ROBUST SINGLE-STEP ADVERSARIAL TRAINING

Anonymous authors
Paper under double-blind review

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. In order to scale adversarial training to large networks and datasets, fast and simple methods (e.g., single-step gradient ascent) are used for generating adversarial samples. 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 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, and (ii) to mitigate this effect, we propose a single-step adversarial training method with dropout scheduling to learn robust models. Unlike models trained using single-step adversarial training method, models trained using the proposed single-step adversarial training method are robust against both single-step and multi-step adversarial attacks, and achieve on-par results compared to the computationally expensive state-of-the-art multi-step adversarial training method, 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 (Huang et al., 2011; Biggio et al., 2013; Szegedy et al., 2013; Biggio et al., 2014; Goodfellow et al., 2015; Papernot et al., 2016). Further, Szegedy et al. (2013) 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 (Liu et al., 2017; Papernot et al., 2017), 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., (Szegedy et al., 2013; Papernot et al., 2015; Metzen et al., 2017)), in this direction *Adversarial Training (AT)* procedure (Szegedy et al., 2013; Tramèr et al., 2018; Madry et al., 2018; Xie et al., 2019) 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 (Goodfellow et al., 2015) to complex optimization methods (Moosavi-Dezfooli et al., 2016). In order to scale adversarial training to large datasets, non-iterative methods such as Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015) are typically used for generating adversarial samples. Further, it has been shown that models trained using single-step adversarial training methods are pseudo robust (Tramèr et al., 2018):

- 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 suscepti-

ble to single-step attacks (non-iterative methods) in black-box attack setting (Tramèr et al., 2018).
- Further, these models are susceptible to multi-step attacks (iterative methods) in both white-box setting (Kurakin et al., 2017) and black-box setting (Dong et al., 2018).

Tramèr et al. (2018) demonstrated that models trained using single-step adversarial training method 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 (sign of) 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. Further, Madry et al. (2018) demonstrated that the adversarially trained model can be made robust against white-box attacks (both single-step and multi-step attacks), if perturbations crafted while training maximizes the model's loss, and this is achieved by generating adversaries using Projected Gradient Descent (PGD) method. However, PGD method is an iterative method, due to which training time increases substantially. Therefore, this method is hard to scale for large datasets and networks. 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 exhibits 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. This 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 (Szegedy et al., 2013; Kurakin et al., 2017). Results for these settings are shown in section 3.1. 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 during training. 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 method i.e., PGD adversarial training method, and achieves on par results.

2 NOTATIONS

Consider a neural network f trained to perform image classification task, and θ 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 δ to the image x. Let, perturbation size (ϵ) represents the norm constraint on the generated adversarial perturbation i.e., $||\delta||_\infty \leq \epsilon$ for l_∞ norm constraint. Please refer to A.1 for details on adversarial attack generation methods.

3 OVER-FITTING AND ITS EFFECT DURING ADVERSARIAL TRAINING

In this section, 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. First, we discuss the criteria for learning robust models using adversarial training method, and then we show that this criteria is not satisfied during single-step adversarial training method. Most importantly, we show the reason for failure to satisfy this criteria is due to over-fitting.

Madry et al. (2018) demonstrated that it is possible to learn robust models using adversarial training method, if adversarial perturbations (typically l_∞ or l_2 norm bounded) crafted while training maximizes the model's loss. This training objective is formulated as a minimax optimization problem (Eq. 1). Where ψ represents the feasible set e.g., for l_∞ norm constraint attacks $\psi = \{\delta : ||\delta||_\infty \leq \epsilon\}$, and D is the training set.

$$\min_{\theta} \left[E_{(x,y)\in D} \left[\max_{\delta \in \psi} J\big(f(x+\delta;\theta), y_{true}\big) \right] \right] \quad (1) \qquad R_\epsilon = \frac{loss_{adv}}{loss_{clean}} \quad (2)$$

At each iteration, norm bounded adversarial perturbations that maximizes the training loss should be generated. Further, the model's parameters (θ) should be updated, so as to decrease the loss on such adversarial samples. Madry et al. (2018) 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_ϵ 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 ϵ. The metric R_ϵ 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.
By definition, adversarial perturbations are those perturbations which manipulate the model's predictions. Such manipulations can be achieved by generating perturbations that increase the loss on samples (Goodfellow et al., 2015). Based on these facts, a perturbation is said to be an *adversarial perturbation* only when it causes loss to increase. This implies that the loss on the perturbed samples should be greater than the loss on the corresponding unperturbed samples i.e., $loss_{adv} > loss_{clean}$. With these facts, R_ϵ can be interpreted in the following manner:

- Generated perturbation is said to be an *adversarial perturbation* if $R_\epsilon > 1$ i.e., $loss_{adv} > loss_{clean}$
- $R_\epsilon < 1$ i.e., $loss_{adv} < loss_{clean}$, implies that the generated perturbation is not an adversarial perturbation. Further, the generation method is unable to generate *adversarial perturbations* for the given model.

We obtain the plot of R_ϵ versus iteration for models trained using single-step adversarial training method (Goodfellow et al., 2015) and multi-step adversarial training method (Madry et al., 2018). Column-1 of Fig. 1 and Fig. 2 shows these plots obtained for LeNet+ (refer table 8) trained on MNIST dataset (LeCun) using single-step and multi-step adversarial training methods respectively. It can be observed that during single-step adversarial training, R_ϵ initially increases and then starts to decay rapidly. Further R_ϵ 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 overfit 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_ϵ 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_ϵ initially increases and then saturates (column-1, Fig. 2). Further, no such over-fitting effect is observed for the entire training duration (column-3,

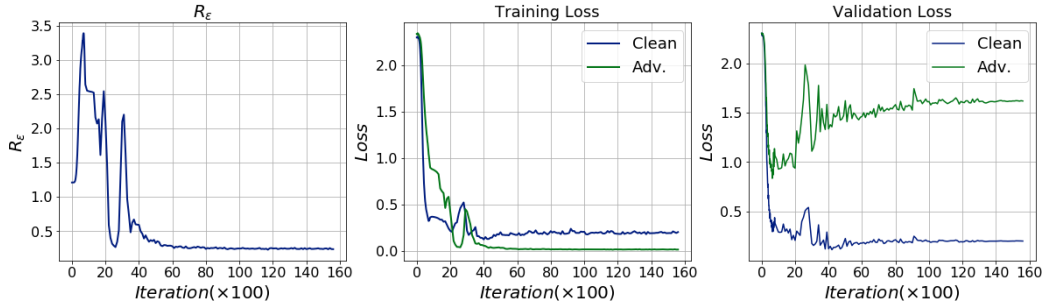

Figure 1: **Single-step adversarial training:** Trend of R_ϵ, training loss, and validation loss during single-step adversarial training, obtained for LeNet+ trained on MNIST dataset. Column-1: plot of R_ϵ versus training iteration. Column-2: training loss versus training iteration. Column-3: validation loss versus training iteration. Note that, when R_ϵ starts to decay, loss on adversarial validation set starts to increase indicating that the model is over-fitting on the adversarial samples.

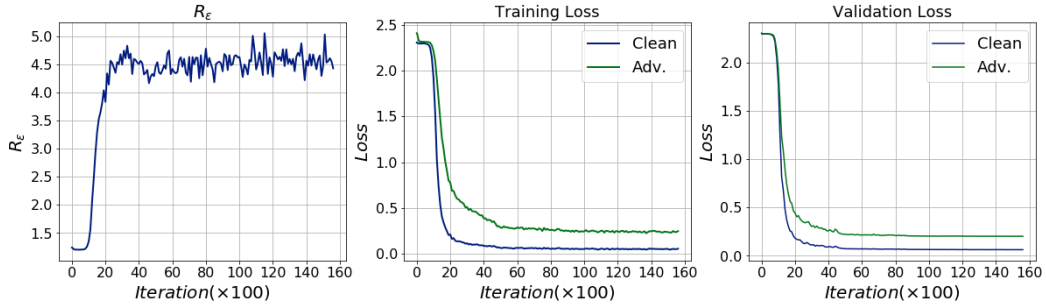

Figure 2: **Multi-step adversarial training:** Trend of R_ϵ, training loss, and validation loss during multi-step adversarial training, obtained for LeNet+ trained on MNIST dataset. Column-1: plot of R_ϵ versus training iteration. Column-2: training loss versus training iteration. Column-3: validation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

Fig. 2). In A.3, we show similar results for models trained on CIFAR-10 dataset. Note that, a normally trained model was used for generating FGSM (ϵ=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 A.3.

3.1 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 (Srivastava et al., 2014) 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 work (Kurakin et al., 2017) 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 (ϵ=0.3, ϵ_{step}=0.01, $steps$=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 probabil-

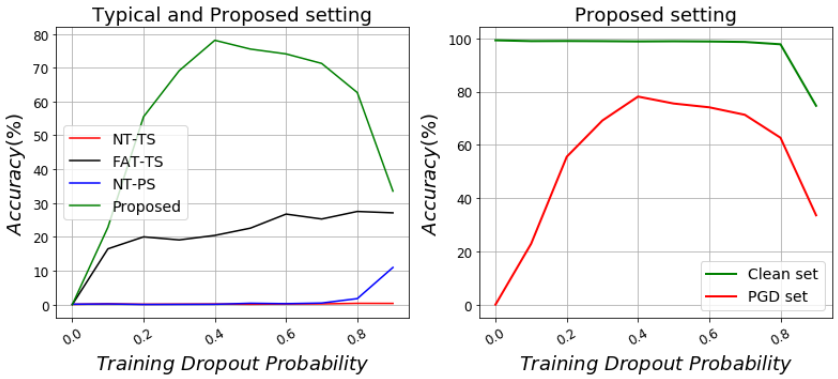

Figure 3: **Column-1:** Effect of dropout probability of dropout layers in typical setting and in the proposed setting on the model's robustness against PGD attack (ϵ=0.3, ϵ_{step}=0.01 and $steps$=40). Obtained for LeNet+ trained on MNSIT dataset. NT-TS: Normal training with dropout layer in typical setting. FAT-TS: Single-step adversarial training with dropout layer in typical setting. NT-PS: Normal training with dropout layer in the proposed setting. Proposed: Single-step adversarial training with dropout layer in the proposed setting. **Column-2:** Effect of dropout probability on the model's accuracy on clean and PGD adversarial validation set (ϵ=0.3, ϵ_{step}=0.01 and $steps$=40). Obtained for LeNet+ with dropout layer in the proposed setting, trained using single-step adversarial training method on MNIST dataset.

ity (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 normally train 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.

3.2 SADS: Single-step Adversarial training with Dropout Scheduling

We have demonstrated that existing single-step adversarial training does not make the model robust against adversarial attack, instead, the model learns to prevent the generation of single-step adversaries. Furthermore, we demonstrated this behavior of model trained using single-step adversaries is due to over-fitting on the adversarial samples during the initial stages of training. 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 4, we show the effectiveness of the proposed training method. Note that dropout layer is only used while training.

4 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 tests described in Athalye et al. (2018), in order to verify that models trained using SADS do not exhibit *obfuscated*

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

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

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

gradients. (Athalye et al. (2018) demonstrated that models exhibiting obfuscated gradients are not robust against adversarial attacks). We show results on MNIST (LeCun), Fashion-MNIST (Xiao et al., 2017) and CIFAR-10 (Krizhevsky et al.) datasets. We use LeNet+ shown in table 8 for both MNIST and Fashion-MNIST datasets. For CIFAR-10 dataset, ResNet-34 (He et al., 2015) 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_∞ based attacks, we use Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015), Iterative Fast Gradient Sign Method (IFGSM) (Kurakin et al., 2016), Momentum Iterative Fast Gradient Sign Method (MI-FGSM) (Dong et al., 2018) and Projected Gradient Descent (PGD) (Madry et al., 2018). For l_2 based attack, we use DeepFool (Moosavi-Dezfooli et al., 2016) and Carlini&Wagner (Carlini & Wagner, 2016).
Perturbation size: For l_∞ based attacks, we set perturbation size (ϵ) to the values described in Madry et al. (2018) i.e., ϵ=0.3, 0.1 and 8/255 for MNIST, Fashion-MNIST and CIFAR-10 datasets respectively.
Comparisons: We compare the performance of the proposed single-step adversarial training method SADS with Normal training (NT), FGSM adversarial training (FAT) (Kurakin et al., 2017), Ensemble adversarial training (EAT) (Tramèr et al., 2018), and PGD adversarial training (PAT) (Madry et al., 2018). Refer to A.2 in appendix for more details on these training methods. Note that, FAT, EAT and SADS (ours) are single-step adversarial training methods, whereas PAT is multi-step adversarial training method.

4.1 Performance in White-box setting

We train models on MNIST, Fashion-MNIST and CIFAR-10 datasets respectively, using NT, FAT, EAT, PAT and SADS (Algorithm 1) training methods (please refer to A.2 for details on training methods). These 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 (0.6, 0.6) 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

Table 1: White-Box setting: Classification accuracy (%) of models trained on MNIST dataset using different training methods. For all attacks ϵ=0.3 is used and for PGD attack ϵ_{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		99.24	11.65	0.31	0.01
FAT		99.34	89.04	1.19	0.17
EAT	A	99.35	83.48	18.75	10.13
	B	99.31	80.16	48.13	37.85
	C	99.20	82.48	4.00	1.29
	D	97.66	56.85	0.87	0.29
PAT		98.41	95.56	92.64	92.08
SADS		98.89 ±0.01	94.78 ±0.19	89.35 ±0.09	88.51 ±0.22

Table 2: White-Box attack: Classification accuracy (%) of models trained on Fashion-MNIST dataset using different training methods. For all attacks ϵ=0.1 is used and for PGD attack ϵ_{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		91.42	6.46	1.01	0.16
FAT		90.45	83.43	21.26	16.65
EAT	A	91.17	76.22	9.52	4.92
	B	90.18	73.48	7.59	3.56
	C	90.18	73.48	7.59	3.56
	D	83.81	47.41	24.23	18.26
PAT		84.55	77.30	75.95	75.18
SADS		85.21 ±0.08	75.81 ±1.31	71.14 ±1.01	69.51 ±1.43

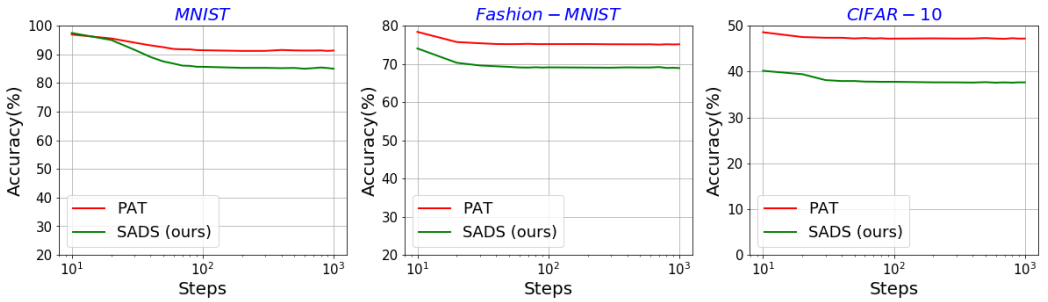

Figure 4: Plot of accuracy of the model trained using PAT and SADS, on PGD adversarial test set versus steps of PGD attack with fixed ϵ. For PGD attack we set (ϵ,ϵ_{step}) to (0.3,0.01), (0.1,0.01) and (8/255,2/255) for MNIST, Fashion-MNIST and CIFAR-10 datasets. Note, x-axis is in logarithmic scale.

setting, rows represent the training method and columns represent the attack generation method. It can be observed that models trained using the existing single-step adversarial training methods are not robust against multi-step attacks. Whereas, models trained using PAT and SADS are robust against both single-step and multi-step attacks. Unlike PAT, the proposed SADS method is a single-step adversarial training method.

Engstrom et al. (2018) demonstrated that the performance of models trained using certain adversarial training methods, degrades significantly with the 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 step=1000, it can be observed that there is no significant degradation in the performance of models trained using PAT and SADS methods. In A.5 we show the effect of hyper-parameters of the proposed training method, and in A.4 we show the trend of R_ϵ, train and validation loss during training of models using SADS.

4.2 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

Table 3: White-Box attack: Classification accuracy (%) of models trained on CIFAR-10 dataset using different training methods. For all attacks ϵ=8/255 is used and for PGD attack ϵ_{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		91.52	14.00	0.00	0.00
FAT		92.42	98.58	0.09	0.05
EAT	A	90.80	82.14	10.56	4.69
	B	90.43	60.59	32.76	28.90
	C	90.28	66.49	36.49	29.41
PAT		79.44	53.25	50.53	50.08
SADS		75.93 $\pm$0.28	48.16 $\pm$0.63	44.29 $\pm$0.53	43.48 $\pm$0.49

Table 4: Comparison of training time per epoch of models trained on MNIST and CIFAR-10 datasets respectively, obtained for different training methods. For PAT, $steps$=40 is used for MNIST dataset and $steps$=7 is used for CIFAR-10 dataset. † For EAT, training time of pre-trained source models are not considered.

Method	Training time per epoch (sec.)	
	MNIST	CIFAR-10
NT	~ 2.7	~ 31
FAT	~ 4.1	~ 53
EAT†	~ 5.5	~ 59
PAT	~ 53.0	~ 238
SADS	~ 4.3	~ 56

Table 5: Black-box setting: Performance of models trained on MNIST, Fashion MNIST and CIFAR-10 datasets using different training method, against adversarial attacks in black-box setting. Source models are used for generating adversarial samples, and the target models are tested on these generated adversarial samples.

MNIST					
Source Model		Target Model			
		NT	FAT	PAT	SADS
Model-A	FGSM (ϵ=0.3)	29.09	79.49	96.01	95.06$\pm$0.055
	MI-FGSM (ϵ=0.3, steps=40)	10.69	72.44	95.83	94.80$\pm$0.160
Model-B	FGSM (ϵ=0.3)	28.13	72.39	96.15	95.11$\pm$0.060
	MI-FGSM (ϵ=0.3, steps=40)	12.32	70.79	95.97	94.81$\pm$0.105
Fashion-MNIST					
Model-A	FGSM (ϵ=0.1)	36.66	88.26	81.32	80.86$\pm$0.303
	MI-FGSM (ϵ=0.1, steps=40)	33.04	88.36	81.20	80.68$\pm$0.375
Model-B	FGSM (ϵ=0.1)	39.03	85.40	80.01	78.94$\pm$0.514
	MI-FGSM (ϵ=0.1, steps=40)	38.01	84.72	79.84	78.59$\pm$0.546
CIFAR-10					
VGG-19	FGSM (ϵ=8/255)	36.49	75.92	77.24	75.06$\pm$0.257
	MI-FGSM (ϵ=8/255, steps=7)	14.33	72.47	77.63	75.31$\pm$0.313
ResNet-18	FGSM (ϵ=8/255)	30.77	74.07	77.66	75.19$\pm$0.210
	MI-FGSM (ϵ=8/255, steps=7)	4.65	68.14	77.62	75.36$\pm$0.325

model (target model). We use FGSM and MI-FGSM methods for generating adversarial samples, since samples generated using these methods show good transfer rates (Dong et al., 2018). Table 5 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 when compared to white-box setting.

4.3 Performance against DeepFool and C&W attacks

DeepFool (Moosavi-Dezfooli et al., 2016) and C&W (Carlini & Wagner, 2016) attacks generates 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 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 6, shows the performance of models trained using NT, FAT, PAT and SADS methods, against DeepFool and C&W attacks. It can be observed that for models trained using PAT and SADS methods have relatively large average l_2 norm. Whereas, for models trained using NT and FAT have small average l_2 norm.

Table 6: 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, higher 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	Success	Mean l_2
NT	99.35	1.837	100	1.659	93.73	0.796	100	0.709	94.11	0.162	100	0.102
FAT	99.37	1.455	100	0.798	93.11	1.514	100	1.167	94.66	0.168	100	0.058
PAT	85.68	4.633	99	2.779	90.29	2.635	100	1.572	88.72	1.156	100	0.710
SADS	95.89	3.692	100	2.321	90.68	2.305	100	1.308	87.85	1.089	100	0.682
	±0.06	±0.033	0±	±0.027	±0.26	±0.102	±0	±0.188	±0.22	±0.033	±0	±0.006

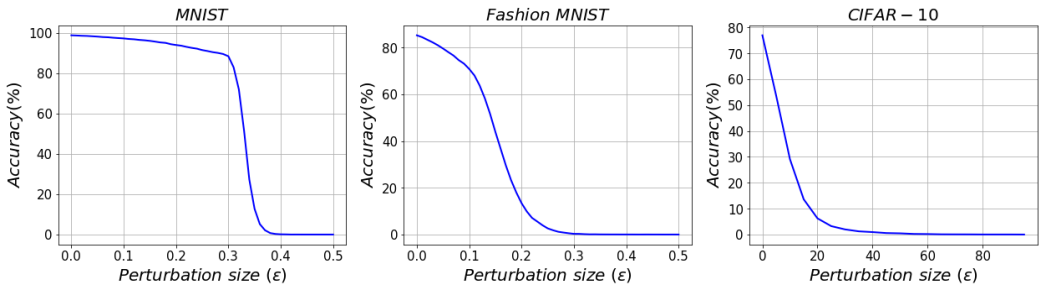

Figure 5: Plot of accuracy versus perturbation size of PGD attack, obtained for models trained using SADS. It can be observed that the accuracy of the model is zero for PGD attack with large perturbation size.

4.4 Tests to rule out Obfuscated Gradients

Athalye et al. (2018) showed that models exhibiting obfuscated gradients are not robust against adversarial attacks. Following are the tests to detect obfuscated gradients:

- ✗ **One-step attacks perform better than iterative attacks**
- ✗ **Black-box attacks are better than white-box attacks**
- ✗ **Unbounded attacks do not reach 100% success**
- ✗ **Increasing distortion bound does not increase success**

We do not observe any of the above trends in models trained using SADS. 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 5, it can be observed that white-box attacks are stronger than black-box attacks for models trained using SADS. Fig. 5 shows the plot of accuracy of 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 (ϵ). Fig. 6 shows the plot of attack success rate (% of test set images that are misclassified by the model) versus perturbation size (ϵ) of PGD attack, obtained for models trained using SADS. It can be observed that attack success rate increases monotonically with increase in the attack perturbation size.

4.5 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 4 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 is due to its 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.

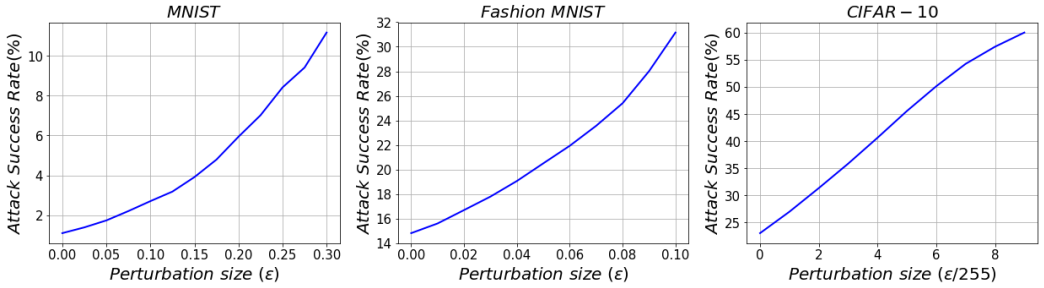

Figure 6: Plot of attack success rate versus perturbation size of PGD attack, obtained for models trained using SADS. Note that, the attack success rate increases monotonically with the increase in the attack perturbation size.

5 Related Works

Following the findings of Szegedy et al. (2013), various attacks (e.g., Goodfellow et al. (2015); Moosavi-Dezfooli et al. (2016); Carlini & Wagner (2016); Mopuri et al. (2017); Dong et al. (2018) have been proposed. Further, in order to defend against adversarial attacks, various schemes such as adversarial training (e.g., Goodfellow et al. (2015); Kurakin et al. (2017); Madry et al. (2018)) and input pre-processing (e.g., Guo et al. (2018); Samangouei et al. (2018)) have been proposed. Athalye et al. (2018) showed that obfuscated gradients give a false of robustness, and broke seven out of nine defense papers (Buckman et al., 2018; Ma et al., 2018; Guo et al., 2018; Xie et al., 2018; Song et al., 2018; Samangouei et al., 2018; Madry et al., 2018; Ma et al., 2018; Dhillon et al., 2018) accepted to ICLR 2018. In this direction, adversarial training method (Madry et al., 2018), shows promising results for learning robust deep learning models. Kurakin et al. (2017) observed that models trained using single-step adversarial training methods are susceptible to multi-step attacks. Further, Tramèr et al. (2018) demonstrated that these models exhibits gradient masking effect, and proposed Ensemble Adversarial Training (EAT) method. During EAT, adversarial samples are generated by the model being trained or by one of the models from the fixed set of pre-trained models. Madry et al. (2018) demonstrated that adversarially trained model can be made robust against white-box attacks, if perturbation computed during training maximizes the loss. On the other hand, works such as Raghunathan et al. (2018) and Wong & Kolter (2017) 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.

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. In order to overcome this over-fitting, we have proposed a novel single-step adversarial training method with dropout scheduling to learn robust models. Unlike existing single-step adversarial training methods, models trained using the proposed method achieve 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 method, and is much faster than multi-step adversarial training method.

References

Anish Athalye, Nicholas Carlini, and David Wagner. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018*, 2018.

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, 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, pp. 387–402, 2013.

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.

Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. Thermometer encoding: One hot way to resist adversarial examples. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=S18Su--CW.

Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. *arXiv preprint arXiv:1608.04644*, 2016.

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*, 2018. URL https://openreview.net/forum?id=H1uR4GZRZ.

Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9185–9193, 2018.

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

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

Chuan Guo, Mayank Rana, Moustapha Cisse, and Laurens van der Maaten. Countering adversarial images using input transformations. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SyJ7ClWCb.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

Ling Huang, Anthony D. Joseph, Blaine Nelson, Benjamin I.P. Rubinstein, and J. D. Tygar. Adversarial machine learning. In *Proceedings of the 4th ACM Workshop on Security and Artificial Intelligence*, AISec '11, 2011.

Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). URL http://www.cs.toronto.edu/~kriz/cifar.html.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016.

Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial machine learning at scale. In *International Conference on Learning Representations (ICLR)*, 2017.

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

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.

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*, 2018. URL https://openreview.net/forum?id=B1gJ1L2aW.

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.

Jan H. Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On Detecting Adversarial Perturbations, 2017.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *IEEE Computer Vision and Pattern Recognition (CVPR)*, 2016.

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.

Nicolas Papernot, Patrick D. McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *arXiv preprint arXiv:1511.04508*, 2015.

Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387. IEEE, 2016.

Nicolas Papernot, Patrick D. McDaniel, Ian J. Goodfellow, Somesh Jha, Z. Berkay Celik, and Ananthram Swami. Practical black-box attacks against deep learning systems using adversarial examples. In *Asia Conference on Computer and Communications Security (ASIACCS)*, 2017.

Aditi Raghunathan, Jacob Steinhardt, and Percy Liang. Certified defenses against adversarial examples. In *International Conference on Learning Representations*, 2018.

Pouya Samangouei, Maya Kabkab, and Rama Chellappa. Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=BkJ3ibb0-.

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*, 2018. URL https://openreview.net/forum?id=rJUYGxbCW.

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(1):1929–1958, 2014.

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)*, 2013.

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.

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

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=Sk9yuql0Z.

Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L. Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

A APPENDIX

A.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 Goodfellow et al. (2015). This method generates l_∞ norm bounded adversarial perturbation based on the linear approximation of loss function.

$$x^* = x + \epsilon.sign(\nabla_x J(f(x;\theta), y_{true})) \tag{3}$$

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

$$x^0 = x \tag{4}$$
$$x^{N+1} = x^N + \alpha.sign\big(\nabla_{x^N} J(f(x^N;\theta), y_{true})\big) \tag{5}$$

Projected Gradient Descent (PGD): Iterative attack method proposed by Madry et al. (2018). Initially, a small random noise is added to the image. Then at each iteration, perturbation of small step size (ϵ_{step}) is added to the image, followed by re-projection.

$$x^0 = x + U\big(-\epsilon_{step}, \epsilon_{step}, shape(x)\big) \tag{6}$$
$$x^{N+1} = x^N + \epsilon_{step}.sign\big(\nabla_{x^N} J(f(x^N;\theta), y_{true})\big) \tag{7}$$
$$x^{N+1} = clip\big(x^{N+1}, min = x - \epsilon, max = x + \epsilon\big) \tag{8}$$

A.2 ADVERSARIAL TRAINING METHODS

In this subsection we explain the existing adversarial training methods.
FGSM Adversarial Training Methods: During training, at each iteration a portion of clean samples in the mini-batch are replaced with its 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): Proposed by Tramèr et al. (2018). At each iteration a portion of clean samples in the mini-batch are replaced with its 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 7 shows the setup used for EAT method.
PGD Adversarial Training Method: Proposed by Madry et al. (2018). At each iteration all the clean samples in the mini-batch are replaced with its corresponding adversarial samples generated using the model being trained. Projected Gradient Descent (PGD) method is used for generating these samples.

Table 7: Setup used for Ensemble Adversarial Training. For MNIST and Fashion-MNIST networks refer table 8.

	Network to be trained	**Pre-trained Models**
CIFAR-10	ResNet-34(Ensemble A)	ResNet-34, ResNet-18
	ResNet-34(Ensemble B)	ResNet-34, VGG-16
	ResNet-34(Ensemble C)	ResNet-18, VGG-16
MNIST and F-MNIST	A(Ensemble A)	A,B,C
	B(Ensemble B)	B, C ,D
	C(Ensemble C)	C, D, A
	D(Ensemble D)	D, A ,B

A.3 SINGLE-STEP ADVERSARIAL TRAINING: TREND OF VALIDATION LOSS

In the main paper, we showed over-fitting effect during the training of LeNet+ on MNIST dataset using single-step adversarial training method. Fig. 7 shows the plot of validation loss, obtained for

Table 8: Architecture of networks used for Ensemble Adversarial Training on MNIST and Fashion-MNIST datasets.

LeNet+	A	B	C	D
conv(32,5,5) + ReLU MaxPool(2,2) conv(64,5,5) + ReLU MaxPool(2,2) FC(1024) + ReLU FC + Softmax	Conv(64,5,5) + ReLU Conv(64,5,5) + ReLU Dropout(0.25) FC(128) + ReLU Dropout(0.5) FC + Softmax	Dropout(0.2) Conv(64,8,8) + ReLU Conv(128,6,6) + ReLU Conv(128,5,5) + ReLU Dropout(0.5) FC + Softmax	Conv(128,3,3) + Tanh MaxPool(2,2) Conv(64,3,3) + Tanh MaxPool(2,2) FC(128) + ReLU FC + Softmax	{ FC(300) +ReLU Dropout(0.5) } × 4 FC + Softmax

ResNet-34 trained on CIFAR-10 dataset using SADS. We observe this over-fitting effect even when model with different architecture is used for generating adversarial validation set. Fig. 8 shows the validation loss obtained for LeNet+ trained on MNIST dataset using single-step adversarial training method. Normally trained models with different architecture is used for generating adversarial validataion set.

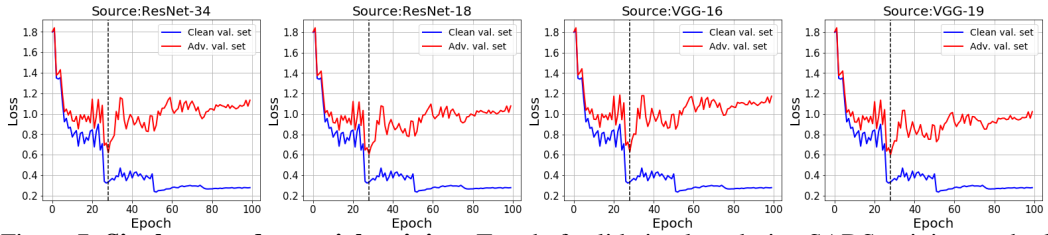

Figure 7: **Single-step adversarial training**: Trend of validation loss during SADS training method, obtained for ResNet-34 trained on CIFAR-10 dataset using SADS. Adversarial validations set is generated using column-1: ResNet-34, column-2: ResNet-18, column-3: VGG-16 and column-4: VGG-19.

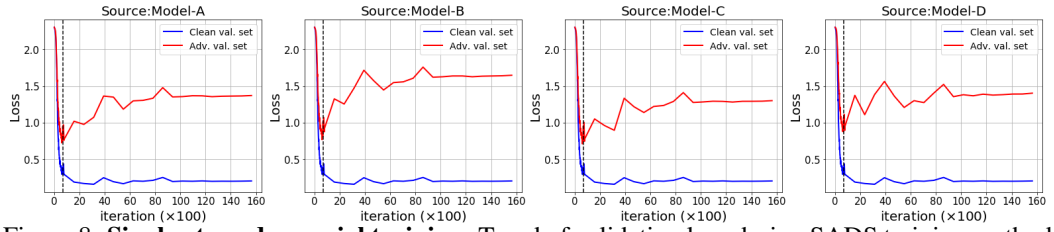

Figure 8: **Single-step adversarial training**: Trend of validation loss during SADS training method, obtained for LeNet+ trained on MNIST dataset using SADS. Adversarial validations set is generated using column-1: Model-A, column-2: Model-B, column-3: Model-C and column-4: Model-D.

A.4 SADS: Trend of R_ϵ, training and validation loss

Fig. 9, 10 and 11 shows the trend of R_ϵ, training and validation loss, obtained for models trained on MNIST, Fashion-MNIST and CIFAR-10 datasets using SADS. It can be observed that for the entire training duration R_ϵ does not decay and no over-fitting effect can be observed.

A.5 Effect of Hyper-Parameters

In order to show the effect of hyper-parameters, we train LeNet+ shown in table 8 on MNIST dataset, using SADS method with different hyper-parameter settings. Validation set accuracy of the model for PGD attack with $\epsilon = 0.3$ and $steps = 40$, is obtained for each 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. Row-1 of Fig. 12 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 model's robustness to decrease, this is due to under-fitting.

Effect of hyper-parameter r_d: The hyper-parameter r_d decides the iteration at which dropout

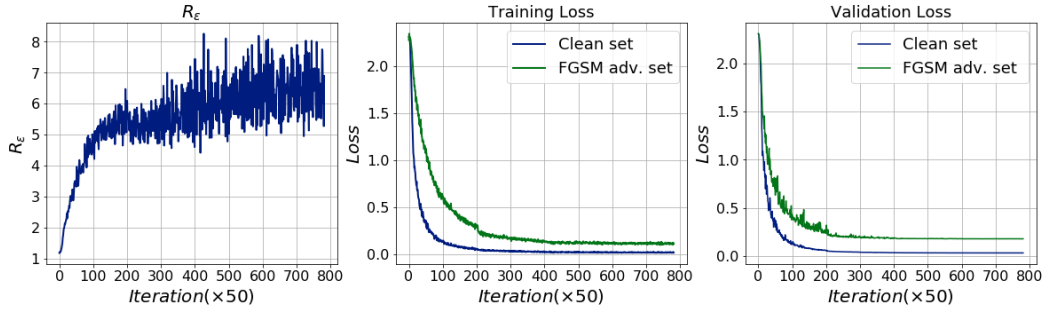

Figure 9: **MNIST**: Trend of R_ϵ, training loss, and validation loss during SADS training method, obtained for LeNet+ trained on MNIST dataset using SADS. Column-1: plot of R_ϵ versus training iteration. Column-2: training loss versus training iteration. Column-3: validation loss versus training iteration. Note that, for the entire training duration R_ϵ does not decay, and no over-fitting effect can be observed.

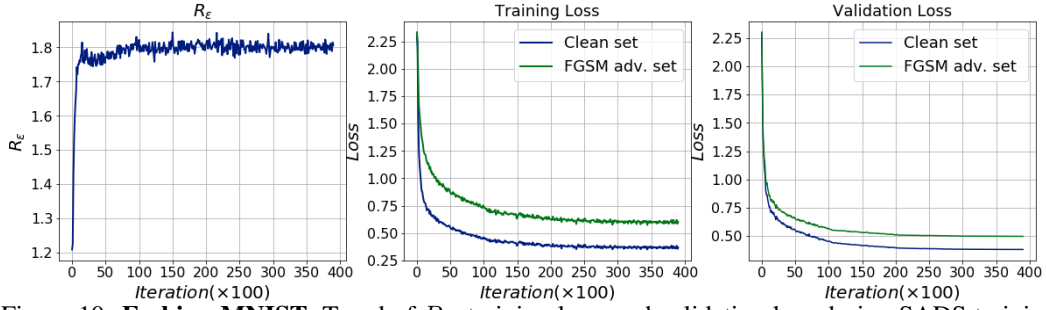

Figure 10: **Fashion-MNIST**: Trend of R_ϵ, training loss, and validation loss during SADS training method, obtained for LeNet+ trained on Fashion-MNIST dataset using SADS. Column-1: plot of R_ϵ versus training iteration. Column-2: training loss versus training iteration. Column-3: validation loss versus training iteration. Note that, for the entire training duration R_ϵ does not decay, and no over-fitting effect can be observed.

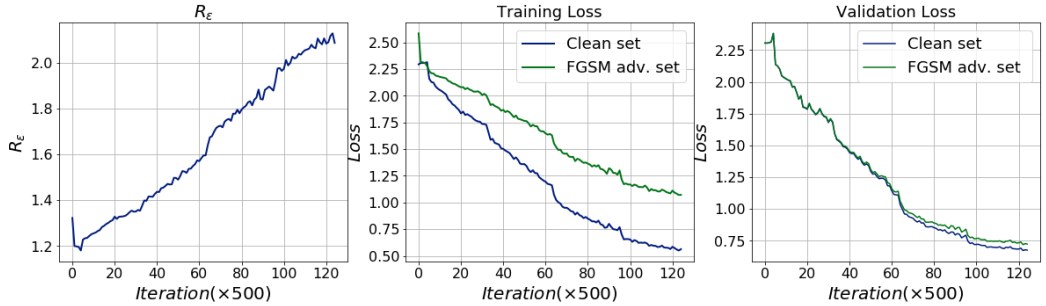

Figure 11: **CIFAR-10**: Trend of R_ϵ, training loss, and validation loss during SADS training method, obtained for ResNet-34 trained on CIFAR-10 dataset using SADS. Column-1: plot of R_ϵ versus training iteration. Column-2: training loss versus training iteration. Column-3: validation loss versus training iteration. Note that, for the entire training duration R_ϵ does not decay, and no over-fitting effect can be observed.

probability reaches zero and is expressed in terms of maximum training iteration. Row-2 of Fig. 12 shows the effect varying r_d from $1/4$ to 1. It can be observed that for $r_d < 0.5$, there is degradation of model's robustness against multi-step attacks. During the initial stages of training learning rate is high and the model can easily over-fit to adversaries generated by single-step method.

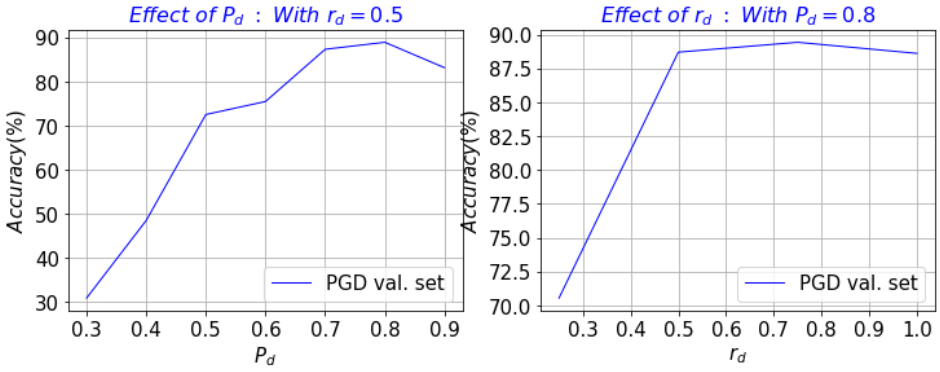

Figure 12: Effect of hyper-parameter P_d and r_d of proposed training method (Algorithm 1)