Faster Provable Robustness Through Adversarial Pre-training

Eric Wallace [*1] **Yexin Wu** [*2] **David Li** [*3]

Abstract

Adversarial examples are carefully crafted inputs which cause intentional errors for machine learning models. Recent work has seen a flurry of heuristic defenses—all of which are swiftly broken by ever strengthening adversarial attacks. To end this cat and mouse game, we focus on training neural network models which are provably robust against norm-bounded adversarial attacks. However, existing provably robust methods are hard to tune properly, often resulting in poor model convergence and accuracy. Moreover, many approaches add significant computational overhead to model training, severely limiting neural network size. In this work, we suggest a simple modification—pre-training models with adversarial training—which ameliorates both of these issues. Concretely, we train models using PGD adversarial training before fine-tuning with existing provably robust methods. This simple two stage process provides a 6% increase in provably robust accuracy on CIFAR-10 using less than half the training time of past work.

1. Introduction

Recent advances in deep learning have put neural network models into the forefront of safety critical applications such as self-driving cars, cancer detection, and financial trading. Yet, recent work exposes the fragility of machine learning models: small amounts of carefully crafted noise causes egregious misclassification errors (Szegedy et al., 2014). In computer vision, for instance, small norm-bounded perturbations can flip a model's prediction despite the adversarial image being visually indistinguishable from the original.

A myriad of recent work explores heuristic defenses to adversarial examples. That is, finding a model f such that $f(\mathbf{x} + \delta) = y$ for all permissible perturbations δ. Unfortunately, promising defense mechanisms are often quickly broken by adaptive adversaries (Athalye et al., 2018). To date, the only partially robust heuristic defense is adversarial training (Goodfellow et al., 2015; Kurakin et al., 2017): create adversarial examples during training time to force models to be invariant to these perturbations. A strong adversarial training method is the work of Madry et al. (2018), who train models using a significantly stronger adversary than past work. Concretely, their attack, Projected Gradient Descent (PGD), is an iterative method which generates adversarial examples of considerably higher loss than single-step attacks.

Although exhibiting some degree of robustness, PGD falls short of complete model security. Instead, we focus on training models which are certifiably robust against adversarial examples. Nearly simultaneously, Raghunathan et al. (2018) and Wong & Kolter (2018) released two methods for providing such a certificate. We focus on the method of Wong & Kolter (2018), as it applies out of the box to arbitrarily sized neural models, requires considerably less computational resources, and provides significantly higher model accuracy in practice.

Yet, this provably robust method is still severely limited by its convergence difficulties and computational requirements. In particular, the method requires careful scheduling of the perturbation size ϵ, among other hyperparameters. Moreover, the largest network trained in existing work is a small residual network on the relatively tiny CIFAR-10 dataset.

In this work, we propose a simple method to ameliorate these two issues. Rather than randomly initializing the network before beginning provably robust adversarial training, we pre-train the model using PGD adversarial training.

This simple pre-training approach provides two simultaneous benefits. First, the pre-trained model requires significantly fewer epochs of expensive provably robust training to converge. Moreover, the final network converges to considerably higher robust accuracy as compared models trained from random initialization (Section 3). We follow past work and experiment on MNIST (LeCun, 1998) and CIFAR-10 (Krizhevsky et al., 2014), notably reaching 6% higher robust accuracy on CIFAR-10 using half the computation time of Wong & Kolter (2018).

[*]Equal contribution [1]UID 113422959 [2]UID 115078277 [3]UID 113982019. Correspondence to: Eric Wallace <ewallac2@umd.edu>.

Proceedings of the 35^{th} International Conference on Machine Learning, Stockholm, Sweden, PMLR 80, 2018. Copyright 2018 by the author(s).

2. Provable Robustness and Adversarial Pre-training

This section introduces existing approaches for training models which are provably robust against norm-bounded adversarial examples, as well as the details of our adversarial pre-training approach. We focus on image classification with L_∞ bounded perturbations.

2.1. Provable Adversarial Robustness

Recent work presents methods for training *provably robust* neural network models. In essence, training a classifier which is guaranteed to be robust against any norm-bounded adversarial perturbations on the training set. The core idea behind these methods is to bound the amount of change in the class scores given a norm-bounded change to the input. If the maximum change in the class score is not enough to "flip" the prediction to a different class, the model cannot be broken by a permissible perturbation.

Numerous instantiations of provably robust training algorithms have appeared in recent work (Mirman et al., 2018; Gowal et al., 2018; Wong et al., 2018; Raghunathan et al., 2018). We focus on the method of Wong et al. (2018), though our approach is generally agnostic to the provably robust training algorithm.

2.2. Adversarial Pre-training

We propose adversarial pre-training to improve both the convergence rate and final accuracy of provably robust models. We follow a simple two stage training process. First, we train models using standard adversarial training using the PGD attack of Madry et al. (2018). Next, we use the pre-trained network as an initialization for training a provably robust classifier.

The insight behind adversarial pre-training lies in the successes of pre-training neural architectures (Erhan et al., 2010) and transfer learning (Yosinski et al., 2014). Loosely speaking, PGD adversarial training provides a useful initialization criteria for training provably robust models as the two methods produce networks with similar model parameters. This explanation is further supported by Gowal et al. (2018), who show that PGD adversarial training, the robust training method of Wong & Kolter (2018), and robust training through simple interval bound propagation all learn sparse convolution filters for MNIST and CIFAR-10. Finally, our intuition behind adversarial pre-training is supported by the transferability of adversarial examples across different models trained to do the same task. This suggests that different models may learn similar functions despite being trained independently (Liu et al., 2017).

3. Experiments

We investigate the effectiveness of our adversarial pre-training approach on MNIST (LeCun, 1998) and CIFAR-10 (Krizhevsky et al., 2014) using the same models and experimental setup of Wong et al. (2018). We build our code base using the Convex Adversarial [1] package. We perform experiments on workstations with NVIDIA GTX 1080 GPUs. We use ℓ_∞ norm-bounded adversaries with $\epsilon = 0.1$ for MNIST and $\epsilon = 8/255$ for CIFAR-10. Our pre-training approach uses 20 epochs of PGD adversarial training and 30 epochs of fine-tuning using provably robust training.

We report accuracy using four separate methods: standard non-adversarial training (*Baseline*), PGD adversarial training (*Madry*), the improved provably robust training method of Wong et al. (2018) (*Provable*), and our PGD pre-training with provably robust fine-tuning approach (*Mix*).

We use a training batch size of 30 for all methods when training on MNIST. Each method is trained for 50 epochs with a maximum run-time of 8000 seconds (all models converge before the limit). For the robust method of Wong et al. (2018), we linearly increase ϵ from 0 to 0.1 over training following the original work. Our method does not use any ϵ schedule, i.e. it is set to 0.1 throughout training.

We show learning curves for MNIST in Figure 1 and final accuracy numbers in Table 1. The left plot in Figure 1 shows the standard, non-adversarial accuracy. All approaches converge to over 99% accuracy (less than 1% error). The right plot shows the metric we are optimizing, the provably robust error, calculated using the bounds from Wong et al. (2018). Our method converges to the same robust error as Wong et al. (2018) (0.2% higher accuracy) using 1950 seconds of training instead of 3200 seconds.

We next investigate CIFAR-10, a significantly more challenging dataset for provable adversarial robustness. For CIFAR-10, we use a batch size of 50 for baseline and PGD training and a batch size of 20 for robust training due to the greater memory requirements of robust training. Figure 2 shows the learning curves, with our method (*Mix*) converging to a significantly higher provable adversarial accuracy in about half the computation time.

We finally compare our adversarial pre-training approach, which does not require ϵ-scheduling, to the provably robust training of (Wong et al., 2018) with the epsilon scheduling turned off. This helps elucidate why our pre-training method provides accuracy gains. Without ϵ-scheduling, the method of Wong et al. (2018) quickly converges but sacrifices both non-adversarial accuracy and provably robust accuracy (Figure 3). This highlights the need for careful tun-

[1] https://github.com/locuslab/convex_adversarial

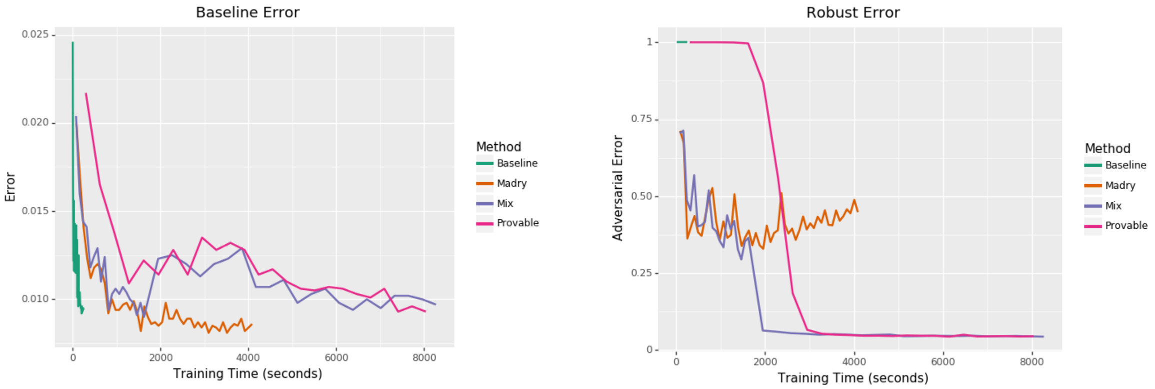


Figure 1. Adversarial pre-training and provably robust fine-tuning (*Mix*) allows faster convergence than provably robust training with a random initialization (*Provable*) on MNIST.

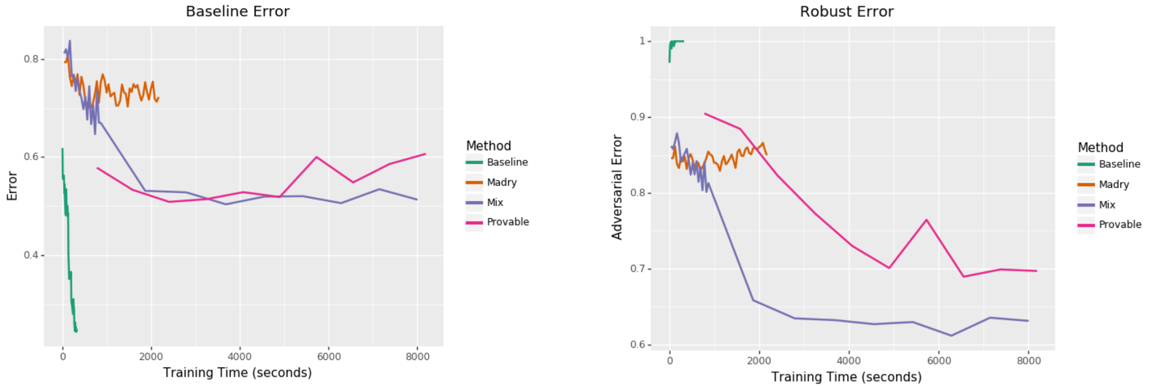


Figure 2. Adversarial pre-training and provably robust fine-tuning (*Mix*) converges to much higher accuracy in about half the computation time compared to provably robust training with randomly initialized models (*Provable*) on CIFAR-10.

Table 1. Final error rates for models trained on MNIST and CIFAR-10. Our pre-training approach, *Mix*, decreases the robust error (calculated using the bounds of Wong et al. (2018)) over randomly initialized models (*Provable*).

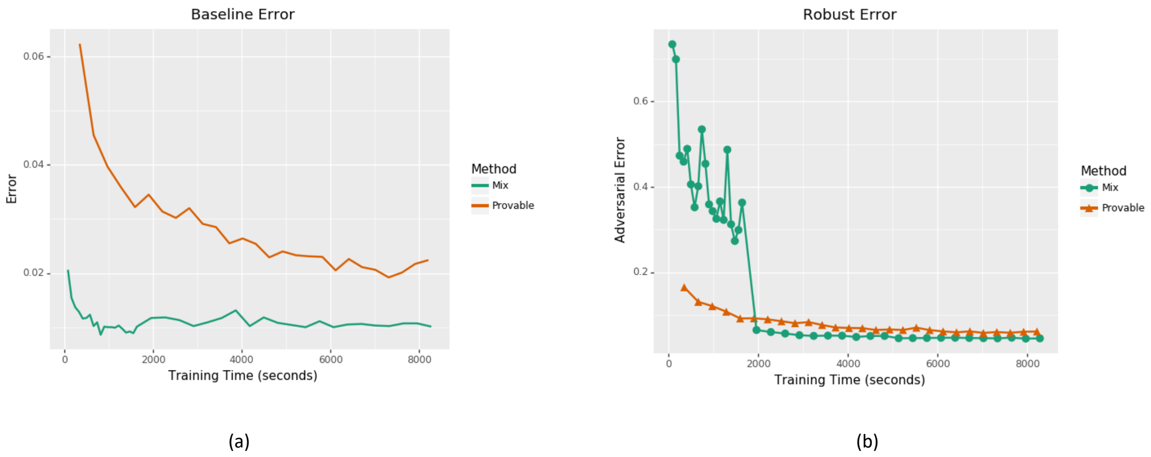
PROBLEM	TRAINING	TEST ERROR	ROBUST ERROR
MNIST	BASELINE	0.94%	100%
MNIST	MADRY	0.86%	44.83%
MNIST	MIX	0.97%	4.37%
MNIST	PROVABLE	0.93%	4.52%
CIFAR	BASELINE	24.96%	100%
CIFAR	MADRY	72.27%	84.97%
CIFAR	MIX	51.33%	63.12%
CIFAR	PROVABLE	60.67%	69.71%

ing of their training method, which our pre-training methods helps abate.

4. Ongoing and Future Work

Pre-training with adversarial examples is one form of model pre-training, though numerous others exist. For instance, a recent approach to adversarial robustness enforces a Lipschitz constraint on the model to guarantee that small changes to the input produce small changes to the prediction. Hence, pre-training models with methods that provably satisfy Lipschitz constraints (Cisse et al., 2017; Yoshida & Miyato, 2017; Qian & Wegman, 2018) is potential alternative to PGD pre-training. Moreover, there is a large space of possible heuristic pre-training methods. For instance, regularizing the gradient (Drucker & Cun, 1992) or adding noise during training time (Kannan et al., 2018).

We also have not fully investigated the limits of pre-training



(a)

(b)

Figure 3. Without ϵ-scheduling, the method of Wong et al. (2018) quickly converges but sacrifices both non-adversarial accuracy and provably robust accuracy. This highlights the need for careful tuning of their training method, which our pre-training methods helps abate.

approaches. For example, we picked a simple fixed PGD pre-training duration of 20 epochs, but one could set the pre-training duration based on the held-out robust error (or similar metrics). Moreover, other heuristics exist such as alternating PGD training and robust training during the fine-tuning stage.

4.1. Co-Designing Models and Algorithms

Our general technique of adversarial pre-training alludes to a principle know as model co-design (Xiao et al., 2018). In essence, rather than theoretically or empirically designing new provably robust *training algorithms*, we can instead modify the *underlying model* to better suit the algorithms we have.

One ongoing work in this direction is to perform neural network compression. We can first train a large network using PGD adversarial training, then compress the network into a smaller version. We can then fine-tune the small network using the provably robust algorithm. The fewer the parameters, the tighter the adversarial bound. Furthermore, fewer parameters accelerates the solution is to the linear program which is run inside the inner loop of Wong et al. (2018).

Another ongoing work is applying the "ReLU stability" idea from Xiao et al. (2018). In essence, every neuron whose lower bound is greater than 0 or whose upper bound is less than 0 is no longer a non-linear function. Hence, the more "stable ReLUs" the model has, the tighter the adversarial bound. We are working to implement their regularization scheme to increase the number of stable ReLUs during adversarial pre-training.

5. Related Work

Adversarial examples are first introduced by Szegedy et al. (2014). This work presents an important observation about deep learning models: although neural networks generalize well, they are fragile to targeted pertubrations. In particular, two different images which are visually indistinguishable cause wildly different outputs from a model. They formalize the adversarial example generation problem: finding the smallest distortion δ such that $x + \delta$ is classified incorrectly. Numerous attacks (Goodfellow et al., 2015; Athalye et al., 2018) and defenses (Madry et al., 2018; Kurakin et al., 2017) have followed. We refer an interested reader to Papernot et al. (2016) for a review.

We introduced provably robust training algorithms briefly in Section 2.1. We review three distinct approaches exist for provable robust training. Aside from the method of Wong & Kolter (2018) and its follow-up Wong et al. (2018) which we use in this work, Gowal et al. (2018) use a simple interval bound propagation approach: compute the bounds for one layer at a time and propagate them through the network.

Raghunathan et al. (2018) present a different approach to provably robust networks. The central idea is that for a particular input to a ReLU network, the model from the input to the class scores is actually linear. As you perturb the input within the epsilon ball, the ReLUs will turn off and on, so the model will represent many different linear functions within the epsilon ball. The best attack an adversary can do to a linear function is to add $\epsilon * sign(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, y))$ (as in FGSM). To limit the effects of adversarial perturbations, they optimize the gradients for every linear segment of the model to be as small as possible.

Finally, we have discussed connections between our work and that of classic neural network pre-training (Erhan et al., 2010) and transfer learning (Yosinski et al., 2014). There is also a deeper connection between the optimization problem of PGD and that of provably robust training. Concretely, PGD adversarial training performs an iterative first-order approximation to the robust classification problem (Madry et al., 2018), whereas provably robust training methods optimize this objective exactly (Wong & Kolter, 2018).

6. Conclusion

In this paper, we investigate the effect of pre-training models with adversarial training prior to provably robust training. Through this simple approach, we achieve significantly higher robust accuracy using less computation time than using randomly initialized networks.

References

Athalye, A., Carlini, N., and Wagner, D. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference of Machine Learning*, 2018.

Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., and Usunier, N. Parseval networks: Improving robustness to adversarial examples. In *International Conference of Machine Learning*, 2017.

Drucker, H. and Cun, Y. L. Improving generalization performance using double backpropagation. 1992.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., and Vincent, P. Why does unsupervised pre-training help deep learning? 2010.

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

Gowal, S., Dvijotham, K., Stanforth, R., Bunel, R., Qin, C., Uesato, J., Mann, T., and Kohli, P. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

Kannan, H., Kurakin, A., and Goodfellow, I. Adversarial logit pairing. 2018.

Krizhevsky, A., Nair, V., and Hinton, G. The cifar-10 dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 2014.

Kurakin, A., Goodfellow, I., and Bengio, S. Adversarial machine learning at scale. In *International Conference on Learning Representations*, 2017.

LeCun, Y. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

Liu, Y., Chen, X., Liu, C., and Song, D. Delving into transferable adversarial examples and black-box attacks. In *International Conference on Learning Representations*, 2017.

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

Mirman, M., Gehr, T., and Vechev, M. Differentiable abstract interpretation for provably robust neural networks. In *International Conference of Machine Learning*, 2018.

Papernot, N., McDaniel, P., Sinha, A., and Wellman, M. Towards the science of security and privacy in machine learning. *arXiv preprint arXiv:1611.03814*, 2016.

Qian, H. and Wegman, M. N. L2-nonexpansive neural networks. *arXiv preprint arXiv:1802.07896*, 2018.

Raghunathan, A., Steinhardt, J., and Liang, P. Certified defenses against adversarial examples. In *International Conference on Learning Representations*, 2018.

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

Wong, E. and Kolter, J. Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference of Machine Learning*, 2018.

Wong, E., Schmidt, F., Metzen, J. H., and Kolter, Z. Scaling provable adversarial defenses. In *Neural Information Processing Systems*, 2018.

Xiao, K. Y., Tjeng, V., Shafiullah, N. M., and Madry, A. Training for faster adversarial robustness verification via inducing relu stability. *arXiv preprint arXiv:1809.03008*, 2018.

Yoshida, Y. and Miyato, T. Spectral norm regularization for improving the generalizability of deep learning. *arXiv preprint arXiv:1705.10941*, 2017.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. How transferable are features in deep neural networks? 2014.

A. Appendix A: Model Architecture

Our MNIST model follows the small model of (Wong et al., 2018). We use a four layer neural network with two convolutional layers and two fully-connected layers. The first convolutional layer receives a $28 \times 28 \times 1$ MNIST image and reduces it to $14 \times 14 \times 16$ tensors using $4 \times 4 \times 1$ kernels with a padding of 1 and a stride of 2. The second convolutional layer turns the $14 \times 14 \times 16$ tensors into $7 \times 7 \times 32$ tensors using $4 \times 4 \times 16$ kernels with a padding of 1 and a stride of 2. The first fully-connected layer consists of 100 nodes outputting a 100 dimension vector for each example. The last layer outputs a 10 dimension vector representing scores for each possible digit. Each layer, except the last, uses a ReLU activation.

Similarly, our CIFAR-10 model also consists of two convolutional layers and two fully-connected layers. The first convolutional layer receives $32 \times 32 \times 3$ color CIFAR-10 images and reduces it to $16 \times 16 \times 16$ tensors from $4 \times 4 \times 3$ kernels with a padding of 1 and stride of 2. The second convolutional layer outputs $8 \times 8 \times 32$ tensors from $4 \times 4 \times 16$ kernels with a padding of 1 and stride of 2. The first fully connected layer outputs a vector of dimension 100. The second fully connected layer outputs a vector of size 10 with each element representing a score for possible each class.