Effect of Boosting on Adversarial Robustness

Simar Kareer

Jacob Abernethy:_____

Vidya Muthukumar:_____

Contents

1 Abstract

In this paper I explore the relationship between boosting and neural networks. We see that our adaptation of ADABOOST.MM for neural networks results in a consistent increase in accuracy, in the nonadversarial setting. This provides a way to increase the accuracy of any model, without modification to the model itself, making it very easy. In addition, we attempt to use these techniques to improve adversarial robustness, that is, a model's performance while under an adversarial attack. While our ensemble does not have a large increase in adversarial accuracy with the addition of weak learners, the ensemble has increased accuracy on non-adversarial examples. The accuracy of an adversarial model on non-adversarial examples is very important in the real world, and we present an easy way to increase that accuracy.

2 Introduction

Machine learning classifiers are now all around us, informing self driving cars, diagnosing disease, trading stocks and more. As these classifiers begin to take on more mission-critical applications, it will be increasingly important that their predictions are correct. However, more than just being correct, it is important that the classifiers are correct even when an adversary is trying to fool them, perhaps with malicious intent. For instance, an image classifier can easily be fooled into claiming that a picture of a stop sign is in fact a green light just by slightly modifying the image it receives. These modifications can even be so minor that a human cannot tell the difference.. This leads itself to many problems when these models are used in critical applications, for instance, someone could

cleverly design a sticker to put on a stop sign, convincing the car that it is in fact a green light and cause an accident.

The fact that these image classifiers could easily be fooled was brought into the limelight by (Szegedy et al), and was furthered by (Goodfellow et al), in which they began to understand how to create these fooling images, known as adversarial examples. Moreover, Goodfellow's work proposed a method known as the Fast Gradient Sign Method (FGSM) to create these examples. As research progressed, further techniques were created. One notable adversarial attack from (Madry et al) is Projected Gradient Descent (PGD). Using PGD to create adversarial examples performs very well, but takes a long time. In an attempt to speed this up, (Wong et al) created a framework based on Goodfellows FGSM, but it incorporated randomness cleverly to improve performance significantly. Specifically, in standard FGSM, the adversarial perturbation Δ used for minimatch T is initialized to match the adversarial delta calculated for minimatch $T-1$. But in the Wong paper they recognized that this was not effective, and they instead randomly initialize Δ for each minibatch. This approach is very fast to run, but is a bit worse than PGD based training. Thus, to improve upon this, better accuracy would be ideal.

This study aims to improve upon this accuracy by considering boosting. Boosting, first introduced by (Schapire) is a technique for combining multiple models together known as "weak learners" to then produce one very high performing "strong learner". In this study we aim to show how the ADABOOST.MM algorithm (Mukherjee and Schapire) can be used to improve upon both regular and adversarial accuracy. This technique has not been used before on neural networks, and with some modifications it yields strong results. Specifically, we demonstrate that we can increase the accuracy of a state of the art learning algorithm by applying ADABOOST.MM on top of it. This is especially novel, because as new state of the art adversarial learning algorithms are formed, this technique to improve them will remain relevant and easy to use.

3 Literature Review

Adversarial robustness was first put into the spotlight by (Szegedy et al), discovering that neural network based image classifiers could be easily broken. A classic example of this is for a neural network to confidently state that an image of a cat is in fact a laundry dryer by making changes to the cat image that a human could not even detect. This phenomenon opened up a lot of research regarding how to prevent models from being fooled, and why models can be fooled in the first place. A common way to increase the accuracy of a model is to boost multiple smaller models. Boosting is a technique first presented by (Schapire), and was very commonly used in earlier machine learning literature. There is a strong theoretical background that would suggest combining that boosting can improve adversarial accuracy, for instance in (Abernethy et al). Ultimately, looking at adversarial robustness from the lens of boosting has the potential to result in greatly decreased training times and increase accuracy.

Boosting provides a general formulation for combining models, but it is often used in a limited context, leaving much room to explore its place in deep learning. Boosting was introduced by (Schapire) in 1999, and was formulated for the multiclass setting in an algorithm called ADABOOST.MM much later by (Mukherjee and Schapire) in 2011. For those unfamiliar, the binary setting refers to when there are two classes of images as opposed to the multiclass setting where there can be as many as desired. Multiclass boosting is very important, because image classification is almost always multiclass. Although boosting is a relatively old technique in the history of machine learning, it took until 2011 in Mukherjee's paper, for a strong theoretical basis for multiclass boosting to be formulated. There were other attempts like (Hastie et al) which proposed the SAMME multiclass boosting algorithm in 2006. SAMME has since been used frequently, but Mukherjee's paper shows that SAMME is less effective than ADABOOST.MM, which suggests that improvements could be made by using ADABOOST.MM instead. The main paper which attempts to combine boosting and neural networks was (Schwenk and Bengio), however this paper was released in 2000, far before Schapire and Mukherjee were able to refine their algorithm to the multiclass setting. Additionally, there is a more recent paper (Badirli et al) which attempts to gradient boost neural networks, which is a slightly different technique. They do not however offer results on the standard training sets, namely CIFAR10 and imagenet, so it is difficult to evaluate its performance. Nonetheless, neural networks and boosting have not been studied together very commonly, as seen by the lack of literature here. Now that boosting theory has become far more mature, this relationship is worth looking into.

The adversarial robustness problem has largely been approached in the same way since its conception, and progress in the field is marked by generating adversarial examples faster and faster. As noted, the problem of adversarial robustness was first put into view by (Szegedy et al). This paper was followed up by (Goodfellow et al) and (Kurakin et al), in which they describe techniques to train models robustly. These techniques rely on injecting quickly generated adversarial examples into training, so that the model would also learn these examples. The main problem with these techniques is that they didn't reach high enough adversarial accuracy. This led to further research like with (Madry et al), introducing the PGD technique. This achieves very high adversarial accuracy, but requires very long training time, making it a strong baseline of techniques. These experiments lead the deep learning community to ask whether one could achieve PGD level accuracy in the same training time as a standard model, essentially making this the gold standard.

The research following this has begun to strike a middle ground, but is lacking in theoretical basis, requires a lot of hyperparameter tuning, and still does not quite reach PGD levels of accuracy. The paper by (Shafahi et al) is a good example in this space, where they claim that they add no extra time to model training, but they actually have to train on the same data multiple times, making it less efficient than the proceeding (Wong et al) paper. Wong's technique is similar to Goodfellow's FGSM, but instead, the adversarial Δ is randomly initialized for each minibatch, whereas in FGSM it uses the delta from the previous minibatch. When I went and reproduced Wong's results, the fast convergence seemed very tied to the hyperparameters I used. So even though training time

may have been fast, it must have taken them a long time to come upon those hyperparameters. In addition, their accuracy still wasn't as good as PGD. This leaves a gap in the research, seeking for easily tunable hyperparameters, as well as increased accuracy. Boosting provides guarantees that given a sufficient model, you should in time reach very high accuracy. In addition, the individual hyperparameters shouldn't matter very much, since the only requirement is that each individual model does better than average. Thus there is reason to believe that boosting in this context could both increase accuracy and decrease the amount of time needed to tune parameters.

These current drawbacks of adversarial learning are what sparked recent interest in the subject. In (Abernethy et al) the authors attempt to create a general boosting framework that takes into account adversarial robustness. This framework is promising because boosting is described as a zero sum game of two players, and in a sense, adversarial robustness can also be viewed as a two player game where the model and a model fooler play against each other. The main problem here is that the experiments were not very clear or well documented. We attempted to replicate these results on a simple model known as decision trees, however the method Abernethy proposed did not seem to work, opening a large research project to formulate a better adversarial boosting framework. This led to multiclass boosting based on Mukherjee's work, as it is more theoretically grounded. However, one problem that came about from Schapire's framework is that one needs to check which classes a given model can be perturbed to. Interestingly, there was some theoretical work in this area given by (Awasthi et al). The work provides a way to check exactly that, but only in the case of simple models. This however opens another area of research. Not only would this make boosting based approaches much faster, but it would also advance the understanding of neural networks considerably. The research in this area approaches adversarial robustness in a more methodical and theoretical way than (Wong et al), which makes it a worthy direction to explore further.

This study will aim to adapt ADABOOST.MM for neural network training. This will fill the void of boosted neural networks which seems to have been last seriously studied in 2000 by (Schwenk et al). Even in the non-adversarial setting, this would be a novel achievement, because it provides a way for anyone to ensemble their neural networks using ADABOOST.MM. It is particularly elegant, because ADABOOST.MM picks the ideal weights α_i for each weak learner in the ensemble. Then further, we will examine what happens when we boost the state of the art adversarial learning frameworks like (Wong et al), to see if ADABOOST.MM can consistently increase the accuracy for those as well. All in all, this study would combine many facets of deep learning to hopefully use the theoretically grounded work of Schapire and Mukherjee to improve empirically well performing deep learning techniques.

4 Methodology

I will cover methodology for both of our main use cases, non-adversarial as well as adversarial. To reiterate, the adversarial setting refers to modifying images with the intent of fooling the model

(like the cat example from earlier), whereas non-adversarial is just standard learning. The non-adversarial setting aims to show how to use ADABOOST.MM to increase the accuracy of regularly trained neural networks. Specifically we describe a modification to ADABOOST.MM which makes it practical to use on neural networks. This is applicable to any standard learning task, meaning that anyone who wants to improve the performance of their models can use our implementation. Second, the adversarial setting aims to show that ADABOOST.MM applied to adversarially trained networks results in an adversarially robust ensemble.

What we find is that there are strong results in the non-adversarial setting. In the adversarial setting, our ensemble has minor increases on adversarial examples, but interestingly, it has a large increase on non-adversarial samples (while maintaining its accuracy on adversarial samples), which is a novel and useful result.

4.1 Non-adversarial

Our non-adversarial contribution is to display that ADABOOST.MM can consistently boost accuracy on a standard MNIST and CIFAR-10 learning task. Furthermore, it will be important to show that ADABOOST.MM will achieve higher accuracy than any single neural network could. We will provide an implementation of ADABOOST.MM in pytorch[1] which is ready to use with any neural network architecture.

First we will begin with a review of ADABOOST.MM seen in figure 1. Boosting works by training multiple "weak learners" on varying subsets of the data, then all of these weak learners are put together with different weights α_i in an ensemble with the intent of making the ensemble accuracy higher than any individual "weak learner" accuracy. The subset of the data that each weak learner is trained on, and the weight α assigned to each weak learner is determined by the learning algorithm. ADABOOST.MM does this in a slightly more complicated way because unlike most boosting algorithms, this algorithm is truly multiclass. Instead of explicitly having a distribution over examples, we instead have a cost matrix C_t where $C(i, l)$ is the cost associated with predicting class l on training example i. Essentially, each row represents a training example, and each column represents a target class. A cost $C(i, l)$ is positive when $y_i \neq l$ and negative when $y_i = l$.

Our implementation is close to what is listed here, however there are some aspects that are non trivial to adapt for neural networks. The line "Receive weak classifier $h_i \dots$" is non trivial for neural networks. ADABOOST.MM expects the weak learner to minimize its error on the cost matrix C_t. There are multiple ways to go about this, but the way we found to work the best was to convert the cost matrix C_t into a distribution D_t. The rows of the cost matrix C_t gives us information on how much further work is needed for each example. Rows with large absolute value sums will represent samples for which some (example, class) pairs have high cost, so we want future weak learners to

[1] https://github.com/MLTheoryGT/AdversarialBoostingNeuralNets

Algorithm 1 AdaBoost.MM

Require: Number of classes k, number of examples m.
Require: Training set $\{(x_1, y_1), \ldots, (x_m, y_m)\}$ with $y_i \in \{1, \ldots, k\}$ and $x_i \in X$.

- Initialize $m \times k$ matrix $f_0(i,l) = 0$ for $i = 1, \ldots, m$, and $l = 1, \ldots, k$.

for $t = 1$ to T **do**

- Choose cost matrix $\mathbf{C}_t$ as follows:

$$C_t(i,l) = \begin{cases} e^{f_{t-1}(i,l) - f_{t-1}(i,y_i)} & \text{if } l \neq y_i, \\ -\sum_{l \neq y_i} e^{f_{t-1}(i,j) - f_{t-1}(i,y_i)} & \text{if } l = 1. \end{cases}$$

- Receive weak classifier $h_t : X \to \{1, \ldots, k\}$ from weak learning algorithm
- Compute edge δ_t as follows:

$$\delta_t = \frac{-\sum_{i=1}^{m} C_t(i, h_t(x_i))}{\sum_{i=1}^{m} \sum_{l \neq y_i} e^{f_{t-1}(i,l) - f_{t-1}(i,y_i)}}$$

- Choose α_t either as

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 + \delta_t}{1 - \delta_t} \right), \tag{61}$$

or, for a slightly bigger drop in the loss, as

$$\alpha_t = \frac{1}{2} \ln \left(\frac{\sum_{i: h_t(x_i) = y_i} \sum_{l \neq y_i} e^{f_{t-1}(i,l) - f_{t-1}(i,y_i)}}{\sum_{i: h_t(x_i) \neq y_i} e^{f_{t-1}(i, h_t(x_i)) - f_{t-1}(i,y_i)}} \right) \tag{62}$$

- Compute $\mathbf{f}_t$ as:

$$f_t(i,l) = f_{t-1}(i,l) + \alpha_t \mathbf{1}[h_t(x_i) = l].$$

end for

- Output weighted combination of weak classifiers $F_T : X \times \{1, \ldots, k\} \to \mathbb{R}$ defined as:

$$F_T(x,l) = \sum_{t=1}^{T} \alpha_t \mathbf{1}[h_t(x) = l]. \tag{63}$$

- Based on F_T, output a classifier $H_T : X \to \{1, \ldots, k\}$ that predicts as

$$H_T(x) = \underset{l=1}{\overset{k}{\operatorname{argmax}}} F_T(x,l).$$

Figure 1: ADABOOST.MM Pseudocode

focus on these examples. Thus, each weak learner samples its training data by giving priority to rows with larger absolute value sums.

In other words, given examples $x_1, ..., x_n$ and batch size b the idea is to sample a batch of examples with repetition according to distribution D

$$p(x_i) = \frac{\sum_{j=1}^{K} |C_{i,j}|}{\sum_{k=1}^{n} \sum_{j=1}^{K} |C_{k,j}|}$$

Another aspect of ADABOOST.MM that is nontrivial for neural networks is the ensemble prediction step. In order to test the ensemble's adversarial robustness, we need the ensemble prediction

to be differentiable, but ensemble prediction operation $H_T(x) = argmax\ {}^{k}_{l=1} F_T(x, l)$ found in ADABOOST.MM is non differentiable. Thus we replace this ensemble prediction method with a differentiable operation, $H_T(x) = \sum_i \alpha_i F_i(x_i)$ where $F_i(x_i)$ is the pre softmax logit of the ith weak learner. This operator is just a simple linear combination of network outputs, which we found to work fine experimentally.

Given these modifications we were able to run boosting experiments on neural networks using the core ADABOOST.MM algorithm. It is important to understand these results because ADABOOST.MM is proven by (Mukherjee and Schapire), to be an optimal multi-class boosting algorithm, and there is no other work using this algorithm in this setting.

4.2 Adversarial

The boosting algorithm undergoes a slight change in the adversarial setting. The core idea of boosting is that each consecutive weak learner focuses on samples that the previous weak learner incorrectly predicted. However in the adversarial setting, consecutive weak learners should focus on samples which were incorrectly predicted while the learner is under an adversarial attack. So to parallel this in the adversarial setting we propose the following change to ADABOOST.MM. First, we instead calculate our edge δ_t as

$$\delta_t = \frac{-\sum_{i=1}^{m} C_t(i, h_t(x_i + \Delta))}{\sum_{i=1}^{m} \sum_{l \neq y_i} e^{f_{t-1}(i,l) - f_{t-1}(i,y_i)}}$$

where Δ is our adversarial perturbation from a PGD attack. Second, we update f as follows

$$f_t = f_{t1}(i, l) + {}_t 1[h_t(x_i + \Delta) = l]$$

Both of these modifications serve to make our boosting algorithm optimize for the adversarial accuracy of the ensemble rather than the standard accuracy.

In order to measure our accuracy we perform a standard Projected Gradient Descent(PGD) attack on the ensembles. The details of this attack are described in (Madry et al). We use this attack after the addition of each weak learner so we can see how the addition of further weak learners improves the ensemble's accuracy. Note that we are able to perform this attack directly on the ensemble because we modified the ensemble prediction to be differentiable.

5 Discussion / Results

At a high level we see strong results in the nonadversarial case. The proposed idea to sample from our defined distribution D seems to be working, because our nonadversarial accuracy increases as

we add weak learners. Further, we see some increases in accuracy in the adversarial case. This is bench marked by a standard 20 iterations of PGD. However, we do not have very large increases in accuracy, suggesting further modifications to this method may be necessary. Interestingly, the accuracy of our adversarial ensemble on non adversarial examples increases as we add weak learners which is a useful result.

5.1 Non-adversarial Results

Here the goal is to show that ADABOOST.MM is a robust way to increase accuracy on any Neural Network. In addition we will show that, a boosted approach results in higher accuracy than a single model ever could. As we see in figure 2, the accuracy of neural networks grows logarithmically. What we see in figure 3 and 4 is that by boosting, we are able to overcome the stagnant accuracy. In fact we achieve a validation accuracy of over 94% whereas the single model has validation accuracy less than 90%. Thus, this leads me to believe that ADABOOST.MM is a robust ensembling algorithm which is effective for increasing the accuracy of neural networks.

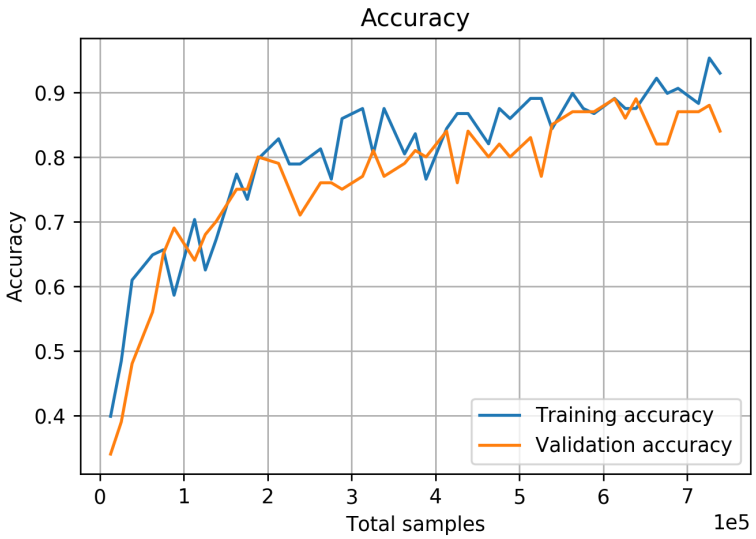

Figure 2: Single model training and validation accuracy

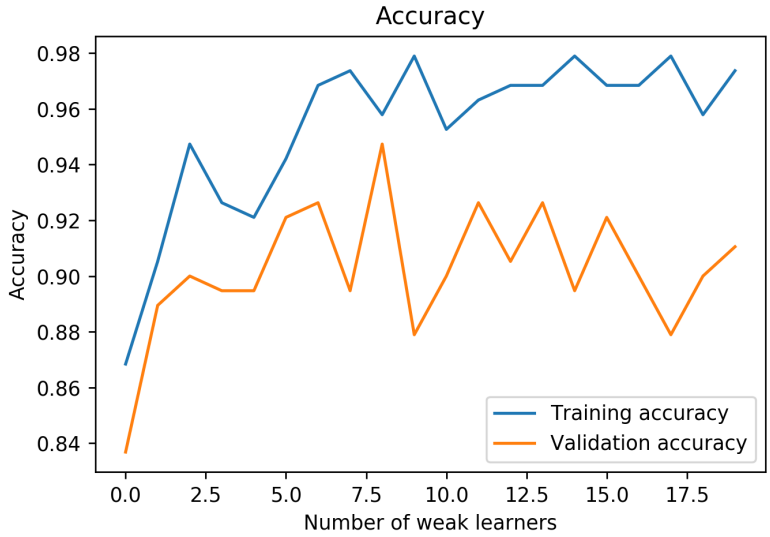

Figure 3: Ensemble training and validation accuracy, 500k samples per WL

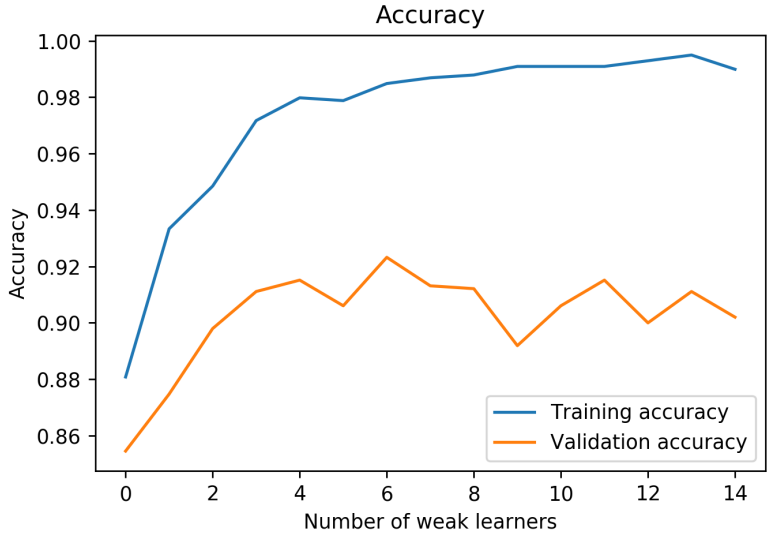

Figure 4: Ensemble training and validation accuracy, 750k samples per WL

An interesting phenomenon we see here is that as we add more weak learners, the gap between

train and validation accuracy seems to grow, which indicates some amount of over-fitting. This phenomena is noted in (Barlett et al) and they indicate that this can happen because the complexity of the ensemble is very high. Notably, to overcome this we may need to use weak learners with smaller architectures in order to generalize better. However, despite this, the validation accuracy still increases, which is ultimately what we optimize for. A potential further direction here would be to understand why this gap occurs to reduce over-fitting.

These results are novel because this is the first study demonstrating the experimental success of ADABOOST.MM as proposed in (Mukherjee and Schapire). In general, there has been a severe lack of research surrounding boosted neural networks. Most notably we can compare to (Schwenk et al) and (Badirli et al), however neither of these papers provide results on the industry standard CIFAR10 dataset like we are.

In addition, one may compare our accuracy of 94% to the current state of the art accuracy from (Foret et al) of 99.7%. However, this comparison is besides the point, because (Foret et al), is using a different, much stronger architecture. Our results serve to show that using an arbitrary architecture, the accuracy achieved by the ensemble is higher than that of the weak learner. In our case we showed that the weak learner plateaus at 90% accuracy, but we are able to achieve 94% accuracy. A further direction would be to replicate these experiments on many neural net architectures to ensure that ADABOOST.MM is a model independent way to boost accuracy.

5.2 Adversarial Results

Here the goal is to show how our boosting algorithm is able to improve upon the (Wong et al) approach. Wong's paper proposes a way to train individual neural networks adversarially. As we see in figure 5, the adversarial accuracy of a single model tapers off over time. Then we see in figure 6, the ensemble is able to slightly increase accuracy, but there is no general upward trend which is concerning. So although this is able to increase the accuracy from 39% to 41%, this still isn't quite as good as just using the (Wong et al) approach, which is able to reach a 45% adversarial accuracy for $\epsilon = 0.127$.

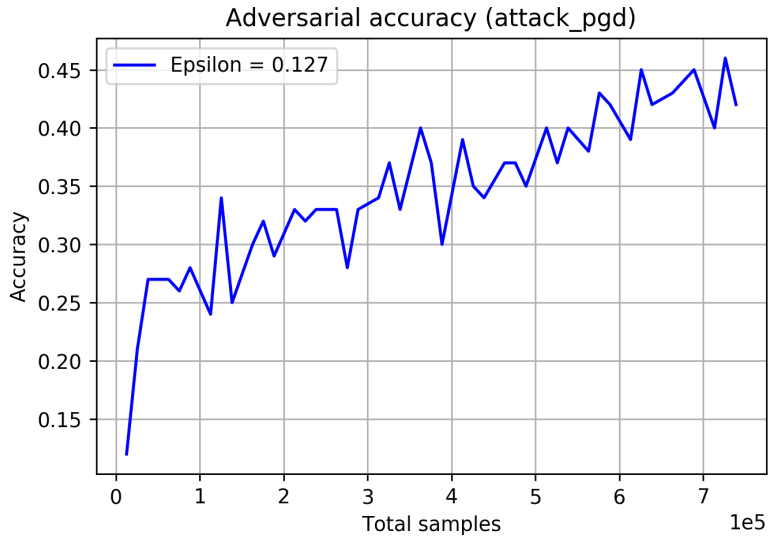

Figure 5: Single model adversarial test accuracy

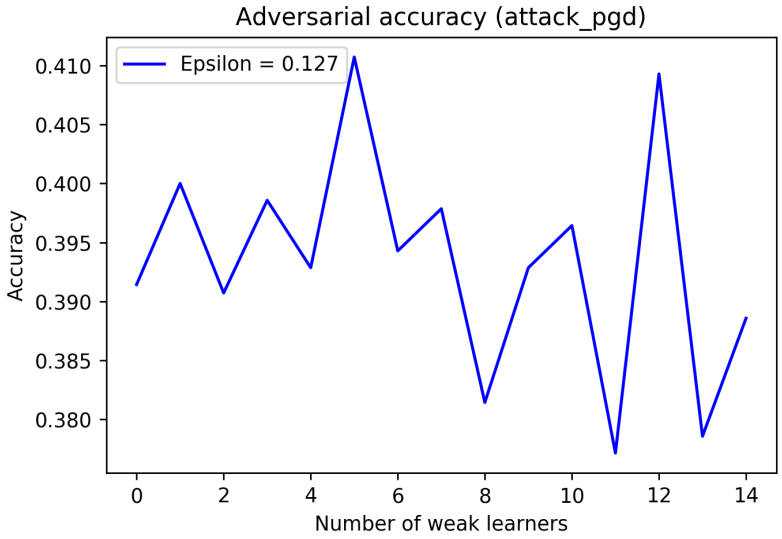

Figure 6: Ensemble adversarial test accuracy

However, as we see in 7, the accuracy of this adversarially trained ensemble on standard samples in-

creases by around 6% which is certainly significant. In the real world, the accuracy of an adversarial model on standard examples is also very important, and here we have shown that ADABOOST.MM is a way to improve that accuracy.

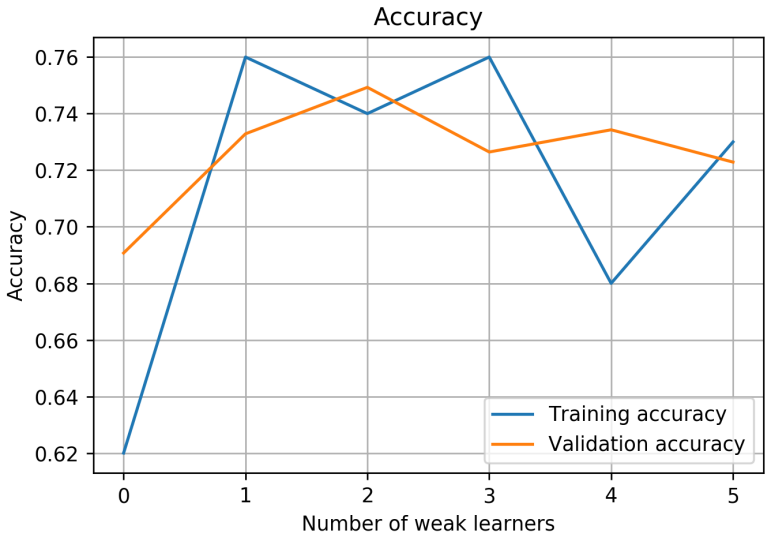

Figure 7: Accuracy of adversarial ensemble on non adversarial samples

So ultimately, this leads us to look into further ways to modify the boosting framework so that we can achieve large increases in adversarial accuracy just as we are able to do in the non-adversarial setting.

6 Conclusion/Future Works

The goal of this paper was twofold. First we aimed to demonstrate a way to use ADABOOST.MM on neural networks and show that we get novel accuracy gains. We were able to show that ADABOOST.MM is a consistent way to increase accuracy in the non-adversarial setting. This is a novel and important result because boosting has not been studied in this setting for a long time. We show that the theoretical work of (Mukherjee and Schapire) is backed up by experimental evidence, specifically in the neural network setting. Moreover, we show the necessary modifications that one should use in order to use ADABOOST.MM for neural networks. This is a significant result because it provides anyone a way to increase the accuracy of their model, simply by applying this boosting algorithm on top of it.

Second, we aimed to make further modifications to ADABOOST.MM to create an adversarially

robust ensemble. The adversarial results, while not stellar, lead us to further look into the work of (Abernethy et al) and find further modifcations to ADABOOST.MM so that we can improve our adversarial accuracy. However, it does provide a useful side result, that being, the accuracy on non-adversarial examples increases significantly.

References

Abernethy, J. (2020). Fast and Provable Adversarial Robustness.

Andriushchenko, M., & Hein, M. (2019). Provably Robust Boosted Decision Stumps and Trees against Adversarial Attacks. *arXiv:1906.03526 [cs, stat]*. Retrieved September 18, 2020, from http://arxiv.org/abs/1906.03526
Comment: Camera-ready version (accepted at NeurIPS 2019)

Awasthi, P., Dutta, A., & Vijayaraghavan, A. (2019). On Robustness to Adversarial Examples and Polynomial Optimization. *arXiv:1911.04681 [cs, stat]*. Retrieved September 18, 2020, from http://arxiv.org/abs/1911.04681
Comment: To appear at NeurIPS2019. 30 pages

Badirli, S., Liu, X., Xing, Z., Bhowmik, A., Doan, K., & Keerthi, S. S. (2020). Gradient Boosting Neural Networks: GrowNet [arXiv: 2002.07971]. *arXiv:2002.07971 [cs, stat]*. Retrieved April 4, 2021, from http://arxiv.org/abs/2002.07971
Comment: Supplementary material starts after references

Bartlett, P., Freund, Y., Lee, W. S., & Schapire, R. E. (1998). Boosting the margin: A new explanation for the effectiveness of voting methods [Publisher: Institute of Mathematical Statistics]. *The Annals of Statistics*, *26*(5), 1651–1686. https://doi.org/10.1214/aos/1024691352

Foret, P., Kleiner, A., Mobahi, H., & Neyshabur, B. (2020). Sharpness-Aware Minimization for Efficiently Improving Generalization [arXiv: 2010.01412]. *arXiv:2010.01412 [cs, stat]*. Retrieved April 4, 2021, from http://arxiv.org/abs/2010.01412

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). Explaining and Harnessing Adversarial Examples. *arXiv:1412.6572 [cs, stat]*. Retrieved September 18, 2020, from http://arxiv.org/abs/1412.6572

Hastie, T., Rosset, S., Zhu, J., & Zou, H. (2009). Multi-class AdaBoost. *Statistics and Its Interface*, *2*(3), 349–360. https://doi.org/10.4310/SII.2009.v2.n3.a8

Ioffe, S., & Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*. Retrieved September 17, 2020, from http://arxiv.org/abs/1502.03167

Kurakin, A., Goodfellow, I., & Bengio, S. (2017). Adversarial Machine Learning at Scale. *arXiv:1611.01236 [cs, stat]*. Retrieved September 18, 2020, from http://arxiv.org/abs/1611.01236
Comment: 17 pages, 5 figures

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2019). Towards Deep Learning Models Resistant to Adversarial Attacks. *arXiv:1706.06083 [cs, stat]*. Retrieved September 20, 2020, from http://arxiv.org/abs/1706.06083
Comment: ICLR'18

Maini, P., Wong, E., & Kolter, J. Z. (2020). Adversarial Robustness Against the Union of Multiple Perturbation Models. *arXiv:1909.04068 [cs, stat]*. Retrieved September 20, 2020, from http://arxiv.org/abs/1909.04068
Comment: ICML 2020 Final Version

Mukherjee, I., & Schapire, R. E. (2011). A theory of multiclass boosting. *arXiv:1108.2989 [cs, stat]*. Retrieved September 18, 2020, from http://arxiv.org/abs/1108.2989
Comment: A preliminary version appeared in NIPS 2010

Schapire, R. E. A Brief Introduction to Boosting. *International Joint Conference on Artificial Intelligence*, 6.

Schwenk, H., & Bengio, Y. (2000). Boosting Neural Networks. *Neural Computation*, *12*(8), 1869–1887. https://doi.org/10.1162/089976600300015178

Shafahi, A., Najibi, M., Ghiasi, A., Xu, Z., Dickerson, J., Studer, C., Davis, L. S., Taylor, G., & Goldstein, T. (2019). Adversarial Training for Free! *arXiv:1904.12843 [cs, stat]*. Retrieved September 18, 2020, from http://arxiv.org/abs/1904.12843
Comment: Accepted to NeurIPS 2019

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., & Fergus, R. (2014). Intriguing properties of neural networks. *arXiv:1312.6199 [cs]*. Retrieved September 18, 2020, from http://arxiv.org/abs/1312.6199

Wong, E., Rice, L., & Kolter, J. Z. (2020). Fast is better than free: Revisiting adversarial training. *arXiv:2001.03994 [cs, stat]*. Retrieved September 18, 2020, from http://arxiv.org/abs/2001.03994