The data-driven transferable adversarial space

Yuan Liu YUAN.LIU@INSA-ROUEN.FR
Séphane Canu STEPHANE.CANU@INSA-ROUEN.FR
Normandie Univ, INSA Rouen, LITIS, Saint-Etienne-du-Rouvray, France

Editors: Vu Nguyen and Hsuan-Tien Lin

Abstract

Deep Neural Network (DNN) models are vulnerable to deception through the intentional addition of imperceptible perturbations to benign examples, posing a significant threat to security-sensitive applications. To address this, understanding the underlying causes of this phenomenon is crucial for developing robust models. A key research area involves investigating the characteristics of adversarial directions, which have been found to be perpendicular to decision boundaries and associated with low-density regions of the data. Existing research primarily focuses on adversarial directions for individual examples, while decision boundaries and data distributions are inherently dataset-dependent. This paper explores the space of adversarial perturbations within a dataset. Specifically, we represent adversarial perturbations as a linear combination of adversarial directions, followed by a non-linear projection. Using the proposed greedy algorithm, we train the adversarial space spanned by the set of adversarial directions. Experiments on Cifar10 and ImageNet substantiate the existence of the adversarial space as an embedded space within the entire data space. Furthermore, the learned adversarial space enables statistical analysis of decision boundaries. Finally, we observe that the adversarial space learned on one DNN model is model-agnostic, and that the adversarial space learned on a vanilla model is a subset of that learned on a robust model, implicating data distribution as the underlying cause of adversarial examples.

Keywords: Adversarial perturbation, Dictionary learning, Deep learning, Robust model, Transferability

1. Introduction

Deep Neural Networks (DNNs) have demonstrated significant advancements over the past decades, finding successful applications in Natural Language Processing (NLP), object recognition, image processing, and other areas. However, it has been revealed that DNNs are often vulnerable to imperceptible, intentionally crafted perturbations, known as adversarial attacks Goodfellow et al. (2015). This vulnerability raises significant concerns, particularly in security-critical domains such as autonomous driving and AI-assisted diagnosis systems.

To address these concerns, current research predominantly focuses on developing algorithms to generate adversarial perturbations within bounded constraints, aiming for a high Attack Success Rate (ASR) Croce and Hein (2020); Carlini and Wagner (2017); Madry et al. (2018); Wang and He (2021). Simultaneously, defensive strategies have evolved to counter these adversarial perturbations effectively Rebuffi et al. (2021); Huang et al. (2023); Wu and Zhu (2020). In contrast, an emerging research direction seeks to uncover the fundamental causes of adversarial vulnerabilities, with the potential to foster the development of truly

robust models. However, this area remains relatively underexplored, representing an open frontier in current research Han et al. (2023).

The fundamental principle behind crafting adversarial examples lies in identifying directions where benign examples cross decision boundaries, resulting in misclassification. Recent studies indicate that these adversarial directions align with gradient directions and are closely orthogonal to decision boundaries Goodfellow et al. (2015); Li et al. (2020). They also extend towards low-density regions of datasets, thereby enhancing transferability Zhu et al. (2021). Moreover, Ilyas et al. Ilyas et al. (2019) propose that adversarial directions can be interpreted as non-robust features, while Tramèr et al. Tramèr et al. (2017) demonstrate that adversarial perturbations form a subspace rather than isolated instances. However, all the aforementioned hypotheses and analyses concerning adversarial perturbations pertain to individual examples. In reality, adversarial perturbations are inherently tied to the decision boundary, which is primarily influenced by the entire data distribution rather than any singular example. Thus, adopting a holistic perspective on the space of adversarial perturbations across a dataset offers deeper insights into their existence and characteristics.

In this study, we propose to investigate the space of adversarial perturbations within a dataset. Specifically, we represent an adversarial perturbation as a linear combination of a set of adversarial directions, followed by a non-linear projection. This adversarial space is learned by solving an optimization problem using a proposed greedy algorithm. Our experiments on Cifar10 and ImageNet yield explicit sets of adversarial directions, enabling a statistical analysis of decision boundaries. Furthermore, we verify that the learned adversarial space is model-agnostic, implicating its dependence on data distribution. In summary, our main contributions are as follows:

1. Demonstrate the existence of the adversarial perturbation space within the dataset, which is an embedded subset of the image space.
2. Propose a greedy algorithm for learning the set of adversarial directions.
3. Provide an indirect analysis of decision boundaries based on statistical results.
4. Illustrate the model-agnostic characteristics of the learned adversarial space. We also find that the adversarial space learned by attacking vanilla models is a subset of the adversarial space learned by attacking robust models, emphasizing its dependence on data distribution.

The remainder of the paper is organized as follows: Section 2 introduces related research. Section 3 presents the problem of adversarial attacks. In Section 4, we detail the greedy algorithm for learning an adversarial space. Finally, Section 5 provides the experiments and analysis.

2. Related works

Adversarial perturbations remain a subject of active investigation, with hypotheses centered on two primary factors: DNN models and data distributions, each subject to ongoing debate and scrutiny.

Researchers focusing on DNN models attribute adversarial examples to inherent flaws in model properties. Goodfellow et al. associated adversarial attacks with the linearity of DNNs, which amplify minor perturbations into significant output shifts Goodfellow et al.

(2015). Ilyas et al. Ilyas et al. (2019) argued that DNNs often rely on non-robust features that are incomprehensible to humans for classification, exploited by adversarial attacks. However, the extent to which adversarial examples reflect genuine features is a topic of ongoing debate Li et al. (2024). Additionally, factors such as model architecture (e.g., skip connections), activation functions, and training procedures contribute to adversarial vulnerabilities Han et al. (2023).

Conversely, researchers examining data distribution contend that adversarial vulnerabilities stem primarily from intrinsic dataset properties. Gilmer et al. proposed that the high-dimensionality of data contributes significantly to the emergence of adversarial examples Gilmer et al. (2018); Pal et al. (2024). Fawzi et al. Fawzi et al. (2018) highlighted the susceptibility of any DNN model on specific datasets to adversarial perturbations. Pal et al. Pal et al. (2024) revealed a data distribution that mitigates adversarial attacks, though achieving this in the original space for complex datasets like Cifar10 remains challenging. Further insights into adversarial perturbations reveal that adversarial directions are orthogonal to the tangent space of the data manifold Li et al. (2020). Moreover, decision boundaries, heavily influenced by the data manifold, exhibit similar characteristics across different models, facilitating adversarial example transferability Tramèr et al. (2017). Zhu et al. Zhu et al. (2021) identified that adversarial directions also point towards low-density regions, enhancing their transferability. Tramèr et al. Tramèr et al. (2017) argued that the adversarial direction associated with a benign example exists not in isolation but within a broader space. However, this space pertaining to a single example does not extend to the entire dataset, thereby limiting our ability to globally analyze the relationship of adversarial perturbations to the data distribution. As of today, research on this topic remains scarce.

3. Statement of the adversarial attacks problem

Let $F\colon \mathcal{X} \subset \mathbb{R}^n \to \mathbb{R}^c$ be a DNN model that maps an input $\mathbf{x}$ of dimension n to a vector of class scores $F(\mathbf{x})$ with dimension c, representing the total number of classes. This vector is also known as the **logit** output. When the **softmax** function is applied, it converts $F(\mathbf{x})$ into a class probabilities vector $f(\mathbf{x}) = \mathbf{softmax}(F(\mathbf{x}))$ of the same dimension. The predicted class for a given input $\mathbf{x}$ is then obtained by $y = \operatorname{argmax}_i f_i(\mathbf{x})$ or $y = \operatorname{argmax}_i F_i(\mathbf{x})$ where $f_i(\mathbf{x})$ and $F_i(\mathbf{x})$ denotes the i-th element of the output vector $f(\mathbf{x})$ and $F(\mathbf{x})$ and $i \in \{1, \ldots, c\}$.

An adversarial attack refers to a method that generates an imperceptible adversarial perturbation $\boldsymbol{\delta}$ to a benign input $\mathbf{x} \in \mathcal{X}$, resulting in the adversarial example $\mathbf{x}' = \Pi_{\mathcal{X}}(\mathbf{x}+\boldsymbol{\delta})$ where $\Pi_{\mathcal{X}}$ denotes the projection onto $\mathcal{X}$, such that $\operatorname{argmax}_i f_i(\mathbf{x}) \neq \operatorname{argmax}_i f_i(\mathbf{x}')$. The problem of adversarial attack can thus be summarized as follows:

$$\begin{aligned} \min_{\boldsymbol{\delta}\in\mathbb{R}^n} \quad & \mathbf{dist}(\mathbf{x}, \mathbf{x}') \\ s.t. \quad & \begin{cases} \mathbf{x}' = \Pi_{\mathcal{X}}(\mathbf{x} + \boldsymbol{\delta}) \\ \operatorname{argmax}_i f_i(\mathbf{x}) \neq \operatorname{argmax}_i f_i(\mathbf{x}')\,, \end{cases} \end{aligned} \tag{1}$$

where **dist** is a distance metric, typically $\mathbf{dist}(\mathbf{x}, \mathbf{x}') = \|\boldsymbol{\delta}\|_p$, the ℓ_p-norm of the adversarial perturbation (usually ℓ_2-norm and ℓ_∞-norm are considered). However, addressing this problem directly is challenging due to the non-continuity of the constraint $\operatorname{argmax}_i f_i(\mathbf{x}) \neq$

$\text{argmax}_i\, f_i(\mathbf{x}')$. To overcome these difficulties, our strategy consists to relax it by introducing a continuous and differentiable loss function, such as the negative cross-entropy loss,

$$L(f(\mathbf{x}), f(\mathbf{x}'), y) = \sum_{0}^{c} y_i \log(f_i(\mathbf{x}')), \tag{2}$$

where $y_j = 1$ if $j = \text{argmax}_i\, f_i(\mathbf{x})$ else $y_j = 0$, or the loss of **logit** difference introduced in Carlini and Wagner (2017), which is defined as,

$$L(f(\mathbf{x}), f(\mathbf{x}'), y) = \max(F_j(\mathbf{x}') - F_h(\mathbf{x}'), -\gamma), \tag{3}$$

where $j = \text{argmax}_i\, f_i(\mathbf{x})$ and $h = \text{argmax}_{i \neq j}\, f_i(\mathbf{x})$, and γ a non-negative parameter. Using the loss function, the adversarial perturbation for an input $\mathbf{x}$ can be produced by solving the problem,

$$\begin{array}{ll} \min\limits_{\boldsymbol{\delta} \in \mathbb{R}^n} & L(f(\mathbf{x}), f(\mathbf{x}'), y) \\ s.t. & \begin{cases} \mathbf{x}' = \Pi_{\mathcal{X}}(\mathbf{x} + \boldsymbol{\delta}) \\ \|\boldsymbol{\delta}\|_p \leq \epsilon\,, \end{cases} \end{array} \tag{4}$$

with $\epsilon \in \mathbb{R}_+$ a predefined small value associated with the chosen norm.

4. Representation of adversarial samples in the adversarial space

This section investigates adversarial perturbations within an image dataset (e.g., CIFAR-10) designed to fool a specific deep neural network (DNN) model. These perturbations are constrained to an embedded lower-dimensional subspace, where we hypothesize that such perturbations can be represented as a function of adversarial directions and a coding vector. Specifically, we define: $\boldsymbol{\delta}(\mathbf{x}) = g(D\mathbf{v}(\mathbf{x}), \epsilon_p)$ (hereafter referred to simply as ϵ for clarity), where $D = \{\mathbf{d}_1, \ldots, \mathbf{d}_i, \ldots, \mathbf{d}_k\} \in \mathcal{D} = [-1, 1]^{n \times k}$ consists of k adversarial directions, referred to as the adversarial dictionary, and $\mathbf{v}(\mathbf{x}) \in \mathbb{R}^k$ is the corresponding composition coefficient vector (hereafter denoted simply as $\mathbf{v}$ for readability). The function g ensures that the resulting perturbations remain within the feasible region defined by the constraints of the problem (4), that is, the adversarial example $\mathbf{x}'$ is a valid input and the ℓ_p-norm of the adversarial perturbation $\boldsymbol{\delta}$ belongs to the ϵ ball.

In this approach, the problem can be divided into two phases: 1) Adversarial dictionary learning, which aims to find the optimal D with the smallest number of directions that can attack the most examples in the dataset; 2) Coding of the adversarial perturbation, where, given an unseen example $\mathbf{x}$ and the trained D, we only need to solve for $\mathbf{v}$.

We then present the problems of adversarial dictionary learning and the coding of adversarial perturbations, along with the associated algorithms to find their solutions.

4.1. Learning the adversarial dictionary

The adversarial dictionary is dataset-specific and consists of a small number of adversarial directions capable of attacking all vulnerable examples under specified conditions within the dataset. Within this theoretical framework, we formulate the adversarial dictionary learning problem.

Problem 4.1 (Adversarial dictionary learning) *Given a DNN model f and a labeled dataset $\mathcal{T} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, the adversarial dictionary learning is to search for the optimal solution for D and $V = [\mathbf{v}_0, \ldots, \mathbf{v}_N]$, by maximizing the number of examples attacked in the dataset using the smallest number of adversarial directions k. The problem can be formulated as the following bi-objective optimization problem,*

$$\begin{aligned} \min_{D\in[-1,1]^{n\times k}, V\in\mathbb{R}^{N\times k}} \quad & \Big[-\sum_{i=1}^{N} \mathbb{1}\{\operatorname{argmax}_l f_l(\mathbf{x}_i) \neq \operatorname{argmax}_l f_l(\mathbf{x}_i^{'}\},\ k \Big] \\ s.t. \quad & \begin{cases} \mathbf{x}_i^{'} = \Pi_{\mathcal{X}}(\mathbf{x}_i + g(D\mathbf{v}_i, \epsilon)) & \forall i = 1, \ldots, N \\ \|\mathbf{d}_i\|_p = 1 & \forall i = 1, \ldots, k, \end{cases} \end{aligned} \tag{5}$$

where $\sum_{i=1}^N \mathbb{1}\{\operatorname{argmax}_l f_l(\mathbf{x}_i) \neq \operatorname{argmax}_l f_l(\mathbf{x}_i^{'}\}$ counts the number of examples successfully attacked.

Unfortunately, this bi-objective optimization problem is untackleable directly. The two objectives are contradictory. Decreasing one objective comes at the expense of the other, as increasing the number of adversarial directions k also increases the attack success rate.

To address this issue, we propose to compute the entire Pareto front of the problem by monitoring the evolution of the attack rate as the size of the dictionary k increases in a greedy fashion. This approach will enable a trade-off between the effectiveness of the attack and the size of the dictionary. Specifically, in the j^{th} iteration, we successively
1). learn a sub-dictionary, where a set of m^j adversarial directions is trained to maximize the number of examples attacked among the remaining unattacked examples in the training data. The sub-dictionary learning problem, formulated using the surrogate loss function introduced in Section 3, can be formulated:

$$\begin{aligned} \min_{D^j\in[-1,1]^{n\times m^j}, V^j\in\mathbb{R}^{m_j\times N_j}} \quad & \sum_{i=0}^{N_j} \mathcal{L}\big(f(\mathbf{x}_i^{'}), f(\mathbf{x}_i), y_i\big) \\ s.t. \quad & \begin{cases} \mathbf{x}_i^{'} = \Pi_{\mathcal{X}}(\mathbf{x}_i + g(D_j\mathbf{v}_i^j, \epsilon)) & \forall i = 1, \ldots, N_j \\ \|\mathbf{d}_i^j\|_p = 1 & \forall i = 1, \ldots, m^j, \end{cases} \end{aligned} \tag{6}$$

where D^j is the j^{th} sub-dictionary, $D_j = [D_{j-1}, D^j] = [D^1, \ldots, D^j]$ represents the current dictionary updated in the j^{th} iteration, and m^j denotes the number of adversarial directions added in the iteration, $m_j = m_{j-1} + m^j$ indicates the total number of adversarial directions in the current dictionary D_j. For the function g, we choose **tanh**. Denoting $\mathbf{v}^j_{(i,1:m_{j-1})}$ as the sub-vector containing the first m_{j-1} elements of $\mathbf{v}_i^j$, and $\mathbf{v}^j_{(i,m_{j-1}+1:m_j)}$ as the sub-vector with indices from $(m_{j-1}+1)$ to m_j, the $D_j\mathbf{v}_i^j$ can thus be represented by

$$D_j\mathbf{v}_i^j = D_{j-1}\mathbf{v}^j_{(i,1:m_{j-1})} + D^j\mathbf{v}^j_{(i,m_{j-1}+1:m_j)}. \tag{7}$$

Now, with the explicit formulation of $\mathbf{x}_i^{'}$ and $D_j\mathbf{v}_i^j$, problem (6) can be tackled using the projected gradient descent algorithm. The algorithm for computing (D^j, V^j) is summarized in Algorithm 1. Specifically, the problem (6) is not strictly convex and is non-differentiable everywhere due to the non-linearity of the deep model f. While Automatic Differentiation

Algorithm 1 Updating sub-dictionary

Require: Remaining unattacked training examples $\mathcal{T}_j = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_j}$, DNN model f, current adversarial dictionary D_{j-1}, sub-dictionary D^j, set of composition vectors V^j, initial learning rate (ρ_D, ρ_v), maximal magnitude of adversarial perturbation ϵ, batch size B, stop criterion τ, list of losses $S\mathcal{L}$, current minimal loss $\mathcal{L}_b$, current best fooling rate fr_b, current best solution $(BD^j, (BV^j))$, length between checkpoints n_c, total loss $\mathcal{L}_t$ and total fooling rate fr_t

1: Initialize $V^{j(0)} = 0$, $D^{j(0)} \sim \mathcal{N}(0, I_{n\times m^j})$, loss $\mathcal{L}_b = 10^6$, $\mathcal{L} = \mathcal{L}_t = 0$, fooling rate $fr_t = fr_b = 0$

2: **while** $\rho_D > \tau$ **do**
3: **for** $(\mathbf{x}_i, \mathbf{y}_i) \in \mathcal{T}_j$ **do**
4: $\boldsymbol{\delta}_i = g(D_j \mathbf{v}_i^j, \epsilon)$ (adversarial perturbation)
5: $\mathbf{x}_i^{'} = \Pi_{\mathcal{X}}(\mathbf{x}_i + \boldsymbol{\delta}_i)$ (adversarial example)
6: $\mathcal{L} = \mathcal{L} + \mathcal{L}(f(\mathbf{x}_i^{'}), f(\mathbf{x}_i), y_i)$ (objective function)
7: $fr_t = fr_t + \mathbb{1}\{f(\mathbf{x}_i^{'}) \neq f(\mathbf{x}_i)\}$ (update attacked examples' number)
8: **if** modulo$(i/B) = 0$ **then**
9: $D^j \leftarrow \Pi_{\mathcal{D}}(\text{Optim}(\nabla_{D^j}\mathcal{L}, D^j, \rho_D))$ (Update D^j by projected gradient descent)
10: $V^j \leftarrow \text{Optim}(\nabla_{V^j}\mathcal{L}, V^j, \rho_v)$ (Update V^j by gradient descent)
11: $\mathcal{L}_t = \mathcal{L}_t + \mathcal{L}$, $\mathcal{L} = 0$, $i = 0$
12: **end if**
13: **end for**
14: $S\mathcal{L} = [S\mathcal{L}, \mathcal{L}_t]$
15: **if** $\mathcal{L}_t < \mathcal{L}_b$ and $fr_t > fr_b$ **then**
16: $(BD^j, BV^j) = (D^j, V^j)$ (update the best solution)
17: $\mathcal{L}_b = \mathcal{L}_t$, $fr_b = fr_t$
18: **end if**
19: **if** len$(S\mathcal{L}) = n_c$ and $(\mathcal{L}_b = S\mathcal{L}[0]$ or $\sum_{l=1}^{n_c-1} \mathbb{1}\{S\mathcal{L}_{l-1} > S\mathcal{L}_l\} < 0.75)$ **then**
20: $\rho_D = \rho_D/2$, $\rho_v = \rho_v/2$ (decrease the learning rate)
21: $S\mathcal{L} = [\mathcal{L}_b]$, $(D^j, V^j) = (BD^j, BV^j)$ (restart using the current best solution)
22: **end if**
23: **end while**
24: **return** D^j and V^j

enables computation of local gradients with respect to D^j and V^j, achieving only a local solution is highly probable. To optimize, we employ techniques introduced in Croce and Hein (2020). To begin, this method explores solutions across the entire feasible region using a fixed, relatively large learning rate to facilitate escaping local minima. Every n_c iterations, we assess whether the current learning rate effectively reduces the loss. If not, and no better solution is found after several iterations or if the loss fails to decrease, the learning rate is halved, as detailed in Algorithm 1, lines 13-16. The process concludes when the learning rate of D^j falls below a stopping criterion τ thereby achieving the final solution.

2). Identify attacked examples Within the trained subspace D_j, we compute adversarial perturbations for each training example $\mathcal{T}_j$ to generate adversarial examples $\mathbf{x}_i^{'} = \Pi_{\mathcal{X}}(\mathbf{x}_i +$

$g(D_j\mathbf{v}_i^j, \epsilon))$, $\forall(\mathbf{x}_i, y_i) \in \mathcal{T}_j$. We then identify the subset of successfully attacked examples $\mathcal{A}_j^0 = \{(\mathbf{x}_i, y_i) \mid f(\mathbf{x}_i^{'}) \neq f(\mathbf{x}_i)), \forall(\mathbf{x}_i, y_i) \in \mathcal{T}_j\}$. Furthermore, it's important to note that ϵ_1 the perturbation magnitude used in the i^{th} iteration, may differ from ϵ. This discrepancy necessitates re-computing the composition vectors for examples that remain unattacked under the condition $\|\boldsymbol{\delta}\|_p < \epsilon$. Consequently, the final ensemble of successfully attacked examples is defined as:

$$\begin{cases} \mathbf{x}_i^{'} = \text{coding}(\mathbf{x}_i, D_j, y_i, \epsilon), \quad \forall(\mathbf{x}_i, y_i) \in \mathcal{T}_j / \mathcal{A}_j^0 \\ \mathcal{A}_j = \left\{ i \mid f(\mathbf{x}_i^{'}) \neq f(\mathbf{x}_i)),\ \forall(\mathbf{x}_i, y_i) \in \mathcal{T}_j \right\}, \end{cases} \tag{8}$$

3) Updating the training data Upon identifying successfully attacked examples, they are not immediately removed from the training data. Instead, a verification step is introduced to manage redundancy in directions. Specifically, if the number of identified attacked examples $|\mathcal{A}_j|$ exceeds the size of the sub-dictionary m^j, those examples are excluded. Subsequently, the training data is updated for the next iteration as $\mathcal{T}_{j+1} = \mathcal{T}_j / \mathcal{A}_j$. Otherwise, we consider the following situations: (a) If the current constraint on $\boldsymbol{\delta}$ is too strict, i.e., $\epsilon_1 < \epsilon$, then we relax ϵ_1 by setting $\epsilon_1 = \epsilon_1 + 1/255$; (b) If the number of training data is small or insufficient to learn a sub-dictionary of size m^j, then m^j is reduced to the size of the training data $|\mathcal{T}_j|$; (c) If there is a large amount of training data and $\epsilon_1 = \epsilon$, but the learned sub-dictionary does not effectively attack m^j examples. In this case, the D^j and V^j re-initialized using a well-developed attack method (e.g., AutoAttack), as detailed below:

$$\begin{cases} \mathbf{d}_i^{temp} = g^{-1}((\mathbf{atk}_{(f,\epsilon)}(\mathbf{x}_i, y_i) - \mathbf{x}_i)/\epsilon), \quad \forall i = 1, \ldots, m^j \\ \mathbf{v}_i^j[m_{j-1} + i] = ||\mathbf{d}_i^{temp}||_\infty, \quad \forall i = 1, \ldots, N_j \\ \mathbf{d}_i^j = \mathbf{d}_i^{temp} / ||\mathbf{d}_i^{temp}||_\infty, \quad \forall i = 1, \ldots, m^j, \end{cases} \tag{9}$$

where $\mathbf{d}_i^{temp}$ denotes the adversarial direction in the space of D, $\mathbf{v}_i^j[m_{j-1}+i]$ assigns the l_∞-norm of $\mathbf{d}_i^{temp}$ and $\mathbf{d}_i$ is derived by normalizing $\mathbf{d}_i^{temp}$. Whenever any of the aforementioned situations arises, the D^j and V^j are retrained until at least m^j examples are successfully attacked. However, if this process is attempted more than 10 times without any example being successfully attacked, the program stops and returns D_{j-1}. Otherwise, if the model to be deceived is a vanilla model, the program stops and returns D_j. For a robust model, the size of the sub-dictionary is reset as $m^j = \min(\max(1.5 \times N_a, \frac{|\mathcal{T}_j|}{a}), \frac{|\mathcal{T}_j|}{b})$, the number of attempts t_s is reset to 0, and D^j is retrained for a maximum of the last 10 attempts.

4.2. Coding of the adversarial perturbation

When D is learned, the adversarial perturbation of an input, represented by $\boldsymbol{\delta} = g(D\mathbf{v}, \epsilon)$, is determined by solving for $\mathbf{v}$. This problem, known as the coding of the adversarial perturbation, is formulated as follows:

Problem 4.2 (Coding of the adversarial perturbation) *Given a DNN model f and a labeled input $(\mathbf{x}, y)$, the objective is to compute a vector $\mathbf{v} \in \mathbb{R}^k$ that accurately represents an adversarial perturbation while satisfying specific problem constraints:*

$$\min_{\mathbf{v} \in \mathbb{R}^k} \quad \mathcal{L}\big(f(\mathbf{x}'), f(\mathbf{x}), y\big) \quad s.t. \quad \mathbf{x}' = \Pi_{\mathcal{X}}(\mathbf{x} + g(D\mathbf{v}, \epsilon)) \tag{10}$$

Algorithm 2 Adversarial dictionary learning

Require: Initial training examples $\mathcal{T}_1 = \{(\mathbf{x}_i, y_i)\}_{i=0}^{N}$, initial attacked examples $\mathcal{A} = \emptyset$, DNN model f, maximal magnitude of perturbations ϵ, initial magnitude of perturbations ϵ_1, initial adversarial dictionary $D_0 = \emptyset$, initial sub-dictionary $D^1 \sim \mathcal{N}(0, I_{n\times m_1})$, initial coefficient vectors $V^1 = 0$, attack method **atk**, threshold of small number of training data remaining N_s, number of attempts t_e, best number of attacked examples N_a, best adversarial dictionary BD and best coefficient vectors BV, coefficient a and b, mark of last sub-dictionary **last**=False

Initialize j = 1, number of attacked examples $N_e = 0$, number of examples $N_1 = N$, $N_a = 0$, initial sub-dictionary size m^1
while $N_e < 0.999\,N$ **do**
  (D^j, V^j) = Updating sub-dictionary$(\mathcal{T}_j, D_{j-1}, \epsilon_j)$ (Algorithm 1)
  $\mathcal{A}_j$ = identify attacked examples$(\mathcal{T}_j, D_{j-1}, D^j, \epsilon)$
  $N_e = N_e + |\mathcal{A}_j|$
  // update the training data if an insufficient number of examples are attacked
  if $\epsilon_j < \epsilon$ and $N_e < m^j$ **then**
    $\epsilon_j = \epsilon_j + 1/255$
  else if $N_e < m^j$ **then**
    if $N_e > N_a$ **then**
      $N_a = N_e$, $BD = D^j$, $BV = V^j$ (refine the best solution)
    end if
    if $t_s = 10$ and $N_a = 0$ **then**
      return $D = D_{j-1}$ (stop training when no additional examples attacked)
    else if $t_s = 10$ and **last then**
      $D^j = BD$, $V^j = BV$, $t_s = t_s + 1$ (stop the trial and keep the best solution)
    end if
    //mark as the final sub-dictionary training and resize the sub-dictionary accordingly
    if $(|T_j| < m^j$ or $|T_j| < N_s)$ and $t_s = 0$ **then**
      $m^j = |T_j|$, **last**=True
    else if $fr\,N < m^{j+1}$ and f is a robust model and $t_s = 10$ **then**
      $m^j = \min(\max(1.5N_a, \frac{|\mathcal{T}_j|}{a}), \frac{|\mathcal{T}_j|}{b})$, $t_s = 0$, **last**=True
    end if
    (D^j, V^j) = Reinitialize by attack method$(\mathbf{atk}, \mathcal{T}_j, g, m_{j-1}, m^j)$
    $t_s = t_s + 1$
  end if
  if $N_e >= m^j$ and $t_s > 10$ **then**
    $\mathcal{T}_{j+1} = \mathcal{T}_j / \mathcal{A}_j$, $D_j = [D_{j-1}, D^j]$ (Update the training examples and dictionary)
    reset $j = j + 1$, $N_a = 0$
    return if **last**
  end if
end while
return $D = D_{j-1}$

The problem of coding the adversarial perturbation with respect to $\mathbf{v}$ remains non-convex and non-differentiable everywhere in the feasible region due to the non-linearity introduced by the DNN model f. Similar to solving the adversarial dictionary learning problem, we utilize auto-differentiation and employ gradient descent to search for a solution of $\mathbf{v}$. However, due to the risk of converging to local minima, we apply also the aforementioned technique to gradually reduce the learning rate.

5. Experiences and results

In this section, we present our experiments on adversarial dictionary learning conducted using the CIFAR-10 and ImageNet datasets. We begin by detailing the experimental setup and methodology. Following this, we showcase the learned dictionaries and analyze their characteristics to provide deeper insights.

5.1. Experimental Settings

Dataset. The experiments are conducted using two datasets: CIFAR-10[1] and ImageNet ILSVRC2012[2]. These experiments are performed exclusively on the validation sets. The datasets are partitioned into three subsets: the training subset for learning the adversarial dictionary D, the validation subset for hyperparameter tuning, including the initial perturbation magnitude ϵ_1 and the initial learning rate for the composition vector ρ_v and the testing subset for evaluating performance. For CIFAR-10, the dataset is divided into 7,000 training examples, 1,500 validation examples, and 1,000 testing examples. For ImageNet, the dataset is partitioned into 20,000 training examples, 2,000 validation examples, and 1,000 testing examples.

DNN models. Our proposed algorithm is evaluated on CIFAR-10 across five vanilla DNN models: ResNet18, ResNet50, Inception-V3, DenseNet121, and VGG11, as well as three robust DNN models: robust ResNet-18[3], robust ResNet152 [4], and robust WideResNet-34-10 [4] Sehwag et al. (2022). When experimenting on ImageNet, Inception-V3 is excluded due to its different input size compared to other vanilla models.

Parameters. The parameters used in our attack are as follows: During the training phase, we employ the AdamW optimizer with a stop criterion of $\tau = 10^{-4}$. We exclusively use the $p = \infty$ distance metric denoted by $\mathbf{dist} = l_\infty$ and the perturbation magnitude ϵ value is consistent with that used in RobustBench[4]. The initial learning rate ρ_D is fixed at 2ϵ. During the validation and testing phases, the initial learning rate ρ_v is set to 1, and the stop criterion is adjusted to $\tau = 0.1$. This work aims to analyze the properties of the adversarial space and demonstrate its model-agnostic and data-dependent characteristics. Consequently, all experiments are conducted under a white-box setting.

State-of-the-art attacks. In our experiments, we use the attack methods AutoAttack and PGD as reference attacks to measure the total number of examples that can be successfully attacked in the entire dataset.

5.2. Adversarial dictionary

5.2.1. Parameter Determination

The initial constraint on magnitude of adversarial perturbation ϵ_1. The parameter ϵ_1 significantly impacts the learned adversarial dictionary. In the simple scenario illustrated in Figure 1, if $\epsilon_1 = \epsilon_a$, the first learned direction is likely to be $\mathbf{d}_2$. Subsequently, the

1. https://www.cs.toronto.edu/ kriz/cifar.html
2. https://www.image-net.org/challenges/LSVRC/2012/
3. https://robustbench.github.io/
4. https://robustbench.github.io/

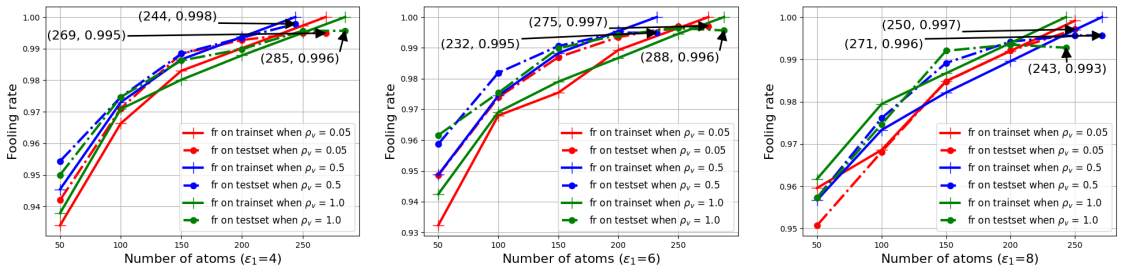


Figure 2: Fooling rate of attacking VGG on CIFAR-10 as a function of the learning rate (ρ_v) and initial perturbation magnitude (ϵ_1)

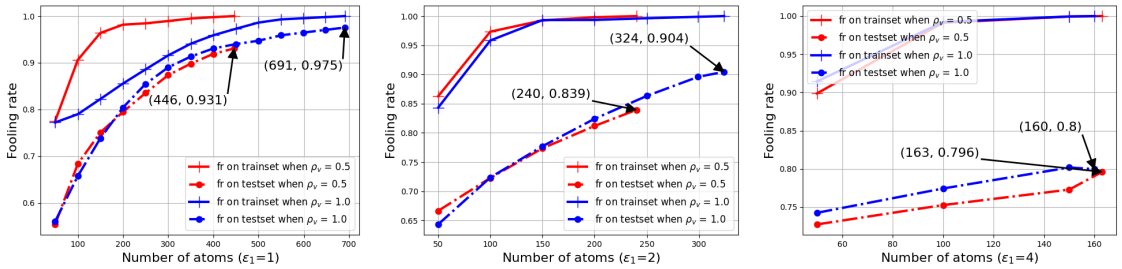


Figure 3: Fooling rate of VGG attacks on ImageNet with varying learning rates (ρ_v) and initial perturbation magnitudes (ϵ_1)

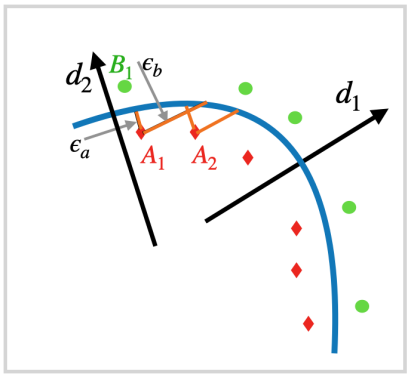
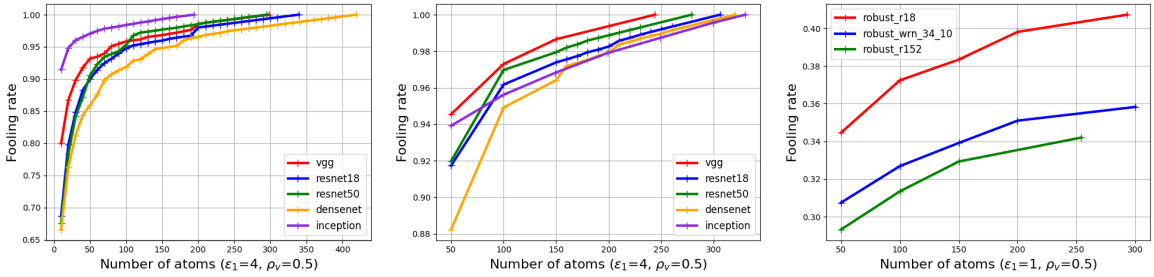


Figure 1: Illustration of the impact of ϵ_1 on adversarial directions learning

algorithm identifies the second direction $\mathbf{d}_1$, thus spanning the complete two-dimensional space $\{\mathbf{d}_1, \mathbf{d}_2\}$ which allows finding all adversarial perturbations in the plane. However, as ϵ_1 increases to ϵ_b, a one-dimensional adversarial dictionary $D = \{\mathbf{d}_1\}$ is likely to be computed. This dictionary cannot cover the adversarial perturbations situated in the direction $\mathbf{d}_2$; for example, the instance B_1 lacks an adversarial perturbation within D under the constraint $\boldsymbol{\delta} < \epsilon_2$. This conclusion is supported by the fooling rate results on ImageNet, as presented in Figure 3, Specifically, the larger ϵ_1 results in fewer trained adversarial directions (i.e., $k(\epsilon_1 = 4/255) < k(\epsilon_1 = 2/255) < k(\epsilon_1 = 1/255)$), leading to a deterioration in fooling rate performance ($\text{fr}(\epsilon_1 = 1/255) > \text{fr}(\epsilon_1 = 2/255) > \text{fr}(\epsilon_1 = 4/255)$). However, this phenomenon can be mitigated or avoided with a sufficiently large training dataset, as shown in Figure 2. For CIFAR-10, training with 700 examples per class showed little difference in the number of learned adversarial directions across different ϵ_1 values. Nonetheless, it is noteworthy that the performance on training data and test data is most consistent when $\epsilon_1 = 4/255$. Consequently, we chose $\epsilon_1 = 1/255$ for ImageNet and $\epsilon_1 = 4/255$ for CIFAR-10, considering the balance between the number of examples attacked and computational complexity.



(a) (b) (c)

Figure 4: Fooling rates on CIFAR-10 when attacking: (a) vanilla models with $N^1 = 10$, (b) vanilla models with $N^1 = 50$, and (c) robust models with $N^1 = 50$

The initial learning rate ρ_v. An adversarial direction can be considered an average adversarial example, which can be updated using the learning rate ρ_D as employed in Croce and Hein (2020). Subsequently, the learning rate ρ_v plays a critical role in determining the distribution of examples that share the same adversarial direction. For instance, in Figure 1, if all examples converge to a single adversarial direction $\mathbf{d}_1$, the resulting dictionary would contain only $\mathbf{d}_1$. Conversely, if subsets like A_1, A_2, and B_1 share one direction while others share a different one, the dictionary D would span both $\mathbf{d}_1$ and $\mathbf{d}_2$. Optimal selection of ρ_v ensures comprehensive coverage of training examples while generalizing well to unseen data. On ImageNet, setting $\rho_v = 1.0$ facilitates escaping local minima to find optimal solutions, whereas a smaller value like $\rho_v = 0.5$ tends to trap the search in local minima. In contrast, the impact of ρ_v on CIFAR-10 is minimal, suggesting exhaustive exploration of the feasible region when sufficient training data is available. Specifically, with $\rho_v = 0.5$ and $\epsilon_1 = 4/255$, we achieve optimal performance in attacking test data using the fewest adversarial directions. In conclusion, we select $\rho_v = 1.0$ for ImageNet and $\rho_v = 0.5$ for CIFAR-10, based on their respective abilities to balance exploration and exploitation in the adversarial space.

The initial size of subdictionary m_1. As illustrated in Figure 4, increasing m_1 enhances the discovery of an adversarial dictionary solution with fewer adversarial directions. This highlights the superior performance of global search algorithms over greedy methods in solution exploration. Therefore, we adopt $m_1 = 50$ for training the adversarial dictionary on both ImageNet and CIFAR-10 across various DNN models. We refrain from further increasing m_1 to manage computational complexity and memory requirements effectively.

5.2.2. Learned adversarial dictionary

Existence of the adversarial space. Adversarial perturbations reside within an embedded subspace of the image space $\mathcal{X} = [0, 1]^n$. This embedded space exists and is specific to the dataset on which it is trained. We consider the adversarial space to be adequately found when the total fooling rate on the training data approximates the results achieved with state-of-the-art attack methods. Furthermore, the dimension of the adversarial space varies when trained with different DNN models. Specifically, for CIFAR-10, the adversarial space has a dimension ranging from 244 to 330, as illustrated in Figure 4, For ImageNet, the

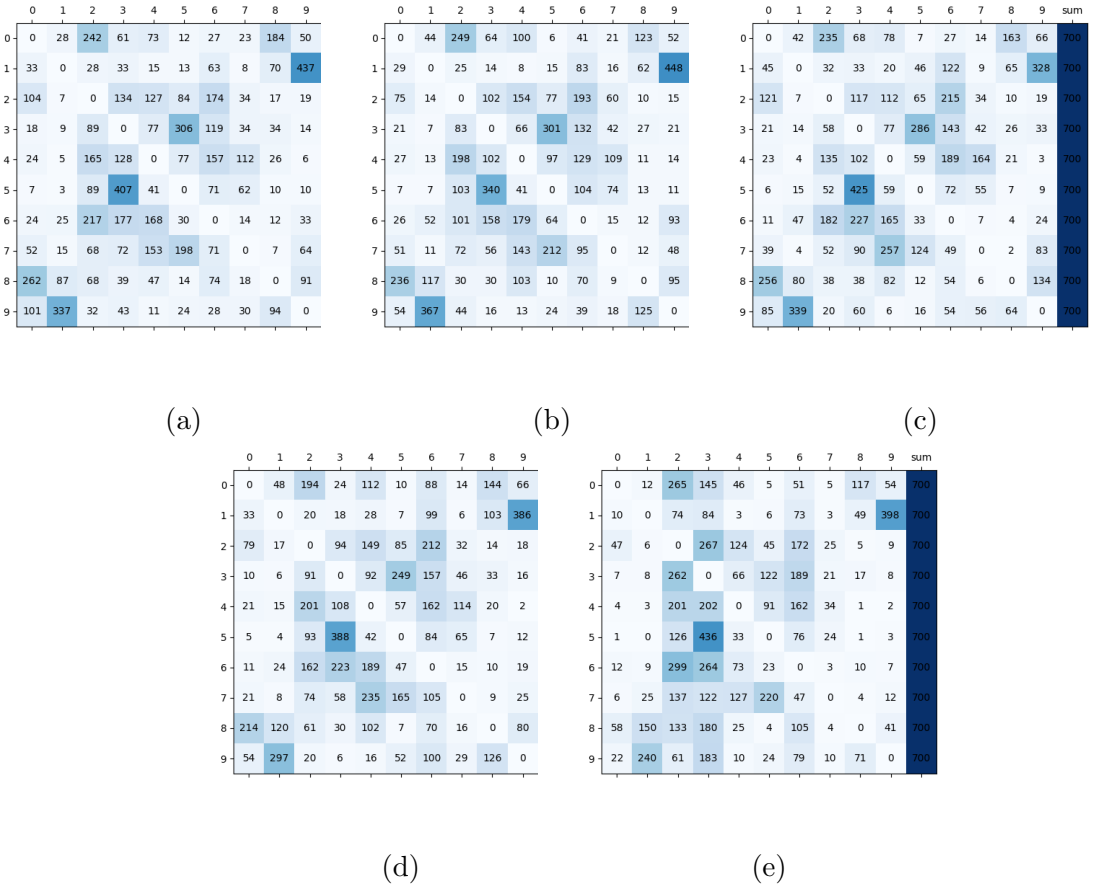
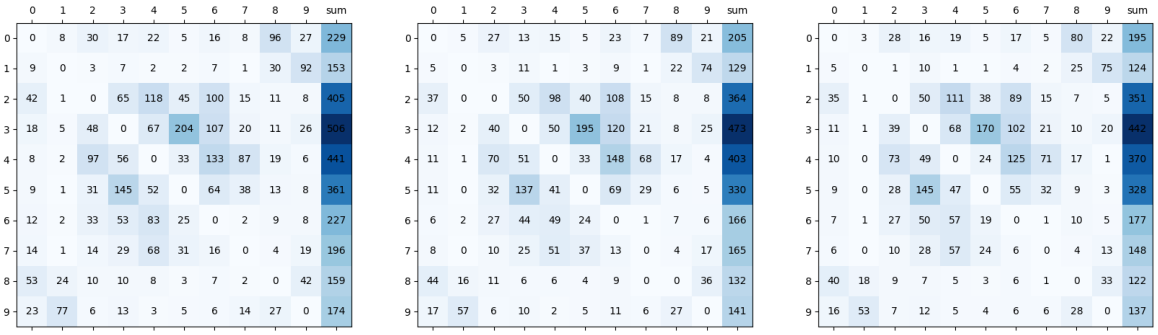


Figure 5: Confusion matrices with rows representing the initial labels and columns representing the shifted labels after attacking with the target models (a) ResNet18, (b) ResNet50, (c) VGG, (d) DenseNet, and (e) Inception.

range is from 233 to 691, as shown in Figure 6. The dimension of the adversarial space is significantly smaller than that of the image space; approximately 300 vs. 3072 for CIFAR-10 (with 10 classes), and approximately 450 vs. 150528 for ImageNet (with 1000 classes). Notably, the ratio for ImageNet is much smaller, which suggests that more decision boundaries are present within the same subspace. This could explain why images from ImageNet are easier to attack.

Decision boundaries defined by different DNN models. It was demonstrated that the adversarial direction associated with an example is typically perpendicular to the decision boundary. However, when adversarial directions are not constrained to be perpendicular but intersect with decision boundaries, we cannot always guarantee finding a minimal $\boldsymbol{\delta}$ for an example $\mathbf{x}$, For instance, in the case where the trained adversarial dictionary is $\{\mathbf{d}_1\}$, it may result in a non-minimal perturbation ϵ_b for the example A_1.

Due to high dimensionality, direct analysis of decision boundaries is impractical. However, statistical analysis using confusion matrices obtained from fooling a specific DNN model provides insights. Specifically, one-vs-one decision boundaries exist where examples



(a) (b) (c)

Figure 7: Confusion matrices with rows representing the initial labels and columns representing the shifted labels after attacking with the target models (a) robust Resnet18, (b) robust WideResnet-34-10, (c) robust Resnet152

of class $j \neq i$. In the confusion matrix, all off-diagonal elements are non-zero, as shown in Figure 5. Certain decision boundaries, such as those between classes 1 and 9, and classes 3 and 5, have relatively large surface areas. This increases the likelihood of examples from class 1 being misclassified as class 9, and examples from class 3 as class 5, and vice versa. Moreover, confusion matrices for different vanilla models exhibit similar patterns, indicating common decision boundary characteristics across architectures. This similarity underscores the feasibility of transferring adversarial perturbations between different DNN models. Notably, the confusion matrix for the Inception model shows some unique features. For example, the matrix is not symmetric, indicating that the decision boundary separating classes i and j is closer to the data of one class, say class i. Consequently, examples of class i are more likely to be misclassified as class j but not vice versa. This asymmetry significantly impacts the transferability of adversarial perturbations from other DNN models to Inception, a point we discuss further in the subsequent analysis.

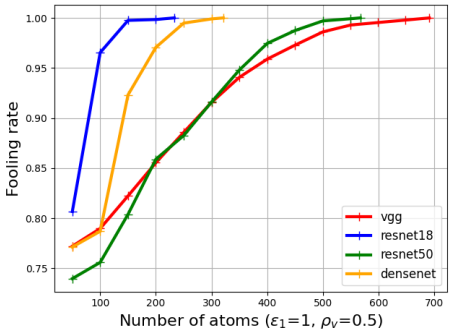


Figure 6: Fooling rate of attacking vanilla models on ImageNet

For robustly trained DNN models utilizing techniques such as data augmentation Sehwag et al. (2022), the one-vs-one decision boundaries remain largely unchanged. Consequently, the off-diagonal elements of the confusion matrix for robust DNN models are almost all non-zero. Specifically, examples from classes 2, 3, 4, and 5 are more susceptible to being attacked, constituting approximately 60% of the attacked examples. Furthermore, the adversarial space trained on vanilla models is a subspace of that trained on robust models. This can be verified by examining the performance of transferring the adversarial space to attack vanilla models, as shown in Table1.

Table 1: Transferable performance of ℓ_∞-attacks on CIFAR-10 $\epsilon = 8/255$ in terms of fooling rates (fr)

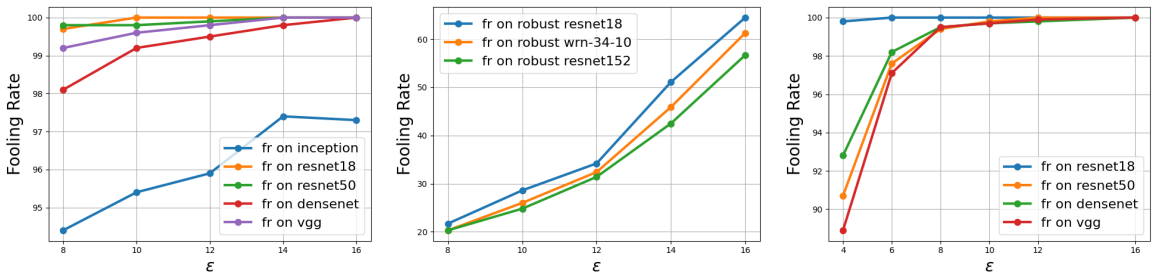
		inception	resnet18	ResNet50	DenseNet	VGG	R-r18	R-wrn-34-10	R-r152
Inception	Ours	94.6	99.1	99.7	98.3	99.1	20.7	19.9	18.7
resnet18	Ours	94.4	99.7	99.8	98.1	99.2	21.7	20.3	20.3
resnet50	Ours	92.5	99.5	99.8	98.6	99.1	22.0	18.2	19.3
densenet	Ours	94.5	99.7	99.8	98.9	99.1	22.6	20.8	19.3
vgg	Ours	93.5	99.5	99.5	98.5	99.4	20.8	19.8	19.9
robust-r18	Ours	92.1	99.1	99.8	98.2	99.2	31.7	27.6	25.8
robust-wrn-34-10	Ours	91.6	99.3	99.6	98.6	98.7	29.7	28.0	25.2
robust-r152	Ours	90.8	99.1	99.6	97.2	98.3	31.5	27.3	27.4
	AutoAttack	**100**	**100**	**100**	**100**	**100**	**35**	**31.5**	**28.6**
reference attacks	PGD	87.8	97.7	98.7	94.7	97.2	29.5	26.2	24.2

Table 2: Transferable performance of ℓ_∞-attacks on ImageNet $\epsilon = 4/255$ in terms of fooling rates (fr)

		resnet18	ResNet50	DenseNet	VGG
resnet18	Ours	99.8	90.7	92.8	88.9
resnet50	Ours	97.4	97.8	96.8	93.9
densenet	Ours	97.0	96.3	98.2	94.7
vgg	Ours	99.5	99.2	99.6	99.2
	AutoAttack	100	100	100	100
reference attacks	PGD	100	100	100	99.9

Model-agnostic adversarial space. The results regarding the transferability of the learned adversarial space for CIFAR-10 are presented in Table 1, and for ImageNet, they are in Table 2. Specifically, for CIFAR-10, the adversarial space learned using vanilla models demonstrates significant transferability across different architectures. For instance, the likelihood of finding an adversarial perturbation capable of deceiving ResNet18 within the space learned with ResNet50, DenseNet, VGG, or Inception exceeds 99%. Furthermore, adversarial spaces learned with vanilla models can also be utilized to search for adversarial perturbations that can attack robust models, achieving performance approximately 70% of that observed in spaces learned with robust models. Similarly, adversarial spaces learned with various robust models exhibit transferability among themselves. For instance, the success rate in fooling a robust ResNet18 model within its own adversarial space is comparable to that achieved in spaces learned with robust ResNet152 and slightly superior to those learned with robust WideResNet34-10 models. Notably, adversarial spaces learned with robust models are effective in attacking vanilla models as well. Consequently, it can be inferred that the adversarial space learned with vanilla models for CIFAR-10 is a subset of that learned with robust models. However, vulnerabilities persist within the same adversarial space, contingent upon the underlying data distribution. Similarly, these conclusions extend to the ImageNet dataset, where adversarial spaces learned with one vanilla model can effectively transfer to attack other models. Particularly, VGG demonstrates superior performance in generating adversarial spaces capable of attacking a majority of examples, comparable to state-of-the-art attack models.

As mentioned previously, a smaller ϵ_1 tends to result in an adversarial space of higher dimension, enhancing its capability to generalize to test data. Conversely, the adversarial



(a) (b) (c)

Figure 8: Trend in the fooling rate as a function of ϵ using a space trained on ResNet18, when attacking (a) vanilla models on CIFAR-10, (b) robust models on CIFAR-10, and (c) vanilla models on ImageNet.

space learned with a smaller ϵ includes the adversarial space learned with a larger ϵ. For example, on CIFAR-10, the fooling rate when attacking vanilla models (excluding Inception) increases to 100% when ϵ increase to $\epsilon = 16/255$. On ImageNet, this increase occurs only when $\epsilon = 10/255$ as shown in Figure 8. Even for robust models on CIFAR-10, the fooling rate reaches approximately 60 when $\epsilon = 16/255$.

Acknowledgements This work was partially supported by a grant from ANR (#ANR-20-CHIA-0021 Raimo to Stéphane Canu) and the grant 20E02165 from Normandy region.

References

N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 39–57, 2017.

Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks, 2020.

Alhussein Fawzi, Hamza Fawzi, and Omar Fawzi. Adversarial vulnerability for any classifier. *Advances in neural information processing systems*, 31, 2018.

Justin Gilmer, Luke Metz, Fartash Faghri, Samuel S Schoenholz, Maithra Raghu, Martin Wattenberg, Ian Goodfellow, and G Brain. The relationship between high-dimensional geometry and adversarial examples. *arXiv preprint arXiv:1801.02774*, 2018.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

Sicong Han, Chenhao Lin, Chao Shen, Qian Wang, and Xiaohong Guan. Interpreting adversarial examples in deep learning: A review. *ACM Computing Surveys*, 55(14s): 1–38, 2023.

Shihua Huang, Zhichao Lu, Kalyanmoy Deb, and Vishnu Naresh Boddeti. Revisiting residual networks for adversarial robustness. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8202–8211, 2023.

Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Logan Engstrom, Brandon Tran, and Aleksander Madry. Adversarial examples are not bugs, they are features. *Advances in neural information processing systems*, 32, 2019.

Ang Li, Yifei Wang, Yiwen Guo, and Yisen Wang. Adversarial examples are not real features. *Advances in Neural Information Processing Systems*, 36, 2024.

Yueru Li, Shuyu Cheng, Hang Su, and Jun Zhu. 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*, pages 753–769. Springer, 2020.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

Ambar Pal, Jeremias Sulam, and René Vidal. Adversarial examples might be avoidable: The role of data concentration in adversarial robustness. *Advances in Neural Information Processing Systems*, 36, 2024.

Sylvestre-Alvise Rebuffi, Sven Gowal, Dan Andrei Calian, Florian Stimberg, Olivia Wiles, and Timothy A Mann. Data augmentation can improve robustness. *Advances in Neural Information Processing Systems*, 34:29935–29948, 2021.

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 *ICLR*, 2022. URL https://openreview.net/forum?id=WVX0NNVBBkV.

Florian Tramèr, Nicolas Papernot, Ian J. Goodfellow, Dan Boneh, and Patrick Mcdaniel. The space of transferable adversarial examples. *ArXiv*, abs/1704.03453, 2017.

Xiaosen Wang and Kun He. Enhancing the transferability of adversarial attacks through variance tuning. In *Proceedings of the IEEE/CVF CVPR*, pages 1924–1933, 2021.

Lei Wu and Zhanxing Zhu. Towards understanding and improving the transferability of adversarial examples in deep neural networks. In *Asian Conference on Machine Learning*, pages 837–850. PMLR, 2020.

Yao Zhu, Jiacheng Sun, and Zhenguo Li. Rethinking adversarial transferability from a data distribution perspective. In *International Conference on Learning Representations*, 2021.