Linear Modeling of the Adversarial Noise Space

Jordan Patracone[1](✉), Lucas Anquetil[2], Yuan Liu[2], Gilles Gasso[2], and Stéphane Canu[2]

[1] Université Jean Monnet Saint-Etienne, CNRS, Institut d'Optique Graduate School, Inria, Laboratoire Hubert Curien UMR 5516, F-42023, SAINT- ETIENNE, France
[2] Normandie Univ, INSA Rouen UNIROUEN, UNIHAVRE, LITIS, Saint-Etienne-du-Rouvray, France

Abstract. Recent works have revealed the vulnerability of deep neural network (DNN) classifiers to adversarial attacks. Among such attacks, it is common to distinguish specific attacks adapted to each example from universal ones, referred to as example-agnostic. Even though specific adversarial attacks are efficient on their target DNN classifier to attack, they struggle to transfer to others. Conversely, universal adversarial attacks suffer from lower attack success. To reconcile universality and efficiency, we propose LIMANS, a model of the adversarial noise space, allowing to frame any specific adversarial perturbation as a linear combination of the universal adversarial directions. We bring in two stochastic gradient based algorithms for learning these universal directions and the associated adversarial attacks. Empirical analyses conducted with the CIFAR-10 and ImageNet datasets show that LIMANS (i) enables crafting specific and robust adversarial attacks with high probability, (ii) provides a deeper understanding of DNN flaws, and (iii) shows significant ability in transferability.

Keywords: Adversarial attacks · Dictionary learning · Manifold learning.

1 Introduction

With recent technological advances, deep neural networks (DNNs) have found widespread applications ranging from biomedical imaging to autonomous vehicles. However, DNNs have been shown to be vulnerable to adversarial attacks [31]. These attacks are slight perturbations of *clean* examples that are well-classified by the DNN, ultimately leading to misclassification. These perturbations may take the form of common corruptions (e.g., for images it can be a change in lightning conditions, colorimetry or rotations) or visually imperceptible learned adversarial noises.

There essentially exist two ways of crafting adversarial noises. The first strategy consists in finding a paired adversarial noise for each example to attack [5,25]. As such, it is deemed to be *specific* since each adversarial noise is *specifically* designed for a given example. The second strategy aims at finding a unique *universal* noise which, added to any example, is likely to fool the DNN [22]. Each strategy comes with its pros and cons. On the one hand, although specific attacks

achieve great performances on the target DNN, the learned adversarial noises do not fool other DNNs on the same examples, i.e., they transfer poorly. On the other hand, universal attacks have shown higher transferability at the expense of a weaker ability to fool the target DNN on which the universal adversarial noise is learned.

Contributions. To reconcile specificity and universality, we suggest to linearly model the adversarial noise space and propose LIMANS. For each example to attack, an adversarial noise is crafted as a linear combination of adversarial directions. While the adversarial directions are *universal*, the linear combination coefficients are *specific* to each example to perturb. In order to learn both, we propose two optimization problems associated to scalable stochastic numerical solutions. We claim that:

- Adversarial directions learned by LIMANS demonstrate transferability across diverse classifiers;
- The learned adversarial directions provide insights into DNN vulnerabilities;
- Adversarial examples generated by LIMANS exhibit great robustness to existing attack detectors.

Outline. The rest of the paper is organized as follows: the state-of-the-art researches on specific attacks, universal attacks and manifold of adversarial perturbations are presented in Section 2. The proposed framework, as well as the corresponding algorithmic solutions are detailed in Section 3. Finally, Section 4 showcases the adversarial noise model and provides experimental evaluations on both CIFAR10 and ImageNet.

2 Preliminaries and Related Works

Let us consider the task of predicting labels in $\mathcal{Y} = \{1, \dots, c\}$ associated to data from a P-dimensional feature space $\mathcal{X} \subseteq \mathbb{R}^P$. We denote $\mathcal{D}$ the corresponding distribution on $\mathcal{X} \times \mathcal{Y}$ while $\mathcal{D}_{\mathcal{X}}$ stands for the marginal distribution on $\mathcal{X}$. We stand in a white-box setting, where we assume to have fully access to a DNN $f\colon \mathbb{R}^P \to \mathbb{R}^c$ trained on this task. For any $\mathbf{x} \sim \mathcal{D}_{\mathcal{X}}$, $f(\mathbf{x}) \in \mathbb{R}^c$ represents the vector of scores indicating the likelihood of the example to belong to each of c classes. The predicted class is determined by $C_f(\mathbf{x}) = \operatorname{argmax}_{k \in \mathcal{Y}} f(\mathbf{x})_k$.

The purpose of adversarial attacks is to craft, for any $\mathbf{x} \sim \mathcal{D}_{\mathcal{X}}$, an example $\mathbf{x}' \in \mathcal{X}$, called adversarial, such that $\mathbf{x}'$ is close to $\mathbf{x}$ and $C_f(\mathbf{x}') \neq C_f(\mathbf{x})$. In other words, the adversarial example manages to change the prediction of the DNN f while being very similar to a well-classified example. Hereafter, we restrict to the peculiar case of perturbation-based attacks where $\mathbf{x}' = \mathbf{x} + \boldsymbol{\epsilon}$ for some adversarial noise $\boldsymbol{\epsilon}$. In order for the perturbation to be imperceptible, it is customary to impose some constraints of the form $\|\boldsymbol{\epsilon}\|_p \leq \delta_p$ for some ℓ_p-norm (typically ℓ_2 or ℓ_∞) and some small budget $\delta_p > 0$.

We now recall the two current paradigms: namely *specific perturbations*, also called *example-based perturbations*, and *universal perturbations* which are *example-agnostic*. Then, we motivate our approach from recent works aiming to learn the manifold of adversarial perturbations.

2.1 Specific Perturbations

Specific perturbations are learned adversarial noises individually designed to attack given examples [5,25]. More precisely, for any $\mathbf{x} \sim \mathcal{D}_{\mathcal{X}}$, the corresponding attack is of the form $\mathbf{x}' = \mathbf{x} + \boldsymbol{\epsilon}(\mathbf{x})$ where the perturbation $\boldsymbol{\epsilon}(\mathbf{x})$ depends on $\mathbf{x}$.

Among the most prominent specific attacks are one-shot gradient methods such as the Fast Gradient Sign Method (FGSM) [9], and more elaborate iterative procedures like Projected Gradient Descent (PGD) [21], DeepFool [23], and the method by Carlini and Wagner (CW) [3]. Notably, AutoAttack [6], an ensemble of diverse parameter-free attacks, has emerged as the state-of-the-art for evaluating neural network robustness.

The specificity of these attacks enables them to effectively deceive the targeted classifier. However, recent works have shown their limitations in fooling other classifiers trained on the same task. In other words, adversarial examples generated by specific attacks often demonstrate poor transferability across different classifiers. To enhance their transferability, [36] proposed incorporating input transformations, while [34] suggested stabilizing the update directions of adversarial noise to avoid poor local optima, leading to the development of VNI-FGSM and VMI-FGSM methods. Additionally, Neuron Attribution-based Attacks (NAA) improved transferability by conducting feature-level attacks [39]. Lastly, the Reverse Adversarial Perturbation (RAP) attack aimed to find adversarial attacks located on a plateau of the loss function [26].

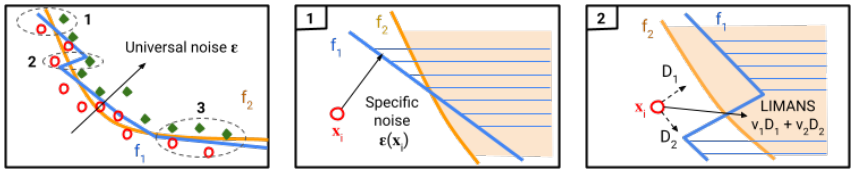
Learning these perturbations can be computationally extensive in the sense that, for any new example to attack, it requires to run an algorithmic solution once again. To alleviate this burden, universal perturbations were proposed [22].

2.2 Universal Perturbations

A universal perturbation is a single perturbation which, added to any example, manage to fool a given classifier with moderate to high probability. Formally, it amounts in finding $\boldsymbol{\epsilon}$ so that, for any $\mathbf{x} \sim \mathcal{D}_{\mathcal{X}}$, $\mathbf{x}' = \mathbf{x} + \boldsymbol{\epsilon}$ is an adversarial attack.

Among them, *universal adversarial perturbations* are meticulously crafted to exploit the most vulnerable areas of a classifier's performance, the so-called worst-case, to maximize their effectiveness across various inputs. The first universal adversarial perturbation (UAP) was introduced in [22] by hinging on the DeepFool attack. Later, different extensions of UAP were proposed: UAP-PGD [29] used a projected gradient-descent algorithm to compute the universal perturbation, CD-UAP [37] optimized an universal perturbation on a given subset of classes while the Class-Wise Universal Adversarial Perturbations (CW-UAP) [2] elaborated one universal perturbation per class.

Besides, another line of universal perturbations, called *common corruptions*, aimed a reproducing types of harm that exist in real-world imaging systems [13]. These perturbations are visually understandable modifications of the examples, readily comprehensible to humans, such as adjustments in brightness, rotation, tilt of an image or added Gaussian or impulse noise. They are average-case perturbations beyond the scope of this work. Even though common corruptions



(a) Universal noise. (b) Specific noise (zoom on region 1). (c) LIMANS noise (zoom on region 2).

Fig. 1: **Schematic representation of adversarial attacks.** Two DNN classifiers f_1 (blue) and f_2 (orange) are trained to classify red circles vs. green diamonds.

are closer to real-world harm, they need to be explicitly defined prior to any perturbation crafting, thereby limiting the modeling of the perturbation space.

Universal attacks are fast but their applicability is still far-fetched because of poor performances compared to specific attacks [4]. In an attempt to bridge the gap between adversarial perturbations and common corruptions, techniques have been explored to learn the manifold within which adversarial perturbations are embedded.

2.3 Manifold of Adversarial Perturbations

Learning and understanding the manifold of adversarial perturbations is crucial for comprehending the reasons for the existence and transferability of adversarial attacks, as well as devising effective defense mechanisms against them.

The study done in [32] highlighted that the space of the adversarial noise paired to an input example is a large continuous region. Then, [33] discovered that the decision boundaries of different classifiers are closed and proposed to establish a transferable subspace of the adversarial space across different classifiers. However, the subspace was inferred based on the adversarial noise generated by the FGSM method, significantly compromising its accuracy. Moreover, constraining the hypothesis of transferability solely based on the dimension of this space restricted its efficacy, particularly concerning CNN classifiers. In addition, certain studies have investigated the overall structure of classifiers to elucidate the emergence of adversarial attacks. For instance, [8] demonstrated that the decision boundaries of a classifier reside close to examples and remain flat in most directions. More recently, [7] theoretically studied the sufficient conditions on the existence of effective low-dimensional adversarial perturbations while [18] claimed that adversarial noise is caused by gradient leakage and the adversarial directions are perpendicular to the classifier boundaries. To learn this manifold, it has been proposed to use the singular vectors of some adversarial perturbations [14] or principal component analysis [40], to train a specific neural network as a generative network [1,12] or with generative adversarial networks [35].

Fig. 1 reports a schematic representation of adversarial attacks in the context of a binary classification task (red circles vs. green diamonds). In particular, Fig. 1a

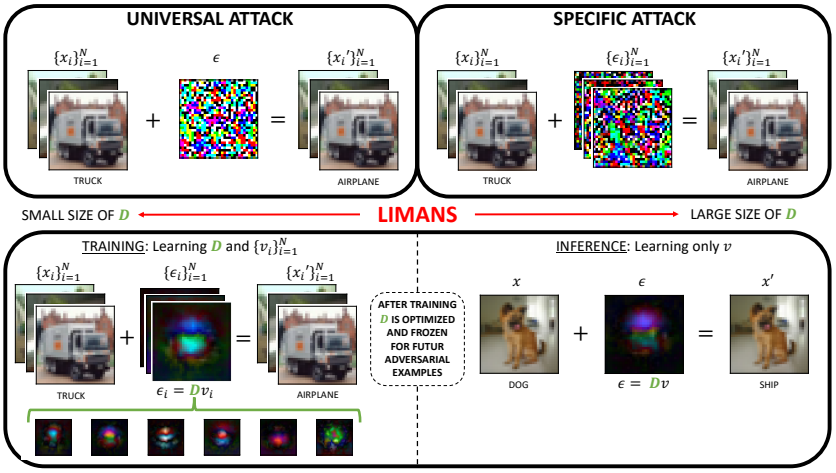


Fig. 2: **High-level overview of LIMANS.** During training, the directions D of the linear adversarial noise space are learned. During inference, the coding vector v is crafted to attack any example to fool a given classifier.

illustrates how data points are located in the vicinity of the decision boundaries of classifiers f_1 (blue) and f_2 (orange) trained on this task. Hence, adding a small perturbation, learned on f_1 can easily make them misclassified. Attacks can also transfer to f_2 when there is little bias between the two classifiers, as it is the case in the Region 3. However, in practical scenarios, the situations depicted in Region 1 and 2 are more prevalent: adversarial examples tend to deviate beyond the decision boundary of f_1 without necessarily crossing the boundary shared by f_1 and f_2. Overall, universal perturbations (see $\boldsymbol{\epsilon}$ in Fig. 1a) can achieve superior transferability by averaging directions across the entire dataset, at the expense of lower attack success rate. Conversely, specific perturbations (see $\boldsymbol{\epsilon}(\mathbf{x}_i)$ in Fig. 1b) show great attack performance on the target f_1 but fail to transfer to f_2.

In order to bridge the gap between specific attacks and universal attacks, we propose to linearly model the adversarial noise space (LIMANS). The underlying idea is to learn a set of adversarial directions (see $\{D_1, D_2\}$ in Fig. 1c). The spanned space enables the construction of specific adversarial examples by linearly combining the adversarial directions. In essence, this learned space exhibits transferability across different classifiers, leading to a higher attack success rate.

3 Linear Modeling of the Adversarial Noise Space

In this section, we introduce the proposed framework LIMANS, sketched in Fig. 2, for modeling the adversarial noise space. Then, we propose two relaxations along with their algorithmic solutions.

3.1 Problem Formulation

The originality of the present paper is to express adversarial perturbations using a linear model. More formally, for any $\mathbf{x} \sim \mathcal{D}_{\mathcal{X}}$, the corresponding adversarial attack is written as $\mathbf{x}' = \mathbf{x} + \boldsymbol{\epsilon}(\mathbf{x})$ where the adversarial noise is decomposed as $\boldsymbol{\epsilon}(\mathbf{x}) = D\mathbf{v}(\mathbf{x})$. On the one hand, $D = [D_1, \ldots, D_M] \in \mathbb{R}^{\mathrm{P} \times \mathrm{M}}$ is a dictionary made of $\mathrm{M} \in \mathbb{N}_+$ normalized adversarial directions (also called noise atoms), i.e., $\|D_j\|_p = 1$ for every $j \in \{1, \ldots, M\}$. On the other hand, $\mathbf{v}(\mathbf{x}) \in \mathbb{R}^{\mathrm{M}}$ plays the role of a coding vector. While the dictionary D is shared across all attacks, $\mathbf{v}(\mathbf{x})$ allows combining components of D in order to individually tailor perturbations for each example $\mathbf{x}$ to attack. In other words, D is universal while $\mathbf{v}(\mathbf{x})$ is example-specific. In the remaining of the paper, we simply write $\mathbf{v}(\mathbf{x})$ as $\mathbf{v}$ for the ease of reading.

We advocate to choose a number of atoms $\mathrm{M} \ll \mathrm{P}$ so that the linear adversarial noise space spanned by the atoms, i.e., $\mathcal{A} = \{\sum_{j=1}^{M} v_j D_j \mid v_j \in \mathbb{R}\}$, is low dimensional. In order to learn the directions $\{D_j\}$ of $\mathcal{A}$, we suggest maximizing the chances of fooling the classifier f on some training set $\mathcal{T} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{\mathrm{N}}$, that is the *fooling rate* (also known as *attack success rate*).

$$\frac{1}{N} \sum_{i=1}^{\mathrm{N}} \mathbb{1}_{\{C_f(\mathbf{x}^{(i)\prime}) \neq C_f(\mathbf{x}^{(i)})\}} \,. \tag{1}$$

This gives rise to the following optimization problem.

Problem 1 (Training). Given a classifier f and a training set $\mathcal{T} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{\mathrm{N}}$ where each $(\mathbf{x}^{(i)}, y^{(i)}) \sim \mathcal{D}$, solve

$$\begin{aligned} \underset{\substack{D=[D_1,\ldots,D_{\mathrm{M}}]\in\mathbb{R}^{\mathrm{P}\times\mathrm{M}} \\ V=[\mathbf{v}^{(1)},\ldots\mathbf{v}^{(\mathrm{N})}]\in\mathbb{R}^{\mathrm{M}\times\mathrm{N}}}}{\text{maximize}} \quad & \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \mathbb{1}_{\{C_f(\mathbf{x}^{(i)\prime}) \neq C_f(\mathbf{x}^{(i)})\}}, \\ s.t. \quad & \begin{cases} \mathbf{x}^{(i)\prime} = \mathbf{x}^{(i)} + D\mathbf{v}^{(i)} \in \mathcal{X} & ,(\forall i \in \{1, \ldots, \mathrm{N}\}), \\ ||D\mathbf{v}^{(i)}||_p \leq \delta_p & ,(\forall i \in \{1, \ldots, \mathrm{N}\}), \\ \|D_j\|_p = 1 & ,(\forall j \in \{1, \ldots, \mathrm{M}\}), \end{cases} \end{aligned} \tag{2}$$

where $\mathbb{1}_{\mathcal{S}}$ denotes the indicator function of the set S.

The number of atoms M acts as a trade-off quantifying the level of similarity between all adversarial perturbations.

When $\mathrm{M} = 1$, each perturbation is of the form $\boldsymbol{\epsilon}(\mathbf{x}) = v_1 D_1$, so they all vary by a multiplicative constant $v_1 \in \mathbb{R}$. Actually, it is more likely that all differ by their sign due to the constraint $||D\mathbf{v}^{(i)}||_p \leq \delta_p$ in Eq. (2) and the fact that greater perturbations (in norms) are more amenable to fool the classifier. Hence, we are in an under-fitting regime where the adversarial perturbations somewhat boil down to a universal perturbation which cannot manage to fool all examples of the training set but achieve great transferability across different DNN classifiers.

On the other hand of the spectrum, when $\mathrm{M} = \mathrm{N}$, there are enough degrees of freedom to specifically tailor one direction D_j for each example of the training set to attack. Therefore, we stand in an over-fitting regime, reminiscent of specific attacks, where the learned adversarial noise space may transfer poorly.

Our proposed framework LIMANS bridges the gap between the two ends of the spectrum by allowing the attacker to choose the number of atoms M better suited for the task at hand. Once the adversarial noise space $\mathcal{A}$ is learned, it can be used to attack unseen examples on any classifier f (potentially different than the one used to learn $\mathcal{A}$) as follows.

Problem 2 (Inference). Given $\mathcal{A}$ spanned by directions $D \in \mathbb{R}^{\mathrm{P} \times \mathrm{M}}$, a classifier f, and an example $\mathbf{x} \sim \mathcal{D}_{\mathcal{X}}$. The associated attack reads $\mathbf{x}' = \mathbf{x} + D\mathbf{v}$ where $\mathbf{v}$ solves

$$\underset{\mathbf{v} \in \mathbb{R}^{\mathrm{M}}}{\text{maximize}} \ \mathbb{1}_{\{C_f(\mathbf{x}') \neq C_f(\mathbf{x})\}}, \quad \text{s.t.} \quad \begin{cases} \mathbf{x}' = \mathbf{x} + D\mathbf{v} \in \mathcal{X}, \\ ||D\mathbf{v}||_p \leq \delta_p. \end{cases} \tag{3}$$

Problem 2 bears similarities with Problem 1 when the optimization over D is dropped. In that case the optimization over $V = [\mathbf{v}^{(1)}, \dots \mathbf{v}^{(\mathrm{N})}] \in \mathbb{R}^{\mathrm{M} \times \mathrm{N}}$ becomes separable into N distinct problems: one for every $\mathbf{v}^{(i)}$. Problem 2 is one instance of them.

In the next section, we suggest two algorithmic solutions for solving Problem 1 and, by extension, also Problem 2.

3.2 Relaxations and Algorithmic Solutions

Solving Problem 1 is a challenge for three main reasons: i) the indicator function $\mathbb{1}_{\mathcal{S}}$, ii) the presence of the DNN f, and iii) the dictionary-based formulation of the adversarial noise.

First, it is customary to replace the fooling rate by a surrogate loss function (to minimize) in order to avoid algorithmic concerns regarding the nonconvexity and nonsmoothness of the indicator function. Hereafter, we resort to one of them, namely the logit margin loss of [3] that we recall below. Given a DNN f, an example $\mathbf{x} \sim \mathcal{D}_{\mathcal{X}}$ classified as $C_f(\mathbf{x}) = y$, it reads

$$\mathcal{L}_\gamma(f(\mathbf{x}'), f(\mathbf{x})) = \max\big(-\gamma, \ f_y(\mathbf{x}') - \max_{k \neq y} f_k(\mathbf{x}')\big), \tag{4}$$

where $\gamma > 0$ is a hyperparameter.

Second, the optimization problem is nonconvex because of the highly non-linear mapping f. Therefore, studying the Lipschitz regularity of the DNN f plays a pivotal role in the derivation of convergence guarantees of first order algorithms toward critical points of Problem 1. Since, we would like our approach to be general enough for any f, we do not pursue the estimation of an upper-bound on the Lipschitz constant of peculiar f, as conducted in [10].

Third, Problem 1 is difficult due to nonconvexity inherent to the dictionary-based formulation of the adversarial noise. Although classical dictionary learning problems are nonconvex, they are usually solved by alternating the optimization over D and V since each alternating problem is convex. However, here this no longer the case because of the presence of f being a highly non-linear mapping. A direct optimization scheme over (D, V), in the spirit of the nonconvex proximal splitting framework of [30], later applied in the context of classical dictionary

learning [27], was first considered but it that happened to be less effective than expected in practice. Since we are only interested in finding a good approximate critical point, we suggest two alternative problems which turned out to be more fruitful from the practitioner's point of view.

Regularized-LIMANS The first problem is designed by adding a soft penalization of the constraint $||D\mathbf{v}^{(i)}||_p \leq \delta_p$ and relaxing the conditions $\mathbf{x}^{(i)'} \in \mathcal{X}$ from Problem 1.

Problem 3. Given a classifier f and a set $\mathcal{T} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{\mathrm{N}}$ where each $(\mathbf{x}^{(i)}, y^{(i)}) \sim \mathcal{D}$, solve

$$\begin{aligned} \underset{\substack{D=[D_1,\dots,D_{\mathrm{M}}]\in\mathbb{R}^{\mathrm{P}\times\mathrm{M}} \\ V=[\mathbf{v}^{(1)},\dots\mathbf{v}^{(\mathrm{N})}]\in\mathbb{R}^{\mathrm{M}\times\mathrm{N}}}}{\text{mimize}} \quad & \tfrac{1}{\mathrm{N}}\textstyle\sum_{i=1}^{\mathrm{N}} \mathcal{L}_\gamma(f(\mathbf{x}^{(i)} + D\mathbf{v}^{(i)}), f(\mathbf{x}^{(i)})) + \lambda h_{(\delta_p,p)}(D, \mathbf{v}^{(i)}), \\ s.t. \quad & \|D_j\|_p = 1 \quad , (\forall j \in \{1,\dots,\mathrm{M}\}), \end{aligned} \tag{5}$$

where $\lambda > 0$ is a regularization parameter and $h_{(\delta_p,p)}$ is a penalty defined as

$$h_{(\delta_p,p)} \colon (D, \mathbf{v}) \mapsto \max(||D\mathbf{v}||_p - \delta_p, 0). \tag{6}$$

Algorithm 1 summarizes the optimization scheme of Regularized-LIMANS. The parameters (D, V) are optimized stochastically, making them suitable for large-scale datasets. More specifically, D is updated using a projected gradient descent to ensure that the constraints $||D_j||_p = 1, \forall j$ are satisfied.

In order to select the hyperparameter $\lambda > 0$, a cross-validation is conducted to ensure the generalization of the model. In practice, for the selected value of λ, it might happened that the constraint on $\|D\mathbf{v}\|_p$ is slightly violated. To ensure the respect of the constraint, a post-processing is performed. Details are provided in the supplementary material.

Remark 1. At inference-time, adversarial attacks are crafted as $\mathbf{x}' = \mathrm{Proj}_{\mathcal{X}}(\mathbf{x} + D\mathbf{v})$, where $\mathrm{Proj}_{\mathcal{X}}$ denotes the projection operator onto $\mathcal{X}$, and $\mathbf{v}$ is learned by solving Problem 3 with D fixed.

Simple-LIMANS In the second problem relaxation, the constraints $\|D\mathbf{v}^{(i)}\|_p \leq \delta_p$ and $\mathbf{x}^{(i)'} \in \mathcal{X}$ from Problem 1 are directly integrated in the objective function as follows.

Problem 4. Given a classifier f and a set $\mathcal{T} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{\mathrm{N}}$ where each $(\mathbf{x}^{(i)}, y^{(i)}) \sim \mathcal{D}$, solve

$$\underset{\substack{D=[D_1,\dots,D_{\mathrm{M}}]\in\mathbb{R}^{\mathrm{P}\times\mathrm{M}} \\ V=[\mathbf{v}^{(1)},\dots\mathbf{v}^{(\mathrm{N})}]\in\mathbb{R}^{\mathrm{M}\times\mathrm{N}}}}{\text{mimize}} \quad \frac{1}{\mathrm{N}}\sum_{i=1}^{\mathrm{N}} \mathcal{L}_\gamma\left(f\Big(\mathrm{Proj}_{\mathcal{X}}(\mathbf{x}^{(i)} + \delta_p \frac{D\mathbf{v}^{(i)}}{\|D\mathbf{v}^{(i)}\|_p})\Big), f(\mathbf{x}^{(i)})\right). \tag{7}$$

Algorithm 1 Regularized-LIMANS

Require: Classifier f; Learning rate ρ; Dataset $\mathcal{T}$; Budget $\delta_p > 0$; Optimizer Optim; Batch size B; Parameter $\lambda > 0$

$D = \mathcal{N}(0, \mathbb{1}_{\mathrm{M} \times \mathrm{P}})$; $V = \mathcal{N}(0, \mathbb{1}_{\mathrm{P} \times \mathrm{M}})$
for k = 0 to MAXEPOCH **do**
  loss = 0
  for $(\mathbf{x}^{(i)}, y^{(i)}) \subset \mathcal{T}$ **do**
    $\mathbf{x}^{(i)'} = \mathbf{x}^{(i)} + D\mathbf{v}^{(i)}$
    $\hat{y}_{adv} = f(\mathbf{x}^{(i)'})$; $\hat{y} = f(\mathbf{x}^{(i)})$
    $\text{loss}_i = \mathcal{L}_\gamma(\hat{y}_{adv}, \hat{y}) + \lambda h_{(\delta_p, p)}(D, \mathbf{v}^{(i)})$
    loss = loss+loss_i
    if modulo$(i) = B$ **then**
      $D \leftarrow \text{Optim}(\nabla_D \text{loss})(\text{Update})$
      $V \leftarrow \text{Optim}(\nabla_V \text{loss})(\text{Update})$
      $D = \text{Proj}_{\{D \mid \|D\|_p = 1\}}(D)$
      loss = 0
    end if
  end for
end for
$D\mathbf{v}^{(i)} \leftarrow \text{Proj}_{\{D\mathbf{v} \mid \|D\mathbf{v}\|_p \leq \delta\}}(D\mathbf{v}^{(i)})$
$\mathbf{x}^{(i)'} \leftarrow \text{Proj}_{\mathcal{X}}(\mathbf{x}^{(i)} + D\mathbf{v}^{(i)})$
return $\{\mathbf{x}^{(i)'}\}_{i=1}^{\mathrm{N}}$, (D, V)

Algorithm 2 Simple-LIMANS

Require: Classifier f; Learning rate ρ; Dataset $\mathcal{T}$; Budget $\delta_p > 0$; Optimizer Optim; Batch size B

$D = \mathcal{N}(0, \mathbb{1}_{\mathrm{M} \times \mathrm{P}})$; $V = \mathcal{N}(0, \mathbb{1}_{\mathrm{P} \times \mathrm{M}})$
for k = 0 to MAXEPOCH **do**
  loss = 0
  for $(\mathbf{x}^{(i)}, y^{(i)}) \subset \mathcal{T}$ **do**
    $\text{noise}^{(i)} = D\mathbf{v}^{(i)}$
    $\mathbf{x}^{(i)'} = \text{proj}_{\mathcal{X}}(\mathbf{x}^{(i)} + \frac{\delta_p \text{noise}^{(i)}}{\|\text{noise}^{(i)}\|_p})$
    $\hat{y}_{adv} = f(\mathbf{x}^{(i)'})$; $\hat{y} = f(\mathbf{x}^{(i)})$
    $\text{loss}_i = \mathcal{L}_\gamma(\hat{y}_{adv}, \hat{y})$
    loss = loss + loss_i
    if modulo$(i) = B$ **then**
      $D \leftarrow \text{Optim}(\nabla_D \text{loss})(\text{Update})$
      $V \leftarrow \text{Optim}(\nabla_V \text{loss})(\text{Update})$
      loss = 0
    end if
  end for
end for
$V \leftarrow [||D_{\bullet j}||_p V_{j\bullet}]\ \forall j \in \{1, \ldots, \mathrm{M}\}$
$D \leftarrow \text{Proj}_{\{D \mid \|D\|_p = 1\}}(D)$
return $\{\mathbf{x}^{(i)'}\}_{i=1}^{\mathrm{N}}$, (D, V)

By reparametrizing each adversarial attack $\mathbf{x}^{(i)'} = \text{Proj}_{\mathcal{X}}(\mathbf{x}^{(i)} + \delta_p \frac{D\mathbf{v}^{(i)}}{\|D\mathbf{v}^{(i)}\|_p})$, it guarantees that the ℓ_p-norm of the adversarial noise is at most δ_p and that $\mathbf{x}^{(i)'} \in \mathcal{X}$, by construction. The proposed algorithmic solution, reported in Algorithm 2, proves to be computationally efficient since it does not require additional hyperparameter tuning, contrary to Regularized-LIMANS.

Remark 2. One can determine normalized directions D by solving Problem 4 and, at termination, post-process D by normalizing each of the components.

4 Experiments

This section showcases a numerical assessment of LIMANS' efficiency. We present the experimental settings, offer insights, and compare its performance against baseline attacks. Finally, we measure the transferability of the learned adversarial noise space across different classifiers.

4.1 Experimental Settings

Experiments are conducted on two datasets: CIFAR-10[17] and ImageNet [16]. As suggested in [38], we discard the training set, used to train the classifiers, and learn attacks only on the validation set. The latter is split it into three parts, the first for learning the adversarial directions, the second for tuning hyperparameters, and the last one for testing.

CIFAR-10 Experiments. We use 8000, 1000 and 1000 examples for training, validating and testing, respectively. Attacks are evaluated on four vanilla classifiers (MobileNet-V2, ResNet50, DenseNet121, VGG11) and two robust ones (ResNet-18, WideResNet-34-10) obtained from the RobustBench repository[3]. Experiments are conducted on a GPU Nvidia RTX 2080.

ImageNet Experiments. We split into 10000 training examples, 2000 validation examples, and 5000 test examples. Attacks are designed on four vanilla classifiers (ResNet-18, MobileNet-V2, DenseNet121, VGG11) and two robust classifiers[3] (R-r18 and R-wrn-50-2, equivalent to ResNet-18 and WideResNet-50-2). Experiments are conducted on 4 GPU Volta V100-SXM2-32GB.

Baseline Attacks. Specific attacks are crafted by resorting to the TorchAttacks library [15], while universal attacks are implemented using publicly available resources. Additional details concerning LIMANS' attack as well as the hyperparameter selection are provided in the supplementary material. Attack budgets δ_p are the same as the one used in RobustBench, namely $\delta_\infty = 8/255$ and $\delta_2 = 0.5$ for CIFAR-10 and $\delta_\infty = 4/255$ for ImageNet experiments.

4.2 Insights and Attack Performance

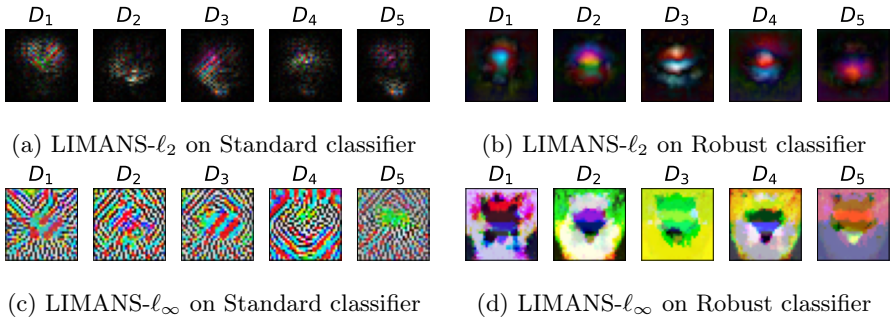
In this section, we provide an analysis of the Simple-LIMANS attack (see Algorithm 2) on CIFAR-10. We consider the pre-trained VGG11 with batch normalization [24] and the robust ResNet-18 [28] classifier, referred to as the *Standard Classifier* and *Robust Classifier* respectively.

Visualisation of Adversarial Directions Having a linear model of the adversarial noise space allows for visual inspection of the adversarial directions, which is advantageous for understanding the attack behavior. Fig. 3 shows how the learned directions for $\mathrm{M} = 5$ exhibit patterns which vary depending on the classifier and the ℓ_p-norm. Overall, they spotlight recurring patterns in classification such as edges and corners for the ℓ_∞ atoms and local spots in the images for ℓ_2 atoms. In particular, the directions of LIMANS-ℓ_2 on the robust classifier are reminiscent of Fourier modes. Similar conclusions, detailed in the supplementary material, can be drawn on the MNIST dataset.

Impact of the Number of Directions Fig. 4 reports the test fooling rate of ℓ_2-based attacks[4]. The performance of LIMANS are displayed as functions of the

[3] https://robustbench.github.io/

[4] A similar experiment for ℓ_∞-based attacks is reported in the supplementary material.



(a) LIMANS-ℓ_2 on Standard classifier (b) LIMANS-ℓ_2 on Robust classifier

(c) LIMANS-ℓ_∞ on Standard classifier (d) LIMANS-ℓ_∞ on Robust classifier

Fig. 3: Visualization of the learned universal adversarial directions (atoms of the dictionary D) when M = 5, on CIFAR-10 and corresponding to the classifier (left) VGG11 and (right) robust ResNet-18. All atoms have been rescaled for display.

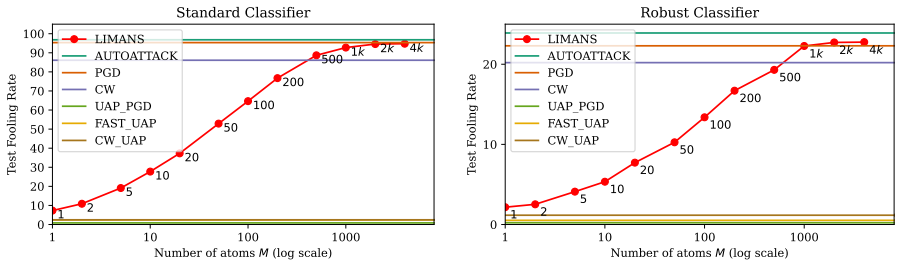


Fig. 4: **Impact of the number of directions.** Test fooling rate of adversarial attacks under the ℓ_2 norm constraint ($\delta_2 = 0.5$) on CIFAR-10 test data when fixing a number of atoms M, associated to a standard VGG11 classifier (left) and robust ResNet-18 classifier (right).

number of directions M. Interestingly, results show that LIMANS consistently outperforms all universal attacks, even with just a single direction, i.e., M = 1. In light of the discussion in Section 3.1, this indicates that allowing each individual perturbation to tailor its own multiplicative constant in front of the universal perturbation D_1 already permits to yield more efficient attacks. As M increases, LIMANS progressively narrows the performance gap with specific attacks. Between M = 500 and M = 4000, LIMANS achieves performance akin to state-of-the-art specific attacks. It's noteworthy that all specific attack methods necessitate learning individual perturbations of size $32 \times 32 \times 3 = 3072$ (for RGB CIFAR-10 images) while LIMANS attacks only require to optimize over a coding vector $\mathbf{v}$ of dimension M.

Attack Robustness Attacks, while effective in fooling classifiers, can also be detected by trained detectors. To gauge the robustness of attacks, we employ

Table 1: Robustness performance of the LIMANS ℓ_∞-attack ($\delta_\infty = 8/255$) in terms of RAUD on the CIFAR-10 test data and against the attack detectors plugged in both standard classifier (S.C.) and robust classifier (R.C.). **The smaller the RAUD, the more robust the adversarial attack is**. The best performance is marked in black bold font.

Detectors d	d_{FGSM}		d_{PGD}		$d_{Autoattack}$		$d_{LIMANS_{10}}$	
Classifiers f	S.C.	R.C	S.C.	R.C	S.C.	R.C	S.C.	R.C
SA	91.1	85.1	91.1	85.1	91.1	85.1	91.1	85.1
FGSM	*91.1*	*85.1*	*91.1*	*85.1*	*91.1*	*85.1*	83.4	79.5
PGD	90.6	84.9	*91.1*	85.0	*91.1*	*85.1*	55.9	73.7
Autoattack	89.9	84.6	90.9	85.0	*91.1*	85.0	52.7	71.5
$LIMANS_{10}$	75.7	81.0	81.0	80.8	81.6	81.0	88.9	79.6
$LIMANS_{500}$	17.5	71.5	25.6	72.2	31.8	74.2	26.6	69.4
$LIMANS_{1000}$	15.9	70.1	26.1	70.9	32.1	72.5	**21.7**	68.7
$LIMANS_{4000}$	**15.6**	**69.6**	**23.7**	**70.4**	**28.2**	**72.6**	31.1	**68.4**

the *Robust Accuracy Under Defense* (RAUD) [20], quantifying the percentage of successful attacks detected. Lower RAUD values indicate greater attack robustness. We provide the RAUD for both LIMANS and specific ℓ_∞-attacks in Table 1, with ℓ_2-attack results available in the supplementary material. Notably, LIMANS attacks consistently evade detection, outperforming specific attacks even at $M = 10$ and exhibiting robustness from $M \geq 500$.

4.3 Transferability of the Adversarial Noise Space

In [33], it's suggested that decision boundaries of classifiers trained on the same dataset are close. This implies that if LIMANS learns an adversarial space aligned with directions perpendicular to these boundaries, it can transfer effectively between classifiers. Here, we empirically support this claim using Regularized-LIMANS (see Algorithm 1). All attacks are ℓ_∞-bounded perturbations.

CIFAR-10 Experiment. We select $M = 150$ and report the test fooling rates in Table 2. The results validate our intuition: the adversarial directions derived from both ResNet50 and VGG exhibit superior transfer performance compared to state-of-the-art attacks across standard classifiers. Moreover, the efficacy of the LIMANS attack on a target classifier isn't contingent on its performance on the source classifier, but rather on the characteristics of the target classifier. For instance, the fooling rate of the LIMANS specific attack on ResNet is 91.3%. However, when employing the learned LIMANS model to generate adversarial perturbations to deceive MobileNet, its performance improves further to 96.0%. This discrepancy arises because MobileNet is simpler and more susceptible to attacks. Ultimately, through comparative analysis, we deduce that a model trained on a robust classifier is more readily transferable to other classifiers.

Table 2: Performance of ℓ_∞-attacks on CIFAR-10 ($\delta_\infty = 8/255$), in terms of FR, where the left column lists the source classifiers and the first line presents the target classifiers. The best results of transferability are marked in red bold style. That of the specific attacks are shown in blue bold style.

		MobileNet	ResNet50	DenseNet	VGG	R-r18	R-wrn-34-10
ResNet50	AutoAttack	63.3	**100**	54.6	25.1	1.2	2.4
	VNI-FGSM	78.3	95.9	80.3	57.2	2.7	2.1
	NAA	50.7	64.7	22.9	18.4	1.4	2.1
	RAP	49.0	75.1	52.5	35.4	1.6	2.8
	Ours	**96.0**	91.3	**81.8**	**82.1**	**11.7**	**13.2**
VGG	AutoAttack	62.5	43.0	44.0	**100**	2.7	2.7
	VNI-FGSM	69.3	62.6	61.4	96.5	3.0	2.6
	NAA	42.3	14.5	1.8	71.6	1.6	1.2
	RAP	46.5	39.5	40.9	73.8.	3.3	3.4
	Ours	**97.4**	**87.5**	**81.5**	91.0	**11.5**	**12.6**

ImageNet Experiment. Due to memory limitations inherent to the large images sizes, we only consider M = 100 which may be far from the actual dimension of the underlying adversarial noise space. However, it still offers evidence of transferability, as shown in Table 3. Nevertheless, in the case of robust classifiers, the decision boundaries tend to be more intricate. This complexity results in the inability to close the performance gap with AutoAttack for M = 100. Despite this challenge, the LIMANS attack still demonstrates strong performance overall.

5 Conclusion

This work introduced LIMANS, a linear model of the adversarial noise space designed to bridge the gap between universal and specific adversarial attacks. This is achieved by framing attacks as specific linear combinations of universal adversarial directions. Additionally, we proposed two implementations: Simple-LIMANS, a parameter-free algorithm, and Regularized-LIMANS, which exhibits greater efficiency when its regularization parameter is appropriately tuned.

Empirical findings demonstrated that adversarial examples generated by LIMANS exhibit enhanced resilience against adversarial example detectors. Furthermore, the study confirmed the transferability of the adversarial noise space across various classifiers. Future research works involve extending its scope to encompass for generating black-box attacks and learning adversarial directions jointly across multiple DNNs. Such a step holds promise for enhancing its universality, efficiency, and alignment with the genuine adversarial threats encountered in practical applications.

Table 3: Performance of ℓ_∞-attacks on ImageNet ($\delta_\infty = 4/255$), in terms of FR, where the left column lists the source classifiers and the first line presents the target classifiers. The best results of transferability are marked in red bold font. Those of the specific attacks are shown in blue bold font.

		MobileNet	ResNet18	DenseNet	VGG	R-r18	R-50-2
ResNet18	AutoAttack	40.30	**100**	35.76	34.90	1.80	1.34
	VNI-FGSM	56.74	99.98	51.40	**51.42**	2.84	2.04
	NAA	22.54	97.94	14.84	19.30	2.12	1.20
	RAP	53.36	96.74	51.30	50.60	3.80	3.14
	LIMANS	**59.16**	59.16	**53.14**	48.28	**10.48**	**6.62**
VGG	AutoAttack	47.94	40.06	32.62	**100**	2.34	1.42
	VNI-FGSM	**57.98**	53.96	42.88	99.84	2.76	2.24
	NAA	19.62	14.92	12.18	79.96	2.18	1.40
	RAP	53.14	53.12	42.68	95.68	3.48	2.84
	LIMANS	57.68	**54.14**	**50.04**	51.62	**10.68**	**6.24**
R-r18	AutoAttack	13.70	15.8	10.82	14.60	**71.74**	10.78
	VNI-FGSM	16.14	17.66	12.48	16.08	63.22	11.74
	NAA	11.46	10.86	9.34	11.42	21.48	4.90
	RAP	11.32	10.80	8.16	10.32	45.80	7.94
	LIMANS	**37.14**	**33.2**	**33.76**	**29.90**	29.84	**12.94**
R-50-2	AutoAttack	20.14	22.76	17.36	19.44	15.42	**59.02**
	VNI-FGSM	23.88	26.22	19.68	23.28	18.00	52.28
	NAA	14.08	13.12	10.20	14.04	9.82	12.58
	RAP	13.82	14.06	10.52	13.50	15.54	34.10
	LIMANS	**42.18**	**42.50**	**42.46**	**34.22**	**23.70**	18.02

References

1. Baluja, S., Fischer, I.: Learning to attack: Adversarial transformation networks. Proceedings of the AAAI Conference on Artificial Intelligence **32**(1) (Apr 2018)
2. Benz, P., Zhang, C., Karjauv, A., Kweon, I.S.: Universal adversarial training with class-wise perturbations. In: 2021 IEEE International Conference on Multimedia and Expo (ICME). pp. 1–6. IEEE (2021)
3. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 39–57 (2017)
4. Chaubey, A., Agrawal, N., Barnwal, K., Guliani, K.K., Mehta, P.: Universal adversarial perturbations: A survey. arXiv preprint arXiv:2005.08087 (2020)
5. Croce, F., Andriushchenko, M., Sehwag, V., Debenedetti, E., Flammarion, N., Chiang, M., Mittal, P., Hein, M.: Robustbench: a standardized adversarial robustness benchmark. In: NeurIPS, Datasets and Benchmarks Track (Round 2) (2021)
6. Croce, F., Hein, M.: Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In: ICML. vol. 119, pp. 2206–2216 (2020)
7. Dohmatob, E., Guo, C., Goibert, M.: Origins of low-dimensional adversarial perturbations. In: Ruiz, F., Dy, J., van de Meent, J.W. (eds.) Proceedings of The 26th

International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 206, pp. 9221–9237. PMLR (25–27 Apr 2023)
8. Fawzi, A., Moosavi-Dezfooli, S.M., Frossard, P., Soatto, S.: Empirical study of the topology and geometry of deep networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3762–3770 (2018)
9. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9 (2015)
10. Gupta, K., Kaakai, F., Pesquet-Popescu, B., Pesquet, J.C., Malliaros, F.D.: Multivariate lipschitz analysis of the stability of neural networks. Frontiers in Signal Processing **2** (2022)
11. Harder, P., Pfreundt, F.J., Keuper, M., Keuper, J.: Spectraldefense: Detecting adversarial attacks on cnns in the fourier domain. In: 2021 International Joint Conference on Neural Networks (IJCNN). pp. 1–8. IEEE (2021)
12. Hayes, J., Danezis, G.: Learning universal adversarial perturbations with generative models. In: 2018 IEEE Security and Privacy Workshops (SPW). pp. 43–49. IEEE (2018)
13. Hendrycks, D., Dietterich, T.G.: Benchmarking neural network robustness to common corruptions and perturbations. In: ICLR (2019)
14. Khrulkov, V., Oseledets, I.: Art of singular vectors and universal adversarial perturbations. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8562–8570 (2018)
15. Kim, H.: Torchattacks: A pytorch repository for adversarial attacks. arXiv preprint arXiv:2010.01950 (2020)
16. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Communications of the ACM **60**(6), 84–90 (2017)
17. Krizhevsky, A., et al.: Learning multiple layers of features from tiny images (2009)
18. Li, Y., Cheng, S., Su, H., Zhu, J.: Defense against adversarial attacks via controlling gradient leaking on embedded manifolds. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16. pp. 753–769. Springer (2020)
19. Lorenz, P., Strassel, D., Keuper, M., Keuper, J.: Is robustbench/autoattack a suitable benchmark for adversarial robustness? (2021)
20. Lorenz, P., Strassel, D., Keuper, M., Keuper, J.: Is autoattack/autobench a suitable benchmark for adversarial robustness? In: The AAAI-22 Workshop on Adversarial Machine Learning and Beyond (2022)
21. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: 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)
22. Moosavi-Dezfooli, S.M., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1765–1773 (2017)
23. Moosavi-Dezfooli, S., Fawzi, A., Frossard, P.: 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. pp. 2574–2582. IEEE Computer Society (2016)
24. Phan, H.: Pytorch_cifar10 (2021). https://doi.org/10.5281/zenodo.4431043
25. Qian, Z., Huang, K., Wang, Q.F., Zhang, X.Y.: A survey of robust adversarial training in pattern recognition: Fundamental, theory, and methodologies. Pattern Recognition **131**, 108889 (2022)

26. Qin, Z., Fan, Y., Liu, Y., Shen, L., Zhang, Y., Wang, J., Wu, B.: Boosting the transferability of adversarial attacks with reverse adversarial perturbation. In: NeurIPS (2022)
27. Rakotomamonjy, A.: Direct optimization of the dictionary learning problem. IEEE Transactions on Signal Processing **61**(22), 5495–5506 (2013)
28. Sehwag, V., Mahloujifar, S., Handina, T., Dai, S., Xiang, C., Chiang, M., Mittal, P.: Robust learning meets generative models: Can proxy distributions improve adversarial robustness? In: ICLR (2022)
29. Shafahi, A., Najibi, M., Xu, Z., Dickerson, J., Davis, L.S., Goldstein, T.: Universal adversarial training. Proceedings of the AAAI Conference on Artificial Intelligence **34**(04), 5636–5643 (Apr 2020)
30. Sra, S.: Scalable nonconvex inexact proximal splitting. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems. vol. 25. Curran Associates, Inc. (2012)
31. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I.J., Fergus, R.: Intriguing properties of neural networks. In: Bengio, Y., LeCun, Y. (eds.) 2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16 (2014)
32. Tabacof, P., Valle, E.: Exploring the space of adversarial images. In: 2016 international joint conference on neural networks (IJCNN). pp. 426–433. IEEE (2016)
33. Tramèr, F., Papernot, N., Goodfellow, I.J., Boneh, D., Mcdaniel, P.: The space of transferable adversarial examples. ArXiv **abs/1704.03453** (2017)
34. Wang, X., He, K.: Enhancing the transferability of adversarial attacks through variance tuning. In: Proceedings of the IEEE/CVF CVPR. pp. 1924–1933 (2021)
35. Xiao, C., Li, B., Zhu, J.Y., He, W., Liu, M., Song, D.: Generating adversarial examples with adversarial networks. In: IJCNN. p. 3905–3911. IJCAI'18 (2018)
36. Xie, C., Zhang, Z., Zhou, Y., Bai, S., Wang, J., Ren, Z., Yuille, A.L.: Improving transferability of adversarial examples with input diversity. In: Proceedings of the IEEE/CVF CVPR. pp. 2730–2739 (2019)
37. Zhang, C., Benz, P., Imtiaz, T., Kweon, I.S.: CD-UAP: Class discriminative universal adversarial perturbation. Proceedings of the AAAI Conference on Artificial Intelligence **34**(04), 6754–6761 (Apr 2020)
38. Zhang, C., Benz, P., Lin, C., Karjauv, A., Wu, J., Kweon, I.S.: A survey on universal adversarial attack. In: IJCAI-21. pp. 4687–4694 (8 2021), survey Track
39. Zhang, J., Wu, W., Huang, J.t., Huang, Y., Wang, W., Su, Y., Lyu, M.R.: Improving adversarial transferability via neuron attribution-based attacks. In: IEEE/CVF CVPR. pp. 14993–15002 (2022)
40. Zhang, Y., Tian, X., Li, Y., Wang, X., Tao, D.: Principal component adversarial example. IEEE Transactions on Image Processing **29**, 4804–4815 (2020)

A Adversarial Detector

Regarding the design of an adversarial examples' detector, we followed the guidelines proposed by [19] and [11]. In these two works, authors propose to use a random forest binary classifier with at least 100 trees, in the Fourier domain either of the input images or of the Fourier Features of Feature-Maps from the images. Authors showed that by using either one of the inputs, the random forest binary classifier is able to discriminate with high precision adversarial example computed from state-of-the-art specific adversarial attacks on several complex datasets such as CIFAR-10, CIFAR-100, ImageNet or Celeba. In order to lower as much as possible the bias introduced by the detector, we chose to use a random forest binary classifier with 300 trees in the Fourier domain of the input images. Our choice has been confirmed by extensive experiments reported in Table 4.

Table 4: Confusion matrices of the detectors used to compute the RAUD table from the main paper. All detectors have been trained on the same training dataset as the one used in the training of LIMANS and the displayed values of computed over the validation dataset, such that fair performances are considered. TN: True Negative, FP: False Positive, FN: False Negative, TP: True Positive.

Detectors d		d_{FGSM}	d_{PGD}	$d_{Autoattack}$	$d_{LIMANS_{10}}$
Confusion Matrix	TN FP	863 57	814 106	790 130	846 74
	FN TP	0 920	0 920	0 920	95 825
Accuracy		96.9 %	94.2 %	92.9 %	90.8 %
Precision		94.1 %	89.6 %	87.6 %	91.7 %

Indeed, Table 4 illustrates the confusion matrix of the detectors involved in the RAUD table of the main paper. These detectors were trained on the same dataset as LIMANS and the values displayed are computed over the validation dataset. The RAUD metric for various adversarial attacks using the detector is performed on the unseen test dataset, ensuring fairness.

We empirically observe highly effective detectors that discriminate real images from adversarial ones with high accuracy, minimizing False-Positive and False-Negative values. These robust performances instill confidence in utilizing these detectors for the RAUD metric, crucial for evaluating the harmfulness and transferability of adversarial attacks.

B Experimental Details

This section presents the details of the different implementations used in both the Simple-LIMANS and Regularized-LIMANS' experiments.

B.1 Simple-LIMANS

Modeling Details Simple-LIMANS algorithms operates a relaxation on the definition of the adversarial noise. Given the original example $\mathbf{x}^{(i)}$, Simple-LIMANS consider its adversarial noise as $\boldsymbol{\epsilon}^{(i)} = D\mathbf{v}^{(i)} + \mathbf{b}$ the product of the universal adversarial noise model D with its corresponding coding vector $\mathbf{v}^{(i)}$, to which is added the offset b universal to all LIMANS adversarial noises.

Algorithmic Parameters The learning rate is managed automatically using a cheduler to reduce the learning rate when the loss plateaus. Initially, a high learning rate is used to avoid poor local minima caused by random initialization. As optimization progresses, the learning rate decreases, allowing the parameters to converge to a better minimum. The parameters of the scheduler are: patience=40, factor=0.1, and threshold=0.1.

All Simple-LIMANS experiments were optimized using the Adam optimizer. Among several optimizers tested, Adam was found to be the most effective, as illustrated in Figure 5. While similar performance levels can be achieved with alternative optimizers like RMSProp, the Adam optimizer or one of its variants emerges as a compelling optimization choice overall.

As depicted in Figure 5, the batch size B is not a highly sensitive hyper-parameter when using Simple-LIMANS. We observed that varying batch sizes often led to similar performance outcomes, with differences primarily affecting computational time. Accordingly, we set the batch size B to 256 during training and 64 during inference in our experiments.

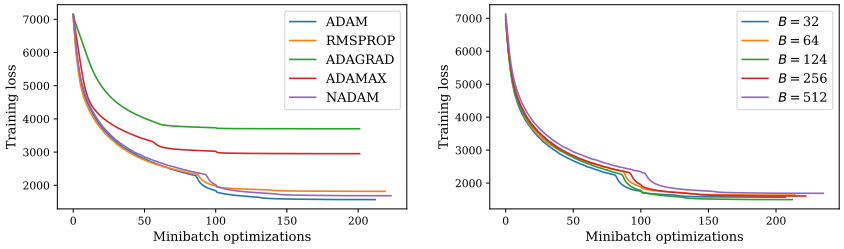


Fig. 5: Evolution of LIMANS_{10} training loss according to (left) different optimizers and (right) different batch sizes B using the Simple-LIMANS algorithm on the standard classifier under the ℓ_∞ norm.

B.2 Regularized-LIMANS

Hyperparameters Figure 6 illustrates the influence of the hyperparameter λ and the number of atoms M on attack performance. Increasing M appropriately leads to improved performance, consistent with the findings presented in the

paper. In our experiments, to strike a balance between attack performance and memory constraints, we choose M = 150 for CIFAR-10 and M = 100 for ImageNet. Moreover, for $p = \infty$, setting $\lambda = 1$ yields optimal performance, while $\lambda = 0.1$ is suitable for $p = 2$. This conclusion holds true when extrapolated to other classifiers, as depicted in Table 5.

Algorithmic Parameters The learning rate ρ is fixed to 0.001. In the training phase, we set the number of iterations to MAXEPOCH=1000 while, in the validation and the test phases, MAXEPOCH is set 150 when $p = 2$ and 300 when $p = \infty$.

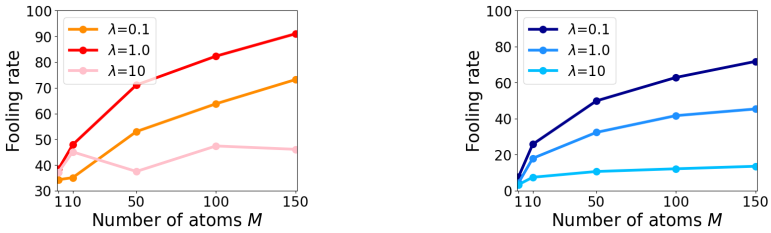


Fig. 6: Performance of LIMANS (left) ℓ_∞-attacks and (right) ℓ_2-attacks on CIFAR-10 when attacking VGG, under different settings of hyperparameter in Regularized LIMANS λ, and different number of atoms M.

Table 5: Performance of LIMANS attacks on CIFAR-10, in terms of FR, when the number of atoms M = 150. The best results are marked in red bold style.

	ℓ_∞-attack			ℓ_2-attack		
λ \ Classifiers	VGG	MobileNet	R-R18	VGG	MobileNet	R-R18
0.1	73.2	85.3	15.8	**71.7**	**95.4**	**17.6**
1.0	**91.0**	**97.3**	**25.3**	45.4	49.6	12.8
10	46.1	91.7	19.9	13.6	24.4	10.3

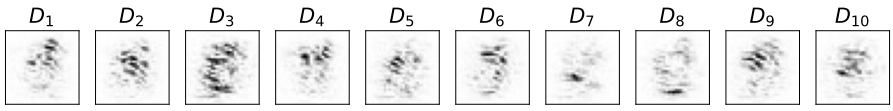
Classifiers To assess the transferability of the adversarial noise space, we conducted experiments on CIFAR-10 using five vanilla DNNs (MobileNet, Inception, ResNet, DenseNet, VGG) and two robust DNNs (Robust ResNet18, Robust WideResNet-34-10). Their respective accuracies are 94.00% (MobileNet), 94.10% (Inception), 93.2% (ResNet), 92.8% (DenseNet), 92.1% (VGG), 82.3% (Robust ResNet18), and 85.1% (Robust WideResNet-34-10). For ImageNet, Inception

was excluded due to its different input size. All classifiers were obtained from the PyTorch model zoo and achieved accuracies of 70.95% (MobileNet), 68.20% (ResNet), 73.65% (DenseNet), 67.60% (VGG), 51.25% (Robust ResNet18), and 66.55% (Robust WideResNet-50-2). In our experiments, each DNN served as the source classifier to learn the adversarial noise space, and then crafted adversarial perturbations in this learned space to deceive the other classifiers, which acted as the target classifiers.

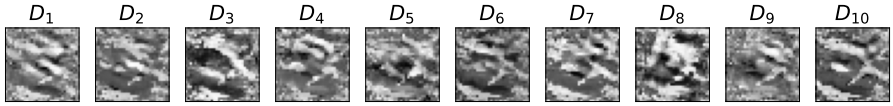
C Additional Results: "Insights and Attack Performance"

C.1 Visualisation of Adversarial Directions

By linearly modeling adversarial noise space, we find that LIMANS' directions hold crucial information for deceiving classifiers, akin to capturing semantic essence within the classification space. Additional experiments on the MNIST dataset further illustrate this observation. We present in Figure 7 the $M = 10$ adversarial directions learned to fool a LeNet classifier achieving more than 98.8% of test accuracy.



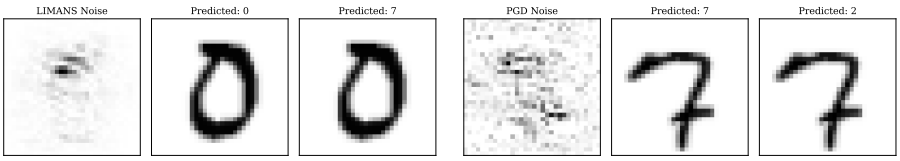
(a) LIMANS-ℓ_2



(b) LIMANS-ℓ_∞

Fig. 7: Visualization of the $M = 10$ universal adversarial directions on MNIST.

In addition to these visually interesting universal directions, we report in Fig.8 how LIMANS operates to specifically craft adversarial perturbations. LIMANS generates significantly more compelling adversarial perturbations compared to state-of-the-art specific adversarial perturbations, which are essentially random. LIMANS strategically targets the most sensitive areas to deceive the classifiers, rendering its adversarial attack far more realistic than state-of-the-art specific adversarial attacks.



(a) LIMANS adversarial perturbation and example

(b) PGD adversarial perturbation and example

Fig. 8: Examples of ℓ_2 adversarial perturbations produced by LIMANS and PGD on the MNIST dataset for a LeNet classifier achieving more than 98.8% of test accuracy.

C.2 Impact of the Number of Directions

We present here the performance of the LIMANS attacks as the number of atoms M increases from 1 to 4000. It's worth noting that both the performance on training data, depicted in Figure 10, and on test data, illustrated in Figure 9, indicate that LIMANS attacks are capable of bridging the gap between universal attacks and specific attacks.

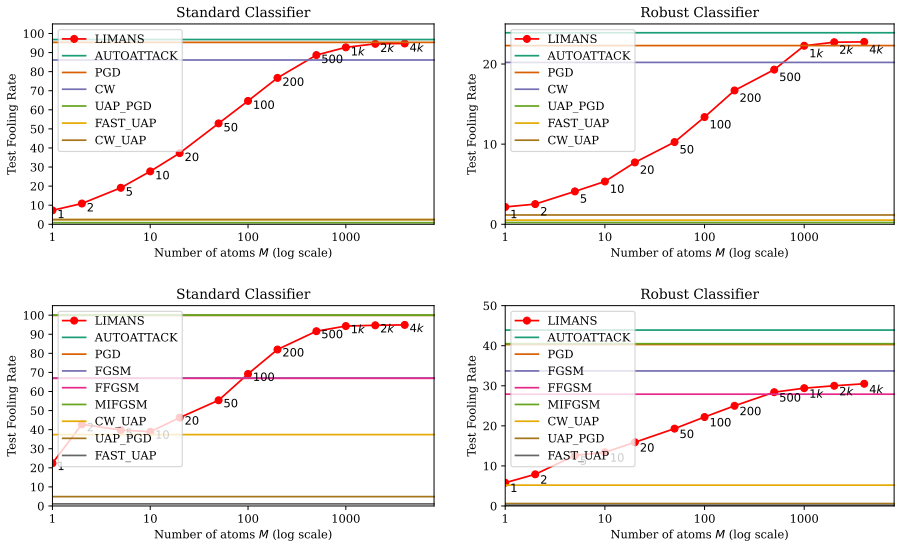


Fig. 9: Evolution of LIMANS' test fooling rate according to the number of atoms M. Both specific and universal adversarial attack baselines are shown. The problem is solved under the ℓ_2 norm constraint (first line) and ℓ_∞ norm constraint (second line) on the standard classifier (left figure) and the robust classifier (right figure) on CIFAR-10 using Simple-LIMANS. On average over 5 random seeds the fooling rates vary around 0.4% of FR for the standard model and around 0.1% of FR for the robust model, errorbars are plotted but so tiny, are invisible.

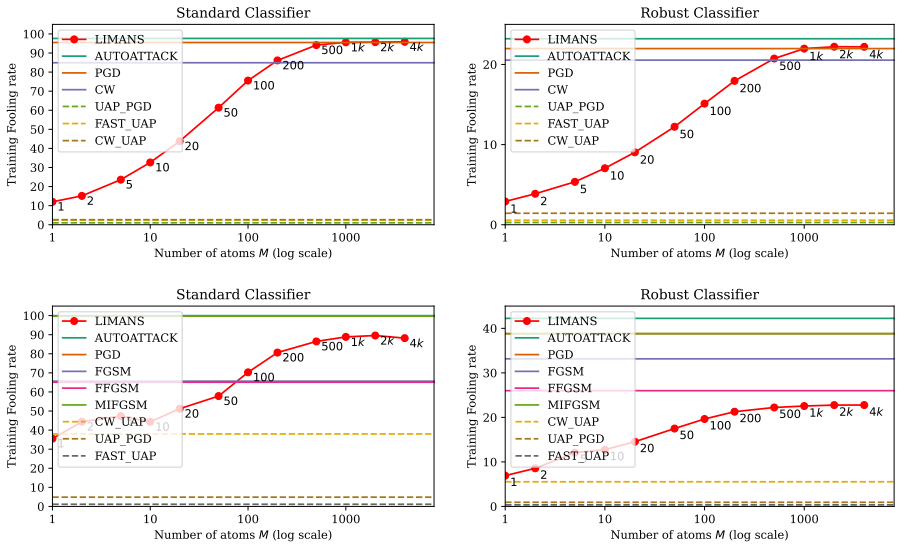


Fig. 10: Evolution of LIMANS' training fooling rate according to the number of atoms M. Both specific and universal adversarial attack baselines are shown. The problem is solved under the ℓ_2 norm constraint (first line) and ℓ_∞ norm constraint (second line) on the standard classifier (left figure) and the robust classifier (right figure) on CIFAR-10 using Simple-LIMANS. On average over 5 random seeds the fooling rates vary around 0.4% of FR for the standard model and around 0.1% of FR for the robust model, error bars are plotted but so tiny, are invisible.

C.3 Robustness of the Attack

Table 6 and Table 7 display the RAUD of Simple-LIMANS and the specific baselines for both the ℓ_∞ and ℓ_2 norms, considering various adversarial example detectors d. The RAUD values are accompanied by their standard deviation over 5 different random seeds. These performances are reported for both the standard and robust classifier on the CIFAR-10 dataset.

Table 6: Robustness performance of the LIMANS ℓ_∞-attack ($\delta_\infty = 8/255$) in terms of RAUD on the CIFAR-10 test data and against the attack detectors plugged in both standard classifier (S.C.) and robust classifier (R.C.). **The smaller the RAUD, the more robust the adversarial attack is**. The best performances are marked in bold red.

Detectors d	d_{PGD}		$d_{Autoattack}$		$d_{LIMANS_{10}}$	
Classifiers f	S.C.	R.C	S.C.	R.C	S.C.	R.C
SA	91.1	85.1	91.1	85.1	91.1	85.1
FGSM	*91.1* ± 0.0	*85.1* ± 0.0	*91.1* ± 0	*85.1* ± 0.0	89.3 ± 0.0	79.5 ± 0.0
PGD	*91.0* ± 0.0	84.9 ± 0.1	*91.0* ± 0.0	*85.0* ± 0.0	80.7 ± 0.5	73.3 ± 0.3
Autoattack	91.0 ± 0.0	85.0 ± 0.0	*91.1* ± 0.0	85.0 ± 0.0	78.2 ± 0.3	71.5 ± 0.3
$LIMANS_{10}$	78.3 ± 2.4	79.6 ± 0.0	81.7 ± 2.0	8.0 ± 0.0	86.4 ± 1.6	79.5 ± 0.1
$LIMANS_{500}$	26.3 ± 0.3	71.3 ± 0.1	32.1 ± 0.4	73.2 ± 0.2	36.5 ± 3.1	69.4 ± 0.2
$LIMANS_{1000}$	24.7 ± 0.9	70.4 ± 0.2	31.6 ± 1.1	72.4 ± 0.1	36.9 ± 4.6	68.5 ± 0.2
$LIMANS_{4000}$	**23.7 ± 0.5**	**69.8 ± 0.0**	**30.8 ± 0.9**	**72.9 ± 0.3**	**35.6 ± 2.2**	**68.2 ± 0.1**

Table 7: Robustness performance of the LIMANS ℓ_2-attack ($\delta_2 = 0.5$) in terms of RAUD on the CIFAR-10 test data and against the attack detectors plugged in both standard classifier (S.C.) and robust classifier (R.C.). **The smaller the RAUD, the more robust the adversarial attack is**. The best performance is marked in red bold font.

Detectors d	d_{PGD}		$d_{Autoattack}$		$d_{LIMANS_{10}}$	
Classifiers f	S.C.	R.C	S.C.	R.C	S.C.	R.C
SA	91.1	85.1	91.1	85.1	91.1	85.1
PGD	64.0 ± 1.2	82.4 ± 0.0	65.3 ± 0.6	81.8 ± 0.1	42.9 ± 0.6	80.1 ± 0.0
Autoattack	63.1 ± 1.1	82.19 ± 0.1	65.8 ± 0.6	**81.1 ± 0.2**	**42.1 ± 0.3**	**79.4 ± 0.0**
$LIMANS_{10}$	83.5 ± 0.5	87.4 ± 0.1	82.9 ± 0.4	88.1 ± 0.0	86.9 ± 0.9	87.6 ± 0.1
$LIMANS_{500}$	62.9 ± 0.2	84.0 ± 0.4	64.7 ± 1.0	83.8 ± 0.3	51.7 ± 1.3	81.7 ± 0.1
$LIMANS_{1000}$	63.7 ± 0.6	82.7 ± 0.3	63.6 ± 0.9	82.6 ± 0.2	46.8 ± 0.7	80.1 ± 0.1
$LIMANS_{4000}$	**62.6 ± 1.2**	**82.16 ± 0.2**	**62.2 ± 0.6**	82.6 ± 0.3	46.9 ± 0.2	80.0 ± 0.2

D Additional results: "Transferability of the Adversarial Noise Space"

In this section, we present additional findings concerning the transferability of the adversarial noise space. Consistent with our paper, the learned space under LIMANS ℓ_∞-attack demonstrates robust transferability across different classifiers, as corroborated by the results in Table 9 (part 1) and Table 8 (part 2), as well as Table 11. Additionally, the adversarial noise space acquired using the LIMANS ℓ_2-attack also exhibits transferability, as depicted in Table 10.

Table 8: Transferability performance of the LIMANS ℓ_∞-attacks on CIFAR-10 ($\epsilon = 8/255$), in terms of fooling rates (FR). The best transferable results are marked in red bold font and the best specific attacking results are marked in black bold font: Part 2.

		MobileNet	Inception	ResNet50	DenseNet	VGG	R-r18	R-wrn-34-10
R-wrn-34-10	AutoAttack	17.6	17.9	18.0	18.5	19.1	**27.7**	**39.4**
	UAP	12.4	9.9	6.6	5.3	4.4	1.7	1.3
	UAPPGD	27.0	21.5	12.9	11.7	13.1	2.5	2.6
	TI-FGSM	7.9	6.6	6.8	7.7	8.3	17.2	21.2
	VMI-FGSM	26.8	25.2	26.1	24.3	26	26.3	32.3
	VNI-FGSM	30.0	27.7	28.6	26.3	27.4	26.7	32.1
	NAA	13.7	11.3	10.1	10.5	11.4	9.7	15.5
	RAP	11.5	9.4	7.7	7.7	8.4	1.5	19.5
	Ours	**84.9**	**76.6**	**72.8**	**68.9**	**64.0**	23.2	21.6

Table 9: Transferability performance of the LIMANS ℓ_∞-attacks on CIFAR-10 ($\epsilon = 8/255$), in terms of fooling rates (FR). The best transferable results are marked in red bold font and the best specific attacking results are marked in black bold font: Part 1.

		MobileNet	Inception	ResNet50	DenseNet	VGG	R-r18	R-wrn-34-10
MobileNet	AutoAttack	**100**	87.1	37.2	32.8	22.4	1.7	1.5
	UAP	47.3	36.1	9.3	8.3	8.7	1.3	0.8
	UAPPGD	86.2	56.1	20.5	19	21.3	1.6	1.5
	TI-FGSM	87.2	25.2	25.9	29.1	16.1	2.1	2.0
	VMI-FGSM	**100**	87.3	53.1	49.8	38.9	2.2	2.5
	VNI-FGSM	**100**	88.1	54.8	53.1	40.7	2.4	2.9
	NAA	72.2	25.3	6.8	5.9	6.4	1.3	1.0
	RAP	86.7	60.3	38.5	35.7	25.8	3.0	1.6
	Ours	97.3	**92.2**	**73.6**	**66.4**	**67.7**	**10.2**	**10.7**
Inception	AutoAttack	54.7	**100**	14.7	12.9	12.0	1.2	1.1
	UAP	39.2	32.9	9.3	9.7	9.4	1.5	1.1
	UAPPGD	73.9	75.5	26.3	23.8	27.3	2.2	1.5
	TI-FGSM	19.7	60.2	19.8	21	11.4	2.2	1.5
	VMI-FGSM	69.8	86.1	40.8	38.3	31.3	2.6	1.6
	VNI-FGSM	75.5	89.5	44.4	42.4	36.3	3.2	2.3
	NAA	38.7	70.5	8.4	8.1	9.2	1.1	1.5
	RAP	61.9	90.2	42.0	41.7	30.3	2.3	2.7
	Ours	**98**	95.1	**79.6**	**73.9**	**75.8**	**10.7**	**10.8**
ResNet50	AutoAttack	63.3	52.6	**100**	54.6	25.1	1.2	2.4
	UAP	31.4	23.6	12.1	12.5	11.2	1.3	1.7
	UAPPGD	63.3	49.4	39.4	35.1	26.1	1.1	2.3
	TI-FGSM	18.4	17.1	74.0	38.5	20.4	2.2	3.0
	VMI-FGSM	74.9	75.3	96.0	78.1	53.5	2.1	3.2
	VNI-FGSM	78.3	76.9	95.9	80.3	57.2	2.7	2.1
	NAA	50.7	38.6	64.7	22.9	18.4	1.4	2.1
	RAP	49.0	45.7	75.1	52.5	35.4	1.6	2.8
	Ours	**96.0**	**92.9**	91.3	**81.8**	**82.1**	**11.7**	**13.2**
DenseNet	AutoAttack	56.9	51.6	48.8	**100**	21.8	2.1	2.0
	UAP	27.6	20.6	10.6	12.8	11.4	1.6	1.4
	UAPPGD	61.1	49.9	29.3	47.4	27.3	2.7	2.1
	TI-FGSM	17.4	15.8	26.3	65.2	17	2.9	2.3
	VMI-FGSM	73.7	71.8	77.2	93.1	47.9	3.3	3.7
	VNI-FGSM	78.1	76.2	79.5	94.0	53.3	3.5	4.2
	NAA	37.2	31.1	23.7	74.9	12.5	1.2	1.5
	RAP	47.8	43.5	48.7	75.9	35.6	3.2	3.5
	Ours	**96.7**	**93.5**	**88.4**	85.5	**82.7**	**12.3**	**13.4**
VGG	AutoAttack	62.5	58.0	43.0	44.0	**100**	2.7	2.7
	UAP	22.0	18.4	10.2	10.2	10.0	1.1	1.3
	UAPPGD	63.6	55.9	27.6	29.4	41.9	3.1	2.1
	TI-FGSM	19.7	16.7	25.6	27.6	74.4	3.7	2.2
	VMI-FGSM	66.2	64.2	57.5	56.9	96.5	3.0	2.6
	VNI-FGSM	69.3	68	62.6	61.4	96.5	3.0	2.6
	NAA	42.3	38.3	14.5	1.8	71.6	1.6	1.2
	RAP	46.5	44.5	39.5	40.9	73.8.	3.3	3.4
	Ours	**97.4**	**95.1**	**87.5**	**81.5**	91.0	**11.5**	**12.6**
R-r18	AutoAttack	17.5	15.7	17.2	15.6	17.5	**44.3**	**23.4**
	UAP	14.5	9.5	7.1	6.4	7.6	1.9	2.6
	UAPPGD	18.6	13.3	9.7	8.6	10.5	3.1	3.5
	TI-FGSM	8.4	5.5	8.2	7.8	8.6	26.2	13.1
	VMI-FGSM	24	22.9	24.2	21.9	24.8	38	22.7
	VNI-FGSM	27.1	23.1	25.4	23.8	25.6	38.1	22.9
	NAA	16.2	11.5	11.2	10.4	10.4	18.7	7.2
	RAP	10.9	8.4	7.9	8.9	9.7	23.8	12.2
	Ours	**81.3**	**73.2**	**71.7**	**68.3**	**61.7**	25.3	21.6

Table 10: Transferability performance of the LIMANS ℓ_2-attacks on CIFAR-10 ($\epsilon = 0.5$), in terms of fooling rates (FR). The best transferable results are marked in red bold font and the best specific attacking results are marked in black bold font.

		MobileNet	Inception	ResNet50	DenseNet	VGG	R-r18	R-wrn-34-10
MobileNet	AutoAttack	**100**	50.20	14.20	13.30	8.20	0.90	0.50
	UAP	7.50	5.20	3.00	2.50	2.40	0.30	0.40
	UAPPGD	37.90	15.20	2.00	1.10	0.90	0.30	0.20
	CW	97.50	11.00	4.20	3.20	2.40	0.30	0.00
	RAP	67.30	11.20	4.20	3.90	2.60	0.50	0.10
	Ours	95.40	**91.50**	**61.70**	**59.30**	**51.50**	**4.60**	**5.00**
Inception	AutoAttack	32.80	**100**	6.60	7.90	5.50	0.50	0.50
	UAP	9.80	7.50	2.50	3.50	2.90	0.20	0.10
	UAPPGD	26.90	16.70	1.30	2.30	2.00	0.30	0.10
	CW	16.30	82.80	5.00	5.20	3.70	0.30	0.00
	RAP	13.60	43.50	3.50	3.60	2.70	0.40	0.30
	Ours	**94.60**	94.10	**64.30**	**63.90**	**57.20**	**5.10**	**5.20**
ResNet50	AutoAttack	31.00	23.40	**99.70**	26.10	10.00	1.20	0.70
	UAP	5.10	3.80	2.40	1.90	2.80	0.50	0.30
	UAPPGD	4.10	3.20	2.20	2.30	2.20	0.40	0.20
	CW	13.50	9.80	82.40	13.10	6.10	0.50	0.40
	RAP	10.20	8.60	33.00	8.60	4.90	0.40	0.30
	Ours	**92.60**	**87.50**	78.10	**71.70**	**61.70**	**7.90**	**7.50**
DenseNet	AutoAttack	32.60	25.20	27.30	**99.50**	10.20	0.50	0.50
	UAP	4.90	3.70	2.60	3.30	1.90	0.20	0.20
	UAPPGD	5.00	4.50	3.20	3.70	2.10	0.20	0.20
	CW	14.60	13.70	14.80	80.00	6.40	0.40	0.30
	RAP	8.00	7.30	7.70	29.00	4.50	0.30	0.30
	Ours	**91.10**	**87.60**	**74.00**	74.10	**62.70**	**8.40**	**7.70**
VGG	AutoAttack	32.00	28.20	19.50	21.10	**98.90**	0.80	0.60
	UAP	4.70	3.80	2.20	2.70	2.00	0.50	0.40
	UAPPGD	4.80	5.60	2.00	2.70	2.80	0.40	0.40
	CW	10.00	8.20	5.70	7.40	79.10	0.60	0.30
	RAP	8.80	7.10	5.20	6.50	32.10	0.30	0.50
	Ours	**94.20**	**89.00**	**74.80**	**71.00**	71.70	**8.00**	**7.10**
R-r18	AutoAttack	6.70	8.30	8.00	8.50	9.60	**24.60**	11.00
	UAP	3.40	3.10	2.50	2.30	1.80	0.50	0.40
	UAPPGD	2.70	2.10	2.00	1.70	2.60	0.30	0.10
	CW	9.50	11.60	10.60	10.00	11.60	22.90	3.90
	RAP	8.70	7.70	7.60	8.10	9.60	10.70	4.50
	Ours	**58.70**	**53.80**	**50.30**	**50.70**	**41.80**	17.60	**14.60**
R-wrn-34-10	AutoAttack	7.70	7.80	8.20	7.80	8.90	15.20	**22.50**
	UAP	3.00	3.10	2.40	2.90	2.60	0.90	0.40
	UAPPGD	2.90	2.80	2.20	1.00	1.60	0.70	0.60
	CW	10.30	9.10	13.00	10.60	10.40	8.80	21.20
	RAP	8.10	7.30	8.10	7.60	7.70	7.10	9.90
	Ours	**59.10**	**54.80**	**51.80**	**50.00**	**42.50**	**17.00**	14.70

Table 11: Transferability performance of the LIMANS ℓ_∞-attacks on ImageNet ($\epsilon = 4/255$), in terms of fooling rates. The best transferable results are marked in red bold font, and the best specific attacking results are marked in black bold font.

		MobileNet	ResNet18	DenseNet	VGG	R-r18	R-50-2
MobileNet	AutoAttack	**100**	26.38	20.44	26.94	1.64	1.24
	UAP	48.48	11.5	10.46	17.28	1.8	0.84
	UAPPGD	69.94	18.04	14.34	22.34	2.72	1.56
	TI-FGSM	99.74	36.98	31.66	31.24	3.2	2.56
	VMI-FGSM	**100**	44.84	37.92	42.92	2.92	2.04
	VNI-FGSM	99.98	44.64	36.54	43.62	2.88	2.00
	NAA	84.56	15.1	11.72	16.88	2.1	1.2
	RAP	96.52	**54.58**	**47.24**	**49.16**	3.72	3.16
	Ours	75.24	50.06	46.94	44.34	**10.02**	**5.62**
ResNet18	AutoAttack	40.3	**100**	35.76	34.9	1.8	1.34
	UAP	13.34	11.3	9.00	11.72	1.36	0.86
	UAPPGD	25.3	47.22	18.44	23.26	2.5	1.44
	TI-FGSM	32.06	99.84	31.38	31.66	2.98	2.8
	VMI-FGSM	56.5	**100**	51.78	50.2	2.9	2.04
	VNI-FGSM	56.74	99.98	51.4	**51.42**	2.84	2.04
	NAA	22.54	97.94	14.84	19.3	2.12	1.2
	RAP	53.36	96.74	51.30	50.60	3.80	3.14
	Ours	**59.16**	59.16	**53.14**	48.28	**10.48**	**6.62**
DenseNet	AutoAttack	37.72	40.4	**100**	30.22	1.8	1.3
	UAP	12.76	9.94	9.8	11.42	1.24	0.92
	UAPPGD	22.72	20.7	40.04	20.18	2.48	1.28
	TI-FGSM	30.1	35.56	99.66	27	3.12	2.32
	VMI-FGSM	52.22	55.44	99.98	44.82	2.9	2.06
	VNI-FGSM	53.88	**56.9**	99.98	46.16	2.64	2.1
	NAA	24.22	25.68	98.34	21.38	1.34	1.42
	RAP	48.16	54.12	96.76	42.00	3.12	3.30
	Ours	**58.86**	**56.9**	57.26	**47.74**	**11.3**	**7.32**
VGG	AutoAttack	47.94	40.06	32.62	**100**	2.34	1.42
	UAP	13.34	9.8	8.82	13.6	1.34	0.78
	UAPPGD	24.42	23.16	18.12	46.26	2.54	1.6
	TI-FGSM	33.2	38.26	29.3	99.4	2.96	2.28
	VMI-FGSM	57.52	53.46	43.76	99.86	2.9	2.2
	VNI-FGSM	**57.98**	53.96	42.88	99.84	2.76	2.24
	NAA	19.62	14.92	12.18	79.96	2.18	1.4
	RAP	53.14	53.12	42.68	95.68	3.48	2.84
	Ours	57.68	**54.14**	**50.04**	51.62	**10.68**	**6.24**
R-r18	AutoAttack	13.7	15.8	10.82	14.6	**71.74**	10.78
	UAP	11.52	9.32	8.46	10.90	1.44	1.16
	UAPPGD	14.00	12.34	11.20	13.56	3.14	1.66
	TI-FGSM	11.88	13.42	10.08	11.02	54.46	10.14
	VMI-FGSM	17.00	17.80	12.12	16.08	64.98	11.94
	VNI-FGSM	16.14	17.66	12.48	16.08	63.22	11.74
	NAA	11.46	10.86	9.34	11.42	21.48	4.9
	RAP	11.32	10.80	8.16	10.32	45.80	7.94
	Ours	**37.14**	**33.2**	**33.76**	**29.90**	29.84	**12.94**
R-50-2	AutoAttack	20.14	22.76	17.36	19.44	15.42	**59.02**
	UAP	9.88	7.60	6.96	8.62	1.84	1.24
	UAPPGD	14.54	12.56	10.92	14.36	2.16	1.38
	TI-FGSM	14.16	16.34	12.68	13.68	17.16	43.66
	VMI-FGSM	24.22	26.66	20.12	23.86	17.82	54.56
	VNI-FGSM	23.88	26.22	19.68	23.28	18.00	52.28
	NAA	14.08	13.12	10.20	14.04	9.82	12.58
	RAP	13.82	14.06	10.52	13.5	15.54	34.1
	Ours	**42.18**	**42.5**	**42.46**	**34.22**	**23.7**	18.02