Plug-And-Pipeline: Efficient Regularization for Single-Step Adversarial Training

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

Abstract

Adversarial Training (AT) is a straight forward solution to learn robust models by augmenting the training mini-batches with adversarial samples. Adversarial attack methods range from simple non-iterative (single-step) methods to computationally complex iterative (multi-step) methods. Although the single-step methods are efficient, the models trained using these methods merely appear to be robust, due to the masked gradients. In this work, we propose a novel regularizer named Plug-And-Pipeline (PAP) for single-step AT. The proposed regularizer attenuates the gradient masking effect by promoting the model to learn similar representations for both single-step and multi-step adversaries. Further, we present a novel pipelined approach that allows an efficient implementation of the proposed regularizer. Plug-And-Pipeline yields robustness comparable to multi-step AT methods, while requiring a low computational overhead, similar to that of single-step AT methods.

1. Introduction

The success of Deep Neural Networks is attributed to large scale annotated datasets. However, DNNs are shown to be susceptible to adversarial attacks [26], wherein an image perturbed with a specially crafted imperceptible noise is misclassified by the model with high confidence. This acts as a deterrent to the real-world deployment [15, 21] of deep models. In the literature, several works have studied the poor generalizability of deep models to physical transformations [10], domain-shift [12, 25] and out-of-distribution samples [17]. Early works explored synthetic augmentation of the training dataset as a solution to improve the generalization. Nevertheless, even such approaches exhibited poor robustness to adversarial noise [26, 13, 7, 20, 11]. This phenomenon has motivated a line of works specifically aiming to defend against adversarial samples [26, 13, 16, 22, 19].

Along this direction, the most successful approaches are adversarial training (AT) methods [13, 18, 5, 31, 4], wherein mini-batches of training data are augmented with adversarial samples generated by the model being trained. Particularly, these samples are generated by crafting an image perturbation (additive noise) via gradient ascent on the loss function. As this systematic procedure approximates the worst-case adversaries, the model is expected to be robust to real-world adversaries. Adversarial training methods can be broadly classified into two categories, non-iterative (single-step AT) and iterative (multi-step AT) methods.

Non-iterative. Notably, non-iterative AT methods such as FGSM-AT [13, 16] require a single additional backpropagation step (gradient ascent) for each mini-batch. While, non-iterative methods are computationally efficient, they yield suboptimal results by converging to a degenerative minimum [27]. This effect is widely known as the gradient masking effect [27], wherein the model prevents the generation of strong adversaries. As a result, models that mask the gradients often show poor robustness to transfer attacks, *i.e*. adversaries generated by other models [27].

Iterative. Methods such as PGD-AT [18] and TRADES [31] perform multiple steps of gradient ascent for each mini-batch. The perturbation added to the image at each iteration is small, which allows them to generate strong adversaries over multiple iterations. Training with such strong adversaries avoids degenerative minimum by preventing masked gradients. Thus, they yield robust models. However, they are computationally intensive, requiring multiple steps of backpropagation for each mini-batch.

In general, we find that the computational efficiency of adversarial training is at odds with the robustness of the trained model. Motivated by this trade-off, we aim to explore an efficient regularization strategy [1, 3] that enjoys the benefit of both iterative and non-iterative methods. To this end, we present Plug-And-Pipeline (PAP), a regularization scheme that can be readily plugged into single-step AT methods to mitigate the effect of gradient masking. The PAP regularizer is scalable, in that it can be applied over a desired fraction of samples in each mini-batch.

The key idea we present in this paper is to harness a pipelined approach for generating multi-step adversaries, such that the computational cost at each mini-batch is com-

*Equal contribution

parable to non-iterative AT methods. Thus, our pipelined approach is designed to be a plug-and-play regularizer that can improve the performance of non-iterative adversarial training methods at minimal computational overhead.
We summarize our prime contributions as follows:

1. We propose Plug-And-Pipeline (PAP), a regularization scheme that yields robustness comparable to multi-step AT methods, while requiring a low computational overhead, similar to that of single-step AT methods.
2. PAP can be enforced on a suitable fraction of the mini-batch, resulting in a scalable framework.

2. Background

Notations. The notations used in this paper are as follows.

Symbol	Description
D	Distribution of clean samples
$\mathcal{U}(a,b)$	Uniform distribution in $[a,b]$
$\mathcal{N}(a,b)$	Normal distribution with mean a and std. b
x, y	Clean image & its ground-truth label
x^*	Potential adversarial sample
$x^{*\,\{i\}}$	Adv. sample at i^{th} step of multi-step attack
f_θ	Neural network parameterized by θ
J	Primary loss function *e.g.* cross-entropy
∇	Gradient operator (Jacobian).
δ	Perturbation added to the image x
ϵ	Maximum strength of the perturbation δ
α	Step size used in iterative methods
N	Number of steps used in iterative methods
γ	Measure of computational overhead

2.1. Classical Adversarial Training

Notably, most adversarial training methods approximately perform the following min-max training strategy,

$$\min_{\theta} \mathop{\mathbb{E}}_{(x,y)\sim D} \Big[\max_{\delta} \; J\big(f_\theta(x+\delta), y\big) \Big] \tag{1}$$

where, J is the standard cross-entropy loss, and δ is a perturbation added to the clean image x, under the L_p norm constraint $||\delta||_p \leq \epsilon$ for some $p \in \mathbb{W}$. The L_p norm constraint is enforced such that the generated adversarial noise (δ) is imperceptible to the human eye. We study adversarial robustness under the L_∞ norm constraint.

In Eq. 1, the inner-maximization step corresponds to the generation of worst-case adversaries, while the outer minimization step aims to achieve a low error rate on such samples. The key idea of adversarial training is to prevent a drastic change in the model's prediction when an imperceptible noise is added to the image. We now describe some popular approaches for generating adversarial samples.

a) Fast Gradient-Sign Method (FGSM) [13]. In this method, a single-step gradient ascent is performed on the loss J to obtain the perturbation δ that maximizes the loss,

$$\begin{aligned} x^* &= x + \delta \\ \delta &= \epsilon \cdot \text{sign}\Big(\nabla_x J(f_\theta(x), y)\Big) \end{aligned} \tag{2}$$

Intuitively, this process entails the first order approximation of the loss function J, which is expected to be maximized via a single gradient ascent step of size ϵ. While this method is computationally efficient, the first order approximation is often not accurate as the presence of non-linear activation functions disturbs the local-lipschitzness [8, 23, 30] of the loss landscape. Therefore, models trained with single-step adversaries exhibit a sharp curvature in the neighborhood of the training samples, which causes a highly irregular (noisy) gradient pattern [27].

As a result, the gradient computed at x ($\text{sign}(\nabla_x(\cdot))$), does not represent the true direction corresponding to the loss maximization. Thus, a single perturbation of strength ϵ does not effectively maximize the loss J. This is the gradient masking effect [27] observed in the models trained with FGSM-AT. Such models show poor robustness to iterative attacks that are more effective in maximizing the loss.

b) Iterative FGSM (IFGSM) [15]. A more effective approach to approximate the inner-maximization step (Eq. 1) is to perform a fine-grained gradient ascent over several iterations. Here, FGSM is applied for N iterations, with a perturbation of strength α in each iteration ($\alpha = \epsilon/N$) as,

$$\begin{aligned} x^{*\,\{i+1\}} &= x^{*\,\{i\}} + \delta_i \\ \delta_i &= \alpha \cdot \text{sign}\Big(\nabla_{x^{*\{i\}}} J\big(f_\theta(x^{*\,\{i\}}), y\big)\Big) \end{aligned} \tag{3}$$

where, $i = 0, 1, ..., N-1$, and $x^{*\,\{0\}} = x$, the clean sample. The adversarial sample is obtained at the last iteration, *i.e.* $x^* = x^{*\,\{N\}}$. A fine-grained iterative optimization with a small step-size (α) yields perturbed samples x^* that are more likely to fall in a local maxima around the neighborhood of the clean sample x. Performing the outer-minimization (Eq. 1) for such samples ensures that the loss landscape is smooth and the model prediction does not drastically change around the neighborhood of the clean sample.

c) Projected Gradient Descent (PGD) [18]. This method is a variant of IFGSM, where a random perturbation is added to the image before the first step, and re-projection is preformed after every step. Specifically,

$$\begin{aligned} x_0 &= x + \mathcal{U}(-\epsilon, \epsilon) \\ x^{*\,\{i+1\}} &= x^{*\,\{i\}} + \delta_i \\ \delta_i &= \alpha \cdot \text{sign}\Big(\nabla_{x^{*\{i\}}} J\big(f_\theta(x^{*\,\{i\}}), y\big)\Big) \end{aligned} \tag{4}$$

where, δ_i is within the ϵ-bound, *i.e.* $\text{clip}_\epsilon(\delta_i)$, and, usually $\epsilon/N < \alpha \leq \epsilon$. The stochasticity resulting from the initial random perturbation further prevents a degenerative minimum, yielding stronger adversarial samples.

A key observation is that FGSM-AT tends to overfit on the FGSM samples early on during the training [28, 5]. In other words, instead of exhibiting robustness to adversaries, the model resists the generation of plausible adversarial samples. On the other hand, though IFGSM and PGD require significantly more computation, they generate strong adversarial samples, owing to a fine-grained iterative process. This mitigates the gradient masking effect.

2.2. Generalization Trade-off

While iterative methods yield robust models, the min-max formulation in Eq. 1 depends upon the extent to which the inner-maximization and outer-minimization steps are approximated. Note, classical AT methods optimize a single objective function $\rho(\theta)$ (*i.e.* training occurs only with adversarial samples). As a result, adversarial training often degrades the performance on the clean samples [29], due to the limited flexiblity offered by the formulation in Eq. 1.

TRADES [31] formally studied this trade-off between the accuracy on the clean samples and the generalization to adversarial samples. Particularly, [31] proposed the following,

$$\begin{aligned} &\min_{\theta} \quad \mathop{\mathbb{E}}_{(x,y)\sim D} \quad \rho(\theta) \\ &\rho(\theta) = J\big(f_\theta(x), y\big) + \max_{\delta} \; \eta \cdot KL\big(f_\theta(x), f_\theta(x+\delta)\big) \end{aligned} \tag{5}$$

where, J is the cross-entropy loss, KL is the KL-divergence between the softmax distributions of the output $f_\theta(\cdot)$ for the clean image x and the perturbed image $x^* = x + \delta$. Here, η is a hyperparameter that controls the trade-off between the cross-entropy loss J (*i.e.* learning to classify clean images), and the KL-divergence between the clean and the perturbed image outputs (*i.e.* generalization to the perturbed images). TRADES demonstrated state-of-the-art robustness to adversarial samples. However, similar to PGD-AT, the inner-maximization (Eq. 5) is performed over N iterations, to obtain a perturbation δ that maximizes KL.

We observe that to effectively approximate the inner-maximization in both Eq. 1 and Eq. 5, one has to perform multiple steps of gradient ascent. This introduces additional computational burden, on top of the compromise made on the accuracy on clean samples, for improving the robustness to adversarial samples. Motivated by this trade-off, we aim to propose a regularizer that aids in minimizing the computational effort in adversarial training. Particularly, we propose a regularizer which aims to improve the effectiveness of the inner-maximization step in Eq. 1 performed in single-step AT methods, by mitigating the gradient-masking effect.

2.3. Computational Efficiency

Before formalizing the approach, we qualitatively analyze the primary computational overhead of adversarial training, *i.e.* the generation of adversarial samples. Accordingly, we define the following qualitative measure.

Definition 1. (γ-FGSM). *For a given mini-batch of m clean samples, γ-FGSM indicates that on an average (averaged over the number of clean samples m), γ backpropagation operations are required to generate adversarial samples.*

Notionally, we say that an adversarial training method is γ-FGSM, if the method requires a backpropagation operation equivalent to that of γ FGSM steps, to generate an adversarial sample corresponding to a clean sample. Thus, γ signifies an expected computational overhead for generating adversarial samples during training.

Note that, for every epoch of adversarial training, the backpropagation computation would be equivalent to that of $1 + \gamma$ epochs of normal training (ceteris paribus). Further, γ is independent of the underlying loss formulation. Thus, a computationally efficient adversarial training strategy should achieve a reasonably low value of γ.

By definition, FGSM-AT has $\gamma = 1$, *i.e.* for each clean sample x, the major overhead incurred to generate the corresponding FGSM sample x^* is a single backpropagation (gradient ascent). Similarly, for iterative methods using N iterations (PGD-AT and TRADES), we have $\gamma = \frac{N \cdot m}{m} = N$, since each clean sample requires N backpropagations to obtain the corresponding adversarial sample $x^* = x^{*\{N\}}$.

Suppose that FGSM-AT is performed on a fraction of the mini-batch [16], *i.e.*, given a mini-batch of m clean samples, we generate FGSM samples only for n samples ($n < m$) while $m-n$ samples are unchanged. Here, $\gamma = \frac{n}{m}$, signifying that on an average, $\frac{n}{m} < 1$ additional backpropagations are required to generate adversarial samples. Intuitively, an adversarial training strategy with $\gamma < 1$ is expected to be more efficient than FGSM-AT ($\gamma = 1$). However, single-step AT methods generally do not yield promising robustness, due to gradient masking, which makes approaches with $\gamma < 1$ infeasible according to the current literature.

Nevertheless, we aim to address the significant computational overhead incurred in training robust models. To this end, we propose a regularizer that significantly improves the performance of single-step AT by mitigating the effect of gradient masking. Our regularizer named Plug-And-Pipeline (PAP), when used in FGSM-AT has $\gamma = 1 + \gamma'$, where, the first term (1) is due to the generated FGSM samples in FGSM-AT while the second term (γ') is the computational overhead of the samples taking part in regularization. In the next section, we demonstrate the design of our regularizer that achieves $\gamma' < 1$.

3. Approach

In this section, we present our approach viz. Plug-And-Pipeline (PAP) regularizer. Motivated from the discussion in Sec. 2, we systematically identify the challenges involved to formalize our approach. An overview is given in Fig. 1.

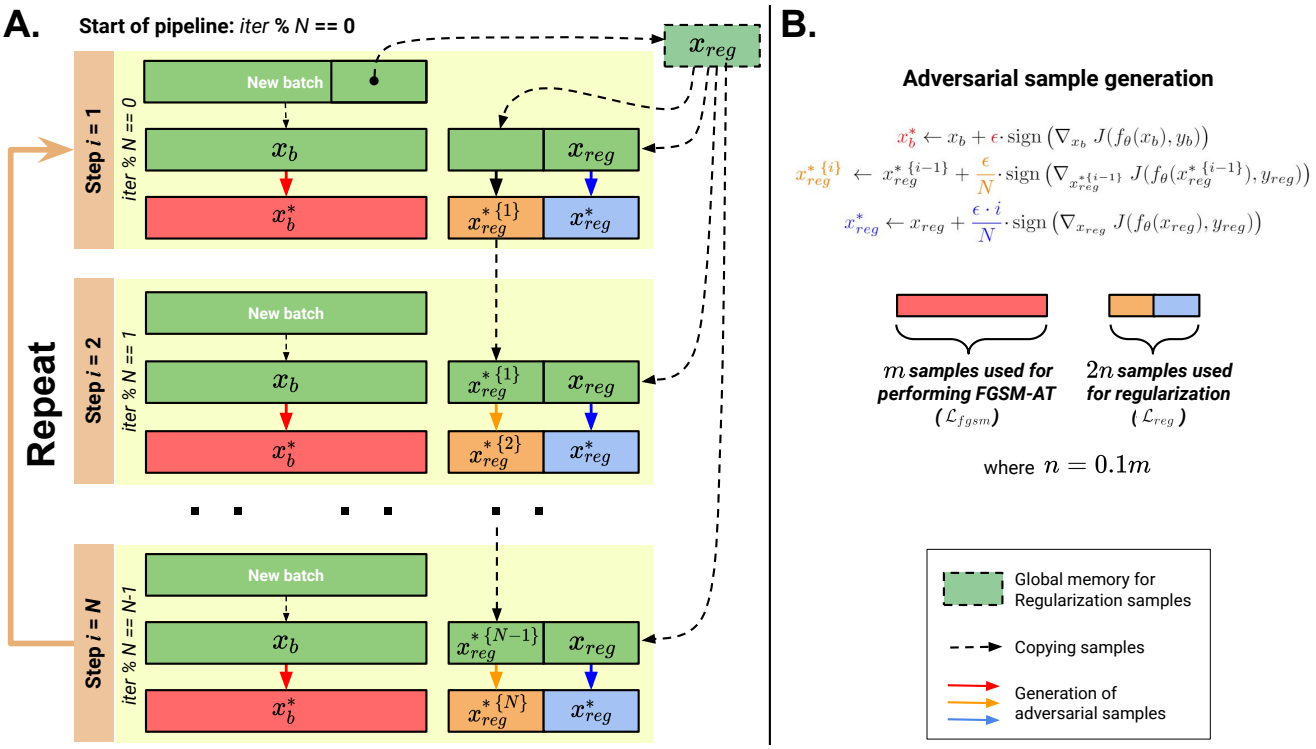

Figure 1. **A.** Overview of the proposed regularization scheme. During FGSM-AT (using samples x_b^*), we utilize a fraction of the mini-batch (x_{reg}) to perform regularization. **B.** Description of the notations used in A.

3.1. Mitigating Gradient Masking

a) Challenges. To generate a single-step adversarial sample such as FGSM, a large perturbation (strength ϵ) is added to the clean image along the direction of the gradient $\nabla_x(\cdot)$. In such methods, the loss is minimized on the adversarial sample $x + \delta$ (Eq. 2). However, this does not guarantee a suitable loss landscape between the clean image x and the perturbed image $x + \delta$. As a result, the loss landscape possesses sharp curvatures [27] in the intermediate region $[x, x + \delta]$, thereby yielding false gradient directions $\nabla_x(\cdot)$. Thus, we must ensure a smooth (locally-lipschitz [30]) loss landscape around the neighborhood of the clean samples.

b) Solution. Arguably, the most effective way to attenuate the gradient masking effect is to train with iterative adversarial samples *e.g.* IFGSM. We observe that IFGSM uses a fine-grained iterative optimization to effectively identify local-maxima around the neighborhood of the clean samples, while FGSM samples are less effective in this regard. This results in a discrepancy between the output distributions of the FGSM and IFGSM samples, generated using the same perturbation strength for a given image. We hypothesize that in order to reduce this discrepancy between the FGSM and IFGSM outputs, the model must achieve a smooth loss landscape. With this intent, we formalize our regularizer as a loss between the pre-softmax activations of the FGSM and the IFGSM samples for a given clean image.

A naive implementation of such a regularizer can be as follows. For a clean sample x, with a perturbation strength ϵ, generate an FGSM sample x^* (Eq. 2) and an N-step IFGSM sample $x^{*\{N\}}$ (Eq. 3), and penalize the discrepancy in the pre-softmax distributions using the mean-squared-loss, *i.e.* $||f_\theta(x^*) - f_\theta(x^{*\{N\}})||_2^2$. However, this would entail $\gamma = 1 + N$, due to the generation of FGSM and N-step IFGSM samples for each mini-batch of clean samples. Thus, we aim to explore a pipelined approach for the generation of IFGSM samples, described as follows.

3.2. Amortized Computational Costs

a) Challenges. In general, we observe that the representations learned by a model do not change drastically within a few training iterations. Leveraging this property one could pipeline IFGSM-AT as follows.

Consider, a fixed set of clean images x for N iterations. At each iteration i, perform a single gradient ascent step of IFGSM (with $\alpha = \frac{\epsilon}{N}$) on the sample $x^{*\{i-1\}}$ (previous iterative sample) to obtain the sample $x^{*\{i\}}$. Thereafter, update the model parameters θ_i by imposing a cross-entropy loss on $x^{*\{i\}}$. The next iteration of IFGSM $(i + 1)$ is performed on the new parameters θ_{i+1}. In this manner, our regularizer could also be pipelined while performing FGSM-AT. At each iteration i, one could generate the i^{th} step IFGSM sample and an FGSM sample with an equivalent strength $i \cdot \alpha$, and penalize the resulting discrepancy.

Methods such as [24] employ such a pipelined approach. However, [24] "replays" a mini-batch of clean samples for N iterations, resulting in $\gamma = N$. Likewise, the regularizer described above would yield $\gamma = 2N$, since for N replays, we perform an FGSM and an IFGSM step.

b) Solution. We propose a two-fold solution to the above challenge. Firstly, we separate the FGSM-AT process from the regularization process. More concretely, we do not replay each mini-batch of clean samples. Instead, as shown in Fig. 1A, we maintain a separate set of samples which are used for regularization, while FGSM-AT proceeds normally. Secondly, to obtain these samples, we randomly select a fraction k of the clean samples ($n = km$) at every N^{th} mini-batch. Thus, for each mini-batch of m clean samples, we generate m FGSM samples for FGSM-AT, and $2n = 2km$ samples taking part in the regularization (FGSM and IFGSM). In our experiments, we use $k = 0.1$, *i.e.* 10% of samples. This reduces to $\gamma = 1 + 0.2 = 1.2$.

3.3. Training Algorithm

In Fig. 1A, we illustrate N iterations of FGSM-AT with the PAP regularizer. At each iteration $iter$, a new mini-batch of m clean samples x_b is used to generate samples for FGSM-AT (x_b^*). Additionally, at every N^{th} mini-batch, we randomly select $n = 0.1m$ clean samples for regularization (x_{reg}). Note that these samples are loaded into a global memory for N iterations while the mini-batch is refreshed with new clean samples (x_b) at each iteration (see Fig. 1A).

At each mini-batch, FGSM-AT is performed by generating FGSM samples x_b^* (Fig. 1B) and imposing the loss,

$$\mathcal{L}_{fgsm} = \frac{1}{m} \sum J(f_\theta(x_b^*), y_b) \tag{6}$$

where the summation is over m FGSM samples. Simultaneously, we generate the samples for regularization, *i.e.*, the i^{th} IFGSM sample $x_{reg}^{*\,\{i\}}$ with $\alpha = \epsilon/N$ from $x^{*\,\{i-1\}}$ and a single-step FGSM sample x_{reg}^* with an equivalent perturbation strength $(\epsilon \cdot i)/N$ (Fig. 1B) and enforce,

$$\mathcal{L}_{reg} = \frac{1}{n} \sum ||f_\theta(x_{reg}^*) - f_\theta(x_{reg}^{*\,\{i\}})||_2^2 \tag{7}$$

where $n = 0.1m$ is the number of samples selected for regularization and $||\cdot||_2$ represents the L_2 norm of the difference between the pre-softmax output vectors produced by the single-step and iterative samples used for regularization. Thus, the total loss employed at each iteration is,

$$\mathcal{L} = \mathcal{L}_{fgsm} + \lambda \cdot \mathcal{L}_{reg} \tag{8}$$

where λ is a suitable hyperparameter. We summarize the algorithm in Algo. 1. The selection of samples for regularization is performed in lines 7-10. Lines 11-15 demonstrate the generation of adversarial samples, which is followed by training the model with the total loss $\mathcal{L}$ (lines 16-17).

Algorithm 1 Plug-And-Pipeline (PAP)

1: **require:** Training dataset of clean images (x, y), Number of epochs n_{epochs}, Model parameters θ.

2: **notations:** x^* is single-step adversarial sample, $x^{*\,\{i\}}$ is i^{th} step adversarial sample of an iterative attack.

3: $iter \leftarrow 0$

4: **for** epoch in n_{epochs} **do**

5: Sample a mini-batch (x_b, y_b) from the train set

6: $i \leftarrow (iter\%N) + 1$ */* Initialize counter */*

7: **if** iter $\%N$ == 0 **then**

/* Select samples for regularization*/

8: $(x_{reg}, y_{reg}) \leftarrow$ select 10% of (x_b, y_b)

9: $x_{reg}^{*\,\{0\}} \leftarrow x_{reg}$

10: **end if**

/* Generate FGSM samples for FGSM-AT */

11: $x_b^* \leftarrow x_b + \epsilon \cdot \text{sign}\left(\nabla_{x_b} J(f_\theta(x_b), y_b)\right)$

/* Generate FGSM samples for Regularization */

12: $\delta_1 \leftarrow \frac{\epsilon \cdot i}{N} \cdot \text{sign}\left(\nabla_{x_{reg}} J(f_\theta(x_{reg}), y_{reg})\right)$

13: $x_{reg}^* \leftarrow x_{reg} + \delta_1$

/* Generate IFGSM samples for Regularization */

14: $\delta_2 \leftarrow \frac{\epsilon}{N} \cdot \text{sign}\left(\nabla_{x_{reg}^{*\{i-1\}}} J(f_\theta(x_{reg}^{*\{i-1\}}), y_{reg})\right)$

15: $x_{reg}^{*\,\{i\}} \leftarrow x_{reg}^{*\,\{i-1\}} + \delta_2$

16: Compute total loss $\mathcal{L} = \mathcal{L}_{fgsm} + \lambda \cdot \mathcal{L}_{reg}$

17: Update parameters (θ) by minimizing $\mathcal{L}$

18: $iter \leftarrow iter + 1$

19: **end for**

Note that this process results in a plug-and-play regularizer that can be readily integrated into any single-step AT method. The regularizer mitigates the gradient-masking effect by encouraging the output distributions of single-step and iterative attacks to be similar, thereby improving the efficacy of the single-step adversarial samples (here, the generated FGSM samples x_b^*). Furthermore, this approach is highly efficient requiring a mere 10% additional batch size. The pipeline also confers a fine-grained regularization, by amortizing the discrepancy minimization $\mathcal{L}_{reg}$ in the intermediate region between the clean sample x and the corresponding adversarial sample $x + \delta$. One can perceive this as a curriculum [2], where the minimization of loss $\mathcal{L}_{reg}$ gradually occurs from $i = 1$ (easiest) to $i = N$ (hardest).

Extension to TRADES. Having proposed the PAP regularizer for FGSM-AT, we now demonstrate the applicability of PAP to other approaches. As a use-case, we consider TRADES [31] that also exhibits gradient masking when the inner-maximization step (Eq. 5) is approximated using a

Table 1. Experimental setup. For MNIST dataset, we use network described in [31].

			FGSM [13]	IFGSM [15]			PGD[18]			MIFGSM [9]		
Datasets	Network	n_{epochs}	ϵ	ϵ	α	N	ϵ	α	N	ϵ	α	N
MNIST	Small CNN [31]	50	0.3	0.3	ϵ/N	40, 100	0.3	0.01	40, 100	0.3	ϵ/N	100
CIFAR-10	ResNet-18 [14]	100	8/255	8/255	ϵ/N	7, 20	8/255	2/255	7, 20	8/255	ϵ/N	20

single-step gradient ascent. We propose to integrate the PAP regularizer into a single-step TRADES formulation as,

$$\mathcal{L} = \frac{1}{m}\sum J(f_\theta(x_b), y_b) + \eta \cdot KL(f_\theta(x_b), f_\theta(x_b^*)) + \lambda \cdot \frac{1}{n}\sum ||f_\theta(x_{reg}^*) - f_\theta(x_{reg}^{*\{i\}})||_2^2 \quad (9)$$

where the $J(\cdot)$ and $KL(\cdot)$ correspond to the TRADES loss function (Eq. 5), while the last term is the PAP regularizer. Note that KL in Eq. 9 is obtained on the softmax distributions of the outputs $f_\theta(\cdot)$, while the regularizer is applied at the pre-softmax layer. Here, x_b^* is obtained as,

$$x_b^* \leftarrow x_b + \epsilon \cdot \text{sign}\left(\nabla_{x_b} KL(f_\theta(x_b), f_\theta(x_b + \delta'))\right) \quad (10)$$

where, δ' is a small random perturbation. Following [31], we use $\delta' = 0.001 \cdot \mathcal{N}(0, 1)$). Further, note that the process used to generate the samples x_{reg}^* and $x_{reg}^{*\{i\}}$ remains the same as in Fig. 1. Similar to FGSM-AT, this formulation of single-step TRADES combined with the PAP regularizer achieves $\gamma = 1.2$, in contrast to the regular TRADES formulation with N steps ($\gamma = N$).

4. Experimental Results

In this section, we demonstrate the performance of models trained using Plug-And-Pipeline against adversarial attacks in white-box and black-box settings. We perform sanity checks described in [6] to verify the adversarial robustness of models. Following this, we perform a rigorous validation of our approach.

a) Experimental Setup. Table 1 shows the experimental setup. Note that, a method such as [24] that "replays" a mini-batch of clean samples for N iterations, must reduce the number of epochs by N times (since, $\gamma = N$ for such a method). However, our approach allows us to aim for a total number of epochs similar to standard FGSM training owing to $\gamma = 1.2$. We follow Madry et al. [18] for attack parameter settings. We refer FGSM-AT with the proposed regularization as "PAP FGSM" (Eq. 8), and single-step TRADES with the proposed regularization as "PAP TRADES" (Eq. 9).

b) Threat Model. The threat model that we consider in this work is in line with [18, 31]. The goal of an adversarial attack method is to craft an imperceptible small perturbation δ for an image x such that the resulting adversarial image x^* is misclassified by the model, *i.e.* the adversarial noise δ is bounded within L_∞ norm. The goal of a defense method is to exhibit robustness against these adversarial attacks.

Table 2. Hyper-parameters of Plug-And-Pipeline. Where k is the number of clean samples used for regularization and is expressed in terms of percentage of mini-batch size. λ represents the weightage given to the regularization.

		k	λ
MNIST	PAP FGSM	10%	0.5
	PAP TRADES		0.5
CIFAR-10	PAP FGSM		0.1
	PAP TRADES		1

c) Training. We train models on MNIST and CIFAR-10 datasets for 50 and 100 epochs respectively. For FGSM-AT [16], PGD-AT [18] and TRADES [31], we follow training procedure described in the respective papers. Table 2 shows the hyper-parameter setting of Plug-And-Pipeline.

4.1. White-box setting

Table 3 and 4 shows the performance of models trained using Plug-And-Pipeline in a white-box setting. It can be observed that models trained using Plug-And-Pipeline are robust against non-iterative and iterative attacks. Whereas, models trained using FGSM-AT are susceptible to iterative attack i.e. IFGSM and PGD. Further, the performance of models trained using the proposed training method is on-par with models trained using multi-step adversarial methods which are computationally expensive.

Ablation. We perform ablation study to show the effectiveness of the proposed regularizer. We train ResNet-18 on CIFAR-10 dataset using the proposed method with $\lambda = 0$. However, we compute cross-entropy loss J on all samples, *i.e.* on both FGSM (x_b^*) and pipelined adversarial samples (x_{reg}^* and $x_{reg}^{*\{i\}}$) at each iteration. The last row of Table 4, shows the performance of this model. It can be observed that the model is susceptible to adversarial attacks. The efficacy of our regularizer is attributed to the enforcement of local smoothness in the loss landscape.

4.2. Black-box setting

In a black-box setting, transfer attacks are performed by a normally trained model of the same architecture, typically referred to as a source model. Source model is used for generating adversarial samples and these samples are transferred to the target models. Adversarial samples are generated using MIFGSM, since this method is effective in a black-box setting [9]. Table 5 shows the performance of models in a black-box setting. It can be observed that the susceptibility of models trained using Plug-And-Pipeline (ours) is less than compared in a white-box setting.

Table 3. MNIST: Recognition accuracy (%) of models in a white-box attack setting. We refer FGSM-AT with the proposed regularization as "PAP FGSM" (Eq. 8), and single-step TRADES with the proposed regularization as "PAP TRADES" (Eq. 9).

Method	Clean	FGSM	IFGSM		PGD	
			N=40	N=100	N=40	N=100
Normal	99.19	28.20	4.30	4.13	1.43	0.02
FGSM-AT	99.29	98.61	29.90	28.14	22.05	12.41
PGD-AT	99.26	96.95	95.72	95.69	95.33	94.57
TRADES	99.48	97.86	96.51	96.50	96.13	95.15
PAP FGSM	99.29	96.69	95.58	95.56	95.05	94.04
PAP TRADES	99.10	97.11	96.07	96.05	95.62	94.66

Table 4. CIFAR-10: Recognition accuracy (%) of models in a white-box attack setting.

Method	Clean	FGSM	IFGSM		PGD	
			N=7	N=20	N=7	N=20
Normal	92.45	14.44	0.01	0.00	0.00	0.00
FGSM-AT	91.46	98.30	1.27	0.48	0.17	0.00
PGD-AT	81.42	55.62	52.45	52.12	50.77	48.87
TRADES	81.66	56.43	53.37	52.93	51.75	50.21
PAP FGSM	80.25	52.71	49.63	49.32	48.10	46.34
PAP TRADES	79.05	54.27	51.55	51.36	50.10	48.71
Ablation	81.06	97.78	21.31	17.29	0.78	0.00

4.3. Sanity Checks

In this subsection, we perform sanity checks prescribed in [6] to verify the robustness of the models trained using the Plug-And-Pipeline regularizer.

- *Iterative attacks should be stronger than non-iterative attacks in a white-box setting:* From Table 3 and 4 it can be observed that iterative attacks i.e. IFGSM and PGD are stronger than non-iterative attack i.e. FGSM.
- *White-box attacks should be stronger than black-box attacks:* In Table 5 it can be observed that MIFGSM attack in a white-box setting is stronger than MIFGSM in a black-box setting.
- *Model's accuracy should degrade with increase in the perturbation size of an attack:* From Fig. 2 it can be observed that the model's accuracy decreases with increase in the perturbation size of PGD attack.
- *Model's accuracy should reach zero/random for an attack with large perturbation size:* From Fig. 2 it can be observed that the model's accuracy reaches zero for PGD attack with large perturbation size.
- *Untargeted attacks should be stronger than targeted attacks:* From Table 6 it can be observed that untargeted attack i.e. PGD is stronger than targeted attacks i.e. PGD-R (random class is chosen as a target class) and PGD-LL (least likely prediction of the model is chosen as a target class). Note that, targeted attacks generate an adversarial perturbation that aims at improving the confidence of the target class.

Table 5. Performance of models trained on MNIST and CIFAR-10 dataset against MIFGSM attack in white-box and black-box settings. Normally trained model is used for generating MIFGSM adversarial samples in a black-box setting, and the generated adversarial samples are transferred to the target models.

Method	MNIST		CIFAR-10	
	MIFGSM (N=100)		MIFGSM (N=20)	
	White-box	Black-box	White-box	Black-box
FGSM-AT	46.52	75.74	3.01	72.85
PGD-AT	95.67	96.44	53.45	79.30
TRADES	96.59	96.99	54.34	79.56
PAP FGSM	95.57	96.26	50.70	78.31
PAP TRADES	95.91	96.59	52.42	76.77

Table 6. Comparison between untargeted and targeted attacks. (i) untargeted attack: PGD, (ii) targeted attacks: PGD-R and PGD-LL. We set the attack steps (N) equal to 100 and 20 for MNIST and CIFAR-10 datasets respectively.

Method	MNIST			CIFAR-10		
	PGD	PGD-R	PGD-LL	PGD	PGD-R	PGD-LL
PAP FGSM	94.04	98.30	98.65	46.34	72.25	77.43
PAP TRADES	94.66	98.33	98.81	48.71	72.47	76.70

Table 7. Computational overhead of adversarial training methods, measured in terms of γ. Refer to Sec. 2.3 for the definition of γ.

	γ	
	MNIST	CIFAR-10
FGSM-AT	1	1
PGD-AT	40	7
TRADES	40	7
PAP (ours)	1.2	1.2

4.4. PGD attack with random restarts

In Fig. 3, we show the performance of models trained using Plug-And-Pipeline against PGD attack with various random restarts. It can be observed that the model's accuracy decreases slightly and then saturates with the increase in the number of random restarts, suggesting that the model is truly robust for such samples.

4.5. Computational Overhead

In Table 7, we compare the computational overhead of Plug-And-Pipeline (PAP) against other approaches. Adversarial samples generation is the major overhead in adversarial training. We measure this using γ-FGSM (Sec. 2.3).

For multi-step AT, $\gamma = N$ (where N is the number of iterations used to generate adversarial samples), while for single-step AT $\gamma = 1$. In PGD-AT and TRADES, adversarial samples are generated using multi-step attacks with steps equal to 40 and 7 for MNIST and CIFAR-10 respectively.

It can be observed that PGD-AT and TRADES have the highest computational overhead, where as FGSM-AT has the lowest computational overhead. The proposed PAP regularizer confers robustness similar to iterative methods, while having significantly less computational overhead.

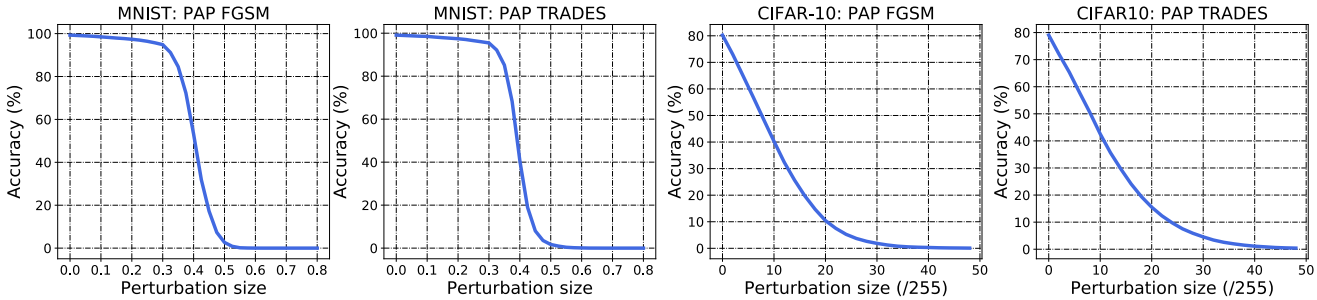

Figure 2. Recognition accuracy (%) of models trained using Plug-And-Pipeline, for PGD attack with different perturbation size (ϵ). Note that, models' accuracy decreases with increase in the perturbation size (ϵ).

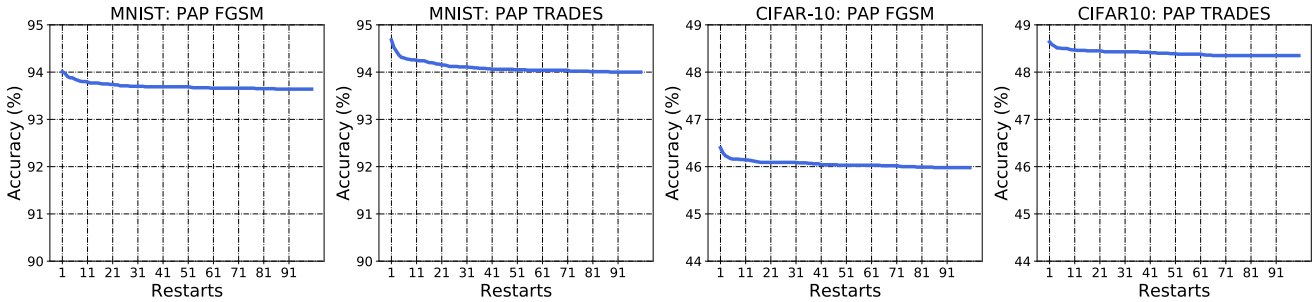

Figure 3. Recognition accuracy (%) of models trained using Plug-And-Pipeline, for PGD attack with different number of random restarts. Note that, the models' accuracy does not decrease drastically with increase in the number of random restarts of PGD attack.

5. Conclusion

In this work, we proposed a novel regularizer for single-step adversarial training methods to suppress gradient masking effect. The proposed regularizer mitigates the gradient masking effect by promoting the model to learn similar final representation for both single-step and multi-step adversaries. In this regard, a computationally efficient pipeline approach is proposed to generate multi-step adversaries. Finally, the performance of models trained using Plug-And-Pipeline is on par with models trained using existing multi-step adversarial training methods.

References

[1] Sravanti Addepalli, Vivek B.S., Arya Baburaj, Gaurang Sriramanan, and R. Venkatesh Babu. Towards achieving adversarial robustness by enforcing feature consistency across bit planes. In *CVPR*, 2020.

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *ICML*, 2009.

[3] Vivek B.S. and R. Venkatesh Babu. Single-step adversarial training with dropout scheduling. In *CVPR*, 2020.

[4] Vivek B.S., Arya Baburaj, and R. Venkatesh Babu. Regularizer to mitigate gradient masking effect during single-step adversarial training. In *CVPRW*, 2019.

[5] Vivek B.S., Konda Reddy Mopuri, and R. Venkatesh Babu. Gray-box adversarial training. In *ECCV*, 2018.

[6] 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.

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

[8] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. Certified adversarial robustness via randomized smoothing. In *ICML*, 2019.

[9] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *CVPR*, 2018.

[10] Logan Engstrom, Brandon Tran, Dimitris Tsipras, Ludwig Schmidt, and Aleksander Madry. Exploring the landscape of spatial robustness. In *ICML*, 2019.

[11] Aditya Ganeshan, Vivek B.S., and R. Venkatesh Babu. FDA: Feature Disruptive Attack. In *ICCV*, 2019.

[12] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *JMLR*, 17:2096–2030, Jan. 2016.

[13] Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and Harnessing Adversarial Examples. In *ICLR*, 2015.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

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

[16] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial Machine Learning at Scale. In *ICLR*, 2017.

[17] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In *NeurIPS*, 2018.

[18] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Tsipras Dimitris, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

[19] Jan H. Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On Detecting Adversarial Perturbations. In *ICLR*, 2017.

[20] Konda Reddy Mopuri, Aditya Ganeshan, and R. Venkatesh Babu. Generalizable Data-Free Objective for Crafting Universal Adversarial Perturbations. *TPAMI*, 41(10):2452–2465, Oct. 2019.

[21] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. Practical black-box attacks against machine learning. In *ACM ASIACCS*, 2017.

[22] 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.

[23] Chongli Qin, James Martens, Sven Gowal, Dilip Krishnan, Krishnamurthy Dvijotham, Alhussein Fawzi, Soham De, Robert Stanforth, and Pushmeet Kohli. Adversarial robustness through local linearization. In *NeurIPS*, 2019.

[24] Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *NeurIPS*. 2019.

[25] Chuanbiao Song, Kun He, Liwei Wang, and John E Hopcroft. Improving the generalization of adversarial training with domain adaptation. *ICLR*, 2019.

[26] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014.

[27] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Dan Boneh, and Patrick McDaniel. Ensemble Adversarial Training: Attacks and Defenses. In *ICLR*, 2018.

[28] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble adversarial training: Attacks and defenses. In *ICLR*, 2018.

[29] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. Robustness may be at odds with accuracy. In *ICLR*, 2019.

[30] Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Ruslan Salakhutdinov, and Kamalika Chaudhuri. Adversarial robustness through local lipschitzness. *arXiv:2003.02460*, 2020.

[31] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019.