Improving the affordability of robustness training for DNNs

Sidharth Gupta
University of Illinois at Urbana-Champaign
gupta67@illinois.edu

Parijat Dube
IBM Research
pdube@us.ibm.com

Ashish Verma
IBM Research
ashish.verma1@ibm.com

Abstract

Projected Gradient Descent (PGD) based adversarial training has become one of the most prominent methods for building robust deep neural network models. However, the computational complexity associated with this approach, due to the maximization of the loss function when finding adversaries, is a longstanding problem and may be prohibitive when using larger and more complex models. In this paper we show that the initial phase of adversarial training is redundant and can be replaced with natural training which significantly improves the computational efficiency. We demonstrate that this efficiency gain can be achieved without any loss in accuracy on natural and adversarial test samples. We support our argument with insights on the nature of the adversaries and their relative strength during the training process. We show that our proposed method can reduce the training time by a factor of up to 2.5 with comparable or better model test accuracy and generalization on various strengths of adversarial attacks.

1. Introduction

With the impressive performance of deep neural networks in multiple tasks, these models are being deployed in a variety of domains including entertainment, finance, healthcare, safety and security. However, a peculiar characteristic of these models is their extreme sensitivity to specially designed imperceptible small perturbations to the input data, called *Adversarial Samples*. For example, it is possible to make an otherwise highly accurate neural network classifier misclassify by adding a small imperceptible non-random perturbation to a test image [21].

Out of the various approaches proposed to improve the robustness of deep neural network models, *Adversarial Training* is found to be most effective [6, 21, 11, 24, 3, 7, 5]. In a typical adversarial training procedure, at each training iteration adversarial versions of the training dataset are generated and are then used to train the model to increase its robustness on such samples [6].

There are multiple methods to generate adversarial samples and these are known as the type of attack [6, 4, 18, 12, 11]. The general principle behind most attacks is to identify data points in the input space which are imperceptibly close to the training data points but result in the highest loss function value. This results in the following general formulation

$$\max_{\tilde{\boldsymbol{x}}} \mathcal{L}(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), y) \quad \text{s.t.} \quad d(\boldsymbol{x}, \tilde{\boldsymbol{x}}) \leq \epsilon \tag{1}$$

where $\mathcal{L}$ is the loss function between the output of a classifier $f_{\boldsymbol{\theta}}$ and the actual label y, d is a distance metric between the original training sample $\boldsymbol{x}$ and the corresponding adversarial sample $\tilde{\boldsymbol{x}}$ and ϵ is a predetermined threshold. Different attack methods are designed by choosing different techniques to maximize Equation (1) and choosing different distance metrics.

In this work, we use one of the most prominent adversarial training frameworks proposed by Madry et al. [11] which incorporates a min-max optimization of the overall objective function with respect to adversarial samples and model parameters. It has been shown to be one of the most effective training methods [1] and is also used as a state-of-the-art benchmark [7, 24]. The models trained through this framework are shown to be robust against strong PGD attacks [9], with MNIST [10] achieving about 90% accuracy on adversarial samples. Although we have tested our approach with the framework proposed by Madry et al., our approach is generic and can be applied to other frameworks as well.

Adversarial training by Madry et al. [11] aims to solve the following min-max robust optimization problem,

$$\min_{\boldsymbol{\theta}} \rho(\boldsymbol{\theta}); \quad \rho(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x},y)\sim\mathcal{D}} \left[\max_{\|\tilde{\boldsymbol{x}}-\boldsymbol{x}\|_\infty \leq \epsilon} \mathcal{L}(f_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}), y) \right], \tag{2}$$

where $f_{\boldsymbol{\theta}}(.)$ is a deep neural network with parameters $\boldsymbol{\theta}$ and $\tilde{\boldsymbol{x}}$ is an ϵ-ball ℓ_∞ adversarial sample of natural sample $\boldsymbol{x}$ having class label y. $\tilde{\boldsymbol{x}}$ is generated using Projected Gradient Descent (PGD) [9] as explained below in Equation (3). The true data comes from distribution $\mathcal{D}$ and $\mathcal{L}(\cdot)$ is the loss function. The maximization seeks to find an adversary which maximizes the loss and the minimization seeks to find the model parameters that minimizes the loss due to the adversary. The minimization is solved via standard neural network optimization techniques. The maximization is

MODEL ARCHITECTURE	NATURAL TRAINING	REGULAR ADVERSARIAL TRAINING
ResNet-50	1.1 HOURS	6.8 HOURS
WideResNet-28x10	2.2 HOURS	14.7 HOURS

Table 1. Natural and adversarial training times in hours for CIFAR-10 image dataset classification. All training runs are done for 155 epochs with two NVIDIA V100 GPUs, the same training hyper-parameters and the standard CIFAR-10 training dataset split. Adversarial training is done using 10-step projected gradient descent adversaries with ball size $8/255$ and step size $2/255$.

typically solved using Projected Gradient Descent (PGD),

$$\boldsymbol{x}^{t+1} = \Pi\left(\boldsymbol{x}^t + \alpha\,\mathrm{sign}(\nabla_{\boldsymbol{x}^t}\mathcal{L}(f_{\boldsymbol{\theta}}(\boldsymbol{x}^t), y))\right), \qquad (3)$$

where α is the step size and $\Pi(\cdot)$ projects the result of the gradient step into the ϵ-ball around the original sample, $\boldsymbol{x}$. It is a T-step PGD attack if $t+1=T$. As the value of T increases, the adversaries become stronger which results in a greater chance of misclassification by a trained model.

We are required to make T forward and backward passes of the deep neural network to complete the iterative procedure in Equation (3). This is a significant computational overhead to the training process and can be prohibitive for large models. This is especially relevant because models with larger capacity have been shown to be more robust [11]. Furthermore, some adversarial defense strategies involve training an ensemble of neural networks [22, 19, 13] and any per network training time reduction can scale quickly.

Table 1 confirms that the training time for adversarial training is significantly higher than natural training when classifying the CIFAR-10 image dataset [8] for two popular architectures. The same hardware, training dataset, training hyperparameters and total number of epochs were used to obtain all timings. Due to the popularity of adversarial training, we are motivated to improve its computational cost. Specifically, we question the need to always perform the expensive maximization, Equation (3), in Equation (2). We also wish to maintain the classification accuracy on both natural and adversarial test samples. Our method and insights are not restricted to this particular adersarial training framework and can be extended for other variants of adversarial training [3, 4].

Our contribution. We show that adversaries generated in the initial phase of adversarial training are treated more or less like natural samples by the final model. This means that they have minimal influence on the learned model and in fact may even hurt the final accuracy. As a consequence of this finding, we demonstrate that using natural, and not adversarial samples, for the initial phase of training gives comparable model test accuracy. We further show that this initial phase lasts for a specific number of epochs and therefore a fully converged naturally trained model cannot be taken as the initial phase. Importantly, our proposed training method significantly reduces the training time because the expensive maximization in (2) is not required for a large fraction of the training epochs.

2. Related work

The problem of overfitting and reduced generalization with regular adversarial training has been pointed out by several recent works including [3, 24, 15, 14]. It has been attributed to training with strong adversarial samples from the beginning [3, 24]. This has led to curriculum adversarial training [3] in which the strength of adversaries (as measured by the number of PGD steps in [11]) in a training batch is gradually increased as training progresses. Wang et al. further explained that the number of steps is not the right measure of adversary strength and they defined a new criterion by linking the strength of adversarial samples to the convergence of the inner maximization in Equation 2 [24]. These works suggest that using lower quality adversaries during the initial phases of training helps improve accuracy. However, their focus is not computational efficiency and still requires adversaries to be computed in the initial phases of training. We also note that generalization of adversarially robust classifiers cannot be improved by algorithmic design alone as it is inherently tied to the complexity of the underlying data distribution [14].

The majority of research so far has focused on improving adversarial robustness through designing defenses and less consideration has been given to computational requirements and scalability, both of which are concerns with complex networks and big datasets. Shafahi et al. [16] makes use of gradient information from model updates to reduce the overhead of generating adversarial samples and claims about 7x improvement in time for CIFAR-10 over regular adversarial training. However, multiple runs of their method may be required to select the parameters for their method which reduces the overall time saving. This is particularly important as they note an incorrect parameter drastically reduces natural accuracy. In a different line of work, ensembles of pretrained neural networks were used to reduce the computational cost of regular adversarial training in [22] for ImageNet. The training was done by interspersing the adversarial samples generated using the trained network with adversarial samples generated from the ensemble, thereby creating a mixture of black-box and white-box attacks. Our approach can be used to efficiently develop pretrained networks for ensembles and can also be used to delay the generation of adversarial samples in the method proposed by Shafahi et al.

Adversarial training has also been accelerated by formulating it as a discrete-time differential game [27]. A recent

work by Wong et al. [25] claimed that adversarial training using FGSM with appropriate random initialization can be as effective as PGD thus making adversarial training much less costly. Though our work is focused on improving the efficiency of PGD based adversarial training, any method such as the one proposed by Wong et al. which requires adversarial samples to be generated during training can be made more efficient by using our method which delays the point at which adversarial samples are required.

The trade-offs between generalization and robustness are inherent [23, 20]. Recently, there has been work suggesting that certain values of adversarial training hyperparameters may have unexpected benefits [5]. Duesterwald et al. used traditional approaches [2] to tune adversarial training hyperparameters to achieve different tradeoffs between robustness and accuracy. The appeal of our approach lies in its simplicity as its devoid of any (costly) hyperparameter optimization sub-steps.

Switching from natural samples to adversarial samples can be interpreted as a transfer learning scenario. The current literature considers the scenario where the model is adversarially trained on both the source and target datasets to obtain better robustness [7]. Or where the model is adversarially trained on the source dataset and naturally trained on the target dataset [17]. In these approaches, while training on the target datasets may be fast because many epochs are not required, training the model on the source dataset is costly and we must efficiently make a model robust from the beginning.

3. Delayed adversarial training

3.1. Usefulness of adversaries from the initial epochs of regular adversarial training

In the adversarial training framework natural training samples are replaced by their adversarial counterparts and used as training data from the start of training. Adversarial samples generated using Equation (3) are dependent on the evolving model parameters which are initially randomly initialized. Generally at initialization, the model's parameters are relatively far from their final values. Therefore the adversarial samples generated in the initial training iterations are quite different from the adversaries that the model will face towards the end of training because they would not maximize the adversarial loss in Equation (2) with the final model parameters. Hence, initial non-maximizing adversarial samples are weak adversaries for the final model and may not help improve robustness as was also noted by Wang et al. [24]. Yet generating them adds computational overhead and they influence the model.

To investigate the usefulness of the initial adversarial samples we perform regular adversarial training on a model and test the final model with adversaries that are generated from the model's parameters at previous epochs. We use CIFAR-10 and the test images come from the dataset's standard train-test split which are never seen during training. We use the WideResNet-28x10 architecture [26] and adversarial test samples are generated using Equation (2) and (3) with $T = 10, \epsilon = \frac{8}{255}$ and $\alpha = \frac{2}{255}$. This is a standard architecture and set of adversarial sample hyperparameters for CIFAR-10 in the adversarial training literature [11, 7].

Figure 1 (Top) plots the classification accuracy when the final model is tested against adversaries from previous epochs. The green and red lines indicate the final natural and adversarial test accuracy of the models. We can see that adversarial samples from the initial epochs are treated more or less like natural samples by the final model. The adversaries become more potent as the model parameters start to approach their final value and the model starts to stabilize. The sharp increase in the adversary strength corresponds to the first learning rate drop as there is little change in model parameters from here on. We see that samples from the initial phase of training have limited impact on improving robustness. In spite of this, the computationally expensive maximization (3) is performed to generate these samples for training and these samples are allowed to influence the model parameters. Figure 10 in the supplementary material shows that the same observation can be made for the ResNet-50.

3.2. Delayed Adversarial Training (DAT): Initial training with natural samples

As adversarial samples are computationally expensive to generate and not useful in the initial phase of training, we would like to replace them with natural training samples until the model reaches some form of stability. Natural samples for the initial training phase are an obvious choice as we also want the trained model to perform as well as possible on these samples.

Training on natural samples does not require the costly maximization in Equation (3) and therefore significantly reduces the training time. Also the time taken to get the model to a state where adversarial samples are relevant reduces. Furthermore, the model is expected to correctly classify natural samples, but they are never seen in regular adversarial training.

Now we investigate how to quantify the "initial phase" of training and determine a switch point from training on natural samples to training on adversarial samples. In Figure 1 (Bottom), we plot the training loss evolution of full *natural* training for the WideResNet-28x10 architecture with CIFAR-10. We see that the natural training loss flattens roughly at the same epoch at which the strength of the adversaries previously trained become reasonable in the top figure. This is due to the fact that at this point the model parameters have started to converge towards a local mini-

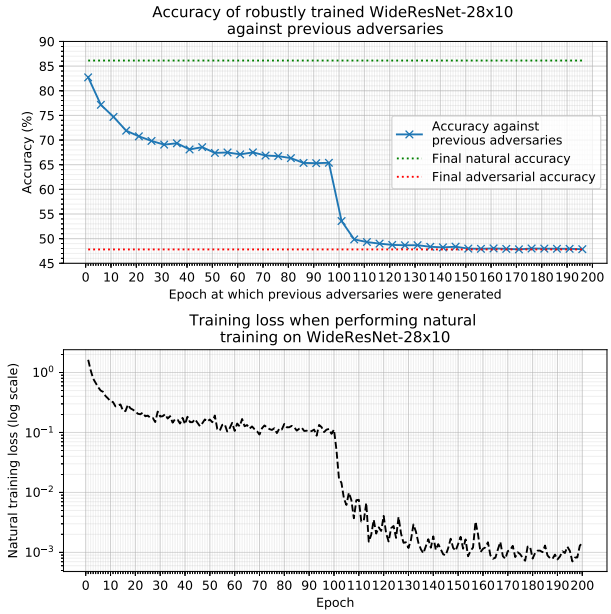

Figure 1. Top: Accuracy of a fully adversarially trained WideResNet-28x10 model when tested with adversaries that are generated using the model's parameters at previous epochs. The model is trained using the CIFAR-10 dataset. The green and red lines show the final model's test accuracy on natural and adversarial samples. Training and test samples are generated using (2) and (3) with $T = 10, \epsilon = \frac{8}{255}$ and $\alpha = \frac{2}{255}$. Stochastic gradient descent is used and the drop is due to a learning rate decrease; Bottom: The training loss (logarithmic scale on the right) when performing *full* natural training on WideResNet-28x10 with CIFAR-10 samples.

mum. This motivates us to use the training loss on natural samples to determine when to switch to training on adversarial samples. We will use this motivation to propose an algorithm for a modified version of the adversarial training process later in this section.

Before coming up with an exact algorithm for switching, we explore our proposition with observations on a set of experiments where we switched from natural training to adversarial training at various points. This is shown in Figure 2. Here we trained the WideResNet-28x10 on the CIFAR-10 dataset. We performed the initial phase of training with natural samples and then switched to adversarial samples generated using PGD at various switching points. Training and test adversaries are generated using (2) and (3) with $T = 10, \epsilon = \frac{8}{255}$ and $\alpha = \frac{2}{255}$. The learning rate drops after epochs 100, 105 and 150. We show how the accuracy on the natural and adversarial test sets varies during regular adversarial training and adversarial training with different switching epochs.

Figure 2 provides evidence that performing the initial phase of training with natural samples is indeed a promising approach. We see that except for continuing the training with natural samples for too long (switching after the learning rate drops), we in fact get better performance than regular adversarial training. The training time is also reduced significantly as computing the expensive maximization in Equation (3) is not required until after the switch is made. Before switching, the adversarial accuracy is almost 0% because the model is not robust and is undergoing natural training. Note, that the adversarial accuracy rises very quickly with all switching points as compared to the case of regular adversarial training where adversarial samples are used from the very beginning (the blue curve). We observe similar curves for CIFAR-10 and CIFAR-100 classification with the ResNet-50 and ResNet-18 architectures in Section B of the supplementary material.

Based on the above observation, we first provide a basic algorithm (Algorithm 1) for our modification to the adversarial training method. We call it *Delayed Adversarial Training (DAT)*. We introduce a new hyperparameter for delayed adversarial training, *i.e.*, the epoch after which training data should be switched from natural samples to adversarial samples. We call this hyperparameter, the switching point, S. In Algorithm 1, S must be chosen in advance. The value of this hyperprameter lies between 0 and the total number of training epochs, N. For maximum computational saving, we would want to use natural samples for as long as possible and hence S should be as close to N as possible. However, as we have seen in Figure 2, switching too late affects the final accuracy (both on natural samples and on adversarial samples). Specifically, we see this happening when the switch is after the learning rate drop, *i.e.*, explicit actions to promote convergence because this will prevent the model from adapting to the newly introduced adversarial samples. Hence, a naive approach to naturally pretrain the model till full convergence and then switch to adversarial samples is not feasible.

Now we provide a method to automatically determine the value of the hyperparameter S. From Figure 2, a quick recipe for deciding S is to choose an epoch after the test accuracy on natural samples flattens and before any learning rate drops. Hence, we use the evolution of training loss from Figure 1 to come-up with a dynamic strategy to switch from natural training to adversarial training. Specifically, we dynamically determine the value of S as the epoch number where the training loss begins to converge. Convergence is determined if the training loss value at the current epoch is within $D\%$ of the running average of training losses over the previous W epochs. We explain Delayed Adversarial Training with this switching strategy in Algorithm 2.

It is easy to note that even though we have based our motivation and experimentation on the min-max approach [11], our proposed modification to regular adversarial training can be easily extended to other frameworks which use different type of attacks in their training process.

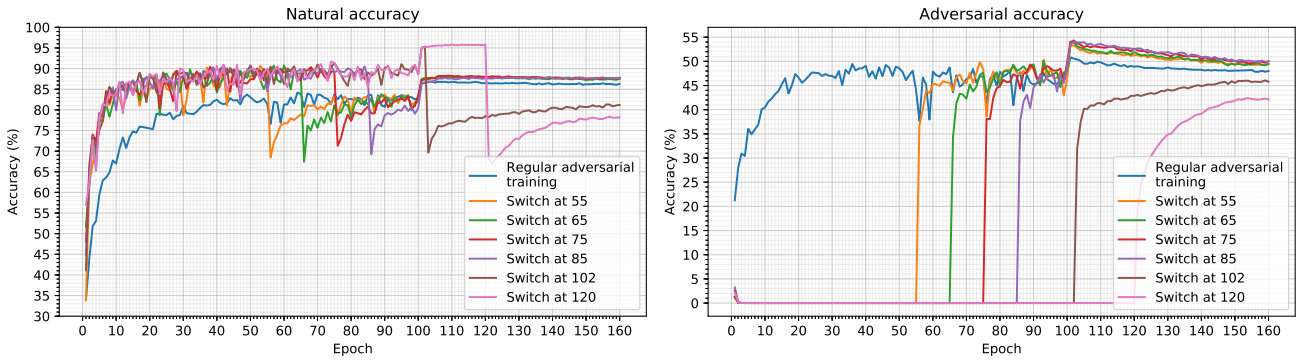

Figure 2. Natural and adversarial test accuracy during regular adversarial training and adversarial training with different switches. CIFAR-10 images are classified using the WideResNet-28x10. Adversarial samples with $T = 10, \epsilon = \frac{8}{255}$ and $\alpha = \frac{2}{255}$ are used. SGD learning rate drops are after epochs 100, 105 and 150.

Algorithm 1 Delayed Adversarial Training

Input: Switching hyperparameter S; J training examples $\{(\boldsymbol{x}_j, y_j)\}_{j=1}^{J}$; Number of epochs N; Optimizer and its parameters; PGD attack $\mathcal{A}_{T,\epsilon,\alpha}$ with parameters T, ϵ, α (number of steps, ball size, step size)

Output: Model parameters, $\boldsymbol{\theta}$

Randomly initialize $\boldsymbol{\theta}$
for $i = 0$ **to** $N - 1$ **do**
  for each minibatch $(\boldsymbol{x}_b, y_b)$ **do**
    if $i > S$ **then**
      Replace $\boldsymbol{x}_b$ with $\boldsymbol{x}_b \leftarrow \mathcal{A}_{T,\epsilon,\alpha}(\boldsymbol{\theta}, \boldsymbol{x}_b, y_b)$
    end if
// if condition replaces natural samples with adversarial samples after epoch S. $S = 0$ corresponds to regular adversarial training.
    Update $\boldsymbol{\theta}$ with the optimizer using $(\boldsymbol{x}_b, y_b)$
  end for
end for

Algorithm 2 Delayed Adversarial Training with Automated Switching

Input: Length of training loss window to consider W; Training loss deviation D from average window loss; J training examples $\{(\boldsymbol{x}_j, y_j)\}_{j=1}^{J}$; Number of epochs N; Optimizer and its parameters; PGD attack $\mathcal{A}_{T,\epsilon,\alpha}$ with parameters T, ϵ, α (number of steps, ball size, step size)

Output: Model parameters $\boldsymbol{\theta}$

Randomly initialize $\boldsymbol{\theta}$
$S \leftarrow N$ // initialize switch
for $i = 0$ **to** $N - 1$ **do**
  for each minibatch $(\boldsymbol{x}_b, y_b)$ **do**
    if $i > S$ **then**
      Replace $\boldsymbol{x}_b$ with $\boldsymbol{x}_b \leftarrow \mathcal{A}_{T,\epsilon,\alpha}(\boldsymbol{\theta}, \boldsymbol{x}_b, y_b)$
    end if
// if condition replaces natural samples with adversarial samples after epoch S.
    Update $\boldsymbol{\theta}$ with optimizer and using $(\boldsymbol{x}_b, y_b)$
  end for
  if $S == N$ **then**
    Calculate epoch's training loss L_i
    if L_i is within $D\%$ of the average of $\{L_{i-W}, \ldots, L_{i-2}, L_{i-1}\}$ **then**
      $S \leftarrow i$
    end if
  end if
end for

3.3. DAT helps generalization

From Figure 2 we see that with delayed adversarial training, the test accuracy is often better as compared to regular adversarial training. Also, there is a larger adversarial accuracy jump after the learning rate reduction at epoch 100 which is indicative of better generalization. This happens because in the case of delayed adversarial training, the model is not overfitting to adversaries of little relevance in the initial phase of training (see Figure 1). Instead it is learning to classify natural samples which we always desire to be correctly classified.

We can further confirm that our method helps generalization by looking at the training loss and adversarial test accuracy for regular adversarial training and delayed adversarial training. Figure 3 shows how these values evolve during the course of training. In the case of delayed adversarial training (red dotted curve), the training loss is low until the switch, because the training has only been done on natural samples. This shoots up almost vertically at the time of switch to adversarial samples and then comes down gradually as the model get trained on adversarial samples. The learning rate drop at epoch 100 causes the training loss of both methods to drop significantly. As is evident from the figure, delayed adversarial training has higher training loss and higher test accuracy which indicates better generalization as compared to regular adversarial training.

As mentioned in Section 2 Cai et al. [3] also demonstrated overfitting on adversarial samples with regular ad-

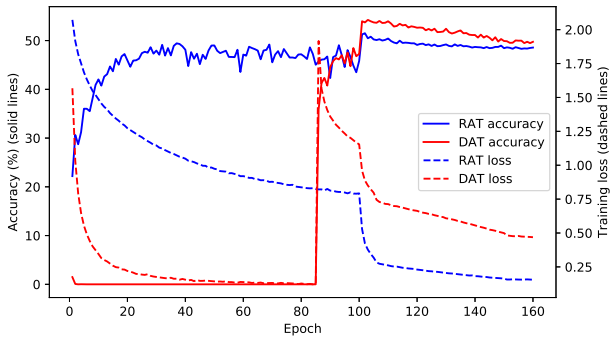

Figure 3. Adversarial test accuracy and training loss evolution for the WideResNet-28x10 for CIFAR-10. We show regular adversarial training (RAT) and delayed adversarial training (DAT). The left axis gives the accuracy and right axis gives the training loss. Solid lines are accuracy curves and dashed lines are training loss curves.

versarial training. They show adversarial accuracy begins to drop during regular adversarial training. We additionally demonstrate this with ResNet-18 in supplementary material Section B.2. We see that this drop is reduced with delayed adversarial training. Furthermore, Wang et al. [24] also observed overfitting on adversarial samples when using strong adversarial samples in the initial phase of training. However, these works did not use this to make robustness training more efficient.

4. Experiments

In this section we evaluate our proposed method for efficiently training robust deep neural network models. We compare the training time and test accuracy of the models trained using our proposed method against regular adversarial training [11]. We use PGD attacks which is one of the most powerful first-order ℓ_∞ attacks [1]. We use the standard CIFAR-10, CIFAR-100 and MNIST datasets in our evaluation.

For CIFAR-10, we use the WideResNet-28x10, ResNet-50 and ResNet-18 architectures with a batch size of 128. With CIFAR-100, we use ResNet-50 and ResNet-18 with a batch size of 128. For MNIST we use a model with two convolutional layers, which have 32 and 64 filters, followed by a fully connected layer of size 1024. Each convolutional layer is followed by a ReLU and 2×2 max pooling. The batch size for MNIST experiments is 50.

SGD with momentum 0.9 and weight decay 2×10^{-4} is used to train the WideResNet-28x10 and all the ResNets for CIFAR. The initial learning rate for the WideResNet-28x10 is 0.1 and it is reduced by a factor of ten after epochs 100, 105 and 150 and the model is trained for a total of 155 epochs. The ResNets also have an initial learning rate of 0.1 which is reduced by a factor of ten after epochs 100 and 150 and the model is trained for a total of 155 epochs. Adam with an initial learning rate of 1×10^{-4} is used for MNIST. The Adam optimizer state and learning rate is reinitialized after switching. MNIST training is for 80 epochs. W and D parameters from Algorithm 2 are empirically tuned.

We parameterized PGD adversaries as $\{T, \epsilon, \alpha\}$ where T is the number of PGD steps, ϵ is the maximum ℓ_∞ perturbation and α is the gradient ascent step size (see Equations (2) and (3)). For CIFAR-10 and CIFAR-100 all adversarial samples used during training are of strength $\{10, \frac{8}{255}, \frac{2}{255}\}$. MNIST adversarial samples for training are generated with $\{40, 0.3, 0.01\}$. These are standard strengths in the literature [11, 7]. All PGD adversaries are constructed by adding an initial random perturbation, $\boldsymbol{\delta}$, where $\|\boldsymbol{\delta}\|_\infty \leq \epsilon$.

CIFAR experiments are performed on a system with two NVIDIA V100 GPUs, 42 CPU cores and 48GB memory. For MNIST experiments we use a system with one NVIDIA V100 GPU, 26 CPU cores and 24GB memory.

As ultimately the time taken to reach a given accuracy is what matters and not the total number of epochs, we also have an early stopping criteria for the CIFAR-10 models to regularize and avoid training the models for too long without much gain in accuracy. In this case, we stop training after the first learning rate drop once the test loss on the natural samples of the current epoch is within 5% of the average test loss of the previous five epochs. Furthermore early stopping helps prevent overfitting to adversarial samples.

4.1. Training times

Table 2 shows the training times for all the models and datasets. We compare the time taken for regular adversarial training and our method. The timings for the early stopping of CIFAR-10 models are also shown. Note that MNIST experiments are done on a less powerful system than the CIFAR experiments.

The training times are significantly reduced using our method with often better accuracy. The importance of our method is particularly felt when training large models such as the WideResNet-28x10 on the more complex CIFAR-10 dataset. In this case we get a 46.9% reduction in training time. With regularization via early stopping, we get 62.4% savings in training time. On all other dataset and models also we get significant reduction in training time with accuracy very close to that of regular adversarial training. In fact, our robustness accuracy is mostly higher as compared to that of regular adversarial training due to better generalization on adversarial samples as discussed in Section 3.

4.2. Generalization to other attacks

We evaluate the robustness of our models against attacks of strengths which they were not trained to resist. We vary the number of PGD steps and the ϵ-ball size of the test attacks. Figures 4, 5 and 6 show the performance with regular adversarial training and our method for the CIFAR-10 models (Figure 15 in supplementary material Section B.2 shows

		CIFAR-10			
		Training time	*Time saved*	$T = 10$, $\epsilon = 8/255$	*Natural accuracy*
WideResNet-28x10	*RAT*	14.7 HOURS	46.9%	48.5%	86.8%
	DAT	7.8 HOURS		49.7%	87.9%
	RAT early stop	10.9 HOURS	62.4%	49.2%	87.1%
	DAT early stop	4.1 HOURS		53.6%	87.9%
ResNet-50	*RAT*	6.8 HOURS	45.6%	41.0%	75.2%
	DAT	3.7 HOURS		41.1%	74.4%
	RAT early stop	5.0 HOURS	64.0%	42.3%	73.2%
	DAT early stop	1.8 HOURS		41.6%	72.2%
ResNet-18	*RAT*	2.5 HOURS	36.0%	37.0%	73.8%
	DAT	1.6 HOURS		40.4%	72.8%
	RAT early stop	1.9 HOURS	52.6%	40.6%	71.0%
	DAT early stop	0.9 HOURS		41.1%	69.9%
		CIFAR-100			
		Training time	*Time saved*	$T = 10$, $\epsilon = 8/255$	*Natural accuracy*
ResNet-50	*RAT*	6.9 HOURS	42.0%	15.2%	44.2%
	DAT	4.0 HOURS		15.2%	46.6%
Resnet-18	*RAT*	2.6 HOURS	46.2%	14.2%	44.7%
	DAT	1.4 HOURS		14.2%	46.7%
		MNIST			
		Training time	*Time saved*	$T = 40$, $\epsilon = 0.3$	*Natural accuracy*
Two-layer CNN	*RAT*	2.2 HOURS	13.6 %	91.4%	98.2%
	DAT	1.9 HOURS		91.9%	98.2%

Table 2. Training time and test accuracy with natural samples and adversaries of same strength as training adversaries for Regular adversarial training (RAT) and Delayed Adversarial Training (DAT). MNIST experiments are done on a less powerful system.

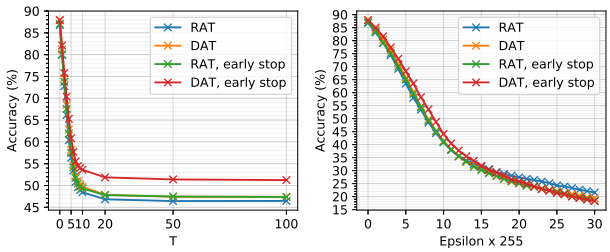

Figure 4. Accuracy of fully trained WideResNet-28x10 with CIFAR-10 when tested with attacks of different strength. Adversaries used during training were of strength $\{10, \frac{8}{255}, \frac{2}{255}\}$.

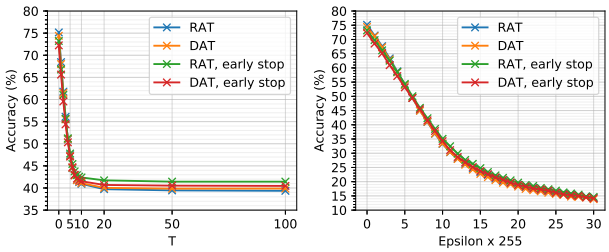

Figure 5. Accuracy of fully trained ResNet-50 with CIFAR-10 when tested with attacks of different strength. Adversaries used during training were of strength $\{10, \frac{8}{255}, \frac{2}{255}\}$.

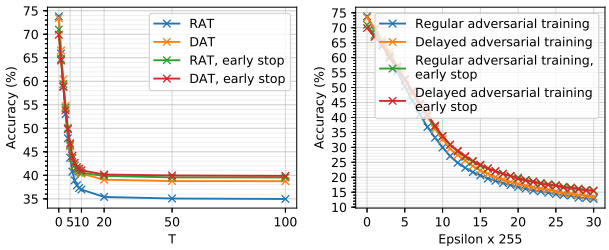

Figure 6. Accuracy of fully trained ResNet-18 with CIFAR-10 when tested with attacks of different strength. Adversaries used during training were of strength $\{10, \frac{8}{255}, \frac{2}{255}\}$.

the robustness for CIFAR-100 with ResNet-18). Figure 7 shows it for MNIST. The models trained using our method are comparably robust against a wide variety of attacks and follow the same pattern as seen in Madry et al. [11]. As expected [1] a larger number of PGD iterations and ϵ-ball size decreases accuracy. Table 4 in the supplementary material summarizes the differences between regular adversarial training and our method for typical benchmark adversaries.

We note that increasing the model capacity improves robustness for all datasets which is consistent with observations made by Madry et al. Also consistent is the sharp fall in MNIST robustness when tested with ϵ larger than what the model was trained for ($\epsilon = 0.3$). Our method delays the collapse in robustness as we can see from Figure 7.

Black-box attacks We check the performance of models trained using our method under black-box attacks from independently trained copies of the network. We test with CIFAR-10 and the WideResNet-28x10 with adversaries of

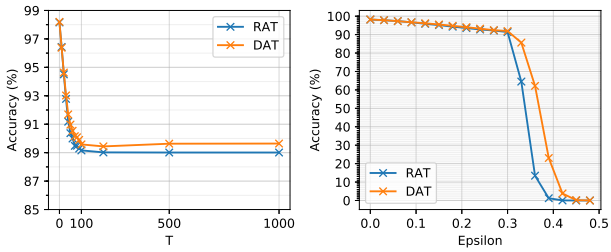

Figure 7. Accuracy of fully trained two-layer CNN with MNIST when tested with attacks of different strength. Adversaries used during training were of strength $\{40, 0.3, 0.01\}$.

strength $\{10, \frac{8}{255}, \frac{2}{255}\}$. From Table 2 the white-box accuracy using delayed adversarial training is 49.7%.

Independent copies of the network were trained with 1) Natural training; 2) Regular adversarial training (RAT); 3) Delayed adversarial training (DAT). Table 3 shows that black-box accuracy is higher than white-box accuracy as expected [1] and reports accuracy values that are consistent with prior work [11].

Source	WHITE-BOX	NATURAL	RAT	DAT
Accuracy	49.7%	86.7%	69.4%	65.7%

Table 3. CIFAR-10 Black-box robustness against attacks from independent copies of the WideResNet-28x10.

4.3. Performance with modified training schedules

Until now we used similar training and learning rate schedules across models to fairly compare against regular adversarial training accuracy values reported in literature. Next we show that our results are independent of the schedule. We reduce the adversarial training time by accelerating the learning rate drops and finishing the training process in a relatively smaller number of epochs. The total number of epochs, learning rate drop epochs and the switching hyper-parameter S, are approximately halved. Figure 8 shows the natural and adversarial test accuracy evolution during training for this accelerated training procedure for CIFAR-10 with the WideResNet-28x10 model. The measured training times for the regular adversarial training and our proposed delayed adversarial training for this experiments are 9.5 hours and 6.8 hours respectively—a 28% saving. Furthermore, we see that this training schedule gives an almost 2% improvement in adversarial accuracy over regular adversarial training.

Figure 9 shows the performance of the models trained with accelerated training against attacks of different strengths. There is a slight reduction in overall performance of the accelerated training models at higher values of ϵ. However, when only comparing accelerated models, the performance of the models trained with our proposed method is comparable to the model trained with regular adversarial training.

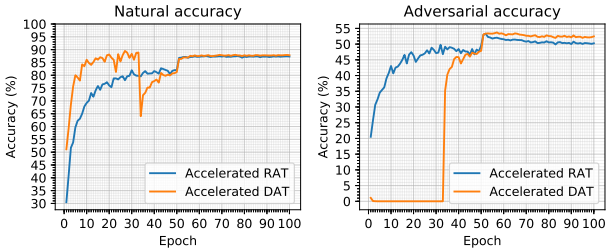

Figure 8. Natural and adversarial test accuracy during regular adversarial training and delayed adversarial training when the training process is accelerated. CIFAR-10 images are classified using the WideResNet-28x10. Adversarial samples with $T = 10, \epsilon = \frac{8}{255}$ and $\alpha = \frac{2}{255}$ are used.

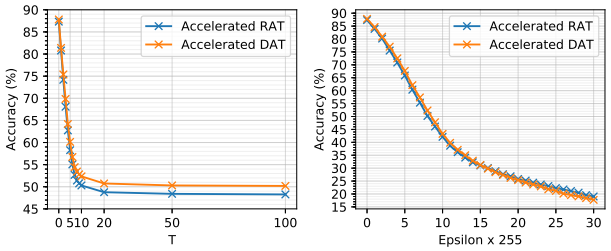

Figure 9. Accuracy of fully trained WideResNet-28x10 with CIFAR-10 when training is accelerated and tested with attacks of different strength. Adversaries used during training were of strength $\{10, \frac{8}{255}, \frac{2}{255}\}$.

5. Conclusion

In this paper we analyzed the computational efficiency of regular adversarial training to efficiently improve the robustness of deep neural networks. We presented insights about the usefulness of training with adversarial samples in the initial and later phases of regular adversarial training and proposed a modified version of the training framework which significantly improves its computational requirement. We further show through various experiments that the neural network models trained through the proposed framework are as accurate or better as the ones trained through the earlier framework and generalize quite well to adversarial attacks of various strengths.

In the future, we plan to adapt our general method to other adversarial training frameworks. Furthermore, although our goal is to improve training time, we get better accuracy in many cases. Future work aims to explore this further to develop a methodology which optimizes accuracy and computational efficiency in a more integrated way.

References

[1] Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *International Conference on Machine Learning*, pages 274–283, 2018. 1, 6, 7, 8

[2] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*, pages 2546–2554, 2011. 3

[3] Qi-Zhi Cai, Chang Liu, and Dawn Song. Curriculum adversarial training. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 3740–3747, 2018. 1, 2, 5

[4] Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 3–14. ACM, 2017. 1, 2

[5] Evelyn Duesterwald, Anupama Murthi, Ganesh Venkataraman, Mathieu Sinn, and Deepak Vijaykeerthy. Exploring the hyperparameter landscape of adversarial robustness. *arXiv preprint arXiv:1905.03837*, 2019. 1, 3

[6] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations*, 2015. 1

[7] Dan Hendrycks, Kimin Lee, and Mantas Mazeika. Using pre-training can improve model robustness and uncertainty. In *International Conference on Machine Learning*, pages 2712–2721, 2019. 1, 3, 6

[8] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009. 2

[9] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016. 1

[10] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1

[11] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. 1, 2, 3, 4, 6, 7, 8

[12] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 86–94. IEEE, 2017. 1

[13] Tianyu Pang, Kun Xu, Chao Du, Ning Chen, and Jun Zhu. Improving adversarial robustness via promoting ensemble diversity. In *International Conference on Machine Learning*, pages 4970–4979, 2019. 2

[14] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In *Advances in Neural Information Processing Systems*, pages 5014–5026, 2018. 2

[15] Lukas Schott, Jonas Rauber, Matthias Bethge, and Wieland Brendel. Towards the first adversarially robust neural network model on MNIST. In *International Conference on Learning Representations*, 2019. 2

[16] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Advances in Neural Information Processing Systems*, pages 3353–3364, 2019. 2

[17] Ali Shafahi, Parsa Saadatpanah, Chen Zhu, Amin Ghiasi, Christoph Studer, David Jacobs, and Tom Goldstein. Adversarially robust transfer learning. *arXiv preprint arXiv:1905.08232*, 2019. 3

[18] Yash Sharma and Pin-Yu Chen. Attacking the Madry Defense Model with l_1-based adversarial examples. *arXiv preprint arXiv:1710.10733*, 2017. 1

[19] Thilo Strauss, Markus Hanselmann, Andrej Junginger, and Holger Ulmer. Ensemble methods as a defense to adversarial perturbations against deep neural networks. *arXiv preprint arXiv:1709.03423*, 2017. 2

[20] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is robustness the cost of accuracy?–a comprehensive study on the robustness of 18 deep image classification models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 631–648, 2018. 3

[21] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014. 1

[22] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *International Conference on Learning Representations*, 2018. 2

[23] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness

may be at odds with accuracy. In *International Conference on Learning Representations (ICLR)*, 2019. 3

[24] Yisen Wang, Xingjun Ma, James Bailey, Jinfeng Yi, Bowen Zhou, and Quanquan Gu. On the convergence and robustness of adversarial training. In *International Conference on Machine Learning*, pages 6586–6595, 2019. 1, 2, 3, 6

[25] Eric Wong, Leslie Rice, and J. Zico Kolter. Fast is better than free: Revisiting adversarial training. In *International Conference on Learning Representations*, 2020. 3

[26] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 3

[27] Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Accelerating adversarial training via maximal principle. In *Advances in Neural Information Processing Systems*, pages 227–238, 2019. 2