Abstract

Deep learning models have shown impressive performance across a spectrum of computer vision applications including medical diagnosis and autonomous driving. One of the major concerns that these models face is their susceptibility to adversarial attacks. Realizing the importance of this issue, more researchers are working towards developing robust models that are less affected by adversarial attacks. Adversarial training method shows promising results in this direction. In adversarial training regime, models are trained with mini-batches augmented with adversarial samples. Fast and simple methods (e.g., single-step gradient ascent) are used for generating adversarial samples, in order to reduce computational complexity. It is shown that models trained using single-step adversarial training method (adversarial samples are generated using non-iterative method) are pseudo robust. Further, this pseudo robustness of models is attributed to the gradient masking effect. However, existing works fail to explain when and why gradient masking effect occurs during single-step adversarial training. In this work, (i) we show that models trained using single-step adversarial training method learn to prevent the generation of single-step adversaries, and this is due to over-fitting of the model during the initial stages of training, and (ii) to mitigate this effect, we propose a single-step adversarial training method with dropout scheduling. Unlike models trained using existing single-step adversarial training methods, models trained using the proposed single-step adversarial training method are robust against both single-step and multi-step adversarial attacks, and the performance is on par with models trained using computationally expensive multi-step adversarial training methods, in white-box and black-box settings.

1. Introduction

Machine learning models are susceptible to adversarial samples: samples with imperceptible, engineered noise designed to manipulate model’s output [15, 2, 34, 3, 13, 27]. Further, Szegedy et al. [34] observed that these adversarial samples are transferable across multiple models i.e., adversarial samples generated on one model might mislead other models. Due to which, models deployed in the real world are susceptible to black-box attacks [20, 28], where limited or no knowledge of the deployed model is available to the attacker. Various schemes have been proposed to defend against adversarial attacks (e.g., [13, 29, 23]), in this direction Adversarial Training (AT) procedure [13, 35, 22, 40] shows promising results.

In adversarial training regime, models are trained with mini-batches containing adversarial samples typically generated by the model being trained. Adversarial sample generation methods range from simple methods [13] to complex optimization methods [24]. In order to reduce computational complexity, non-iterative methods such as Fast Gradient Sign Method (FGSM) [13] are typically used for generating adversarial samples. Further, it has been shown that models trained using single-step adversarial training methods are pseudo robust [35]:

- Although these models appears to be robust to single-step attacks in white-box setting (complete knowledge of the deployed model is available to the attacker), they are susceptible to single-step attacks (non-iterative methods) in black-box attack setting [35].
- Further, these models are susceptible to multi-step attacks (iterative methods) in both white-box setting [18] and black-box setting [10].

Tramer et al. [35] demonstrated that models trained using single-step adversarial training method converges to degenerative minima, and exhibit gradient masking effect. Single-step adversarial sample generation methods such as FGSM, compute adversarial perturbations based on the linear approximation of the model’s loss function i.e., image is perturbed in the direction of the gradient of loss with respect to the input image. Gradient masking effect causes this linear approximation of loss function to become unreliable for generating adversarial samples during single-step adversarial training. Madry et al. [22] demonstrated that models trained using adversarial samples that maximize the training loss are robust against single-step and multi-step at-
attacks. Such samples could be generated using the Projected Gradient Descent (PGD). However, PGD method is an iterative method, due to which training time increases substantially. Though prior works have enabled to learn robust models, they fail to answer the following important questions: (i) Why models trained using single-step adversarial training method exhibit gradient masking effect? and (ii) At what phase of the single-step adversarial training, the model starts to exhibit gradient masking effect?

In this work, we attempt to answer these questions and propose a novel single-step adversarial training method to learn robust models. First, we show that models trained using single-step adversarial training method learn to prevent the generation of single-step adversaries, and this is due to over-fitting of the model during the initial stages of training. Over-fitting of the model on single-step adversaries causes linear approximation of loss function to become unreliable for generating adversarial samples i.e., gradient masking effect. Finally, we propose a single-step adversarial training method with dropout scheduling to learn robust models. Note that, just adding dropout layer (typical setting: dropout layer with fixed dropout probability after FC+ReLU layer) does not help the model trained using single-step adversarial training method to gain robustness. Prior works observed no significant improvement in the robustness of models (with dropout layers in typical setting), trained using normal training and single-step adversarial training methods [13, 18]. Results for these settings are shown in section 4.1. Unlike typical setting, we introduce dropout layer after each non-linear layer (i.e., dropout-2D after conv2D+ReLU, and dropout-1D after FC+ReLU) of the model, and further decay its dropout probability as training progress. Interestingly, we show that this proposed dropout setting has significant impact on the model’s robustness. The major contributions of this work can be listed as follows:

- We show that models trained using single-step adversarial training method learns to prevent the generation of single-step adversaries, and this is due to over-fitting of the model during the initial stages of training.

- Harnessing on the above observation, we propose a single-step adversarial training method with dropout probability scheduling. Unlike models trained using existing single-step adversarial training methods, models trained using the proposed method are robust against both single-step and multi-step attacks.

- The proposed single-step adversarial training method is much faster than multi-step adversarial training methods, and achieves on par results.

2. Notations

Consider a neural network $f$ trained to perform image classification task, and $\theta$ represents parameters of the neural network. Let $x$ represents the image from the dataset and $y_{true}$ be its corresponding ground truth label. The neural network is trained using loss function $J$ (e.g., cross-entropy loss), and $\nabla_x J$ represents the gradient of loss with respect to the input image $x$. Adversarial image $x_{adv}$ is generated by adding norm-bounded perturbation $\delta$ to the image $x$. Perturbation size ($\epsilon$) represents the $l_\infty$ norm constraint on the generated adversarial perturbation i.e., $||\delta||_\infty \leq \epsilon$. Please refer to supplementary document for details on adversarial training and attack generation methods.

3. Related Works

Following the findings of Szegedy et al. [34], various attacks (e.g., [13, 24, 8, 26, 25, 10, 12] have been proposed. Further, in order to defend against adversarial attacks, various schemes such as adversarial training (e.g., [13, 18, 22, 40, 5, 4]) and input pre-processing (e.g., [14, 31]) have been proposed. Athalye et al. [1] showed that obfuscated gradients give a false sense of robustness, and broke seven out of nine defense papers [6, 21, 14, 38, 32, 31, 21, 21, 9] accepted to ICLR 2018. In this direction, adversarial training method [22], shows promising results for learning robust deep learning models. Kurakin et al. [18] observed that models trained using single-step adversarial training methods are susceptible to multi-step attacks. Further, Tramer et al. [35] demonstrated that these models exhibit gradient masking effect, and proposed Ensemble Adversarial Training (EAT) method. However, models trained using EAT are still susceptible to multi-step attacks in white-box setting. Madry et al. [22] demonstrated that adversarially trained model can be made robust against white-box attacks, if perturbation crafted while training maximizes the loss. Zhang et al. [40] proposed a regularizer for multi-step adversarial training, that encourages the output of the network to be smooth. On the other hand, works such as [30] and [36] propose a method to learn models that are provably robust against norm bounded adversarial attacks. However, scaling these methods to deep networks and large perturbation sizes is difficult. Whereas, in this work we show that it is possible to learn robust models using single-step adversarial training method, if over-fitting of the model on adversarial samples is prevented during training. We achieve this by introducing dropout layer after each non-linear layer of the model with a dropout schedule.

4. Over-fitting and its effect during adversarial training

In this section, we show that models trained using single-step adversarial training method learn to prevent the genera-
Figure 1: **Single-step adversarial training:** Trend of $R_\epsilon$, training loss, and validation loss during single-step adversarial training, obtained for LeNet+ trained on MNIST dataset. Column-1: plot of $R_\epsilon$ versus training iteration. Column-2: training loss versus training iteration. Column-3: validation loss versus training iteration. Note that, when $R_\epsilon$ starts to decay, loss on adversarial validation set starts to increase indicating that the model is over-fitting on the adversarial samples.

Figure 2: **Multi-step adversarial training:** Trend of $R_\epsilon$, training loss, and validation loss during multi-step adversarial training, obtained for LeNet+ trained on MNIST dataset. Column-1: plot of $R_\epsilon$ versus training iteration. Column-2: training loss versus training iteration. Column-3: validation loss versus training iteration. Note that, for the entire training duration $R_\epsilon$ does not decay, and no over-fitting effect can be observed.

At each iteration, norm bounded adversarial perturbations that maximizes the training loss should be generated. Further, the model’s parameters ($\theta$) should be updated so as to decrease the loss on such adversarial samples. Madry et al. [22] solves the maximization step by generating adversarial samples using an iterative method named Projected Gradient Descent (PGD). In order to quantify the extent of inner maximization of Eq. (1), we compute loss ratio $R_\epsilon$ using Eq. (2). Loss ratio is defined as the ratio of loss on the adversarial samples to the loss on its corresponding clean samples for a given perturbation size $\epsilon$. The metric $R_\epsilon$ captures the extent of inner maximization achieved by the generated adversarial samples i.e., factor by which loss has increased by perturbing the clean samples. A sample is said to be an adversarial sample if it is capable of manipulating the model’s prediction. Such manipulations could be achieved by perturbing the samples along the adversarial direction [13]. A perturbation is said to be an adversarial perturbation when it causes loss on the perturbed sample to increase. This implies that the loss...
on the adversarially perturbed samples should be greater than the loss on the corresponding unperturbed samples i.e., $\text{loss}_{\text{adv}} > \text{loss}_{\text{clean}}$. Based on these facts, $R_c$ can be interpreted in the following manner:

- Generated perturbation is said to be an adversarial perturbation if $R_c > 1$ i.e., $\text{loss}_{\text{adv}} > \text{loss}_{\text{clean}}$.

- $R_c < 1$ i.e., $\text{loss}_{\text{adv}} < \text{loss}_{\text{clean}}$, implies that the generated perturbation is not an adversarial perturbation. The attack method fails to generate adversarial perturbations for the given model.

We obtain the plot of $R_c$ versus iteration for models trained using single-step adversarial training method [13] and multi-step adversarial training method [22]. Column-1 of Fig. 1 and Fig. 2 show these plots obtained for LeNet+ trained on MNIST dataset [19] using single-step and multi-step adversarial training methods respectively. It can be observed that during single-step adversarial training, $R_c$ initially increases and then starts to decay rapidly. Further $R_c$ becomes less than one after 20 ($\times 100$) iterations. This implies that single-step adversarial sample generation method is unable to generate adversarial perturbations for the model, leading to adversarial training without useful adversarial samples.

We demonstrate this behavior of the model to prevent the inclusion of adversarial samples is due to over-fitting on the adversarial samples. Typically during normal training, loss on the validation set is monitored to detect over-fitting effect i.e., validation loss increases when the model starts to over-fit on the training set. Unlike normal training, during adversarial training we monitor the loss on the clean and adversarial validation set. A normally trained model is used for generating adversarial validation set, so as to ensure that the generated adversarial validation samples are independent of the model being trained. Column-2 and column-3 of Fig. 1 shows the plot of loss versus iteration during training of LeNet+ on MNIST dataset using single-step adversarial training. It can be observed that, when $R_c$ starts to decay, loss on the adversarial validation set starts to increase. This increase in the validation loss indicates over-fitting of the model on the single-step adversaries. Whereas, during multi-step adversarial training method, $R_c$ initially increases and then saturates (column-1, Fig. 2). Further, no such over-fitting effect is observed for the entire training duration (column-3, Fig. 2). Note that, a normally trained model was used for generating FGSM ($\epsilon=0.3$) adversarial validation set, and we observe similar trend if a normally trained model of different architecture is used for generating FGSM adversarial validation set, please refer to supplementary document.

4.1. Effect of dropout layer

In the previous section, we showed that models trained using single-step adversarial training learn to prevent the generation of single-step adversaries. Further, we demonstrated that this behavior of models is due to over-fitting. Dropout layer [33] has been shown to be effective in mitigating over-fitting during training, and typically dropout-1D layer is added after FC+ReLU layers in the networks. We refer to this setting as typical setting. Prior works which used dropout layer during single-step adversarial training observed no significant improvement in the model’s robustness. This is due to the use of dropout layer in typical setting. Whereas, we empirically show that it is necessary to introduce dropout layer after every non-linear layer of the model (proposed dropout setting i.e., dropout-2D after Conv2D+ReLU layer and dropout-1D after FC+ReLU layer) to mitigate over-fitting during single-step adversarial training, and to enable the model to gain robustness against adversarial attacks (single-step and multi-step attacks). We train LeNet+ with dropout layer in typical setting and in the proposed setting respectively, on MNIST dataset using single-step adversarial training method for different values of dropout probability. After training, we obtain the performance of these resultant models against PGD attack ($\epsilon=0.3, \epsilon_{\text{step}}=0.01, \text{steps}=40$). Column-1 of Fig. 3 shows the trend of accuracy of these models for PGD attack with respect to the dropout probability used while training. It can be observed that the gain in the robustness of adversarially trained model with dropout layer in the proposed setting is significantly better compared to the adversarially trained model with dropout layer in typical setting (FAT-TS). From column-2 of Fig. 3, it can be observed that the robustness of adversarially trained model with dropout layer in the proposed setting, increases with the increase in the dropout probability ($p$) and reaches a peak value at $p=0.4$. Further increase in the dropout probability causes decrease in the accuracy on both clean and adversarial samples. Based on this observation, we propose an improved single-step adversarial training in the next subsection. Furthermore, we perform normal training of LeNet+ with dropout layers in typical setting and in the proposed setting, on MNIST dataset. From column-1 of Fig. 3, it can be observed that there is no significant improvement in the robustness of these normally trained models.

4.2. SADS: Single-step Adversarial training with Dropout Scheduling

Column-1 of Fig. 3 indicates that use of dropout layer in typical setting is not sufficient to avoid over-fitting on adversarial samples, and we need severe dropout regime involving all the layers (i.e., proposed setting: dropout layer after Conv2D+ReLU and FC+ReLU layers) of the network in order to avoid over-fitting. For the proposed dropout regime,
determining exact dropout probability is network dependent and is difficult. Further, having high dropout probability causes under-fitting of the model, and having low dropout probability causes the model to over-fit on the adversarial samples.

Based on these observations, we propose a single-step adversarial training method with dropout scheduling (Algorithm 1). In the proposed training method, we introduce dropout layer after each non-linear layer of the model to be trained. We initialize these dropout layers with a high dropout probability $P_d$. Further, during training we linearly decay the dropout probability of all the dropout layers and this decay in the dropout probability is controlled by the hyper-parameter $r_d$. The hyper-parameter, $r_d$ is expressed in terms of maximum training iterations (e.g., $r_d = 1/2$ implies that dropout probability reaches zero when the current training iteration is equal to half of the maximum training iterations). In experimental section 5, we show the effectiveness of the proposed training method. Note that dropout layer is only used while training.

5. Experiments

In this section, we show the effectiveness of models trained using the proposed single-step adversarial training method (SADS) in white-box and black-box settings. We perform the sanity tests described in [7], in order to verify that models trained using SADS are robust and does not exhibit obfuscated gradients (Athalye et al. [1] demonstrated that models exhibiting obfuscated gradients are not robust against adversarial attacks). We show results on MNIST [19], Fashion-MNIST [37] and CIFAR-10 [16] datasets. We use LeNet+ (please refer to supplementary document for details on network architecture) for both MNIST and Fashion-MNIST datasets. For CIFAR-10 dataset, WideResNet-28-10 [39] is used. These models are trained using SGD with momentum. Step-policy is used for learning rate scheduling. For all datasets, images are pre-processed to be in [0,1] range. For CIFAR-10, random crop and horizontal flip are performed for data-augmentation.

Evaluation: We show the performance of models against adversarial attacks in white-box and black-box setting. For SADS, we report mean and standard deviation over three runs.

Attacks: For $l_\infty$ based attacks, we use Fast Gradient Sign Method (FGSM) [13], Iterative Fast Gradient Sign Method (IFGSM) [17], Momentum Iterative Fast Gradient Sign Method (MI-FGSM) [10] and Projected Gradient Descent

Algorithm 1: Single-step Adversarial training with Dropout Scheduling (SADS)

Input:
Training mini-batch size ($m$)
Maximum training iterations ($Max_{iteration}$)
Hyper-parameters: $P_d$, $r_d$

1 **Initialization**
Randomly initialize network $N$
iteration = 0
prob = $P_d$
Insert dropout layer after each non-linear layer of the network $N$
Set dropout probability ($p$) of all the dropout layers with $prob$

while iteration $\leq Max_{iteration}$ do

2 Read minibatch $B=\{x^1, ... x^m\}$ from training set
3 Compute FGSM adversarial sample
   $\{x_{adv}^1, ... x_{adv}^m\}$ from corresponding clean samples $\{x^1, ... x^m\}$ using the current state of the network $N$
4 Make new minibatch $B^* = \{x_{adv}^1, ... x_{adv}^m\}$
/*Forward pass, compute loss, backward pass, and update parameters*/
5 Do one training step of Network $N$ using minibatch $B$
/*Update dropout probability of Dropout-1D and Dropout-2D layers with $prob$*/
6 prob = max($0, P_d \cdot (1 - \frac{iteration}{r_d \cdot Max_{iteration}})$)
7 iteration = iteration + 1

end
Table 1: MNIST: White-Box setting. Classification accuracy (%) of models trained on MNIST dataset using different training methods. For all attacks $\epsilon=0.3$ is used and for PGD attack $\epsilon_{step}=0.01$ is used. For both IFGSM and PGD attacks, steps is set to 40.

| Training Method | Attack Method | Clean | FGSM | IFGSM | PGD |
|-----------------|---------------|-------|------|-------|-----|
| NT              | Clean         | 99.24 | 11.65| 0.31  | 0.01|
|                 | Multi-step adversarial training | | | | |
| PAT             |              | 98.41 | 95.56| 92.64 | 92.08|
| TRADES          |              | 98.70 | 96.30| 95.14 | 95.05|
|                 | Single-step adversarial training | | | | |
| FAT             |              | 99.34 | 89.04| 1.19  | 0.17|
| SADS            |              | 98.89 | 94.78| 89.35 | 88.51|
|                 | ±0.01        | ±0.19 | ±0.09| ±0.22 | |

Table 2: Fashion-MNIST: White-box attack. Classification accuracy (%) of models trained on Fashion-MNIST dataset using different training methods. For all attacks $\epsilon=0.1$ is used and for PGD attack $\epsilon_{step}=0.01$ is used. For both IFGSM and PGD attacks, steps is set to 40.

| Training Method | Attack Method | Clean | FGSM | IFGSM | PGD |
|-----------------|---------------|-------|------|-------|-----|
| NT              | Clean         | 91.42 | 6.46 | 1.01  | 0.16|
|                 | Multi-step adversarial training | | | | |
| PAT             |              | 84.55 | 77.30| 75.95 | 75.18|
| TRADES          |              | 86.69 | 80.39| 78.94 | 78.04|
|                 | Single-step adversarial training | | | | |
| FAT             |              | 90.45 | 83.43| 21.26 | 16.65|
| SADS            |              | 85.21 | 75.81| 71.14 | 69.51|
|                 | ±0.08        | ±1.31 | ±1.01| ±1.43 | |

Table 3: CIFAR-10: White-Box attack. Classification accuracy (%) of models trained on CIFAR-10 dataset using different training methods. For all attacks $\epsilon=8/255$ is used and for PGD attack $\epsilon_{step}=2/255$ is used. For both IFGSM and PGD attacks, steps is set to 7.

| Training Method | Attack Method | Clean | FGSM | IFGSM | PGD |
|-----------------|---------------|-------|------|-------|-----|
| NT              | Clean         | 94.75 | 28.16| 0.07  | 0.03|
|                 | Multi-step adversarial training | | | | |
| PAT             |              | 85.70 | 53.96| 48.65 | 47.23|
| TRADES          |              | 87.20 | 56.34| 51.21 | 50.03|
|                 | Single-step adversarial training | | | | |
| FAT             |              | 94.04 | 98.54| 0.31  | 0.09|
| SADS            |              | 82.01 | 51.99| 46.37 | 45.66|
|                 | ±0.06        | ±1.02 | ±1.17| ±1.26 | |

5.1. Performance in White-box setting

We train models on MNIST, Fashion-MNIST and CIFAR-10 datasets respectively, using NT, FAT, PAT, TRADES and SADS (Algorithm 1) training methods. Models are trained for 50, 50 and 100 epochs on MNIST, Fashion-MNIST and CIFAR-10 datasets respectively. For SADS, we set the hyper-parameter $P_d$ and $r_d$ to (0.8, 0.5), (0.8, 0.75) and (0.5, 0.5) for MNIST, Fashion-MNIST and CIFAR-10 datasets respectively. Table 1, 2 and 3 shows the performance of these models against single-step and multi-step attacks in white-box setting, rows represent the training method and columns represent the attack generation method. It can be observed that models trained using FAT are not robust against multi-step attacks. Whereas, models trained using PAT, TRADES and SADS are robust against both single-step and multi-step attacks. Unlike PAT and TRADES, the proposed SADS method is a single-step adversarial training method.

PGD attack with large steps: Engstrom et al. [11] demonstrated that the performance of models trained using certain adversarial training methods degrade significantly with increase in the number of steps of PGD attack. In order to verify that such behavior is not observed in models trained using SADS, we obtain the plot of classification accuracy on PGD test-set versus steps of PGD attack. Fig. 4 shows these plots obtained for models trained using PAT and SADS on MNIST, Fashion-MNIST and CIFAR-10 datasets respectively. It can be observed that the accuracy of models on PGD test set initially decreases slightly and then saturates. Even for PGD attack with large steps, there is no significant degradation in the performance of models trained using PAT and SADS methods. In supplementary document, we show the effect of hyper-parameters of the proposed training method.
5.2. Performance in Black-box setting

In this subsection, we show the performance of models trained using different training methods against adversarial attacks in black-box setting. Typically, a substitute model (source model) is trained on the same task using normal training method, and this trained substitute model is used for generating adversarial samples. The generated adversarial samples are transferred to the deployed model (target model). We use FGSM and MI-FGSM methods for generating adversarial samples, since samples generated using these methods show good transfer rates [10]. Table 4 shows the performance of models trained using different methods, in black-box setting. It can be observed that the performance of models trained using PAT and SADS in black-box setting is better than that in white-box setting. Further, it can be observed that the performance of models trained on MNIST and CIFAR-10 datasets using FAT is worse in black-box setting than compared in white-box setting. Please refer to supplementary file for details on network architecture of source models.

5.3. Performance against DeepFool and C&W attacks

DeepFool [24] and C&W [8] attacks generate adversarial perturbations with minimum $l_2$ norm, that is required to fool the classifier. These methods measure the robustness of the model in terms of the average $l_2$ norm of the generated adversarial perturbations for the test set. For an undefended model, adversarial perturbation with small $l_2$ norm...
Table 5: **DeepFool and C&W attacks:** Performance of models trained using different training methods against DeepFool and C&W attacks. These attack methods measure the robustness of the model based on the average $l_2$ norm of the generated perturbations, the better the better. Success defines the percentage of samples of test set that has been misclassified. Note that, for models trained using PAT and SADS, perturbations with relatively large $l_2$ norm is required to fool the classifier.

| Method | MNIST | F-MNIST | CIFAR-10 |
|--------|-------|---------|----------|
|        | DeepFool | CW | DeepFool | CW | DeepFool | CW |
|        | Success | Mean $l_2$ | Success | Mean $l_2$ | Success | Mean $l_2$ | Success | Mean $l_2$ | Success | Mean $l_2$ |
| NT     | 99.35 | 1.837 | 100 | 1.659 | 93.73 | 0.796 | 100 | 1.709 | 96 | 0.20 |
| FAT    | 99.37 | 1.455 | 100 | 0.798 | 93.11 | 1.514 | 100 | 1.167 | 96 | 0.25 |
| PAT    | 85.68 | 4.633 | 99 | 2.779 | 90.29 | 2.635 | 100 | 1.572 | 92 | 1.22 |
| SADS   | 95.89 | 3.692 | 100 | 2.321 | 90.68 | 2.305 | 100 | 1.308 | 93 | 0.97 |
|        | ±0.06 | ±0.033 | 0± | ±0.027 | ±0.26 | ±0.102 | ±0 | ±0.188 | ±0.32 | ±0.043 | ±0 | ±0.014 |

Table 6: Comparison of training time per epoch of models trained on MNIST and CIFAR-10 datasets respectively, obtained for different training methods.

| Method | Training time per epoch (sec.) |
|--------|-------------------------------|
|        | MNIST | CIFAR-10 |
| NT     | ~2.7 | ~104 |
| FAT    | ~4.1 | ~159 |
| PAT    | ~53  | ~820 |
| TRADES | ~104 | ~1558 |
| SADS   | ~4.3 | ~187 |

is enough to fool the classifier. Whereas for robust models, adversarial perturbation with relatively large $l_2$ norm is required to fool the classifier. Table 5 shows the performance of models trained using NT, FAT, PAT and SADS methods, against DeepFool and C&W attacks. It can be observed that models trained using PAT and SADS have relatively large average $l_2$ norm. Whereas, for models trained using NT and FAT have small average $l_2$ norm.

### 5.4. Sanity tests

We perform sanity tests described in [7] to verify whether models trained using SADS are adversarially robust and are not exhibiting obfuscated gradients. We perform following sanity tests:

- **Iterative attacks should perform better than non-iterative attacks**
- **White-box attacks should perform better than black-box attacks**
- **Unbounded attacks should reach 100% success**
- **Increasing distortion bound should increase attack success rate**

Models trained using SADS pass above tests. From Table 1, 2 and 3, it can be observed that iterative attacks (IFGSM and PGD) are stronger than non-iterative attack (FGSM) for models trained using SADS. Comparing results in Tables 1, 2 and 3 with results in Table 4, it can be observed that white-box attacks are stronger than black-box attacks for models trained using SADS. Fig. 5 shows the accuracy plot for the model on test set versus perturbation size of PGD attack, obtained for models trained using SADS. It can be observed that the model’s accuracy falls to zero for large perturbation size ($\epsilon$). From Fig. 5, it can be observed that PGD attack success rate (attack success rate is equal to (100 - model’s accuracy)%) increases with increase in the distortion bound (perturbation size) of the attack.

### 5.5. Time Complexity

In order to quantify the complexity of different training methods, we measure training time per epoch (seconds) for models trained using different training methods. Table 6 shows the training time per epoch for models trained on MNIST and CIFAR-10 datasets respectively. Note that the training time of SADS and FAT is of the same order. The increase in the training time for PAT and TRADES is due to their iterative nature of generating adversarial samples. We ran this timing experiment on a machine with NVIDIA Titan Xp GPU, with no other jobs on this GPU.

### 6. Conclusion

In this work, we have demonstrated that models trained using single-step adversarial training methods learn to prevent the generation of adversaries due to over-fitting of the model during the initial stages of training. To mitigate this effect, we have proposed a novel single-step adversarial training method with dropout scheduling. Unlike existing single-step adversarial training methods, models trained using the proposed method achieves robustness not only against single-step attacks but also against multi-step attacks. Further, the performance of models trained using the proposed method is on par with models trained using multi-step adversarial training methods, and is much faster than multi-step adversarial training methods.

**Acknowledgment:** This work was supported by Uchhatar Avishkar Yojana (UAY) project (IISC_010), MHRD, Govt. of India.
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 (ICML), 2018. 2, 5

[2] Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Srdić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In Joint European Conference on Machine Learning and Knowledge Discovery in Databases, 2013. 1

[3] Battista Biggio, Giorgio Fumera, and Fabio Roli. Pattern recognition systems under attack: Design issues and research challenges. International Journal of Pattern Recognition and Artificial Intelligence, 28(07), 2014. 1

[4] Vivek B.S., Arya Baburaj, and R. Venkatesh Babu. Regularizer to mitigate gradient masking effect during single-step adversarial training. In IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2019. 2

[5] Vivek B.S., Konda Reddy Mopuri, and R. Venkatesh Babu. Gray-box adversarial training. In European Conference on Computer Vision (ECCV), 2018. 2

[6] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer Encoding: One Hot Way To Resist Adversarial Examples. In International Conference on Learning Representations (ICLR), 2018. 2

[7] Nicholas Carlini, Anish Athalye, Nicolas Papernot, Wieland Brendel, Jonas Rauber, Dimitris Tsipras, Ian Goodfellow, and Aleksander Madry. On evaluating adversarial robustness. arXiv preprint arXiv:1902.06705, 2019. 5, 8

[8] Nicholas Carlini and David Wagner. Towards Evaluating the Robustness of Neural Networks. arXiv preprint arXiv:1608.04644, 2016. 2, 6, 7

[9] Guneet S. Dhillon, Kamyar Azizzadenesheli, Jeremy D. Bernstein, Jean Kossaifi, Aran Khanna, Zachary C. Lipton, and Animashree Anandkumar. Stochastic activation pruning for robust adversarial defense. In International Conference on Learning Representations (ICLR), 2018. 2

[10] Yinpeng Dong, Fangzhao Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jiaguo Li. Boosting adversarial attacks with momentum. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018. 1, 2, 5, 7, 11

[11] Logan Engstrom, Andrew Ilyas, and Anish Athalye. Evaluating and understanding the robustness of adversarial logit pairing. arXiv preprint arXiv:1807.10272, 2018. 6

[12] Aditya Ganeshan, Vivek B.S., and R. Venkatesh Babu. FDA: Feature Disruptive Attack. In IEEE International Conference on Computer Vision (ICCV), 2019. 2

[13] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In International Conference on Learning Representations (ICLR), 2015. 1, 2, 3, 4, 5, 11

[14] Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering Adversarial Images using Input Transformations. In International Conference on Learning Representations (ICLR), 2018. 2

[15] Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. Adversarial Machine Learning. In ACM Workshop on Security and Artificial Intelligence, AISec, 2011. 1

[16] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 5, 13

[17] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. arXiv preprint arXiv:1607.02533, 2016. 5, 11

[18] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial Machine Learning at Scale. In International Conference on Learning Representations (ICLR), 2017. 1, 2, 6

[19] Yann LeCun. The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/. 4, 5

[20] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In International Conference on Learning Representations (ICLR), 2017. 1

[21] Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Michael E. Houle, Dawn Song, and James Bailey. Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality. In International Conference on Learning Representations (ICLR), 2018. 2

[22] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Tsipras Dimitris, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In International Conference on Learning Representations (ICLR), 2018. 2

[23] Jan H. Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On Detecting Adversarial Perturbations. In International Conference on Learning Representations (ICLR), 2018. 1, 2, 3, 4, 6, 11, 12

[24] Jan H. Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On Detecting Adversarial Perturbations. In International Conference on Learning Representations (ICLR), 2018. 1

[25] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfoul: A simple and accurate method to fool deep neural networks. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. 1, 2, 6, 7

[26] Konda Reddy Mopuri, Aditya Ganeshan, and R. Venkatesh Babu. Generalizable Data-Free Objective for Crafting Universal Adversarial Perturbations. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 41(10):2452–2465, Oct. 2019. 2

[27] Konda Reddy Mopuri, Utsav Garg, and R Venkatesh Babu. Fast Feature Fool: A data independent approach to universal adversarial perturbations. In Proceedings of the British Machine Vision Conference (BMVC), 2017. 2

[28] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In IEEE European Symposium on Security and Privacy (EuroS&P), 2016. 1

[29] Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a Defense to Adver-
sariable Perturbations against Deep Neural Networks. *arXiv preprint arXiv:1511.04508*, 2015. 1

[30] Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified Defenses against Adversarial Examples. In *International Conference on Learning Representations (ICLR)*, 2018. 2

[31] Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting Classifiers Against Adversarial Attacks Using Generative Models. In *International Conference on Learning Representations (ICLR)*, 2018. 2

[32] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples. In *International Conference on Learning Representations (ICLR)*, 2018. 2

[33] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(56):1929–1958, 2014. 4

[34] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014. 1, 2

[35] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble Adversarial Training: Attacks and Defenses. In *International Conference on Learning Representations (ICLR)*, 2018. 1, 2, 6, 12

[36] Eric Wong and J Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. *arXiv preprint arXiv:1711.00851*, 2017. 2

[37] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017. 5

[38] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating Adversarial Effects Through Randomization. In *International Conference on Learning Representations (ICLR)*, 2018. 2

[39] Sergey Zagoruyko and Nikos Komodakis. Wide Residual Networks. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2016. 5, 13

[40] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically Principled Trade-off between Robustness and Accuracy. In *International Conference on Machine Learning (ICML)*, 2019. 1, 2, 6, 12
Supplementary

S 1. Contents
- Section S 2: Network architecture.
- Section S 3: Adversarial training and Attack generation methods.
- Section S 4: Additional plots to illustrate over-fitting effect during single-step adversarial training.
- Section S 5: Effect of hyper-parameters
- Section S 6: Comparison with EAT
- Section S 7: Trend of $R_{\epsilon}$, training and validation loss during SADS.

S 2. Network Architecture
Network architecture of models used for MNIST and Fashion-MNIST datasets are shown in Table 7. Model-A and Model-B are used for generating adversarial samples in black-box setting.

S 3. Adversarial training and Attack generation methods

S 3.1. Adversarial Sample Generation Methods
In this subsection, we discuss the formulation of adversarial attacks.

Fast Gradient Sign Method (FGSM): Non-iterative attack method proposed by [13]. This method generates $l_{\infty}$ norm bounded adversarial perturbation based on the linear approximation of loss function.

$$x^* = x + \epsilon \cdot \text{sign}(\nabla_x J(f(x;\theta), y_{true}))$$  \hspace{1cm} (3)

Iterative Fast Gradient Sign Method (IFGSM) [17]: Iterative version of FGSM attack. At each iteration, adversarial perturbation of small step size ($\alpha$) is added to the image. In our experiments, we set $\alpha = \epsilon / \text{steps}$.

$$x^0 = x$$ \hspace{1cm} (4)
$$x^{N+1} = x^N + \alpha \cdot \text{sign}(\nabla_x J(f(x^N;\theta), y_{true}))$$ \hspace{1cm} (5)

Projected Gradient Descent (PGD) [22]: Initially, a small random noise sampled from Uniform distribution ($U$) is added to the image. Then at each iteration, perturbation of small step size ($\epsilon_{\text{step}}$) is added to the image, and followed by re-projection.

$$x^0 = x + U(-\epsilon_{\text{step}}, \epsilon_{\text{step}}, \text{shape}(x))$$ \hspace{1cm} (6)
$$x^{N+1} = x^N + \epsilon_{\text{step}} \cdot \text{sign}(\nabla_x J(f(x^N;\theta), y_{true}))$$ \hspace{1cm} (7)
$$x^{N+1} = \text{clip}(x^{N+1}, \text{min} = x - \epsilon, \text{max} = x + \epsilon)$$ \hspace{1cm} (8)

Momentum Iterative Fast Gradient Sign Method (MI-FGSM) [10]: Introduces a momentum term into the IFGSM formulation. Here, $\mu$ represents the momentum term. $\alpha$ represents step size and is set to $\epsilon / \text{steps}$.

$$x^{0} = x$$ \hspace{1cm} (9)
$$g^{N+1} = \mu \cdot g^{N} + \nabla_x J(f(x^N;\theta), y_{true})$$ \hspace{1cm} (10)
$$x^{N+1} = x^{N} + \alpha \cdot \text{sign}(g^{N+1})$$ \hspace{1cm} (11)

S 3.2. Adversarial Training Methods
In this subsection we explain the existing adversarial training methods.

| LeNet+ | Model-A | Model-B | Model-C | Model-D |
|--------|---------|---------|---------|---------|
| Conv(32,5,5) + Relu MaxPool(2,2) | Conv(64,5,5) + Relu | Dropout(0.2) | Conv(128,3,3) + Tanh MaxPool(2,2) | FC(300) + Relu | 
| Conv(64,5,5) + Relu MaxPool(2,2) | Dropout(0.25) | Conv(128,6,6) + Relu | Conv(64,3,3) + Tanh MaxPool(2,2) | Dropout(0.5) × 4 |
| FC(1024) + Relu | FC(128) + Relu | Dropout(0.5) | FC(128) + Relu | FC + Softmax |
| FC + Softmax | FC + Softmax | FC + Softmax | FC + Softmax | |

Table 7: Architecture of networks used for MNIST and Fashion-MNIST datasets.

11
FGSM Adversarial Training (FAT): During training, at each iteration a portion of clean samples in the mini-batch are replaced with their corresponding adversarial samples generated using the model being trained. Fast Gradient Sign Method (FGSM) is used for generating these adversarial samples.

Ensemble Adversarial Training (EAT) [35]: At each iteration a portion of clean samples in the mini-batch are replaced with their corresponding adversarial samples. These adversarial samples are generated by the model being trained or by one of the model from the fixed set of pre-trained models. Table 8 shows the setup used for EAT method.

PGD Adversarial Training (PAT): Multi-step adversarial training method proposed by [22]. At each iteration all the clean samples in the mini-batch are replaced with their corresponding adversarial samples generated using the model being trained. Projected Gradient Descent (PGD) method is used for generating these samples.

TRADES: Multi-step adversarial training method proposed by [40]. The method proposes a regularizer that encourages the output of the network to be smooth. The training mini-batches contain clean and their corresponding adversarial samples. These adversarial samples are generated using Projected Gradient Descent with modified loss function.

S 4. Additional plots to illustrate over-fitting effect

In the main paper, we showed over-fitting effect during training of LeNet+ on MNIST dataset using single-step adversarial training. Fig. 6 shows the plot of validation loss, obtained for ResNet-34 trained on CIFAR-10 dataset using single-step adversarial training. We observe over-fitting effect even when model with different architecture is used for generating adversarial validation set. Fig. 7 shows the validation loss obtained for LeNet+ trained on MNIST dataset using single-step adversarial training. Normally trained models with different architecture are used for generating adversarial validation set.
S 5. Effect of Hyper-Parameters

In order to show the effect of hyper-parameters, we train LeNet+ shown in table 7 on MNIST dataset, using SADS with different hyper-parameter settings. Validation set accuracy of the model for PGD attack ($\varepsilon = 0.3$ and $steps = 40$) is obtained for each hyper-parameter setting with one of them being fixed and the other being varied.

Effect of hyper-parameter $P_d$: The hyper-parameter $P_d$ defines the initial dropout probability applied to all dropout layers. We train LeNet+ on MNIST dataset, using the proposed method for different initial dropout probability $P_d$. Column-1 of Fig. 8 shows the effect of varying dropout probability from 0.3 to 0.9. It can be observed that the robustness of the model to multi-step attack initially increases with the increase in the value of $P_d$ ($P_d < 0.8$), and further increase in $P_d$ causes the model’s robustness to decrease, and this is due to under-fitting.

Effect of hyper-parameter $r_d$: The hyper-parameter $r_d$ decides the iteration at which dropout probability reaches zero and is expressed in terms of maximum training iteration. Column-2 of Fig. 8 shows the effect varying $r_d$ from $1/4$ to $1$. It can be observed that for $r_d < 0.5$, there is degradation in the robustness of the model against multi-step attacks. This is because, during the initial stages of training, learning rate is high and the model can easily over-fit to adversaries generated by single-step method.

S 6. Comparison with Ensemble Adversarial Training

We train WideResNet-28-10 [39] on CIFAR-10 dataset using EAT and SADS. Table 8 shows the setup used for EAT. Pre-trained models are used for generating adversarial samples during EAT. Table 9 shows the recognition accuracy of models trained using EAT and SADS in white-box attack setting. It can be observed that the model trained using SADS is robust to both single-step (FGSM) and multi-step attacks (PGD), whereas models trained using EAT are susceptible to multi-step attack.
S 7. SADS: Trend of $R_e$, training and validation loss

Fig. 9 and 10 show the trend of $R_e$, training and validation loss, obtained for models trained using SADS. It can be observed that for the entire training duration $R_e$ does not decay and no over-fitting effect can be observed.

Table 8: Setup used for Ensemble Adversarial Training.

| Network to be trained | Pre-trained Models          |
|-----------------------|-----------------------------|
| WRN-28-10 (Ens-A)     | WRN-28-10, ResNet-34        |
| CIFAR-10              | WRN-28-10 (Ens-B) WRN-28-10, VGG-19 |
| WRN-28-10 (Ens-C)     | ResNet-34, VGG-19          |

Table 9: CIFAR-10: White-Box attack. Classification accuracy (%) of models trained on CIFAR-10 dataset using different training methods. For all attacks $\epsilon=8/255$ is used and for PGD attack $\epsilon_{\text{step}}=2/255$ and $\text{steps}=7$ is used.

| Training Method | Attack Method | Clean | FGSM | PGD-7 |
|-----------------|---------------|-------|------|-------|
| EAT Ens-A       | 92.92         | 59.56 | 19.21|
| EAT Ens-B       | 92.75         | 63.40 | 5.34 |
| EAT Ens-C       | 93.11         | 59.74 | 12.03|
| SADS            | 82.01         | 51.99 | 45.66|
|                 | ±0.06         | ±1.02 | ±1.26|