Universiteit Leiden

Master Computer Science

Unsupervised adversarial attack detection via student–teacher-based anomaly detection

Name:	Pablo Rojas
Student ID:	s3721000
Date:	[27/08/2025]
Specialisation:	Artificial Intelligence
1st supervisor:	Jan van Rijn
2nd supervisor:	Holger Hoos

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Detecting adversarial attacks remains challenging: most detectors either collapse under white-box scrutiny or fail to generalize across attack types. In order to address these matters, we revisit the uninformed students anomaly detection framework–originally devised for industrial defect inspection–and show that, with two simple modifications, it becomes a powerful detector of adversarial examples. First, we replace the distilled teacher by a pre-trained ResNet-18, eliminating the extra distillation stage while increasing architectural mismatch; second, we train an ensemble of students on clean data only, so the ℓ_2 imitation discrepancy between teacher and students serves as an attack-agnostic anomaly score. In an extensive experimental evaluation across four well-known image datasets, our detector was able to outperform other state-of-the-art methods in half of the explored datasets, and it performs similarly at detecting large magnitude attacks on the other datasets. Furthermore, when the attacker also targets the detector model with white-box knowledge, our detector was the only one to remain robust in two out of the four dataset, whereas in the other two datasets all detectors fail. These results demonstrate a simple, label-free route to harden vision models against white-box adversaries.

Contents

Chapter 1

Introduction

Adversarial attacks pose an ongoing and critical challenge to the robustness of neural networks. Such attacks involve carefully crafted perturbations–often indistinguishable by humans–that manipulate a model into making incorrect predictions. Despite substantial progress, reliably detecting adversarial examples remains elusive, primarily due to adversaries' adaptability and the limitations inherent in current detection strategies.

Numerous detection mechanisms have been proposed, varying from supervised classifiers trained explicitly on adversarial examples [19, 15, 33, 21, 13], to unsupervised methods aiming to capture anomalies in data distribution [20, 12, 15, 24, 30, 28]. However, supervised detectors frequently fail to generalize beyond the attack types they are trained on and are therefore inherently vulnerable to novel attacks [9]. Conversely, unsupervised detectors, though theoretically attractive due to their attack-agnostic nature, often exhibit poor resilience when subjected to perfect knowledge adversaries [9], that is, adversaries possessing complete knowledge not only of the classifier but also of the detection strategy. In this regard, the goal is to design a robust system where an attacker is only able to successfully attack either the target classifier, or the detector, but not both at the same time. This remains, to date, a requirement that has proven elusive, and to address this challenge, we assess our detector within this context and show significant progress over existing solutions.

In recent years, anomaly detection has emerged prominently in industrial and manufacturing applications, effectively distinguishing defective products from normal samples through models trained exclusively on non-defective (normal) data. Notably, the *uninformed students* framework by Bergmann et al. [7] demonstrated exceptional performance in such industrial contexts by leveraging a teacher-student paradigm where student networks are trained to imitate a teacher's feature representations from normal data, with discrepancies highlighting anomalies.

Motivated by this promising paradigm, we propose adapting the uninformed students framework specifically for adversarial attack detection. Our approach introduces two critical innovations. First, we directly employ a pretrained ResNet-18 as the teacher network, thus skipping the additional distillation step originally proposed to reduce the computational cost. This modification enhances robustness by ensuring an architectural mismatch between teacher and students, preventing students from mimicking anomalous (adversarial) features inadvertently. Second, we train an ensemble of student models exclusively on clean images,

using the resulting imitation discrepancies as an unsupervised, universally-applicable anomaly score.

In a nutshell, the resulting method will take as input an image to be classified, and returns an anomaly score. Here, a low anomaly score indicates that the image is likely a natural image, and a high anomaly score indicates that the image is likely an adversarial attack. Generally, if the method is well-behaved, adversarial attacks with a low magnitude will result in a relatively low anomaly score, whereas adversarial attacks with a high magnitude will result in a high anomaly score. As a result, the method is expected to perform well on detecting adversarial attacks with a high magnitude. While it is arguably equally important to detect adversarial attacks with a low magnitude, it is important to note that in this regime many other complementary methods exists, such as robust training methods [14, 31, 48] and neural network verification [23, 47, 49].

Summarizing, our contributions are the following:

- A new adversarial attack detection method, inspired by the uninformed students framework by Bergmann et al. [7]. This is expected to perform well on detecting adversarial attacks with a high magnitude.
- We extensively evaluate the performance of this method across various attack magnitudes on 4 common image datasets. We compare the method against 3 state-of-the-art methods.
- Following the evaluation protocol of Carlini and Wagner [9], we perform a *perfect knowledge attack*, in which the attacker does not only have access to all information of the classifier, but also to the detection method. We compare the performance of the proposed method in this setting against the same state-of-the-art methods.

Chapter 2

Background

This chapter surveys the concepts we build on. We review adversarial robustness and attack methods, outline major defense families, and position adversarial detection within that landscape, connecting it to unsupervised anomaly detection.

2.1 Foundations of adversarial robustness

The notion of neural network robustness has evolved from simple tolerance to sensor noise toward resilience against meticulously engineered adversarial examples. A pivotal milestone was the Fast Gradient Sign Method (FGSM) [14], which revealed the vulnerability of modern deep models and started a surge of research into certified bounds, adversarial training, and robustness metrics. Today, robustness is often used to refer to a model's ability to maintain correct predictions across both natural and worst-case input variations, providing a complementary lens to conventional generalization.

2.1.1 Adversarial attacks

Adversarial attacks, illustrated in Figure 2.1, are often described as finding a perturbation δ that can be applied to an input image x to generate an adversarial example $x' = x + \delta$ that can lead the network defined by its parameters θ, and its forward function $f(x, \theta)$ to predict an output different than the ground truth label y, or even a particular target label y'. This latter scenario is often referred to as a targeted attack, whereas the first one is usually called an untargeted attack. That perturbation δ is often limited to a maximum value ϵ under a distance metric ℓ_p-norm that measures how large the perturbation is. We use the ℓ_p-norm definition provided by Carlini and Wagner [10], where for a given δ:

$$\ell_p = || \ \delta \ ||_p = \sqrt[p]{\sum_{\forall \delta_i \in \delta} | \ \delta_i \ |^p} \tag{2.1}$$

Here, i indexes each individual coordinate of δ (a pixel or pixel-channel entry), and p specifies how these per-coordinate magnitudes are aggregated into a single value. With the constraint $\|\delta\|_p \leq \epsilon$, common choices of p yield:

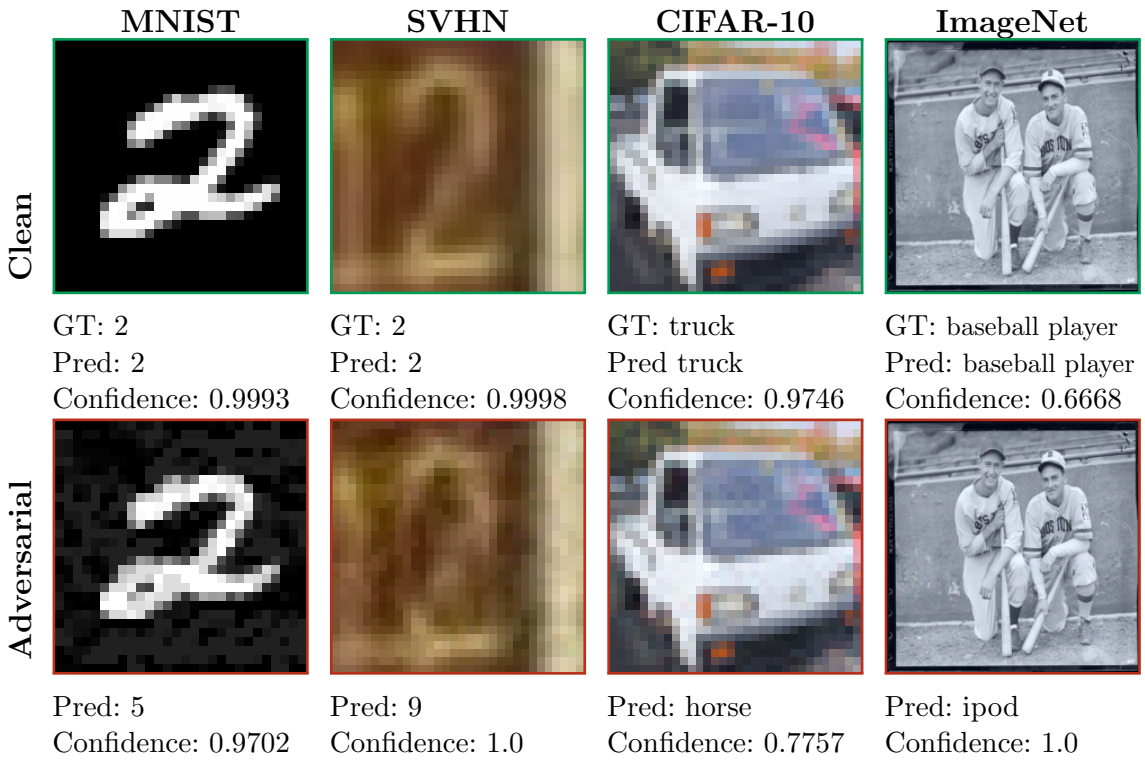


Figure 2.1: Clean versus adversarial examples across datasets. Rows show the original and perturbed inputs; column headers name the dataset. Confidence values are soft-max probabilities. Borders: green = correct, red = mis-classified.

- ℓ_0: Number of modified coordinates, $\sum_i [\delta_i \neq 0]$ (how many pixels are changed);
- ℓ_2: Euclidean distance, $\sqrt{\sum_i \delta_i^2}$ (many small changes can sum to a small overall shift);
- ℓ_∞: Maximum absolute change, $\max_i |\delta_i|$ (distance of the most perturbed pixel).

Then, within those bounds different attacks can be defined, based on how the δ is found. For instance, using FSGM, a one-step attack where we obtain the perturbation by maximizing the classifier's loss $\mathcal{L}$:

Then, within those bounds, different attacks can be defined based on how the perturbation δ is found. For example, in FGSM, the goal is to obtain the perturbation that maximizes the classifier's loss $\mathcal{L}$ under the ℓ_p constraint. Denoting this optimal perturbation by δ^*:

$$\delta^* = \arg\max_{\|\delta\|_p \leq \epsilon} \mathcal{L}\big(f(x+\delta, \theta), y\big), \tag{2.2}$$

This would correspond to the variant used for an untargeted attack, the type of attack that only attempts to force a misclassification. On the other hand, a targeted attack attempts to force the network to predict a particular class y', which for the FGSM can be achieved via the following:

$$\delta^* = \arg\max_{\|\delta\|_p \leq \epsilon} \mathcal{L}\big(f(x+\delta,\theta), y\big) - \mathcal{L}\big(f(x+\delta,\theta), y'\big). \tag{2.3}$$

In essence, this approach searches for the point which has the minimum loss in the ϵ-neighbor ball via a single step. The Basic Iterative Method (BIM), proposed by Kurakin et al. [26], is an evolution of this concept, which employs several of these steps in an iterative search to craft a more effective attack. In order to make this attack work, two key modifications have to be made: First, a step size η has to be introduced to control the aggressiveness of the search. Then, a *Clip* function has to be introduced to ensure the δ remains within the defined bounds:

$$\delta_{t+1} = Clip_{\delta,\epsilon}(\delta_t + \eta \cdot sign(\nabla \mathcal{L}(f(x+\delta_t,\theta), y))) \tag{2.4}$$

Here the $Clip_{\delta,\epsilon}$ function denotes the projection of its argument to the ϵ-neighbor ball $\mathscr{B}_\varepsilon(0) = \big\{\, \delta \in \mathbb{R}^d : \|\delta\|_p \leq \varepsilon \,\big\}$, and it is used after every step. When in addition, the δ is randomly initialized, as proposed in Madry et al. [31], this algorithm is referred to as Projected Gradient Descent (PGD), which one of the strongest adversarial attacks in the literature.

2.1.2 Methods against adversarial attacks

As a counterpart to adversarial attacks, researchers have been coming up with different defensive strategies to prevent or mitigate the effect of such attacks. In this regard, a large variety of defense mechanisms have been proposed.

For instance, some works have focused on addressing the existence of these attacks by incorporating various techniques during the training phase of a neural network that enhance their robustness during inference. Notable examples of this are adversarial training techniques [14, 31, 48], that expose the network to adversarial examples during the training phase, which has been proven to significantly enhance the robustness of the trained neural network. This type of technique usually comes with the trade-off of slightly reduced accuracy on natural images, as well as longer training time.

A complementary line of work, certified defenses [37, 11, 47], aims to provide provable guarantees. Certified defenses integrate a mathematical bound (e.g., interval bound propagation, linear relaxation, or randomized smoothing) into the loss function. The resulting model comes with a certificate that no perturbation within a prescribed ℓ_p ball of radius ε can change the predicted label. These guarantees come at the price of increased computational cost (during training, inference, or both) and often limit model complexity.

Another line of work is neural network verification, which applies formal methods to determine whether a fixed, trained model is provably robust for a given input (or region) and perturbation budget. Unlike certified defenses, which integrate robustness guarantees directly into training, verification is typically applied post hoc: solvers either prove that no adversarial example exists within an ℓ_p ball or return a concrete counterexample. Early exact methods such as ReLUplex [23] reason over ReLU activations to provide sound, complete guarantees at the cost of scalability, while later relaxations accelerate verification by over-approximating nonlinear layers and tightening bounds layer by layer [49]. In practice, these

techniques offer strong, instance-level assurances but still face trade-offs between tightness and runtime, making them most suitable for auditing safety-critical inputs or as a complement to training-time defenses.

The last approach, and the one where our method falls in, is the adversarial attack detectors. This type of defensive mechanism aims to identify which images have been adversarially modified. Usually this comes at the cost of some extra computation, as well as accepting a certain false positive rate on natural images, but offers the advantage of identifying that an attack has happened, in contrast to remaining robust, but unaware of the attack. Another significant difference with regards to the other adversarial defenses is that they usually are more effective against larger attacks. While these are usually not recognized as the most concerning attacks, in many cases they can only be addressed via an adversarial attack detector, placing this type of techniques in a unique place.

2.1.3 Threat model

When designing an adversarial defense, it is very important to consider the threat model against which it is intended to defend. The threat model, in essence, is the adversary's knowledge of the system it targets. Usually this is focused on the target classifier and framed in terms of white box (that is, full knowledge of the classifier, including training data, architecture weights...) or black box (no direct knowledge, apart from what can be inferred from a limited number of queries), but in practice there is a full spectrum of possibilities, including intermediate cases between those two (gray box) where partial information is available. In addition, when designing a defensive strategy, a new class of attacks must also be considered based on the knowledge they possess about the defensive mechanism itself. In the field of adversarial attack detection, usually one of two threat models is used:

Zero detector knowledge attack The adversary possesses white-box access to the classifier (fully aware of its architecture, parameters, and training procedure) but has no knowledge about the detection mechanism. FGSM, PGD or BIM, in their default configurations, are typical examples of *zero detector knowledge attacks*.

Perfect knowledge attack The adversary has complete white-box access to both the classifier and the detection system, including all parameters, architectures, and detection strategies employed. For this we implement a modified version of the PGD attack, following the guidelines described in Carlini and Wagner [9], and following the design implemented in Wang et al. [46]:

- *Detection-aware PGD*: A modified version of the PGD attack, under the *perfect knowledge attack* scenario.

  Let $f(x, \theta_c) : \mathbb{R}^d \to \mathbb{R}^C$ be the classifier and $\mathrm{AS}(x)$ the detector's anomaly score. A joint white-box adversary minimizes

$$\mathcal{L}_{wb}(x) = \underbrace{\mathcal{L}\big(f(x,\theta), y'\big)}_{\mathcal{L}_{\text{cls}}} - \alpha \cdot \underbrace{\mathrm{AS}(x)}_{\mathcal{L}_{\text{det}}}, \qquad \alpha > 0, \tag{2.5}$$

with the constraint $\|\delta\|_p \leq \varepsilon$, and a trade-off parameter α is used to control how much the attack is focused on the classifier or the detector. Then, at each PGD step we update with the gradient direction

$$\delta_{t+1} = Clip_{\delta,\epsilon}(\delta_t + \eta \cdot sign(\nabla \mathcal{L}_{wb}(x)) \tag{2.6}$$

Where $Clip_{\delta,\epsilon}$ denotes the projection that clips any updated point back onto the ℓ_p ball $\mathscr{B}_\varepsilon(0) = \{ \delta \in \mathbb{R}^d : \|\delta\|_p \leq \varepsilon \}$, ensuring the perturbation budget is never exceeded.

There exist additional attack scenarios beyond those explicitly covered here, such as situations where the attacker does not possess complete knowledge of the classifier (partial knowledge) [38], or cases where the attacker has only limited insights into the detection system, such as knowledge restricted to either the detection strategy, the model architecture, and/or the training data. For instance, Meng and Chen [32], define a scenario where many identical detectors are trained under different random seeds, then at inference it is expected that one of this detectors will be used, but that the attacker is not aware of which. Although these alternative scenarios may be more realistic than perfect-knowledge attacks, the common practice is to (i) assume full white-box knowledge of the classifier to make the setting maximally challenging, and (ii) complement this with evaluations at the two extremes of detector knowledge (no knowledge and perfect knowledge) to cover the full spectrum of possible adversarial conditions.

2.2 The landscape of adversarial attack detectors

Adversarial detection methods might be broadly decomposed into three families: auxiliary classifiers that complement a primary model (either via a dedicated "detector" head, or via a separate classifier) trained to distinguish clean from perturbed inputs [19, 15, 33, 21, 13]; statistical-fingerprinting techniques that monitor shifts in feature-space or in the images themselves [20, 12, 15, 24, 30, 28]; and input-transformation or consistency-based schemes that apply pre-processing and reject examples whose predicted labels vary excessively under these transformations [29, 51]. For a more extensive look into adversarial attack detectors, we refer the reader to the survey by Aldahdooh et al. [1].

Despite their apparent diversity, these approaches commonly have some fundamental shortcomings [9]: They either (i) are trained in a supervised manner for a given type of attack, which in turn means that they will not be reliable at detecting other types of attacks, or (ii) are only effective on a given dataset, or (iii) rely on some statistic that, when known, is easy to incorporate into an undetectable attack. Following up on that, most detectors do not test a setting where both the classifier and the detector (and all their settings) are known and exploited by the attacker.

Park and Kang [36] propose a similar framework to we will propose, using a student-teacher setting, but with an approach closer to the feature pyramid matching method proposed by Wang et al. [45]. The main difference is that they implement it using supervised learning, as they train their method by simultaneously minimizing the difference between the teacher and the student both when using natural images, and when the teacher uses a natural image and the student an FGSM adversarial example. This training technique explicitly

maximizes the learned anomaly score when provided with an adversarial image for the attack used during training, meaning it is clearly a supervised method. While this is an interesting and powerful approach, in our work we focus on unsupervised detectors.

In the following, we provide a brief overview of several representative detection methods from the literature that will serve as baselines in our experiments. While these approaches differ in their underlying principles, they illustrate the range of strategies explored for adversarial detection and highlight the challenges our proposed unsupervised method aims to address.

2.2.1 ACGAN

In their work, Wang et al. [46] exploit a class-conditional GAN scheme to train a discriminator model that can later on be used as an adversarial attack detector. In more detail, a GAN network usually consists of:

- A generator network with parameters θ_G and forward function $f_G(z, \theta_G)$, that transforms a uniform random vector z into a synthetic image $x_g = f(z, \theta_G)$.
- A discriminator network with parameters θ_D and forward function $f_D(x, \theta_D)$ that takes the input image x and outputs the predicted probability that x comes from the "real" training set of images.

Then, during training, the generator and discriminator are optimized in an adversarial manner: the generator aims to produce samples that are indistinguishable from real data, while the discriminator seeks to correctly classify inputs as real or generated. This is typically formulated as a min-max game, where the discriminator maximizes the probability of correctly labeling real and generated samples, and the generator minimizes the probability of the discriminator correctly identifying its outputs. Over successive iterations, this competition drives the generator to produce increasingly realistic images, while the discriminator learns more refined decision boundaries.

In their work they use the Auxiliary Classifier GAN variant proposed by Odena et al. [35], which adds a classification head to the discriminator, that is trained to classify the images regardless if they are generated by $f(z, \theta_G)$, or from the training set. Then, instead of a single value, the discriminator will provide two different outputs:

- A probability $f_D(x, \theta_D)$ that the input image is from the training set (or a natural image for this case). Note how this remains unchanged from the usual GAN model.
- The softmax prediction probabilities for each class $f_C(x, \theta_D)$, that they then use as $f_C(y \mid x, \theta_D)$: the posterior probability that the image x is from class y, for every class.

Then, in their work they test different ways of aggregating the results from the discriminator and the generator to detect adversarial attacks, but we will focus on the one that consistently delivered the best results across their testing, the single discriminator test statistic, which combines the two outputs of the discriminator network as:

$$S_D(x, y) = \log f_D(x, \theta_D) + \log f_C(y \mid x, \theta_D) \tag{2.7}$$

The intuition behind this statistic is that $f_D(x, \theta_D)$ will produce small values for natural images (training set), but larger ones for adversarial images. Although not trained explicitly for this, by using the generator images to train the model to distinguish which images do not belong to the training set, they argue that the model will learn to distinguish adversarial images as well. Then, having trained $f_C(y \mid x, \theta_D)$ to work both when using natural images or generated images, the expectation is that the classifier will work also even when presented with adversarial images, a reminiscent trait from a robust training technique. Then, for natural images the predicted class (by the classifier) y will be expected to be correct and aligned with $f_C(y \mid x, \theta_D)$, meaning it will produce a larger value, but if x has been attacked, y will likely be incorrect and misaligned with $f_C(y \mid x, \theta_D)$, and so the value will be lower. Given how both values are expected to range from $[0, 1]$, with 0 being low, and 1 being the larges value, when we use the logarithm of both outputs, and sum them, when working correctly, natural images should render values close to 0 and adversarial images should render large negative numbers.

For this model we use the implementation provided with their paper [1], with the hyperparameters described in their paper, with some minor modifications to adapt it to our benchmark. Namely, they upscale mnist to 32x32, which we avoid and instead use the 28x28 original image size. In addition, we also train it for SVHN using the same hyperparameters as in CIFAR-10.

2.2.2 LID

Proposed by [30], this detector characterizes the dimensional properties of adversarial regions, using Local Intrinsic Dimensionality (LID) [22] to construct an adversarial attack detector.

In practice, the true LID is unknown and must be estimated from data. A common approach is to use the distances to the k nearest neighbors of a point to approximate the local distance distribution. Ma et al. adopt the *Maximum Likelihood Estimator* (MLE) [2], which provides an efficient way to compute $\widehat{\text{LID}}$ from these neighbor distances:

$$\widehat{\text{LID}}(x) = -\left(\frac{1}{k} \sum_{i=1}^{k} \log \frac{r_i(x)}{r_k(x)} \right)^{-1}, \tag{2.8}$$

where $r_i(x)$ is the distance from x to its i-th nearest neighbor within a sample from P, and $r_k(x)$ is the largest of these k neighbor distances.

Ma et al. observe that adversarial perturbations move a sample x off its original low-dimensional data manifold into a region (the *adversarial subspace*) of much higher intrinsic dimensionality. When $\widehat{\text{LID}}$ is estimated in this new neighborhood, the value for an adversarial example x' tends to be significantly higher than that of its clean counterpart x. To exploit this, they compute $\widehat{\text{LID}}$ for each example at multiple layers of a pre-trained DNN, using minibatch-based k-nearest neighbor estimation in the activation space. The resulting LID values serve as features to train a linear regression model that distinguishes adversarial from natural images. This last part means that although the statistic itself is unsupervised, the method as a whole is not.

[1] https://github.com/wanghangpsu/acgan-ada

We used the implementation provided by Lee et al. [28],[2] which used a linear regressor to ensemble the LID statistic across layers.

2.2.3 Mahalanobis

Lee et al. [28] designed a detector that models the features of each layer of a neural network and then uses the mahalanobis distance to quantify how much it deviates. To do so, they first compute the empirical class mean $\hat{\mu}_c$ and covariance $\hat{\Sigma}$ of training samples $\{(\mathbf{x}_,1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ for each layer ℓ:

$$\hat{\mu}_{c,\ell} = \frac{1}{N_c} \sum_{i:y_i=c} f(\mathbf{x}_i, \theta_\ell), \quad \hat{\Sigma}_\ell = \frac{1}{N} \sum_c \sum_{i:y_i=c} \left(f(\mathbf{x}_i, \theta_\ell) - \hat{\mu}_{c,\ell}\right) \left(f(\mathbf{x}_,\theta_\ell) - \hat{\mu}_{c,\ell}\right)^\top, \tag{2.9}$$

where N_c is the number of training samples with label c. This is equivalent to fitting the class-conditional Gaussian distributions with a tied covariance to training samples under the maximum likelihood estimator.

Then, the mahalanobis distance can be obtained as

$$M_\ell(\mathbf{x}) = \max_c \; -\left(f(\mathbf{x}, \theta_\ell) - \hat{\mu}_{c,\ell}\right)^\top \hat{\Sigma}_\ell^{-1} \left(f(\mathbf{x}, \theta_\ell) - \hat{\mu}_{c,\ell}\right). \tag{2.10}$$

Then, similar to Ma et al. [30] they aggregate the score across multiple layers by training a linear regression model with the natural (and noisy) images as negative (0) samples and the FGSM adversarial samples as positive (1).

We used a wrapper around the code from the original repository to implement this method.[2] For the hyperparameters we use the ones that were found to be optimal in their testing.

2.3 Anomaly detection

Anomaly detection is a well-know deep learning task, widely used in many applications in the manufacturing and automation industry. It is formulated as the binary classification problem of detecting whether a given sample is defective or not. This task usually involves visually inspecting each manufactured sample to search for defects originating from the production process. The main difficulty in solving such task lies in the scarcity of defective samples [6, 8, 50, 44, 18], which usually results in class imbalance that makes it an unfeasible task for traditional classifiers. In such scenario, an anomaly detection model is used in an unsupervised setting to learn exclusively the 'good' samples during training. Then, at inference it will quantify the difference between a new unseen sample and the defect-free images seen during training via an anomaly score. Such anomaly score can then be used to determine whether a sample is defective or not, under the assumption that 'good' samples will have a lower score, and defective samples a higher score. Multiple approaches have been proposed to perform this task, such as approaches based on auto-encoders [5, 43], generative adversarial networks [42, 41] or a student-teacher setting [7, 45].

[2] `https://github.com/pokaxpoka/deep_Mahalanobis_detector`

Within the context of neural network robustness, one may relate the task of detecting anomalies (or defects in industrial parts) with that of detecting adversarial attacks. At its core both tasks aim to identify deviations from the training image distributions, with some key conceptual differences: In anomaly detection these deviations are on how the sample looks (e.g. scratches, dents...) and so are usually more localized and visible. In contrast adversarial attacks show more subtle differences in the images, are not necessarily localized (in particular for ℓ_∞-type of attacks), but show larger differences in the feature maps of the target classifier by design. Based on this, it may arise the question of how this unsupervised framework can be exploited to detect adversarial examples. In this regard, there have already been several works applying the principles of anomaly detection to adversarial example detection: For example, MagNet [32] employs an auto-encoder–based adversarial detector, following principles similar to those used in earlier anomaly detection methods. Similarly, Wang et al. [46] implement an approach based on a Generative Adversarial Network to detect the adversarial attacks.

Chapter 3

Methods

This chapter describes our detector. We start from the uninformed-students framework, introduce architectural changes tailored to adversarial settings, define the training objective and anomaly score, and state the desired detection properties.

3.1 Uninformed students

For our adversarial attack detector we start from the student-teacher-based anomaly detection framework proposed by Bergmann et al. [7], uninformed students. The key idea of this method is to train an ensemble of student networks imitate a teacher network, which in their method can be obtained via a knowledge distillation step from a classifier network trained in a large dataset (ImageNet). This training (Figure 3.1) is achieved by optimizing the student networks to minimize an ℓ_2-loss over the feature maps of the teacher's output feature map on the training images, which critically, consists only of anomaly-free images.

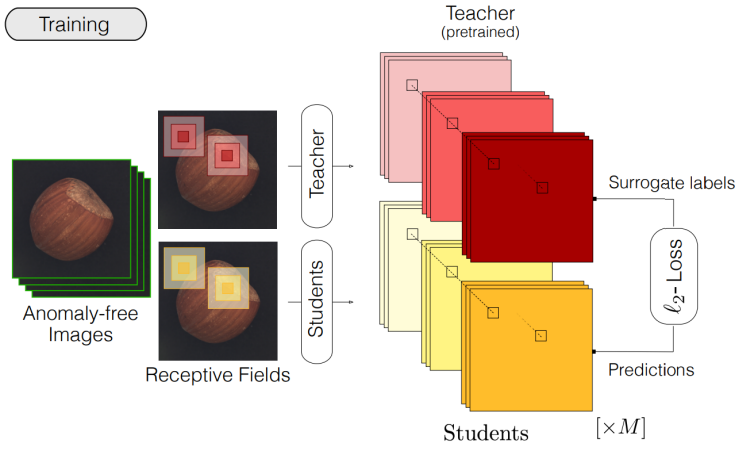


Figure 3.1: Diagram of the training of the student-teacher network, extracted from Bergmann et al. [7]. During training patches of the image are extracted and forwarded to the teacher network. After that, the ensemble of students will be trained to regress the output of the teacher via an ℓ_2 loss.

Then, as exemplified in Figure 3.2, during inference deviations from the training distribution are identified (the so called anomalies) as an increase in the student teacher imitation discrepancy, measured via the regression error and predictive uncertainty. Let $f(x_{r,c}, \theta_T)$ be the prediction made by the teacher (T) at a given pixel coordinates r, c, and $f(x_{r,c}, \theta_{S_n})$ the prediction of the n-th student (S_n), the regression error $e(r, c)$ and the predictive uncertainty $v(r, c)$ are computed as:

$$e(r,c) \;=\; \frac{1}{M}\sum_{n=1}^{M} \| f(x_{r,c}, \theta_{S_n}) - f(x_{r,c}, \theta_T)\|_2^2 \tag{3.1}$$

$$v(r,c) \;=\; \frac{1}{M}\sum_{n=1}^{M} \left\| f(x_{r,c}, \theta_{S_n}) - \frac{1}{M}\sum_{m=1}^{M} f(x_{r,c}, \theta_{S_m}) \right\|_2^2 \tag{3.2}$$

The intuition is that either a large teacher–student regression error (3.1) or a high predictive variance (3.2) suggests that the input lies outside the training manifold; a unified imitation discrepancy metric should therefore respond to both phenomena. The combined score, which is obtained in (3.3) below by normalizing both scores and adding them together, quantifies the student teacher discrepancy, and it is subsequently thresholded to identify anomalous regions.

$$\tilde{e}(r,c) + \tilde{v}(r,c) \;=\; \frac{e(r,c) - e_\mu}{e_\sigma} \;+\; \frac{v(r,c) - v_\mu}{v_\sigma} \tag{3.3}$$

This normalization is obtained by computing the mean (e_μ, v_μ) and standard deviation (e_σ, v_{sigma}) of both scores over a validation set of anomaly-free images, and its dimensionality can be reduced from image size to a single value via averaging. When observing the anomaly score for natural images, the anomaly score will be very close to 0, whereas for anomalous (adversarial) images it will be larger, with higher values indicating a larger the deviation from the training distribution. This behavior is studied in more detail (and confirmed) in Section 4.3.

3.2 Architectural adaptations for adversarial detection

In the original paper, the teacher and student networks share identical architectures. As mentioned in section 3.1, the teacher network typically undergoes an initial pretraining phase involving knowledge distillation from a larger pretrained network (usually trained on ImageNet) to a smaller *patch descriptor network* (see network summaries in Section 4.1.2), which notably shares the architecture design with the student models. In contrast, we eliminate this intermediate step by directly employing a pretrained ResNet-18 model from ImageNet as the teacher. To facilitate efficient extraction of patches, we implement the method described by Bailer et al. [3], which modifies the original network architecture by introducing additional padding, removing strides, and applying a non-dimensionality-reducing MaxPooling operation.

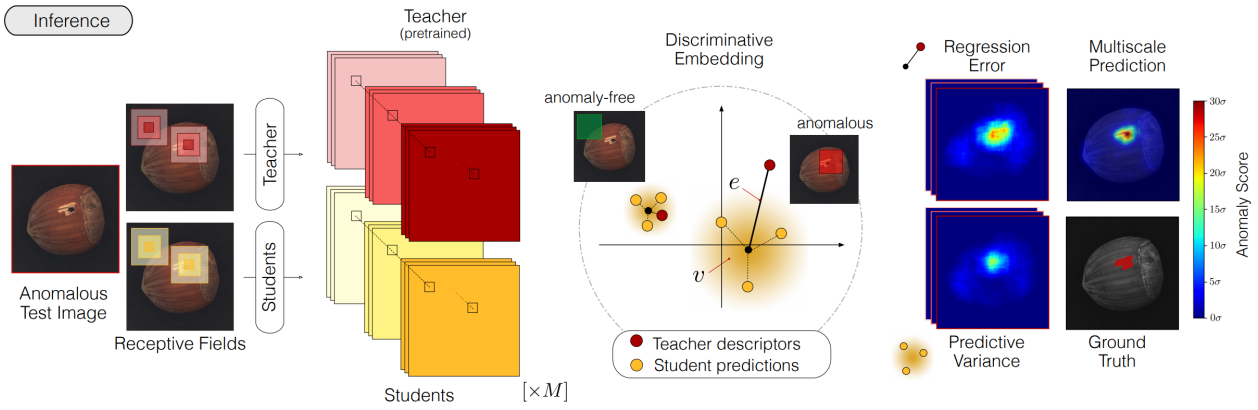


Figure 3.2: Diagram of the inference of the student-teacher network, extracted from Bergmann et al. [7]. For each patch the output of the teacher and ensemble of students will be calculated and compared. Then, the regression error and predictive variance will be used to assess how likely it is to be an anomaly.

We speculate that this architectural modification provides two primary benefits. First, existing research has demonstrated the inherent advantage of using different architectures for teacher and student networks [39]. When both networks share an identical architecture, the students risk overfitting not only to the teacher's output but also to its weights. Consequently, the student might perfectly mimic the teacher even for anomalous inputs, severely reducing detection capabilities. This risk becomes even more critical in the context of adversarial examples, known to be particularly subtle and challenging to detect.

Secondly, a byproduct of using the same pretrained model and weights for the teacher network as the classifier targeted by adversarial attacks ensures consistency in the feature representation. Adversarial perturbations crafted against this classifier are expected to significantly alter the feature maps generated by the teacher. Crucially, the student networks–having been trained exclusively on natural images–would fail to accurately reproduce these perturbed feature maps, thereby amplifying the detectability of adversarial examples.

3.3 Training objective and anomaly score

Then, working in this same direction of preventing the students learning to imitate the teacher even on adversarial images, we implement the hard feature loss described in Batzner et al. [4], which only uses the output elements with the highest loss for backpropagation via a p_{hard} parameter: Given a training image x, we apply a teacher network and a student network to obtain the feature tensors

$$f(x, \theta_t),\ f(x, \theta_s) \in \mathbb{R}^{C \times W \times H}.$$

For every tuple (c, w, h) we define the element-wise squared difference

$$D_{c,w,h} = \left(f(x, \theta_t)_{c,w,h} - f(x, \theta_s)_{c,w,h}\right)^2.$$

Let $p_{\text{hard}} \in [0, 1]$ be a mining factor. Compute d_{hard} as the p_{hard}-quantile of all elements of D. The *hard feature loss* is then the mean over the elements whose error exceeds this threshold:

$$\mathcal{L}_{\text{hard}} = \frac{\sum_{D_{c,w,h} \geq d_{\text{hard}}} D_{c,w,h}}{|\{(c, w, h) \mid D_{c,w,h} \geq d_{\text{hard}}\}|}.$$

Setting $p_{\text{hard}} = 0$ reduces $\mathcal{L}_{\text{hard}}$ to the original student–teacher loss. In our experiments we use either $p_{\text{hard}} = 0$,or $p_{\text{hard}} = 0.999$, which on average keeps approximately 10% of the values in each of the three dimensions of D for back-propagation, depending on the dataset (see Section 4.1.2 for more details).

3.4 Desiderata

We identify three desiderata that a practical adversarial detector should satisfy, together with how we represent them in our evaluation and why the proposed student–teacher design is expected to meet them in principle.

1. **Monotonicity.** Detection accuracy should grow monotonically with the perturbation budget (ϵ); the larger the perturbation, the easier it should be to flag.

2. **Sensitivity at low ϵ.** This growth must be steep enough to catch even very small, sub-effective attacks that do not yet fool the base classifier.

3. **No safe haven.** Taken together, the operating regions of the classifier and detector should overlap so broadly that no perturbation within a practically large ϵ-ball can both mislead the classifier *and* slip past the detector.

We now outline the design aspects that make our detector well-suited to meet these desiderata in principle:

- **Desiderata 1 (Monotonicity).** The teacher shares the classifier backbone, so perturbations that increase classification loss typically move teacher features away from their clean manifold. Students, trained only on clean data, cannot reproduce these off-manifold features. Both the teacher–student regression error and the ensemble variance therefore increase with ϵ, and the normalized sum in (3.3) reflects this as a larger anomaly score.

- **Desiderata 2 (Sensitivity at low ϵ).** Three design choices help at small budgets: (i) *architectural mismatch* (ResNet teacher vs. lightweight patch-descriptor students) reduces the chance that students inadvertently learns to imitate perturbed features; (ii) *ensemble uncertainty* provides an early-warning signal even when the bias (regression error) is modest; (iii) the *hard-feature loss* (Section 3.3) focuses learning on the most informative teacher dimensions, increasing contrast when those are slightly disturbed.

- **Desiderata 3 (No safe haven).** With the teacher tied to the classifier, flipping the label usually requires a feature shift that also enlarges the teacher–student discrepancy; keeping the discrepancy small, in turn, tends to preserve the label. Because our score combines regression error and variance, the joint optimization faced by a detection-aware attacker induces a structural trade-off, which limits the set of inputs that can both fool the classifier and evade detection.

Chapter 4

Experiments

This chapter details our evaluation protocol and results. We specify datasets, models, and threat models, compare against strong baselines across attacks and budgets, and analyze detector behavior and robustness under perfect-knowledge attacks.

4.1 Experimental setup

This section consolidates the datasets, training details, baselines, threat models, attack parameters, and metrics used throughout our study.

4.1.1 Datasets

We conducted our experiments using four benchmark datasets extensively employed in robustness research: MNIST[27], CIFAR-10[25], SVHN[34], and ImageNet [40]. For our evaluations we use the standard dataset splits and normalization where due. See further details in Table 4.1 below:

Dataset	Resolution	#Classes	#Images	Description
MNIST	28×28, grayscale	10	70,000	Handwritten digit images, centered and size-normalized.
SVHN	32×32, RGB	10	99,289	Real-world house numbers from street scenes, cropped and centered on a single digit.
CIFAR-10	32×32, RGB	10	60,000	Low-resolution natural images across diverse object categories.
ImageNet	224x224, RGB	1,000	1,330,000	Large-scale natural image dataset with high intra-class variation; standard ILSVRC subset.

Table 4.1: Summary of datasets used, including resolution, number of classes, total images, and brief description. Sources: [27, 34, 25, 40].

4.1.2 Model & training settings

All networks were trained with using the Adam Optimizer ($\beta_1 = 0.9$, $\beta_2 = 0.999$, and no weight decay), for a maximum of 100000 iterations (and early stopping after 5 epochs). The rest of the hyperparameters were adapted for each task according to what some short testing showed to work best, and are described in Table 4.2.

Parameters	MNIST	SVHN	CIFAR-10	ImageNet
Learning Rate	0.0001	0.0005	0.0005	0.00001
Batch Size	32	32	32	32
Number of Students	10	10	10	10
Patch Size	7	7	17	65
p_{hard}	0	0	0	0.999

Table 4.2: Summary of the hyperparameters used to train our method for the different datasets.

All teacher and target models used ResNet-18 architecture as described in He et al. [16], with the MNIST and SVHN networks using the same modifications as the CIFAR-10 version, with the particularity of the MNIST network having 1 input channel instead of 3. The ImageNet model uses the implementation and weights provided in the torchvision framework.

The student networks are based on the patch descriptor networks described on Bergmann et al. [7], with the ImageNet model using the patch 65 version, CIFAR-10 the patch 17 version modified with double the number for filters for each hidden layer, and MNIST and SVHN using a modified version of patch 17 with a smaller receptive field for a patch size of 7. See Table 4.3 for a summary of these architectures.

For the training, we use only natural images, ensuring the students remain uninformed about adversarial perturbations. For model validation and selection of the best-performing weights, we utilized a validation set attacked using the Fast Gradient Sign Method (FGSM). It must be noted here, that while weight selection via the validation set is useful, it is not necessary to train our method. In addition, while the validation is performed using a FGSM attack method, this one has only been chosen for its efficiency, and that the selection of the attack method should not have as much of an impact on the final model as in a supervised method. To this end, our experiments are carried out using different attacks than the one used for validation.

4.1.3 Adversarial attacks

In our evaluations we mainly used two adversarial attacks, selected due to their prevalence in research:

Fast Gradient Sign Method (FGSM) [14] A single-step gradient-based attack that perturbs input images with minimal computational complexity.

Patch 7 Descriptor

Layer	Output size	Kernel	Stride
Input	7×7×3	–	–
Conv1	5×5×128	3×3	1
Conv2	3×3×256	3×3	1
Conv3	1×1×128	3×3	1
Decode	1×1×512	1×1	1

Patch 17 Descriptor

Layer	Output size	Kernel	Stride
Input	17×17×3	–	–
Conv1	13×13×256	5×5	1
Conv2	9×9×512	5×5	1
Conv3	5×5×512	5×5	1
Conv4	1×1×256	5×5	1
Decode	1×1×512	1×1	1

Patch 65 Descriptor

Layer	Output size	Kernel	Stride
Input	65×65×3	–	–
Conv1	61×61×128	5×5	1
MaxPool	30×30×128	2×2	2
Conv2	26×26×128	5×5	1
MaxPool	13×13×128	2×2	2
Conv3	9×9×128	5×5	1
MaxPool	4×4×128	2×2	2
Conv4	1×1×256	4×4	1
Conv5	1×1×128	3×3	1
Decode	1×1×512	1×1	1

Table 4.3: Layer-wise breakdown of the three patch-descriptor networks. All convolutions use same padding and a Leaky-ReLU with slope $5{\times}10^{-3}$. Note that same padding is used in all layers and a modified version of the max-pooling operation that doe snot reduce dimensionality is used, so that the input size is kept throughout the whole network. The output size is used to show how the field of view is achieved and propagated through the network.

Projected Gradient Descent (PGD) [31] A more powerful, iterative gradient-based attack that provides a stronger baseline for robustness evaluation. We use this attack with 20 iterations, and a step size of $2.5 \cdot \epsilon$/number of steps following up the methodology described in Madry et al. [31] to ensure the attack can reach the boundary. For simplicity no random restarts are used.

Notably PGD is able to craft more effective attacks whilst remaining harder to detect. For this reason we mostly base our analysis on the results obtained under this attack.

4.2 Baselines

In order to assess the performance of our method, we selected some of the most commonly used adversarial detection methods from the state-of-the-art (see Section 4.4). For our selection criteria we took into consideration their detection performance, popularity in the literature, and availability of PyTorch code. This last point was quite important, as in order

to perform the tests under Section 4.5, which needed to access the gradients through the detector network, we needed all models to be implemented using PyTorch. These methods are:

- **ACGAN [46]:** For this model we use the implementation provided with their paper [1], with the hyperparameters described in their paper, with some minor modifications to adapt it to our benchmark. Namely, they upscale MNIST to 32x32, which we avoid and instead use the 28x28 original image size. In addition, we also train it for SVHN using the same hyperparameters as in CIFAR-10.
- **LID [30]:** We used the implementation provided by Lee et al. [28],[2] which used a linear regressor to ensemble the LID statistic across layers.
- **Mahalanobis [28]:** We used a wrapper around the code from the original repository to implement this method.[2] For the hyperparameters we use the ones that were found to be optimal in their testing.

4.2.1 Evaluations metrics

The primary metric employed in our analysis is the pAUC-0.2 (partial Area Under Curve) metric introduced by Wang et al. [46], which quantifies the area under the ROC curve up to a false positive rate of 20%. The reason behind this choice, in comparison to using the AUC, is that restricting evaluation to the lower end of maximum false positive rate allows us to exclude higher rates (e.g., 50%), which would be impractically high for real-world scenarios, and which would skew an evaluation when using the full AUC. Additionally, in some of our tests we will use the *Adversarial Detection Accuracy*, defined as the percentage of adversarial examples correctly identified as anomalous by our detection mechanism, evaluated at fixed false positive rates of 10% and 1%, based on the natural (unperturbed) counterparts from the same test set. For contextual comparison, we also report the accuracy of the underlying classifier.

Lastly, to comprehensively evaluate the robustness of our entire system, on our *perfect knowledge attack* we measure the *Attack Success Rate* (ASR), defined as the proportion of samples where an adversarial attack successfully bypassed both the classifier and the detection system simultaneously, considering a false positive rate of 10% for the detector. To calculate this metric, we only consider those samples whose natural image was successfully classified by the target model.

4.3 Behavior of the anomaly score under an adversarial attack

Having defined our detector, first we aim to inspect the behavior of the anomaly score with regards to an adversarial attack strength. What we hope to see here is both a steep increase

[1] https://github.com/wanghangpsu/acgan-ada

[2] `https://github.com/pokaxpoka/deep_Mahalanobis_detector`

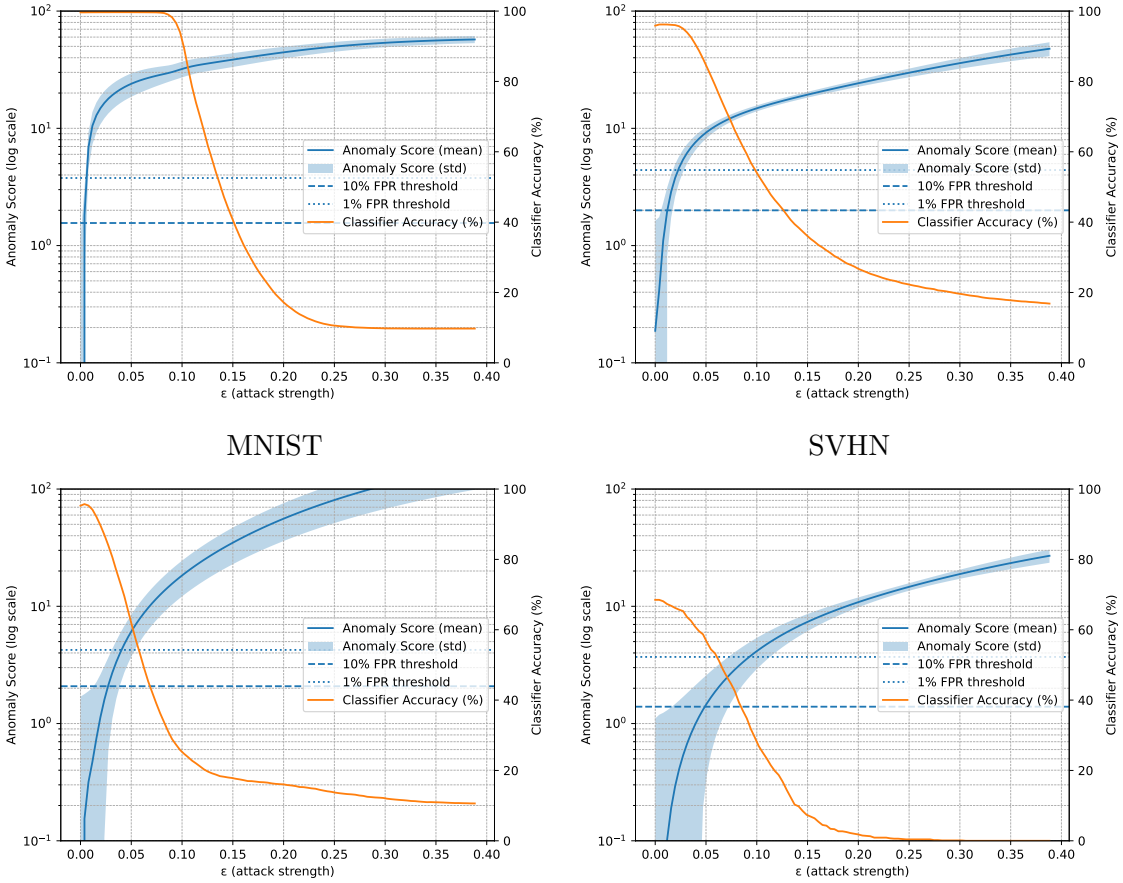


Figure 4.1: Anomaly score obtained by our detector and classifier accuracy displayed for varying FGSM ℓ_∞ perturbation magnitude (ϵ) on MNIST (a), SVHN (b), CIFAR-10 (c) and ImageNet (d). The 10% and 1% false positive rate thresholds have also been displayed for reference.

in the anomaly score in comparison to what is obtained for natural images, as well as the monotonicity characteristic described in Section 3.4.

Experiment We attack a target classifier model on all four datasets with an ℓ_∞ FGSM attack under a *zero detector knowledge attack* scenario. This choice allows us to change the magnitude of the attack (ϵ) without changing the direction, and thus allowing us to inspect the behavior of our Anomaly Score when changing *only* the magnitude (or strength) of the attack. We do so by computing anomaly score for every $\epsilon \in [1/255, 2/255, ...100/255)$, and then calculating the mean and standard deviation for each ϵ. This allows us to examine whether the anomaly score rises rapidly and monotonically with increasing ϵ, and whether it crosses the false-positive thresholds well before the classifier's accuracy begins to drop.

The results can be observed in Figure 4.1, and they show that the anomaly score will quickly increase proportionally to the attack strength.

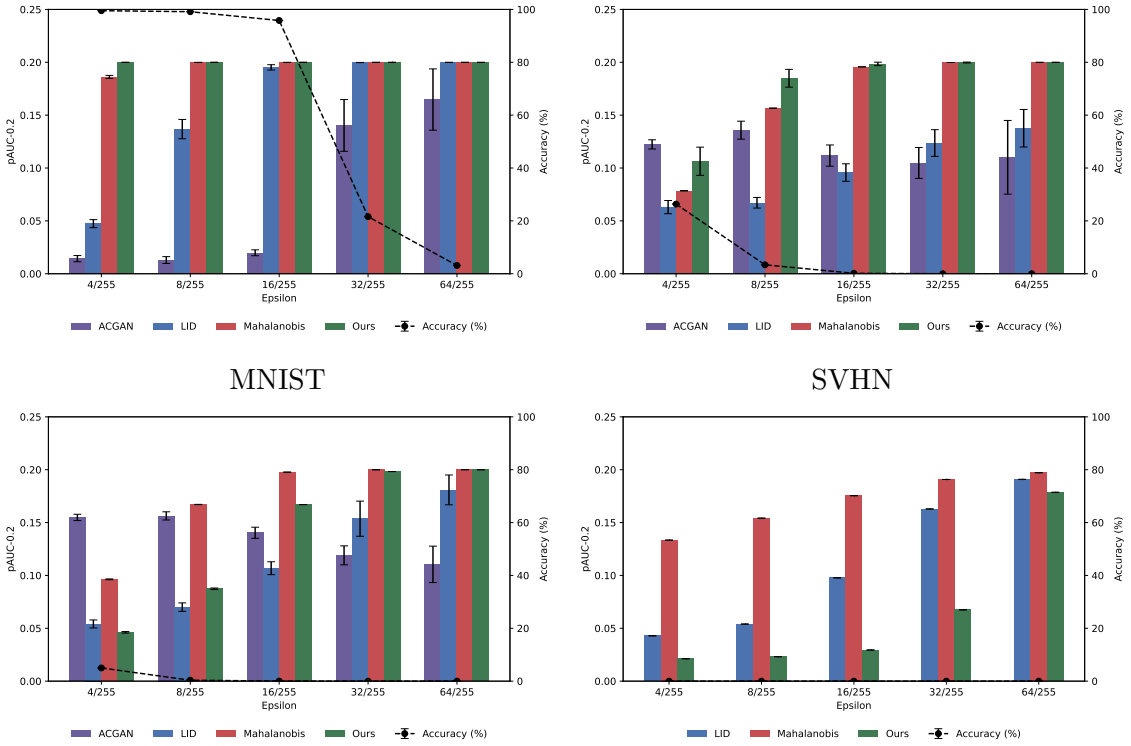
Results Interestingly the anomaly score behaves differently based on the difficulty of the dataset: On MNIST and SVHN it increases very quickly, surpassing both the 1% and 10% false positive thresholds for very small ϵ values, much earlier than any value that allows a significant drop in classifying performance in the target model. On CIFAR-10 and ImageNet, behavior is a bit different, with anomaly score still increasing early, but detectable for much more reasonable ϵ values, between 8/255 and 16/255. By those values, we already see a measurable decrease in classifying performance. This result indicates that our detector is highly sensitive to adversarial perturbations and that the rate at which the score crosses a predefined false-positive threshold is dataset-dependent. In particular, the near-instantaneous rise on MNIST suggests that simpler, low-resolution data manifolds leave little room for imperceptible perturbations, so even minimal changes are deemed anomalous. Conversely, the more complex, high-dimensional manifolds of CIFAR-10 and ImageNet tolerate small perturbations before the detector fires. Taken together, these observations (i) confirm that the proposed anomaly score is a reliable early-warning signal across diverse visual domains, and (ii) underscore the necessity of calibrating detection thresholds per dataset (or model) to strike the desired balance between robustness and false alarms.

4.4 Evaluating adversarial detection accuracy

Whilst the results obtained in the previous experiment are interesting, and confirming the monotonicity characteristic in our method is valuable, in practice empirical performance is the key benchmark to assess a good detector. In this second experiment we aim to measure the detection performance of our model under the previously defined *zero detector knowledge attack*. By observing simultaneously the performance of our detector and the classifier under this attack, the goal is to inspect the *sensitivity at low* ϵ, that is to see that the detector is capable of detecting even those attacks that have not been able to bypass the classifier.

Experiment We evaluate our detector model, as well as all baselines in all datasets against a PGD-20 (ℓ_∞) attack (results from other attack configurations reported in the Appendix A) under the *zero detector knowledge* threat model from Section 2.1.3. In all datasets but ImageNet (due to computational limitations) we trained five detector models on different random seeds, and report the mean and standard deviation between them. The results were obtained by first computing their detection scores on 10 000 natural images, and then attacking a shared classifier model with varying ϵ values. In Figure 4.2 we report the pAUC-0.2 metric for each (calculated with the score of the 10000 natural images and 10 000 adversarial counterparts), as well as the accuracy of the classifier on the attacked images that the detector aim to identify as adversarial. In interpreting these results, two aspects are of particular interest: (i) how the pAUC changes with ϵ, and (ii) the accuracy of each detector at the point where the classifier begins to lose accuracy.

Results When evaluating the results fo all four detector models, it first surprised us the outstanding performance of the ACGAN detector for small ϵ values, but it comes at the cost of decreasing performance on attacks of higher magnitude. Although it is not the goal of



MNIST SVHN

CIFAR-10 ImageNet

Figure 4.2: pAUC-0.2 for PGD-20 ℓ_∞ on different ϵ on MNIST (a), SVHN (b), CIFAR-10 (c) and ImageNet (d). The bars represent the mean over 5 random seeds, with the standard deviation displayed as the error bars in all datasets except ImageNet, where a single seed was used. For additional context, the accuracy of the target model under each attack is represented as the black dotted line.

this experiment, this gives us a glimpse on the *no safe haven* characteristic. Although it is capable of detecting very small attacks, it is not as good on the larger attacks, where the classifier's performance starts to decrease. This behavior is better understood when considering the design of this algorithm, which combines the discriminator statistic with the class log-likelihood of the predicted class, which has been trained during the GAN training to classify even the generated images. In essence it behaves as robust training classifier, which can be exploited in inference to detect adversarial samples. The disadvantage of this technique is that it only works as long as the classifier head remains relatively informative. Although it is an interesting and powerful technique, we believe that for the task of adversarial attack detection, the other three detectors are more suitable, so we will focus our analysis for this experiment on them.

On MNIST, all three detectors show very good performance, being able to detect adversarial samples at least as soon as they start to be effective at decreasing the classifier performance. Among st them, LID shows the worse performance, with our detector showing a small improvement over the Mahalanobis detector for the smaller epsilons, being able to reliably detect even the smallest attack.

On SVHN we see a similar pattern, with LID being the worse, and our detector having a small improvement over Mahalanobis. Critically here, the attacker is able to be more effective, even at the lowest ϵ setting. This magnifies the importance of being able to detect such small attacks for this dataset.

On CIFAR-10 and ImageNet we see a different pattern: the Mahalanobis detector manifests clearly the strongest performance, with the other two being slightly behind, and at a relative tie. On CIFAR-10, our detector still outperforms the LID detector, whereas on ImageNet is the other way around. We explain this change of dynamics due to the nature of the datasets, which show higher variability on the images. On a very simplified way, we could distinguish our method from the ones proposed for the Mahalanobis and LID detector with a very simple concept: the LID and Mahalanobis detector work by modeling the features produced by the classifier model, whereas in our method we are modeling the feature extractor part of the classifier (that is the images to features function). By increasing the variability of the images, modeling the feature extractor might grow in complexity, while modeling the features themselves (considering the same model architecture) remains relatively similar in comparison. We speculate that this gap might be further closed by increasing the complexity of the student models (which remains a relatively simple one), which we could not do due to computational constraints, so it remains to be tested.

4.5 Evaluation under the perfect knowledge attack

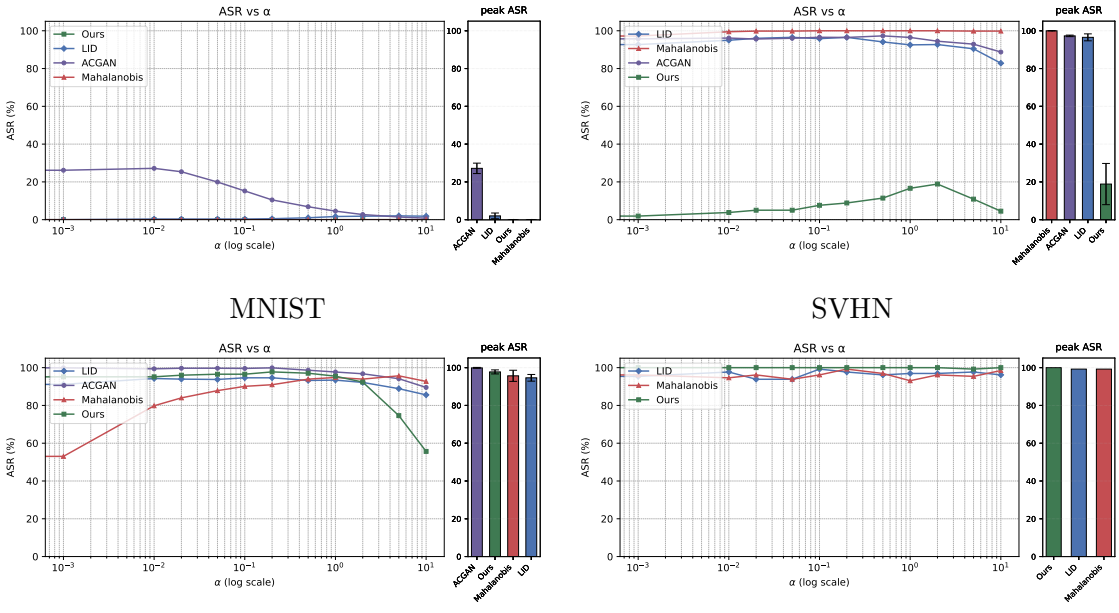


Figure 4.3: Attack Success Rate (ASR) vs α. For each dataset we display the line as the mean over the results for different seeds for a given α, and the deviation on the left plot. On the right of each of these plots we report the maximum ASR achieved (worst case scenario).

A completely different adversarial attack scenario than the one covered in Section 4.4. In this section we evaluate the model under the *perfect knowledge attack*. The goal of this experiment is two-fold: First, we can assess the performance of the detector under the most challenging scenario, but also it allows us to evaluate the third characteristic *no safe haven*, as by crafting an attack that targets the classifier and detector simultaneously, we perform in practice a search for large attacks that could pass undetected by the detector. Although this is not the type of exhaustive search that such characteristic would require, it serves the purpose of testing if such adversarial example is possible to be found within a more reasonable computation time.

Experiment In order to evaluate our model when the attacker is aware of the detection strategy, we use the *detection-aware PGD* attack, under the *perfect knowledge attack* threat model, as defined in Section 2.1.3. For this attack, a trade-off parameter α has to be set to influence how much the attack will focus on the detector or on the classifier. We address this in our experiment by performing a sweep over $\alpha \in [0.0, 0.001, 0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0, 5.0, 10.0]$ and report the performance of the detector, the classifier, and the *Attack Success Rate* (see Section 4.2.1 for more details). By doing this we hope to give a an overview of the trade-off of setting this parameter, as well as a clear answers as to what the perfect-knowledge attack can achieve against our detector. It is important to note here that for this type of attack, the role of ϵ is not as much to set the magnitude, but more that of limiting the perturbation space: Whereas for the attack in the previous experiments the most optimal point is usually (though admittedly no necessarily) in the boundary of the $\epsilon-$ball, as larger perturbations are usually more effective against the classifier, this is not usually the case when the detector is incorporated into the attack, as the larger perturbations are usually more detectable. And so usually the most effective attack (when a large enough ϵ is set) will be located in a middle point, that is effective against both the classifier and detector model. Following this, we limited the perturbation space with a generous $\epsilon = 0.1$ to maximize the search space, and used 20 iterations. Although using more iterations (and adding random restarts), or just carrying an exhaustive search might increase the success rate of the attacker, we argue that it is computationally impractical. With the current configuration it already takes around 100 seconds on an NVIDIA RTX 3090 to sweep across all α with a single image for our detector, any meaningful increase in the number of iterations will further increase this time.

Results In Figure 4.3, we report both the ASR for all α tested, as well as the peak ASR, which informs us of the maximum effectiveness the *detection-aware PGD* attacker is able to achieve.

When looking into MNIST, the attack was unsuccessful against all detectors. This is consistent with what we could expect from the results in Section 4.4, where on MNIST all detectors where very accurate even for very small ϵ. On SVHN we start noticing some differences: Our detector is the only one that remains robust against this type of attack, that is only effective around 20% of the times, whilst being almost perfectly effective against all other detectors. Inspecting the results of CIFAR-10 and ImageNet, we see that the attack can bypass all methods with a very large success rate.

Comparing this to the results from Section 4.4, we notice that there is a mostly large correlation: achieving a high *Sensitivity at low* ϵ, ensures that there is no overlap between the regions where both the classifier and the detector can be bypassed. That being said, we notice some non-direct discrepancies, as from the previous experiment one might expect mahalanobis to perform similarly or better than our detector, depending on the dataset. We explain this from the design philosophy of both methods: Detectors such as LID or mahalanobis model the classifier features at each layer $f(\mathbf{x}, \theta_\ell)$, and detect adversarial examples based on how these deviate from those obtained on natural images. In contrast our method models the feature extractor $f(\mathbf{x}, \theta_\ell) : \mathbb{R}^{d_0} \to \mathbb{R}^{d_\ell}$. This means that methods such as LID and mahalanobis only need large shifts on the feature space to detect and adversarial attack, whereas our detector also needs a shift in the image space. For this reason, these two methods can work better at CIFAR and ImageNet, where smaller ϵ can already allow a successful attack, while ours struggles. On the other hand, they are susceptible to attacks that can successfully bypass the classifier while keeping the feature map similar to those of natural images, while in our method the shift in the image space is already enough to trigger a large enough anomaly score. In this regard, it remains an open question whether employing a more robust classifier—one that requires a larger ϵ to be bypassed—would improve the relative performance of our detector. Because LID and mahalanobis model only the classifier's feature space, a more robust classifier might yield smaller feature-space deviations and thus reduce their effectiveness; our detector, which also leverages image-space information, might be less affected by such changes.

4.6 Limitations

While the experiments demonstrate competitive and, in some cases, superior performance compared to strong baselines, several limitations should be acknowledged.

4.6.1 Choice and tuning of baselines.

Our baseline set is necessarily incomplete. Certain methods could not be included due to the absence of public code, or because available implementations were not in PyTorch—an essential requirement for our evaluation framework, which demands gradient access for perfect-knowledge attacks. Lack of code availability is itself a reproducibility concern, but it still means our comparisons omit potentially relevant approaches. Furthermore, for baselines we retained the hyperparameters recommended in their original publications. Although re-optimizing might yield marginal improvements, these configurations were already reported as near-optimal by their authors (who already carried a hyperparameter search) and thus represent a fair reference point.

4.6.2 Hyperparameter and architecture search for our method.

In contrast, the hyperparameter search for our detector was deliberately limited, especially for the computationally demanding ImageNet setting. This constraint likely leaves untapped

performance potential—perhaps more so than for the baselines, some of which (e.g., Mahalanobis) are simpler and were developed with substantially greater resources. Our student–teacher setup introduces additional degrees of freedom, such as student network architecture and choice of pretrained teacher. We mostly reused student designs from Bergmann et al. [7], with only minor adjustments (e.g., reduced patch size for MNIST/SVHN, doubled filters for CIFAR-10). Even these small changes already yielded measurable gains in some datasets, suggesting that a more systematic search, possibly via Neural Architecture Search (NAS), could further improve results. For ImageNet, we used the default patch-65 descriptor without exploration, so it is reasonable to expect that more suitable architectures exist. Likewise, our manual hyperparameter tuning was minimal; a broader automated search could significantly enhance robustness and detection accuracy. Finally, we restricted our teacher model to a single pretrained classifier architecture (ResNet-18). The choice of the pre-trained feature extration model has already been demonstrated to have a significant impact [17], and so exploring alternative pretrained classifiers—including deeper ResNets, Vision Transformers, or other modern architectures—could also lead to better detection performance.

Chapter 5

Conclusions and Future Work

We have proposed a novel unsupervised adversarial attack detector that is capable of detecting adversarial examples with high accuracy–an accuracy that rises with perturbation magnitude–while preserving a low false positive rate on clean images and remaining robust to fully adaptive, white-box attackers.

We have performed three sets of experiments with white-box attacks, in which we (i) analyze the behavior of the method, (ii) compare it against state-of-the-art detectors across various epsilon values with no knowledge on the detector, and (iii) compare it against the same detectors using a *perfect knowledge attack*, in which the attacker has access to both the classifier as well as the detector.

When inspecting the behavior of the method, it worked as expected across all four datasets, where the anomaly score on average increases as the magnitude of the attack increases.

When comparing it to the other datasets, our method performs consistently across all datasets among the top-scoring methods for the scenarios with high magnitude perturbations (i.e., $\epsilon = 32/255$ and $\epsilon = 64/255$). In addition, when evaluating the more challenging ϵ values, we observe that our model outperforms the baselines in two out of the four datasets. While arguably a relative simple method such as Mahalanobis obtains similar performances on these scenarios, we think that there is value in having an unsupervised data-driven learned method, as there remains room for further improvement (e.g., through data acquisition and better neural architectures), whereas Mahalanobis performs likely at its full potential.

Finally, when evaluating the methods in the most challenging *perfect knowledge attack* scenario, where the attacker has access to the weights of both the classifier as well as the detector, we observe that for two datasets (CIFAR and ImageNet) plenty of advances need to be made before this can be useful. On the other datasets, our method performs consistently with the best methods (MNIST) or better than the best methods (SVHN).

Additionally, our approach might substantially benefit from future advancements in the anomaly detection methodologies, a field that is still evolving. Integrating advances from both fields could yield substantial gains in overall system robustness.

Furthermore, internal testing suggested that the choice of student network architectures—which critically affects the receptive field—plays a crucial role in detection performance. Therefore, exploring optimized network architectures through Neural Architecture Search

(NAS) represents an interesting and promising avenue for enhancing the proposed approach.

Finally, while this requires significant advances in the field of neural network verification, applying this method together with network that is provably robust against small perturbations, would be an ambitious step towards making ai applications more robust in practice.

Bibliography

[1] Ahmed Aldahdooh, Wassim Hamidouche, Sid Ahmed Fezza, and Olivier Déforges. Adversarial example detection for DNN models: A review and experimental comparison. *Artificial Intelligence Review*, 55(6):4403–4462, 2022.

[2] Laurent Amsaleg, Oussama Chelly, Teddy Furon, Stéphane Girard, Michael E Houle, Ken-ichi Kawarabayashi, and Michael Nett. Estimating local intrinsic dimensionality. In *Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 29–38, 2015.

[3] Christian Bailer, Tewodros Habtegebrial, Didier Stricker, et al. Fast feature extraction with CNNs with pooling layers. *arXiv preprint arXiv:1805.03096*, 2018.

[4] Kilian Batzner, Lars Heckler, and Rebecca König. EfficientAD: Accurate visual anomaly detection at millisecond-level latencies. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 128–138, 2024.

[5] Christoph Baur, Benedikt Wiestler, Shadi Albarqouni, and Nassir Navab. Deep autoencoding models for unsupervised anomaly segmentation in brain MR images. In *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries: 4th International Workshop*, pages 161–169. Springer, 2019.

[6] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. MVTec AD–A comprehensive real-world dataset for unsupervised anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9592–9600, 2019.

[7] Paul Bergmann, Michael Fauser, David Sattlegger, and Carsten Steger. Uninformed students: Student-teacher anomaly detection with discriminative latent embeddings. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4182–4191, 2020.

[8] Paul Bergmann, Xin Jin, David Sattlegger, and Carsten Steger. The MVTec 3D-AD dataset for unsupervised 3D anomaly detection and localization. In *Proceedings of the 17th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. SCITEPRESS-Science and Technology Publications, 2022.

[9] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14, 2017.

[10] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57. IEEE, 2017.

[11] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *International Conference on Machine Learning*, pages 1310–1320. PMLR, 2019.

[12] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. Detecting adversarial samples from artifacts. *arXiv preprint arXiv:1703.00410*, 2017.

[13] Zhitao Gong and Wenlu Wang. Adversarial and clean data are not twins. In *Proceedings of the sixth international workshop on exploiting artificial intelligence techniques for data management*, pages 1–5, 2023.

[14] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR*, 2015.

[15] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern ecognition*, pages 770–778, 2016.

[17] Lars Heckler, Rebecca König, and Paul Bergmann. Exploring the importance of pretrained feature extractors for unsupervised anomaly detection and localization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2916–2925, 2023.

[18] Lars Heckler, Jan-Hendrik Neudeck, Ulla Scheler, Rebecca König, and Carsten Steger. The MVTec AD 2 dataset: Advanced scenarios for unsupervised anomaly detection. *arXiv preprint arXiv:2503.21622*, 2025.

[19] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *5th International Conference on Learning Representations, ICLR*. OpenReview.net, 2017.

[20] Dan Hendrycks and Kevin Gimpel. Early methods for detecting adversarial images. In *5th International Conference on Learning Representations, ICLR*. OpenReview.net, 2017.

[21] Hossein Hosseini, Yize Chen, Sreeram Kannan, Baosen Zhang, and Radha Poovendran. Blocking transferability of adversarial examples in black-box learning systems. *arXiv preprint arXiv:1703.04318*, 2017.

[22] Michael E Houle. Local intrinsic dimensionality ii: multivariate analysis and distributional support. In *International Conference on Similarity Search and Applications*, pages 80–95. Springer, 2017.

[23] Guy Katz, Clark Barrett, David L Dill, Kyle Julian, and Mykel J Kochenderfer. Reluplex: An efficient SMT solver for verifying deep neural networks. In *International Conference on Computer Aided Verification*, pages 97–117. Springer, 2017.

[24] Anouar Kherchouche, Sid Ahmed Fezza, Wassim Hamidouche, and Olivier Déforges. Detection of adversarial examples in deep neural networks with natural scene statistics. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.

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

[26] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial Machine Learning at Scale. In *5th International Conference on Learning Representations, ICLR*, 2017.

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

[28] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

[29] Julia Lust and Alexandru Paul Condurache. Gran: An efficient gradient-norm based detector for adversarial and misclassified examples. In *28th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN*, pages 7–12, 2020.

[30] Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi N. R. Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E. Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. In *6th International Conference on Learning Representations, ICLR*. OpenReview.net, 2018.

[31] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks. In *6th International Conference on Learning Representations, ICLR*, 2018.

[32] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 135–147, 2017.

[33] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. In *5th International Conference on Learning Representations, ICLR*. OpenReview.net, 2017.

[34] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Baolin Wu, Andrew Y Ng, et al. Reading digits in natural images with unsupervised feature learning. In *NIPS workshop on deep learning and unsupervised feature learning*, 2011.

[35] Augustus Odena, Christopher Olah, and Jonathon Shlens. Conditional image synthesis with auxiliary classifier GANs. In *International Conference on Machine Learning*, volume 70, pages 2642–2651. PMLR, 2017.

[36] Kyoungchan Park and Pilsung Kang. Detection and defense: Student-teacher network for adversarial robustness. *IEEE Access*, 12:82742–82752, 2024.

[37] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *International Conference on Learning Representations*, 2018.

[38] Shahbaz Rezaei and Xin Liu. A target-agnostic attack on deep models: Exploiting security vulnerabilities of transfer learning. In *8th International Conference on Learning Representations, ICLR*, 2020.

[39] Marco Rudolph, Tom Wehrbein, Bodo Rosenhahn, and Bastian Wandt. Asymmetric student-teacher networks for industrial anomaly detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2592–2602, 2023.

[40] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115: 211–252, 2015.

[41] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Ursula Schmidt-Erfurth, and Georg Langs. Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In *International Conference on Information Processing in Medical Imaging*, pages 146–157. Springer, 2017.

[42] Thomas Schlegl, Philipp Seeböck, Sebastian M Waldstein, Georg Langs, and Ursula Schmidt-Erfurth. f-anogan: Fast unsupervised anomaly detection with generative adversarial networks. *Medical Image Analysis*, 54:30–44, 2019.

[43] Aleksei Vasilev, Vladimir Golkov, Marc Meissner, Ilona Lipp, Eleonora Sgarlata, Valentina Tomassini, Derek K Jones, and Daniel Cremers. q-space novelty detection with variational autoencoders. In *Computational Diffusion MRI: MICCAI Workshop*, pages 113–124. Springer, 2020.

[44] Chengjie Wang, Wenbing Zhu, Bin-Bin Gao, Zhenye Gan, Jiangning Zhang, Zhihao Gu, Shuguang Qian, Mingang Chen, and Lizhuang Ma. Real-iad: A real-world multi-view dataset for benchmarking versatile industrial anomaly detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22883–22892, 2024.

[45] Guodong Wang, Shumin Han, Errui Ding, and Di Huang. Student-teacher feature pyramid matching for anomaly detection. In *32nd British Machine Vision Conference 2021, BMVC 2021*, page 306. BMVA Press, 2021.

[46] Hang Wang, David J. Miller, and George Kesidis. Anomaly detection of adversarial examples using class-conditional generative adversarial networks. *Computers & Security*, 124(C):102956, Jan. 2023. ISSN 0167-4048. doi: 10.1016/j.cose.2022.102956. URL `https://doi.org/10.1016/j.cose.2022.102956`.

[47] Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pages 5286–5295. PMLR, 2018.

[48] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International Conference on Machine Learning*, pages 7472–7482. PMLR, 2019.

[49] Huan Zhang, Tsui-Wei Weng, Pin-Yu Chen, Cho-Jui Hsieh, and Luca Daniel. Efficient neural network robustness certification with general activation functions. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 31, 2018.

[50] Yang Zou, Jongheon Jeong, Latha Pemula, Dongqing Zhang, and Onkar Dabeer. Spot-the-difference self-supervised pre-training for anomaly detection and segmentation. In *European Conference on Computer Vision*, pages 392–408. Springer, 2022.

[51] Fei Zuo and Qiang Zeng. Exploiting the sensitivity of l2 adversarial examples to erase-and-restore. In *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security*, pages 40–51, 2021.

Appendix A

Full Detection Performance Tables

This appendix contains the full quantitative results from our detection accuracy experiments, complementing the summary plots in Chapter 4. For each dataset, we report the performance of our method and all baselines across multiple attack types, perturbation budgets (ϵ), and norms (ℓ_∞ and ℓ_2).

Each table is organized as follows:

- **Attack**: Specifies the adversarial method, its ℓ_p-norm, and ϵ.
- **Detector columns** (*acgan*, *lid*, *mahalanobis*, *Ours*): pAUC-0.2 detection scores, reported as mean ± standard deviation over 5 random seeds (1 seed for ImageNet).
- **Classifier Acc**: Accuracy of the target classifier on the adversarially perturbed images corresponding to that row.All natural-image (unperturbed) accuracies are shown at the bottom of each table for reference.

Attack	acgan	lid	mahalanobis	Ours	Classifier Acc
FGSM(l_∞, $\epsilon = \frac{4}{255}$)	0.0143±0.00288	0.0481±0.00380	0.1884±0.00127	0.2000±0.00000	99.42±0.00000
FGSM(l_∞, $\epsilon = \frac{8}{255}$)	0.0125±0.00327	0.1365±0.00911	0.1999±0.00000	0.2000±0.00000	98.86±0.00000
FGSM(l_∞, $\epsilon = \frac{16}{255}$)	0.0161±0.00264	0.1956±0.00251	0.2000±0.00000	0.2000±0.00000	96.37±0.00000
FGSM(l_∞, $\epsilon = \frac{32}{255}$)	0.0631±0.00900	0.2000±0.00003	0.2000±0.00000	0.2000±0.00000	69.06±0.00000
FGSM(l_∞, $\epsilon = \frac{64}{255}$)	0.1361±0.02908	0.2000±0.00000	0.2000±0.00000	0.2000±0.00000	12.08±0.00000
PGD-20(l_∞, $\epsilon = \frac{4}{255}$)	0.0143±0.00294	0.0474±0.00380	0.1862±0.00142	0.2000±0.00001	99.52±0.00000
PGD-20(l_∞, $\epsilon = \frac{8}{255}$)	0.0129±0.00325	0.1368±0.00914	0.1999±0.00001	0.2000±0.00000	99.14±0.00000
PGD-20(l_∞, $\epsilon = \frac{16}{255}$)	0.0198±0.00279	0.1952±0.00253	0.1999±0.00000	0.2000±0.00000	95.82±0.00400
PGD-20(l_∞, $\epsilon = \frac{32}{255}$)	0.1403±0.02448	0.1998±0.00010	0.2000±0.00000	0.2000±0.00000	21.61±0.00400
PGD-20(l_∞, $\epsilon = \frac{64}{255}$)	0.1648±0.02900	0.1999±0.00002	0.2000±0.00001	0.1999±0.00000	3.13±0.00800
PGD-20(l_2, $\epsilon = 0.25$)	0.0182±0.00107	0.0230±0.00081	0.0269±0.00023	0.1538±0.00315	99.36±0.00000
PGD-20(l_2, $\epsilon = 0.5$)	0.0178±0.00177	0.0348±0.00237	0.0971±0.00230	0.1949±0.00054	98.50±0.00748
PGD-20(l_2, $\epsilon = 1$)	0.0305±0.00240	0.1119±0.00609	0.1908±0.00114	0.1996±0.00004	88.61±0.04176
PGD-20(l_2, $\epsilon = 2$)	0.1352±0.02245	0.1899±0.00194	0.1987±0.00011	0.1999±0.00002	24.59±0.12816
				Nat Accuracy	99.63%

Table A.1: Detection accuracy evaluation under different targeted attacks on MNIST.

Attack	acgan	lid	mahalanobis	Ours	Classifier Acc
FGSM(l_∞, $\epsilon = \frac{4}{255}$)	0.0647±0.00398	0.0538±0.00642	0.0622±0.00012	0.1268±0.01676	52.34±0.00000
FGSM(l_∞, $\epsilon = \frac{8}{255}$)	0.0807±0.00707	0.0703±0.00745	0.1229±0.00004	0.1932±0.00586	35.04±0.00000
FGSM(l_∞, $\epsilon = \frac{16}{255}$)	0.1108±0.01950	0.1031±0.01159	0.1858±0.00001	0.1996±0.00066	19.12±0.00000
FGSM(l_∞, $\epsilon = \frac{32}{255}$)	0.1313±0.02957	0.1226±0.01558	0.1987±0.00000	0.1999±0.00010	8.24±0.00000
FGSM(l_∞, $\epsilon = \frac{64}{255}$)	0.1349±0.03719	0.1654±0.01360	0.1999±0.00000	0.2000±0.00000	6.16±0.00000
PGD-20(l_∞, $\epsilon = \frac{4}{255}$)	0.1223±0.00435	0.0630±0.00626	0.0785±0.00024	0.1064±0.01332	26.35±0.01200
PGD-20(l_∞, $\epsilon = \frac{8}{255}$)	0.1358±0.00851	0.0671±0.00502	0.1566±0.00017	0.1849±0.00842	3.37±0.00490
PGD-20(l_∞, $\epsilon = \frac{16}{255}$)	0.1118±0.01004	0.0957±0.00827	0.1957±0.00002	0.1985±0.00162	0.17±0.00490
PGD-20(l_∞, $\epsilon = \frac{32}{255}$)	0.1048±0.01462	0.1236±0.01264	0.1998±0.00000	0.1998±0.00041	0.00±0.00000
PGD-20(l_∞, $\epsilon = \frac{64}{255}$)	0.1101±0.03486	0.1376±0.01775	0.1999±0.00000	0.2000±0.00007	0.00±0.00000
PGD-20(l_2, $\epsilon = 0.25$)	0.0605±0.00196	0.0620±0.00738	0.0475±0.00013	0.0272±0.00029	69.50±0.02482
PGD-20(l_2, $\epsilon = 0.5$)	0.1106±0.00412	0.0636±0.00590	0.0681±0.00021	0.0553±0.00212	29.61±0.06560
PGD-20(l_2, $\epsilon = 1$)	0.1206±0.01012	0.0667±0.00533	0.1424±0.00012	0.1471±0.00679	4.70±0.01960
PGD-20(l_2, $\epsilon = 2$)	0.0917±0.01439	0.0924±0.00869	0.1930±0.00005	0.1929±0.00265	0.19±0.01166
				Nat Accuracy	95.85%

Table A.2: Detection accuracy evaluation under different targeted attacks on SVHN.

Attack	acgan	lid	mahalanobis	Ours	Classifier Acc
FGSM(l_∞, $\epsilon = \frac{4}{255}$)	0.0530±0.00310	0.0901±0.00912	0.1616±0.00030	0.0346±0.00016	55.51±0.00000
FGSM(l_∞, $\epsilon = \frac{8}{255}$)	0.0631±0.00647	0.1216±0.01662	0.1985±0.00002	0.1154±0.00170	46.54±0.00000
FGSM(l_∞, $\epsilon = \frac{16}{255}$)	0.0902±0.01652	0.1727±0.01788	0.2000±0.00000	0.1990±0.00003	32.14±0.00400
FGSM(l_∞, $\epsilon = \frac{32}{255}$)	0.1230±0.03628	0.1938±0.00606	0.2000±0.00000	0.2000±0.00000	16.18±0.00000
FGSM(l_∞, $\epsilon = \frac{64}{255}$)	0.1337±0.04103	0.1963±0.00451	0.1496±0.01311	0.2000±0.00000	12.78±0.00400
PGD-20(l_∞, $\epsilon = \frac{4}{255}$)	0.1549±0.00297	0.0541±0.00379	0.0963±0.00040	0.0462±0.00073	4.99±0.00400
PGD-20(l_∞, $\epsilon = \frac{8}{255}$)	0.1563±0.00389	0.0700±0.00399	0.1671±0.00014	0.0874±0.00072	0.35±0.00748
PGD-20(l_∞, $\epsilon = \frac{16}{255}$)	0.1403±0.00527	0.1068±0.00609	0.1977±0.00002	0.1670±0.00004	0.04±0.00400
PGD-20(l_∞, $\epsilon = \frac{32}{255}$)	0.1190±0.00899	0.1537±0.01663	0.2000±0.00000	0.1983±0.00001	0.00±0.00000
PGD-20(l_∞, $\epsilon = \frac{64}{255}$)	0.1105±0.01722	0.1809±0.01410	0.2000±0.00000	0.2000±0.00000	0.00±0.00000
PGD-20(l_2, $\epsilon = 0.25$)	0.0995±0.00194	0.0732±0.00363	0.0629±0.00073	0.0264±0.00025	37.24±0.02498
PGD-20(l_2, $\epsilon = 0.5$)	0.1476±0.00255	0.0525±0.00306	0.0825±0.00038	0.0393±0.00058	8.24±0.07060
PGD-20(l_2, $\epsilon = 1$)	0.1535±0.00279	0.0619±0.00304	0.1515±0.00019	0.0725±0.00064	0.90±0.06177
PGD-20(l_2, $\epsilon = 2$)	0.1427±0.00365	0.0956±0.00509	0.1935±0.00007	0.1346±0.00019	0.12±0.01414
				Nat Accuracy	95.27%

Table A.3: Detection accuracy evaluation under different targeted attacks on CIFAR-10.

Attack	lid	mahalanobis	Ours	Classifier Acc
FGSM(l_∞, $\epsilon = \frac{4}{255}$)	0.0388	0.0314	0.0252	4.00
FGSM(l_∞, $\epsilon = \frac{8}{255}$)	0.0821	0.1098	0.0403	4.10
FGSM(l_∞, $\epsilon = \frac{16}{255}$)	0.1285	0.1744	0.1237	3.10
FGSM(l_∞, $\epsilon = \frac{32}{255}$)	0.1731	0.1940	0.1964	1.60
FGSM(l_∞, $\epsilon = \frac{64}{255}$)	0.1860	0.1990	0.2000	0.10
PGD-20(l_∞, $\epsilon = \frac{4}{255}$)	0.0429	0.1334	0.0212	0.00
PGD-20(l_∞, $\epsilon = \frac{8}{255}$)	0.0540	0.1542	0.0231	0.00
PGD-20(l_∞, $\epsilon = \frac{16}{255}$)	0.0978	0.1754	0.0295	0.00
PGD-20(l_∞, $\epsilon = \frac{32}{255}$)	0.1629	0.1908	0.0675	0.00
PGD-20(l_∞, $\epsilon = \frac{64}{255}$)	0.1909	0.1972	0.1786	0.00
PGD-20(l_2, $\epsilon = 0.25$)	0.0195	0.0170	0.0200	61.90
PGD-20(l_2, $\epsilon = 0.5$)	0.0212	0.0180	0.0201	33.10
PGD-20(l_2, $\epsilon = 1$)	0.0264	0.0364	0.0202	2.70
PGD-20(l_2, $\epsilon = 2$)	0.0382	0.0987	0.0204	0.00
			Nat Accuracy	68.50%

Table A.4: Detection accuracy evaluation under different targeted attacks on ImageNet.