Exploring Adversarial Input Spaces for Convolutional Neural Network Defense

Austin Wang
austinw2@stanford.edu

Daniel Kunin
kunin@stanford.edu

Justin Pyron
pyron@stanford.edu

Abstract

Convolutional neural networks have made great strides in a wide range of domains such as image classification, object detection, facial recognition, and captioning, but adversarial attacks have exposed their fragility and lack of robustness. In this paper, we use the MNIST and CIFAR-10 datasets to generate adversarial images for CNN models, explore various statistics of the adversarial perturbations, and test possible defense strategies. In particular, we analyze the effects of using a dropout ensemble technique at test time, adding a dropout-enabled autoencoder ensemble to remove adversarial perturbations, and training a variation of a GAN to improve resistance to attacks. We find that the dropout ensemble technique is extremely effective in defending against adversaries that have been minimally perturbed, but fails when it encounters more heavily perturbed images. On the other hand, the dropout-enabled autoencoder ensemble handles both minimal and heavy perturbations decently well, but at the cost of losing prediction accuracy on the original, unperturbed images. The GAN variant that we test only provides slight improvements over base models, but offers a promising new approach to increasing the robustness of a model.

1. Introduction

Despite the remarkable success of deep learning in recent years, the susceptibility of deep networks to being fooled by adversaries remains an ever present problem. Szegedy et al. demonstrated that images can be strategically perturbed so that the resulting images are indistinguishable to humans, but the network confidently misclassifies them [12]. See Figure 1 for a grid of adversarial examples. Adversarial images fall into two categories: targeted and untargeted. In a targeted attack, an original image is perturbed to cause the model to misclassify to a specific target class, while in an untargeted attack, an original image is perturbed with the goal of causing misclassification to *any* class. A clearer understanding of the structure of these adversarial inputs would help shed light on ways to defend against adversarial attacks and increase the robustness of deep networks.

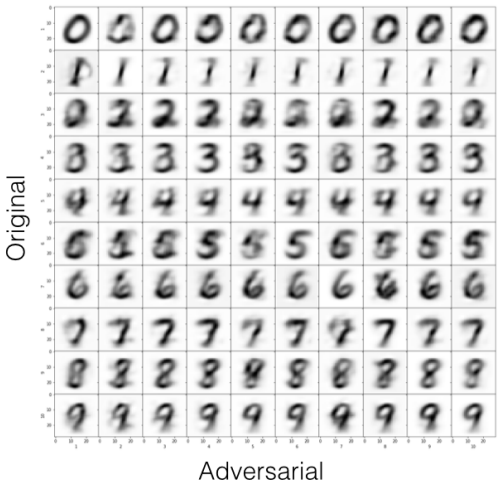

Figure 1. A grid showing all the average adversarial images we generated for MNIST. Image i, j is an image originally labeled as class i, but perturbed such that it would be classified as class j. The diagonal images are original MNIST images.

1.1. Problem Statement

Our goal is to conduct an exploration of the space of adversarial inputs for a convolutional neural network to help motivate strategies for adversarial defense. To this end, we use the MNIST and CIFAR-10 datasets and generate thousands of adversarial images for each. We also compute the differences between these adversarial images and the images used to generate them, which we call perturbations.

By looking at statistics of these adversarial images and perturbations such as mean image, standard deviation, histograms of pixel values, and norm size, we attempt to uncover some type of discernible structure, especially since the transferability of adversarial inputs across networks has been documented in literature [12]. We attempt to use the analysis of the perturbations and adversarial images to find new approaches to adversarial defense. A metric we use for evaluating the robustness of our new model is the proportion of adversarial image labels we were able to fix, without causing significant damage to the model's ability to correctly classify the original, unperturbed images.

1.2. Background and Data

When generating targeted adversarial images, the objective is to determine how to modify each pixel in order to cause the model to misclassify to the target class. This is done by computing the gradient of the score of the target class with respect to each pixel. Just as how a model is trained with gradient descent, pixels are updated via gradient *ascent* to produce a new image with higher scores for the target class. There exist several methods of iteratively perturbing an input image with the gradient of its pixels, which are discussed in section 2.2.

In our project, we analyze adversarial images for the MNIST and CIFAR-10 datasets. For an image in a subset of each dataset, we generate a targeted adversarial image for each of the remaining nine classes using a model trained on that dataset. As such, we assume that adversaries have access to the model they are trying to compromise, a so-called "white box" attack [14]. In total, we generate 90,000 adversarial images for both MNIST and CIFAR-10.

1.3. Related Work

The vulnerability of deep networks to adversarial attack has spurred attempts to increase robustness of networks. The most popular method to defend against adversarial images so far has involved generating a set of adversarial images, assigning to them the proper label, and retraining the model with these adversarial images in the training set [14]. Such techniques lead to increased robustness, but make no attempt to understand the properties of the network responsible for the vulnerabilities, nor attempt to modify the model to remedy such vulnerabilities. In addition, new adversarial images can be generated for the model that has been retrained with previously generated adversarial images, making this strategy particularly weak.

Another popular approach involves building a classifier that discriminates between natural images and adversarial images, with the ultimate goal of using it to detect adversarial inputs if they are fed to the original model [2]. Classifiers that successfully discriminate between original images and adversarial images generated from them have been successfully built. However, most models augmented with a classifier in this way do not correctly predict the class to which the adversarial images should belong, but rather simply state that they belong to an adversarial class. This fails to address the network's lack of robustness. In addition, such models again remain vulnerable to attack, since new adversarial images can be generated to trick the new model with an attached classifier filter.

A final approach makes no attempt to re-train the original model or detect adversarial examples, but instead attempts to remove adversarial perturbations by passing input images through a filtering network before feeding them to the model. Gu and Rigazio [7] propose using a denoising autoencoder to filter out adversarial perturbations. Autoencoders for images are a type of neural network model that first transform an image into a lower dimensional representation using a series of downsampling layers (encoding), followed by a series of additional upsampling layers that attempt to replicate the input image (decoding). The rationale behind this approach is that idiosyncratic perturbations will be washed out and smoothed in the encoding stage.

2. Motivation and Methodology

2.1. An Analogy to Numerical Linear Algebra

In numerical linear algebra, we are very concerned with how error propagates through an algorithm. Because numbers cannot be represented with infinite precision, roundoff errors are introduced into any calculation. The goal is to bound this error and design algorithms that minimize its propagation. To this end, we generally talk about three forms of error: Forward Error, Backward Error and Sensitivity.

We assume our algorithm can be represented as a surjective function $f : X \to Y$ (a map from the input space X to the output space Y), however $\tilde{f} : X \to Y$ is the actual result of running our algorithm when accounting for roundoff error. The goal of error analysis in numerical linear algebra is to understand the relationship between f and $\tilde{f}$. These three forms of error can formally and pictorially be defined as shown in Figure 2.

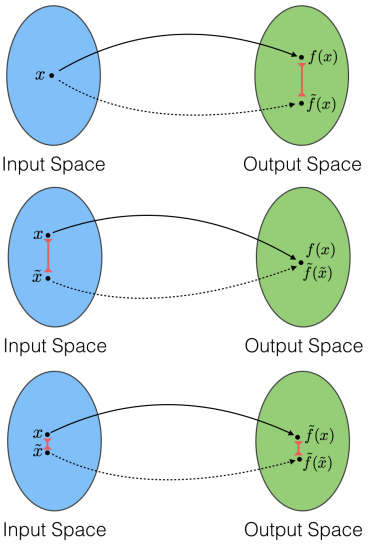

Figure 2. Forward Error (top) is the distance in output space for a given input: $||f(x) - \tilde{f}(x)||$. Backward Error (middle) is the distance in input space such that the outputs are equal: $||x - \tilde{x}||$. Sensitivity (bottom) is the ratio of forward error with backward error. It can be understood as a measure of how much our output, $\tilde{f}(x)$, changes when our input, x, is perturbed. [3]

Linear algebra is the study of linear transformations, however, the error analysis paradigm described above is not unique to linear transformations. In fact, this framework lends itself very nicely to deep learning. A deep learning task can be understood as learning a non-linear transformation $\tilde{f}$ by minimizing some loss function evaluated on a set of training samples $(x, f(x))$. This loss function is generally defined as a distance or distortion measure between $f(x)$ and $\tilde{f}(x)$. Notice that this scheme is analogous to minimizing forward error in the numerical linear algebra setting. Similarly, we can understand adversarial attacks as the sensitivity of our neural network. An adversarial attack is a small perturbation in the input space that propagates to large errors in the output space. We use this analogy for understanding adversarial robustness as a framework for motivating future experiments and defense strategies.

2.2. Generating Adversarial Examples

Several methods exist for generating adversarial images, each of which utilizes the gradient of the outputs of a network with respect to input pixels. Let F denote a model that maps input image x to a vector of predicted class scores $F(x)$. Letting F_j denote the function that maps input image x to the score for class j, the gradient $\nabla_x F_j(x)$ tells how impactful each input pixel is on the score for class j. Methods that generate adversarial images utilize this information about the responsiveness of class scores with respect to pixels to strategically modify pixel values in order to push outputted class scores in the desired direction.

The Fast Gradient Sign Method (FGSM) is one approach that generates targeted adversarial examples. For target class t, gradients of the score of class t with respect to input pixels are computed. These signs of the gradients are scaled by a small value ϵ and then added to the original image x to produce an adversarial image x' [1]:

$$x' = x + \epsilon \operatorname{sign}\Big(\nabla_x F_t(x)\Big)$$

The Jacobian Based Saliency Map Approach (JSMA) follows a similar approach by constructing a saliency map, which is the Jacobian of class output scores with respect to input pixels. But rather than update all pixels, JSMA is more selective and only updates the pixels that have the greatest impact on fooling the model. The impact is measured by the pixel's ability to both increase the score to the target class *and* decrease the score to all other classes. For a single pixel $x_{i,j}$, one way to measure impactfulness is a sum of derivatives:

$$\frac{\partial F_t(x)}{\partial x_{i,j}} + \sum_{k \neq t} -\frac{\partial F_k(x)}{\partial x_{i,j}}$$

Only the fraction of pixels with the highest impactfulness are updated.

For our analysis, we train a classification model for both MNIST and CIFAR-10. [1] Each model is relatively simple, and contains several sequential convolution-ReLU-maxpool layers. Using these models, we generate a dataset of adversarial images following an approach similar to FGSM, but slightly modified. Instead of perturbing images by the sign of the gradient, we update by the gradient itself. As a result, the perturbations applied to a pixel are not uniform. Under this method, we generate adversarial images in two slightly different ways. In the first, we iteratively add perturbations to original images only until the model is fooled, after which we immediately halt the iterations ("minimal perturbation adversaries"). In the second, we iteratively add perturbations until the model is tricked and subsequently continue for several more iterations ("extended perturbation adversaries"). We apply Method 1 to both MNIST and CIFAR-10, and Method 2 just to CIFAR-10. See Figure 1 for a grid of generated adversarial images.

2.3. Perturbation Statistics

With our adversarial images generated, we create a Perturbation Stats Python class equipped with functionality to read in and store the original images, adversarial images, associated perturbations, and labels along with methods to produce summary statistics and plots. These include histograms of the L_1 norm, L_2 norm, and the raw pixel values, tables of the mean, standard deviation, maximum, and minimum for arbitrary adversarial images or perturbations, and visualization capabilities for images with a given original label and adversarial label. Using this class, we are able to quickly analyze key features of adversarial inputs to help us learn more about this space and guide our defense approaches. Table 1 shows the key statistics for the adversarial perturbations we generate.

For the MNIST dataset, we notice that our perturbations appear to be highly correlated to the original and target class. As seen in Figure 3 the perturbations resemble two superimposed images, one subtracting the original image class and the second adding the target class. This is slightly unexpected as most of the adversarial perturbations reported in literature resemble something closer to random noise [6]. We believe this is probably a condition of the simplicity of MNIST and that the method we used to generate the perturbations is targeted to a specific class.

The CIFAR-10 dataset produces perturbations that are less discernible, as seen in the grid in Figure 4. In many of these images, the average perturbations look like noise, but in a structured concentric pattern. A likely cause of this is that most of the structure in the CIFAR-10 images comes from the edges of objects that appear in the center of the frame. Because our adversarial image generation uses gra-

[1] Code used to train models loosely adapted from http://cs231n.stanford.edu/notebooks/pytorch_tutorial.ipynb

		MNIST (Minimal)	CIFAR-10 (Minimal)	CIFAR-10 (Extended)
Pixel Values	Mean	0.009	8.03e-6	2.48e-5
	Std.	0.049	0.008	0.012
	Min	-0.949	-0.195	-0.253
	Max	0.703	0.148	0.204
L1-Norm	Mean	17.3	16.5	25.9
	Std.	5.08	8.04	9.10
	Min	0.170	1.08	5.15
	Max	32.2	57.3	74.6
L2-Norm	Mean	1.35	0.418	0.651
	Std.	0.362	0.201	0.225
	Min	0.014	0.027	0.130
	Max	2.00	1.41	1.94

Table 1. Statistics on the perturbations (difference between adversarial image and original image) for the three adversarial image datasets generated. Note that all pixel values for the original images have been rescaled to be between 0 and 1.

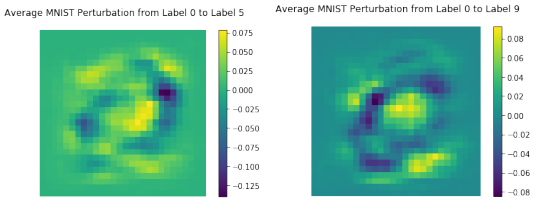

Figure 3. Average perturbation between original MNIST images with label 0 and their adversarial examples with label 5 (left) and label 9 (right).

dient ascent on the input pixels, this observation indicates that the most influential pixels for predicting image class come in bands radiating out from the center of the image.

3. Defense Strategy Experiments and Results

We use our understanding of the statistics and general structure of the adversarial perturbations to motivate three different defense strategies: Dropout Ensemble, Dropout-Enabled Autoencoder Ensemble, and Generative Adversarial Classifier.

3.1. Dropout Ensemble

The first defense strategy we test is motivated by a hypothesis that the process which generates adversarial examples is highly tailored to specific components of a model's architecture. Although the transferability of adversarial examples to distinct models is well-documented, we hypothesize that there exists common regularity in the internal architecture of models trained to classify specific types of data (e.g. CIFAR-10), and that adversarial examples exploit this common regularity. If our hypothesis is correct, then disrupting the ordinary architecture of the network should attenuate the effectiveness of adversarial images.

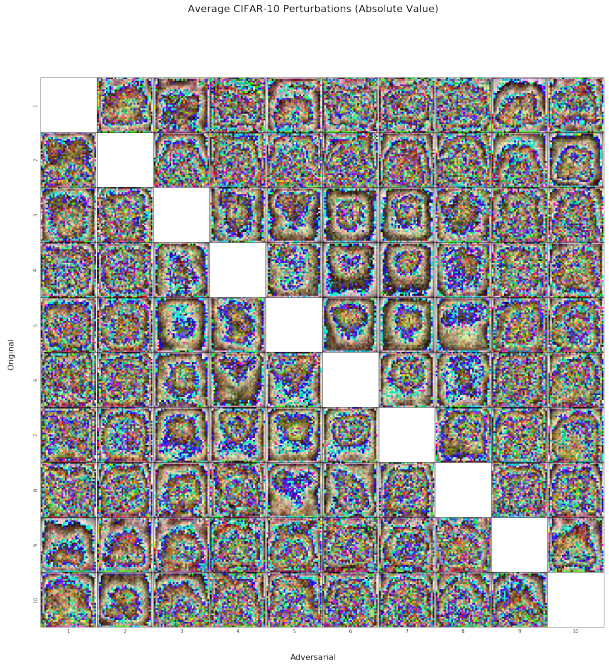

Figure 4. Grid of the average perturbations for the CIFAR-10 dataset, depicting specific (original label, target label) pairs. Note that because perturbations include negative pixel values, these color plots use the absolute values of each of the color channels and plot the RGB result.

To embed disruptive behavior into a model, we propose an approach that injects randomness in predictions through dropout. More specifically, we test an approach where a collection of predictions is computed for each input, where each prediction is the result of a forward pass through the network with dropout enabled between layers. The final output classification is the class which receives a plurality of votes from the collection of predictions for that example.

We hypothesize that when an adversarial example is fed forward through the network in such a manner, the components of the architecture that it depends on exploiting to successfully fool the model will be jumbled, decreasing its adversarial strength. However, the injection of randomness will also weaken the model's ability to predict legitimate images. We hypothesize that legitimate images, on average, will not be significantly disrupted by dropout, and that averaging predictions through an ensemble of forward passes will remedy any predictive variability that would hinder performance on original images.

Even if the dropout ensemble method is not strong enough to cause the correct classification of adversarial inputs, it still provides information potentially useful toward detecting an adversarial image. In particular, since we believe that the injection of randomness in the prediction stage disproportionately affects adversarial images, it would follow that the distribution of class predictions for adversarial examples will have much greater variability than that of original images. In preliminary investigations, this phenomenon was confirmed. See Figure 5.

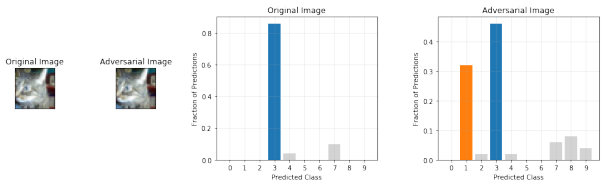

Figure 5. Histogram of predictions from an ensemble of dropout-enabled forward passes for a single legitimate image and a corresponding minimal perturbation adversarial image. Blue bars correspond to the original class, while orange bars correspond to the targeted class.

Guided by these preliminary hypotheses, we test a dropout enabled ensemble model on two sets of adversarial CIFAR-10 images: minimal perturbation adversaries and extended perturbation adversaries, which are described in section 2.2.

On a dataset of minimally perturbed images, dropout defense works fairly well and is able to successfully thwart a large fraction of adversaries. For each (original label, target label) pair, we compute the percentage of adversaries that fail to fool the dropout defense enabled model, which exceeded 60% for several (original label, target label) pairs (see Figure 6). Furthermore, 100% of original images continue to be correctly predicted by the defended model. We additionally compute the variance of the ensemble of predictions for each input image. We find that the variance of ensemble predictions on adversarial images is far larger than that on original images (see Figure 7).

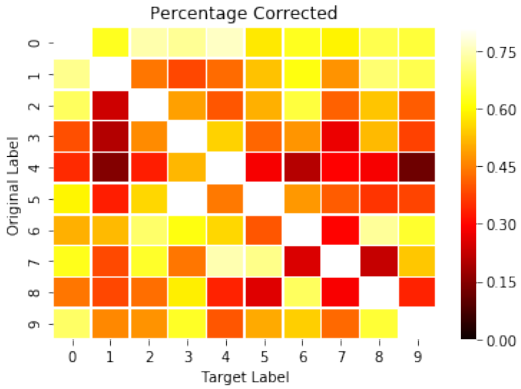

Figure 6. Heatmap of percentage of minimally perturbed CIFAR-10 adversarial images that failed to fool model when dropout defense is enabled.

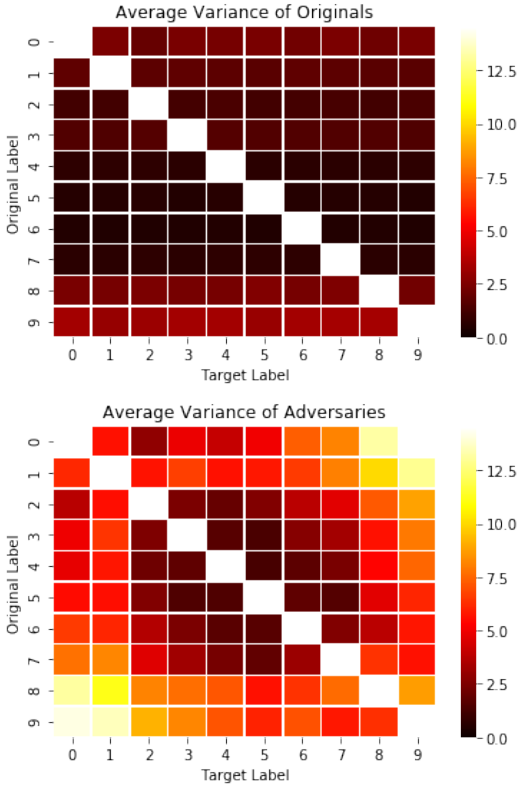

Figure 7. Average variance of ensemble of predictions on original CIFAR-10 images (top). Average variance of ensemble of predictions on minimally perturbed CIFAR-10 adversarial images (bottom).

Suspicious that the above results are due to the delicacy of minimally perturbed adversarial images, we next apply the same procedure to extended perturbation adversarial images. In this scenario, dropout defense performs much more poorly. Nearly all extended perturbation images success-

fully fool the model with dropout defense. Furthermore, the variance of the ensemble of predictions is essentially identical for original images and adversaries.

The results of our experimentation with ensembles of dropout-enabled forward passes imply that the performance of dropout defense depends heavily on the fragility of adversaries. Minimally perturbed images are thwarted quite easily by dropout defense, while extended perturbation adversaries remain impervious.

Despite the failure of dropout to thwart extended perturbation images, we still find these results encouraging. Since dropout defense filters out minimally-perturbed images quite well, efforts to overcome such a defense strategy would require adding greater perturbations to images, thus making them more liable to detection. This suggests a potential area of new research, since more sophisticated dropout defense strategies may be capable of defending against even extended perturbation adversaries.

3.2. Dropout-Enabled Autoencoder Ensemble

Our decision to use an autoencoder as a method for adversarial defense is motivated by our perturbation and adversarial image statistics study. The perturbations we generate tend to be relatively small in norm and to resemble targeted noise; as a result, it seems natural to use a technique similar to a denoising autoencoder to attempt to remove the adversarial perturbation from a given adversarial image. The training of the autoencoder uses as inputs a subset of the adversarial images generated for CIFAR-10 (or MNIST), with associated responses the original, unperturbed images from which the adversarial images originated. Therefore, this is not a traditional autoencoder that attempts to recreate the input, but rather one that attempts to slightly alter the input to make it no longer adversarial. The autoencoder architecture is purely convolutional, using a combination of convolutional 2D layers for downsampling and convolutional 2D transpose layers for upsampling, and includes ReLU activations and batch normalization layers[2].

The main issue with using an autoencoder technique such as this one, as documented in [10], is that new adversarial attacks can be generated for the autoencoder-augmented model. To address this issue, we introduce test-time dropout in the autoencoder itself. The idea is that we trade accuracy of the autoencoder reconstruction for increased resistance to new adversarial attacks, as it is harder to target specific model attacks when the model employs an aggregation of random forward passes when predicting. We call this method dropout-enabled autoencoder ensemble defense, which combines previous denoising techniques with concepts from section 3.1.

The dropout-enabled autoencoder defense technique works fairly well at thwarting both minimal perturbation as well as extended perturbation adversarial CIFAR-10 images. Roughly half of both minimal perturbation and extended perturbation adversaries are thwarted by the dropout-enabled autoencoder. See Figure 8. However, the cost of this increased robustness to adversaries is a significantly reduced ability to correctly classify original images. Because it only makes sense to generate adversaries for images that get correctly classified by the model, 100% of the original images with which we test the dropout-enabled autoencoder get correctly classified by the original model. However, only roughly 60% to 70% of original images get correctly classified by the model equipped with dropout enabled autoencoder defense, depending on the class of the tested image.

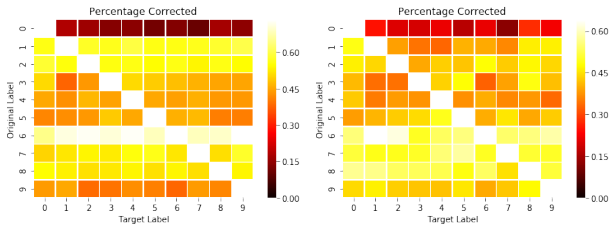

Figure 8. Percentage of minimal perturbation CIFAR-10 adversaries that are thwarted by dropout enabled autoencoder defense (left). Percentage of extended perturbation CIFAR-10 adversaries that are thwarted by dropout enabled autoencoder defense (right).

3.3. Generative Adversarial Classifier

Inspired by our analogy of adversarial robustness to numerical linear algebra and the recent successes of Generative Adversarial Networks (GAN) at solving complex min-max problems, we develop a GAN variant strategy to adversarial defense. In a recent paper on adversarial attacks [9], the authors propose viewing adversarial robustness as an inner maximization problem that attempts to craft an effective adversarial attack given an input and an outer minimization problem that attempts to train a network which minimizes the loss associated with this attack. Combining this paper's suggestion with our numerical linear algebra discussion in section 2.1, we propose the following min-max formulation. Let x denote training images, y training labels, C a classifier, $G(x)$ an adversarial attack generator, and L the loss function. Then our objective is:

$$\min_C \max_G \left\{ L\Big(C(x), y\Big) + L\Big(C(G(x)), y\Big) - \lambda ||G(x) - x||_2^2 \right\}$$

Noticing the resemblance of this formulation to the min-max loss objective of GANs, we hypothesize that a variant of a GAN could be used to train classifiers that are more

[2] Autoencoder skeleton adapted from:
https://gist.github.com/okiriza/16ec1f29f5dd7b6d822a0a3f2af39274

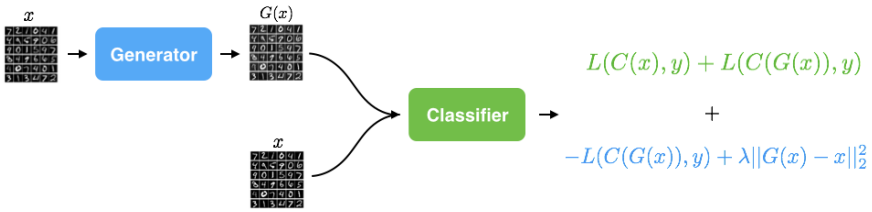

Figure 9. Depiction of the structure of our GAN variant model. The equation in blue is the loss used to train the generator and the equation in green is used to train the classifier.

robust to adversaries. See Figure 9 for a depiction of the structure of our GAN variant.

There are four major differences between an ordinary GAN and our GAN variant, which we term a Generative Adversarial Classifier (GAC).

1. The input to the GAC generator is not random noise, but rather an original training image, which the generator attempts to perturb in a way that maximizes the classifier's loss.

2. In place of a binary discriminator, a GAC uses a multiclass classifier, which attempts to correctly classify both the generated and original images as their original training label.

3. The generator's loss function has an additional term which regularizes the L2 norm of the perturbation added by the generator. This regularization term penalizes large perturbations, which prevents the generated image from becoming distinguishable from the input image.

4. The generator uses a ResNet style identity block architecture, which assures that the generated image resembles the input image throughout training.

We implement two versions of our Generative Adversarial Classifier, one resembling a vanilla GAN and another resembling a DC-GAN[3]. We find that both networks' generators are able to produce images of high fidelity to the original image, but with targeted noise. Additionally, we find that both networks' classifiers obtain high accuracies on the original images, while demonstrating slightly increased robustness to adversarial attacks. However, these successes are highly dependent on a correct choice of the regularization constant λ.

Surprisingly, we find that our generators are able to create minimal perturbations such that the resulting perturbed images resemble hybrids between images of different classes. As shown in Figure 10, with minimal training on MNIST, our generator learns to add and remove pixels such that certain digits morph into others without ever being given a specific target. This suggests that our generator is exploring the loss landscape and decision boundaries of our classifier to determine the smallest perturbation needed in order to trick the classifier. This process directly parallels how adversarial attacks are generated.

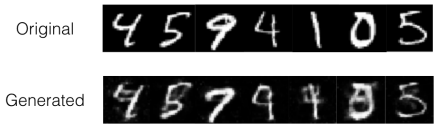

Figure 10. Samples of original and generated MNIST images using our vanilla GAC after 20 epochs of training. From left to right we observe perturbations of the following type: $(4 \to 7), (5 \to 8), (9 \to 7), (4 \to 9), (1 \to 4), (0 \to 8), (5 \to 3)$.

However, as noted previously, the quality of the generated images are highly dependent on the choice of the regularization constant λ. As seen in Figure 11, if the choice of λ is too low, the generated images become noisy. Conversely, if the choice of λ is too high, the generated images are identical to the input image. A correctly chosen λ will lead to generated images that encourage a classifier to more effectively learn the boundaries between classes.

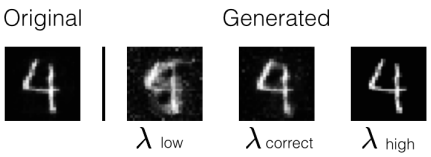

Figure 11. Generated images with different choices for the regularization constant.

To test the effectiveness of our GAC networks to adversarial defense we train a standard "vanilla" classifier with

[3]Starter code for these GANs come from Stanford University's CS231n course, in Assignment 3:
http://cs231n.github.io/assignments2018/assignment3/

the same architecture as our GAC classifier concurrently on the same training data. Throughout training we keep track of both models' accuracies on the entire train and test set and the set of adversarial attacks produced in section 2.2 for the respective dataset. For the correct choice of λ we observe a 1-3% boost in accuracy between the GAC classifier and the vanilla classifier on the adversarial dataset as shown in Figure 12.

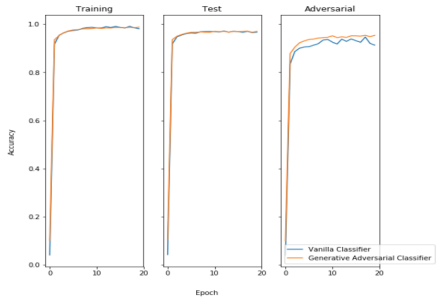

Figure 12. Accuracy plots for our GAC model and a vanilla classifier with the same architecture trained on 50% of MNIST.

We repeat this analysis using CIFAR-10 and find similar results for our GAC classifier. However, because of the higher complexity of CIFAR-10 images, we do not observe hybrid generated images as shown in Figure 13.

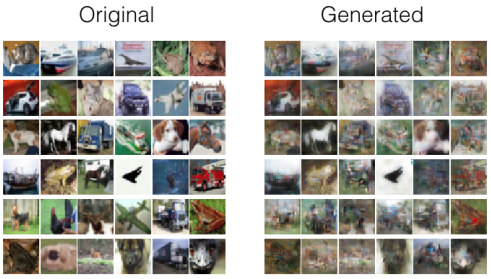

Figure 13. A grid of original and generated images from our DCGAN style GAC trained on CIFAR-10 for 100 epochs.

4. Conclusion

We test three different adversarial defense strategies and find that each possesses its own strengths and weaknesses. In particular, each strategy presents a trade-off to be made regarding susceptibility to adversaries and ability to maintain performance on original images. We also find that the effectiveness of a defense strategy will be highly dependent on how heavily images are perturbed. An encouraging result we find is that the presence of a defense mechanism requires successful adversaries to have larger perturbations, which makes them easier to detect.

From our work on this project, there are several areas of further exploration that we think would be worthwhile. Our GAN variant offers a promising approach to boost the robustness of a model, but remains sensitive to the regularization constant and is difficult to train. A combination of more careful tuning and different classifier and generator architectures could lead to more stable, effective results. In addition, we note that the model for which we generate adversarial examples is quite shallow in comparison to many state-of-art networks, and that we use a very specific generation technique. Further studies therefore might analyze the effects of model complexity and different generation techniques on both the structure of adversarial examples and the effectiveness of our proposed defense strategies.

References

[1] N. Carlini and D. A. Wagner. Towards evaluating the robustness of neural networks. *CoRR*, abs/1608.04644, 2016.

[2] N. Carlini and D. A. Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. *CoRR*, abs/1705.07263, 2017.

[3] E. Darve and M. Wootters. Numerical linear algebra with julia.

[4] Y. Dong, F. Liao, T. Pang, X. Hu, and J. Zhu. Boosting adversarial examples with momentum. *CoRR*, abs/1710.06081, 2017.

[5] M. A. et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[6] I. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015.

[7] S. Gu and L. Rigazio. Towards deep neural network architectures robust to adversarial examples. *CoRR*, abs/1412.5068, 2014.

[8] A. Kurakin, I. J. Goodfellow, and S. Bengio. Adversarial machine learning at scale. *CoRR*, abs/1611.01236, 2016.

[9] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[10] N. Papernot, P. D. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami. The limitations of deep learning in adversarial settings. *CoRR*, abs/1511.07528, 2015.

[11] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in pytorch. In *NIPS-W*, 2017.

[12] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *CoRR*, abs/1312.6199, 2013.

[13] F. Tramèr, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel. The Space of Transferable Adversarial Examples. *ArXiv e-prints*, Apr. 2017.

[14] X. Yuan, P. He, Q. Zhu, R. R. Bhat, and X. Li. Adversarial examples: Attacks and defenses for deep learning. *CoRR*, abs/1712.07107, 2017.