Supplementary: Single-step Adversarial training with Dropout Scheduling

Vivek B.S. and R. Venkatesh Babu
Video Analytics Lab, Department of Computational and Data Sciences
Indian Institute of Science, Bangalore, India

Contents

1. Network Architecture

Network architecture of models used for MNIST and Fashion-MNIST datasets are shown in Table 1. Model-A and Model-B are used for generating adversarial samples in black-box setting.

2. Adversarial training and Attack generation methods

2.1. 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 [2]. This method generates l_∞ norm bounded adversarial perturbation based on the linear approximation of loss function.

$$x^* = x + \epsilon.sign(\nabla_x J(f(x;\theta), y_{true})) \tag{1}$$

Iterative Fast Gradient Sign Method (IFGSM) [4]: 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 $\alpha = \epsilon/steps$.

$$x^0 = x \tag{2}$$
$$x^{N+1} = x^N + \alpha.sign\big(\nabla_{x^N} J(f(x^N;\theta), y_{true})\big) \tag{3}$$

Projected Gradient Descent (PGD) [5]: Initially, a small random noise sampled from Uniform distribution (U) is added to the image. Then at each iteration, perturbation of small step size (ϵ_{step}) is added to the image, and followed by re-projection.

$$x^0 = x + U\big(-\epsilon_{step}, \epsilon_{step}, shape(x)\big) \tag{4}$$
$$x^{N+1} = x^N + \epsilon_{step}.sign\big(\nabla_{x^N} J(f(x^N;\theta), y_{true})\big) \tag{5}$$
$$x^{N+1} = clip\big(x^{N+1}, min = x - \epsilon, max = x + \epsilon\big) \tag{6}$$

Momentum Iterative Fast Gradient Sign Method (MI-FGSM) [1]: Introduces a momentum term into the IFGSM formulation. Here, μ represents the momentum term. α represents step size and is set to $\epsilon/steps$.

$$x^0 = x \tag{7}$$
$$g^{N+1} = \mu.g^N + \frac{\nabla_{x^N} J(f(x^N;\theta), y_{true})}{||\nabla_{x^N} J(f(x^N;\theta), y_{true})||_1} \tag{8}$$
$$x^{N+1} = x^N + \alpha.sign\big(g^{N+1}\big) \tag{9}$$

Table 1: Architecture of networks used for MNIST and Fashion-MNIST datasets.

LeNet+	Model-A	Model-B	Model-C	Model-D
Conv(32,5,5) + Relu	Conv(64,5,5) + Relu	Dropout(0.2)	Conv(128,3,3) + Tanh	$\left\{ \begin{array}{c} \text{FC(300) +Relu} \\ \text{Dropout(0.5)} \end{array} \right\} \times 4$
MaxPool(2,2)	Conv(64,5,5) + Relu	Conv(64,8,8) + Relu	MaxPool(2,2)	
Conv(64,5,5) + Relu	Dropout(0.25)	Conv(128,6,6) + Relu	Conv(64,3,3) + Tanh	FC + Softmax
MaxPool(2,2)	FC(128) + Relu	Conv(128,5,5) + Relu	MaxPool(2,2)	
FC(1024) + Relu	Dropout(0.5)	Dropout(0.5)	FC(128) + Relu	
FC + Softmax	FC + Softmax	FC + Softmax	FC + Softmax	

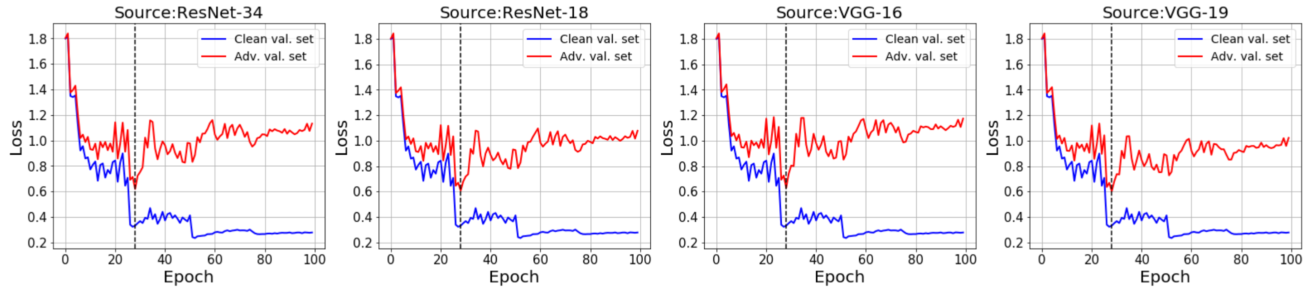

Figure 1: **Single-step adversarial training**: Trend of validation loss during single-step adversarial training, obtained for ResNet-34 trained on CIFAR-10 dataset. Adversarial validation set is generated using column-1: ResNet-34, column-2: ResNet-18, column-3: VGG-16 and column-4: VGG-19.

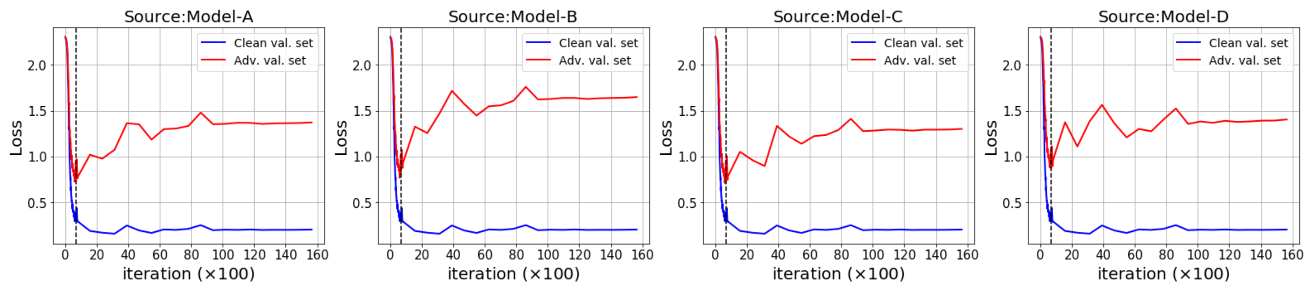

Figure 2: **Single-step adversarial training**: Trend of validation loss during single-step adversarial training, obtained for LeNet+ trained on MNIST dataset. Adversarial validation set is generated using column-1: Model-A, column-2: Model-B, column-3: Model-C and column-4: Model-D.

2.2. Adversarial Training Methods

In this subsection we explain the existing adversarial training methods.

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) [6]: 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 2 shows the setup used for EAT method.

PGD Adversarial Training (PAT): Multi-step adversarial training method proposed by [5]. 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 [8]. 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.

3. 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. 1 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. 2 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.

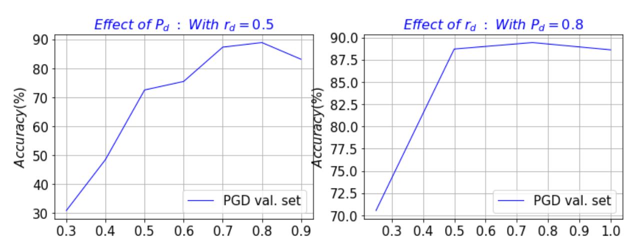

Figure 3: Effect of hyper-parameter P_d and r_d of SADS

4. Effect of Hyper-Parameters

In order to show the effect of hyper-parameters, we train LeNet+ shown in table 1 on MNIST dataset, using SADS with different hyper-parameter settings. Validation set accuracy of the model for PGD attack ($\epsilon = 0.3$ 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. 3 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. 3 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.

5. Comparison with Ensemble Adversarial Training

We train WideResNet-28-10 [7] on CIFAR-10 [3] dataset using EAT and SADS. Table 2 shows the setup used for EAT. Pre-trained models are used for generating adversarial samples during EAT. Table 3 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.

Table 2: Setup used for Ensemble Adversarial Training.

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

6. SADS: Trend of R_ϵ, training and validation loss

Fig. 4 and 5 show the trend of R_ϵ, training and validation loss, obtained for models trained 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.

Table 3: **CIFAR-10: 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 and $steps$=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 ±0.06	51.99 ±1.02	45.66 ±1.26

References

[1] 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*, pages 9185–9193, 2018. 1

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

[3] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research). 3

[4] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*, 2016. 1

[5] 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. 1, 2

[6] 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. 2

[7] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, *Proceedings of the British Machine Vision Conference (BMVC)*, pages 87.1–87.12. BMVA Press, September 2016. 3

[8] 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*, pages 7472–7482, 2019. 2

Figure 4: **MNIST**: Trend of R_ϵ, training loss, and validation loss during SADS training method, obtained for LeNet+ trained on MNIST dataset. Column-1: plot of R_ϵ versus iteration. Column-2: training loss versus iteration. Column-3: validation loss versus iteration. Note that, for the entire training duration R_ϵ does not decay, and no over-fitting effect can be observed.

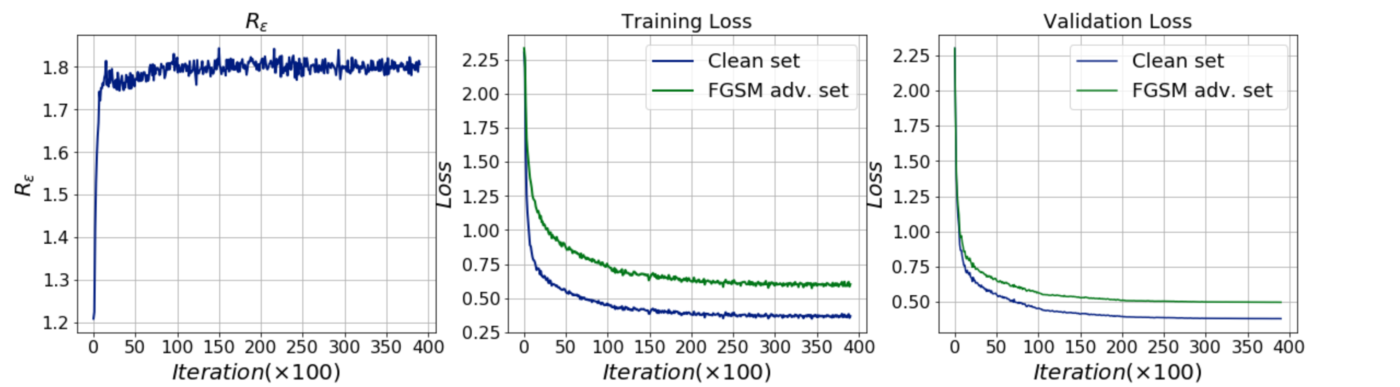

Figure 5: **Fashion-MNIST**: Trend of R_ϵ, training loss, and validation loss during SADS training method, obtained for LeNet+ trained on Fashion-MNIST dataset. Column-1: plot of R_ϵ versus iteration. Column-2: training loss versus iteration. Column-3: validation loss versus iteration. Note that, for the entire training duration R_ϵ does not decay, and no over-fitting effect can be observed.