Better Generalization with Adaptive Adversarial Training

Sandesh Kamath [1] **Amit Deshpande** [2] **K V Subrahmanyam** [1]

Abstract

An effective method to obtain an adversarial robust network is to train the network with adversarially perturbed samples. Training with adversarially perturbed samples increases the robustness of the network significantly, but affects the generalization of the network to unperturbed points. We propose an adaptive training method which aims to perturb only a *small fraction* of the training samples which aids not only adversarial robustness but also better generalization as compared to perturbing all the training samples. This method is also faster than perturbing the entire training set.

1. Introduction

Neural networks are currently the de-facto method for many classification, object detection and other machine learning tasks. But they have a severe vulnerability as pointed to by Szegedy et al. (2013), Biggio et al. (2017) by which a small, pixel-wise perturbation that is almost imperceptible to the human eye when added to the test data will be grossly misclassified by the network. These small perturbations can be obtained either using box-constrained L-BFGS as proposed by Szegedy et al. (2013) or a quicker method using gradients by Goodfellow et al. (2015), the Fast Gradient Sign Method (FGSM) where the adversarial perturbation is given by $x' = x + \epsilon \text{ sign}(\nabla_x J(\theta, x, y))$, where x is the input, y represents the targets, θ represents the model parameters, and $J(\theta, x, y)$ is the cost used to train the network. Subsequent work has introduced multi-step variants of FGSM, notably, an iterative method by Kurakin et al. (2017) and Projected Gradient Descent (PGD) by Madry et al. (2018). Largely, these adversarial perturbations are studied by searching around the ℓ_∞-ball around the input x with a fixed ϵ, which roughly quantifies the allowed pixel wise perturbation budget for x.

An obvious solution to this problem in obtaining a network which is robust towards such perturbations is to train the network with perturbed samples. This is referred to as adversarial training, where the input samples are perturbed either with FGSM/PGD before using it to train the network. Apart from such simple methods there are many elaborate defense mechanisms proposed for example Papernot et al. (2015), Xie et al. (2017), and others, but such schemes have immediately been shown not to be strong enough by Athalye et al. (2018).

Many have tried to address the adversarial robustness issue from different aspects of the network. Sabour et al. (2017) have tried to address in the architecture level without adversarial training and show the network gains a certain degree of FGSM robustness. The work by Schmidt et al. (2018) claim that to achieve adversarial robustness a much larger input sample set is needed. While Galloway et al. (2018) observe that weight decay itself can give a robust network which generalized better than robustness achieved by adversarial training. While some like Yao et al. (2018), Moosavi-Dezfooli et al. (2018) propose training methods which exploit the curvature information associated with adversarial training to be the fix for better robustness.

More recent work by Tsipras et al. (2018) give a theoretical model to understand the tension between adversarial robustness and generalization and give examples where adversarial robustness can be obtained only by significant reduction in generalization. They show how a classification task on their example could achieve 99% test accuracy but adversarial accuracy could be as low as 10%. They also point out that adversarial training could be an important step towards obtaining a robust network model. But not all classification tasks need 99% accuracy, hence, leading to the question that can we have a effective training method which could obtain good robustness and generalization.

Motivation for the Algorithm
One common issue with all the robustness methods is that the training method is expensive and also has poor generalization. Another known property of training samples is that not all samples lie near the decision boundary. Motivated by this we wanted to exploit such properties and incorporate it as part of the training, there by obtaining a robust network

[1]Chennai Mathematical Institute, Chennai [2]Microsoft Research, India. Correspondence to: Sandesh Kamath <ksandeshk@cmi.ac.in>, Amit Deshpande <amitdesh@microsoft.com>.

ICML 2019 Workshop on Understanding and Improving Generalization in Deep Learning, Long Beach, California, 2019.

with better generalization. Since gradient information is readily available to attacks like FGSM/PGD we too make use of them to obtain a sampling method for the training which would make the network more robust without affecting the generalization and also improve training speed depending on the sampling size.

Our Results We observe that our algorithm achieves better generalization compared to training with all samples adversarially perturbed. The algorithm also achieves robustness comparable to training with all samples perturbed. Currently we get these results with upto 10K samples of the total training samples perturbed without any fine tuning of the algorithm.

2. Adaptive Adversarial Training Algorithm

It is known that adversarially training with all inputs perturbed in each epoch despite giving a more robust network is very expensive and leads to poorer generalization. With the main aim to improve the training for better generalization along with robustness we propose the following algorithm which only perturbs a small sample of the input for each epoch of training. Our algorithm is mainly based on a sampling technique obtained from gradients of the loss function with respect to the data points. Just like the attack methods like FGSM/PGD which exploit the gradients of the loss function with respect to the data to create attacks, we use the gradients at the beginning of each epoch to get a sampling method to obtain the list of candidate points which are to be adversarially perturbed either with FGSM/PGD (we currently report results with FGSM perturbed points only) before training the network.

In Algorithm 1 we take the gradients with respect to the loss function eg. cross entropy for all training samples and obtain their norms e.g. $g_i^{'} = ||g_i||_2$, where g_i is the gradient of the i^{th} sample. Using these norms we obtain a probability distribution on the samples as $p_i = g_i^{'}/\sum_{i=1}^{T} g_i^{'}$, where p_i is the probability associated with each sample. Using these probabilities p_i we draw n samples from the training set and perturb them with FGSM with $\epsilon = 0.3$ for MNIST and Fashion MNIST and $\epsilon = 0.02$ for CIFAR10. This step is repeated during each epoch of training the network. Similarly, we also run a version of the algorithm with the probability distribution on the samples obtained by taking the squared norm of the gradients e.g. $g_i^{'} = ||g_i||_2^2$, where g_i is the gradient of the i^{th} sample.

Algorithm 1: Adaptive Adversarial Training Algorithm

Data: Network N, any input-dependent adversarial attack A, ϵ for A and sample size n to be used for adversarial training, T - Training data size. epochs - E

Result: Network N adversarially trained using n samples perturbed using A attack with ϵ budget.

repeat
- Obtain gradients for unperturbed training data $g_1, ..., g_T$ for the T training samples $s_1, ..., s_T$.
- Obtain probability distribution on the training samples using $g_i^{'} = ||g_i||_2$, for $i = 1$ to T, $p_i = g_i^{'}/\sum_{i=1}^{T} g_i^{'}$.
- OR
- Obtain probability distribution P on the training samples using $g_i^{'} = ||g_i||_2^2$, for $i = 1$ to T, $p_i = g_i^{'}/\sum_{i=1}^{T} g_i^{'}$.
- Obtain n samples from probability distribution P, K is set of indices associated with n. Perturb n samples using A with budget ϵ to get $s_i^{'}$ where $i \in K$.
- Train network with new training data $s_1, ..., s_i^{'}, ..., s_j^{'}, ..., s_T$, where s_i are original samples and $s_i^{'}$ are perturbed samples.

until *E epochs*;

2.1. Robustness achieved by Algorithm 1

In Figures 1 to 3 we plot the test accuracy with PGD attack with changing ϵ budget. Plots in blue, red and green correspond to the robustness obtained with training using Algorithm 1 with 100, 1000, 10000 samples of the training data perturbed, respectively per epoch. Plots in black were trained without adversarial inputs while plots in brown were obtained by training with all the input samples perturbed by FGSM with $\epsilon = 0.3$ for MNIST and Fashion MNIST and $\epsilon = 0.02$ for CIFAR10. We observe that even in the current scheme where we only perturb the samples with FGSM we get better robustness to PGD attacks. This could point towards a possibility that with more fine tuning we could perform adversarial training with a simpler perturbation method like FGSM and achieve stronger robustness to attacks like PGD.

2.2. Generalization with Algorithm 1

We observe that we get better generalization with our training method compared to full adversarial training. For plots in Figure 4 the x-axis represents the number of training samples adversarially perturbed while training. Therefore, the left most point in the plots represents a network trained with unperturbed data while the rightmost point represents a network trained with all its training samples adversarially perturbed. In between points in the plots were obtained by using Algorithm 1 for training the networks. We observe in Figure 4 that we get better generalization by training with Algorithm 1 as the change in test accuracy compared to un-

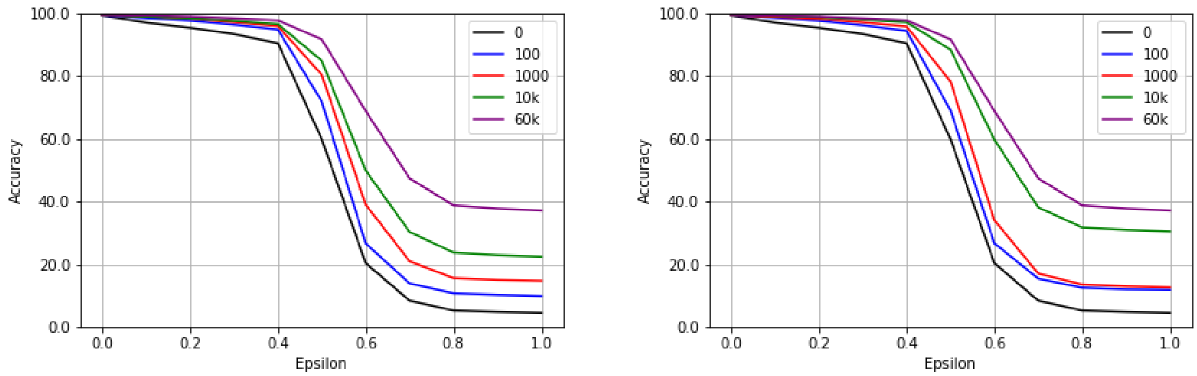


Figure 1: On StdCNN, Test Accuracy for MNIST trained with varying samples perturbed with FGSM ($\epsilon = 0.3$) and attacked with PGD with varying ϵ budget. Sampling method in Algorithm 1 based on (left) Norm, (right) Squared Norm of gradients.

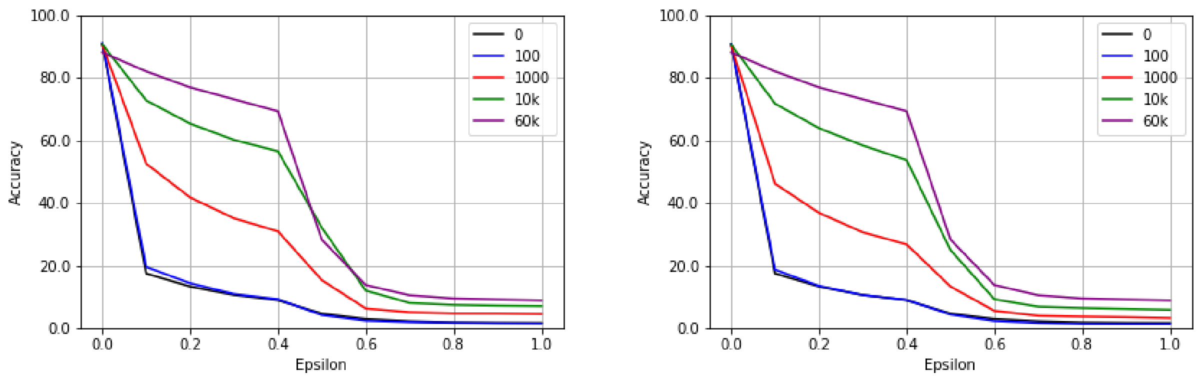


Figure 2: On StdCNN, Test Accuracy for Fashion MNIST trained with varying samples perturbed with FGSM ($\epsilon = 0.3$) and attacked with PGD with varying ϵ budget. Sampling method in Algorithm 1 based on (left) Norm, (right) Squared Norm of gradients.

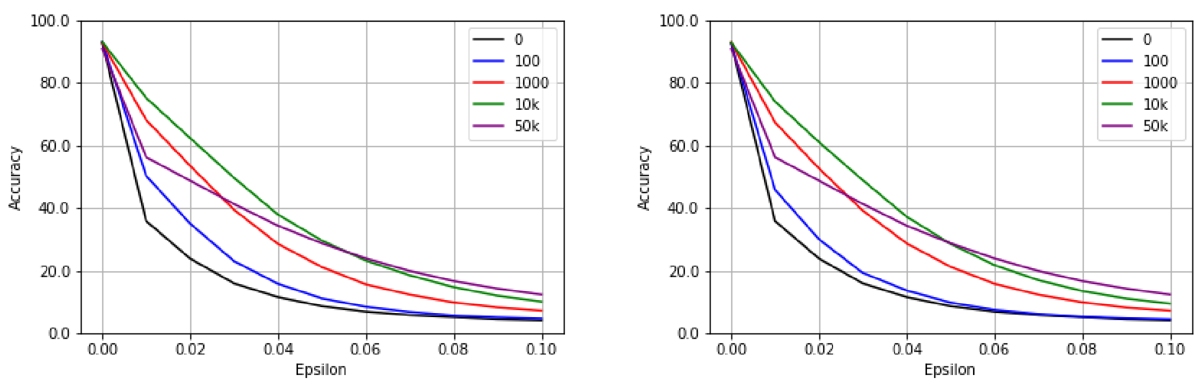
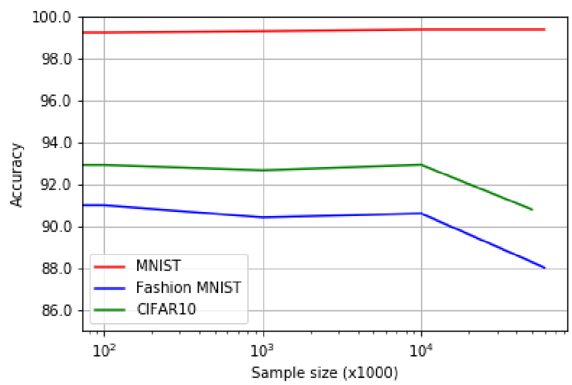


Figure 3: On ResNet18, Test Accuracy for CIFAR10 trained with varying samples perturbed with FGSM ($\epsilon = 0.02$) and attacked with PGD with varying ϵ budget. Sampling method in Algorithm 1 based on (left) Norm, (right) Squared Norm of gradients.

perturbed training is within 0.5% for all the datasets. While a network trained with all samples adversarial perturbed has smaller change (within 0.5%) for MNIST while the is drop upto $2-4\%$ for Fashion MNIST and CIFAR10. These results are similar to that obtained by Tsipras et al. (2018). They too observe smaller drop in generalization for MNIST with adversarial training while for CIFAR10 there is a larger reduction.



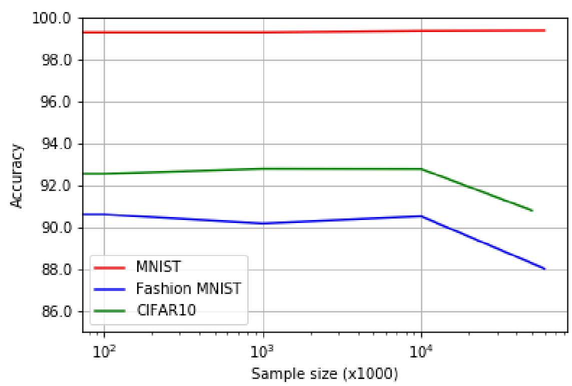


Figure 4: Test Accuracy for Datasets on unperturbed test data, trained with varying samples perturbed with FGSM ($\epsilon = 0.3$ for MNIST/Fashion MNIST, $\epsilon = 0.02$ for CIFAR10). Sampling method in Algorithm 1 based on (top) Norm, (bottom) Squared Norm of gradients.

3. Details of Datasets, Model Parameters and Training Methods

All experiments performed on neural network-based models were done using MNIST, Fashion MNIST and CIFAR10 datasets.

Data sets MNIST dataset consists of 70,000 images of 28×28 size, divided into 10 classes. 55,000 used for training, 5,000 for validation and 10,000 for testing. Fashion MNIST dataset consists of 70,000 images of 28×28 size, divided into 10 classes. 55,000 used for training, 5,000 for validation and 10,000 for testing. CIFAR10 dataset consists of 60,000 images of 32×32 size, divided into 10 classes. 40,000 used for training, 10,000 for validation and 10,000 for testing.

Model Architectures For the MNIST and Fashion MNIST based experiments we use the architecture as given in the Table 1 referred to as StdCNN.

For the CIFAR10 based experiments we used ResNet18 architecture as mentioned in He et al. (2016). Input training data was augmented with random cropping and random horizontal flips by default.

Table 1: Architectures used for experiments

Standard CNN
Conv(10,3,3) + Relu
Conv(10,3,3) + Relu
Max Pooling(2,2)
Conv(20,3,3) + Relu
Conv(20,3,3) + Relu
Max Pooling(2,2)
FC(50) + Relu
Dropout(0.5)
FC(10) + Softmax

Adversarial Training Settings All the networks with or without adversarial inputs were trained for 100 epochs. Perturbations applied to the training data were obtained by FGSM method with $\epsilon = 0.3$ for MNIST and Fashion MNIST and $\epsilon = 0.02$ for CIFAR10. Test time adversarial robustness was checked using PGD for all the datasets irrespective of the training method.

4. Conclusion

Adaptive adversarial training as given by Algorithm 1 achieves better generalization and obtains comparable adversarial robustness even to a strong attack like PGD in comparison to training with all input samples adversarially perturbed.

References

Athalye, A., Carlini, N., and Wagner, D. A. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. *CoRR*, abs/1802.00420, 2018. URL http://arxiv.org/abs/1802.00420.

Biggio, B., Corona, I., Maiorca, D., Nelson, B., Srndic, N., Laskov, P., Giacinto, G., and Roli, F. Evasion attacks against machine learning at test time. *CoRR*, abs/1708.06131, 2017. URL http://arxiv.org/abs/1708.06131.

Galloway, A., Tanay, T., and Taylor, G. W. Adversarial training versus weight decay. *CoRR*, abs/1804.03308, 2018. URL http://arxiv.org/abs/1804.03308.

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

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

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2017.

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

Moosavi-Dezfooli, S., Fawzi, A., Uesato, J., and Frossard, P. Robustness via curvature regularization, and vice versa. *CoRR*, abs/1811.09716, 2018. URL http://arxiv.org/abs/1811.09716.

Papernot, N., McDaniel, P. D., Wu, X., Jha, S., and Swami, A. Distillation as a defense to adversarial perturbations against deep neural networks. *CoRR*, abs/1511.04508, 2015. URL http://arxiv.org/abs/1511.04508.

Sabour, S., Frosst, N., and Hinton, G. E. Dynamic routing between capsules. *CoRR*, abs/1710.09829, 2017.

Schmidt, L., Santurkar, S., Tsipras, D., Talwar, K., and Madry, A. Adversarially robust generalization requires more data. *CoRR*, abs/1804.11285, 2018. URL http://arxiv.org/abs/1804.11285.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. J., and Fergus, R. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., and Madry, A. Robustness may be at odds with accuracy. *CoRR*, abs/1805.12152, 2018. URL http://arxiv.org/abs/1805.12152.

Xie, C., Wang, J., Zhang, Z., Ren, Z., and Yuille, A. L. Mitigating adversarial effects through randomization. *CoRR*, abs/1711.01991, 2017. URL http://arxiv.org/abs/1711.01991.

Yao, Z., Gholami, A., Keutzer, K., and Mahoney, M. W. Large batch size training of neural networks with adversarial training and second-order information. *CoRR*, abs/1810.01021, 2018. URL http://arxiv.org/abs/1810.01021.

A. Complete Adversarial Training with PGD

We include results where in the complete training was done using PGD on all samples with $\epsilon = 0.3$ for MNIST and Fashion MNIST and $\epsilon = 0.02$ for CIFAR10.

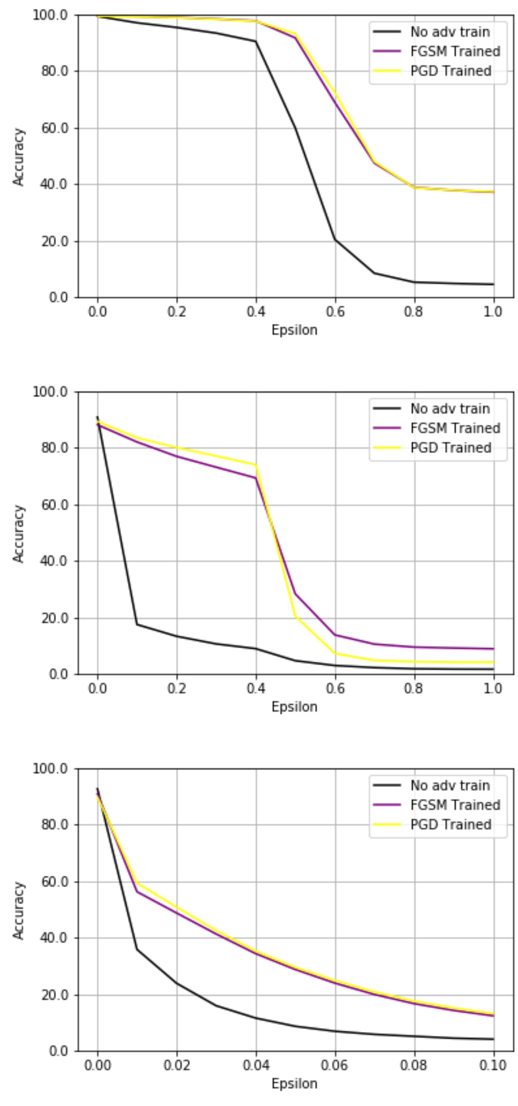


Figure 5: PGD Test Accuracy for Datasets with training using PGD perturbed data with $\epsilon = 0.3$ for MNIST/Fashion MNIST, $\epsilon = 0.02$ for CIFAR10. (top) MNIST, (middle) Fashion MNIST (bottom) CIFAR10.