Semi-Universal Adversarial Perturbations

Jordan Frecon, Paul Viallard, Emilie Morvant, Gilles Gasso, Amaury Habrard, Stéphane Canu

SEMI-UNIVERSAL ADVERSARIAL PERTURBATIONS

A PREPRINT

Jordan Frecon* **Paul Viallard†** **Emilie Morvant*** **Gilles Gasso‡** **Amaury Habrard*** **Stéphane Canu‡**

ABSTRACT

This paper introduces semi-universal perturbations that bridge the gap between specific and universal adversarial perturbations. The original idea is to craft a specific perturbation by choosing it among a set of L universal perturbations. We propose to jointly learn the perturbations of this set to maximize the chances to attack each example by allowing it to choose its own perturbation. To do so, we derive an algorithm, with convergence guarantees under Lipschitz continuity assumptions. Semi-universal perturbations offer a better flexibility, interpretability and diversity, confirmed by our experiments. Additionally, we provide a generalization bound on the abilities of the perturbations to attack new examples.

1 Introduction

Embedded technologies using artificial Neural Networks (NN) are increasingly present in our daily lives. Their high expressive power has shown a great success in various complex tasks [36, 21]. However, some concerns have been raised about their safety and more particularly for the safety of the user [24] since the pioneer work of Szegedy et al. [50] which has shown the existence of adversarial attacks. The most striking example is that of automated vehicles, where malicious attacks could lead the car to take unwanted action with dramatic consequences [46, 42].

Most of the adversarial attacks are quasi-negligible perturbations that fool the NN prediction. From a fast one-shot method [19] to the first iterative procedures [43, 39, 30, 9, 34], the crafting of adversarial perturbations has lately received a lot of attention from the machine learning community. To this regard, momentum-based methods [16, 53] have shown a promising boost in the transferability of the attacks learned on one NN to other NNs. In addition, various contributions have investigated algorithmic concerns leading to accelerated and scale-invariant attacks [33] as well as parameter-free attacks [13]. In another line of research, Finlay et al. [18] designed attacks exploiting the decision boundary of NNs, while Zhang et al. [61] proposed to take into account the structure of images through a principal component analysis. A key particularity of all the above attacks is that they are *specific* (or *example-based*), meaning that they are crafted to attack a *single* example. Henceforth, to attack a new example, one needs to learn the associated perturbation once again. Although they are very effective, they have the major drawback of being time consuming.

On the other end of the spectrum, *universal* (or *example-agnostic*) attacks [60] aim to find an attack which, once learned, can be applied to every new example. Moosavi-Dezfooli et al. [40] first demonstrated that there exists a single perturbation, coined universal adversarial perturbation (UAP), which, when added to any new example, is very likely to fool the classifier; a variant exploiting the orientations of the perturbation vectors was proposed by Dai and Shu [15]. Later, Shafahi et al. [49] devised a more efficient method by hinging on a projected gradient descent algorithm. In addition, inspired by the observation that UAP does not attack all classes equally, Benz et al. [7] proposed a class-based universal perturbation. Although these perturbations are universal, it is hard to interpret why they work on a case-to-case basis. In general, current state-of-the-art universal attacks remain hardly interpretable out-of-the-box and require *a posteriori* tailored studies [59]. These works have suggested that a reasonable assumption is that the perturbations should live in a low-dimensional manifold. This assumption has been justified by Gu and Rigazio [22], and later by Tabacof and Valle [51]. Some works proposed solutions to learn such a manifold [27, 61, 4, 23, 55]. Finally, Zhang et al. [60] suggest that simple gradient-based UAP methods may lead to better fooling performance.

*Université Jean Monnet Saint-Etienne, CNRS, Institut d'Optique Graduate School, Laboratoire Hubert Curien UMR 5516, F-42023, Saint-Etienne, France

†Inria, CNRS, Ecole Normale Supérieure, PSL Research University, Paris, France

‡Normandie Univ, INSA Rouen, UNIROUEN, UNIHAVRE, LITIS, Rouen, France

Contributions. We propose a new kind of perturbations, that we call *semi-universal perturbations* and intend to bridge the gap between specific and universal perturbations. The original idea is to *specifically* craft an attack for each example by choosing, in an unsupervised manner, a perturbation among a set of L different *universal* perturbations. To do so, we define an optimization problem to jointly learn the L perturbations allowing each example to choose its own perturbation (L can be seen as a tuning parameter controlling the amount of diversity between the perturbations). To solve it, we derive a gradient-based solution with convergence guarantees. Additionally, to justify that the L semi-universal perturbations learned can be used to attack a model on unseen examples, we derive a generalization bound on the deviation between the true and the empirical fooling risks. Put into words, we get a bound on how much the learned perturbations are able to fool new examples, confirming that we can use the learned perturbations to attack a model on data coming from the same task. We then propose a simple attack procedure. Lastly, our experiments confirm the effectiveness of the semi-universal perturbations over previous gradient-based UAP, in line with the conclusion of [60]. Our results additionally show that they lead to more interpretable and transferable attacks.

Outline. We recall the general framework of adversarial perturbations in Section 2. We introduce in Section 3 our semi-universal perturbations as well as an algorithmic solution to learn them. Then Section 4 is devoted to attacking with these perturbations: we derive a generalization bound on the quality of the perturbations to attack unseen examples in order to justify our attack procedure. Before concluding, numerical experiments are conducted in Section 5 on multiple benchmark datasets.

2 Preliminaries and related works

2.1 Adversarial perturbations

We stand in a multiclass setting where $\mathcal{X} \subseteq \mathbb{R}^P$ is a P-dimensional input space and $\mathcal{Y} = \{1, \dots, c\}$ is the set of $c \in \mathbb{N}_+$ classes. We consider an unknown data distribution D on $\mathcal{X} \times \mathcal{Y}$ that models the task; the associated marginal distribution on $\mathcal{X}$ is $D_{\mathcal{X}}$. We use the notation D^n, *resp.* $D_{\mathcal{X}}^n$, for the distribution of a sample constituted of n data points *i.i.d.* sampled from D, *resp.* $D_{\mathcal{X}}$.

We consider a *trained* model $f : \mathbb{R}^P \to \mathbb{R}^c$ which associates each example[4] $x \in \mathcal{X}$ to its probabilities $f(x) \in \mathbb{R}^c$ to belong to any of the c classes from $\mathcal{Y}$. The predicted class of x by f is then defined as $C_f(x) = \operatorname{argmax}_{y \in \mathcal{Y}} f(x)_y$. Adversarial learning aims to find for each *original example* $x \sim D_{\mathcal{X}}$ a point $a \in \mathcal{X}$ close to x such that $C_f(a) \neq C_f(x)$. Since a is close to x, one can expect f to predict the same class for both examples. Thus, a is called *adversarial example* and said to fool the classifier C_f.

There exists a vast literature on the ways to build adversarial examples, measure their closeness to original examples, and quantify how much they affect the decision process of f. Here, we embrace the common setting of adversarial example $a = x + \varepsilon$ built by adding an *adversarial perturbation* ε to an original example x. We consider that the two are close if the ℓ_p-norm of the added perturbation is small enough [31], *i.e.*, $\|\varepsilon\|_p \leq \delta$, for some small budget $\delta > 0$. In addition, since asking $C_f(a) \neq C_f(x)$ can lead to numerical instabilities [39] during the learning of ε, we consider a loss function $H : \mathbb{R}^c \times \mathcal{Y} \to \mathbb{R}$ taking as inputs $f(a)$ and a class k from $\mathcal{Y}$ that is either $C_f(x)$ or the original class y of x. In this paper, we use the cross-entropy loss (or its $[0, 1]$-bounded counterpart [17]).

Given the model f, and an unlabeled data set $S_{\mathcal{X}} = \{x_i\}_{i=1}^n \sim D_{\mathcal{X}}^n$, there exist two current paradigms for crafting adversarial perturbations: *(i) specific attacks*, where for each $x \in S_{\mathcal{X}}$ we look for a perturbation $\varepsilon(x) \in \mathcal{B}_p(\delta) = \{e \in \mathbb{R}^P \mid \|e\|_p \leq \delta\}$, specifically tailored to attack the example x (hereafter, we drop the dependency on x and simply denote ε); *(ii) universal attacks*, where we look for a perturbation ε such that $a = x + \varepsilon$ is an adversarial example for all x from $S_{\mathcal{X}}$. To learn such adversarial perturbations, we assume that we have a labeled sample $S = \{(x_i, y_i)\}_{i=1}^n \in (\mathcal{X} \times \mathcal{Y})^n$ consisting of n data points. We recall in the next subsections the most popular specific and universal attacks [60]. Note that, given some convex set $\mathcal{C}$, we denote by $\operatorname{Proj}_{\mathcal{C}}$ the projection onto $\mathcal{C}$.

2.2 Most popular specific attacks

DeepFool [39]. For a given $x \in S_{\mathcal{X}}$, the DeepFool attack is the smallest ℓ_p perturbation managing to fool the classifier C_f. More formally, it solves $\operatorname{minimize}_{\varepsilon \in \mathbb{R}^P} \|\varepsilon\|_p$ s.t. $C_f(x + \varepsilon) \neq C_f(x)$.

PGD [34]. Given S, and a loss function H, the adversarial perturbation for a given $(x, y) \in S$ is defined as the one inside the ℓ_p-ball which maximizes the loss between $x + \varepsilon$ and y, *i.e.*,

$$\underset{\varepsilon \in \mathbb{R}^P}{\operatorname{maximize}} \; H(f(x + \varepsilon), y) \quad \text{s.t.} \quad \|\varepsilon\|_p \leq \delta. \tag{1}$$

[4] 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]).

In practice, the opposite of the objective in Equation (1) is minimized by resorting to a projected gradient method, hence the name of the attack.

2.3 Most popular universal attacks

UAP [40]. The first universal perturbation ε was crafted by aggregating the DeepFool perturbations $\Delta\varepsilon_i$ associated to all samples $x_i \in \{x \in S_{\mathcal{X}} \mid C_f(x+\varepsilon) = C_f(x)\}$. The aggregation step is defined as $\varepsilon \leftarrow \mathrm{Proj}_{\mathcal{B}_p(\delta)}(\varepsilon+\Delta\varepsilon_i)$ and is repeated until some fooling rate is reached.

Fast-UAP [15]. This attack follows the UAP procedure but, instead of aggregating all the perturbations $\Delta\varepsilon_i$, it only adds the perturbation with the closest orientation to the current iterate ε.

UAP-PGD [49]. Given S, the UAP-PGD attack elaborates upon PGD by framing the universal perturbation as the solution of the following optimization problem

$$\underset{\varepsilon\in\mathbb{R}^P}{\text{maximize}} \ \frac{1}{n}\sum_{(x_i,y_i)\in S} H(f(x_i+\varepsilon), y_i) \quad \text{s.t.} \quad \|\varepsilon\|_p \leq \delta. \tag{2}$$

CW-UAP [7]. Recently, UAP-PGD has been extended to class-wise UAP where a universal perturbation is built for each class. Let us denote $\forall k \in \mathcal{Y},\ S_k = \{x_i \mid (x_i, k) \in S\}$ the set of training points of the k-th class and n_k the size of S_k, then CW-UAP aims at solving

$$\underset{\{\varepsilon_k\in\mathbb{R}^P\}_{k\in\mathcal{Y}}}{\text{maximize}} \ \sum_{k\in\mathcal{Y}} \frac{1}{n_k} \sum_{x_i\in S_k} H(f(x_i+\varepsilon_k), k) \quad \text{s.t.} \quad \forall k \in \mathcal{Y},\ \|\varepsilon_k\|_p \leq \delta. \tag{3}$$

The solution amounts in learning multiple independent UAP-PGD perturbations, one for each class.

Our contribution starts from this principle of learning one perturbation per class, or at least learning a set of perturbations. Our original idea is to craft a *specific* perturbation by choosing it among a set of *universal* perturbations, that we call *semi-universal perturbations*. To do so, instead of learning the perturbations independently, we propose in Section 3 an approach that consists of learning $L \in \mathbb{N}_+$ perturbations jointly. We believe that this increases the chances of attacking each example by providing a better expressiveness implied by the facts *(i)* that the perturbations are learned jointly and *(ii)* that for each example we choose the perturbation maximizing H the most. This is confirmed by our experiments in Section 5 which supports the ability to efficiently attack with $L < c$ perturbations.

3 Crafting semi-universal perturbations

3.1 Learning the set of semi-universal perturbations

The originality of our work is to jointly learn $L \in \mathbb{N}_+$ universal adversarial perturbations $\{\varepsilon_1, \ldots, \varepsilon_L\}$ where each $\varepsilon_l \in \mathcal{B}_p(\delta)$. To do so, we propose to frame the learning procedure as follows.

Problem 1 (Learning semi-universal ℓ_p-perturbations)**.** *Let L be the number of semi-universal perturbations to learn. Given $S = \{(x_i, y_i)\}_{i=1}^n$, and the model f, find $\boldsymbol{\varepsilon} = [\varepsilon_1, \ldots, \varepsilon_L]$ solving*

$$\underset{\boldsymbol{\varepsilon}\in\mathcal{B}_p(\delta)^L}{\text{maximize}} \left\{ \mathcal{L}(\boldsymbol{\varepsilon}) := \frac{1}{n}\sum_{i=1}^{n} \max_{\varepsilon_l\in\boldsymbol{\varepsilon}} H(f(x_i+\varepsilon_l), y_i) \right\}, \tag{4}$$

For each pair $(x_i, y_i) \in S$, the objective in Equation (4) only picks one of the L perturbations in $\boldsymbol{\varepsilon}$ which maximizes the loss between $f(x_i+\varepsilon_l)$ and y_i the most. In other words, learning aims at being specific enough for each example while being sufficiently universal to cover all the pairs in S. As later shown in Section 5, it results that $\boldsymbol{\varepsilon}$ shows a great diversity. Note that when $L = 1$ (*i.e.*, for a single perturbation) Equation (4) boils down to Equation (2). In addition, it bears similarities with Equation (3) when L equals the number of classes and each ε_l is independently learned on S_l.

Due to the model f, it is worth stressing that Problem 1 is a difficult non-concave maximization problem. Finding its global solution is thus out of reach. To tackle this challenge, we detail below an algorithmic solution to efficiently find an approximate solution. Note that we defer in Appendix A an alternative solution based on a stochastic solver fully exploiting the finite-sum nature of Problem 1.

3.2 Gradient-based numerical solution

To maximize $\mathcal{L}$ in Problem 1, we embrace a projected gradient ascent algorithm augmented with an Armijo-like line-search strategy (for ensuring 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 $\boldsymbol{\varepsilon} =$

Algorithm 1 L-SUAP

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

$[\varepsilon_1, \ldots, \varepsilon_L]$ and some sequence of step-sizes $\{\gamma_k\}_{k \in \mathbb{N}_+}$. Then, at each iteration $k \in \mathbb{N}_+$ the algorithm looks for $\boldsymbol{\varepsilon}^{(k+1/2)} \in \mathcal{B}_p(\delta)^L$ which maximizes the linearized surrogate $h^{(k)}(\boldsymbol{\varepsilon}) = \mathcal{L}(\boldsymbol{\varepsilon}^{(k)}) + \left\langle \nabla \mathcal{L}(\boldsymbol{\varepsilon}^{(k)}), \boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^{(k)} \right\rangle - (1/2\gamma_k)\|\boldsymbol{\varepsilon} - \boldsymbol{\varepsilon}^{(k)}\|^2$ with $\langle \cdot, \cdot \rangle$ the Frobenius inner product and $\|\cdot\|$ the induced norm. Such choice is motivated by the fact that, for concave and μ-smooth functions $\mathcal{L}$, and for all $\gamma_k \leq \frac{1}{\mu}$, we have $\mathcal{L}(\boldsymbol{\varepsilon}) \geq h^{(k)}(\boldsymbol{\varepsilon})$. Henceforth, we have

$$\boldsymbol{\varepsilon}^{(k+1/2)} = \underset{\boldsymbol{\varepsilon} \in \mathcal{B}_p(\delta)^L}{\operatorname{argmax}} \, h^{(k)}(\boldsymbol{\varepsilon}) = \mathrm{Proj}_{\mathcal{B}_p(\delta)^L}\left(\boldsymbol{\varepsilon}^{(k)} + \gamma_k \nabla \mathcal{L}(\boldsymbol{\varepsilon}^{(k)})\right), \tag{5}$$

which recasts into one projected gradient ascent step. Note that the differentiability of $\mathcal{L}$ depends on the choice H and on the NN f to attack. For instance, for ReLu-based NN, it is likely that $\nabla \mathcal{L}(\boldsymbol{\varepsilon}^{(k)})$ is not well-defined[5]. 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 $\boldsymbol{\varepsilon}^{(k+1)} = (1-\alpha_k)\boldsymbol{\varepsilon}^{(k)} + \alpha_k \boldsymbol{\varepsilon}^{(k+1/2)}$, where the relaxation parameter $\alpha_k \in (0,1]$ is appropriately chosen by an Armijo-like line-search strategy to ensure [8] some sufficient increase in $\mathcal{L}$. This algorithmic procedure is sketched in Algorithm 1 and referred as L-SUAP. In practice, we suggest initializing 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 $\mathcal{B}_p(\delta)$. This procedure comes with convergence guarantees stated below.

Theorem 1 (Convergence [8])**.** *Let $\{\boldsymbol{\varepsilon}^{(k)}\}_{k \in \mathbb{N}}$ be the sequence of Algorithm 1. Suppose that $\nabla \mathcal{L}$ is Lipschitz continuous. Then each limit point of $\{\boldsymbol{\varepsilon}^{(k)}\}_{k \in \mathbb{N}}$ is a stationary point of Problem 1 and $\{\mathcal{L}(\boldsymbol{\varepsilon}^{(k)})\}_{k \in \mathbb{N}}$ converges towards the objective value at the limit point. 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 is central to ensure convergence guarantees. Note that studying the Lipschitz continuity of NN and obtaining sharp Lipschitz constant is difficult (*e.g.*, [12, 20]).

Remark 1. *Many functions met in NNs (e.g., activation functions, loss) are semi-algebraic or tame, and thus, satisfy the KŁ property [2, 58]. Since these concepts are stable under many operations, it is reasonable to assume that many deep NNs f are likely to satisfy the KŁ property and so does $\mathcal{L}$.*

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

Now we detail a solution to attack unseen examples with semi-universal perturbations. To justify our procedure presented in Section 4.2, we first provide in Section 4.1 a bound on the generalization capacities of the learned perturbations on unseen examples.

[5]Note that $\mathcal{L}$ is first and foremost not differentiable because of the $\max$ term. However, empirically, iterates almost never lie at such singularities. To avoid this concern, one can replace $\max$ by a smooth approximation.

4 Attacking unseen examples

4.1 Generalization guarantees: we can attack on unseen examples

The performance of the learned semi-universal perturbations on unseen examples can be approximated through *generalization bounds* [37]. Formally, given a model $f : \mathbb{R}^P \to \mathbb{R}^c$, and a loss function $H : \mathbb{R}^c \times \mathcal{Y} \to [0,1]$, we are interested in the *quality* of the learned perturbations $\boldsymbol{\varepsilon} \in \mathcal{B}_p(\delta)^L$ on new examples that we call the *true fooling risk* and defined as $R_D^f(\boldsymbol{\varepsilon}) = \mathbb{E}_{(x,y)\sim D} \max_{\varepsilon_l \in \boldsymbol{\varepsilon}} H(f(x+\varepsilon_l), y)$. Since D is unknown, we have no access to its value, but we can compute its empirical counterpart on a dataset $S \sim D^n$ that we call *empirical fooling risk* and defined by $R_S^f(\boldsymbol{\varepsilon}) = \frac{1}{n}\sum_{i=1}^n \max_{\varepsilon_l \in \boldsymbol{\varepsilon}} H(f(x_i+\varepsilon_l), y_i)$. Note that these definitions extend the fooling rate, for which H is the 0-1-loss. Since our objective is to learn perturbations that fool the model, our goal is to maximize the risk. To ensure that $R_S^f(\boldsymbol{\varepsilon})$ is a good estimate of $R_D^f(\boldsymbol{\varepsilon})$, we prove the following theorem based on the Rademacher complexity [5]. This theorem is a generalization bound on the deviation between the two values. For the sake of completeness, we provide in Appendix the version of the bound based on the empirical Rademacher complexity.

Theorem 2 (*Proof deferred in Appendix*)**.** *For any distribution D on $\mathcal{X} \times \mathcal{Y}$, for any loss function $H : \mathbb{R}^c \times \mathcal{Y} \to [0,1]$, for any model $f : \mathbb{R}^P \to \mathbb{R}^c$, for any $L \in \mathbb{N}_+$, for any budget $\delta > 0$, for any ℓ_p-norm with $p \geq 0$, for any $\lambda \in (0,1]$, we have*

$$\mathop{\mathbb{P}}_{S\sim D^n}\left(\forall \boldsymbol{\varepsilon} \in \mathcal{B}_p(\delta)^L,\ \left|R_D^f(\boldsymbol{\varepsilon}) - R_S^f(\boldsymbol{\varepsilon})\right| \leq 2\,\mathfrak{R}_S^{\boldsymbol{\sigma}}\left[\mathcal{B}_p(\delta)^L\right] + 3\sqrt{\frac{\ln\frac{4}{\lambda}}{2n}}\right) \geq 1-\lambda, \tag{6}$$

where we define the Rademacher complexity [6] of the "family of perturbations" $\mathcal{B}_p(\delta)^L$ as

$$\mathfrak{R}_S\left[\mathcal{B}_p(\delta)^L\right] = \mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim\Sigma^n} \sup_{\boldsymbol{\varepsilon}\in\mathcal{B}_p(\delta)^L}\left[\frac{1}{n}\sum_{i=1}^n \sigma_i \max_{\varepsilon_l\in\boldsymbol{\varepsilon}} H(f(x_i+\varepsilon_l), y_i)\right], \tag{7}$$

with $\boldsymbol{\sigma} = \{\sigma_i\}_{i=1}^n \sim \Sigma^n$, and Σ being the Rademacher distribution, i.e., $\Sigma(-1) = \frac{1}{2}$ and $\Sigma(+1) = \frac{1}{2}$.

Equation (6) tells us that for all the possible sets of $L \in \mathbb{N}_+$ perturbations $\boldsymbol{\varepsilon}$ belonging to $\mathcal{B}_p(\delta)^L$, the empirical risk $R_S^f(\boldsymbol{\varepsilon})$ does not deviate too much from the true risk $R_D^f(\boldsymbol{\varepsilon})$ when the Rademacher complexity $\mathfrak{R}_S^{\boldsymbol{\sigma}}[\mathcal{B}_p(\delta)^L]$ is small. From the attacker's point of view, the associated lower bound on $R_D^f(\boldsymbol{\varepsilon})$ is of interest and gives an estimate of the "chances" to fool the model. Put into words, the more the perturbations learned manage to fool a model on S and the lower the Rademacher complexity, the higher the chances to fool the model on new examples.

Note that Theorem 2 is valid for any perturbations $\boldsymbol{\varepsilon}$ that live in $\mathcal{B}_p(\delta)^L$ (whatever the value of L, the budget δ and the p-norm), implying that our result is general enough to stand not only for the classical universal perturbations from $\mathcal{B}_p(\delta)$ (*e.g.*, for UAP-PGD with $L=1$), but also for semi-universal perturbations learned from Problem 1. It is important to notice that Theorem 2 holds for any model f we want to attack that is not necessarily the one we used to learn $\boldsymbol{\varepsilon}$. Hence, the bound holds in the context of adversarial transferability.

The form of this bound is quite classical, but it has the originality to state a theoretical certification from the attacker's point of view to estimate—given a model f—the true fooling risk $R_D^f(\boldsymbol{\varepsilon})$ to quantify *how much the learned perturbations are able to fool a given model on unseen examples*. Indeed, in the literature many recent works aim to understand the generalization abilities in an adversarial setting, but they take the defender's point of view and provide generalization bounds for the so-called *true adversarial risk* that measures *how much the learned model is able to face adversarial attacks on unseen examples*. In other words, while we study the fooling abilities of the perturbations for a given model, they study the performance (or robustness) of the model when this latter is attacked by adversarial perturbations (without knowing the attack). Among these works, we can mention Yin et al. [57], Khim and Loh [26], Awasthi et al. [3] that are based on an *adversarial* Rademacher complexity of the class of the models. Other generalization bounds have been derived for the adversarial risk, such as with VC-dimension [1, 38], with covering numbers [41], with algorithmic stability [56], or in PAC-Bayes [52].

4.2 Methodology: how to attack unseen examples

According to Theorem 2, once $\boldsymbol{\varepsilon}$ has been learned with Algorithm 1 from a sample $S \sim D^n$, we can attack a new example by picking one perturbation amongst the L perturbations as follows.

Problem 2 (Attacking unseen example)**.** *Given some data pair $(x, y) \sim D$, given a model f, the corresponding adversarial attack reads $a = \mathrm{Proj}_{\mathcal{X}}(x+\hat{\varepsilon})$ where $\hat{\varepsilon} = \mathrm{argmax}_{\varepsilon_l\in\boldsymbol{\varepsilon}} H(f(x+\varepsilon_l), y)$. If we do not have access to y, and assuming that f is a well-performing model, then y is replaced by $C_f(x)$.*

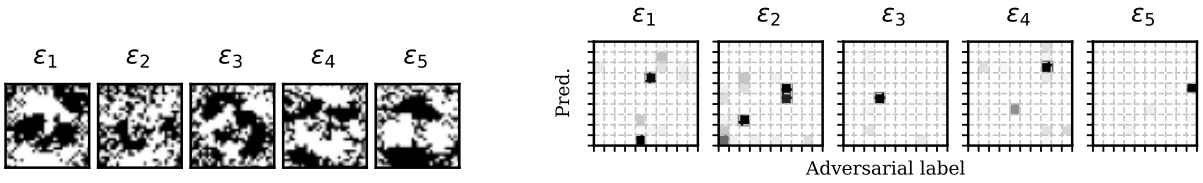


Figure 1: ℓ_∞**-based SUAP attacks on MNIST**. (Left panel) Learned adversarial perturbations. (Right panel) In the fooling matrix, labels range from 0 to 9 from top to bottom and from left to right.

When f is a NN, in order to evaluate which perturbation from $\boldsymbol{\varepsilon} = [\varepsilon_1, \dots, \varepsilon_L]$ maximizes the loss function, solving Problem 2 requires performing L independent forward passes through f. Note that, since they are independent, they can be performed in parallel to accelerate the computation. We provide below a complexity comparison between specific, semi-universal and universal attacks.

Remark 2. *Given a model f that is a neural network 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,* $\text{cost} \sim O(2Kd)$*; and* (semi-universal) *for L perturbations,* $\text{cost} \sim O(Ld)$*;* (universal) $\text{cost} \sim O(1)$.

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 [19]) and our proposed semi-universal attack achieve comparable complexity for small L.

5 Numerical experiments

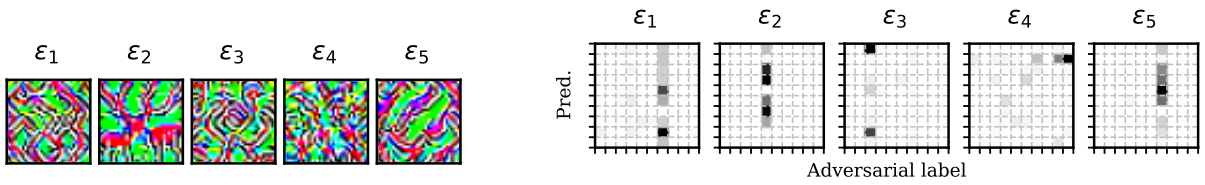
We now conduct experiments on 3 popular benchmark classification datasets and 2 NN architectures: a differentiable multi-layer perceptron (MNIST) and ResNets (CIFAR-10 and ImageNet). Hereafter, we consider ℓ_∞-attacks with a maximum budget $\delta = 8/255$. For reproducibility purposes we report implementation details such as pre-processing, data splitting and models tuning in the supplementary material, as well as results on ℓ_2-attacks.

5.1 MNIST Experiments

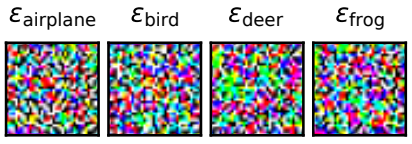
We first consider the MNIST dataset [32] useful to interpret the learned adversarial perturbations.

Illustration and role of the universal perturbations. We report in Figure 1 (left) the learned universal perturbations of 5-SUAP. Interestingly, they all exhibit strong patterns. In particular, we observe that ε_1 and ε_5 are very similar up to the sign difference. Indeed, since our framework does not handle the tuning of the sign of the perturbation, it might happen that 2 perturbations are the opposite of each others. It is worth noticing that the universal perturbations learned are consistent throughout multiple splits and random initializations. Moreover, we report in Figure 1 (right) the fooling matrices associated to each of the perturbations $\{\varepsilon_l\}_{l=1}^5$. The latter shows the correspondence between the predicted target $C_f(x)$ of some image x (in lines) and the label of the associated adversarial attack (in columns), *i.e.*, $C_f(x{+}\hat{\varepsilon})$ (see Problem 2). The fooling matrices highlight that each universal perturbation plays a different role. For instance, ε_1 mostly allows to attack images of digits "3" and "9" into being misclassified as "5" and "4", respectively. Instead, ε_3 is mainly used to attack images of "5" into "3". Coincidentally, one can distinguish the tilted digit "3" in ε_3. As opposed to CW-UAP, our SUAP attack automatically captures the similarity between multiple digits such as "3" and "9".

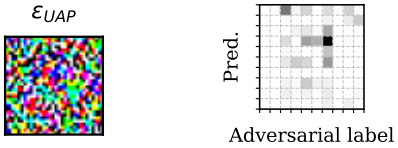
Illustration of the behavior of the generalization bound. We report in Figure 3 an estimation of the lower and upper bounds on $R_D^f(\boldsymbol{\varepsilon})$ from Theorem 2. The Rademacher complexity is approximated by resorting to the maximization Algorithm 1 where $\mathcal{L}$ is replaced by the quantity inside the $\sup$ of Equation (7). The maximum value of the objective is then averaged over multiple draw of $\boldsymbol{\sigma} \sim \Sigma^n$. As expected, we observe that the empirical fooling risk increases with L. While the Rademacher complexity (and so the generalization gap between $R_D^f(\boldsymbol{\varepsilon})$ and $R_S^f(\boldsymbol{\varepsilon})$) increases also with L it is interesting to remark that the lower bound on $R_D^f(\boldsymbol{\varepsilon})$ tends to grow, suggesting that considering semi-universal perturbations can increase the chances to fool a model.



(a) 5-SUAP attacks.



(b) CW-UAP attacks. Each row of the fooling matrix indicates the adversarial label obtained for each 10 class-wise attacks.



(c) UAP attack.

Figure 2: ℓ_∞**-based attacks on CIFAR-10**. In the fooling matrices, labels range {airplane, automobile, bird, cat, deer, dog, frog, horse, ship, truck} from top to bottom and left to right.

5.2 CIFAR-10 experiments

We turn to CIFAR-10 dataset [29] and compare the performance of our attack with the baselines.

Baselines. Our L-SUAP attack is brought into comparison against the following universal attacks[6].

• We compare with the UAP-PGD proposed by Shafahi et al. [49] which is closely related to our 1-SUAP with a single perturbation, but 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 [49]). Second, the authors resort to the stochastic normalized gradient method ADAM to learn the perturbation. Since their code is not publicly available, we tried to reproduce their version as closely as possible.

• For the sake of consistency, we implemented a Pytorch version of the FAST-UAP [15] originally designed for TensorFlow. We use the same hyper-parameters as in their code: a desired fooling rate of 80%, a maximum of 10 iterations for DeepFool, an overshoot of 0.02 to prevent vanishing updates.

• We compare against the CW-UAP [7] method whose code was kindly granted by the authors.

• We consider standard specific attacks such as FGSM [19] and PGD [34] as well as more advanced techniques, *i.e.*, MI-FGSM [16] and AutoAttack [13], in order to grasp the existing gap of performance between specific and universal attacks. To this effect, we resort to the *TorchAttacks* repository [28].

Illustration and insights about SUAP attacks. Similarly to the MNIST experiment, we report in Figure 2a the learned 5-SUAP universal perturbations (left) and their fooling matrices (right). We observe that each SUAP universal perturbation plays a different role and illustrates diversity in perturbations. Indeed, for instance, ε_2 is mostly used to attack images of animals (*bird*, *cat*, *dog*, *frog*, *horse*) so that they become misclassified as *deer*: in ε_2 one can distinguish two deer facing each other. Another example is ε_3 that is mostly used to misclassify images of *airplane* and *ship* as *bird*; in ε_3 one can distinguish a bird. In addition, we report in Figures 2b and 2c the fooling matrices and some universal perturbations of the CW-UAP attack and UAP attack, respectively. 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 Figure 2b (left) corresponds to the adversarial label obtained for each of 10 independent class-wise attacks. Unsurprisingly, many of the same couples (predicted label, adversarial label) appear in the fooling matrices of SUAP, CW-UAP and UAP. This makes sense since, ultimately, each couple (predicted label, adversarial label) depends on the similarity between images classes and how the classifier proceeds to distinguish between the classes. Despite this resemblance, the key point here is that all three methods operate differently. Especially, by its construction L-SUAP is able to find overlapping decomposition 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 evaluated on a distinct test set. Results, averaged over multiple splits, are reported in Figure 4. Overall, we observe that increasing L improves the performance, thus

[6]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 [28].

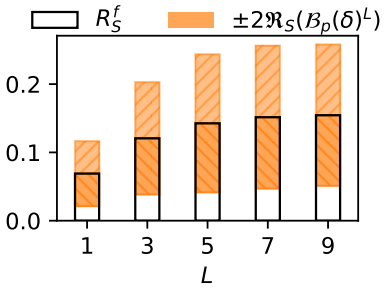


Figure 3: **Generalization bound**. The top of the orange bar, *resp.* the bottom, is the approximation of the upper, *resp.* lower, bound of R_D^f.

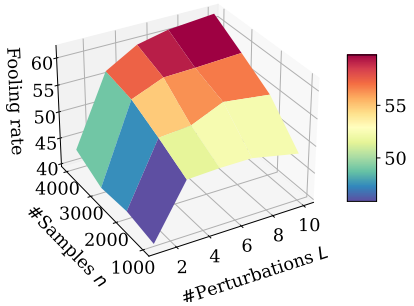


Figure 4: **Impact of parameters.** Depending on the number of CIFAR-10 samples, we report the fooling rate of ℓ_∞-based L-SUAP attacks.

confirming that having more perturbations is beneficial to attack the network f. This observation has to be contrasted with the fact that the 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 $n=1$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.*, $n=2$K).

Comparison with baselines. We report in Table 1 the performances of ℓ_∞-attacks, in terms of fooling rate. First of all, 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 our algorithmic solution which benefits from better optimization guarantees. In addition, as L grows, we observe an increase in performance of SUAP attacks, thus justifying the advantages of having more degrees of freedom. Interestingly, the SUAP attacks manage to improve upon the one-shot specific FGSM attack. However, the performances 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 2). 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 [47] models. We additionally consider two robust models from the *RobustBench* repository [14], namely r-ResNet18 [48] and r-ResNet50 [10], which are trained with some defense mechanisms against ℓ_∞-attacks of budget $\delta = 8/255$. Results are reported in Table 1. Overall, L-SUAP systematically yields a better transferability than all universal attacks, as shown by the higher fooling rates. In addition, it 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, L-SUAP shows competitive results on robust models.

5.3 ImageNet experiments

We tackle a large scale scenario made of 1K classes. Such setting is problematic for CW-UAP as computing or storing 1K perturbations exceeds most memory storage spaces: It is not studied here.

Results. We report the performances in Table 2. Once again, we observe a drastic gap of performance between UAP-PGD and 1-SUAP, confirming the superiority of the numerical solution of Algorithm 2 for $L=1$ perturbation over the standard UAP-PGD solver [49]. Overall, SUAP achieves performance of the order of magnitude as specific attacks (*e.g.*, MI-FGSM). It suggests that, for large-scale settings with numerous classes, solely a few universal perturbations are enough to attack most of images.

6 Conclusion

This paper introduces a framework for crafting semi-universal attacks. This new family of attacks is halfway between specific and universal attacks by jointly leaning multiple universal perturbations. When facing an unseen example, an adversarial example is built by selecting, in an unsupervised manner, the appropriate perturbation among 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

Table 1: **Performance and 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	SOURCE	TRANSFER			
	ResNet18	ResNet50	MobileNetv2	r-ResNet18	r-ResNet50
UAP-PGD [49]	12.53 ± 0.60	21.51 ± 0.18	39.37 ± 0.19	2.01 ± 0.01	2.54 ± 0.05
FAST-UAP [15]	11.16 ± 1.03	19.65 ± 1.22	36.51 ± 0.30	1.94 ± 0.01	2.33 ± 0.05
CW-UAP [7]	$\mathbf{13.85 \pm 0.18}$	$\mathbf{21.95 \pm 0.28}$	$\mathbf{39.62 \pm 0.33}$	$\mathbf{2.26 \pm 0.07}$	$\mathbf{2.26 \pm 0.06}$
1-SUAP	36.83 ± 0.93	27.45 ± 0.81	44.11 ± 0.35	2.27 ± 0.02	2.27 ± 0.08
3-SUAP	54.03 ± 0.54	28.49 ± 0.56	45.42 ± 0.32	2.55 ± 0.03	2.95 ± 0.08
5-SUAP	$\mathbf{55.56 \pm 0.57}$	$\mathbf{28.87 \pm 0.07}$	$\mathbf{46.09 \pm 0.10}$	$\mathbf{2.56 \pm 0.05}$	$\mathbf{3.10 \pm 0.05}$
FGSM [19]	53.82 ± 0.00	28.55 ± 0.00	38.10 ± 0.00	$\mathbf{3.02 \pm 0.00}$	3.14 ± 0.00
MI-FGSM [16]	80.76 ± 0.00	29.95 ± 0.01	35.46 ± 0.01	2.60 ± 0.00	$\mathbf{3.41 \pm 0.00}$
PGD [34]	$\mathbf{93.61 \pm 0.06}$	30.47 ± 0.12	$\mathbf{38.17 \pm 0.37}$	1.94 ± 0.05	2.53 ± 0.08
AutoAttack [13]	93.07 ± 0.00	$\mathbf{31.79 \pm 0.16}$	38.08 ± 0.27	1.91 ± 0.02	2.41 ± 0.02

Table 2: **Performance of ℓ_∞-attacks on a ResNet18 trained on ImageNet.** Bold fonts highlight the best fooling rate in universal (left), semi-universal (middle) and specific (right) attacks.

ATTACK	ResNet18
UAP-PGD [49]	$\mathbf{27.36 \pm 0.00}$
FAST-UAP [15]	23.46 ± 0.25

ATTACK	ResNet18
1-SUAP	83.17 ± 2.62
5-SUAP	$\mathbf{88.98 \pm 1.06}$
10-SUAP	87.24 ± 1.16

ATTACK	ResNet18
FGSM [19]	84.53 ± 0.05
MI-FGSM [16]	90.04 ± 0.02
PGD [34]	$\mathbf{94.99 \pm 0.06}$
AutoAttack [13]	88.23 ± 0.05

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 defense mechanisms against semi-universal attacks as well as the derivation of the associated generalization bounds. Indeed, we believe that such results open directions to improve the robustness of models against various and diverse attacks.

References

[1] Idan Attias, Aryeh Kontorovich, and Yishay Mansour. Improved generalization bounds for adversarially robust learning. *The Journal of Machine Learning Research*, 23(1):7897–7927, 2022.

[2] Hedy Attouch, Jérôme Bolte, and Benar Fux Svaiter. Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized gauss–seidel methods. *Mathematical Programming*, 137(1-2):91–129, 2011.

[3] Pranjal Awasthi, Natalie Frank, and Mehryar Mohri. Adversarial learning guarantees for linear hypotheses and neural networks. In *International Conference on Machine Learning*, 2020.

[4] Shumeet Baluja and Ian Fischer. Learning to attack: Adversarial transformation networks. *AAAI Conference on Artificial Intelligence*, 32(1), 2018.

[5] Peter L. Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *The Journal of Machine Learning Research*, 3:463–482, 2002.

[6] Peter L. Bartlett, Stéphane Boucheron, and Gábor Lugosi. Model selection and error estimation. *Machine Learning*, 48(1-3):85–113, 2002.

[7] Philipp Benz, Chaoning Zhang, Adil Karjauv, and In So Kweon. Universal adversarial training with class-wise perturbations. In *IEEE International Conference on Multimedia and Expo*, 2021.

[8] Silvia Bonettini, Ignace Loris, Federica Porta, Marco Prato, and Simone Rebegoldi. On the convergence of a linesearch based proximal-gradient method for nonconvex optimization. *Inverse Problems*, 33(5):055005, 2017.

[9] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *IEEE Symposium on Security and Privacy (SP)*, 2017.

[10] Tianlong Chen, Sijia Liu, Shiyu Chang, Yu Cheng, Lisa Amini, and Zhangyang Wang. Adversarial robustness: From self-supervised pre-training to fine-tuning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[11] Dan Claudiu Cireşan, Ueli Meier, Luca Maria Gambardella, and Jürgen Schmidhuber. Deep, Big, Simple Neural Nets for Handwritten Digit Recognition. *Neural Computation*, 22(12):3207–3220, 12 2010. ISSN 0899-7667.

[12] Patrick L Combettes and Jean-Christophe Pesquet. Lipschitz certificates for layered network structures driven by averaged activation operators. *SIAM Journal on Mathematics of Data Science*, 2(2):529–557, 2020.

[13] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International Conference on Machine Learning*, 2020.

[14] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020.

[15] Jiazhu Dai and Le Shu. Fast-UAP: An algorithm for expediting universal adversarial perturbation generation using the orientations of perturbation vectors. *Neurocomputing*, 422:109–117, 2021.

[16] Yinpeng Dong, Fangzhou Liao, Tianyu Pang, Hang Su, Jun Zhu, Xiaolin Hu, and Jianguo Li. Boosting adversarial attacks with momentum. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

[17] Gintare Karolina Dziugaite and Daniel M Roy. Data-dependent pac-bayes priors via differential privacy. In *Advances in Neural Information Processing Systems*, 2018.

[18] Chris Finlay, Aram-Alexandre Pooladian, and Adam Oberman. The logbarrier adversarial attack: making effective use of decision boundary information. In *IEEE/CVF International Conference on Computer Vision*, 2019.

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

[20] Henry Gouk, Eibe Frank, Bernhard Pfahringer, and Michael J Cree. Regularisation of neural networks by enforcing lipschitz continuity. *Machine Learning*, 110(2):393–416, 2021.

[21] Sorin Grigorescu, Bogdan Trasnea, Tiberiu Cocias, and Gigel Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3):362–386, 2020.

[22] Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. In *ICLR, Workshop Track Proceedings*, 2015.

[23] Jamie Hayes and George Danezis. Learning universal adversarial perturbations with generative models. In *IEEE Security and Privacy Workshops*, 2018.

[24] Xiaowei Huang, Daniel Kroening, Wenjie Ruan, James Sharp, Youcheng Sun, Emese Thamo, Min Wu, and Xinping Yi. A survey of safety and trustworthiness of deep neural networks: Verification, testing, adversarial attack and defence, and interpretability. *Computer Science Review*, 37:100270, 2020. ISSN 1574-0137.

[25] Sashank J. Reddi, Suvrit Sra, Barnabas Poczos, and Alexander J Smola. Proximal stochastic methods for nonsmooth nonconvex finite-sum optimization. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[26] Justin Khim and Po-Ling Loh. Adversarial risk bounds via function transformation. *arXiv preprint arXiv:1810.09519*, 2018.

[27] Valentin Khrulkov and Ivan Oseledets. Art of singular vectors and universal adversarial perturbations. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[28] Hoki Kim. Torchattacks: A pytorch repository for adversarial attacks. *arXiv preprint arXiv:2010.01950*, 2020.

[29] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

[30] Alexey Kurakin, Ian J. Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *ICLR Workshop Track Proceedings*, 2017.

[31] Cassidy Laidlaw, Sahil Singla, and Soheil Feizi. Perceptual adversarial robustness: Defense against unseen threat models. In *International Conference on Learning Representations*, 2021.

[32] Yann LeCun and Corinna Cortes. MNIST handwritten digit database, 2010.

[33] Jiadong Lin, Chuanbiao Song, Kun He, Liwei Wang, and John E. Hopcroft. Nesterov accelerated gradient and scale invariance for adversarial attacks. In *International Conference on Learning Representations*, 2020.

[34] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018.

[35] Nishant A. Mehta. Machine learning theory (csc 482a/581a) - lectures 15–18, 2021.

[36] Riccardo Miotto, Fei Wang, Shuang Wang, Xiaoqian Jiang, and Joel T Dudley. Deep learning for healthcare: review, opportunities and challenges. *Briefings in bioinformatics*, 19(6):1236–1246, 2018.

[37] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. Adaptive computation and machine learning. MIT Press, 2012.

[38] Omar Montasser, Steve Hanneke, and Nathan Srebro. Vc classes are adversarially robustly learnable, but only improperly. In *Conference on Learning Theory*, 2019.

[39] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: A simple and accurate method to fool deep neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[40] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[41] Waleed Mustafa, Yunwen Lei, and Marius Kloft. On the generalization analysis of adversarial learning. In *International Conference on Machine Learning*, 2022.

[42] Ben Nassi, Yisroel Mirsky, Dudi Nassi, Raz Ben-Netanel, Oleg Drokin, and Yuval Elovici. Phantom of the adas: Securing advanced driver-assistance systems from split-second phantom attacks. In *ACM SIGSAC Conference on Computer and Communications Security*, 2020.

[43] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z. Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *IEEE European Symposium on Security and Privacy*, 2016.

[44] Nhan H. Pham, Lam M. Nguyen, Dzung T. Phan, and Quoc Tran-Dinh. Proxsarah: An efficient algorithmic framework for stochastic composite nonconvex optimization. *Journal of Machine Learning Research*, 21(110): 1–48, 2020.

[45] Huy Phan. huyvnphan/pytorch_cifar10, January 2021.

[46] Adnan Qayyum, Muhammad Usama, Junaid Qadir, and Ala Al-Fuqaha. Securing connected amp; autonomous vehicles: Challenges posed by adversarial machine learning and the way forward. *IEEE Communications Surveys Tutorials*, 22(2):998–1026, 2020.

[47] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2018.

[48] Vikash Sehwag, Saeed Mahloujifar, Tinashe Handina, Sihui Dai, Chong Xiang, Mung Chiang, and Prateek Mittal. Robust learning meets generative models: Can proxy distributions improve adversarial robustness? In *International Conference on Learning Representations*, 2022.

[49] Ali Shafahi, Mahyar Najibi, Zheng Xu, John Dickerson, Larry S. Davis, and Tom Goldstein. Universal adversarial training. *AAAI Conference on Artificial Intelligence*, 2020.

[50] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian J. Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[51] Pedro Tabacof and Eduardo Valle. Exploring the space of adversarial images. In *2016 international joint conference on neural networks (IJCNN)*, pages 426–433. IEEE, 2016.

[52] Paul Viallard, Eric Guillaume VIDOT, Amaury Habrard, and Emilie Morvant. A pac-bayes analysis of adversarial robustness. *Advances in Neural Information Processing Systems*, 2021.

[53] Xiaosen Wang, Jiadong Lin, Han Hu, Jingdong Wang, and Kun He. Boosting adversarial transferability through enhanced momentum. *arXiv preprint arXiv:2103.10609*, 2021.

[54] Zhe Wang, Kaiyi Ji, Yi Zhou, Yingbin Liang, and Vahid Tarokh. *SpiderBoost and Momentum: Faster Stochastic Variance Reduction Algorithms*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[55] Chaowei Xiao, Bo Li, Jun yan Zhu, Warren He, Mingyan Liu, and Dawn Song. Generating adversarial examples with adversarial networks. In *International Joint Conference on Artificial Intelligence*, 2018.

[56] Yue Xing, Qifan Song, and Guang Cheng. On the algorithmic stability of adversarial training. *Advances in neural information processing systems*, 34:26523–26535, 2021.

[57] Dong Yin, Ramchandran Kannan, and Peter Bartlett. Rademacher complexity for adversarially robust generalization. In *International Conference on Machine Learning*, 2019.

[58] Jinshan Zeng, Tim Tsz-Kit Lau, Shaobo Lin, and Yuan Yao. Global convergence of block coordinate descent in deep learning. In *International Conference on Machine Learning*, 2019.

[59] Chaoning Zhang, Philipp Benz, Tooba Imtiaz, and In So Kweon. Understanding adversarial examples from the mutual influence of images and perturbations. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[60] Chaoning Zhang, Philipp Benz, Chenguo Lin, Adil Karjauv, Jing Wu, and In So Kweon. A survey on universal adversarial attack. In *International Joint Conference on Artificial Intelligence*, 2021. Survey Track.

[61] Yonggang Zhang, Xinmei Tian, Ya Li, Xinchao Wang, and Dacheng Tao. Principal component adversarial example. *IEEE Transactions on Image Processing*, 29:4804–4815, 2020.

Notations. For $x \in \mathbb{R}^P$ and $\mathcal{I} \subseteq \{1, \dots, P\}$, $x_{\mathcal{I}}$ stands for the restriction of x to the indices in $\mathcal{I}$.

A Stochastic solver

We propose an additional 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.*,

$$\mathcal{L}(\varepsilon) = \frac{1}{n}\sum_{i=1}^{n} \ell_i(\varepsilon), \quad \text{with } \ell_i(\varepsilon) = \max_{l \in \{1,\dots,L\}} H(f(x_i + \varepsilon_l), y_i).$$

Hereafter, we resort to a stochastic solver based on the well-known variance reduction techniques (see [25, 54, 44]). Since the main computational load comes from the backpropagation through the neural network, we favor the proxSAGA algorithm [25] which does not require an additional loop over multiple epochs. The corresponding algorithmic solution is summarized in Algorithm 2.

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 $\varepsilon^{(K)}$

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 3 (J. Reddi et al. [25])**.** *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:*

$$\mathbb{E}\left[\|G_\gamma(\varepsilon^{(k)})\|^2\right] \leq \frac{50\beta n^2}{5n-2}\frac{\mathcal{L}(\varepsilon^\star) - \mathcal{L}(\varepsilon^{(0)})}{K}, \tag{8}$$

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 3 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.

B About Theorem 2

The proof of Theorem 2 relies on Theorem 3.3 of Mohri et al. [37] recalled below.

Theorem 3.3 of Mohri et al. [37]. For any distribution D on $\mathcal{X} \times \mathcal{Y}$, for any set $\mathcal{G}$ of functions $g : \mathcal{X} \times \mathcal{Y} \to [0,1]$, for any $\lambda \in (0,1]$, we have

$$\mathop{\mathbb{P}}_{S\sim D^n}\left(\forall g \in \mathcal{G},\ \mathop{\mathbb{E}}_{(x,y)\sim D} g(x,y) - \frac{1}{n}\sum_{i=1}^n g(x_i,y_i) \right.$$
$$\left. \leq 2 \mathop{\mathbb{E}}_{S'\sim D^n} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim\Sigma^n}\left[\sup_{g'\in\mathcal{G}} \frac{1}{n}\sum_{i=1}^n \sigma_i g'(x_i',y_i')\right] + \sqrt{\frac{\ln\frac{1}{\lambda}}{2n}}\right) \geq 1-\lambda, \tag{9}$$

and
$$\mathop{\mathbb{P}}_{S\sim D^n}\left(\forall g \in \mathcal{G},\ \mathop{\mathbb{E}}_{(x,y)\sim D} g(x,y) - \frac{1}{n}\sum_{i=1}^n g(x_i,y_i) \right.$$
$$\left. \leq 2 \mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim\Sigma^n}\left[\sup_{g'\in\mathcal{G}} \frac{2}{n}\sum_{i=1}^n \sigma_i g'(x_i,y_i)\right] + 3\sqrt{\frac{\ln\frac{2}{\lambda}}{2n}}\right) \geq 1-\lambda. \tag{10}$$

Before proving Theorem 2 in Section B.2, we recall how we can obtain two-sided generalization bounds from Mohri et al. [37]'s Theorem 3.3.

B.1 From one-sided to two-sided generalization bounds

Mohri et al. [37]'s Theorem 3.3 brings a one-sided generalization bound, *i.e.*, an upper bound on $\mathbb{E}_{(x,y)\sim D} g(x,y) - \frac{1}{n}\sum_{i=1}^n g(x_i,y_i)$. That being said, Theorem 2 provides a two-sided bound, *i.e.*, an upper bound on the term $\left|\mathbb{E}_{(x,y)\sim D} g(x,y) - \frac{1}{n}\sum_{i=1}^n g(x_i,y_i)\right|$. To do so, one common solution is to use the union bound (*e.g.* [35]). For the sake of completeness, we state the two-sided bound associated to Theorem 3.3. of Mohri et al. [37] in the following lemma.

Lemma 1 (Two-sided generalization bounds)**.** *For any distribution D on $\mathcal{X}\times\mathcal{Y}$, for any set $\mathcal{G}$ of functions $g : \mathcal{X}\times\mathcal{Y} \to [0,1]$, for any $\lambda \in (0,1]$, we have*

$$\mathop{\mathbb{P}}_{S\sim D^n}\left(\forall g \in \mathcal{G},\ \left|\mathop{\mathbb{E}}_{(x,y)\sim D} g(x,y) - \frac{1}{n}\sum_{i=1}^n g(x_i,y_i)\right| \right.$$
$$\left. \leq 2 \mathop{\mathbb{E}}_{S'\sim D^n} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim\Sigma^n}\left[\sup_{g'\in\mathcal{G}} \frac{1}{n}\sum_{i=1}^n \sigma_i g'(x_i',y_i')\right] + \sqrt{\frac{\ln\frac{2}{\lambda}}{2n}}\right) \geq 1-\lambda, \tag{11}$$

and
$$\mathop{\mathbb{P}}_{S\sim D^n}\left(\forall g \in \mathcal{G},\ \left|\mathop{\mathbb{E}}_{(x,y)\sim D} g(x,y) - \frac{1}{n}\sum_{i=1}^n g(x_i,y_i)\right| \right.$$
$$\left. \leq 2 \mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim\Sigma^n}\left[\sup_{g'\in\mathcal{G}} \frac{1}{n}\sum_{i=1}^n \sigma_i g'(x_i,y_i)\right] + 3\sqrt{\frac{\ln\frac{4}{\lambda}}{2n}}\right) \geq 1-\lambda. \tag{12}$$

Proof. We can go through the exact same proof of Mohri et al. [37]'s Theorem 3.3 but with $\frac{1}{n}\sum_{i=1}^n g(x_i,y_i) - \mathbb{E}_{(x,y)\sim D} g(x,y)$ instead of $\mathbb{E}_{(x,y)\sim D} g(x,y) - \frac{1}{n}\sum_{i=1}^n g(x_i,y_i)$. Hence, we obtain with probability at least $1-\frac{\lambda}{2}$ over $S\sim D^n$

$$\forall g \in \mathcal{G},\quad \frac{1}{n}\sum_{i=1}^n g(x_i,y_i) - \mathop{\mathbb{E}}_{(x,y)\sim D} g(x,y)$$
$$\leq 2 \mathop{\mathbb{E}}_{S'\sim D^n} \mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim\Sigma^n}\left[\sup_{g'\in\mathcal{G}} \frac{1}{n}\sum_{i=1}^n \sigma_i g'(x_i',y_i')\right] + \sqrt{\frac{\ln\frac{2}{\lambda}}{2n}}, \tag{13}$$

and
$$\forall g \in \mathcal{G},\quad \frac{1}{n}\sum_{i=1}^n g(x_i,y_i) - \mathop{\mathbb{E}}_{(x,y)\sim D} g(x,y)$$
$$\leq 2 \mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim\Sigma^n}\left[\sup_{g'\in\mathcal{G}} \frac{1}{n}\sum_{i=1}^n \sigma_i g'(x_i,y_i)\right] + 3\sqrt{\frac{\ln\frac{4}{\lambda}}{2n}}. \tag{14}$$

Hence, by combining Equations (13) and (9) from a union bound (and with $\lambda/2$ instead of λ), we obtain Equation (11). Similarly, from Equations (14) and (10), we get Equation (12). □

B.2 Proof of Theorem 2

We are now ready to prove Theorem 2.

Theorem 2. For any distribution D on $\mathcal{X}\times\mathcal{Y}$, for any loss function $H : \mathbb{R}^c \times \mathcal{Y} \to [0,1]$, for any model $f\colon \mathbb{R}^P \to \mathbb{R}^c$, for any $L\in\mathbb{N}_+$, for any budget $\delta>0$, for any ℓ_p-norm with $p\geq 0$, for any $\lambda\in(0,1]$, we have

$$\mathop{\mathbb{P}}_{S\sim D^n}\left(\forall \boldsymbol{\varepsilon}\in\mathcal{B}_p(\delta)^L,\ \left|R_D^f(\boldsymbol{\varepsilon})-R_S^f(\boldsymbol{\varepsilon})\right|\leq 2\,\mathfrak{R}_S\left[\mathcal{B}_p(\delta)^L\right]+3\sqrt{\frac{\ln\frac{4}{\lambda}}{2n}}\right)\geq 1-\lambda, \tag{6}$$

$$\text{and}\ \mathop{\mathbb{P}}_{S\sim D^n}\left(\forall \boldsymbol{\varepsilon}\in\mathcal{B}_p(\delta)^L,\ \left|R_D^f(\boldsymbol{\varepsilon})-R_S^f(\boldsymbol{\varepsilon})\right|\leq 2\mathop{\mathbb{E}}_{S'\sim D^n}\mathfrak{R}_{S'}\left[\mathcal{B}_p(\delta)^L\right]+\sqrt{\frac{\ln\frac{2}{\lambda}}{2n}}\right)\geq 1-\lambda, \tag{15}$$

where we define the Rademacher complexity [6] of the "family of perturbations" $\mathcal{B}_p(\delta)^L$ as

$$\mathfrak{R}_S\left[\mathcal{B}_p(\delta)^L\right]=\mathop{\mathbb{E}}_{\boldsymbol{\sigma}\sim\Sigma^n}\sup_{\boldsymbol{\varepsilon}\in\mathcal{B}_p(\delta)^L}\left[\frac{1}{n}\sum_{i=1}^n\sigma_i\max_{\varepsilon_l\in\boldsymbol{\varepsilon}}H(f(x_i{+}\varepsilon_l),y_i)\right],$$

with $\boldsymbol{\sigma}=\{\sigma_i\}_{i=1}^n\sim\Sigma^n$, and Σ being the Rademacher distribution, i.e., $\Sigma(-1){=}\frac{1}{2}$ and $\Sigma(+1){=}\frac{1}{2}$.

Note that Equation (15) is not in the main paper and involves the Rademacher complexity defined by $\mathbb{E}_{S'\sim D^n}\mathfrak{R}_{S'}\left[\mathcal{B}_p(\delta)^L\right]$; the remarks in the main paper for Equation (6) also apply for Equation (15).

Proof. Given $L>0$, we define the set of function $\mathcal{G}$ by

$$\mathcal{G}=\left\{g:(x,y)\mapsto\max_{\varepsilon_l\in\boldsymbol{\varepsilon}}H'(f(x+\varepsilon_l),y)\ \middle|\ \boldsymbol{\varepsilon}=[\varepsilon_1,\ldots,\varepsilon_L]\in\mathcal{B}_p(\delta)^L\right\}.$$

By applying Lemma 1 on the set $\mathcal{G}$, we have the desired results. □

C Experimental settings

In this section, we further detail how the numerical experiments were conducted.

C.1 MNIST experiments

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 1). To this effect, we resort to the simple multi-layer perceptron from [11] 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.

C.2 CIFAR10 experiments

Data splitting and pre-processing. If not mentioned otherwise, we split CIFAR10 test set into 2K images for learning (semi-)universal perturbations and 8K independent images for evaluating the attacks.

Model to attack. We consider the trained ResNet18 model from [45] augmented with an input normalizing layer of channel-wise means $(0.4914,0.4822,0.4465)$ and channel-wise standard deviations $(0.2471,0.2435,0.2616)$.

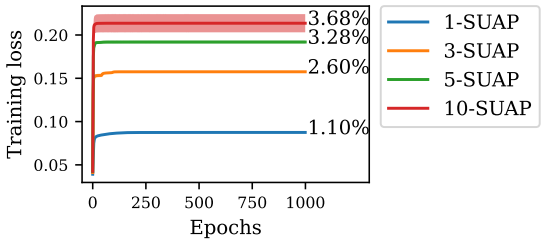


Figure 5: **Training behavior of ℓ_∞-based SUAP attacks on MNIST**. The averaged training loss is reported for 1, 3, 5 and 10 universal perturbations along with the associated test fooling rate.

C.3 ImageNet experiments

Data splitting and pre-processing. We resort to the popular ILSVRC2012 validation subset of the ImageNet dataset. 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 to attack. We analyze a pretrained ResNet18 model from the Torchvision library 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%.

L-SUAP solver. Contrary to the previous experiments, we consider the ProxSAGA solver of Algorithm 2 in order to learn the L-SUAP perturbations. The step-size and batch-size are set to $\gamma = 0.05$ and $b = 1$, respectively.

D Additional results

In the next sections, we provide complementary results on both MNIST and CIFAR10 datasets.

D.1 MNIST experiments

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 Figure 5. 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 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 Figure 5.

D.2 CIFAR10 experiments

We report in Table 3 the performance comparison of both ℓ_∞ and ℓ_2-attacks with a maximum allowable budget of $\delta = 8/255$ and $\delta = 0.5$, respectively.

Table 3: **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.

Attack	ℓ_∞-fooling rate (%)	ℓ_2-fooling rate (%)
UAP-PGD [49]	12.53 ($\pm$ 0.60)	2.67 ($\pm$ 0.21)
FAST-UAP [15]	11.16 ($\pm$ 1.03)	2.53 ($\pm$ 0.19)
CW-UAP [7]	**13.85 ($\pm$ 0.18)**	**2.77 ($\pm$ 0.09)**
1-SUAP	36.83 ($\pm$ 0.93)	3.43 ($\pm$ 0.26)
3-SUAP	54.03 ($\pm$ 0.54)	4.93 ($\pm$ 0.50)
5-SUAP	**55.56 ($\pm$ 0.57)**	**7.09 ($\pm$ 1.22)**
FGSM [19]	53.82 ($\pm$ 0.00)	N/A
MI-FGSM [16]	80.76 ($\pm$ 0.00)	N/A
PGD [34]	**93.61 ($\pm$ 0.06)**	89.23 ($\pm$ 0.02)
AutoAttack [13]	93.07 ($\pm$ 0.00)	**92.41 ($\pm$ 0.01)**