SEMI-UNIVERSAL ADVERSARIAL PERTURBATIONS

A PREPRINT

Jordan Frecon
Normandie Univ, INSA Rouen
UNIROUEN, UNIHAVRE, LITIS
Saint-Etienne-du-Rouvray, France
jordan.frecon@insa-rouen.fr

Gilles Gasso
Normandie Univ, INSA Rouen
UNIROUEN, UNIHAVRE, LITIS
Saint-Etienne-du-Rouvray, France
gilles.gasso@insa-rouen.fr

Stéphane Canu
Normandie Univ, INSA Rouen
UNIROUEN, UNIHAVRE, LITIS
Saint-Etienne-du-Rouvray, France
stephane.canu@insa-rouen.fr

ABSTRACT

The present work introduces a framework for learning and selecting semi-universal adversarial perturbations. It relies on a joint estimation of multiple universal adversarial perturbations which are chosen in an unsupervised manner depending on the sample to attack. Two algorithmic solutions, with convergence guarantees under Lipschitz continuity assumptions, are proposed to handle either small scale or large scale datasets. Numerical experiments, conducted on benchmark datasets, support its unifying aspect between universal and specific attacks as the number of perturbations grows. In addition, the learned perturbations display strong patterns indicative of the existing similarities between the training instances of different classes.

1 Introduction

Embedded technologies using artificial neural networks are increasingly present in our daily lives. Their high expressive power have shown a great success to predict various complex tasks [1, 2]. However, since the pioneer work of [3] which has shown the existence of adversarial attacks, some concerns have been raised about their safety and more particularly for the safety of the user [4]. The most striking example is that of automated vehicles where malicious attacks could lead the car to take unwanted action with dramatic consequences [5, 6].

Most of adversarial attacks are quasi negligible perturbations (see, a contrario, [7]) which, added to examples correctly predicted, manage to fool the prediction of the neural network. From a fast one-shot method [8] to the first iterative procedures [9, 10, 11, 12, 13], the crafting of adversarial perturbations has lately received a lot of attention from the machine learning community. To this regard, momentum-based methods [14, 15] have shown a promising boost in the transferability of the attacks learned on one neural network to other neural networks. In addition, various contributions have investigated algorithmic concerns leading to accelerated and scale-invariant attacks [16] as well as parameter-free attacks [17]. In another line of research, attacks exploiting the decision boundary of neural networks have been designed in [18] while the structure of the images to attack has been taken into account through a principal component analysis in [19]. A key particularity of all the prior attacks is that they are *example-based* (or *specific*), meaning that they are specifically crafted to attack a single example. Henceforth, in order to attack a new example, one needs to learn the associated perturbation once again. While they are very effective, they have the major drawback of being time consuming.

On the other end of the spectrum, *example-agnostic* (or *universal*) attacks aim to find an attack which, once learned, can be applied to every example of the test set. The work of [20] have first demonstrated that there exist a single perturbation, coined universal adversarial perturbations (UAP), which, added to any test images, has a high chance of fooling the classifier. Later, a more efficient method has been devised in [21] by hinging on a projected gradient descent algorithm. A variant of [20] additionally exploiting the orientations of the perturbation vectors has been devised in [22]. In addition, inspired from the observation that UAP does not attack all classes equally, a class-based universal perturbation (CW-UAP) has recently been proposed in [23]. The reader is invited to refer to [24] for a survey of universal adversarial attacks. However, although these perturbations are universal, it is difficult to interpret precisely why they work in a case to case basis. More generally, state-of-the-art universal attacks still suffer from a lack of interpretability.

Contributions and outline. The present work intends to bridge the gap between specific and universal perturbations through semi-universal adversarial perturbations (SUAP).

Our framework is closely related to that of CW-UAP in the sense that multiple universal perturbations are learned but differs from the fact that they are not intended to attack specific classes. Instead, each sample is attacked by an universal perturbation chosen in an unsupervised manner. As a consequence, it allows to capture both the similarity between multiple examples and their distance to the decision boundary, irrespectively of their class. In addition, the number of perturbations acts as a tuning parameter controlling the amount of diversity between the attacks.

After discussing the preliminaries and reviewing the related works in Section 2, we introduce the proposed semi-universal adversarial perturbations in Section 3. Two algorithmic solutions are devised in Section 4 by hinging on either full-batch or stochastic solvers. Numerical experiments are conducted in Section 5 on multiple benchmark datasets.

Notations. We let $\|\cdot\|_p$, with $p \geq 0$, to denote the ℓ_p-norm. In addition, $\|\cdot\|$ acts as a shorthand for the ℓ_2-norm (resp. the Frobenius norm) of vectors (resp. matrices). Similarly, $\langle\cdot\rangle$ is used to denote the dot product both in the vector and matrix cases. Given some convex set $\mathcal{C}$, we let $\mathrm{Proj}_{\mathcal{C}}$ be the projection onto $\mathcal{C}$. Finally, for $x \in \mathbb{R}^P$ and $\mathcal{I} \subseteq \{1,\ldots,P\}$, $x_{\mathcal{I}}$ stands for the restriction of x to the indices in $\mathcal{I}$.

2 Preliminaries and Related Works

Without loss of generality, we consider a trained neural network $f\colon \mathbb{R}^P \to \mathbb{R}^c$ which associate every example $x \in \mathcal{X} \subseteq \mathbb{R}^P$ to its probabilities $f(x) \in \mathbb{R}^c$ to belong to any of the $c \in \mathbb{N}_+$ classes. We make the distinction between $\mathcal{X}$ and $\mathbb{R}^P$ since the input data can live inside a manifold (e.g., the space of images whose pixels' intensity lies within $[0,1]$). The predicted label of x by f is then defined as the maximizer $C_f(x) = \mathrm{argmax}_{k\in\{1,\ldots,c\}}\, f(x)_k$. The goal of adversarial learning is to find, for every example $x \in \mathcal{X}$, a sample $a \in \mathcal{X}$ close to x such that $C_f(a) \neq C_f(x)$. Since a is close to x, one would have expected both examples to be equally classified by f. Hence, a is called adversarial and said to fool the classifier C_f.

In the next sections, we recap the two current paradigms for crafting adversarial attacks.

2.1 Specific Attacks

In the remaining of the paper, we consider the peculiar case of adversarial images built by adding a perturbation. Namely, for each $x \in \mathcal{X}$, the aim is to find a perturbation $\varepsilon(x)$ so that $a = x + \varepsilon(x)$ is an adversarial example [10, 8, 12]. Note that, in the space of images, many works have suggested that ℓ_p-norms are reasonable approximations of human perceptual distance and thus can be chosen as a measure of closeness between x and a (see [25] and references therein). More formally, in order to ensure closeness, we will look for a perturbation $\varepsilon(x) \in \mathcal{B}_p(\delta) = \{e \in \mathbb{R}^P \mid \|e\|_p \leq \delta\}$ for some small budget $\delta > 0$. Hereafter, we drop the dependency on x and simply denote ε. We recall below two of the most popular specific attacks.

DeepFool [10]. The DeepFool attack is the smallest ℓ_p perturbation managing to fool the classifier C_f. More formally, it is the solution of the following optimization problem

$$\underset{\varepsilon\in\mathbb{R}^P}{\mathrm{minimize}}\ \|\varepsilon\|_p \quad \text{s.t.} \quad C_f(x+\varepsilon) \neq C_f(x). \tag{1}$$

PGD [13]. This attack hinges on some similarity function H (typically the cross-entropy) measuring the (dis-)similarity between the output $f(x+\varepsilon)$ of the neural network and the original label y. Then, the adversarial perturbation is defined as the one inside the ℓ_p-ball which maximizes the dissimilarity between $x+\varepsilon$ and y, i.e.,

$$\underset{\varepsilon\in\mathbb{R}^P}{\mathrm{maximize}}\ H(f(x+\varepsilon), y) \quad \text{s.t.} \quad \|\varepsilon\|_p \leq \delta. \tag{2}$$

In practice, the opposite of the objective in (2) is minimized by resorting to a projected gradient method, hence the name of the attack.

2.2 Universal Attacks

Departing from specific attacks, universal attacks look for a perturbation ε such that, for every $x \sim \mu$ sampled from some data distribution μ, $a = x + \varepsilon$ is an adversarial example with high probability. In practice, such perturbation is crafted to fool the classifier on almost all the examples from a given dataset $\{x_i, y_i\}_{i=1}^n$ made of $n \in \mathbb{N}_+$ data pairs. In the following we recall the most popular universal attacks [24].

UAP [20]. The first universal perturbation was crafted by aggregating individual DeepFool perturbations [10].

```
$\varepsilon = 0_P$
While the target fooling rate is not achieved
    For each $x_i$ such that $C_f(x_i) = C_f(x_i+\varepsilon)$        (3)
        $\Delta\varepsilon_i = \mathrm{DeepFool}(x_i)$
        $\varepsilon = \mathrm{Proj}_{\mathcal{B}_p(\delta)}(\varepsilon + \Delta\varepsilon_i)$
```

where DeepFool represents the attack of (1).

Fast-UAP [22]. This attack follows the UAP procedure but, instead of aggregating all the perturbations $\Delta\varepsilon_i$, only adds the perturbation with the closest orientation to the current iterate ε.

UAP-PGD [21]. The UAP-PGD attack elaborates upon PGD by framing of the universal perturbation as the solution of the following optimization problem

$$\underset{\varepsilon\in\mathbb{R}^P}{\mathrm{maximize}}\ \frac{1}{n}\sum_{i=1}^{n} H(f(x_i+\varepsilon), y_i) \quad \text{s.t.} \quad \|\varepsilon\|_p \leq \delta. \tag{4}$$

CW-UAP [23]. Recently, UAP-PGD has been extended to class-wise UAP where a universal perturbation is build for each label. Let n_k be the number of training samples of the k-th class, then CW-UAP aims at solving

$$\underset{\{\varepsilon_k \in \mathbb{R}^P\}_{k=1}^c}{\text{maximize}} \sum_{k=1}^{c} \frac{1}{n_k} \sum_{\substack{i=1 \\ y_i = k}}^{n} H(f(x_i + \varepsilon), y_i)$$
$$\text{s.t.} \quad (\forall k \in \{1, \ldots, c\}, \ \|\varepsilon_k\|_p \leq \delta. \quad (5)$$

The corresponding numerical solution amounts in learning multiple independent UAP-PGD perturbations, i.e., one for each label value $\{1, \ldots, c\}$.

3 Semi-universal Adversarial Attack

In this section, we introduce the proposed framework for learning and selecting semi-universal adversarial perturbations.

The originality of the proposed work is to jointly learn $L \in \mathbb{N}_+$ universal adversarial perturbations $\{\varepsilon_1, \ldots, \varepsilon_L\}$ where each $\varepsilon_l \in \mathbb{R}^P$. To do so, we propose to address the following optimization problem.

Problem 1 (Learning semi-universal ℓ_p-perturbations)**.** *Let $L \in \mathbb{N}_+$ be the number of semi-universal perturbations to learn. Given some dataset $\{x_i, y_i\}_{i=1}^n$ made of $n \in \mathbb{N}_+$ data pairs $(x_i, y_i) \in \mathcal{X} \times \mathbb{R}^c$, find $\varepsilon = [\varepsilon_1, \ldots, \varepsilon_L]$ solving*

$$\underset{[\varepsilon_l \in \mathbb{R}^P]_{l=1}^L}{\text{maximize}} \left\{ \mathcal{L}(\varepsilon) = \frac{1}{n} \sum_{i=1}^{n} \max_{l \in \{1,\ldots,L\}} H(f(x_i + \varepsilon_l), y_i) \right\},$$
$$\textit{s.t.} \quad (\forall l \in \{1, \ldots, L\}), \ \|\varepsilon_l\|_p \leq \delta. \quad (6)$$

For each pair (x_i, y_i), the objective in (6) will only pick one of the L perturbations $\{\varepsilon_1, \ldots, \varepsilon_L\}$ which maximizes the dissimilarity between $f(x_i + \varepsilon_l)$ and y_i the most. In addition, the L constraints permits to enforce that each perturbation lives inside a ℓ_p-ball of radius $\delta > 0$. Note that, the formulation of (6) boils down to (4) for a single perturbation (i.e., $L = 1$).

It is worth stressing that Problem 1 is a difficult nonconcave maximization problem due to the presence of the neural network f. Hence finding the global solution of Problem 1 is out of reach. Instead, we will propose in Section 4 two algorithmic solutions for finding an approximate solution in an efficient manner.

Once the universal adversarial perturbations have been learned, one can attack a new example by picking one perturbation amongst the L perturbations as follows.

Problem 2 (Attacking unseen example)**.** *Given some data pair $(x, y) \in \mathcal{X} \times \mathbb{R}^c$, the corresponding adversarial attack reads*

$$a = \mathcal{P}_{\mathcal{X}}\left(x + \varepsilon_{\hat{l}}\right) \quad \textit{where} \quad \hat{l} = \underset{l \in \{1,\ldots,L\}}{\text{argmax}} H(f(x + \varepsilon_l), y). \quad (7)$$

Algorithm 1 SUAP-PG

```
Require: Parameter $\rho \in ]0, 1[$
  Initialize $\varepsilon^{(0)} = [\varepsilon_l^{(0)}]_{l=1}^L$
  for $k = 0$ to $K - 1$ do
      Provide a rough estimate of $\gamma_k > 0$
      Projected gradient step
      $\varepsilon^{(k+1/2)} = \text{Proj}_{\mathcal{B}_p(\delta)^L}(\varepsilon^{(k)} + \gamma_k \nabla \mathcal{L}(\varepsilon^{(k)}))$
      Choice of relaxation parameter
      $i_k = 0$
      repeat
          $\varepsilon^{(k+1)} = (1 - \rho^{i_k})\varepsilon^{(k)} + \rho^{i_k}\varepsilon^{(k+1/2)}$
          $i_k = i_k + 1$
      until $\mathcal{L}(\varepsilon^{(k+1)}) \geq \mathcal{L}(\varepsilon^{(k)}) + \rho^{i_k - 1} h^{(k)}(\varepsilon^{(k+1/2)})$
  end for
  return Semi-universal adversarial perturbations $\varepsilon^{(K)}$
```

Solving Problem 2 requires to perform L independent forward passes through the neural network f in order to evaluate which perturbation $\{\varepsilon_1, \ldots, \varepsilon_L\}$ maximizes the dissimilarity. Note that, since they are independent, they can be performed in parallel in order to accelerate the computation. We provide below a complexity comparison between specific, semi-universal and universal attacks.

Remark 1. *Given a neural network f whose forward complexity is of $O(d)$ for a single input sample, then the complexity to compute $\nabla H(f(x), y)$ is of order $O(2d)$, since the backward pass is also of order $O(d)$. Then, it follows that*

(specific) for K iterations, cost $\sim$ $O(2Kd)$,

(semi-universal) for L perturbations, cost $\sim$ $O(Ld)$,

(universal) cost $\sim$ $O(1)$.

Hence, from the standpoint of computational complexity, universal attacks are the most efficient. To a lesser extent, one-shot specific attacks (i.e., $K = 1$, such as in FGSM [8]) and the proposed semi-universal attack achieve comparable complexity for small L.

4 Algorithmic Solutions

In this section, we present two algorithmic solutions for solving Problem 1.

4.1 Full-Batch Solver

In order to maximize $\mathcal{L}$ in Problem 1, we embrace a projected gradient ascent algorithm augmented with an Armijo-like line-search strategy in order to ensure some sufficient increase at each iteration. Its principle is inspired from the minorize-maximization algorithm where, at each step, a lower-bound of the objective function $\mathcal{L}$ is maximized.

Let $\varepsilon = [\varepsilon_1, \ldots, \varepsilon_L]$ be the concatenation of the L perturbations and $\{\gamma_k\}_{k \in \mathbb{N}_+}$ be some sequence of step-sizes.

Then, at each iteration $k \in \mathbb{N}_+$ the algorithm looks for $\varepsilon^{(k+1/2)} \in \mathcal{B}_p(\delta)$ which maximizes the linearized surrogate of the form

$$h^{(k)}(\varepsilon) = \mathcal{L}(\varepsilon^{(k)}) + \nabla\mathcal{L}(\varepsilon^{(k)})^\top(\varepsilon - \varepsilon^{(k)}) - \frac{1}{2\gamma_k}\|\varepsilon - \varepsilon^{(k)}\|^2. \tag{8}$$

Such choice is motivated by the fact that, for concave and μ-smooth functions $\mathcal{L}$, then for every $\gamma_k \leq 1/\mu$, $\mathcal{L}(\varepsilon) \geq h^{(k)}(\varepsilon)$. Henceforth,

$$\varepsilon^{(k+1/2)} = \operatorname*{argmax}_{\varepsilon \in \mathcal{B}_p(\delta)} h^{(k)}(\varepsilon), \tag{9}$$

$$= \mathcal{P}_{\mathcal{B}_p(\delta)}\left(\varepsilon^{(k)} + \gamma_k \nabla\mathcal{L}(\varepsilon^{(k)})\right), \tag{10}$$

which recast into one projected gradient ascent step. Note that the differentiability of $\mathcal{L}$ depends on the choice of the loss function H and on the neural network f to attack. For instance, for ReLu-based neural network, it is likely that $\nabla\mathcal{L}(\varepsilon^{(k)})$ is not well defined. In that case and whenever $\mathcal{L}$ is not differentiable, we resort to a sub-gradient instead. We additionally consider a relaxation step of the form

$$\varepsilon^{(k+1)} = (1 - \alpha_k)\varepsilon^{(k)} + \alpha_k \varepsilon^{(k+1/2)}, \tag{11}$$

where the relaxation parameter $\alpha_k \in (0, 1]$ is appropriately chosen by an Armijo-like line-search strategy to ensure some sufficient increase in $\mathcal{L}$ [26]. The full algorithmic procedure is sketched in Algorithm 1. In practice, we suggest to initialize the L universal perturbations in a non-informative manner by randomly sampling each $\varepsilon_l^{(0)} \sim [-\delta, \delta]^P$ and additionally projecting onto the ball $\mathcal{B}_p(\delta)$. We recall below the convergence guarantees.

Theorem 1 (Convergence [26])**.** *Let $\{\varepsilon^{(k)}\}_{k\in\mathbb{N}}$ be the sequence of Algorithm 1 and suppose that $\nabla\mathcal{L}$ is Lipschitz continuous. Then each limit point of $\{\varepsilon^{(k)}\}_{k\in\mathbb{N}}$ is a stationary point of Problem 1 and $\{\mathcal{L}(\varepsilon^{(k)})\}_{k\in\mathbb{N}}$ converges towards the objective value at the limit point. In addition, if $\mathcal{L}$ satisfies the Kurdyka-Łojasiewicz (KŁ) property at any point, then the sequence converges to a stationary point of Problem 1.*

The existence of the Lipschitz constant plays a central role for ensuring convergence guarantees of the algorithm. Note that studying the Lipschitz continuity of neural networks and obtaining sharp Lipschitz constant is difficult (see, e.g., [27, 28] and references therein).

Remark 2. *Many functions met in neural networks (e.g., activation functions, loss) are semi-algebraic or tame, and therefore satisfy the KŁ property (see, e.g., [29, 30]). Since these "concepts" are stable under many operations, it is reasonable to assume that many deep neural network f are likely to satisfy the KŁ property and so does $\mathcal{L}$.*

While few attention is usually devoted to these concerns for crafting adversarial attacks, we will show in Section 5 the superiority of the corresponding principled algorithmic solution even though these assumptions do not always hold.

Algorithm 2 SUAP-ProxSAGA

```
Initialize $\varepsilon^{(0)} = [\varepsilon_l^{(0)}]_{l=1}^L$
Set $g_i = \nabla\ell_i(\varepsilon^{(0)})$ for every $i \in \{1, \dots, n\}$
Set $\bar{g}^{(0)} = (1/n)\sum_{i=1}^n g_i$
for $k = 0$ to $K - 1$ do
    Instant gradient computation
    Uniformly pick a batch $\mathcal{I}_k \subset \{1, \dots, n\}$ of size $b$
    $g_{\mathcal{I}_k} = \sum_{i\in\mathcal{I}_k} \nabla\ell_i(\varepsilon^{(k)})$
    Projected gradient step
    $\alpha^{(k)} = \frac{1}{b}(g_{\mathcal{I}_k} - \tilde{g}_{\mathcal{I}_k}) + \bar{g}^{(k)}$
    $\varepsilon^{(k+1)} = \mathrm{Proj}_{\mathcal{B}_p(\delta)}(\varepsilon^{(k)} + \gamma_k\alpha^{(k)})$
    Updates
    $\bar{g}^{(k+1)} = \frac{1}{n}(g_{\mathcal{I}_k} - \tilde{g}_{\mathcal{I}_k}) + \bar{g}^{(k)}$
    $\tilde{g}_{\mathcal{I}_k} = g_{\mathcal{I}_k}$
end for
return Semi-universal adversarial perturbations $\varepsilon^{(K)}$
```

4.2 Stochastic Solver

We now propose a different solver fully exploiting the finite-sum nature of the loss in Problem 1. To this regard, we begin by rewriting it by means of the sample-wise losses ℓ_i, i.e., $\mathcal{L}(\varepsilon) = \frac{1}{n}\sum_{i=1}^n \ell_i(\varepsilon)$ with $\ell_i(\varepsilon) = \max_{l\in\{1,\dots,L\}} H(f(x_i + \varepsilon_l), y_i)$.

Hereafter, we resort to a stochastic solver based on the well-known variance reduction technique [31, 32, 33]. Since the main computational load comes from the backpropagation through the neural network, we favor the proxSAGA algorithm [31] which does not require an additional loop over multiple epochs. The corresponding algorithmic solution is reported in Algorithm 2. Such solver should become particularly useful to deal with large datasets by treating one sample at a time. We recall below the convergence guarantees under the assumption of Lipschitz continuity.

Theorem 2 (Convergence [31])**.** *Suppose that $\nabla\mathcal{L}$ is Lipschitz continuous with Lipschitz constant β. Let $\{\varepsilon^{(k)}\}_{k\in\mathbb{N}}$ be the sequence of Algorithm 2 with fixed step-size $\gamma_k = \gamma \leq 1/(5\beta n)$ and batch-size $b = 1$. Then, for k uniformly sampled from $\{1, \dots, K\}$, the following holds:*

$$\mathbb{E}\left[\|G_\gamma(\varepsilon^{(k)})\|^2\right] \leq \frac{50\beta n^2}{5n - 2}\frac{\mathcal{L}(\varepsilon^\star) - \mathcal{L}(\varepsilon^{(0)})}{K}, \tag{12}$$

where $\varepsilon^\star$ is a maximizer of $\mathcal{L}$ and $G_\gamma \colon \varepsilon \mapsto \gamma^{-1}(\varepsilon - \mathcal{P}_{\mathcal{B}_p(\delta)}(\varepsilon + \gamma\nabla\mathcal{L}(\varepsilon)))$ is the gradient mapping.

Note that Theorem 2 relies on the Lipschitz constant β whose calculation is out of reach. Instead, in practice we suggest either to choose β large enough or to compute rough estimate at each iteration.

5 Numerical Experiments

Throughout the section, we conduct numerical experiments on three of the most popular benchmark classification

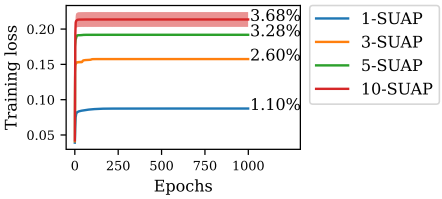

Figure 1: **Training behavior of ℓ_∞-based SUAP attacks on MNIST**. The averaged training loss is reported for different number of universal perturbations (1, 3, 5 and 10) along with the associated test fooling rate.

datasets. Hereafter, we consider the classical scenarios of ℓ_∞ and ℓ_2-based attacks with a maximum budget δ of $8/255$ and 0.5, respectively.

5.1 MNIST Experiments

We begin with the simple yet interesting MNIST dataset which happens to be useful to interpret the learned adversarial perturbations.

Data splitting and pre-processing. The 60K samples of the training set undergo random affine transformations keeping the center invariant. To this effect, we use random rotations between $[11.25, +11.25]$ degrees and a random scaling selected in $[-0.825, +0.825]$. These deformed samples are used to learn f while we randomly pick 500 original un-deformed images from the training dataset to learn the (semi-)universal attacks. The 10K images of the test set are used to evaluate the performance of the attacks. All images are flatten into 784 dimensional rescaled vectors so that the pixel intensity lies within $[0, 1]$.

Model to attack. We consider a differentiable model satisfying the KŁ property assumed in Theorem 1 (see Remark 2). To this effect, we resort to the simple multi-layer perceptron from [34] which manages to achieve under 1% test accuracy. It is made of scaled hyperbolic tangent activation functions as well of an input layer, 8 hidden layers and an output linear layer of sizes 784×1000, 1000×1000 and 1000×10, respectively. The network is trained using a stochastic gradient descent with batch size 100 with a learning rate linearly decreasing from 10^{-3} to 10^{-6} over 10^3 epochs.

Training behavior. We analyze the training behavior of L-SUAP attacks with $L \in \{1, 3, 5, 10\}$ universal perturbations learned with the Algorithm 1. The experiment is repeated over 5 independent seeds and the averaged training loss is reported in Fig. 1. Independently of L, it shows the well-behaved increasing behavior of the loss along the number of epochs. In addition, it supports the fact that having more universal perturbations does permit to achieve higher dissimilarity hence higher loss values. This is also seconded by the mean test fooling rate reported for each of the L-SUAP attacks since we observe an increased fooling rate as L grows. On a side note, on this simple dataset, it is difficult to fool the studied network f, hence justifying the small fooling rates depicted in Fig. 1.

Illustration and role of the universal perturbations. For illustration purposes, we report in Fig. 2 the learned universal perturbations of 5-SUAP decomposed into positive parts (top plots) and negative parts (bottom plots). Interestingly, they all exhibit strong patterns. In particular, we observe that ε_1 and ε_5 are very similar up to the sign difference (see ε_1^+ and ε_5^-). Indeed, since the proposed framework does not handle the tuning of the sign of the perturbation, it might happen that two perturbations are the opposite of each others. It is also worth noticing that the universal perturbations learned are consistent throughout multiple splits and random initializations.

We additionally report in Fig. 3 the fooling matrices associated to each of the perturbations $\{\varepsilon_l\}_{l=1}^5$. The latter show the correspondence between the predicted target $C_f(x)$ of some image x and the label of the associated adversarial attack, i.e.., $C_f(x + \varepsilon_{\hat{l}})$ (see Problem 2). The fooling matrices highlight that each universal perturbation plays a different role. For instance, ε_1 mostly permits to attack images of digits "3" and "9" into being misclassified as "5" and "4", respectively. Instead, ε_3 is principally used to attack images of "5" into "3". Coincidentally, one can distinguish the tilted number three in ε_3 (see Fig. 2). As opposed to CW-UAP, the proposed SUAP attack allows to automatically capture the similarity between multiple digits such as "3" and "9".

5.2 CIFAR-10 Experiments

We now turn to CIFAR-10 dataset [35] and compare the performance of the proposed attack with the baselines.

Setting. We consider the pre-trained ResNet18 model from [36] augmented with an input normalizing layer of channel-wise means $(0.4914, 0.4822, 0.4465)$ and channel-wise standard deviations $(0.2471, 0.2435, 0.2616)$. In addition, if not mentioned otherwise, we split the test set into 20K images for learning (semi-)universal perturbations and 80K independent images used for evaluating the attacks.

Baselines. The proposed SUAP attack is brought into comparison against the following universal attacks[1].

We compare with the UAP-PGD from [21] which is closely related to our proposed SUAP with a single perturbation. However it differs from two aspects. First, the authors have considered a capped loss with parameter β to prevent any single sample from dominating the objective (hereafter we use the value $\beta = 9$ that was found to be the best in [21]). Second, the authors resort to the stochastic normalized gradient method ADAM to learn the perturbation. Since their

[1] Pytorch codes of UAP and Fast-UAP baselines will be made publicly available along with our proposed SUAP attack in order to contribute to the *TorchAttacks* repository [37].

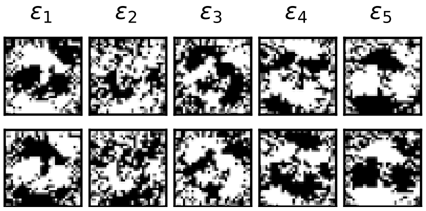

Figure 2: **Illustration of ℓ_∞-based SUAP perturbations learned on MNIST dataset.** Each of the 5 universal perturbations of 5-SUAP are reported from left to right. The top plots correspond to their positive part while the bottom plots are their negative parts.

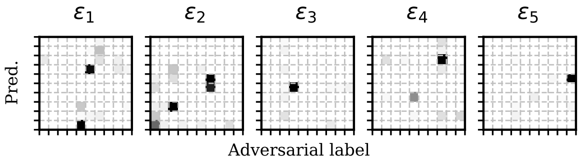

Figure 3: **Fooling matrices of ℓ_∞-based SUAP attacks on MNIST.** For each universal perturbation ε_l permitting to attack an image x, we report the predicted label $C_f(x)$ and the adversarial label $C_f(x + \varepsilon_l)$.

code is not publicly available, we have tried to reproduce their version as closely as possible.

For the sake of consistency, we have implemented a Pytorch version of the FAST-UAP [22] originally designed for TensorFlow. We use the same hyper-parameters as in their code, namely a desired fooling rate of 80%, a maximum of 10 iterations for DeepFool [10] and an overshoot of 0.02 to prevent vanishing updates.

We also compare against the CW-UAP [23] method whose code was kindly granted by the authors.

In addition, we consider standard specific attacks such as FGSM [8] and PGD [13] as well as more advanced techniques, i.e., MI-FGSM [14] and AutoAttack [17], in order to grasp the existing gap of performance between specific and universal attacks. To this effect, we resort to the *TorchAttacks* repository [37].

Illustration and insights about SUAP attacks. Similarly to the MNIST experiment, we report in Fig. 4 and Fig. 5 the learned 5-SUAP universal perturbations and their fooling matrices. We do observe that each SUAP universal perturbation plays a different role. For instance, ε_2 is mostly used to attack images of animals (*bird*, *cat*, *dog*, *frog* and *horse*) so that they become misclassified as *deer*. Indeed, with a little imagination one can distinguish two deer facing each other in ε_2. Let us also note that ε_3 is mostly employed to misclassify images of *airplane* and *ship* as *bird*.

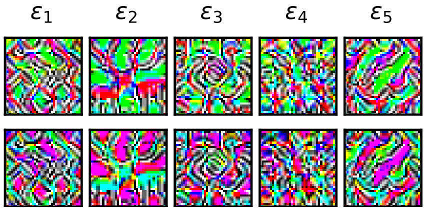

Figure 4: **Illustration of ℓ_∞-based SUAP attacks learned on CIFAR-10 dataset.** Each of the 5 universal perturbations of 5-SUAP are reported from left to right. The top plots correspond to their positive part while the bottom plots are their negative parts.

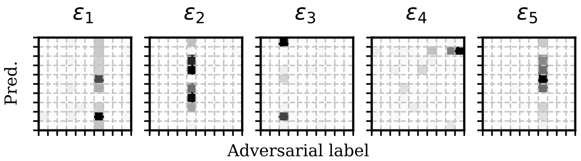

Figure 5: **Fooling matrices of ℓ_∞-based SUAP attacks on CIFAR-10.** For each universal perturbation ε_l permitting to attack an image x, we report the predicted label $C_f(x)$ and the adversarial label $C_f(x + \varepsilon_l)$. The labels are {airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck}.

We also report in Fig. 6 and Fig. 7 the fooling matrices and some universal perturbations of the CW-UAP attack and UAP attack, respectively. Note that, contrary to SUAP, we have merged all 10 fooling matrices of CW-UAP (one for each class) into a single one since they are all disjointed. Thus, each row of Fig. 6 (left) corresponds to the adversarial label obtained for each of 10 independent class-wise attacks. Unsurprisingly, the sum of SUAP fooling matrices bear some similarities with the fooling matrices corresponding to CW-UAP and UAP. Indeed, ultimately, the couple (predicted label, adversarial label) depends on the similarity between images classes and how the classifier proceeds to distinguish between the classes. Hence it makes sense that all summed fooling matrices look somewhat similar. However, the interesting point to highlight is how all three methods operate. Here, SUAP finds overlapping decompositions of the (predicted label, adversarial label) couple. As such, it automatically unveils the similarity between examples belonging to two different classes.

Impact of the numbers of training samples. Herein, we take a deeper look at the impact of two parameters on the performance of SUAP attacks. More precisely, we study the influence of the amount of training samples n and the number of perturbations L (see Problem 1) on the test fooling rate. To this end, we let n and L vary in $\{1\text{K}, 2\text{K}, 3\text{K}, 4\text{K}\}$ and $\{1, 3, 5, 7, 10\}$, respectively. All learned L-SUAP attacks are then evaluated on a distinct test set. Results, averaged over multiple splits, are reported in Fig. 8. Overall, we observe that increasing L improves the performance, thus confirming that having more perturbations is beneficial to attack the network f. However, this observation has to be contrasted with the fact that the

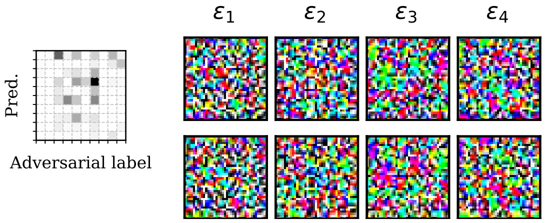

Figure 6: **Illustration of ℓ_∞-based CW-UAP attacks of the CIFAR-10 dataset.** Left: global fooling matrix where each row indicates the adversarial label obtained for each of the 10 class-wise attacks. Right: universal attacks learned for fooling the classes 0 (airplane), 2 (bird), 4 (deer) and 6 (frog).

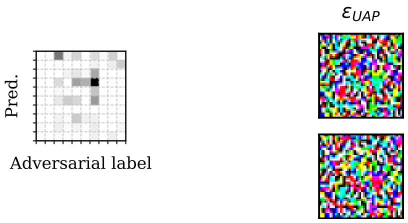

Figure 7: **Illustration of ℓ_∞-based UAP attacks of the CIFAR-10 dataset.** Left: fooling matrix. Right: universal attack

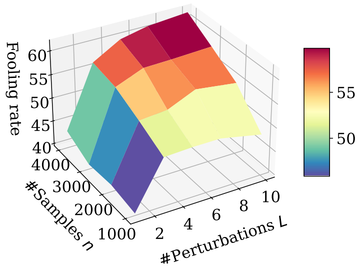

Figure 8: **Influence of the number of CIFAR-10 samples.** Depending on the number of samples $n \in \{1\text{K}, 2\text{K}, 3\text{K}, 4\text{K}\}$, we report the fooling rate of ℓ_∞-based SUAP attacks for various number of perturbations L.

Attack	ℓ_∞-fooling rate (%)	ℓ_2-fooling rate (%)
UAP-PGD [21]	12.53 (± 0.60)	2.67 (± 0.21)
FAST-UAP [22]	11.16 (± 1.03)	2.53 (± 0.19)
CW-UAP [23]	**13.85 (± 0.18)**	**2.77 (± 0.09)**
1-SUAP	36.83 (± 0.93)	3.43 (± 0.26)
3-SUAP	54.03 (± 0.54)	4.93 (± 0.50)
5-SUAP	**55.56 (± 0.57)**	**7.09 (± 1.22)**
FGSM [8]	53.82 (± 0.00)	N/A
MI-FGSM [14]	80.76 (± 0.00)	N/A
PGD [13]	**93.61 (± 0.06)**	89.23 (± 0.02)
AutoAttack [17]	93.07 (± 0.00)	**92.41 (± 0.01)**

Table 1: **Performance of attacks on a ResNet18 trained on CIFAR-10.** Bold fonts highlight the best fooling rate in universal (top), semi-universal (middle) and specific (bottom) attacks.

amount of data n required to achieve good performance goes in pair with the complexity of the learning Problem 1, hence with L. As such, for $n = 1\text{K}$ or 2K, the performance does not significantly improve (or worse, decrease) with larger L. In what follows, we restrict to a setting made of few samples (i.e. $n = 2\text{K}$).

Comparison with baselines. Performances, in terms of fooling rate, are reported in Table 1 for both ℓ_∞ and ℓ_2-attacks. First of all, we observe that 1-SUAP outperforms all universal attacks (i.e., UAP-PGD, FAST-UAP and CW-UAP) and, most importantly, it surpasses UAP-PGD which is closely related. We believe that this is due to the proposed algorithmic solution which benefits from better optimization guarantees. In addition, as the number of universal perturbation grows, we do observe an increase in performance of SUAP attacks, thus justifying the advantages of having more degrees of freedom. Interestingly, the SUAP attacks also manage to improve upon the one-shot specific FGSM attack. However, the performance are still very far behind the more advanced specific attacks. Nonetheless, such difference in performance have to be contrasted with their associated computational complexity (see Remark 1). Overall, L-SUAP yields a competitive trade-off between universality and specificity by tuning the number L of universal perturbations.

Transferability of attacks. We further evaluate how the learned attacks on the ResNet18 model manage to fool more complex architectures such as the pre-trained ResNet50 and Mobilenetv2 [38] models. We additionally consider two robust models from the *RobustBench* repository [39], namely r-ResNet18 [40] and r-ResNet50 [41], which are trained with some defense mechanisms against ℓ_∞-attacks of budget $\delta = 8/255$. Results are reported in Table 2.

Overall, SUAP systematically yields a better transferability than all universal attacks, as shown by the higher fooling rates. In addition, it also manages to outperform specific attacks when the target model architecture is significantly different than the base model on which the attacks have been learned (i.e., Mobilenetv2 vs. ResNet18). Note that this is precisely the setting where most universal attacks also show greater transferability than specific attacks. Interestingly, SUAP also shows competitive results on robust models.

5.3 ImageNet Experiments

Herein, we tackle a large scale scenario with 1K classes. Note that such setting is known to be problematic for CW-UAP [23] since computing or even storing 1K perturbations exceeds most memory storage spaces. Hence, it will not be studied here.

Attack / Model	ResNet50	Mobilenetv2 [38]	r-ResNet18 [40]	r-ResNet50 [41]
UAP-PGD [21]	21.51 (± 0.18)	39.37 (± 0.19)	2.01 (± 0.01)	2.54 (± 0.05)
FAST-UAP [22]	19.65 (± 1.22)	36.51 (± 0.30)	1.94 (± 0.01)	2.33 (± 0.05)
CW-UAP [23]	**21.95** (± **0.28**)	**39.62** (± **0.33**)	**2.26** (± **0.07**)	**2.26** (± **0.06**)
1-SUAP	27.45 (± 0.81)	44.11 (± 0.35)	2.27 (± 0.02)	2.27 (± 0.08)
3-SUAP	28.49 (± 0.56)	45.42 (± 0.32)	2.55 (± 0.03)	2.95 (± 0.08)
5-SUAP	**28.87** (± **0.07**)	**46.09** (± **0.10**)	**2.56** (± **0.05**)	**3.10** (± **0.05**)
FGSM [8]	28.55 (± 0.00)	38.10 (± 0.00)	**3.02** (± **0.00**)	3.14 (± 0.00)
MI-FGSM [14]	29.95 (± 0.01)	35.46 (± 0.01)	2.60 (± 0.00)	**3.41** (± **0.00**)
PGD [13]	30.47 (± 0.12)	**38.17** (± **0.37**)	1.94 (± 0.05)	2.53 (± 0.08)
AutoAttack [17]	**31.79** (± **0.16**)	38.08 (± 0.27)	1.91 (± 0.02)	2.41 (± 0.02)

Table 2: **Transferability of ℓ_∞-attacks on a ResNet18 trained on CIFAR-10.** Results are divided into universal (top), semi-universal (middle) and specific (bottom) attacks. Bold fonts highlight the best fooling rate in each attack category for each target model (along the columns).

Attack	ℓ_∞-fooling rate (%)
UAP-PGD [21]	**27.36** (± **0.00**)
FAST-UAP [22]	23.46 (± 0.25)
1-SUAP	83.17 (± 2.62)
5-SUAP	**88.98** (± **1.06**)
10-SUAP	87.24 (± 1.16)
FGSM [8]	84.53 (± 0.05)
MI-FGSM [14]	90.04 (± 0.02)
PGD [13]	**94.99** (± **0.06**)
AutoAttack [17]	88.23 (± 0.05)

Table 3: **Performance of ℓ_∞-attacks on a ResNet18 trained on ImageNet.** Bold fonts highlight the best fooling rate in universal (top), semi-universal (middle) and specific (bottom) attacks.

Data splitting and pre-processing. We resort to the popular ILSVRC2012 validation subset of the ImageNet dataset [42]. The 50k images are randomly split into two halves. The first half is used to learn the (semi-)universal perturbations while the second half is regarded as test set to evaluate the attacks. All images are resized into 256×256 followed by a cropping of size 224×224 around the center and a rescaling of the pixels intensity into $[0, 1]$. Results are averaged over 5 splits.

Model and attacks setting. We analyze a pretrained ResNet18 model augmented with a normalizing layer of mean $(0.485, 0.456, 0.406)$ and standard deviation $(0.229, 0.224, 0.225)$ achieving a test accuracy of 69.76%. Contrary to the previous experiments, we now consider the ProxSAGA solver of Algorithm 2 in order to learn the SUAP perturbations. The step-size and batch-size are set to $\gamma = 0.05$ and $b = 1$, respectively.

Results. Performance are reported in Table 3. Once again, we observe a drastic gap of performance between UAP-PGD and 1-SUAP, thus confirming the superiority of the numerical solution of Algorithm 2 for $L = 1$ perturbation over the standard UAP-PGD solver [21]. Overall, SUAP achieves performance of the order of magnitude as specific attacks (e.g. MI-FGSM). Therefore, it suggests that, for large-scale settings with numerous classes, solely a few universal perturbations are enough to attack most of images.

6 Conclusion

The present work introduced a framework for crafting semi-universal attacks. The latter permit to bridge the gap between universal and specific attacks by jointly learning multiple universal perturbations. When facing an unseen example, an adversarial example is built by selecting, in an unsupervised manner, the appropriate perturbation amongst all. Numerical experiments support that the number of perturbations does act as a trade-off between universality and specificity. Beyond the gain in performance, semi-universal attacks pull out of existing attacks by capturing meaningful patterns describing the most common flaws to fool the classifier. The latter shed some light both on how the classifier operates and on the existing similarities between the training instances. Future works will be devoted to the design of a defense mechanisms against semi-universal attacks as well as the derivation of generalization bounds.

References

[1] R. Miotto, F. Wang, S. Wang, X. Jiang, and J. T. Dudley, "Deep learning for healthcare: review, opportunities and challenges," *Briefings in bioinformatics*, vol. 19, no. 6, pp. 1236–1246, 2018.

[2] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.

[3] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. J. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," in *2nd International*

Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings, Y. Bengio and Y. LeCun, Eds., 2014.

[4] X. Huang, D. Kroening, W. Ruan, J. Sharp, Y. Sun, E. Thamo, M. Wu, and X. Yi, "A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability," *Computer Science Review*, vol. 37, p. 100270, 2020.

[5] A. Qayyum, M. Usama, J. Qadir, and A. Al-Fuqaha, "Securing connected amp; autonomous vehicles: Challenges posed by adversarial machine learning and the way forward," *IEEE Communications Surveys Tutorials*, vol. 22, no. 2, pp. 998–1026, 2020.

[6] B. Nassi, Y. Mirsky, D. Nassi, R. Ben-Netanel, O. Drokin, and Y. Elovici, *Phantom of the ADAS: Securing Advanced Driver-Assistance Systems from Split-Second Phantom Attacks*. New York, NY, USA: Association for Computing Machinery, 2020, p. 293–308.

[7] C. Laidlaw and S. Feizi, "Functional adversarial attacks," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32. Curran Associates, Inc., 2019, pp. 10 408–10 418.

[8] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Y. Bengio and Y. LeCun, Eds., 2015.

[9] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, mar 2016.

[10] S. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: A simple and accurate method to fool deep neural networks," in *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 2574–2582.

[11] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," in *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Workshop Track Proceedings*. OpenReview.net, 2017.

[12] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *2017 IEEE Symposium on Security and Privacy (SP)*, 2017, pp. 39–57.

[13] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," in *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[14] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. IEEE, jun 2018.

[15] X. Wang, J. Lin, H. Hu, J. Wang, and K. He, "Boosting adversarial transferability through enhanced momentum," *arXiv preprint arXiv:2103.10609*, 2021.

[16] J. Lin, C. Song, K. He, L. Wang, and J. E. Hopcroft, "Nesterov accelerated gradient and scale invariance for adversarial attacks," in *International Conference on Learning Representations*, 2020.

[17] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 2206–2216.

[18] C. Finlay, A.-A. Pooladian, and A. Oberman, "The logbarrier adversarial attack: making effective use of decision boundary information," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 4862–4870.

[19] Y. Zhang, X. Tian, Y. Li, X. Wang, and D. Tao, "Principal component adversarial example," *IEEE Transactions on Image Processing*, vol. 29, pp. 4804–4815, 2020.

[20] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jul 2017.

[21] A. Shafahi, M. Najibi, Z. Xu, J. Dickerson, L. S. Davis, and T. Goldstein, "Universal adversarial training," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 5636–5643, apr 2020.

[22] J. Dai and L. Shu, "Fast-UAP: An algorithm for expediting universal adversarial perturbation generation using the orientations of perturbation vectors," *Neurocomputing*, vol. 422, pp. 109–117, jan 2021.

[23] P. Benz, C. Zhang, A. Karjauv, and I. S. Kweon, "Universal adversarial training with class-wise perturbations," in *2021 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2021, pp. 1–6.

[24] C. Zhang, P. Benz, C. Lin, A. Karjauv, J. Wu, and I. S. Kweon, "A survey on universal adversarial attack," in *IJCAI*, 2021, pp. 4687–4694.

[25] C. Laidlaw, S. Singla, and S. Feizi, "Perceptual adversarial robustness: Defense against unseen threat models," in *International Conference on Learning Representations*, 2021.

[26] S. Bonettini, I. Loris, F. Porta, M. Prato, and S. Rebegoldi, "On the convergence of a linesearch based proximal-gradient method for nonconvex optimization," *Inverse Problems*, vol. 33, no. 5, p. 055005, 2017.

[27] P. L. Combettes and J.-C. Pesquet, "Lipschitz certificates for layered network structures driven by averaged activation operators," *SIAM Journal on Mathematics of Data Science*, vol. 2, no. 2, pp. 529–557, 2020.

[28] H. Gouk, E. Frank, B. Pfahringer, and M. J. Cree, "Regularisation of neural networks by enforcing lipschitz continuity," *Machine Learning*, vol. 110, no. 2, pp. 393–416, 2021.

[29] H. Attouch, J. Bolte, and B. F. Svaiter, "Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods," *Mathematical Programming*, vol. 137, no. 1-2, pp. 91–129, aug 2011.

[30] J. Zeng, T. T.-K. Lau, S. Lin, and Y. Yao, "Global convergence of block coordinate descent in deep learning," in *Proceedings of the 36th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri and R. Salakhutdinov, Eds., vol. 97. PMLR, 09–15 Jun 2019, pp. 7313–7323. [Online]. Available: http://proceedings.mlr.press/v97/zeng19a.html

[31] S. J. Reddi, S. Sra, B. Poczos, and A. J. Smola, "Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization," in *Advances in Neural Information Processing Systems*, D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, Eds., vol. 29. Curran Associates, Inc., 2016.

[32] Z. Wang, K. Ji, Y. Zhou, Y. Liang, and V. Tarokh, *SpiderBoost and Momentum: Faster Stochastic Variance Reduction Algorithms*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[33] N. H. Pham, L. M. Nguyen, D. T. Phan, and Q. Tran-Dinh, "Proxsarah: An efficient algorithmic framework for stochastic composite nonconvex optimization," *Journal of Machine Learning Research*, vol. 21, no. 110, pp. 1–48, 2020. [Online]. Available: http://jmlr.org/papers/v21/19-248.html

[34] D. C. Cireşan, U. Meier, L. M. Gambardella, and J. Schmidhuber, "Deep, Big, Simple Neural Nets for Handwritten Digit Recognition," *Neural Computation*, vol. 22, no. 12, pp. 3207–3220, 12 2010.

[35] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, Ontario, Tech. Rep. 0, 2009.

[36] H. Phan, "huyvnphan/pytorch_cifar10," Jan. 2021. [Online]. Available: https://doi.org/10.5281/zenodo.4431043

[37] H. Kim, "Torchattacks: A pytorch repository for adversarial attacks," *arXiv preprint arXiv:2010.01950*, 2020.

[38] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "Mobilenetv2: Inverted residuals and linear bottlenecks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 4510–4520.

[39] F. Croce, M. Andriushchenko, V. Sehwag, E. Debenedetti, N. Flammarion, M. Chiang, P. Mittal, and M. Hein, "Robustbench: a standardized adversarial robustness benchmark," 2021.

[40] V. Sehwag, S. Mahloujifar, T. Handina, S. Dai, C. Xiang, M. Chiang, and P. Mittal, "Robust learning meets generative models: Can proxy distributions improve adversarial robustness?" in *International Conference on Learning Representations*, 2022.

[41] T. Chen, S. Liu, S. Chang, Y. Cheng, L. Amini, and Z. Wang, "Adversarial robustness: From self-supervised pre-training to fine-tuning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 699–708.

[42] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.