AUGMENTING ROBUSTNESS USING ADVERSARIAL DETECTION

Bachelor's Project
June 2020

August Wester (201710314)
Rasmus Elias Thisted (201708242)

Advisor: Ira Assent

Abstract

Neural network-based machine learning models have consistently been shown to suffer from a vulnerability to *adversarial attacks* — inputs which are misclassified by a model despite being almost identical to ones that are correctly classified. In this paper, we explore the problem of *adversarial robustness*. We outline how adversarial attacks can be used to fool neural network-based image recognition models as well as how to mitigate this problem using defense mechanisms based on generative modelling. Finally, we propose an extension to the PuVAE defense architecture, which enables it to detect and discard adversarial examples. We find that this extension increases the model's robustness against FGSM and PGD attacks on simple datasets like MNIST and Fashion-MNIST. [1]

1 Introduction

In recent years, deep learning has proven to be an effective tool for an increasing number of tasks, such as autonomous driving [1], fraud detection [2] as well as speech and image recognition [3]. However, as the number of real-world applications increases, the security of these systems becomes a greater concern. In particular, Szegedy et al. [4] found that deep learning based systems are vulnerable to *adversarial attacks*; carefully engineered, yet very slight perturbations of an input that are intended to cause a system to give an incorrect output. These perturbations are often so slight that a human would hardly recognize them. Kurakin et al. [5] found that adversarial attacks translate to the physical world; that is, they showed adversarial attacks to be effective even after printing them and capturing them with a cell-phone camera. This poses many potential security threats to real-world applications of machine learning.

The existence of adversarial examples is an intriguing property of neural networks, and despite years of progress and proposed workarounds, it remains an open research problem. Solving this problem will, in addition to providing *adversarial robustness* and security, undoubtedly lead to a further understanding of the domain area and thus future improvements of deep learning systems.

In this paper, we explore recent research on adversarial robustness, which is mostly concerned with the field of computer vision, although adversarial examples have also been shown to exist for tasks within natural language processing [6] and reinforcement learning [7]. Based on our findings, we propose our own defense, which combines previous research by Hwang et al. [8] on using generative models to achieve adversarial robustness with research by Meng and Chen on detection of adversarial examples [9]. We find that our defense is able to improve on the robustness of both of these methods. However, like [8], the robustness comes at a significant expense of clean accuracy on more complex datasets. We discuss this in further detail.

[1] Code available at: https://colab.research.google.com/drive/1ZI4j51KLmQTB-OjKGqnvUEscfAsw4tHg

2 Background

2.1 Explaining adversarial vulnerability

Work by Szegedy et al. [4] demonstrated that neural networks are vulnerable to adversarial attacks. In an adversarial attack, an adversary with access to a trained neural network can exploit the differentiable nature of the model by using variants of gradient ascent to compute *adversarial examples*: Inputs which are misclassified by the network despite being almost identical to ones that are correctly classified.

The underlying cause of this vulnerability was later explored by Goodfellow et al. [10], who showed that neural networks are vulnerable to the same kinds of attack which fool linear models such as those used for logistic regression. Such an attack can be simply formulated in terms of an input $\boldsymbol{x}$ drawn from a model's target distribution and a perturbation $\boldsymbol{\eta}$ where $||\boldsymbol{\eta}|| \leq \boldsymbol{\epsilon}$ for some small constant $\boldsymbol{\epsilon}$. A corresponding adversarial example $\tilde{\boldsymbol{x}}$ can then be obtained by computing the sum of $\boldsymbol{x}$ and $\boldsymbol{\eta}$:

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \boldsymbol{\eta}.$$

By taking the dot product of $\tilde{\boldsymbol{x}}$ with some weight vector $\boldsymbol{w}$, we obtain the equality

$$\boldsymbol{w}^\top \tilde{\boldsymbol{x}} = \boldsymbol{w}^\top \boldsymbol{x} + \boldsymbol{w}^\top \boldsymbol{\eta}.$$

The last term on the right-hand side highlights how a vector $\boldsymbol{\eta}$ of small perturbations can incur a significant change to the dot product, assuming that its dimensionality is sufficiently large. This observation is in accordance with the work of Ford et al. [11], who found that high-dimensional data amplifies a system's vulnerability to adversarial attacks.

While this reasoning seems intuitive in the context of linear models, the fact that neural networks suffer from the same vulnerability is more surprising. After all, the power and versatility of neural networks largely stem from the fact that they incorporate *nonlinear* activation functions. However, as the authors noted, popular activation functions like ReLU already behave similarly to linear functions, while networks relying on more "classically nonlinear" functions like sigmoid have a natural tendency to stay within their almost-linear regions to avoid the problem of vanishing gradients. In other words, while neural networks certainly *are* nonlinear, they remain sufficiently linear to be vulnerable to the same kinds of attack that fool truly linear models [10].

2.2 Formalizing adversarial examples

Consider some $\boldsymbol{x}$ sampled from a distribution over $\mathbb{R}^n$. The set of valid adversarial examples is then parametrized by the choice of a constant $\boldsymbol{\epsilon}$ as well as a norm $\boldsymbol{\ell}$ (e.g. L_2, or L_∞), such that for a perturbation $\boldsymbol{\eta}$ added to $\boldsymbol{x}$, it holds that $||\boldsymbol{\eta}||_{\boldsymbol{\ell}} \leq \boldsymbol{\epsilon}$. The region around $\boldsymbol{x}$ for which this inequality holds is known in the literature as the $\boldsymbol{\epsilon}$-*ball* of $\boldsymbol{x}$. While L_2 bounds the maximum Euclidean distance of the perturbation, L_∞ bounds the maximum perturbation of each element in $\boldsymbol{\eta}$. During our experiments, we found no significant difference between attacks using L_2 and L_∞. In the following, we therefore choose to only evaluate our models using L_∞.

Naturally, one can easily construct an adversarial example which "fools" a model by simply increasing the size of $\boldsymbol{\epsilon}$ by some sufficient amount. However, this would quickly render the example unrecognizable to humans as well, thus making the notion of "fooling" the model misleading. The criterion is therefore for adversarial examples to remain recognizable to humans by keeping $\boldsymbol{\epsilon}$ at a relatively low value. In the case of images with pixel values in the range $[0, 1]$,

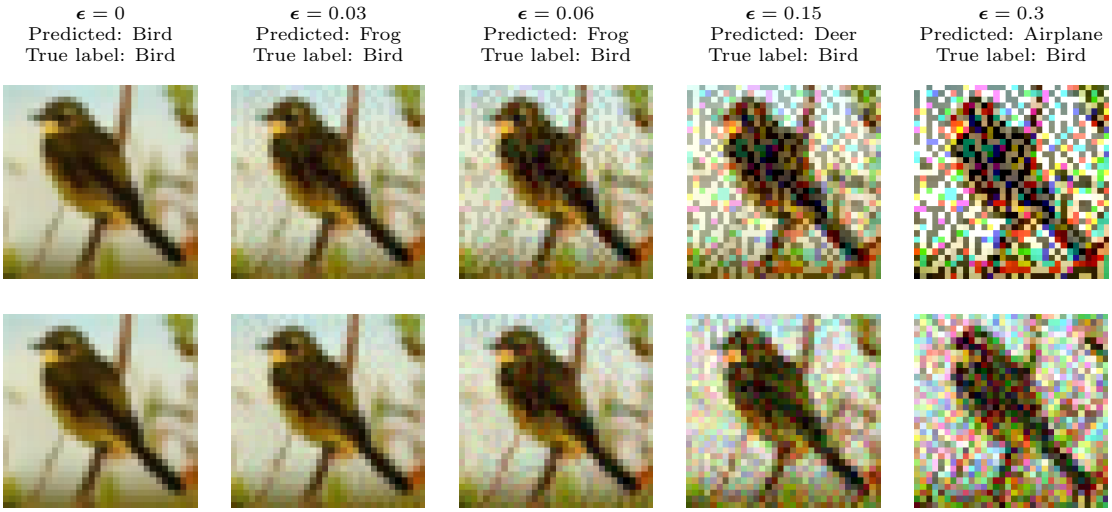


Figure 1: CIFAR-10 image perturbed using different values of ϵ. The images in the top and bottom row were obtained using FGSM and PGD, respectively. Note that during robustness evaluation, CIFAR-10 images are usually not perturbed by values of ϵ greater than 0.06.

the convention is to use 0.3 as the maximum ϵ for simple datasets like MNIST, and 0.03 to 0.06 for more complex datasets like CIFAR-10. These are not necessarily the values for which the images become unrecognizable to humans, but simply the ones for which most models start to fail. Figure 1 shows an image from the CIFAR-10 dataset subjected to attacks using different ϵ.

2.3 Types of attack

The existence of adversarial examples has led to the development of a variety of attacks designed to find them. Although these attacks vary in their specifics, almost all of them are fundamentally similar in that they attempt to increase the probability of misclassification by maximizing the loss of the model under attack. This is achieved by running variations of gradient ascent on the model's loss while remaining inside the ϵ-ball. We list three of the most common attacks below.

2.3.1 FGSM

Introduced by Goodfellow et al. [10], the "fast gradient sign method" or FGSM is a weak but efficient type of adversarial attack. It works by first computing the gradient $\nabla_{\boldsymbol{x}} J$ of the model's loss function J with respect to an input $\boldsymbol{x}$. It then proceeds to compute $\text{sign}(\nabla_{\boldsymbol{x}} J) \cdot \boldsymbol{\epsilon}$ after which it adds the result to $\boldsymbol{x}$ in order to obtain the adversarial example

$$\tilde{\boldsymbol{x}} = \boldsymbol{x} + \text{sign}(\nabla_{\boldsymbol{x}} J) \cdot \boldsymbol{\epsilon}.$$

Intuitively, it is similar to a single step of gradient ascent which always produces an adversarial example lying somewhere on the edge of the L_∞ ϵ-ball. This can be seen by noting that each pixel value will always be perturbed by either ϵ, 0 or $-\epsilon$.

2.3.2 Basic Iterative Method

The "basic iterative method" or BIM was introduced by Kurakin et al. [5] when they extended FGSM by running it for multiple iterations with a smaller step size $\boldsymbol{\alpha}$, where $\boldsymbol{\alpha} < \boldsymbol{\epsilon}$. At each

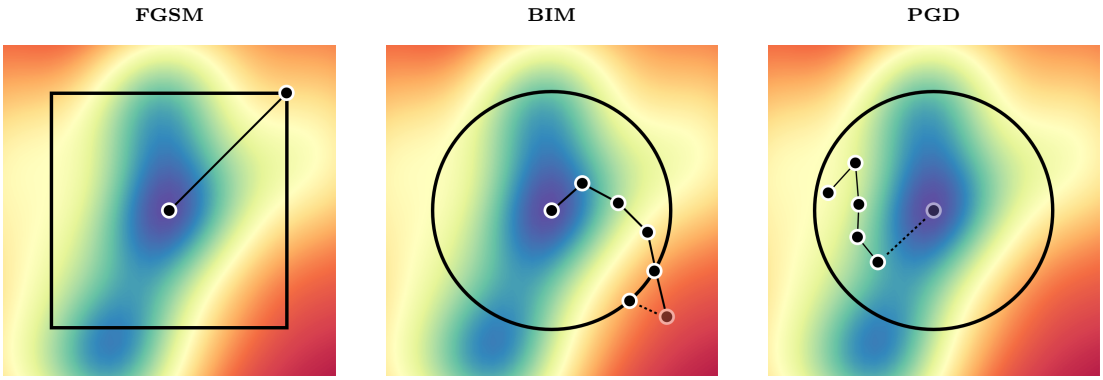


Figure 2: Left: FGSM takes a single step to produce an adversarial example lying on the edge of the L_∞ ϵ-ball. Middle: BIM iteratively explores an L_2 ϵ-ball and clips whenever it moves outside the region. Right: PGD is similar to BIM, but with random initialization. The heat map is a simplified representation of the value of the loss function at each point in input space, with reds representing higher values.

iteration, the pixel values are clipped to ensure that the adversarial example remains within the ϵ-ball of the original image as well as to maintain valid pixel values (in this case [0, 1]):

$$\tilde{\boldsymbol{x}}_0 = \boldsymbol{x}, \quad \tilde{\boldsymbol{x}}_{N+1} = Clip_{\boldsymbol{x},\epsilon}\{\tilde{\boldsymbol{x}}_N + \boldsymbol{\alpha}\ \text{sign}(\nabla_{\tilde{\boldsymbol{x}}_N} J)\}$$

$$Clip_{\boldsymbol{x},\epsilon}\{\boldsymbol{x}'\} = \min\{1, \boldsymbol{x} + \boldsymbol{\epsilon}, \max\{0, \boldsymbol{x} - \boldsymbol{\epsilon}, \boldsymbol{x}'\}\}$$

To run a BIM attack, an adversary needs to specify the number of iterations for which to run the algorithm. Since each iteration requires a forward pass through the target model, BIM is significantly slower than FGSM (which requires only a single forward pass). As opposed to FGSM, however, BIM attacks are usually less pronounced in terms of pixel perturbations and therefore harder for humans to detect. This can also be noted in Figure 1.

2.3.3 PGD

A variant of BIM known as "projected gradient descent" or PGD was proposed by Madry et al. in which $\tilde{\boldsymbol{x}}_0$ is a randomly perturbed image within the ϵ-ball of $\boldsymbol{x}$ [12]. The authors conjectured that PGD reasonably approximates an optimal gradient-based attack. To support this claim, the authors ran PGD on five examples from both the MNIST and CIFAR-10 datasets, each time starting from 10^5 uniformly random points within the ϵ-ball of the example, and then iterated until the loss plateaued. They found that all 10^5 loss values were highly concentrated with no significant outliers, suggesting that while several local maxima exist within the ϵ-ball, their loss values are very similar. From this finding they conjectured, that no other gradient-based adversary is likely to find a significantly better local maximum than PGD.

The difference between FGSM, BIM, and PGD are illustrated in Figure 2.

2.3.4 Black vs. white box attacks

FGSM, BIM, and PGD are all characterized as so-called *white box* attacks, given that they assume full access to the target model (e.g. an ability to compute gradients). This is contrasted

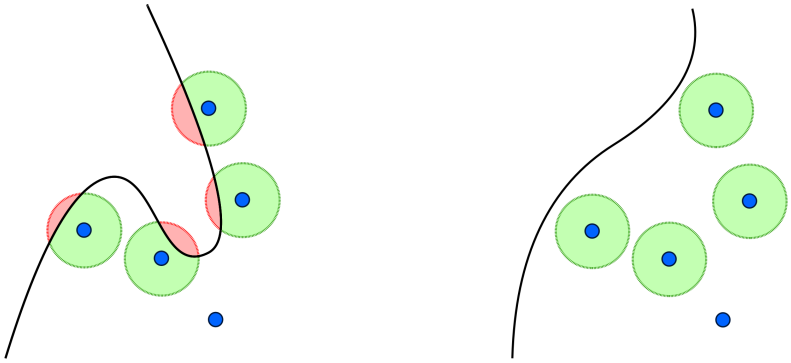


Figure 3: Left: The decision boundary of a classifier trained using unperturbed data will often overlap the ϵ-ball (highlighted in green) of clean examples, thus making the classifier vulnerable to adversarial attacks. Right: Using adversarial training acts as a regularizer on the model. This increases the margin to the decision boundary and makes the model more robust.

with *black box* attacks in which the adversary can only provide input to the model and knows nothing about its architecture (including any defensive measures). However, in the case of black box attacks, adversaries can take advantage of the observation by Papernot et al. that adversarial examples which fool one model typically fool other models as well [13]. This property is known as *transferability*, and is even observed between models that are trained with different data or have different architectures, as long as the models are trained for the same task (such as recognizing handwritten digits). This means that an attacker can train their own model and use it to compute adversarial examples which fool the target model with high probability.

2.4 Defenses

In the time since the discovery of adversarial attacks, finding techniques for mitigating the problem and achieving adversarial robustness has become an active area of research [14]. Yet, to our knowledge, the state of the art in robustness remains below 50% for datasets like CIFAR-10, meaning that more than 50 out of 100 adversarial examples generated by the strongest adversaries still fool even the most robust models. Below, we outline three common approaches to increasing a model's adversarial robustness.

2.4.1 Adversarial training

By including adversarial examples as part of the training data, Goodfellow et al. [10] showed an improvement in the robustness of neural network classifiers. This technique, known as *adversarial training*, was further advanced by Madry et al. [12] when they achieved impressive results by training with a PGD adversary. They found this to also provide significant robustness against other kinds of attacks such as FGSM and Carlini-Wagner [15].

The effectiveness of adversarial training is thought to be a result of an increased margin between clean examples in the input space and the decision boundary learned by the network. Intuitively, this should make sense, as placing additional examples within the ϵ-ball during training will put an outward pressure on the parts of the decision boundary overlapping the region. Figure 3 shows a simplified comparison between the decision boundary of a classifier trained on unperturbed images vs. an adversarially trained classifier.

2.4.2 Adversarial detection

Adversarial detection is the binary classification task of distinguishing between clean and adversarial examples, such that adversarial examples can be rejected or otherwise handled appropriately. For instance, Metzen et al. [16] trained an auxiliary detector network using the activations from the intermediate layers of a neural network classifier as input. From this, it infers whether the input to the classifier is adversarial, in which case the input is discarded. This inference is made possible by the fact that the sequence of a classifier's activations for adversarial examples often differs from that of clean ones [17]. They found this to work well against attacks similar to the ones they trained on, but it did not generalize well to other kinds of attack.

2.4.3 Preprocessing

Preprocessing techniques work by reconstructing clean inputs from adversarial examples before inputting them to a classifier. To do this, a generative model such as a generative adversarial network [18] or variational autoencoder [19] is usually employed in the early stages of the model. These are trained to approximate the distribution of clean examples in the training set, and can therefore be used to sample "purified" versions of the original input during inference. We present a more in-depth introduction to variational autoencoders in the following section.

A common property of preprocessing techniques is that they are *model-agnostic*, meaning that they can be trained once and used to provide robustness for several classifiers. This contrasts with adversarial training, which is *model-specific*, meaning that every model has to perform the computationally expensive task of adversarial training in order to achieve robustness [20].

2.5 Variational autoencoders

Variational autoencoders (VAEs) [19] are a common type of generative model that can be used to construct novel, artificial data, such as audio, text, and images. To achieve this, VAEs are trained to infer an approximation of a target distribution from which new data points can subsequently be sampled. If the approximation is good, the samples will look similar to ones drawn from the true distribution. As we will see in the following sections, this capability can be leveraged in interesting ways to provide defenses against adversarial attacks.

Generative models attempt to approximate the true underlying distribution of the dataset on which they are trained. In particular, VAEs do this by first making two fundamental assumptions about how a random sample $\boldsymbol{x}$ is generated:

1. There is a hidden continuous random variable $\boldsymbol{z}$ which is sampled from a prior distribution $p(\boldsymbol{z})$, usually assumed to be Gaussian.

2. There is an unknown conditional distribution $p(\boldsymbol{x}|\boldsymbol{z})$ from which $\boldsymbol{x}$ is sampled.

With these assumptions in mind, VAEs take the approach of inferring the value of $\boldsymbol{z}$ for a given sample $\boldsymbol{x}$ by modelling the distribution $p(\boldsymbol{z}|\boldsymbol{x})$ using an approximate posterior $q(\boldsymbol{z}|\boldsymbol{x})$. One way to ensure that $q(\boldsymbol{z}|\boldsymbol{x})$ is a good approximation is to minimize the Kullback-Leibler (KL) divergence between the two. However, directly minimizing the KL divergence turns out to be intractable given that it requires knowledge of $p(\boldsymbol{x})$.

Fortunately, the following equality can be derived from the KL divergence between $q(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{z}|\boldsymbol{x})$, and reveals a way to implicitly minimize it:

$$\begin{aligned}\log p(\boldsymbol{x}) &= \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x},\boldsymbol{z}) - \log q(\boldsymbol{z}|\boldsymbol{x})] + D_{KL}(q(\boldsymbol{z}|\boldsymbol{x}) \mid\mid p(\boldsymbol{z}|\boldsymbol{x})) \\ &= \mathcal{L} + D_{KL}(q(\boldsymbol{z}|\boldsymbol{x}) \mid\mid p(\boldsymbol{z}|\boldsymbol{x})),\end{aligned}$$

where $D_{KL}(q(\boldsymbol{z}|\boldsymbol{x}) \mid\mid p(\boldsymbol{z}|\boldsymbol{x}))$ denotes the KL divergence between $q(\boldsymbol{z}|\boldsymbol{x})$ and $p(\boldsymbol{z}|\boldsymbol{x})$.

There are two things to note in this equality: 1) $\log p(\boldsymbol{x})$ represents the log of the true probability of $\boldsymbol{x}$ and is therefore a constant, and 2) the minimum value of any KL divergence is 0. Together, these observations allow us to conclude that maximizing $\mathcal{L}$ is equivalent to minimizing $D_{KL}(q(\boldsymbol{z}|\boldsymbol{x}) \mid\mid p(\boldsymbol{z}|\boldsymbol{x}))$. Since $\mathcal{L} \leq \log p(\boldsymbol{x})$, $\mathcal{L}$ is known as the *variational lower bound*.

By expanding $\mathcal{L}$, we can obtain the following equality

$$\mathcal{L} = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})] - D_{KL}(q(\boldsymbol{z}|\boldsymbol{x}) \mid\mid p(\boldsymbol{z})), \tag{1}$$

which shows that maximizing $\mathcal{L}$ is equivalent to simultaneously maximizing the first term on the right-hand side and minimizing the second. VAEs maximize $\mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x})}[\log p(\boldsymbol{x}|\boldsymbol{z})]$ by using an approximate posterior $q(\boldsymbol{x}|\boldsymbol{z})$ trained to minimize the reconstruction loss between the input $\boldsymbol{x}$ and a reconstruction $\hat{\boldsymbol{x}}$ sampled from $q(\boldsymbol{x}|\boldsymbol{z})$. They minimize the KL divergence $D_{KL}(q(\boldsymbol{z}|\boldsymbol{x}) \mid\mid p(\boldsymbol{z}))$ directly using gradient descent, which is made possible by the initial assumption that $p(\boldsymbol{z})$ follows a Gaussian distribution.

In practice, $q(\boldsymbol{z}|\boldsymbol{x})$ and $q(\boldsymbol{x}|\boldsymbol{z})$ are implemented using two concatenated neural networks, termed the *encoder* and *decoder*, respectively. Given an input $\boldsymbol{x}$, the encoder deterministically computes the mean vector $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ (assumed diagonal) of the Gaussian distribution, with $\boldsymbol{z}$ then being obtained by using what the authors call the *reparameterization trick*:

$$\begin{aligned}\boldsymbol{z} &= \boldsymbol{\mu} + \boldsymbol{\Sigma}\boldsymbol{\epsilon} \\ \boldsymbol{\epsilon} &\sim \mathcal{N}(0, I)\end{aligned}$$

Note that in this case, $\boldsymbol{\epsilon}$ has no relation to the $\boldsymbol{\epsilon}$-ball of adversarial attacks, but is simply a random variable sampled from a standard multivariate Gaussian distribution. By letting the encoder compute $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ and keeping the random variable $\boldsymbol{\epsilon}$ fixed, the stochasticity does not affect the flow of backpropagation during training.

The result is a generative model with which it is not only possible to reconstruct the original input, but with which one can traverse the sample space of $\boldsymbol{z}$ and use the decoder to construct novel and entirely artificial samples that look similar to ones from the target distribution.

A common critique of VAEs is that their reconstructions tend to be somewhat blurry; a problem which other generative models (most notably generative adversarial networks [18]) do not suffer from to the same extent. According to Goodfellow et al., the reason behind this remains unknown, but is speculated by some researchers to be a result of assuming that $p(\boldsymbol{z})$ follows a Gaussian distribution [21]. In the following, we do not take this limitation into account, although we are intrigued by the idea that recent research into improving the sharpness of VAE reconstructions [22, 23] could potentially aid the development of more effective adversarial defense mechanisms. We consider this an interesting area of future work.

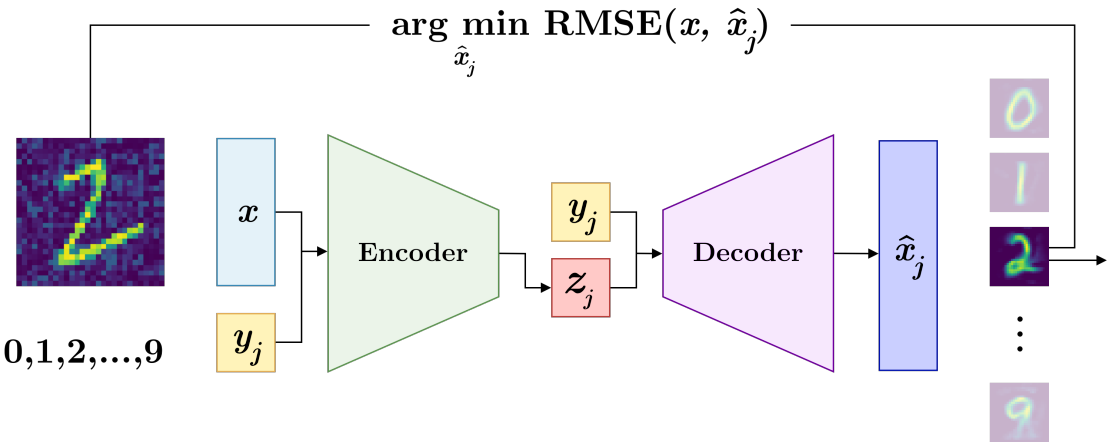


Figure 4: During inference, PuVAE reconstructs the input for each of the possible classes (10 in the case of MNIST) using a CVAE. The reconstruction which minimizes the RMSE with the original input is then fed to the classifier.

3 Related work

3.1 PuVAE

Recently, Hwang et al. proposed PuVAE [8], which attempts to purify adversarial examples by using a so-called *conditional variational autoencoder* or CVAE [24]. CVAEs differ slightly from traditional VAEs in that they incorporate a class label $\boldsymbol{y}$ alongside the usual input $\boldsymbol{x}$ to both the encoder and decoder. This changes the approximate posteriors from $q(\boldsymbol{z}|\boldsymbol{x})$ and $q(\boldsymbol{x}|\boldsymbol{z})$ to $q(\boldsymbol{z}|\boldsymbol{x},\boldsymbol{y})$ and $q(\boldsymbol{x}|\boldsymbol{z},\boldsymbol{y})$, thereby allowing the model to learn class-specific data distributions.

During training, each $\boldsymbol{x}_i$ from the training set is paired with its corresponding ground truth label $\boldsymbol{y}_i$ and the resulting reconstruction $\hat{\boldsymbol{x}}_i$ is input to a pretrained classifier trained on unperturbed data and with fixed parameters. Finally, the classifier outputs the predicted label $\hat{\boldsymbol{y}}_i$.

The loss of PuVAE is specified in such a way as to maximize the variational lower bound by minimizing the KL divergence in (1) as well as the reconstruction error between $\boldsymbol{x_i}$ and $\hat{\boldsymbol{x}}_{\boldsymbol{i}}$. Additionally, PuVAE also incorporates the cross-entropy loss of the classifier which increases the likelihood of the reconstructions being classified correctly.

At inference time, when no label is associated with the input $\boldsymbol{x}$ to the model, PuVAE creates a reconstruction $\hat{\boldsymbol{x}}_j : j = 1, ..., N$ for each of the classifier's N possible class labels. It then picks the reconstruction $\hat{\boldsymbol{x}}^*$ satisfying

$$\hat{\boldsymbol{x}}^* = \arg\min_{\hat{\boldsymbol{x}}_j} \text{RMSE}(\boldsymbol{x}, \hat{\boldsymbol{x}}_j),$$

where RMSE denotes the root mean squared error. $\hat{\boldsymbol{x}}^*$ is then passed on to the classifier, yielding the final prediction $\hat{\boldsymbol{y}}$. The PuVAE inference architecture is shown in Figure 4.

Since PuVAE is trained on clean data, its reconstructions will look similar to samples from the target distribution whether or not the input is adversarial. Additionally, by using a CVAE and

forcing it to reconstruct all N possible classes, PuVAE ensures that an adversarial input cannot fool the model into reconstructing a valid-looking image of an unrelated class. These properties enable PuVAE to display significant robustness against white box attacks on par with state of the art defenses like DefenseGAN [25]. It is important to note, however, that the efficacy of PuVAE relies heavily on the ability of the CVAE to make high-fidelity reconstructions. As a result, the defense may be less suited for more complex datasets.

3.2 MagNet

Meng and Chen proposed MagNet [9]; an adversarial defense architecture built on the idea that clean examples lie on a manifold occupying only a tiny subset of the full input space, and that adversarial examples are misclassified due to being located off the manifold. To exploit this characteristic, MagNet employs a two-pronged approach to adversarial defense, namely one or more *detectors* and a *reformer*.

The detectors are responsible for detecting adversarial examples, and are implemented using autoencoders (AEs) trained to reconstruct their own input using clean examples. Given that the detectors are trained to reconstruct data located *on* the manifold, the reconstruction error is generally higher when, during an attack, the input is *off* the manifold. A detector considers an input adversarial and discards it if its corresponding reconstruction error exceeds a predetermined threshold. If, during inference, this is the case for one or more of the detectors, the input is marked as adversarial and discarded. In the context of MagNet, a correct classification therefore satisfies exactly one of the following conditions:

1. The input $\boldsymbol{x}$ is clean and the model's prediction $\hat{\boldsymbol{y}}$ equals the ground truth $\boldsymbol{y}$.
2. The input $\tilde{\boldsymbol{x}}$ is adversarial and the model's prediction $\hat{\boldsymbol{y}}$ equals the ground truth $\boldsymbol{y}$ or the adversarial class label.

The goal of the reformer is to bring adversarial examples closer to the manifold of the clean examples such that the network can correctly classify attacks with small perturbations that go unnoticed by the detector. In practice, one of the detectors is selected to also act as the reformer.

While MagNet displays strong robustness against black box attacks, its performance degrades in the context of white box attacks where the adversary knows the parameters of the model and is able to compute gradients. This is due to the fact that the entire MagNet pipeline is differentiable, and as such, an adversary is able to straightforwardly maximize its loss. MagNet attempts to alleviate this issue by using what the authors call *diversity*: Training multiple autoencoders and picking one at random during inference. While this goes a long way to increase MagNet's white box robustness, the performance is still well below current state of the art techniques [8, 12, 25].

4 Proposed method

Building on the ideas of PuVAE and MagNet, we propose a method for detecting and discarding adversarial examples while keeping the classification of clean examples intact. Specifically, we propose a model consisting of a pretrained classifier, a PuVAE, a MagNet-style detector, and a new "adversarial" class label. Using the PuVAE, any input to the model will be reconstructed into each of the classifier's N possible classes after which the reconstruction minimizing the RMSE with the input is sent to the detector. The detector then discards the input if the RMSE

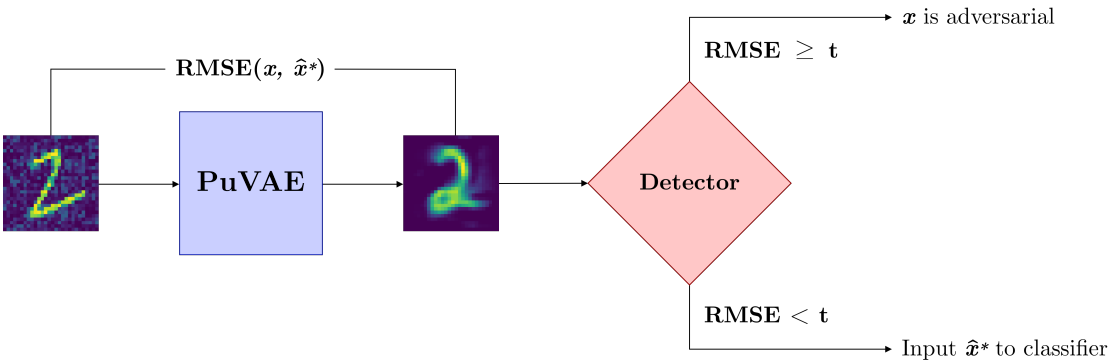


Figure 5: By extending PuVAE with a detector of adversarial examples similar to that of MagNet, the model is able to discard inputs for which the reconstruction error exceeds a specified threshold $\boldsymbol{t}$. Reconstructions that pass the detector stage are input to the classifier and classified normally.

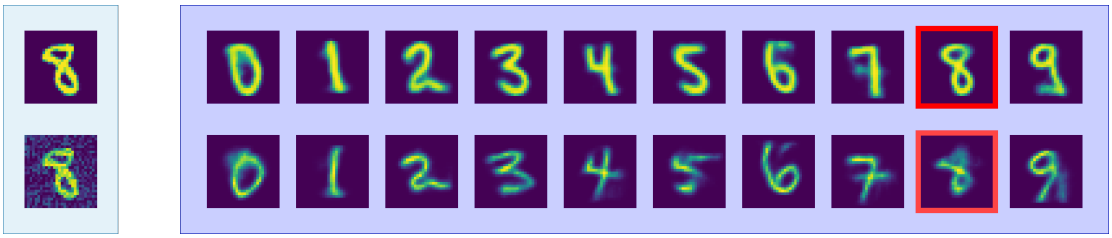


Figure 6: Left: A clean input (top) and an adversarial input (bottom) to PuVAE. Right: Reconstructions for each of the 10 possible class labels. The top row shows the reconstructions corresponding to the clean input, while the bottom row shows the reconstructions for the adversarial input. Notice in both cases how PuVAE is able to capture the slant of the handwritten "8". The reconstructions minimizing the RMSE with the input are highlighted in red.

between the best reconstruction and the input exceeds a predetermined threshold; otherwise it is input to the classifier. This architecture is illustrated in Figure 5.

The benefit of this approach over MagNet is a significantly increased robustness to white box attacks due to the use of PuVAE. Additionally, we find that for some attacks, the detector exerts a downward pressure on the optimal value of $\boldsymbol{\epsilon}$ due to the fact that high values of $\boldsymbol{\epsilon}$ usually produce attacks which are more easily detected. Notably, this downward pressure results in better reconstructions by PuVAE which are, in turn, more likely to be classified correctly.

We arrived at this method by first implementing a PuVAE using the recommended architecture and hyperparameters, and found that we were able to successfully reconstruct faithful representations of each of the classifier's N possible classes. These reconstructions are shown in Figure 6 for the MNIST dataset, both for a clean example and a corresponding adversarial example obtained by running PGD with $\boldsymbol{\epsilon} = 0.3$. After examining the RMSE between the original inputs and their corresponding best reconstruction, we found that the error was generally significantly higher for adversarial inputs compared to clean inputs. Concretely, the average RMSE between clean MNIST inputs and the reconstruction minimizing their RMSE was 0.04 while it was 0.13 for adversarial inputs with $\boldsymbol{\epsilon} = 0.3$. Similarly to Meng and Chen, we interpret this as evidence that the distance between the original input and the PuVAE reconstruction can be used as a measure of the likelihood of the input being adversarial [9].

5 Evaluation

In this section, we evaluate our proposed method against white box attacks on the MNIST, Fashion-MNIST and CIFAR-10 datasets. The architecture used for the underlying classifiers as well as the PuVAE hyperparameters follow the recommendations by Hwang et al. [8]. All architectures can be found in the appendix. The attacks were carried out by Foolbox, a Python library for robustness evaluation [26].

5.1 Threshold selection

We found that setting the threshold $\boldsymbol{t}$ appropriately is crucial to achieving satisfactory performance. Setting it too high will cause most adversarial inputs to go undetected while setting it too low will result in a high percentage of false positives. In the case of MNIST, we follow the approach of MagNet and choose a $\boldsymbol{t}$ for which the false positive rate in a validation set is less than or equal to 0.1%. For the more complex datasets, Fashion-MNIST and CIFAR-10, we allow for a higher false positive rate of 1% and 3%, respectively, since the average reconstruction error is significantly higher in this context. The specific performance penalties can be observed by noting the difference in clean accuracy between PuVAE and our method in Table 3-5.

5.2 Influence of threshold on attacks

In order for us to properly evaluate the efficacy of our method, we need to take into account how the value of $\boldsymbol{t}$ influences the optimal attack strategy. Given that attacks with large perturbations are easier to detect, we found that this put a downward pressure on the value of $\boldsymbol{\epsilon}$ for some but not all attacks. Initially, we believed that if we let $\boldsymbol{\delta}$ denote the average RMSE between clean inputs and their reconstructions, the optimal value of $\boldsymbol{\epsilon}$ would be $\boldsymbol{t} - \boldsymbol{\delta}$. The rationale behind this was that by staying within the boundary of such an $\boldsymbol{\epsilon}$-ball, the RMSE would, on average,

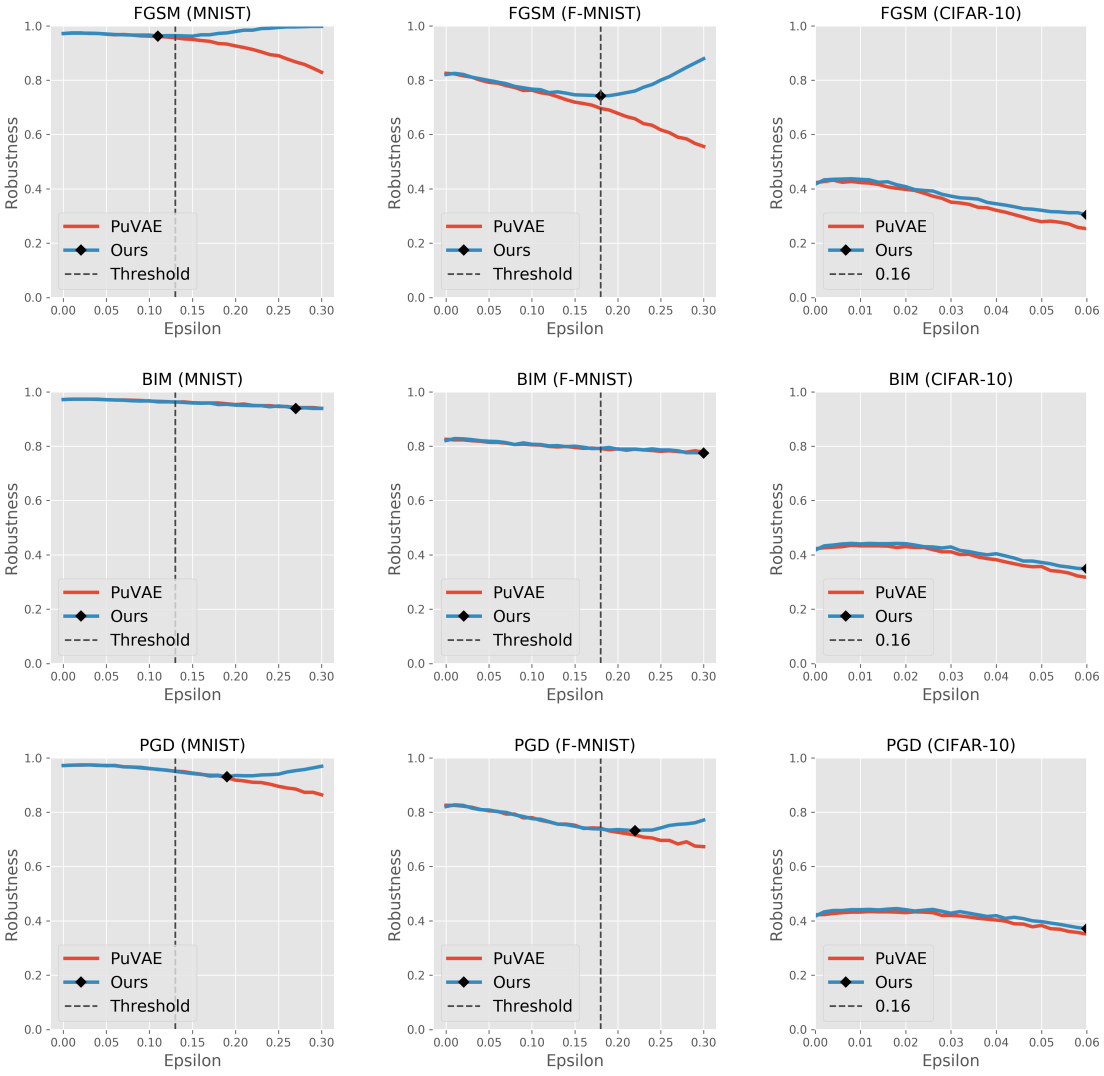


Figure 7: Robustness evaluation for PuVAE vs. our proposed method using 30 values of ϵ in the range [0.01, 0.3] (MNIST & Fashion-MNIST) and [0.002, 0.06] (CIFAR-10). For each dataset, the specific instance of PuVAE is identical with or without the detector. ♦ marks the optimal value of ϵ against our proposed method.

be less than t, and attacks would thus avoid detection.

We attempted to verify this empirically, but only found it to be somewhat the case for FGSM where the value of each pixel is reliably perturbed by ϵ. For BIM, increasing values of ϵ continued to reliably produce decreases in robustness, even after introducing the detector. We believe this is due to the perturbations of BIM being less homogenous than those of FGSM, which enables it to target and perturb individual pixels more than others. Finally, in the case of PGD, we found the optimal attack to use a value of ϵ slightly higher than t with robustness then sharply

increasing after that. We attribute this to PGD's random initialization excessively perturbing too many pixels and thereby "overshooting" a more selective perturbation. Figure 7 shows the robustness of PuVAE vs. our method for varying values of ϵ.

In the following, we make an effort to avoid misleading and overly optimistic results for $\epsilon = 0.3$ by taking into account the difference in worst-case attacks, and always making sure to include our method's worst-case performance on each.

5.3 Defense performance

With only one exception, our proposed method successfully manages to either match or increase the robustness of PuVAE across all datasets and attacks. The increase in robustness is most apparent in the case of FGSM and PGD attacks, while BIM attacks show a difference in robustness so small as to be negligible. As explained in the previous section, this difference in robustness is best explained by the threshold exerting a downward pressure on the optimal value of ϵ for attacks that have a more pronounced effect on a greater number of pixels. The benefit of this downward pressure is an upper bound on the optimal value of ϵ which, in turn, increases the quality of PuVAE's reconstructions and makes a correct classification more likely.

When applying our method to CIFAR-10, we see a sharp drop in both clean accuracy and robustness, which we believe is caused by the fact that the PuVAE reconstructions suffer from a significantly reduced fidelity in this context (see section D in the appendix for examples). Examining the reconstructions, it is obvious that the generative model's ability to capture the target distribution constitutes the primary obstacle to achieving both improved clean accuracy and robustness in the context of more complex datasets. One consequence of this w.r.t. our proposed method is that in order to maintain a sufficiently low false positive rate, the threshold is set much higher than the value of ϵ at which attacks start to become effective. This, in turn, means that most attacks easily pass the detector stage.

It is interesting to note that DefenseGAN, a similar preprocessing technique relying on a generative adversarial network (GAN) instead of a CVAE [25], in most cases does not exhibit robustness levels superior to those of PuVAE [8]. This is despite the fact that GANs are usually better at generating realistic-looking images [27]. We believe this is a hint that better generative models are needed to make preprocessing techniques like PuVAE and DefenseGAN scalable to more complex datasets.

Classifier	Dataset	Batch size	Epochs	Accuracy
A	MNIST	128	64	99.2%
A	F-MNIST	128	256	92.2%
B	CIFAR-10	64	350	80.2%

Table 1: Hyperparameters of classifiers trained on MNIST, Fashion-MNIST, and CIFAR-10.

PuVAE	Dataset	Batch size	Epochs	z dim.
C	MNIST	512	350	32
C	F-MNIST	512	350	128
D	CIFAR-10	256	1024	1024

Table 2: Hyperparameters of PuVAE trained on MNIST, Fashion-MNIST, and CIFAR-10.

Attack ($\epsilon = 0.3$)	No Defense	PuVAE	Ours	Ours (Worst)
No Attack	99.2	97.2	97.2	-
FGSM	23.2	82.9	99.8	96.2
BIM	0.2	93.9	93.9	93.9
PGD	0.0	86.4	96.9	93.0

Table 3: Robustness evaluation for MNIST using $\epsilon = 0.3$ and L_∞ norm (%)

Attack ($\epsilon = 0.3$)	No Defense	PuVAE	Ours	Ours (Worst)
No Attack	92.2	82.5	82.1	-
FGSM	25.4	55.6	87.9	74.2
BIM	20.9	77.8	77.5	77.5
PGD	1.6	67.3	77.1	73.2

Table 4: Robustness evaluation for Fashion-MNIST using $\epsilon = 0.3$ and L_∞ norm (%)

Attack ($\epsilon = 0.06$)	No Defense	PuVAE	Ours	Ours (Worst)
No Attack	80.2	42.3	41.7	-
FGSM	11.4	25.3	30.4	30.4
BIM	4.7	31.7	34.9	34.9
PGD	0.2	35.1	37.1	37.1

Table 5: Robustness evaluation for CIFAR-10 using $\epsilon = 0.06$ and L_∞ norm (%)

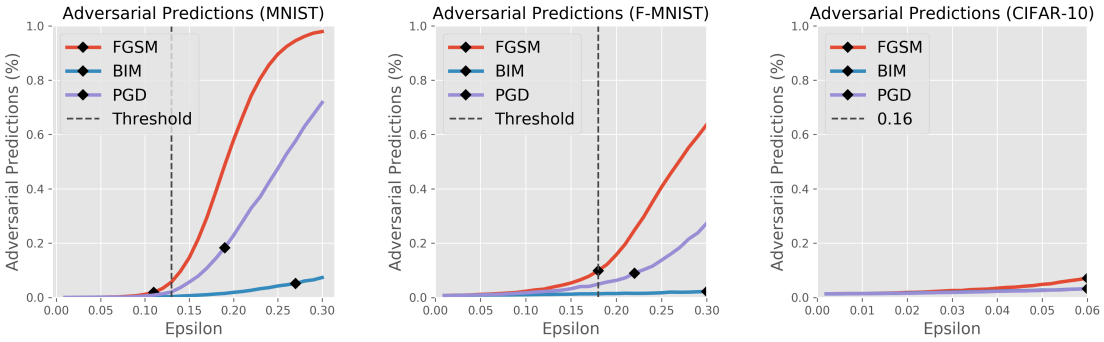


Figure 8: Proportion of predictions equal to the adversarial class label for FGSM, BIM, and PGD using 30 different values of ϵ in the range [0.01, 0.3] (MNIST & Fashion-MNIST) and [0.002, 0.06] (CIFAR-10). ♦ marks the optimal value of ϵ against our proposed method.

5.4 A note on adversarial detection

As we have mentioned, the benefit of using a detector is a decrease in the optimal value of ϵ for some attacks. However, we believe it is worth noting that there is an additional price to this benefit other than a small decrease in clean accuracy. Specifically, by introducing a detector and lowering the optimal value of ϵ, some attacks which would have otherwise resulted in a correct ground truth classification will instead be marked as adversarial and discarded. Figure 8 shows that in the case of MNIST, the percentage of attacks classified as adversarial for the optimal value of ϵ is between 1.8% and 18.3% depending on the type of attack. While this may not be a problem in many instances, it is important to emphasize the difference between *detecting* adversarial examples and the deeper problem of *correctly classifying* them. Since "true" adversarial robustness refers to the ability of a model to correctly classify the ground truth of an adversarial attack, augmenting robustness using adversarial detection can be seen as a way of side-stepping the actual problem. We do, however, still believe that adversarial detection can serve an important practical purpose in many real-world applications; for instance by enabling semi-autonomous systems to request human intervention when detecting an attack.

6 Conclusion

In this paper, we have explained the concept of adversarial vulnerability and robustness. We have outlined various techniques related to the defense of deep learning models, including pre-processing techniques using generative models as well as adversarial detection. We have also touched on some of the ways in which deep learning models can be attacked including the "fast gradient sign method" (FGSM) and "projected gradient descent" (PGD).

We have demonstrated how, against a small performance penalty, an adversarial detection mechanism inspired by the MagNet paper can be used to increase the robustness of the PuVAE pre-processing technique on simple datasets like MNIST and Fashion-MNIST against both FGSM and PGD attacks. We have argued that for some attacks, such a detection mechanism places an upper bound on the optimal value of ϵ, which in turn increases the fidelity of PuVAE's reconstructions when the bound is sufficiently low. We have argued that while detection mechanisms can be useful in this regard, they sidestep the deeper problem of true adversarial robustness by

discarding heavily perturbed examples rather than correctly predicting their ground truth label.

For complex datasets, we found that PuVAE incurred a dramatic decrease in clean accuracy. Unfortunately, this is not unusual among adversarial defense mechanisms, and it remains perhaps the most important factor prohibiting them from being truly useful at scale.

References

[1] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving, 2020.

[2] Raghavendra Chalapathy and Sanjay Chawla. Deep learning for anomaly detection: A survey, 2019.

[3] Mahbubul Alam, Manar D. Samad, Lasitha Vidyaratne, Alexander Glandon, and Khan M. Iftekharuddin. Survey on deep neural networks in speech and vision systems, 2019.

[4] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2013.

[5] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world, 2017.

[6] Moustafa Alzantot, Yash Sharma, Ahmed Elgohary, Bo-Jhang Ho, Mani B. Srivastava, and Kai-Wei Chang. Generating natural language adversarial examples, 2018.

[7] Sandy Huang, Nicolas Papernot, Ian Goodfellow, Yan Duan, and Pieter Abbeel. Adversarial attacks on neural network policies, 2017.

[8] Uiwon Hwang, Jaewoo Park, Hyemi Jang, Sungroh Yoon, and Nam Ik Cho. Puvae: A variational autoencoder to purify adversarial examples, 2019.

[9] Dongyu Meng and Hao Chen. Magnet: a two-pronged defense against adversarial examples, 2017.

[10] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples, 2014.

[11] Nic Ford, Justin Gilmer, Nicolas Carlini, and Dogus Cubuk. Adversarial examples are a natural consequence of test error in noise, 2019.

[12] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks, 2017.

[13] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: From phenomena to black-box attacks using adversarial samples, 2016.

[14] Rey Reza Wiyatno, Anqi Xu, Ousmane Dia, and Archy de Berker. Adversarial examples in modern machine learning: A review, 2019.

[15] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks, 2016.

[16] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations, 2017.

[17] Nicolas Papernot and Patrick McDaniel. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning, 2018.

[18] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes, 2013.

[20] Pratik Vaishnavi, Kevin Eykholt, Atul Prakash, and Amir Rahmati. Towards model-agnostic adversarial defenses using adversarially trained autoencoders, 2020.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[22] Lars Maaløe, Marco Fraccaro, Valentin Liévin, and Ole Winther. Biva: A very deep hierarchy of latent variables for generative modeling, 2019.

[23] Alex Lamb, Vincent Dumoulin, and Aaron Courville. Discriminative regularization for generative models, 2016.

[24] Carl Doersch. Tutorial on variational autoencoders, 2016.

[25] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models, 2018.

[26] Jonas Rauber, Wieland Brendel, and Matthias Bethge. Foolbox: A python toolbox to benchmark the robustness of machine learning models, 2018.

[27] Prateek Munjal, Akanksha Paul, and Narayanan C. Krishnan. Implicit discriminator in variational autoencoder, 2019.

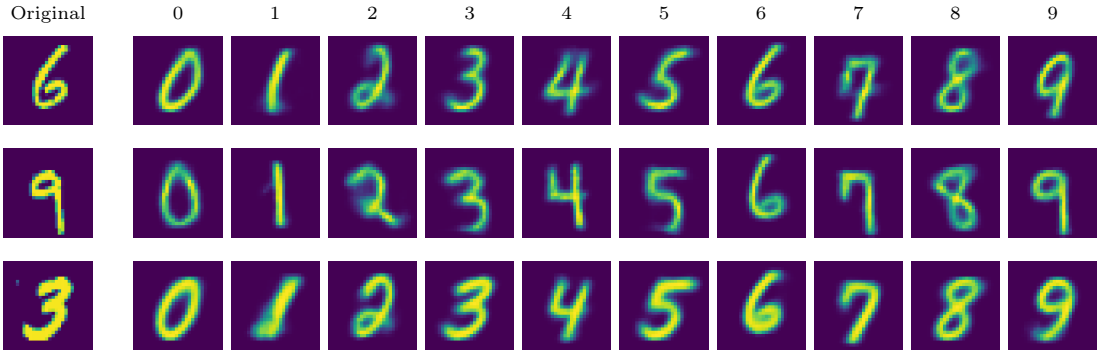
Appendix

A Architectures of Classifiers and PuVAEs

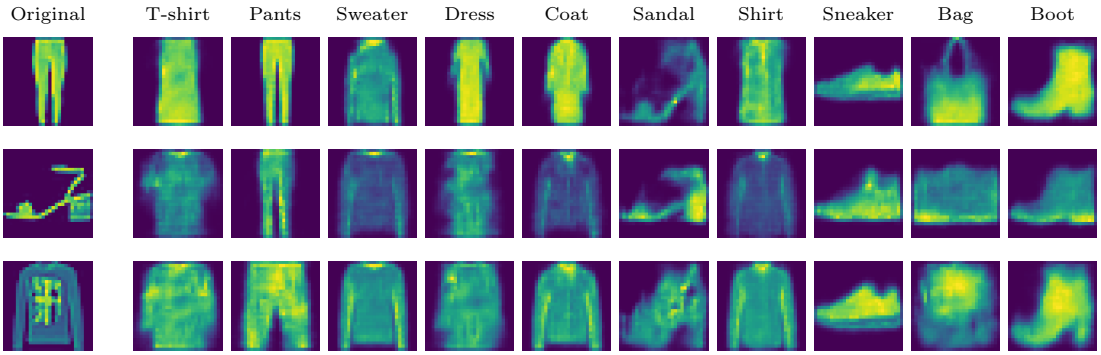
A	B	C	D
Conv(64, 5×5, 1, 1)	Conv(64, 3×3, 1, 1)	Conv(32, 7×7, 1, 2)	Conv(3, 2×2, 1, 1)
ReLU	ReLU	ReLU	ReLU
Conv(64, 5×5, 2, 1)	Conv(128, 3×3, 1, 1)	Conv(32, 7×7, 1, 2)	Conv(32, 2×2, 1, 2)
ReLU	ReLU	ReLU	ReLU
Dropout(0.25)	Conv(256, 3×3, 2, 1)	Conv(32, 7×7, 1, 2)	Conv(32, 2×2, 1, 2)
FC(128)	ReLU	ReLU	ReLU
ReLU	Dropout(0.25)	FC(1024)	Conv(32, 2×2, 1, 2)
Dropout(0.5)	FC(512)	ReLU	ReLU
FC(10)	ReLU	FC(1024)	FC(1024)
	Dropout(0.5)	ReLU	ReLU
	FC(256)	FC(32)	FC(1024)
	ReLU	ReLU	ReLU
	FC(10)	FC(32)	FC(1024)
		Softplus	Softplus
		Sampling	Sampling
		FC(512)	FC(1024)
		ReLU	ReLU
		Deconv(32, 7×7, 2, 1)	FC(8192)
		ReLU	ReLU
		Deconv(32, 7×7, 2, 1)	Deconv(32, 2×2, 1, 1)
		ReLU	ReLU
		Deconv(32, 7×7, 2, 1)	Deconv(32, 2×2, 1, 1)
		ReLU	ReLU
		Deconv(1, 3×3, 1, 1)	Deconv(32, 3×3, 2, 1)
		ReLU	ReLU
			Conv(3, 2×2, 1, 1)
			Sigmoid

Table 6: A: Classifier architecture for MNIST and Fashion-MNIST. B: Classifier architecture for CIFAR-10. C: PuVAE architecture for MNIST and Fashion-MNIST. D: PuVAE architecture for CIFAR-10. Conv(filters, kernel size, stride, dilation rate).

B Reconstructions for MNIST



C Reconstructions for Fashion-MNIST



D Reconstructions for CIFAR-10