116] SAIKIA, T., SCHMID, C., AND BROX, T. Improving robustness against common corruptions with frequency biased models. In *ICCV* (2021).

[117] Salman, H., Ilyas, A., et al. Unadversarial examples: Designing objects for robust vision. *NeurIPS* (2021).

[118] Salman, H., Sun, M., et al. Denoised smoothing: A provable defense for pretrained classifiers. *NeurIPS* (2020).

[119] Samangouei, P., Kabkab, M., and Chellappa, R. D.-G. Protecting classifiers against adversarial attacks using generative models. *arXiv preprint arXiv:1805.06605 1* (2018).

[120] Santhanam, G. K., and Grnarova, P. Defending against adversarial attacks by leveraging an entire gan. *arXiv preprint arXiv:1805.10652* (2018).

[121] Shafahi, A., Najibi, M., et al. Adversarial training for free! In *NeurIPS* (2019).

[122] Sharif, M., Bhagavatula, S., Bauer, L., and Reiter, M. K. A general framework for adversarial examples with objectives. *ACM Transactions on Privacy and Security 22*, 3 (jul 2019), 1–30.

[123] Shi, C., Holtz, C., et al. Online adversarial purification based on self-supervision. *ICLR* (2021).

[124] Simonyan, K., and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *CoRR abs/1409.1556* (2015).

[125] Singh, C., Murdoch, W. J., and Yu, B. Hierarchical interpretations for neural network predictions. In *ICLR* (2019).

[126] Sohl-Dickstein, J., Weiss, E., Maheswaranathan, N., and Ganguli, S. Deep unsupervised learning using nonequilibrium thermodynamics. In *ICML* (2015), PMLR, pp. 2256–2265.

[127] Song, J., Meng, C., and Ermon, S. Denoising diffusion implicit models. *ICLR* (2020).

[128] Song, Y., Kim, T., Nowozin, S., Ermon, S., and Kushman, N. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *ICLR* (2017).

[129] Sun, C., Zhang, Y., Chaoqun, W., Wang, Q., Li, Y., Liu, T., Han, B., and Tian, X. Towards lightweight black-box attack against deep neural networks. *NeurIPS 35* (2022), 19319–19331.

[130] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. *ICLR* (2013).

[131] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. Intriguing properties of neural networks. In *ICLR* (2014).

[132] Tang, S., et al. Robustart: Benchmarking robustness on architecture design and training techniques. *arXiv preprint arXiv:2109.05211* (2021).

[133] Thomas, S., Koleini, F., and Tabrizi, N. Dynamic defenses and the transferability of adversarial examples. In *IEEE 4th International Conference on Trust, Privacy and Security in Intelligent Systems, and Applications (TPS-ISA)* (2022), IEEE, pp. 276–284.

[134] Tramer, F., and Boneh, D. Adversarial training and robustness for multiple perturbations. *NeurIPS 32* (2019).

[135] Tramer, F., Carlini, N., Brendel, W., and Madry, A. On adaptive attacks to adversarial example defenses. *NeurIPS* (2020).

[136] Tsai, Y.-Y., et al. Transfer learning without knowing: Reprogramming black-box machine learning models with scarce data and limited resources. *ICML* (2020).

[137] Tsuzuku, Y., and Sato, I. On the structural sensitivity of deep convolutional networks to the directions of fourier basis functions. *CVPR* (2019), 51–60.

[138] Wang, J. Adversarial examples in physical world. In *IJCAI* (2021), pp. 4925–4926.

[139] Wang, R., Xu, K., et al. On fast adversarial robustness adaptation in model-agnostic meta-learning. In *ICLR* (2021).

[140] Wang, X., and He, K. Enhancing the transferability of adversarial attacks through variance tuning. In *CVPR* (2021), pp. 1924–1933.

[141] Wang, Z., Bao, J., Zhou, W., Wang, W., Hu, H., Chen, H., and Li, H. Dire for diffusion-generated image detection. *ICCV* (2023).

[142] Wang, Z., Pang, T., Du, C., Lin, M., Liu, W., and Yan, S. Better diffusion models further improve adversarial training. *ICML* (2023).

[143] Wójcik, B., Morawiecki, P., et al. Adversarial examples detection and analysis with layer-wise autoencoders. *IEEE International Conference on Tools with Artifical Intelligence* (2020).

[144] WONG, E., AND KOLTER, Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML* (2018), PMLR, pp. 5286–5295.

[145] WONG, E., RICE, L., AND KOLTER, J. Z. Fast is better than free: Revisiting adversarial training. In *ICLR* (2020).

[146] WU, B., CHEN, J., CAI, D., HE, X., AND GU, Q. Do wider neural networks really help adversarial robustness? In *NeurIPS* (2021), A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds.

[147] WU, B., PAN, H., SHEN, L., GU, J., ZHAO, S., LI, Z., CAI, D., HE, X., AND LIU, W. Attacking adversarial attacks as a defense. *arXiv preprint arXiv:2106.04938* (2021).

[148] XIAO, C., CHEN, Z., JIN, K., WANG, J., NIE, W., LIU, M., ANANDKUMAR, A., LI, B., AND SONG, D. Densepure: Understanding diffusion models towards adversarial robustness. *arXiv preprint arXiv:2211.00322* (2022).

[149] XICHEN, A. Pytorch playground, 2017.

[150] XIE, C., WU, Y., ET AL. Feature denoising for improving adversarial robustness. In *CVPR* (2019).

[151] XU, K., XIAO, Y., ZHENG, Z., CAI, K., AND NEVATIA, R. Patchzero: Defending against adversarial patch attacks by detecting and zeroing the patch. In *WACV* (2023), pp. 4632–4641.

[152] YANG, C.-H. H., TSAI, Y.-Y., ET AL. Voice2series: Reprogramming acoustic models for time series classification. In *ICML* (2021), PMLR.

[153] YANG, K., ZHOU, T., ZHANG, Y., TIAN, X., AND TAO, D. Class-disentanglement and applications in adversarial detection and defense. *NeurIPS 34* (2021), 16051–16063.

[154] YANG, L., ZHANG, Z., SONG, Y., HONG, S., XU, R., ZHAO, Y., ZHANG, W., CUI, B., AND YANG, M.-H. Diffusion models: A comprehensive survey of methods and applications. *ACM Computing Surveys 56*, 4 (2023), 1–39.

[155] YANG, P., ET AL. Ml-loo: Detecting adversarial examples with feature attribution. *AAAI* (2019).

[156] YANG, X., LIN, J., ZHANG, H., YANG, X., AND ZHAO, P. Improving the transferability of adversarial examples via direction tuning. *INS* (2023).

[157] YE, S., XU, K., ET AL. Adversarial robustness vs model compression, or both? *ICCV* (2019).

[158] Yin, D., Lopes, R. G., Shlens, J., Cubuk, E. D., and Gilmer, J. A fourier perspective on model robustness in computer vision. *NeurIPS* (2019).

[159] Yoon, J., Hwang, S. J., et al. Adversarial purification with score-based generative models. In *ICML* (2021), PMLR.

[160] Zagoruyko, S., and Komodakis, N. Wide residual networks. In *BMVC* (2017).

[161] Zhang, D., Zhang, T., et al. You only propagate once: Accelerating adversarial training via maximal principle. *NeurIPS* (2019).

[162] Zhang, G., Zhang, Y., et al. Fairness reprogramming. *NeurIPS* (2022).

[163] Zhang, H., Yu, Y., et al. Theoretically principled trade-off between robustness and accuracy. *ICML* (2019).

[164] Zhang, J., Chen, Z., Zhang, H., Xiao, C., and Li, B. {DiffSmooth}: Certifiably robust learning via diffusion models and local smoothing. In *32nd USENIX Security Symposium* (2023), pp. 4787–4804.

[165] Zhang, Q., Li, X., Chen, Y., Song, J., Gao, L., He, Y., and Xue, H. Beyond imagenet attack: Towards crafting adversarial examples for black-box domains. *ICLR* (2022).

[166] Zhang, Y., Chen, X., Jia, J., Liu, S., and Ding, K. Text-visual prompting for efficient 2d temporal video grounding. *CVPR* (2023).

[167] Zhang, Y., Yao, Y., et al. How to robustify black-box ml models? a zeroth-order optimization perspective. *ICLR* (2022).

[168] Zheng, Y., Feng, X., Xia, Z., Jiang, X., Demontis, A., Pintor, M., Biggio, B., and Roli, F. Clawsat: Towards both robust and accurate code models. *Inf. Sci. 632* (2021), 130–143.

[169] Zheng, Y., Feng, X., Xia, Z., Jiang, X., Demontis, A., Pintor, M., Biggio, B., and Roli, F. Why adversarial reprogramming works, when it fails, and how to tell the difference. *Information Sciences 632* (2023), 130–143.