Transferable Adversarial Robustness using Adversarially Trained Autoencoders

Pratik Vaishnavi¹, Kevin Eykholt², Atul Prakash², Amir Rahmati¹
¹Stony Brook University  ²University of Michigan
pvaishnavi@cs.stonybrook.edu, {keykholt,aprákash}@umich.edu, amir@rahmati.com

Abstract

Machine learning has proven to be an extremely useful tool for solving complex problems in many application domains. This prevalence makes it an attractive target for malicious actors. Adversarial machine learning is a well-studied field of research in which an adversary seeks to cause predictable errors in a machine learning algorithm through careful manipulation of the input. In response, numerous techniques have been proposed to harden machine learning algorithms and mitigate the effect of adversarial attacks. Of these techniques, adversarial training, which augments the training data with adversarial inputs, has proven to be an effective defensive technique. However, adversarial training is computationally expensive and the improvements in adversarial performance are limited to a single model. In this paper, we propose Adversarially-Trained Autoencoder Augmentation, the first transferable adversarial defense that is robust to certain adaptive adversaries. We disentangle adversarial robustness from the classifier architecture, in which a denoising autoencoder is used for improving adversarial robustness. We re-examine data preprocessing adversarial defenses and focus on the stacked autoencoder (AAA), a model agnostic technique for improving adversarial robustness. We examine AAA to achieve comparable results to state-of-the-art adversarially trained models on the MNIST, Fashion-MNIST, and CIFAR-10 datasets. Furthermore, we can transfer our approach to other vulnerable models and improve their adversarial performance without additional training. Finally, we combine our defense with ensemble methods and parallelize adversarial training across multiple vulnerable pre-trained models. In a single adversarial training session, the autoencoder can achieve adversarial performance on the vulnerable models that is comparable or better than standard adversarial training.

Introduction

Machine learning algorithms are becoming the preferred tool to empower systems across multiple applications domains ranging from automatically monitoring employee hygiene and safety to influencing control decisions in self-driving cars and trading. With such pervasive use, it is critical to understand and address the vulnerabilities associated with machine learning algorithms so as to mitigate the risks in real systems. Adversarial machine learning attacks are one class of such vulnerabilities in which an attacker can reliably induce predictable errors in machine learning systems. At a high-level, given a model and a correctly labelled input, an adversarial attack computes the necessary modifications on the input such that the model incorrectly labels the input, while ensuring that the modifications are inconspicuous (e.g., imperceptible to a human observer). In an effort to mitigate or prevent the effect of adversarial attacks, multiple defensive techniques have been proposed. Of them, the most promising technique, adversarial training, uses a basic data augmentation strategy to improve a model’s performance in adversarial scenarios (Madry et al. 2018). During training, adversarial examples are computed on-the-fly and added to the training data. This approach has proven to greatly improve the prediction accuracy on adversarial inputs for certain types of adversarial attacks. However, adversarial training requires retraining the classifier on adversarial data and introduces significant performance overhead during training as adversarial examples must be generated during each training step. Modifications to the original approach have been made to reduce the performance overhead on larger datasets such as Imagenet by weakening the adversarial attack algorithm used at training time and diversifying the models used to generate adversarial inputs (Kurakin, Goodfellow, and Bengio 2017; Tramèr et al. 2018). Despite these modifications, there still remains a problem, which is that the benefits of adversarial training only extend to a single model. In order to create additional adversarially robust models, adversarial training procedure must be repeated for each model.

In this work, we propose Adversarially-Trained Autoencoder Augmentation (AAA), a model agnostic technique for improving adversarial robustness. We re-examine data pre-processing adversarial defenses and focus on the stacked classifier architecture, in which a denoising autoencoder is used...
to mitigate the effect of adversarial inputs (Gu and Rigazio 2015; Liao et al. 2018; Meng and Chen 2017). Traditionally, autoencoder based defenses are trained on a static set of adversarial samples generated against a naturally trained classifier. However, these defenses are not robust to an adaptive adversary that is aware of the defense in place. Our approach differs from previous work in that we train the autoencoder against an adaptive adversary using both a supervised and an unsupervised objective, which enhance classifier performance and encourage transferability. Our approach provides comparable performance on Fashion-MNIST and CIFAR-10 datasets, and outperforms classic adversarial training on MNIST by 1.5%. Furthermore, as we’ve designed AAA to be model agnostic, we transfer the adversarially trained autoencoder to different naturally trained classifiers and improve their adversarial accuracy by 91.62% and 48.95% for MNIST and Fashion-MNIST datasets, respectively. Finally, we find that on CIFAR-10, our initial approach does not completely enable the transferability of our pipeline. However, by utilizing ensemble adversarial training, AAA can parallelize adversarial training across multiple classifiers and be partially transferable. In fewer training iterations per model, Adversarially-Trained Autoencoder Augmentation can achieve adversarial performance comparable to an adversarially trained model in the worst case. In the best case, using our pipeline with ensemble training improved a model’s natural and adversarial accuracy by 9.17% compared to normal adversarial training.

Our Contributions.

- We propose Adversarially-Trained Autoencoder Augmentation, the first transferable adversarial defense robust to an adaptive $L_{\infty}$ adversary. Through adversarial training of an autoencoder, we disentangle adversarial robustness and classification enabling us to transfer adversarial robustness improvements across multiple classifiers.

- On MNIST and Fashion-MNIST, Adversarially-Trained Autoencoder Augmentation results in comparable or better performance than traditional adversarial training, while being completely transferable, improving the adversarial accuracy of a different classifier by 91.62% and 48.95% respectively.

- On CIFAR-10, despite comparable performance to adversarial training on the training classifier, our approach reported weaker transferability on a different classifier. Thus, we extended our approach to use ensemble techniques for partial transferability. In doing so, we found that the resulting autoencoder achieved comparable or better adversarial accuracy to individually adversarially training each classifier (9.17% improvement in adversarial accuracy in the best case) and with less overall training.

Background

Adversarial Attacks. Adversarial examples were introduced by Szegedy et al. They observed that by maximizing the prediction error of a classifier for a given input, it was possible to learn imperceptible perturbations to add to the input, characterized by an $L_2$ distance, which cause it to be misclassified (Szegedy et al. 2014). Since then, numerous adversarial attacks have been developed which can be classified based on the adversary’s knowledge of the model. White-box attacks assume the adversary has perfect knowledge of the model parameters. Using this knowledge, an adversary can use back-propagation to precisely compute the necessary adversarial modifications. E.g., the Jacobian-Based Saliency Map (JSMA) attack uses back-propagation to create adversarial saliency maps, which measure the impact of each input feature on the output decision (Papernot et al. 2016a). The Fast-Gradient Sign method (FGSM) modifies the entire input based on the model gradients with respect to the input (Goodfellow, Shlens, and Szegedy 2014; Kurakin, Goodfellow, and Bengio 2016).

Black-box attacks assume the adversary only has the ability to query a model for its soft (probability distribution) or hard (predicted label) output. In these attacks, the adversary queries the model and uses the output to estimate gradient information, which can then be used to re-enact a white-box attack (Naroditska and Kasiviswanathan 2017; Chen et al. 2017; Bhagoji et al. 2018; Cheng et al. 2018). Alternatively, it is known that adversarial inputs are transferable. An adversarial input created to cause misclassification errors for one model can be reused to cause misclassification errors in other models despite differences in the model architectures (Szegedy et al. 2014). Using this property, an adversary creates adversarial inputs using a white-box attack on a model they have white-box access to, and then exposes the created adversarial inputs to the black-box model (Papernot, Mc丹尼尔, and Goodfellow 2016; Liu et al. 2016).

Adversarial Defenses. Given the widespread use of machine learning, successful adversarial attacks against deployed systems could result in dire real-world consequences. As such, it is critical to develop techniques to mitigate the effect of adversarial attacks. Early adversarial defenses such as defensive distillation (Papernot et al. 2016b), input transformations (Guo et al. 2018), and defensive DNNs (Song et al. 2018; Samangouei, Kabkab, and Chellappa 2018) are techniques, which rely on masking or breaking the gradient used in white-box attacks to generate adversarial examples. However, many of these early defenses have been broken as an adaptive adversary can perform end-to-end attacks by approximating the gradient through these defenses (Athalye, Carlini, and Wagner 2018).

Currently, adversarial training is recognized as the state-of-the-art technique to improve a model’s adversarial robustness to white-box projected gradient descent (PGD) attacks. Adversarial training improves a model’s robustness to adversarial examples by generating them on-the-fly to be used during training (Szegedy et al. 2014). More specifically, in each training iteration adversarial examples that maximize the model’s loss are generated iteratively (Madry et al. 2018). Through adversarial training, Madry et al. created MNIST and CIFAR classifiers with significantly improved adversarial robustness. Later, due to the poor scalability of the original approach, the single-step FGSM attack was used to reduce the performance overhead of adversarial training for large datasets (Kurakin, Goodfellow, and Bengio 2017). The performance overhead was further reduced when ensemble techniques were applied to adversarial training (Tramèr et al. 2018). In addition, Tramèr et al. revealed that adversarial training results in overfitting the model to the generated adversarial examples. After training, the adversarially trained model remained susceptible to transferable adversarial examples. Thus, ensemble adversarial training served a second purpose to improve the black-box adversarial robustness of the trained model.

Adversarial Autoencoder Defenses. As adversarial examples
are typically generated by adding noise to a correctly classified input, it is natural to attempt to use denoising algorithms to remove the adversarial noise. Gu and Rigazio explored using denoising autoencoders (AE) when pre-processing the input to defend against adversarial inputs (Gu and Rigazio 2015). Given a deep neural network classifier, they generated adversarial examples, and used them to train an AE that mapped the adversarial examples back to the original inputs based on a reconstruction loss. By stacking the AE and the classifier, the success rate of adversarial examples generated against the original classifier dropped significantly. However, as adversarial inputs are generated with respect to the classifier rather than the full pipeline, adaptive adversarial attacks remain possible. A recent work proposed two modifications to stacked architecture defense (Liao et al. 2018). First, the AE was changed to output inverse adversarial noise to correct the modified input rather than fully reconstruct the original input. This change was based on the hypothesis that learning the adversarial noise added to an input is an easier task than learning to reconstruct the original input. Second, they replaced the pixel-based reconstruction loss with a loss based on the hidden layers of the target classifier they are defending. These two changes resulted in training an AE that removes noise from the input such that the error amplification effect from adversarial modifications is mitigated. This work, though, still generates adversarial inputs with respect to the classifier rather than the full pipeline.

Xie et al. take a different approach choosing to incorporate denoising as part of the network architecture rather than as a pre-processing step (Xie et al. 2019). They characterize the error amplification effect as significant noise in the feature maps of the network. When examining the features maps of adversarial examples, they observe that semantically uninformative portions of the input had higher than normal feature map activation. Thus, they add denoising blocks in between intermediate layers in the network to suppress the noise in the feature maps and refocus the network on semantically informative information. On Imagenet, they adversarially trained a ResNet model containing four denoising blocks and demonstrated improved black-box and white-box adversarial robustness compared to existing works. Although their approach is interesting and helps explain the mechanisms of adversarial attacks, it remains model dependent. Extra denoising layers must added to a model’s architecture in order to achieve adversarial robustness, whereas our approach is designed to be model agnostic.

Adversarially-Trained Autoencoder Augmentation

Adversarial training in its current setting has a major drawback: it introduces significant computational overhead on the training process, hence making it orders of magnitude slower than natural training. Furthermore, an adversarially trained model cannot improve the performance of other vulnerable models for the task. A string of works in recent literature (Shafahi et al. 2019; Zhang et al. 2019) have studied methods to reduce the overhead of adversarial training. We propose Adversarially-Trained Autoencoder Augmentation (AAA), which creates a separate adversarially robust component in the classification pipeline. We design our approach to be transferable so that a single adversarial training session is sufficient to improve the adversarial robustness of multiple vulnerable models trained on the same dataset.

In their work on MNIST adversarial training, Madry et al. observed that only 3 of the 32 filters in the first convolutional layer of the adversarially trained model contribute to the input to the second layer (Madry et al. 2018). They characterized the behavior of these three filters as thresholding filters, which remove most of the adversarial modifications from the input. From their observations, we re-define an adversarially trained model as a two step process: 1) Denoising; 2) Classification. Thus, by disentangling the first step from the model, we can create a model agnostic defense, which is transferable across all vulnerable classifiers trained on a similar data distribution.

Design

To implement our model-agnostic defense, we use an autoencoder (AE) $G_{\phi}(x) = x'$, which consists of an encoder that projects a $n$-dimensional input $x \in [0,1]^n$ to a $d$-dimensional latent space $\mathbb{R}^d$, where $d \ll n$, and a decoder that projects the latent feature vector back into the original input space. The reconstructed input $x'$ is given to a classifier $F_\theta(x')$, which outputs a label $y$. Given a trained classifier $F_\theta$, we train $G_{\phi}$ as follows:

$$\argmin_{\phi} \mathbb{E}_{(x,y) \sim \mathcal{D}} \left[ \max_{\delta \in \mathbb{S}} \mathbb{L}_{\text{adv}}(F_\theta(G_{\phi}(x+\delta)), y) \right]$$

(1)

where $\mathbb{L}_{\text{adv}}$ is the cross entropy loss, $\mathbb{S}$ is the set of allowable adversarial perturbations on $x$, $\mathbb{L}_{\text{AE}}$ is the loss of the AE.

The mean squared error, $\mathbb{L}_{\text{mse}}$, is the traditional loss function used when training AEs as it encourages the AE to map noisy inputs back to the clean data manifold. However, such an objective does not encourage the AE to minimize classification risk. As such, it remains possible that a well-trained AE will map a noisy input to an adversarial input. Thus, we adversarially train our AE with respect to three different loss functions. $\mathbb{L}_{\text{ent}}$ is a cross-entropy loss, $\mathbb{L}_{\text{adv}}$, which encourages the AE towards reconstructions that minimize the classification risk, which provides robustness to adversarial examples. $\mathbb{L}_{\text{mse}}$, the mean squared error, encourages the AE towards reconstructions that lie on the natural data manifold, which improves the transferability of our adversarial defense. Finally, $\mathbb{L}_{\text{ent}} + \mathbb{L}_{\text{mse}}$ takes into account the benefits of both loss functions resulting in reconstructions on the natural data manifold that minimize classification risk.

Evaluation

Experimental Setup

For our experiments, we evaluated adversarially-trained autoencoder augmentation (AAA) on three different datasets: MNIST, Fashion-MNIST, and CIFAR-10 (LeCun et al. 1998; Xiao, Rasul, and Vollgraf 2017; Krizhevsky 2009). As described previously, our defense pipeline consists of the adversarially trained AE and a pre-trained classifier. In all of the experiments using AAA, the classifier used was naturally trained.

For our MNIST experiments, we created a simple convolutional AE consisting of two convolution layers, two fully connected layers, and two deconvolution layers. For the classifier, we used the classifier architecture provided by Madry.
et al.\(^1\). We evaluate transferability using a classifier comprised of two fully connected layers.

For Fashion-MNIST, we use a convolutional AE consisting of 15 convolution layers, two max-pooling layers, and two upsampling layers. For the classifier, we add two convolution layers to the MNIST classifier architecture. We evaluate transferability using a classifier comprised of four fully connected layers.

For CIFAR-10, we used a U-Net AE architecture (Ronneberger, Fischer, and Brox 2016). The main difference between a U-Net AE and a standard convolutional AE is the use of skip connections, which are forward feed connections between the encoding and decoding layers in the network that enable higher fidelity reconstructions. Our U-Net AE uses five encoding convolution layers and four decoding convolution layers. We also use three different classifier architectures. For the first architecture, we use the ResNet architecture provided by Madry et al.\(^2\). For the second model, we used the VGG-19 classifier (Zisserman 2015). For the last model, we created a simple DNN consisting of four convolution layers and a fully connected layer. Additional details regarding the models can be found in the supplementary material.

Training Details We used an Adam optimizer and adversarially trained the AE using the three loss functions mentioned in the previous section. For MNIST and Fashion-MNIST, adversarial examples were generated in each training iteration using a 40-step \(L_\infty\) bounded PGD attack with a step size of 0.01 and \(\epsilon = 0.3\) and \(\epsilon = 0.2\) respectively. For CIFAR-10, adversarial examples were generated in each training iteration using a 10-step \(L_\infty\) bounded PGD attack with a step size of \(\frac{2}{255}\) and \(\epsilon = \frac{\sqrt{2}}{255}\). For all experiments, we set the initial learning rate at 0.001 and decreased it by half if the validation loss did not improve over five epochs.

MNIST Results

We first present our experimental results on the MNIST dataset, which is a grayscale dataset composed of 28x28 sized images of handwritten digits. We evaluate all models against a white-box 40-step and 200-step \(L_\infty\) bounded PGD attack with step size of 0.01 and \(\epsilon = 0.3\). We compare the performance of an adversarially trained model to our approach in Table 1.

| Model                     | Natural Accuracy | Adversarial Accuracy |
|---------------------------|------------------|----------------------|
|                           | 40-step          | 200-step             |
| Nat. Training             | 99.24%           | 0%                   |
| Adv. Training             | 99.10%           | 93.85%               |
| AAA \(L_{xent}\)          | 99.08%           | 95.19%               |
| AAA \(L_{mse}\)           | 99.17%           | 94.29%               |
| AAA \(L_{xent}+L_{mse}\)  | 99.15%           | 95.27%               |

| Model                     | Natural Accuracy | Adversarial Accuracy |
|---------------------------|------------------|----------------------|
|                           | 40-step          | 200-step             |
| FC-Classifier             | 98.00%           | 0%                   |
| AAA \(L_{xent}\)          | 28.75%           | 17.38%               |
| AAA \(L_{mse}\)           | 98.21%           | 88.90%               |
| AAA \(L_{xent}+L_{mse}\)  | 98.12%           | 91.62%               |

Table 2: AAA transferability results on MNIST using a fully connected classifier.

We observe that for all three loss functions, AAA minimally impacts the natural accuracy of the naturally trained classifier, while also improving the adversarial accuracy significantly. Interestingly, we note \(L_{xent}\) only slightly outperforms \(L_{mse}\) despite \(L_{mse}\) having no relation to the classifier objective. This behavior is likely due to the simple nature of the MNIST dataset which allows for the learning of class-distinctive features using an unsupervised objective like \(L_{mse}\) only. Finally, compared to adversarially training the classifier only, AAA has consistently higher adversarial accuracy despite the fact that AAA relies on naturally trained classifiers.

Adversarial training is a non-transferable technique to improve the adversarial performance of a model. That is to say, a pre-trained adversarially trained model cannot be reused to improve the adversarial performance of a different model. In Table 2, we measure the transferability of AAA on a pre-trained fully connected classifier. For these results, we reuse the AE trained on the naturally trained Madry et al. classifier architecture.

| Model                     | Natural Accuracy | Adversarial Accuracy |
|---------------------------|------------------|----------------------|
|                           | 40-step          | 200-step             |
| AAA \(L_{xent}\)          | 28.75%           | 17.38%               |
| AAA \(L_{mse}\)           | 98.21%           | 88.90%               |
| AAA \(L_{xent}+L_{mse}\)  | 98.12%           | 91.62%               |

Table 2: AAA transferability results on MNIST using a fully connected classifier.

Depending on the loss function used during training, the transferability of our technique varies. As we expected, \(AAA L_{xent}\) has higher transferability than \(AAA L_{mse}\) as it seeks to project the input onto the natural data manifold. As such, its reconstructions are more likely to generalize across multiple models as Table 3 illustrates. \(AAA L_{xent}\) encourages the reconstructions to reduce the classification risk with respect to the training model, which may imply that the AE overfits based on the visualizations in Table 3. \(AAA L_{xent}+L_{mse}\), which includes a classification loss, has better transferability than \(AAA L_{mse}\). This improvement can be attributed to the idea that classifiers learn similar decision boundaries with respect to the natural data (Goodfellow, Shlens, and Szegedy 2014).

Fashion-MNIST Results

Fashion-MNIST is a grayscale dataset that contains 28x28 sized images of ten different types of clothes. We evaluate all models against a 40-step and 200-step \(L_\infty\) bounded PGD attack with step size of 0.01 and \(\epsilon = 0.2\). As before, we first compare the performance of an adversarially trained model to AAA in Table 4.

AAA improves the adversarial accuracy of the natural classifier with minimal impact on the natural accuracy. However, unlike in MNIST, \(AAA L_{mse}\) has significantly worse performance compared to \(AAA L_{xent}\), suggesting that some form of classifier loss must be used to train an adversarially robust AE. Finally, we note that \(AAA L_{xent}\) and \(AAA L_{xent}+L_{mse}\) have comparable performance to adversarial training.

---

1https://github.com/MadryLab/mnist_challenge
2https://github.com/MadryLab/cifar10_challenge
3We scale the pixel values to the range \([0, 1]\]
Table 3: We visualize the output of the AE for MNIST, Fashion-MNIST, and CIFAR-10 (ResNet). The first column contains natural images. The other columns show the output of $AAA_{L_{xent}}$, $AAA_{L_{mse}}$, and $AAA_{L_{xent}+L_{mse}}$ respectively.

Table 4: Performance comparison of AAA to the adversarial training approach proposed by Madry et al. (2018) on Fashion-MNIST.

Table 5: AAA transferability results on Fashion-MNIST using a fully connected classifier.

CIFAR-10 Results

CIFAR-10 contains 32x32 sized color images belonging to one of ten possible class labels. We evaluate all models against a 10-step $L_\infty$ bounded PGD attack with step size of $\epsilon = \frac{8}{255}$ and
\( \epsilon = \frac{2}{255} \). In Table 6, we compare the performance of AAA to adversarial training using the ResNet classifier.

| Model   | Natural Accuracy | Adversarial Accuracy |
|---------|------------------|----------------------|
| Nat. Training | 89.67% | 2.13% |
| Adv. Training  | 81.49% | 47.79% |
| AAA_{\text{L}_{\text{exc}}+L_{\text{mse}}} | 77.21% | 47.13% |
| AAA_{\text{L}_{\text{mse}}} | 86.07% | 0.08% |
| AAA_{\text{L}_{\text{exc}}+L_{\text{mse}}} | 80.06% | 47.17% |

Table 6: Performance comparison of AAA to the adversarial training approach proposed by Madry et al. (2018) on CIFAR-10.

\( \text{L}_{\text{exc}} \) and \( \text{L}_{\text{exc}}+\text{L}_{\text{mse}} \) both result in adversarial accuracy improvements comparable to adversarial training, with a similar reduction in natural accuracy. Also, we note again that although \( \text{L}_{\text{mse}} \) preserves the natural accuracy of the classifier, it does not have any impact on adversarial accuracy. Based on our results on MNIST, Fashion-MNIST, and CIFAR-10, we see a trend that as dataset complexity increases, \( \text{L}_{\text{mse}} \) performs increasing worse with respect to mitigating the effect of adversarial attacks.

Next, we augment the naturally trained VGG-19 and DNN classifiers with the AE trained on ResNet. In Table 7, we present the transferability of AAA with respect to these two classifiers. As before, we see that AAA_{\text{L}_{\text{exc}}} is not transferable and AAA_{\text{L}_{\text{mse}}} has only a minimal improvement on adversarial accuracy. However, unlike in our previous experiments, AAA_{\text{L}_{\text{exc}}+L_{\text{mse}}} also performs poorly suggesting a different loss function must be used to obtain transferability.

| Model   | VGG Nat Acc | VGG Adv Acc | DNN Nat Acc | DNN Adv Acc |
|---------|-------------|-------------|-------------|-------------|
| Nat. Training | 91.47% | 1.49% | 78.86% | 1.93% |
| AAA_{\text{L}_{\text{exc}}} | 25.02% | 12.52% | 14.25% | 2.01% |
| AAA_{\text{L}_{\text{mse}}} | 87.78% | 2.70% | 76.77% | 4.65% |
| AAA_{\text{L}_{\text{exc}}+L_{\text{mse}}} | 30.53% | 15.10% | 20.08% | 6.95% |

Table 7: AAA transferability evaluation on CIFAR-10. The AE is trained on ResNet and is transferred to two different classifiers.

Alternatively, we can use ensemble adversarial training to create an AE that improves the adversarial performance of multiple models in a single training session. Ensemble adversarial training is a modification of adversarial training that randomly selects a model from an ensemble each epoch and generates adversarial examples with respect to the chosen model, rather than the target training model (Tramèr et al. 2018). Traditional adversarial training created models that remained vulnerable to transferability attacks. Tramèr et al. showed that their ensemble modification solves this problem, improving a model’s robustness to such attacks. With respect to AAA, we use ensemble adversarial training to create an AE that can improve the adversarial accuracy of all classifiers in the ensemble, while also only modestly reducing natural accuracy of each classifier.

In each training iteration, we randomly select one of the three classifiers and generate adversarial examples with respect to the chosen classifier. Then, we adversarially train the AE with respect to \( \text{L}_{\text{exc}} \) and \( \text{L}_{\text{mse}} \) and the chosen classifier. We choose to use \( \text{L}_{\text{exc}}+\text{L}_{\text{mse}} \) as, based on the previous experiments, this was the most likely loss function to result in an accurate, transferable AE. All of the training parameters remain the same as before including the number of training epochs. In Table 8, we compare the performance of AAA using ensemble adversarial training to the adversarially trained classifiers.

We observe that AAA using ensemble adversarial training has two advantages over standard adversarial training. First, it achieves similar or better performance than standard adversarial training. For VGG and DNN, AAA significantly improved both the natural and adversarial accuracy (e.g., an additional 10.12% and 9.17% for DNN’s natural and adversarial accuracy respectively). For ResNet, it remained competitive with respect to the adversarially trained model. Second, and more importantly, AAA is transferable across all three models while requiring only \( \frac{1}{3} \) of the total training iterations necessary for standard adversarial training. In order obtain adversarially robust classifiers using standard adversarial training, three adversarial training sessions must be run individually.

### Extensions to AAA

In this section, we discuss improvements and future work regarding adversarial autoencoder augmentation.

#### kNN Reconstruction

Traditionally, given an input, the classifier outputs the label associated with the highest predicted probability. However, recent work has found that using a k-nearest neighbors (kNN) algorithm in the hidden layers of the network can improve the explainability of neural networks and establish confidence metrics on classifier predictions (Papernot and McDaniel 2018). For a given input and a given hidden layer, the k closest neighbors are selected and the confidence of a prediction is based on the fraction neighbors that agree with the output prediction. On normal inputs, it was shown that a majority of the k closest
neighbors would often agree on the predicted label in each hidden layer of a naturally trained model. However, for adversarial inputs, which are not part of the training data manifold, there was much more diversity in the labels of the $k$ nearest neighbors, resulting in a low confidence prediction. Based on this observation, we measure the performance benefits if the kNN algorithm is used during the reconstruction step. As Adversarially-Trained Autoencoder Augmentation creates an AE that is robust to adversarial inputs and transferable, we expect that inputs with the same label will be close together in the latent space.

First, we store the latent space vectors of the training data. Then, at runtime, we compute the ten nearest neighbors for a given test input, and average the latent space representations of those neighbors. The average latent space representation is used to compute reconstruction output. Table 9 shows the results of kNN reconstruction on the MNIST and Fashion-MNIST CNN models and Table 10 show the transferability evaluation on the fully connected classifiers. In most cases, kNN reconstruction further improves the adversarial accuracy of AAA. Furthermore, we see that kNN reconstruction significantly improves the transferability of AAA. This behavior is likely because the average latent space representation obtained from kNN projects the adversarial input to a point on the natural data manifold, which in an input recognizable to all the classifiers.

One drawback of kNN reconstruction is the large performance overhead during evaluation. In MNIST, we use a latent space vector of size 128 and the evaluation on the test dataset took approximately five minutes. Contrast this in Fashion-MNIST, where the latent space vector is of size 7x7x256 requiring approximately 13 hours for evaluation on the test dataset. In future work, we will explore optimization techniques such as locality sensitive hashing to improve the speed and reduce the complexity of the nearest neighbors.

The average latent space representation is used during the reconstruction step. As Adversarially-Trained Autoencoder Augmentation represents the first transferable adversarial defense that is robust to adaptive $L_\infty$ adversary.

### Table 9: Evaluation results using the kNN reconstruction modification for a white-box PGD attack. Green and red numbers show the accuracy differences compared to the results in Tables 1 and 4 where kNN reconstruction was not used.

| MNIST | Natural Accuracy | Adversarial Accuracy |
|-------|------------------|----------------------|
| $\text{AAA}_{\text{natural}}$ | 98.55% | 96.03% (+0.84%) |
| $\text{AAA}_{\text{natural}}$ | 97.16% | 94.91% (+0.62%) |
| $\text{AAA}_{\text{natural}}+\text{L}_2$ | 98.21% | 95.92% (+0.65%) |

| Fashion-MNIST | Natural Accuracy | Adversarial Accuracy |
|---------------|------------------|----------------------|
| $\text{AAA}_{\text{natural}}$ | 58.49% | 55.92% (-23.89%) |
| $\text{AAA}_{\text{natural}}$ | 83.30% | 79.52% (+36.42%) |
| $\text{AAA}_{\text{natural}}+\text{L}_2$ | 76.40% | 73.57% (-3.26%) |

### Table 10: AAA transferability results using the kNN reconstruction modification using a fully connected classifier. Green and red numbers show the accuracy differences compared to the results in Tables 2 and 5 where kNN reconstruction was not used.

| MNIST | Natural Accuracy | Adversarial Accuracy |
|-------|------------------|----------------------|
| $\text{AAA}_{\text{natural}}$ | 29.67% | 27.60% (+10.88%) |
| $\text{AAA}_{\text{natural}}$ | 97.12% | 94.11% (+5.21%) |
| $\text{AAA}_{\text{natural}}+\text{L}_2$ | 98.12% | 95.63% (+5.01%) |

| Fashion-MNIST | Natural Accuracy | Adversarial Accuracy |
|---------------|------------------|----------------------|
| $\text{AAA}_{\text{natural}}$ | 38.11% | 37.24% (+10.88%) |
| $\text{AAA}_{\text{natural}}$ | 81.77% | 76.41% (+38.07%) |
| $\text{AAA}_{\text{natural}}+\text{L}_2$ | 77.91% | 75.40% (+21.21%) |

### Conclusion

In this paper, we propose Adversarially-Trained Autoencoder Augmentation as a model agnostic adversarial defense. AAA provides comparable performance to traditional adversarial training, while allowing complete transferability for simpler datasets such as MNIST and MNIST-Fashion. On more complex datasets such as CIFAR-10, AAA can parallelize adversarial training across multiple classifiers, achieving, at minimum, comparable adversarial performance to an adversarially trained model. To our knowledge, Adversarially-Trained Autoencoder Augmentation represents the first transferable adversarial defense that is robust to an adaptive $L_\infty$ adversary.
Acknowledgements
This material is based in part upon work supported by the National Science Foundation under Grant No. 1646392 and the NVIDIA GPU Grant program.

References
Athalye, A.; Carlini, N.; and Wagner, D. A. 2018. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In International Conference on Learning Representations (ICLR).

Bhagoji, A. N.; He, W.; Li, B.; and Song, D. X. 2018. Practical black-box attacks on deep neural networks using efficient query mechanisms. In European Conference on Computer Vision (ECCV).

Chen, P.-Y.; Zhang, H.; Sharma, Y.; Yi, J.; and Hsieh, C.-J. 2017. Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In ACM Workshop on Artificial Intelligence and Security (AiSec).

Cheng, M.; Le, T.; Chen, P.-Y.; Zhang, H.; Yi, J.; and Hsieh, C.-J. 2018. Query-efficient hard-label black-box attack: An optimization-based approach. ArXiv abs/1807.04457.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. In International Conference on Learning Representations (ICLR).

Gu, S., and Rigazio, L. 2015. Towards deep neural network architectures robust to adversarial examples. In International Conference on Learning Representations (ICLR), Workshop Track Proceedings.

Guo, C.; Rana, M.; Cisse, M.; and van der Maaten, L. 2018. Countering adversarial images using input transformations. In International Conference on Learning Representations (ICLR).

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, 770–778.

Juuti, M.; Szyller, S.; Marchal, S.; and Asokan, N. 2019. Prada: Protecting against dnn model stealing attacks. In IEEE European Symposium on Security and Privacy (EuroS&P).

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Kurakin, A.; Goodfellow, I.; and Bengio, S. 2016. Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533.

Kurakin, A.; Goodfellow, I. J.; and Bengio, S. 2017. Adversarial machine learning at scale. In International Conference on Learning Representations (ICLR).

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11):2278–2324.

Liao, F.; Liang, M.; Dong, Y.; Pang, T.; Zhu, J.; and Hu, X. 2018. Defense against adversarial attacks using high-level representation guided denoiser. IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

Liu, Y.; Chen, X.; Liu, C.; and Song, D. 2016. Delving into transferable adversarial examples and black-box attacks. In International Conference on Learning Representations (ICLR).

Madry, A.; Makelov, A.; Schmidt, L.; Tsipras, D.; and Vladu, A. 2018. Towards deep learning models resistant to adversarial attacks. In International Conference on Learning Representation.

Meng, and Chen, H. 2017. MagNet: a two-pronged defense against adversarial examples. In ACM SIGSAC Conference on Computer and Communications Security (CCS).

Naroditskaya, N., and Kasiviswanathan, S. P. 2017. Simple black-box adversarial perturbations for deep networks. ArXiv abs/1612.06299.

Papernot, N., and McDaniel, P. D. 2018. Deep k-nearest neighbors: Towards confident, interpretable and robust deep learning. arXiv preprint arXiv:1803.04765.

Papernot, N.; McDaniel, P. J.; Jha, S.; Fredrikson, M.; Celik, Z. B.; and Swami, A. 2016a. The limitations of deep learning in adversarial settings. In IEEE European Symposium on Security and Privacy (EuroS&P).

Papernot, N.; McDaniel, P. D.; Wu, X.; Jha, S.; and Swami, A. 2016b. Distillation as a defense to adversarial perturbations against deep neural networks. In IEEE Symposium on Security and Privacy (S&P).

Papernot, N.; Abadi, M.; Erlingsson, U.; Goodfellow, I.; and Talwar, K. 2017. Semi-supervised knowledge transfer for deep learning from private training data. In International Conference on Learning Representations (ICLR).

Papernot, N.; McDaniel, P.; and Goodfellow, I. 2016. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. arXiv preprint arXiv:1605.07277.

Ronneberger, O.; Fischer, P.; and Brox, T. 2016. U-net: Convolutional networks for biomedical image segmentation. In International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI).

Samangouei, P.; Kabkab, M.; and Chellappa, R. 2018. Defense-gan: Protecting classifiers against adversarial attacks using generative models. In International Conference on Learning Representations (ICLR).

Shafahi, A.; Najibi, M.; Ghiasi, A.; Xu, Z.; Dickerson, J.; Studer, C.; Davis, L. S.; Taylor, G.; and Goldstein, T. 2019. Adversarial training for free! arXiv preprint arXiv:1904.12843.

Song, Y.; Kim, T.; Nowozin, S.; Ermon, S.; and Kushman, N. 2018. PixelDefend: Leveraging generative models to understand and defend against adversarial examples. In International Conference on Learning Representations (ICLR).

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2014. Intriguing properties of neural networks. In International Conference on Learning Representations (ICLR).

Tramèr, F.; Kurakin, A.; Papernot, N.; Goodfellow, I.; Boneh, D.; and McDaniel, P. 2018. Ensemble adversarial training: Attacks and defenses. In International Conference on Learning Representations (ICLR).

Xiao, H.; Rasul, K.; and Vollgraf, R. 2017. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747.
Supplemental Material

CIFAR-10 Model Architectures

| U-Net          | DNN          |
|----------------|--------------|
| conv-bn × 2    | conv 32-3-1  |
| conv-bn × 2    | conv 128-3-1 |
| conv-bn × 2    | maxpool 2×2  |
| conv-bn × 2    | dropout 0.5  |
| upscale + concat | 2×2 128-3-1 |
| conv-bn × 2    | maxpool 2×2  |
| upscale + concat | 2×2 128-2-1 |
| conv-bn × 2    | dropout 0.5  |
| conv (sigmoid) | dense 1500   |
| conv (sigmoid) | dropout 0.5  |
|               | dense 10     |

Table 13: Model architecture for the U-Net and DNN models used in the CIFAR-10 experiments (kernel size, number of output filters, stride). For the architecture details of ResNet and VGG, the reader can refer to (He et al. 2016; Zisserman 2015). ReLU activation is used unless specified.

MNIST Model Architectures

| Auto-encoder | DNN          |
|--------------|--------------|
| conv 32-3-2  | conv 32-3-1  |
| conv 64-3-2  | maxpool 2×2  |
| dense 1024   | maxpool 64-3-1 |
| dense 7×7×64 | maxpool 2×2  |
| deconv 32-3-2| dense 1024   |
| deconv (sigmoid) | dense 10     |

Table 11: Model architecture for each of the models used in the MNIST experiments (kernel size, number of output filters, stride). ReLU activation is used unless specified.

Fashion-MNIST Model Architectures

| Auto-encoder | DNN          |
|--------------|--------------|
| conv-bn × 2  | conv 32-3-1  |
| maxpool 2×2  | maxpool 2×2  |
| conv-bn × 2  | maxpool 64-3-1 |
| maxpool 2×2  | maxpool 2×2  |
| conv-bn × 2  | dense 1024   |
| conv-bn × 2  | dense 256    |
| conv-bn × 2  | dense 256    |
| conv-bn × 2  | dense 10     |
| conv (sigmoid) | dense 10     |
| conv (sigmoid) | dense 10     |

Table 12: Model architecture for each of the models used in the Fashion-MNIST experiments (kernel size, number of output filters, stride). ReLU activation is used unless specified.