Deep Adversarial Robustness

Davis Liang, Patrick Hayes, Alric Althoff

June 15, 2017

Abstract

Deep learning has recently contributed to learning state-of-the-art representations in service of various image recognition tasks. Deep learning uses cascades of many layers of nonlinear processing units for feature extraction and transformation. Recently, researchers have shown that deep learning architectures are particularly vulnerable to adversarial examples, inputs to machine learning models intentionally designed to cause the model to make a mistake [2]. In this paper, we propose three methods for increasing the robustness of deep learning architectures against adversarial examples. Specifically, we propose a biologically inspired multi-task learning (MTL) model, a downsampling model, and a model trained on images created by the generator from a generative adversarial network. Our results show that using these models improve adversarial robustness against universal adversarial perturbations [5]. Additionally, we present a metric for adversarial robustness and provide extensive analysis on how adversarial images are represented in image vector space.

1 Introduction

2 Previous Work

In this section, we will review relevant technologies that we used to help increase the adversarial robustness of deep neural networks. Specifically, we will cover principal component analysis (PCA), autoencoder networks, convolutional autoencoder networks, generative adversarial models, and multi-task learning models.

Principal component analysis, PCA, is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components. By projecting data on a subset of the principal components (generally the eigenvector PCs with the highest corresponding eigenvalues), we are able to obtain a lower-dimensional representation of the original data. In general, there is a need to simplify this eigenvector problem. In particular,

$$\begin{aligned} \Sigma &= \Phi\Lambda\Phi^T \\ \Sigma &= \frac{U^TU}{N-1} \\ UU^T\Phi' &= \Phi'\Lambda \\ U^TUU^T\Phi' &= U^T\Phi'\Lambda \\ U^TU(U^T\Phi') &= (U^T\Phi')\Lambda \end{aligned} \tag{1}$$

In equation 1, U represents the mean centered data matrix, Φ represents the orthogonal eigenvector matrix, and $\Phi = U^T\Phi'$. For high-dimensional images, this trick is particularly useful as the dimensionality of the images is sometimes larger than the size of the dataset.

Autoencoder networks [6] have played a fundamental role in unsupervised learning, transfer learning, denoising data, and dimensionality reduction. Autoencoders compose of an initial neural network encoder which takes input data and compresses the data by passing it through a smaller layer of hidden units. Afterwards, the compressed data is passed through a decoder, which shares the same parameters as the encoder network but reversed (specifically, we decompress the vector representation in the hidden layer back into the data's original dimensions). A convolutional autoencoder [4] is similar to a classic multi-layer perceptron based autoencoder but with convolutional filters.

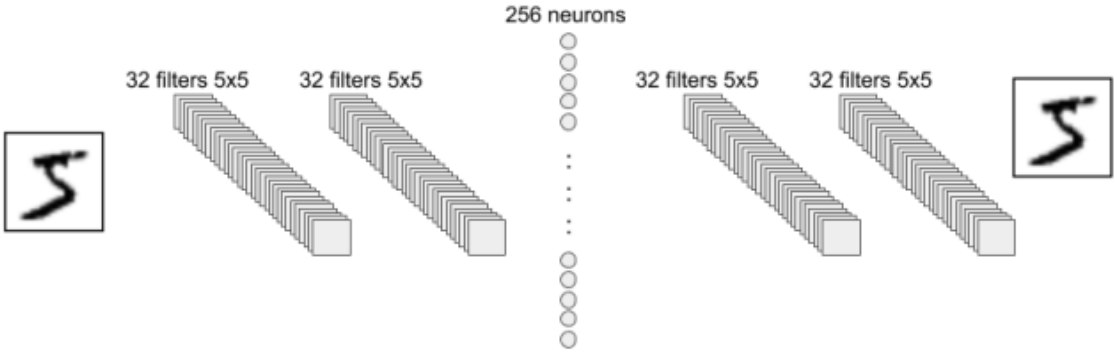

Figure 1: Convolutional Autoencoder.

Generative adversarial networks [1] compose of two networks that compete with each other through a generative and a discriminative task. The generator network attempts to generate images that capture the data distribution based on a random vector seed. The discriminator estimates the probability that the sample came from the training data rather than the generator. The generator attempts to maximize the probability of the discriminator in making a mistake while the discriminator attempts to make the least amount of mistakes. This corresponds to a mini-max two-player game.

$$\min_G \max_D V(D,G) = \boldsymbol{E}_{\boldsymbol{x}\ p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \boldsymbol{E}_{\boldsymbol{z}\ p_{\boldsymbol{z}}(\boldsymbol{z})}[\log D(1 - D(G(\boldsymbol{z})))] \tag{2}$$

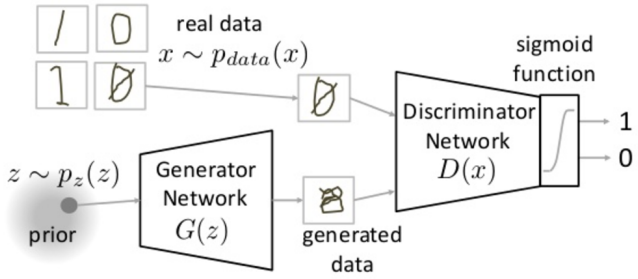

Figure 2: Generative Adversarial Network.

Multi-task learning models are models trained on multiple tasks. Previous work in MTL [3] have shown that training a network on primary and auxiliary tasks result in better performance on the primary task. We hypothesize that this is because the features learned through training on multiple tasks are less brittle and task specific; they are generalizable across multiple domains.

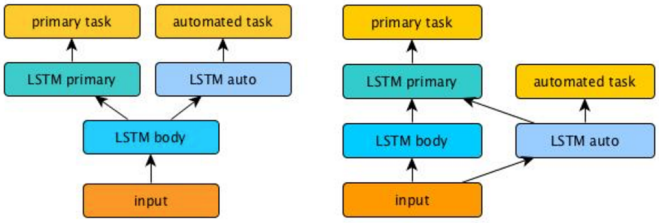

Figure 3: Examples of Multi-Task Learning Models.

3 Robustness Metrics

In order to quantify the robustness of a particular model, we explore several simple metrics that measure a model's accuracy on adversarial examples. These metrics make use of universal adversarial perturbations [5] and common adversarial perturbation generation techniques.

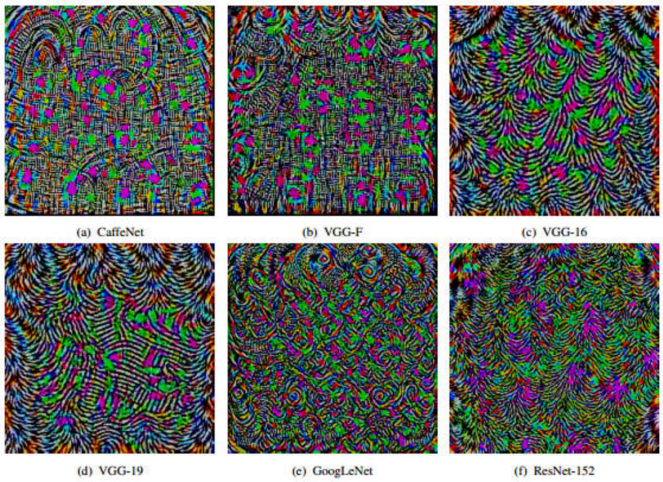

Figure 4: Examples of Universal Adversarial Perturbations Used in Experiments.

Figure 5: Images before and after adding universal perturbations.

Universal (image-agnostic) perturbations have been shown to cause natural images to be misclassified with high probability [5]. These perturbations, when added to an input image, lead to incorrect predictions invariant of image category and classifier type. The universal adversarial perturbations are generated by finding, over each data point in some dataset, X, the minimal perturbation that sends the current perturbed data point to the decision boundary of the classifier. More specifically, we solve the optimization problem

$$\Delta v_i \leftarrow arg\min_r ||r||_2 \ \boldsymbol{s.t.}\ \hat{k}(x_i + v + r) \neq \hat{k}(x_i) \tag{3}$$

Here, x_i corresponds to a particular data point i, v corresponds to a perturbation, and $k(\hat{x}_i)$ corresponds to the classifier prediction for a particular input x_i. By iteratively solving this optimization problem for every datapoint x_i in the data corpus, X, we obtain our universal perturbations. For universal perturbations, we simply measure network robustness by measuring a network's accuracy on a universally perturbed dataset.

Adversarial images can also be generated with a particular target prediction in mind by computing the gradient of the loss function with respect to some particular input image. In particular, for some loss, L, we wish to find some perturbation to the input image, *delta*, that pushes the prediction of the model beyond the decision boundary. Since the adversarial images generated for some model, A, will have a 100% probability of misclassification, we choose a method of cross-testing. Specifically, we generate adversarial images for two models: Model A and Model B. We

test Model A on the adversarial images of Model B and vice versa. We use this metric to compare the adversarial robustness of a baseline model and our augmented models.

$$\begin{aligned} L &= -\sum_j y_j \log p_j \\ L(x + delta) &\approx L(x) + delta \cdot \nabla L(x) \end{aligned} \tag{4}$$

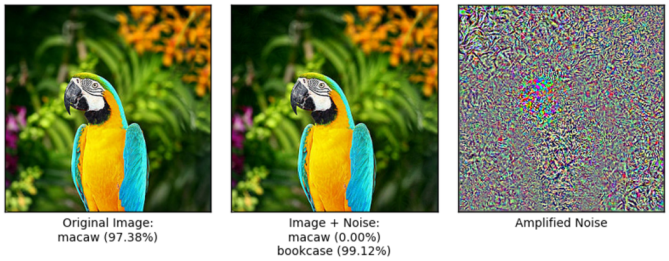

Figure 6: Images before and after adding gradient perturbations.

4 Models

In our experiments, we use a pre-trained Inception model as a control to determine baseline accuracy on adversarial images. In order to increase the robustness of the baseline model, we introduce several additional preprocessing and training methods. Specifically, we use PCA, auto-encoders, convolutional auto-encoders, injecting random noise, down-sampling, adversarial training, and multi-task learning.

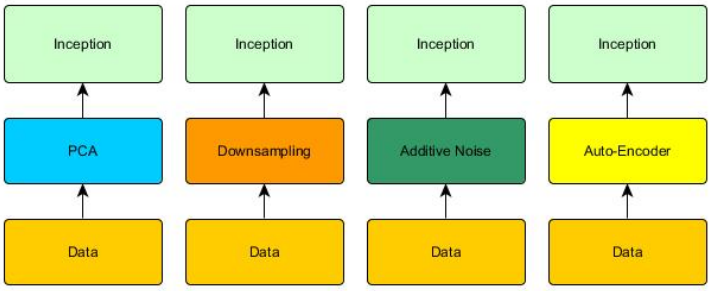

Figure 7: Illustrations of models.

In this paper, we explore reactive models that aim to reduce adversarial noise in images. The PCA model is trained on 10,000 images from a held out set. We extract principal components such that we capture 99% of the dataset variance. Then, we project our test set onto these principal components before projecting them back into image space. This projection decreases the dimensionality of an image and hypothetically reduces the complexity of the data. We believe that PCA should eliminate adversarial perturbations from an image and should increase network robustness by decreasing the complexity of the decision boundary. The Auto-encoder model is trained on 10,000 images from a held out dataset. We use the model to 'denoise' adversarial input data. The hidden layer of the auto-encoder consists of 5,964 units (consistent with the number of principal components we chose for the PCA model). We believe that the auto-encoder should eliminate adversarial perturbations from an image and increase network robustness. The noise model adds varying magnitudes of uniformly distributed noise between a minimum and a maximum value. The down-sampling model downsamples the data by a specified percentage.

5 Experiments and Data

Our datasets consist of three types of data. First, we have the original test data. This data is a set of 2000 images drawn from the testing set of the imagenet dataset [7]. The images are of size 224x224x3. For each category, there will be 2 images in the test set. Second, we have the original data augmented with the universal perturbation filters. Finally, we have the gradient dataset where for each of the original 2,000 images, we choose a target label at random and generate adversarial images based on that target. The gradients are calculated by iteratively for 100 epochs while limiting the movement of each pixel by at most N. N is a chosen parameter.

Figure 8: From left to right: Original Image, Image with Added Noise, Image with Added Universal Perturbations, Image with Added Universal Perturbations and Added Noise.

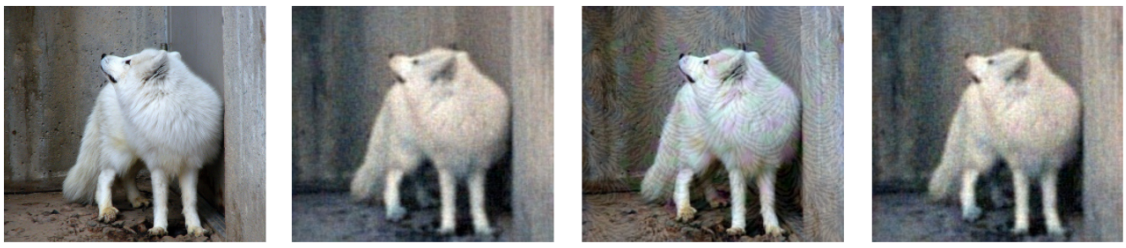

Figure 9: From left to right: Original Image, Image with PCA, Image with Added Universal Perturbations, Image with Added Universal Perturbations and PCA.

Figure 10: From left to right: Original Image, Image with Downsampling, Image with Added Universal Perturbations, Image with Added Universal Perturbations and Downsampling.

For experiment 1, we compare the accuracies of Inception network [8] with the universally perturbed test set against the universally perturbed test set preprocessed through the PCA pipeline.

For experiment 2, we compare the accuracies of the Inception network [8] with the original test set against the universally perturbed test set against the universally perturbed test set preprocessed through the additive-noise pipeline.

For experiment 3, we compare the accuracies of the Inception network [8] with the original test set against the universally perturbed test set against the universally perturbed test set preprocessed through the down-sampling pipeline.

For experiment 4, we compare the accuracies of the Inception network [8] with the original test set against the universally perturbed test set against the universally perturbed test set preprocessed through the auto-encoder pipeline.

Table 1: Final Universal Perturbation Experimental results.

Dataset	Model	Accuracy
`Original`	Inception	75.85%
`Original`	PCA	23.35%
`Original`	Additive Noise	73.05%
`Original`	Downsampling	62.95%
`Universal`	Inception	55.25%
`Universal`	PCA	22.10%
`Universal`	Additive Noise	62.45%
`Universal`	Downsampling	54.30%

Table 2: Gradient Perturbation Experimental results.

Network A	Network B	Accuracy of A	Accuracy of B
`Control`	`Additive Noise`	57.55%	56.10%
`Control`	`PCA`	35.50%	13.80%
`Additive Noise`	`PCA`	41.25%	13.80%

For experiment 5, we compare the accuracies of Inception network [8] with the gradient perturbed test set against the gradient perturbed test set preprocessed through the PCA pipeline.

For experiment 6, we compare the accuracies of the Inception network [8] with the gradient perturbed test set against the gradient perturbed test set preprocessed through the additive-noise pipeline.

For experiment 7, we compare the accuracies of the Inception network [8] with the gradient perturbed set against the gradient perturbed test set preprocessed through the down-sampling pipeline.

For experiment 8, we compare the accuracies of the Inception network [8] with the gradient perturbed set against the gradient perturbed test set preprocessed through the auto-encoder pipeline.

6 Results

Our results are shown in Table 1 and Table 2. In particular, we have found that the auto-encoder model was not very effective at reproducing a useful image due to computational cost constraints. However, our experiments using PCA, additive noise, and downsampling worked quite well.

For PCA, we saw low accuracies for both the original and universal perturbed dataset. However, we noticed that these accuracies were very similar. This result shows that although PCA removes much of the adversarial perturbations from an image, it also removes a lot of the original data. We hypothesize that adding more data to train our principal components would have a positive effect on our network accuracies.

We saw impressive results from our additive noise model. Without additive noise, Inception network suffered an over 20 percent decrease in accuracy down to 55.25%. However, with additive noise, accuracy on universal images was 62.45%. On non-adversarial images, additive noise decreased accuracy by a mere 2.8%.

We saw inconclusive results from our downsampling model. With downsampling, network accuracy on universally perturbed images remains similar to if there were no downsampling. Downsampling also lowers accuracy on non-adversarial tasks by over 10 percent.

Table 3: Downsampling Universal Perturbation Experimental Results .

Downsampling by:	**50%**	**75%**
Accuracy on original	62.95%	70.45%
Accuracy on universal adversarial	54.30%	53.70%

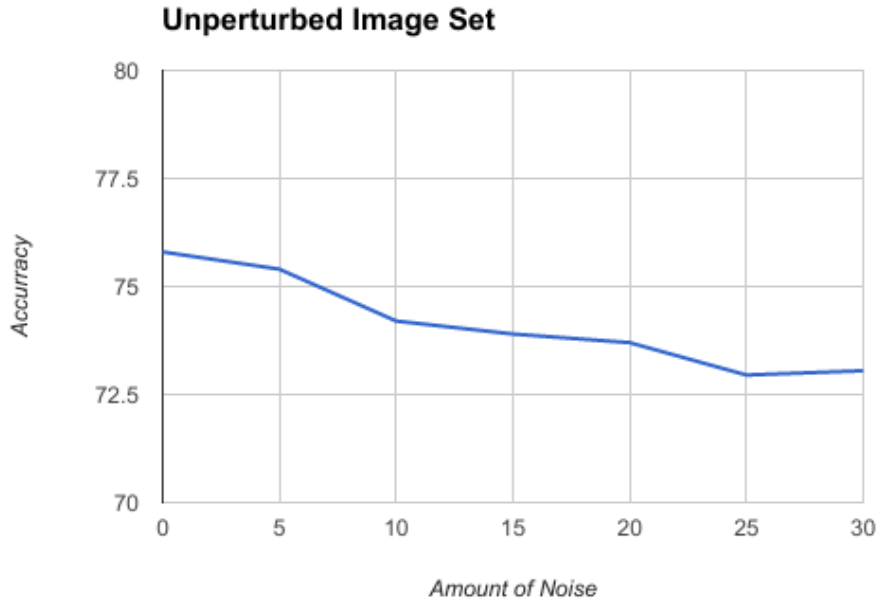

Figure 11: Accuracy of network on original images against increasing random noise magnitude

7 Future Work

In the future, we wish to explore preventative models that aim to increase feature robustness to adversarial inputs. These models include the adversarial training model, which trains on adversarial images generated from a generative adversarial network, and a multi-task learning model, which learns more robust features by training on multiple tasks. For multi-task learning, we drew inspiration from the human brain (which learns tasks in a top-down manner). The features learned by the brain, which are inherently generalizable to multiple-tasks, are not as easily fooled by adversarial inputs.

8 Conclusion

In conclusion, we have learned several things. The most important lesson we learned was that attempting to mask adversarial perturbations works to a certain degree. However, there is always an associated decrease in the accuracy on the original test set when the masking is introduced. Our best method seemed to be the additive noise method as it reduced the accuracy on the original dataset the least but increased accuracy on the universal dataset the most. Of course, further work should be done in learning more robust features. By training on adversarial inputs or applying multi-task learning, we suspect that we would see overall increases in both the accuracy on the original test set and the accuracy on the perturbed test sets.

References

[1] Goodfellow, I., Pouget-Abadie, J., Mehdi, M., Bing, X., David, W., Sherjil, O., Courville, A., and Yoshua, B. Generative adversarial nets. *NIPS* (2014).

[2] Goodfellow, I., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. *ICLR* (2015).

[3] Luong, M., Le, Q., Sutskever, I., Vinyals, O., and Kaiser, L. Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114* (2015).

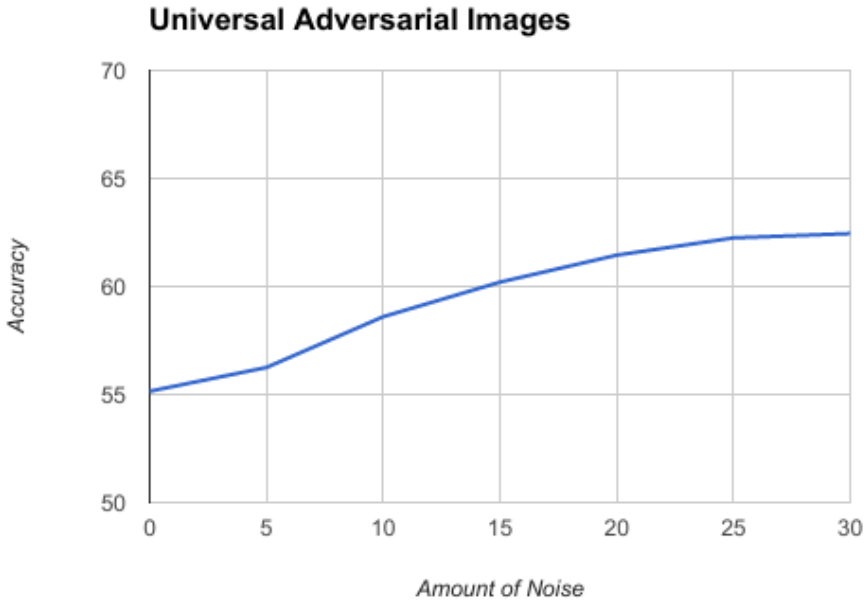

Figure 12: Accuracy of network on universally perturbed images against increasing random noise magnitude.

[4] Masci, J., Meier, U., San, D., and Schmidhuber, J. Stacked convolutional auto-encoders for hierarchical feature extraction. *Proceedings of the 21th international conference on Artificial neural networks* (2011).

[5] Moosavi-Dezfooli, S.-M., Fawzi, A., Fawzi, O., and Frossard, P. Universal adversarial perturbations. *arXiv:1610.08401* (2016).

[6] Rumelhart, D., Hinton, G., and Williams, R. Learning internal representations by error propagation. *Parallel distributed processing, vol. 1* (1986).

[7] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., M., B., Berg, A., and Fei-Fei, L. Imagenet large scale visual recognition challenge. *CoRR, abs/1409.0575* (2014).

[8] Szegedy, C., Ioffe, S., and Vanhoucke, V. Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv:1602.07261* (2016).