IMPROVING LOCAL EFFECTIVENESS FOR GLOBAL ROBUST TRAINING

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite its increasing popularity, deep neural networks are easily fooled. To alleviate this deficiency, researchers are actively developing new training strategies, which encourage models that are robust to small input perturbations. Several successful robust training methods have been proposed. However, many of them rely on strong adversaries, which can be prohibitively expensive to generate when the input dimension is high and the model structure is complicated. We adopt a new perspective on robustness and propose a novel training algorithm that allows a more effective use of adversaries. Our method improves the model robustness at each local patch and then, by combining these patches through a global term, achieves overall robustness. We demonstrate that, by maximizing the use of adversaries, we achieve high robust accuracy with weak adversaries. Specifically, our method reaches a similar robust accuracy level to the state of the art approaches trained on strong adversaries on MNIST, CIFAR-10 and CIFAR-100. As a result, the overall training time is reduced. Furthermore, when trained with strong adversaries, our method matches with the current state of the art on MNIST and outperforms them on CIFAR-10 and CIFAR-100.

1 INTRODUCTION

With the proliferation of deep neural networks (DNN) in areas including computer vision, natural language processing and speech recognition, there has been a growing concern over their safety. For example, Szegedy et al. (2013) demonstrated that naturally trained DNNs are in fact fragile. By adding to each data a perturbation that is carefully designed but imperceptible to humans, DNNs previously reaching almost 100% accuracy performance could hardly make a correct prediction any more. This could cause serious issues in areas such as autonomous navigation or personalised medicine, where an incorrect decision can endanger life. To tackle these issues, training DNNs that are robust to small perturbations has become an active area of research in the machine learning community.

Various algorithms have been proposed (Papernot et al., 2016; Kannan et al., 2018; Zhang et al., 2019b; Qin et al., 2019; Moosavi-Dezfooli et al., 2020; Madry et al., 2018; Ding et al., 2020). Among them, adversarial training (ADV) (Madry et al., 2018) and TRADES (Zhang et al., 2019b) are two of the most frequently used training methods so far. Although developed upon different ideas, both methods require using strong adversarial attacks, generally computed through several steps of projected gradient descent. Such attacks can quickly become prohibitive when model complexity and input dimensions increase, thereby limiting their applicability. Since the cost of finding strong adversaries is mainly due to the high number of gradient steps performed, one potential approach to alleviate the problem is to use cheap but weak adversaries. Weak adversaries are obtained using fewer gradient steps, and in the extreme case with a single gradient step. Based on this idea, Wong et al. (2020) argue that by using random initialization and a larger step-size, adversarial training with weak adversaries found via one gradient step is sufficient to achieve a satisfactory level of robustness. We term this method as one-step ADV from now on. While one-step ADV does indeed exhibit robustness, there is still a noticeable gap when compared with its multi-step counterpart.

In this paper, we further bridge the gap by proposing a new robust training algorithm: Local Effectiveness maximization based Adversarial training through Patches (LEAP). In this context,

patches should be understood as subsets of the input space on which the model makes the correct predictions. Specifically, we make the following contributions:

- We introduce a new perspective of achieving robust accuracy and, based on it, we propose our novel algorithm LEAP that allows more effective use of adversaries. LEAP consists of a local component and a global component. The local component focuses on improving robustness on points around a given adversary to maximize its effectiveness. In other words, local component attempts to enlarge the patch containing the adversary. The global component combines all local patches in a regularized way to achieve the desired robustness performance.
- We show that LEAP makes a more effective use of weak adversaries by favourably comparing it against one-step ADV on all three datasets: MNIST, CIFAR-10 and CIFAR-100.
- We show that LEAP allows efficient robust training. To achieve a similar level of robust accuracy, with LEAP, it is sufficient to use weak adversaries. In our experiments, we see that even with a one-step weak adversary, LEAP manages to achieve comparable level of robust accuracy to multi-step state of the art methods on all datasets.
- Finally, we show that when strong adversaries are used, LEAP matches with the current state of the art on MNIST and outperforms them on CIFAR-10 and CIFAR-100.

2 Related Works

Robust training aims to learn a network such that it is able to give the same correct output even when the input is slightly perturbed. Existing robust training algorithms can be divided into two categories: natural image based methods and adversaries based methods. Within the first category, the common form of loss is a natural loss term plus a regularizer computed at natural images. We briefly mention some of these methods. Moosavi-Dezfooli et al. (2020) observed empirically that reducing the curvature of the loss function and the decision boundary could lead to robust models. The authors thus propose a regularizer based on the Hessian of the loss function. Closely related is the regularizer, introduced in (Jakubovitz & Giryes, 2018; Hoffman et al., 2019; Ross & Doshi-Velez, 2018), that penalizes the Frobenius norm of the Jacobian of the loss function. Jacobian regularizer can also be seen as a way of reducing the curvature of the decision boundary (Jakubovitz & Giryes, 2018). Although calculating the norm is computationally expensive, a fast estimator with empirically high accuracy has been developed in Hoffman et al. (2019).

We focus on adversary based robust training, as they generally perform better in terms of robust accuracy. Under this category, an early fundamental work is the Fast Gradient Sign Method (FGSM) by Goodfellow et al. (2015). Adversarial Training (ADV) (Madry et al., 2018) is a multi-step variant of FGSM. Rather than using one step FGSM, ADV employs multi-step projected gradient descent (PGD) with smaller steps-sizes to generate perturbed inputs. These modifications have allowed ADV to become one of the most effective robust training methods so far (Athalye et al., 2018). Another frequently used robust training method is TRADES (Zhang et al., 2019b). Motivated by its theoretical analyses of the trade-off between natural accuracy and robust accuracy in a binary classification example, TRADES encourages model robustness by adding to the natural loss a regularizer involving adversaries to push away the decision boundary. Recently, Qin et al. (2019) suggest that a robust model can be learned through promoting linearity in the vicinity of the training examples. They designed a local linearity regularizer (LLR), in which adversaries are used to maximize the penalty for non-linearity. Applying LLR also allows efficient robust training. We note that the underlying idea of LLR is complementary to LEAP.

One major drawback of adversaries based methods (Madry et al., 2018; Zhang et al., 2019b; Qin et al., 2019; Wang et al., 2020; Ding et al., 2019) is that most of them rely on strong adversaries, computed via expensive PGD. When the input dimension is high and the model structure is complicated, finding adversaries can be too expensive for these methods to work effectively. Several works have researched possible ways to speed up the process. Algorithmically, Zhang et al. (2019a) cut down the total number of full forward and backward passes for generating multi-step PGD adversaries while Zhang et al. (2020) introduce a parameter to allow early stop PGD. Shafahi et al. (2019) reduce adversary overhead by combining weight update and input perturbation update within a single back propagation and use a single step FGSM adversary. Wong et al. (2020) argue that the main reason one-step

FGSM, generally regarded as not robust, is effective in (Shafahi et al., 2019) is due to the non-zero initialization used. As a result, Wong et al. (2020) proposed that with random initialization and a larger step size, weak adversaries generated by FGSM could lead to models with a high level of robust accuracy. In this study, we adopt a similar viewpoint of accelerating robust training through the use of weak adversaries.

3 Preliminaries and Notations

We consider classification tasks. Let $\boldsymbol{x}$ be an image from the data distribution $\mathcal{X} \subset \mathbb{R}^N$ and $y_{\boldsymbol{x}}$ be its correct label from classes $\mathcal{C} = \{1, \ldots, C\}$. We denote a neural network, parameterized by $\boldsymbol{\theta}$, as $f(\boldsymbol{x}; \boldsymbol{\theta}) : \mathcal{X} \to \mathbb{R}^C$. The function f outputs logits and the predicted class label is chosen to be the element with the largest scalar value. Given a supervised dataset, the parameters $\boldsymbol{\theta}$ are obtained by minimizing a loss function ℓ over the dataset. In natural training, a common loss choice is the cross-entropy loss, denoted as ℓ^{CE}. To compute ℓ^{CE}, we need prediction probabilities $p(\boldsymbol{x}; \boldsymbol{\theta})$, which is evaluated as $p_i(\boldsymbol{x}; \boldsymbol{\theta}) = \frac{\exp(f_i(\boldsymbol{x};\boldsymbol{\theta}))}{\sum_j \exp(f_j(\boldsymbol{x};\boldsymbol{\theta}))}$ element-wise.

Given a tolerance level ϵ and a distance measure norm $\mathcal{L}_p$, we say the network f is robust to adversaries at a given point $\boldsymbol{x} \in \mathcal{X}$, if it satisfies

$$\arg\max_{i \in C} f_i(\boldsymbol{x}'; \boldsymbol{\theta}) = \arg\max_{i \in C} f_i(\boldsymbol{x}; \boldsymbol{\theta}), \qquad \forall \boldsymbol{x}' \text{ s.t } \|\boldsymbol{x}' - \boldsymbol{x}\|_p \leq \epsilon. \tag{1}$$

In this study, we focus on the setting $p = \infty$ and we use infinity-norm ball $\mathcal{B}_\epsilon(\boldsymbol{x}) = \{\boldsymbol{x}' \mid \|\boldsymbol{x}' - \boldsymbol{x}\|_\infty \leq \epsilon\}$ to represent the allowed perturbation. If f is robust on $\mathcal{B}_\epsilon$, we call $\mathcal{B}_\epsilon$ a robust ball.

Evaluating the true robustness of f on $\mathcal{X}$ is challenging. In practice, we replace $\mathcal{X}$ with a test set and evaluate the robustness of f by measuring the percentage of the set satisfying the condition in equation (1). To check whether the condition is met, various attack strategies are applied. Commonly used attacks are PGD based. To be specific, PGD performs the following gradient step during each iteration $t + 1$,

$$\boldsymbol{x}_{t+1} = \Pi_{\mathcal{B}_\epsilon(\boldsymbol{x})}(\boldsymbol{x}_t + \xi \cdot \text{sgn}(\nabla_{\boldsymbol{x}} \ell(f(\boldsymbol{x}_t; \boldsymbol{\theta})), y)), \qquad \xi < \epsilon, \tag{2}$$

and repeats the iteration several times. Here, sgn denotes the sign function. On the other hand, FGSM uses a single gradient step

$$\boldsymbol{x}' = \boldsymbol{x} + \epsilon \cdot \text{sgn}(\nabla_{\boldsymbol{x}} \ell(f(\boldsymbol{x}; \boldsymbol{\theta})), y). \tag{3}$$

Finally, to facilitate the discussion, we introduce the following notations. We first assume that the neural network f contains ReLU activation for the sake of clarity. We note that given the explicit structure of neural networks, at each $\boldsymbol{x} \in \mathcal{X}$, it is easy to find a weight-corresponding matrix $W^{\boldsymbol{x}} \in \mathbb{R}^{\mathcal{C} \times N}$ and a constant vector $\boldsymbol{b}^{\boldsymbol{x}} \in \mathbb{R}^{\mathcal{C}}$ such that $f(\boldsymbol{x}; \boldsymbol{\theta}) = W^{\boldsymbol{x}} \boldsymbol{x} + \boldsymbol{b}^{\boldsymbol{x}}$. To simplify the notation, we define

$$\breve{W}^{\boldsymbol{x}} = [\, W^{\boldsymbol{x}} \mid \boldsymbol{b}^{\boldsymbol{x}} \,] \in \mathbb{R}^{\mathcal{C} \times (N+1)}, \qquad \breve{\boldsymbol{x}}^T = [\, \boldsymbol{x}^T \mid 1 \,] \in \mathbb{R}^{1 \times (N+1)}. \tag{4}$$

As a result, at each point $\boldsymbol{x}$, we have $f(\boldsymbol{x}; \boldsymbol{\theta}) = \breve{W}^{\boldsymbol{x}} \breve{\boldsymbol{x}}$. We now introduce our novel algorithm: *Local Effectiveness maximization based Adversarial training through Patches (LEAP)*.

4 LEAP

Unlike the optimization perspective taken by ADV and the regularization viewpoint adopted by TRADES, LEAP is motivated from a geometric standpoint. We start by briefly mentioning its closely related regularization type approaches. Generally, in such approaches, we start from a natural image $\boldsymbol{x}$. We define $\mathcal{P}_{\boldsymbol{x}} \subset \mathcal{X}$ as a connected set containing $\boldsymbol{x}$ and on which, the model makes correct predictions. We refer to such a subset as a patch. By including a loss term for $\boldsymbol{x}$, regularization type losses ensure the model gives the correct prediction $y_{\boldsymbol{x}}$ at $\boldsymbol{x}$. Therefore, it is highly likely that the volume of the patch $\mathcal{P}_{\boldsymbol{x}}$ is non-zero. It follows that, we are able to find a robust ball $\mathcal{B}_\gamma(\boldsymbol{x}) \subseteq \mathcal{P}_{\boldsymbol{x}}$, where $\gamma \geq 0$ is the maximum radius achievable. Previous use of adversary regularization term encourages the radius γ of the ball $\mathcal{B}_\gamma(\boldsymbol{x})$, centered at $\boldsymbol{x}$, to increase.

In LEAP, we take a different direction. Instead of focusing on one global ball centered at $\boldsymbol{x}$, we take care of local balls centered at various points $\boldsymbol{x}' \in \mathcal{B}_\epsilon(\boldsymbol{x})$ and hopefully, by combining them in a regularized way, we achieve the desired robustness $\gamma \geq \epsilon$.

To be more specific, LEAP consists of a local component and a global component. Both components update the parameters of the network model f. For the local component, at each given local point $\boldsymbol{x}' \in \mathcal{B}_\epsilon(\boldsymbol{x})$, the goal is to increase the volume of each patch $\mathcal{P}_{\boldsymbol{x}'}$ so a robust ball $\mathcal{B}_{\gamma'}(\boldsymbol{x}')$ with large radius can be attained. We maximize the use of adversaries by treating them as local center points $\boldsymbol{x}'$ and by enlarging their associated local robust balls $\mathcal{B}_{\gamma'}(\boldsymbol{x}')$. In terms of the global component, we combine patches together in a controlled way to further improve robustness. We denote the set on which the model makes a correct prediction as $S_{\boldsymbol{\theta}}(\boldsymbol{x}) := \{\hat{\boldsymbol{x}} \mid \arg\max f(\hat{\boldsymbol{x}}; \boldsymbol{\theta}) = \arg\max f(\boldsymbol{x}; \boldsymbol{\theta}) = y_{\boldsymbol{x}}\}$. The set $S_{\boldsymbol{\theta}}(\boldsymbol{x})$ depends on the parameter values of the network. Let $\boldsymbol{\theta}^1$ be the parameter values after the addition of the local term but before the global term and $\boldsymbol{\theta}^2$ be the parameter values after the addition of both local and global terms. We define the sets $S_{\boldsymbol{\theta}^1}(\boldsymbol{x})$ and $S_{\boldsymbol{\theta}^2}(\boldsymbol{x})$ accordingly. Given local patches, we point out that we have $\big((\cup_{\boldsymbol{x}'} \mathcal{P}_{\boldsymbol{x}'}) \cap \mathcal{B}_\epsilon(\boldsymbol{x})\big) \subseteq \big(S_{\boldsymbol{\theta}^1}(\boldsymbol{x}) \cap \mathcal{B}_\epsilon(\boldsymbol{x})\big)$. There are two possible cases for $S_{\boldsymbol{\theta}^1}(\boldsymbol{x})$. In the easy case, we have $\big(S_{\boldsymbol{\theta}^1}(\boldsymbol{x}) \cap \mathcal{B}_\epsilon(\boldsymbol{x})\big) = \mathcal{B}_\epsilon(\boldsymbol{x})$, so no guidance is required over patch combination. In the difficult case, we have $\big(S_{\boldsymbol{\theta}^1}(\boldsymbol{x}) \cap \mathcal{B}_\epsilon(\boldsymbol{x})\big) \subset \mathcal{B}_\epsilon(\boldsymbol{x})$. The hope is that by guiding the merging of patches, we can have $S_{\boldsymbol{\theta}^1}(\boldsymbol{x})$ to be a strict subset of $S_{\boldsymbol{\theta}^2}(\boldsymbol{x})$. We meet the robustness requirement if $\mathcal{B}_\epsilon(\boldsymbol{x}) \subseteq S_{\boldsymbol{\theta}^2}(\boldsymbol{x})$. It should be noted that we define $S_{\boldsymbol{\theta}^1}(\boldsymbol{x})$ and $S_{\boldsymbol{\theta}^2}(\boldsymbol{x})$ for the sake of explanation only. In the experiments, we use the local and global terms at the same time. We now discuss each component in detail.

4.1 Local Robustness

Assume we are given a local point $\boldsymbol{x}' \in \mathcal{B}_\epsilon(\boldsymbol{x})$ such that $\boldsymbol{x}' \neq \boldsymbol{x}$. We need to satisfy two requirements: the model predicts the required label $y_{\boldsymbol{x}}$ at $\boldsymbol{x}'$ and the radius γ' of $\mathcal{B}_{\gamma'}(\boldsymbol{x}')$ should be large.

The first requirement can be easily met by applying a standard loss term at $\boldsymbol{x}'$ with the label $y_{\boldsymbol{x}}$. For the second requirement, to push away the decision boundary, various methods including Jacobian regularizers (Jakubovitz & Giryes, 2018; Ross & Doshi-Velez, 2018), MMA (Ding et al., 2019) and MMR (Croce & Hein, 2019) have been introduced. In this work, we adopt the standard Jacobian approach due to the fact that our focus is on fast robust training and an efficient estimation algorithm for Jacobian has been developed in Hoffman et al. (2019). Since l_∞ norm is equivalent to l_2 norm, by which we mean $\|\boldsymbol{x}\|_\infty \leq \|\boldsymbol{x}\|_2 \leq \sqrt{N}\|\boldsymbol{x}\|_\infty$, we use l_2 norm for the following analyses for simplicity. The same conclusion can be made for the l_∞ norm with a change of constants.

We note that, at the local point $\boldsymbol{x}'$, the function f is a hyper-surface defined as $f(\boldsymbol{x}'; \boldsymbol{\theta}) = \breve{W}^{\boldsymbol{x}'} \breve{\boldsymbol{x}}'$. Let the correct class label be $c = y_{\boldsymbol{x}}$. For any $c' \neq c$, we have the decision boundary

$$f_c(\boldsymbol{x}'; \boldsymbol{\theta}) - f_{c'}(\boldsymbol{x}'; \boldsymbol{\theta}) = 0. \tag{5}$$

Standard computation show that the first order approximation distance $d_{c'}$ of $\boldsymbol{x}'$ to the boundary shown in equation (5) under the l_2 norm is

$$d_{c'} = \frac{|f_c(\boldsymbol{x}'; \boldsymbol{\theta}) - f_{c'}(\boldsymbol{x}'; \boldsymbol{\theta})|}{\|\nabla_{\boldsymbol{x}'} f_c(\boldsymbol{x}'; \boldsymbol{\theta}) - \nabla_{\boldsymbol{x}'} f_c(\boldsymbol{x}'; \boldsymbol{\theta})\|_2}. \tag{6}$$

Since the above equation holds true for any c', we conclude the model is robust on a l_2 norm ball centered at $\boldsymbol{x}'$ with radius $d := \min_{c' \neq c} d_{c'}$. To maximize the radius d, we borrow the following proposition from Jakubovitz & Giryes (2018), which introduced a Jacobian regularizer to the natural loss, to provide a lower bound for d.

Proposition 1. *Assume the model is making the correct prediction $c = y_{\boldsymbol{x}}$ for $\boldsymbol{x}'$ and the distance metric is measured via l_2 norm. The first order approximation of the minimum perturbation d that is required to find an adversary example is lower bounded by*

$$d \geq \frac{1}{\sqrt{2}\|J(\boldsymbol{x}')\|_F} \min_{c' \neq c} |f_c(\boldsymbol{x}'; \boldsymbol{\theta}) - f_{c'}(\boldsymbol{x}'; \boldsymbol{\theta})|, \tag{7}$$

where $J(\boldsymbol{x}') = \nabla_{\boldsymbol{x}'} f(\boldsymbol{x}'; \boldsymbol{\theta})$ is the Jacobian matrix computed at $\boldsymbol{x}'$ and $\|\cdot\|_F$ is the Frobenius norm.

For a larger distance d, we need to both increase the value of $|f_c(\boldsymbol{x}'; \boldsymbol{\theta}) - f_{c'}(\boldsymbol{x}'; \boldsymbol{\theta})|$ and decrease the Frobenius norm of $J(\boldsymbol{x}')$. The first term can be easily taken care of by using a cross-entropy loss

on $\boldsymbol{x}'$ while for the second term, we can include a cheap approximation of $\|J(\boldsymbol{x}')\|_F$, denoted by $\mathcal{A}(J(\boldsymbol{x}'))$ as a penalty term in our loss. Using the idea of Hoffman et al. (2019), $\mathcal{A}(J(\boldsymbol{x}'))$ can be estimated with one backward pass regardless the total number of classes C.

To combine the above analyses, for the local robustness component, we introduce the following loss,

$$\ell_{\text{local}}(\boldsymbol{x}') = \ell^{CE}(\boldsymbol{x}') + \alpha \cdot \mathcal{A}(J(\boldsymbol{x}')), \tag{8}$$

where α is a positive scalar. It is worth noting that our local loss shown in equation (8) is different from that of (Jakubovitz & Giryes, 2018; Hoffman et al., 2019; Ross & Doshi-Velez, 2018) as we evaluate it at an adversary $\boldsymbol{x}'$ instead of the image $\boldsymbol{x}$.

4.2 Global Combination of Patches

By increasing the radius of local balls, we are highly likely to have a larger patch associated to each local ball. We now combine these patches in a regularized way in the hope that the model is robust on $\mathcal{B}_\epsilon(\boldsymbol{x})$. To meet this goal, we first consider a simple binary classification problem.

4.2.1 Binary Classification Problem

Assume we only have two classes $\mathcal{C} = \{c_1, c_2\}$. For an arbitrary point $\boldsymbol{x} \in \mathcal{X}$, the correct class label is $y_{\boldsymbol{x}} = c_1$. Since we are interested in a network's performance in classifying labels, we compute logits difference as

$$f_1(\boldsymbol{x}; \boldsymbol{\theta}) - f_2(\boldsymbol{x}; \boldsymbol{\theta}) = \boldsymbol{d}\breve{W}^{\boldsymbol{x}}\breve{\boldsymbol{x}} \in \mathbb{R},$$

where $\boldsymbol{d} = [1, -1]$ is a row vector. For a binary classification with cross-entropy, the loss decreases monotonically with the increase of the value $\boldsymbol{d}\breve{W}^{\boldsymbol{x}}\breve{\boldsymbol{x}}$. In terms of robust training, we want to have $\boldsymbol{d}\breve{W}^{\boldsymbol{x}'}\breve{\boldsymbol{x}}' > 0$ for all $\boldsymbol{x}' \in \mathcal{B}_\epsilon(\boldsymbol{x})$. We make the following observation.

Proposition 2. *Consider a ball $\mathcal{B}_\epsilon(\boldsymbol{x})$ at an arbitrary point $\boldsymbol{x} \in \mathcal{X}$. For any $\boldsymbol{x}^* \in \mathcal{B}_\epsilon(\boldsymbol{x})$, we define $\breve{W}^{\boldsymbol{x}^*}$ and $\breve{\boldsymbol{x}}^*$ accordingly, as in equation (4). Let u be a constant such that, for all possible $\breve{W}^{\boldsymbol{x}^*}$, the following is satisfied*

$$\|\breve{W}^{\boldsymbol{x}^*}\|_F \leq u. \tag{9}$$

Assume $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$ are arbitrary chosen points in $\mathcal{B}_\epsilon(\boldsymbol{x})$. Then for any

$$\boldsymbol{x}_\eta = \eta \cdot \boldsymbol{x}^1 + (1-\eta) \cdot \boldsymbol{x}^2, \tag{10}$$

we have

$$\boldsymbol{d}\breve{W}^{\boldsymbol{x}_\eta}\breve{\boldsymbol{x}}_\eta \geq \frac{1}{2}(\boldsymbol{d}\breve{W}^{\boldsymbol{x}^1}\breve{\boldsymbol{x}}^1 + \boldsymbol{d}\breve{W}^{\boldsymbol{x}^2}\breve{\boldsymbol{x}}^2) - u \cdot L, \tag{11}$$

where $L = \sqrt{2}(2\|\breve{\boldsymbol{x}}_\eta\| + \frac{1}{2}\|\breve{\boldsymbol{x}}^1 - \breve{\boldsymbol{x}}^2\|)$ is a constant.

We now consider patch combination. At each loss computation, we have two points: the natural image $\boldsymbol{x}$ and the adversary $\boldsymbol{x}'$. Assume the model makes the correct prediction at the natural image and, after the local component loss, at $\boldsymbol{x}'$, so we have two non-empty patches $\mathcal{P}_{\boldsymbol{x}}$ and $\mathcal{P}_{\boldsymbol{x}'}$. We further assume these two patches are disjoint for the sake of simplicity. The same underlying idea should work for more general cases. The goal is that the pre-combination set $S_{\boldsymbol{\theta}^1}(\boldsymbol{x})$ can be a strict subset of the post-combination set $S_{\boldsymbol{\theta}^2}(\boldsymbol{x})$.

To do so, we observe that since $\mathcal{P}_{\boldsymbol{x}}$ and $\mathcal{P}_{\boldsymbol{x}'}$ are disjoint, there exists an adversary point $\boldsymbol{x}^* \in \mathcal{B}_\epsilon(\boldsymbol{x})$ satisfying

$$\boldsymbol{x}^* = \eta^* \cdot \boldsymbol{x} + (1-\eta^*) \cdot \boldsymbol{x}', \tag{12}$$

where $\eta^* \in (0, 1)$ is a constant depending on $\mathcal{P}_{\boldsymbol{x}}$ and $\mathcal{P}_{\boldsymbol{x}'}$. Then a direct application of Proposition 2 gives

$$\begin{aligned} \boldsymbol{d}\breve{W}^{\boldsymbol{x}^*}\breve{\boldsymbol{x}}^* &\geq \frac{1}{2}(\boldsymbol{d}\breve{W}^{\boldsymbol{x}}\breve{\boldsymbol{x}} + \boldsymbol{d}\breve{W}^{\boldsymbol{x}'}\breve{\boldsymbol{x}}') - u \cdot L \\ &= \frac{1}{2}\boldsymbol{d}(\breve{W}^{\boldsymbol{x}}\breve{\boldsymbol{x}} - \breve{W}^{\boldsymbol{x}'}\breve{\boldsymbol{x}}') + \boldsymbol{d}\breve{W}^{\boldsymbol{x}'}\breve{\boldsymbol{x}}' - u \cdot L \end{aligned} \tag{13}$$

where u and L are defined accordingly.

To make $\boldsymbol{x}^*$ no longer an adversary with $\boldsymbol{d}\breve{W}^{\boldsymbol{x}^*}\breve{\boldsymbol{x}}^* > 0$ after patch combination, we should make the lower bound on the right as high as possible. Since both $\boldsymbol{d}\breve{W}^{\boldsymbol{x}}\breve{\boldsymbol{x}}$ and $\boldsymbol{d}\breve{W}^{\boldsymbol{x}'}\breve{\boldsymbol{x}}'$ have the same max value theoretically, we can increase their sum by first increasing the value of $\boldsymbol{d}\breve{W}^{\boldsymbol{x}'}\breve{\boldsymbol{x}}'$ and then decrease the distance between $\boldsymbol{d}\breve{W}^{\boldsymbol{x}}\breve{\boldsymbol{x}}$ and $\boldsymbol{d}\breve{W}^{\boldsymbol{x}'}\breve{\boldsymbol{x}}'$. We recall that during the local robustness step, we have introduced cross-entropy loss and Jacobian penalty at $\boldsymbol{x}'$. The cross-entropy loss directly maximizes the value of $\boldsymbol{d}\breve{W}^{\boldsymbol{x}'}\breve{\boldsymbol{x}}'$ while the Jacobian penalty indirectly reduces u. Given that $\boldsymbol{d}\breve{W}^{\boldsymbol{x}'}\breve{\boldsymbol{x}}'$ and u are already optimized, a sound way to combine patches is minimise the distance between $\boldsymbol{d}\breve{W}^{\boldsymbol{x}}\breve{\boldsymbol{x}}$ and $\boldsymbol{d}\breve{W}^{\boldsymbol{x}'}\breve{\boldsymbol{x}}'$. We note that, for the distance to be zero, it is sufficient to have the relative differences $f_2(\boldsymbol{x}';\boldsymbol{\theta}) - f_1(\boldsymbol{x}';\boldsymbol{\theta})$ and $f_2(\boldsymbol{x};\boldsymbol{\theta}) - f_1(\boldsymbol{x};\boldsymbol{\theta})$ to be equal rather than requiring an equality between logits $f(\boldsymbol{x};\boldsymbol{\theta})$ and $f(\boldsymbol{x}';\boldsymbol{\theta})$. To account for this fact, we use prediction probabilities instead and try to minimize the prediction probability difference $p(\boldsymbol{x};\boldsymbol{\theta}) - p(\boldsymbol{x}';\boldsymbol{\theta})$. Kullback–Leibler (KL) distance is thus a natural penalty choice. As the result, we are likely to turn $\boldsymbol{x}^*$ into a correctly predicted point after patch combination.

4.2.2 General Cases

Motivated by the analyses on binary problems, we thus introduce the following as the global component loss

$$\ell_{\text{global}}(\boldsymbol{x}') = \beta \cdot \text{KL}(p(\boldsymbol{x}';\boldsymbol{\theta}) \| p(\boldsymbol{x};\boldsymbol{\theta})), \tag{14}$$

where β is a positive constant.

When the loss surface is considered, the global component loss encourages the loss surface to be smoothened and flattend on $\mathcal{B}_\epsilon(\boldsymbol{x})$. As suggested in (Zhang et al., 2019b; Cisse et al., 2017; Ross & Doshi-Velez, 2018; Moosavi-Dezfooli et al., 2017), these properties of the loss surface are desired and are possibly indispensable for models to be robust.

Although generally $\text{KL}(p(\boldsymbol{x}';\boldsymbol{\theta}) \| p(\boldsymbol{x};\boldsymbol{\theta})) \neq \text{KL}(p(\boldsymbol{x},\boldsymbol{\theta}) \| p(\boldsymbol{x}',\boldsymbol{\theta}))$ due to the asymmetric nature of KL distance, the same underlying nature of both terms means it is sufficient to use one direction for the penalty. In our loss, we have use the prediction distribution at $\boldsymbol{x}'$ to be the base distribution. There are two reasons for this choice: firstly, we want it to be consistent with the fact that the local component loss is computed on $\boldsymbol{x}'$; and secondly, we hope it could mitigate a possible over-fitting issue. If a model over-fits at the original image $\boldsymbol{x}$ causing an element $p_i(\boldsymbol{x};\boldsymbol{\theta})$ of $p(\boldsymbol{x};\boldsymbol{\theta})$ to be close to 0, the distance between $p_i(\boldsymbol{x};\boldsymbol{\theta})$ and $p_i(\boldsymbol{x}';\boldsymbol{\theta})$ would not be penalized effectively. In extreme cases, the difference will not be penalized at all if $p_i(\boldsymbol{x};\boldsymbol{\theta}) = 0$. Using $p(\boldsymbol{x}';\boldsymbol{\theta})$ as the base might alleviate this issue, as $\boldsymbol{x}'$ keeps changing during each epoch. Under the same reasoning, we point out TRADES relies on finding strong adversaries. Since it consists of a natural loss and a KL regularizer with the natural image as the base distribution, when adversaries are weak, the influence of the natural image will prevail. This is consistent with what we see in the experiments section, TRADES with one step FGSM is not as effective as other methods on challenging datasets.

4.3 Final Loss

Integrating the above analyses, we propose the following as the final loss for LEAP. Given $\alpha > 0$ and $\beta > 0$, the loss is formulated as

$$\ell_{\text{LEAP}} = \frac{1}{|\mathcal{X}|} \sum_{\boldsymbol{x} \in \mathcal{X}} \underbrace{\ell^{\text{CE}}(\boldsymbol{x}') + \alpha \cdot \mathcal{A}(J(\boldsymbol{x}'))}_{\text{local}} + \underbrace{\beta \cdot \text{KL}(p(\boldsymbol{x}';\boldsymbol{\theta}) \| p(\boldsymbol{x};\boldsymbol{\theta}))}_{\text{global}}, \quad \boldsymbol{x}' \in \mathcal{B}_\epsilon(\boldsymbol{x}). \tag{15}$$

To ensure the model make correct predictions at natural images $\boldsymbol{x} \in \mathcal{X}$ and for effective patch combination, we adopt an adaptive value scheme for the pre-determined perturbation epsilon. Specifically, we start by setting $\epsilon = 0$ and gradually increases its value to the required number over epochs during the initial stage of training. Assume the final value of ϵ is v, to be consistent with the magnitude of ϵ, we replace α and β as $\frac{\epsilon}{v} \cdot \alpha$ and $\frac{\epsilon}{v} \cdot \beta$ accordingly.

5 Experiments

We evaluate the performance of LEAP on three datasets: MNIST, CIFAR-10 and CIFAR-100. We consider the cases of training robustness models with weak adversaries generated by FGSM and with strong adversaries by multi-step PGD. To be consistent with the general experimental setting, we use the perturbation value $\epsilon = 0.3$ for MNIST and $\epsilon = 8/255$ for CIFAR-10 and CIFAR-100. MNIST is trained on a 4-layer CNN, which consists of 2 convolutional layers followed by 2 fully connected

layers. CIFAR-10 and CIFAR-100 are trained on Wide-ResNet-28-8 (Zagoruyko & Komodakis, 2016). To encourage efficient robust training, instead of a fixed learning rate scheduler that decreases the learning rate at pre-specified epoch numbers, we randomly choose a subset of the data to be the validation set and then use its robust accuracy to guide the learning rate adjustment and to terminate the training process. For MNIST, 20-step PGD is applied for computing robust accuracy on the validation set while for CIFAR-10 and CIFAR-100, 10-step PGD is used. In addition, we gradually increase epsilon value from 0 to 0.3 or $8/255$ for the first 15 epochs. We use the SGD optmizer for the training. More training related details can be found in the appendix.

Methods We compare against ADV and TRADES. We term methods that use weak FGSM adversaries as **one-step methods**. Following the advice from Wong et al. (2020), we apply FGSM, combined with random initialization and a step size of 1.25ϵ to generate adversarial examples. One-step ADV is an implementation of (Wong et al., 2020). To be consistent with their multi-step variants, we use cross-entropy loss for computing the gradient in FGSM for ADV and LEAP while employ KL distance for TRADES. **Multi-step methods** are trained on strong adversaries generated by PGD: 20 steps for the MNIST and 10 steps for CIFAR-10 and CIFAR-100. We also include the Jacobian penalty loss (Jakubovitz & Giryes, 2018; Hoffman et al., 2019; Ross & Doshi-Velez, 2018) due to its close relation to our local loss component. We refer to it as zero-step JAC to emphasize the fact it does not require adversaries. For each method, a range of parameters are tested and those that give both high clean accuracy and high validation robust accuracy are chosen. Results for all tested parameter choices are provided in the appendix. Due to the limited space, the ablation study on the effect of each component of LEAP: the local part (LEAP-l, by setting $\beta = 0$) and the global part (LEAP-g, by setting $\alpha = 0$) is left to the appendix and the same is for a detailed comparison between TRADES and LEAP-g.

Robustness Evaluation Extensive robustness evaluations are conducted in order to avoid a false sense of security. Black box attack generates adversaries on surrogate models with 1000 PGD steps. In terms of white box attacks, apart from an initial robustness evaluation with PGD-20, we further introduce three strong white attacks: PGD-1000, Untargeted (Un-T) attack (Qin et al., 2019) and Multi-targeted (Multi-T) attack (Qin et al., 2019). Untargeted attack uses the loss $f_u(\boldsymbol{x};\boldsymbol{\theta}) - f_c(\boldsymbol{x};\boldsymbol{\theta})$, where c is the correct class and $u = \arg\max_{i \neq c} f_i(\boldsymbol{x})$. We allow u to change during each gradient step. Regarding Multi-targeted attack, we perform attack with the loss $f_i(\boldsymbol{x};\boldsymbol{\theta}) - f_c(\boldsymbol{x};\boldsymbol{\theta})$ for all $i \in \mathcal{C}$ such that $i \neq c$. On the big Wide-ResNet model, we run 10 restarts with 100 steps for Untargeted attack and 5 restarts with 20 steps for Multi-target attack. We reduce the number of restarts and steps for the Multi-target attack to account for the fact that each incorrect class needs to be tested. Due to the large number of classes, Multi-target is not performed on CIFAR-100. On the other hand, we perform 20 restarts with 50 steps for both attacks on the small CNN model. For each method, attack results are reported for a model at the epoch that gives the best validation robust accuracy during the training.

We summarize all results in Table 1. We mention that with the validation set robust accuracy guided learning rate scheduler, all models achieved their best performance around 30 epochs, except 12 epochs for zero-step JAC on MNIST. In terms of robustness performance, slight performance improvements for all methods and narrower performance gaps will be observed if all models are trained for 70 epochs with a fixed learning rate scheduler on MNIST. On the other hand, on CIFAR-10 and CIFAR-100, we found all methods perform better with the validation robust accuracy guided learning rate scheduler. Since we are interested in efficient robust learning, we use the guided learning rate scheduler for all experiments.

Evaluations on CIFAR-10 We study the challenging CIFAR-10 first. For zero-step Jac, we observe that to achieve a roughly 30% robust accuracy, nominal accuracy is dropped to below 65%. Directly including a Jacobian penalty at natural images could thus lead to over-regularization issues. This may also be the reason why the method is used as a post-processing technique in (Jakubovitz & Giryes, 2018). On the other hand, by using local adversaries instead of natural images, LEAP-l manages to obtain robust accuracy without sacrificing the nominal accuracy too much, which is demonstrated in the Appendix. When TRADES is considered, we find that its effectiveness relies on strong adversaries: robust accuracy increases sharply from one-step case to multi-step case. Since TRADES computes loss at natural images and use them as the base distribution in the KL penalty, natural images will prevail when weak adversaries are used. In addition, we consistently find for one-step TRADES that, after a certain number of epochs, the percentage of the model making

Table 1: Robustness performance for MNIST on a small CNN and for CIFAR-10 and CIFAR-100 on Wide-Resnet 28-8. The higher the better.

	Model	Clean accuracy	Black box attack	White box attack				Time
				PGD-20 attack	PGD-1000 attack	Un-T attack	Multi-T attack	per batch
MNIST	zero-step JAC	98.34%	89.84%	58.09%	2.48%	28.83%	28.72%	0.012s
	one-step ADV	**99.49%**	96.26 %	96.29%	83.25%	91.02%	90.62%	0.009s
	one-step TRADES	99.40%	96.21%	96.28%	83.05%	90.66%	90.02%	0.016s
	one-step LEAP	99.47%	**96.36%**	**96.84%**	**89.56%**	**92.44%**	**92.04%**	0.018s
	multi-step ADV	99.48%	96.40%	97.16%	**92.32%**	93.01%	**92.87%**	0.077s
	multi-step TRADES	**99.49%**	96.21%	97.01%	90.51%	92.49%	92.21%	0.097s
	multi-step LEAP	99.44%	**96.45%**	**97.19%**	92.20%	**93.03%**	92.74%	0.086s
CIFAR-10	zero-step JAC	63.24%	60.45%	34.40%	34.34%	31.13%	30.32%	0.47s
	one-step ADV	86.24%	82.53%	45.73%	44.98%	45.14%	42.97%	0.25s
	one-step TRADES	**86.84%**	**82.63%**	40.03%	39.41%	39.31%	38.14%	0.55s
	one-step LEAP	84.52%	81.06%	**49.01%**	**48.59%**	**48.54%**	**46.55%**	0.73s
	multi-step ADV	**84.77%**	82.53%	50.60%	50.22%	49.74%	47.96%	1.38s
	multi-step TRADES	84.01%	**82.63%**	52.70%	52.45%	50.29%	49.66%	2.02s
	multi-step LEAP	82.94%	81.06%	**53.45%**	**53.10%**	**51.51%**	**50.16%**	1.88s
CIFAR-100	zero-step JAC	47.96%	44.67%	15.39%	15.18%	14.35%	-	0.47s
	one-step ADV	48.88%	45.59%	19.89%	19.72%	19.04%	-	0.25s
	one-step TRADES	**62.58%**	**57.64%**	16.93%	16.21%	14.83%	-	0.53s
	one-step LEAP	60.98%	57.04%	**26.30%**	**25.94%**	**26.28%**	-	0.73s
	multi-step ADV	**60.72%**	**58.15%**	27.71%	27.43%	26.83%	-	1.37s
	multi-step TRADES	59.76%	57.78%	26.27%	26.10%	23.86%	-	2.00s
	multi-step LEAP	59.62%	57.83%	**29.12%**	**28.89%**	**27.77%**	-	1.87s

incorrect decisions on weak adversaries falls sharply, leading to a sudden drop of more than 10% on validation robust accuracy. The same phenomenon is observed on one-step ADV and one-step LEAP but less frequently. As a result, early termination is essential for one-step methods.

Overall, one-step LEAP outperforms other one-step methods in all strong white box attacks. Even when compared with methods trained with 10 steps PGD, the performance gap between one-step LEAP and multi-step ADV under the strongest Multi-T attack is below 1.5%. Although adding local and global penalties result in extra computational time for one-step LEAP (0.73s) than one-step ADV (0.25s), the fact that LEAP achieves comparable high robust accuracy to that of multi-step ADV (1.38s) with weak adversaries still allows a roughly 50% speed-up in training. For a fair comparison, we also evaluate an ADV model with 5 PGD steps. The 5-step ADV models requires slightly more time (0.75s per batch) but is less robust than one-step LEAP by reaching 44.93% accuracy under the Multi-T attack. When trained on strong adversaries, multi-step LEAP wins over the other two multi-step methods in all strong white box attacks.

Evaluations on CIFAR-100 and MNIST Similar performances are observed on the CIFAR-100. The key fact to notice here is that the per batch training cost required for CIFAR-100 is similar to that of CIFAR-10. Since we apply an efficient approximation to evaluate $\|J(\boldsymbol{x})\|_F$, the increase of total class number does not result in a rise of total computational cost. Again, we evaluate ADV models with 5 PGD steps, which requires 0.75s per batch and reaches 25.21% robust accuracy under Un-T attack. On MNIST, one-step LEAP performs the best as well. Since MNIST is a simple dataset, all multi-step methods obtain high robust accuracy on the small CNN model. For the same reason, the unstable sudden drop of validation robust accuracy is not observed on any one-step method.

6 Discussion

We have developed a novel algorithm LEAP to focus on effective use of adversaries. Leap allows efficient robust training, which can be used as an initialisation technique for other complicated tasks. For instance, model trained with LEAP can be employed as the starting point for layer-wise adversarial training in improving certified robustness.

REFERENCES

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

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 854–863. JMLR. org, 2017.

F. Croce and M. Hein. Provable robustness against all adversarial l_p-perturbations for $p \geq 1$. *International Conference on Learning Representations*, 2019.

Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. Mma training: Direct input space margin maximization through adversarial training. In *International Conference on Learning Representations*, 2019.

G.W. Ding, Y. Sharma, K.Y.C. Lui, and R. Huang. Max-margin adversarial (mma) training: Direct input space margin maximization through adversarial training. *International Conference on Learning Representations*, 2020.

I.J. Goodfellow, J. Shlens, and C. Szegedy. Explaining and harnessing adversarial examples. *International Conference on Learning Representations*, 2015.

J. Hoffman, D.A. Roberts, and S Yaida. Robust learning with jacobian regularization. *https://arxiv.org/abs/1908.02729*, 2019.

Daniel Jakubovitz and Raja Giryes. Improving dnn robustness to adversarial attacks using jacobian regularization. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 514–529, 2018.

H. Kannan, A. Kurakin, and I. Goodfellow. Adversarial logit pairing. *arXiv preprint arXiv:1803.06373*, 2018.

A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. Towards deep learning models resistant to adversarial attacks. *International Conference on Learning Representations*, 2018.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, Pascal Frossard, and Stefano Soatto. Analysis of universal adversarial perturbations. *arXiv preprint arXiv:1705.09554*, 2017.

S.M. Moosavi-Dezfooli, A. Fawzi, J. Uesato, and P. Frossard. Robustness via curvature regularization, and vice versa. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020.

N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami. Distillation as a defense to adversarial perturbations against deep neural networks. *IEEE Symposium on Security and Privacy (SP) (pp. 582-597)*, 2016.

C. Qin, J. Martens, S. Gowal, D. Krishnan, K. Dvijotham, A. Fawzi, S. De, R. Stanforth, and P. Kohli. Adversarial robustness through local linearization. *In Advances in Neural Information Processing Systems*, 2019.

Andrew Slavin Ross and Finale Doshi-Velez. Improving the adversarial robustness and interpretability of deep neural networks by regularizing their input gradients. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

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

C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2020.

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

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Painless adversarial training using maximal principle. In *In Advances in Neural Information Processing Systems*, 2019a.

H. Zhang, Y. Yu, J. Jiao, E.P. Xing, L.E. Ghaoui, and M.I. Jordan. Theoretically principled trade-off between robustness and accuracy. *arXiv preprint arXiv:1901.08573*, 2019b.

Jingfeng Zhang, Xilie Xu, Bo Han, Gang Niu, Lizhen Cui, Masashi Sugiyama, and Mohan Kankanhalli. Attacks which do not kill training make adversarial learning stronger. In *International Conference on Machine Learning*, 2020.