Adversarial Sample Detection Through Neural Network Transport Dynamics

Skander Karkar[1,2] [✉], Patrick Gallinari[1,2], and Alain Rakotomamonjy[1]

[1] Criteo AI Lab, Criteo, Paris, France
[2] MLIA, ISIR, Sorbonne Université, Paris, France
{as.karkar,p.gallinari,a.rakotomamonjy}@criteo.com

Abstract. We propose a detector of adversarial samples that is based on the view of neural networks as discrete dynamic systems. The detector tells clean inputs from abnormal ones by comparing the discrete vector fields they follow through the layers. We also show that regularizing this vector field during training makes the network more regular on the data distribution's support, thus making the activations of clean inputs more distinguishable from those of abnormal ones. Experimentally, we compare our detector favorably to other detectors on seen and unseen attacks, and show that the regularization of the network's dynamics improves the performance of adversarial detectors that use the internal embeddings as inputs, while also improving test accuracy.

Keywords: Deep learning · Adversarial detection · Optimal transport

1 Introduction

Neural networks have improved performances on many tasks, including image classification. They are however vulnerable to adversarial attacks which modify an image in a way that is imperceptible to a human but that fools the network into wrongly classifying the image [50]. These adversarial images transfer between networks [39], can be carried out physically (e.g. causing autonomous cars to misclassify road signs [15]), and can be generated without access to the network [34]. Developing networks that are robust to adversarial samples or accompanied by detectors that can detect them is indispensable to deploying them safely [3].

We focus on detecting adversarial samples. Networks trained with a softmax classifier produce overconfident predictions even for out-of-distribution inputs [42]. This makes it difficult to detect such inputs via the softmax outputs. A detector is a system capable of predicting if an input at test time has been adversarially modified. Detectors are trained on a dataset made up of clean and adversarial inputs, after the network training. While simply training the detector on the inputs has been tried, using their intermediate embeddings works better [9]. Detectors vary by which activations to use and how to process them to extract the features that the classifier uses to tell clean samples from adversarial ones.

We make two contributions. First, we propose an adversarial detector that is based on the view of neural networks as dynamical systems that move inputs

in space, time represented by depth, to separate them before applying a linear classifier [55]. Our detector follows the trajectory of samples in space, through time, to differentiate clean and adversarial images. The statistics that we extract are the positions of the internal embeddings in space approximated by their norms and cosines to a fixed vector. Given their resemblance to the Euler scheme for differential equations, residual networks [21, 22, 55] are particularly amenable to this analysis. Skip connections and residuals are basic building blocks in many architectures such as EfficientNet [51] and MobileNetV2 [47], and ResNets and their variants such as WideResNet [60] and ResNeXt [59] remain competitive [57]. Visions Transformers [35, 14] are also mainly made up of residual stages. Besides, [58] show an increased vulnerability of residual-type architectures to transferable attacks, precisely because of the skip connections. This motivates the need for a detector that is well adapted to residual-type architectures. But the analysis and implementation can extend immediately to any network where most layers have the same input and output dimensions.

Our second contribution is to use the transport regularization during training proposed in [26] to make the activations of adversarial samples more distinguishable from those of clean samples, thus making adversarial detectors perform better, while also improving generalization. We prove that the regularization achieves this by making the network more regular on the support of the data distribution. This does not necessarily make it more robust, but it will make the activations of the clean samples closer to each other and further from those of out-of-distribution samples, thus making adversarial detection easier. This is illustrated on a 2-dimension example in Figure 1.

2 Related Work

Given a classifier f in a classification task and $\epsilon>0$, an adversarial sample y constructed from a clean sample x is $y = x + \delta$, such that $f(y) \neq f(x)$ and $\|\delta\|_p \leq \epsilon$ for a certain L_p norm. The maximal perturbation size ϵ has to be so small as to be almost imperceptible to a human. Adversarial attacks are algorithms that find such adversarial samples, and they have been particularly successful against neural networks [50, 8]. We present the adversarial attacks we use in our experiments in Appendix D.1. The main defense mechanisms are robustness, i.e. training a network that is not easily fooled by adversarial samples, and having a detector of these samples.

An early idea for detection was to use a second network [38]. However, this network can also be adversarially attacked. More recent statistical approaches include LID [36], which trains the detector on the local intrinsic dimensionality of activations approximated over a batch, and the Mahalanobis detector [33], which trains the detector on the Mahalanobis distances between the activations and a Gaussian fitted to them during training, assuming they are normally distributed. Our detector is not a statistical approach and does not need batch-level statistics, nor statistics from the training data. Detectors trained in the Fourier domain of activations have also been proposed in [20]. See [1] for a review.

Our second contribution is to regularize the network in a way that makes it Hölder-continuous, but only on the data distribution's support. Estimations of the Lipschitz constant of a network have been used as estimates of its robustness to adversarial samples in [56, 50, 54, 23], and making the network more Lipschitz (e.g. by penalizing an upper bound on its Lipschitz constant) has been used to make it more robust (i.e. less likely to be fooled) in [23, 11]. These regularizations often work directly on the weights of the network, therefore making it more regular on all the input space. The difference with our method is that we only endue the network with regularity on the support of the clean data. This won't make it more robust to adversarial samples, but it makes its behavior on them more distinguishable, since they tend to lie outside the data manifold.

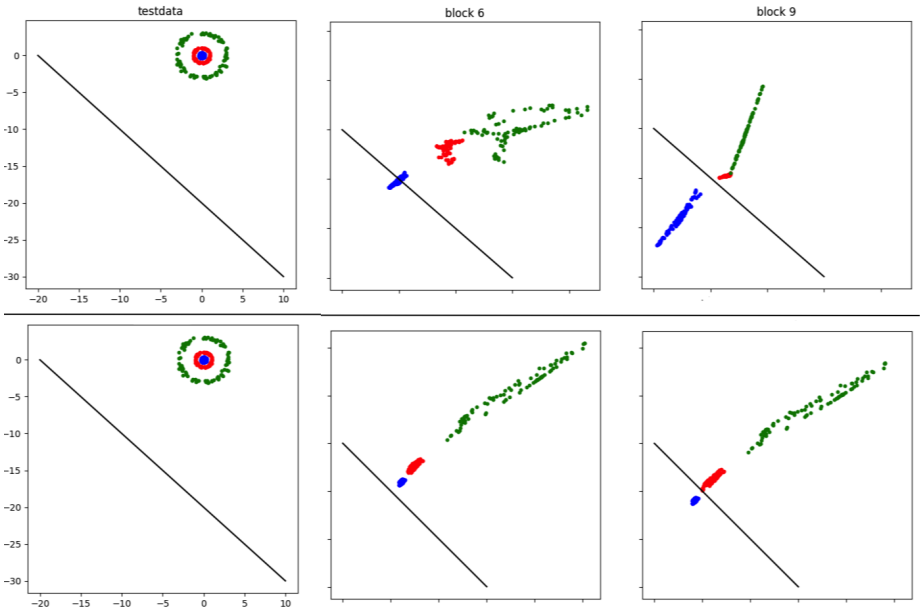

Fig. 1. Transformed circles test set from scikit-learn (red and blue) and out-of-distribution points (green) after blocks 6 and 9 of a small ResNet with 9 blocks. In the second row, we add our proposed regularization during training, which makes the movements of the clean points (red and blue) more similar to each other and more different from the movements of the green out-of-distribution points than when using the vanilla network in the first row. In particular, without the regularization, the green points are closer to the clean red points after blocks 6 and 9 which is undesirable.

That adversarial samples lie outside the data manifold, particularly in its co-dimensions, is a common observation and explanation for why adversarial samples are easy to find in high dimensions [18, 52, 49, 36, 46, 29, 2, 16]. To the best of our knowledge, [44] is the only other method that attempts to improve

detection by encouraging the network during training to learn representations that are more different between clean and adversarial samples. They do this by replacing cross-entropy by a reverse cross-entropy that encourages uniform softmax outputs among the non-predicted classes. We find that our regularization leads to better classification accuracy and adversarial detection than this method.

3 Background

Our detector is based on the dynamic viewpoint of neural networks that followed from the analogy between ResNets and the Euler scheme made in [55]. We present this analogy in Section 3.2. The regularization we use was proposed in [26] to improve generalization and we also present it in Section 3.2. The regularity results that follow from this regularization require the use of optimal transport theory, which we present in Section 3.1.

3.1 Optimal Transport

Let α and β be absolutely continuous densities on a compact set $\Omega \subset \mathbb{R}^d$. The Monge problem is to look for $T:\mathbb{R}^d \to \mathbb{R}^d$ moving α to β, i.e. $T_\sharp \alpha = \beta$, with minimal transport cost:

$$\min_{T \text{ s.t. } T_\sharp \alpha = \beta} \int_\Omega \|T(x) - x\|_2^2 \, \mathrm{d}\alpha(x) \tag{1}$$

and this problem has a unique solution $T^\star$. An equivalent formulation of the Monge problem in this setting is the dynamical formulation. Here, instead of directly pushing points from α to β through T, we continuously displace mass from time 0 to 1 according to velocity field $v_t : \mathbb{R}^d \to \mathbb{R}^d$. We denote ϕ_t^x the position at time t of the particle that was at $x \sim \alpha$ at time 0. This position evolves according to $\partial_t \phi_t^x = v_t(\phi_t^x)$. Rewriting the constraint, Problem (1) is equivalent to the dynamical formulation:

$$\begin{aligned} &\min_v \int_0^1 \|v_t\|^2_{L^2((\phi_t^\cdot)_\sharp \alpha)} \, \mathrm{d}t \\ &\text{s.t. } \partial_t \phi_t^x = v_t(\phi_t^x) \text{ for } x \in \text{support}(\alpha) \text{ and } t \in [0,1[\\ &\quad \phi_0^\cdot = \text{id}, (\phi_1^\cdot)_\sharp \alpha = \beta \end{aligned} \tag{2}$$

3.2 Least Action Principle Residual Networks

A residual stage made up of M residual blocks applies $x_{m+1} = x_m + h r_m(x_m)$ for $0 \leq m < M$, with x_0 being the input and h=1 in practice. The final point x_M is then classified by a linear layer F. The dynamic view considers a residual network as an Euler discretization of a differential equation:

$$x_{m+1} = x_m + h r_m(x_m) \iff \partial_t x_t = v_t(x_t) \tag{3}$$

where r_m approximates the vector field v_t at time $t = m/M$. The dynamic view allows to consider that ResNets are transporting their inputs in space by following a vector field to separate them, the depth representing time, before classification by a linear layer. [26] look for a network $F \circ T$ that solves the task while having minimal transport cost:

$$\begin{aligned} \inf_{T,F} \quad & \int_\Omega \|T(x) - x\|_2^2 \, \mathrm{d}\alpha(x) \\ \text{s.t.} \quad & \mathcal{L}(F, T_\sharp \alpha) = 0 \end{aligned} \tag{4}$$

where T is made up of the M residual blocks, α is the data distribution, F is the classification head and $\mathcal{L}(F, T_\sharp \alpha)$ is the (cross-entropy) loss obtained from classifying the transformed data distribution $T_\sharp \alpha$ through F. Given Section 3.1, the corresponding dynamical version of (4) is

$$\begin{aligned} \inf_{v,F} \quad & \int_0^1 \|v_t\|^2_{L^2((\phi_t^\cdot)_\sharp \alpha)} \, \mathrm{d}t \\ \text{s.t.} \quad & \partial_t \phi_t^x = v_t(\phi_t^x) \text{ for } x \in \text{support}(\alpha) \text{ and } t \in [0, 1[\\ & \phi_0^\cdot = \text{id}, \quad \mathcal{L}(F, (\phi_1^\cdot)_\sharp \alpha) = 0 \end{aligned} \tag{5}$$

[26] show that (4) and (5) are equivalent and have a solution such that T is an optimal transport map. In practice, (5) is discretized using a sample $\mathcal{D}$ from α and an Euler scheme, which gives a residual architecture with residual blocks r_m (parametrized along with the classifier by θ) that approximate v. This gives the following problem

$$\begin{aligned} \min_\theta \quad & \mathcal{C}(\theta) = \sum_{x \in \mathcal{D}} \sum_{m=0}^{M-1} \|r_m(\varphi_m^x)\|_2^2 \\ \text{s.t} \quad & \varphi_{m+1}^x = \varphi_m^x + h r_m(\varphi_m^x), \ \varphi_0^x = x \ \ \forall \ x \in \mathcal{D} \\ & \mathcal{L}(\theta) = 0 \end{aligned} \tag{6}$$

In practice, we solve Problem (6) using a method of multipliers (see Section 4.2). Our contribution is to show, theoretically and experimentally, that this makes adversarial examples easier to detect.

4 Method

We take the view that a ResNet moves its inputs through a discrete vector field to separate them, points in the same class having similar trajectories. Heuristically, for a successful adversarial sample that is close to clean samples, the vector field it follows has to be different at some step from that of the clean samples, so that it joins the trajectory of the points in another class. In Section 4.1, we present how we detect adversarial samples by considering these trajectories. In Section 4.2, we apply the transport regularization by solving (6) to improve detectability of adversarial samples.

4.1 Detection

Given a network that applies $x_{m+1} = x_m + hr_m(x_m)$ to an input x_0 for $0 \leq m < M$, we consider the embeddings x_m for $0 < m \leq M$, or the residues $r_m(x_m)$ for $0 \leq m < M$. To describe their positions in space, we take their norms and their cosine similarities with a fixed vector as features to train our adversarial detector on. Using only the norms already gave good detection accuracy. Cosines to other orthogonal vectors can be added to better locate the points at the price of increasing the number of features. We found that using only one vector already gives state-of-the-art detection, so we only use the norms and cosines to a fixed vector of ones. We train the detector (a random forest in practice, see Section 5.2) on these features. The embeddings x_m and the residues $r_m(x_m)$ can equivalently describe the trajectory of x_0 in space through the blocks. In practice, we use the residues $r_m(x_m)$, with their norms squared and averaged. So the feature vector given to the random forest for each input x_0 that goes through a network that applies $x_{m+1} = x_m + h\ r_m(x_m)$ is

$$\left(\frac{1}{d_m}\|r_m(x_m)\|_2^2,\ \cos\Big(r_m(x_m), \mathbf{1}_m\Big)\right)_{0 \leq m < M} \tag{7}$$

and the label is 0 if x_0 is clean and 1 if it is adversarial. Here cos is the cosine similarity between two vectors and $\mathbf{1}_m$ is a vector of ones of size d_m where d_m is the size of $r_m(x_m)$. For any non-residual architecture $x_{m+1} = g_m(x_m)$, the vector $x_{m+1} - x_m$ can be used instead of $r_m(x_m)$ on layers that have the same input and output dimension, allowing to apply the method to any network with many such layers. And we do test the detector on a ResNeXt, which does not fully satisfy the dynamic view, as the activation is applied after the skip-connection, i.e. $x_{m+1} = \mathrm{ReLU}(x_m + h\ r_m(x_m))$.

The number of features is twice that of residual blocks (a norm and a cosine per block). This is of the same order as for other popular detectors such as Mahalanobis [33] and LID [36] that extract one feature per residual stage (a residual stage is a group of blocks that keep the same dimension). Even for common large architectures, twice the number of residual blocks is still a small number of features for training a binary classifier (ResNet152 has 50 blocks). More importantly, the features we extract (norms and cosines) are quick to calculate, whereas those of other methods require involved statistical computations on the activations. We include in Appendix D.10 a favorable time comparison of our detector to the Mahalanobis detector. Another advantage is that our detector does not have a hyper-parameter to tune unlike the Mahalanobis and LID detectors.

4.2 Regularization

Regularity of neural networks (typically Lipschitz continuity) has been used as a measure of their robustness to adversarial samples [56, 50, 54, 23, 11]. Indeed, the smaller the Lipschitz constant L of a function f satisfying $\|f(x) - f(y)\| \leq L\|x - y\|$, the less f changes its output $f(y)$ for a perturbation (adversarial or

not) y of x. Regularizing a network to make it more Lipschitz and more robust has therefore been tried in [23] and [11]. For this to work, the regularization has to apply to adversarial points, i.e. outside the support of the clean data distribution. Indeed, the Lipschitz continuity obtained though most of these methods and analyses apply on the entire input space $\mathbb{R}^d$ as they penalize the network's weights directly. Likewise, a small step size h as in [62] will have the same effect on all inputs, clean or not.

We propose here an alternative approach where we regularize the network only on the support of the input distribution, making it η-Hölder on this support (a function f is η-Hölder on X if $\forall\ a, b \in X$, we have $\|f(a) - f(b)\| \leq C\|a - b\|^\eta$ for some constants $C{>}0$ and $0{<}\eta{\leq}1$, and we denote this $f \in \mathcal{C}^{0,\eta}(X)$). Since this result does not apply outside the input distribution's support, particularly in the adversarial spaces, then this regularity that only applies to clean samples can serve to make adversarial samples more distinguishable from clean ones, and therefore easier to detect. We show experimentally that the behavior of the network will be more distinguishable between clean and adversarial samples in practice in Section 5.1. We discuss the implementation of the regularization in Section 4.2 and prove the regularity it endues the network with in Section 4.2.

Implementation. We regularize the trajectory of the samples by solving Problem (6). This means finding, among the networks that solve the task (condition $\mathcal{L}(\theta) = 0$ in (6)), the network that moves the points the least, that is the one with minimal kinetic energy $\mathcal{C}$. The residual functions r_m we find are then our approximation of the vector field v that solves the continuous version (5) of Problem (6).

We solve Problem (6) via a method of multipliers: since $\mathcal{L} \geq 0$, Problem (6) is equivalent to the min-max problem $\min_\theta \max_{\lambda>0}\ \mathcal{C}(\theta) + \lambda\ \mathcal{L}(\theta)$, which we solve, given growth factor $\tau > 0$, and starting from initial weight given to the loss λ_0 and initial parameters θ_0, through

$$\begin{cases} \theta_{i+1} = \arg\min_\theta\ \mathcal{C}(\theta) + \lambda_i\ \mathcal{L}(\theta) \\ \lambda_{i+1} = \lambda_i + \tau\ \mathcal{L}(\theta_{i+1}) \end{cases} \tag{8}$$

We use SGD for $s{>}0$ steps (i.e. batches) for the minimization in the first line of (8), starting from the previous θ_i. When using a ResNeXt, where a residual block applies $x_{m+1} = \mathrm{ReLU}(x_m + r_m(x_m))$, we regularize the norms of the true residues $x_{m+1}{-}x_m$ instead of $r_m(x_m)$.

Theoretical Analysis. We take $\Omega{\subset}\mathbb{R}^d$ convex and compact and the data distribution $\alpha{\in}\mathcal{P}(\Omega)$ absolutely continuous such that $\delta\Omega$ is α-negligible. We suppose that there exists an open bounded convex set $X{\subset}\Omega$ such that α is bounded away from zero and infinity on X and is zero on $X^\complement$. From [26], Problems (4) and (5) are equivalent and have solutions (T, F) and (v, F) such that T is an optimal transport map between α and $\beta{:=}T_\sharp\alpha$. We suppose that β is absolutely continuous and that there exists an open bounded convex set $Y{\subset}\Omega$ such that β

is bounded away from zero and infinity on Y and is zero on $Y^{\complement}$. In the rest of this section, v solves (5) and we suppose that we find a solution to the discretized problem (6) that is an $\varepsilon/2$-approximation of v, i.e. $\|r_m - v_{t_m}\|_\infty \leq \varepsilon/2$ for all $0 \leq m < M$, with $t_m = m/M$.

Definition 1. *A function f is η-Hölder on X if $\forall\ a, b \in X$, we have $\|f(a) - f(b)\| \leq C\|a - b\|^\eta$ for some constants $C > 0$ and $0 < \eta \leq 1$. We denote this $f \in \mathcal{C}^{0,\eta}(X)$.*

In Theorem 1, we show that the regularization makes the residual blocks of the network η-Hölder (with an error of ε) on the support of the input distribution as it moves according to the theoretical vector field solution v. The results hold for all norms on $\mathbb{R}^d$.

Theorem 1. *For $a, b \in$ support(α_{t_m}), $\alpha_t := (\phi_t^{\cdot})_\sharp \alpha$ where ϕ solves (5) along with v, we have*

$$\|r_m(a) - r_m(b)\| \leq \varepsilon + K\|a - b\|^{\zeta_1} \text{ if } \|a - b\| \leq 1$$
$$\|r_m(a) - r_m(b)\| \leq \varepsilon + K\|a - b\|^{\zeta_2} \text{ if } \|a - b\| > 1$$

for constants $K > 0$ and $0 < \zeta_1 \leq \zeta_2 \leq 1$.

Proof. The detailed proof is in Appendix C.1. First, we have that $v_t = (T - \mathtt{id}) \circ T_t^{-1}$ where $T_t := (1 - t)\mathtt{id} + tT$ and T solves (4). Being an optimal transport map, T is η-Hölder. So for all $a, b \in \text{support}(\alpha_t)$ and $t \in [0, 1[$, where $\alpha_t = (\phi_t^{\cdot})_\sharp \alpha = (T_t)_\sharp \alpha$ with ϕ solving (5) with v, we have

$$\|v_t(a) - v_t(b)\| \leq \|T_t^{-1}(a) - T_t^{-1}(b)\| + C\|T_t^{-1}(a) - T_t^{-1}(b)\|^\eta \tag{9}$$

We then show that T_t^{-1} is an optimal transport map and so is η_t-Hölder with $0 < \eta_t \leq 1$. Using the hypothesis on r and the triangle inequality, we get, for all $a, b \in \text{support}(\alpha_{t_m})$

$$\|r_m(a) - r_m(b)\| \leq\ \varepsilon + C_{t_m}\|a - b\|^{\eta_{t_m}} + CC_{t_m}^{\eta}\|a - b\|^{\eta\eta_{t_m}} \tag{10}$$

Then set the constants K, ζ_1 and ζ_2 as necessary.

We use Theorem 1 to now bound the distance between the residues at depth m as a function of the distance between the network's inputs. For inputs a_0 and b_0 to the network, the intermediate embeddings are $a_{m+1} = a_m + hr_m(a_m)$ and $b_{m+1} = b_m + hr_m(b_m)$, and the residues used to compute features for adversarial detection are $r_m(a_m)$ and $r_m(b_m)$. So we want to bound $\|r_m(a_m) - r_m(b_m)\|$ as a function of $\|a_0 - b_0\|$. This is usually done by multiplying the Lipschitz constants of each block up to depth m, which leads to an overestimation [24], or through more complex estimation algorithms [54, 32, 6]. Bound (9) allows through T_t^{-1} to avoid multiplying the Hölder constants of the blocks. If a_0 and b_0 are on the clean data support X, we get Theorem 2 below with proof in Appendix C.2.

Theorem 2. *For $a_0, b_0 \in X$ and constants $C, L>0$,*

$$\|r_m(a_m) - r_m(b_m)\| \leq \varepsilon + \|a_0 - b_0\| + C\|a_0 - b_0\|^\eta + \\ + L(\|a_m - \phi_{t_m}^{a_0}\| + \|b_m - \phi_{t_m}^{b_0}\|)$$

Term $\mu(a_0) := \|a_m - \phi_{t_m}^{a_0}\|$ (and $\mu(b_0) := \|b_m - \phi_{t_m}^{b_0}\|$) is the distance between the point a_m after m residual blocks and the point $\phi_{t_m}^{a_0}$ we get by following the theoretical solution vector field v up to time t_m starting from a_0. If a_0 and b_0 are not on the data support X, an extra term has to be introduced to use bound (9). Bounding the terms $\mu(a_0)$ and $\mu(b_0)$ is possible under more regularity assumptions on v. We assume then that v is $\mathcal{C}^1$ and Lipschitz in x, which is not stronger than the regularity we get on v through our regularization, as it does not give a similar result to bound (9). We have for all inputs a_0 and b_0, whether they are clean or not, Theorem 3 below with proof in Appendix C.2.

Theorem 3. *For $a_0, b_0 \in \mathbb{R}^d$ and constants $R, S > 0$,*

$$\|r_m(a_m) - r_m(b_m)\| \leq \varepsilon + LS\varepsilon + LSRh + \|a_0 - b_0\| + C\|a_0 - b_0\|^\eta + \\ + LS(\mathit{dist}(a_0, X) + \mathit{dist}(b_0, X))$$

Terms $\mathrm{dist}(a_0, X)$ and $\mathrm{dist}(b_0, X)$ show that the regularity guarantee is increased for inputs in X. The trajectories of clean points are then closer to each other and more different from those of abnormal samples outside X.

5 Experiments

We evaluate our method on adversarial samples found by 8 attacks. The threat model is as follows. We use 6 white-box attacks that can access the network and its weights and architecture but not its training data: FGM [19], BIM [31], DF [40], CW [8], AutoAttack (AA) [13] and the Auto-PGD-CE (APGD) variant of PGD [37], and 2 black-box attacks that only query the network: HSJ [10] and BA [7]. We assume the attacker has no knowledge of the detector and use the untargeted (i.e. not trying to direct the mistake towards a particular class) versions of the attacks. We use a maximal perturbation of ϵ=0.03 for FGM, APGD, BIM and AA. We use the L_2 norm for CW and HSJ and L_∞ for the other attacks. We compare our detector (which we call the Transport detector or TR) to the Mahalanobis detector (MH in the tables below) of [33] and to the detector of [28, 27] that uses natural scene statistics (NS in the tables below), and our regularization to reverse cross entropy training of [44], which is also meant to improve detection of adversarial samples. We use ART [43] and its default hyperparameter values (except those specified) to generate the adversarial samples, except for AA for which we use the authors' original code. The code is available at `github.com/skander-karkar/adv`. See Appendix D.1 for more details.

We use 3 networks and datasets: ResNeXt50 on CIFAR100, ResNet110 on CIFAR10 and WideResNet on TinyImageNet. Each network is trained normally

with cross entropy, with the transport regularization added to cross entropy (called a LAP-network for Least Action Principle), and with reverse cross entropy instead of cross entropy (called an RCE-network). For LAP training, we use (8) with τ=1, s=1 and λ_0=1 for all networks. These hyper-parameters are chosen to improve validation accuracy during training not adversarial detection. Training details are in Appendix D.2.

In Section 5.1, we conduct preliminary experiments to show that LAP training improves generalization and stability, and increases the difference between the transport costs of clean and adversarial samples. In Section 5.2, we test our detector when it is trained and tested on samples generated by the same attack. In Section 5.3, we test our detector when it is trained on samples generated by FGM and tested on samples from the other attacks. We then consider OOD detection and adaptive attacks on the detector.

5.1 Preliminary Experiments

Our results confirm those in [26] that show that LAP training improves test accuracy. Vanilla ResNeXt50 has an accuracy of 74.38% on CIFAR100, while LAP-ResNeXt50 has an accuracy of 77.2%. Vanilla ResNet110 has an accuracy of 92.52% on CIFAR10, while LAP-ResNet110 has an accuracy of 93.52% and the RCE-ResNet110 of 93.1%. Vanilla WideResNet has an accuracy of 65.14% on TinyImageNet, while LAP-WideResNet has an accuracy of 65.34%. LAP training is also more stable by allowing to train deep networks without batch-normalization in Figure 4 in Appendix D.4.

We see in Figure 2 that LAP training makes the transport cost $\mathcal{C}$ more different between clean and adversarial points. Using its empirical quantiles on clean points allows then to detect samples from some attacks with high recall and a fixed false positive rate, without seeing adversarial samples.

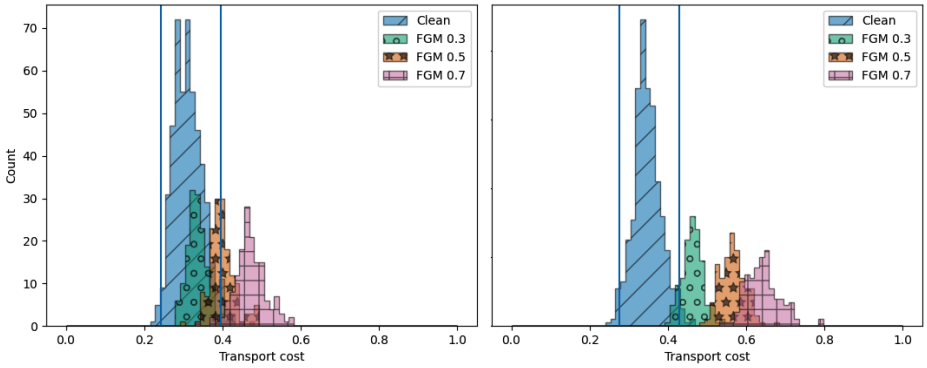

Fig. 2. Histogram of transport cost $\mathcal{C}$ for clean and FGM-attacked test samples with different values of ϵ on CIFAR100. The vertical lines represent the 0.02 and 0.98 empirical quantiles of the transport cost of the clean samples. Left: ResNeXt50. Right: LAP-ResNeXt50.

5.2 Detection of Seen Attacks

For detection training, the test set is split in 0.9/0.1 proportions into two datasets, B1 and B2. For each image in B1 (respectively B2), an adversarial sample is generated and a balanced detection training set (respectively a detection test set) is created. Since adversarial samples are created for a specific network, this is done for the vanilla version of the network and its LAP and RCE versions. We tried augmenting the detection training dataset with a randomly perturbed version of each image, to be considered clean during detection training, as in [33], but we found that this does not improve detection accuracy. This dataset creation protocol is standard and is depicted in Figure 3 in Appendix D.3. We did not limit the datasets to successfully attacked images only as in [33], as we consider the setting of detecting all adversarial samples, whether or not they fool the network, more challenging (which is seen in the results). It also allows to detect any attempted interference with the network, even if it fails at fooling it.

Samples in the detection training set are fed through the network and the features for each detector are extracted. We tried three classifiers (logistic regression, random forest and SVM) trained on these features for all detectors, and kept the random forest as it always performs best. We tried two methods to improve the accuracy of all detectors: class-conditioning and ensembling. In class-conditioning, the features are grouped by the class predicted by the network, and a detector is trained for every class. At test time, the detector trained on the features of the predicted class is used. A detector is also trained on all samples regardless of the predicted class and is used in case a certain class is never targeted by the attack. We also tried ensembling the class-conditional detector with the general all-class detector: an input is considered an attack if at least one detector says so. This ensemble of the class-conditional detector and the general detector performs best for all detectors, and is the one we use.

We report the accuracy of each detector on the detection test set for both the vanilla and the LAP network in Table 1. In each cell, the first number corresponds to the vanilla network and the second to the regularized LAP-network. Since the NS detector takes the image and not its embeddings as input, the impact of LAP and RCE training on its performance is minimal and we report its performance on the vanilla network only. These results are averaged over 5 runs and the standard deviations (which are tight) are in Tables 3 to 7 in Appendix D.5, along with results on RCE-networks. Since some attacks are slow, we don't test them on all network-dataset pairs in this experiment. Results in Table 1 show two things. First, our detector performs better than both other detectors, with or without the regularization. Second, both the TR and MH detectors work better on the LAP-networks most times. The MH detector benefits more from the regularization, but on all attacks, the best detector is always the Transport detector. In the tables in Appendix D.5, RCE often improves detection accuracy in this experiment, but clearly less than LAP training. On CIFAR10, our detector outperforms the MH detector by 9 to 16 percentage points on the vanilla ResNet110, and the NS detector by up to 5 points. LAP training improves the accuracy of our detector by an average 1.5 points and that of the MH detector by a substantial

8.3 points on average. On CIFAR100, our detector outperforms the MH detector by 1 to 5 points on the vanilla ResNeXt50, and the NS detector by up to 3 points. LAP training improves the accuracy of both detectors by an average 1 point. On TinyImageNet, our detector greatly outperforms the MH detector by 3 to 15 points on the vanilla WideResNet, and the NS detector slightly. LAP training does not change the accuracy of our detector and improves that of the MH detector by 0.85 points on average. Detection rates of successful adversarial samples (i.e. those that fool the network) are in Table 14 in Appendix D.7 and are higher than 95% on our detector. False positive rates (positive meaning adversarial) are in Table 16 in Appendix D.8 and are always less than 5% on our detector. The AUROC is in Table 18 in Appendix D.9. On all these metrics, our detector outperforms the other detectors largely, and LAP-training greatly improves the performance of the Mahalanobis detector.

Table 1: Average accuracy of detectors on adversarial samples from seen attacks on Network/LAP-Network over 5 runs.

Attack	Detector	ResNet110 CIFAR10	ResNeXt50 CIFAR100	WideResNet TinyImageNet
FGM	TR	97.14/**98.70**	97.26/**98.32**	**95.36**/95.14
	MH	87.78/95.64	95.82/96.82	81.06/85.26
	NS	94.56	94.70	94.90
APGD	TR	94.10/**97.50**	96.04/**97.84**	**95.22**/95.20
	MH	82.08/90.70	93.94/94.60	79.66/85.10
	NS	94.28	94.18	94.86
BIM	TR	97.54/**99.28**	98.02/**98.92**	**95.26**/95.12
	MH	86.78/95.38	96.06/97.76	81.20/82.46
	NS	95.04	94.72	95.00
AA	TR	88.88/**94.08**	84.90/**87.56**	**81.38**/81.24
	MH	80.46/89.96	83.90/86.58	78.40/78.40
	NS	88.78	84.82	81.32
DF	TR	**99.98**/99.84	**99.80**/99.58	
	MH	91.50/96.70	97.30/97.12	
	NS	99.78	99.6	
CW	TR	**98.04**/97.96	97.04/**97.80**	
	MH	85.58/93.36	95.38/96.42	
	NS	93.86	90.7	
HSJ	TR	**99.94**/99.92		
	MH	85.50/94.56		
	NS	99.68		
BA	TR	96.56/**97.02**		
	MH	80.20/89.62		
	NS	92.10		

5.3 Detection of Unseen Attacks

An important setting is when we don't know which attack might be used or only have time to train detectors on samples from one attack. We still want our detector to generalize well to unseen attacks. To test this, we use the same vanilla networks as above but the detectors are now trained on the detection training set created by the simplest and quickest attack (FGM) and tested on the detection test sets created by the other attacks. Results are in Table 2. We see that our detector has very good generalization to unseen attacks, even those very different from FGM, comfortably better than the MH detector, by up to 19 percentage points, while the NS detector only generalizes to variants of FGM (APGD and BIM), and fails on the other attacks. These results are averaged over 5 runs and the standard deviations are in Tables 8 to 13 in Appendix D.6. On our detector, the detection rate of successful adversarial samples remains higher than 90% in most cases (Table 15 in Appendix D.7) and the FPR is always lower than 10% (Table 17 in Appendix D.8). The AUROC is in Table 19 in Appendix D.9. Our detector almost always outperforms the other detectors on all these metrics.

Table 2: Average accuracy of detectors on samples from unseen attacks after training on FGM over 5 runs.

Attack	Detector	ResNet110 CIFAR10	ResNeXt50 CIFAR100	WideResNet TinyImageNet
APGD	TR	89.32	91.94	93.26
	MH	77.34	90.86	76.96
	NS	**92.08**	**92.16**	**94.06**
BIM	TR	**96.02**	**95.02**	**94.66**
	MH	77.24	93.16	77.02
	NS	93.88	93.88	94.62
AA	TR	**85.10**	**73.32**	**77.04**
	MH	72.12	73.08	60.36
	NS	51.82	51.32	65.60
DF	TR	**91.02**	**85.16**	**90.62**
	MH	80.12	82.72	73.18
	NS	51.40	51.62	72.82
CW	TR	**93.18**	**78.18**	**91.42**
	MH	79.92	76.44	75.52
	NS	50.84	51.02	71.96
HSJ	TR	**93.00**	**85.04**	
	MH	79.70	82.82	
	NS	52.12	52.04	
BA	TR	**90.92**	**92.14**	
	MH	79.32	84.46	
	NS	59.88	57.90	

However, this experiment shows that our regularization has some limitations. We see in Tables 8 to 13 in Appendix D.6 that LAP training does not improve detection accuracy as much, and sometimes reduces it. It still improves it for the MH detector on all attacks on ResNet110 and WideResNet by up to 10 points, and LAP training still always does better than RCE training. We claim this is because these methods reduce the variance of features extracted on the seen attack, harming generalization to unseen attacks. This explains why detection of APGD and BIM, variants of FGM, improves.

5.4 Detection of Out-Of-Distribution Samples

Since our analysis applies to all out-of-distribution (OOD) samples, we test detection of OOD samples in a similar setting to [33]. We train a model on a first dataset (ResNet110 on CIFAR10 and ResNeXt50 on CIFAR100), then train detectors to tell this first dataset from a second dataset (which can be an adversarially attacked version of the first dataset), then test their ability to tell the first dataset from a third unseen dataset (SVHN). Our detector does very well and better than the MH detector on both experiments, and detection accuracy of samples from the unseen distribution is higher than 90% when using the CW attack to create the second dataset. Details are in Appendix D.11.

5.5 Attacking the Detector

We consider the case where the detector is also attacked (adaptive attacks). We try 2 attacks on the TR and MH detectors. Both are white-box with respect to the network. The first is black-box with respect to the detector and only knows if a sample has been detected or not. The second has some knowledge about the detector. It knows what features it uses and can attack it directly to find adversarial features. We test these attacks by looking at the percentage of detected successful adversarial samples that they turn into undetected successful adversarial samples. For the first attack, this is 6.8% for our detector and 12.9% for the MH detector on the LAP-ResNet110, and is lowered by LAP training. For the second attack it is 14% on our detector. Given that detection rates of successful adversarial samples are almost 100% (see Appendix D.7), this shows that an adaptive attack does not circumvent the detector, as detection rates drop to 85% at worst. Details are in Appendix D.12.

6 Conclusion

We proposed a method for detecting adversarial samples, based on the dynamical view of neural networks. The method examines the discrete vector field moving the inputs to distinguish clean and abnormal samples. The detector requires minimal computation to extract the features it uses for detection and achieves state-of-the-art detection accuracy on seen and unseen attacks. We also use a transport regularization that both improves test classification accuracy and the accuracy of adversarial detectors.

Ethical Statement

Adversarial detection and robustness are essential to safely deploy neural networks that attackers might target for nefarious purposes. But adversarial attacks can be used to evade neural networks that are deployed for nefarious purposes.

References

1. Aldahdooh, A., Hamidouche, W., Fezza, S.A., Déforges, O.: Adversarial example detection for dnn models: a review and experimental comparison. Artificial Intelligence Review (2022)
2. Alemany, S., Pissinou, N.: The dilemma between data transformations and adversarial robustness for time series application systems. In: Proceedings of the Workshop on Artificial Intelligence Safety 2022 (SafeAI 2022) co-located with the Thirty-Sixth AAAI Conference on Artificial Intelligence (AAAI2022), Virtual, February, 2022. CEUR Workshop Proceedings, vol. 3087. CEUR-WS.org (2022)
3. Amodei, D., Olah, C., Steinhardt, J., Christiano, P., Schulman, J., Mané, D.: Concrete problems in ai safety (2016). https://doi.org/10.48550/ARXIV.1606.06565, https://arxiv.org/abs/1606.06565
4. Andriushchenko, M., Croce, F., Flammarion, N., Hein, M.: Square attack: a query-efficient black-box adversarial attack via random search. In: Computer Vision – ECCV 2020. Springer (2020)
5. Benamou, J., Brenier, Y.: A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. Numerische Mathematik (2000)
6. Bhowmick, A., D'Souza, M., Raghavan, G.S.: Lipbab: Computing exact lipschitz constant of relu networks. CoRR **abs/2105.05495** (2021), https://arxiv.org/abs/2105.05495
7. Brendel, W., Rauber, J., Bethge, M.: Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=SyZI0GWCZ
8. Carlini, N., Wagner, D.: Towards evaluating the robustness of neural networks. In: 2017 IEEE Symposium on Security and Privacy (SP). pp. 39–57. IEEE Computer Society, Los Alamitos, CA, USA (may 2017). https://doi.org/10.1109/SP.2017.49, https://doi.ieeecomputersociety.org/10.1109/SP.2017.49
9. Carlini, N., Wagner, D.: Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods, p. 3–14. Association for Computing Machinery, New York, NY, USA (2017), https://doi.org/10.1145/3128572.3140444
10. Chen, J., Jordan, M.I., Wainwright, M.J.: Hopskipjumpattack: A query-efficient decision-based attack. In: 2020 IEEE Symposium on Security and Privacy. pp. 1277–1294. IEEE (2020), https://doi.org/10.1109/SP40000.2020.00045
11. Cisse, M., Bojanowski, P., Grave, E., Dauphin, Y., Usunier, N.: Parseval networks: Improving robustness to adversarial examples. In: Proceedings of the 34th International Conference on Machine Learning. p. 854–863. PMLR (2017)
12. Croce, F., Hein, M.: Minimally distorted adversarial examples with a fast adaptive boundary attack. In: Proceedings of the 37th International Conference on Machine Learning. PMLR (2020)
13. Croce, F., Hein, M.: Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In: Proceedings of the 37th International Conference on Machine Learning. PMLR (2020)

14. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021), https://openreview.net/forum?id=YicbFdNTTy
15. Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., Song, D.: Robust physical-world attacks on deep learning visual classification. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1625–1634. IEEE Computer Society, Los Alamitos, CA, USA (jun 2018). https://doi.org/10.1109/CVPR.2018.00175, https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00175
16. Feinman, R., Curtin, R.R., Shintre, S., Gardner, A.B.: Detecting adversarial samples from artifacts (2017)
17. Figalli, A.: The Monge-Ampere Equation and Its Applications. Zurich lectures in advanced mathematics, European Mathematical Society (2017)
18. Gilmer, J., Metz, L., Faghri, F., Schoenholz, S.S., Raghu, M., Wattenberg, M., Goodfellow, I.: Adversarial spheres: The relationship between high-dimensional geometry and adversarial examples (2018), https://arxiv.org/abs/1801.02774
19. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: ICLR (2015)
20. Harder, P., Pfreundt, F.J., Keuper, M., Keuper, J.: Spectraldefense: Detecting adversarial attacks on cnns in the fourier domain (2021). https://doi.org/10.48550/ARXIV.2103.03000, https://arxiv.org/abs/2103.03000
21. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
22. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks. In: ECCV (2016)
23. Hein, M., Andriushchenko, M.: Formal guarantees on the robustness of a classifier against adversarial manipulation. In: Advances in Neural Information Processing Systems. p. 2263–2273. Curran Associates Inc., Red Hook, NY, USA (2017)
24. Huster, T., Chiang, C.Y.J., Chadha, R.: Limitations of the lipschitz constant as a defense against adversarial examples. In: ECML PKDD 2018 Workshops. pp. 16–29. Springer International Publishing (2019)
25. Kantchelian, A., Tygar, J.D., Joseph, A.D.: Evasion and hardening of tree ensemble classifiers. In: Proceedings of the 33th International Conference on Machine Learning. PMLR (2016)
26. Karkar, S., Ayed, I., de Bézenac, E., Gallinari, P.: A principle of least action for the training of neural networks. In: ECML-PKDD (2020)
27. Kherchouche, A., Fezza, S.A., Hamidouche, W.: Detect and defense against adversarial examples in deep learning using natural scene statistics and adaptive denoising. Neural Computing and Applications **34**(24), 21567–21582 (Dec 2022). https://doi.org/10.1007/s00521-021-06330-x, https://hal.science/hal-03330258
28. Kherchouche, A., Fezza, S.A., Hamidouche, W., Déforges, O.: Detection of adversarial examples in deep neural networks with natural scene statistics. In: 2020 International Joint Conference on Neural Networks (IJCNN). pp. 1–7 (2020). https://doi.org/10.1109/IJCNN48605.2020.9206959
29. Khoury, M., Hadfield-Menell, D.: On the geometry of adversarial examples (2018), https://arxiv.org/abs/1811.00525

30. Krizhevsky, A.: Learning multiple layers of features from tiny images. University of Toronto Technical Report (2009)
31. Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial examples in the physical world. In: ICLR (Workshop) (2017)
32. Latorre, F., Rolland, P., Cevher, V.: Lipschitz constant estimation of neural networks via sparse polynomial optimization. In: International Conference on Learning Representations (2020), https://openreview.net/forum?id=rJe4_xSFDB
33. Lee, K., Lee, K., Lee, H., Shin, J.: A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018), https://proceedings.neurips.cc/paper/2018/file/abdeb6f575ac5c6676b747bca8d09cc2-Paper.pdf
34. Liu, Y., Chen, X., Liu, C., Song, D.: Delving into transferable adversarial examples and black-box attacks. In: International Conference on Learning Representations. OpenReview.net (2017), https://openreview.net/forum?id=Sys6GJqxl
35. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2021)
36. Ma, X., Li, B., Wang, Y., Erfani, S.M., Wijewickrema, S., Schoenebeck, G., Song, D., Houle, M.E., Bailey, J.: Characterizing adversarial subspaces using local intrinsic dimensionality. In: ICLR (2018)
37. Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards deep learning models resistant to adversarial attacks. In: ICLR (2018), https://openreview.net/forum?id=rJzIBfZAb
38. Metzen, J.H., Genewein, T., Fischer, V., Bischoff, B.: On detecting adversarial perturbations. In: ICLR (2017)
39. Moosavi-Dezfooli, S., Fawzi, A., Fawzi, O., Frossard, P.: Universal adversarial perturbations. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 86–94. IEEE Computer Society, Los Alamitos, CA, USA (2017). https://doi.org/10.1109/CVPR.2017.17, https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.17
40. Moosavi-Dezfooli, S.M., Fawzi, A., Frossard, P.: Deepfool: A simple and accurate method to fool deep neural networks. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 2574–2582 (2016)
41. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NIPS Workshop on Deep Learning and Unsupervised Feature Learning (2011)
42. Nguyen, A., Yosinski, J., Clune, J.: Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 427–436 (2015). https://doi.org/10.1109/CVPR.2015.7298640
43. Nicolae, M.I., Sinn, M., Tran, M.N., Buesser, B., Rawat, A., Wistuba, M., Zantedeschi, V., Baracaldo, N., Chen, B., Ludwig, H., Molloy, I.M., Edwards, B.: Adversarial robustness toolbox v1.0.0 (2018). https://doi.org/10.48550/ARXIV.1807.01069, https://arxiv.org/abs/1807.01069
44. Pang, T., Du, C., Dong, Y., Zhu, J.: Towards robust detection of adversarial examples. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018)

45. Quarteroni, A., Sacco, R., Saleri, F.: Numerical Mathematics. Springer Berlin, Heidelberg (2007)
46. Samangouei, P., Kabkab, M., Chellappa, R.: Defense-GAN: Protecting classifiers against adversarial attacks using generative models. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=BkJ3ibb0-
47. Sandler, M., Howard, A.G., Zhu, M., Zhmoginov, A., Chen, L.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: 2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018. pp. 4510–4520. Computer Vision Foundation / IEEE Computer Society (2018)
48. Santambrogio, F.: Optimal Transport for Applied Mathematicians. Birkhäuser (2015)
49. Song, Y., Kim, T., Nowozin, S., Ermon, S., Kushman, N.: Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=rJUYGxbCW
50. Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., Fergus, R.: Intriguing properties of neural networks (2013). https://doi.org/10.48550/ARXIV.1312.6199, https://arxiv.org/abs/1312.6199
51. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 6105–6114. PMLR (09–15 Jun 2019), https://proceedings.mlr.press/v97/tan19a.html
52. Tanay, T., Griffin, L.: A boundary tilting persepective on the phenomenon of adversarial examples (2016)
53. Villani, C.: Optimal Transport: Old and New. Springer-Verlag (2008)
54. Virmaux, A., Scaman, K.: Lipschitz regularity of deep neural networks: analysis and efficient estimation. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018), https://proceedings.neurips.cc/paper/2018/file/d54e99a6c03704e95e6965532dec148b-Paper.pdf
55. Weinan, E.: A proposal on machine learning via dynamical systems. Commun. Math. Stat (2017)
56. Weng, T.W., Zhang, H., Chen, P.Y., Yi, J., Su, D., Gao, Y., Hsieh, C.J., Daniel, L.: Evaluating the robustness of neural networks: An extreme value theory approach. In: International Conference on Learning Representations (2018)
57. Wightman, R., Touvron, H., Jégou, H.: Resnet strikes back: An improved training procedure in timm. arXiv (2021)
58. Wu, D., Wang, Y., Xia, S.T., Bailey, J., Ma, X.: Skip connections matter: On the transferability of adversarial examples generated with resnets. In: International Conference on Learning Representations (2020)
59. Xie, S., et al.: Aggregated residual transformations for deep neural networks. In: CVPR (2017)
60. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: BMVC (2016)
61. Zhang, C., Zhang, H., Hsieh, C.J.: An efficient adversarial attack for tree ensembles. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., Lin, H. (eds.) Advances in Neural Information Processing Systems. vol. 33, pp. 16165–16176. Curran Associates, Inc. (2020), https://proceedings.neurips.cc/paper/2020/file/ba3e9b6a519cfddc560b5d53210df1bd-Paper.pdf

62. Zhang, J., et al.: Towards robust resnet: A small step but a giant leap. In: Proceedings of the Twenty-Eight International Joint Conference on Artificial Intelligence (IJCAI-19) (2019)
63. Zhang, J., et al.: Towards robust resnet: A small step but a giant leap. In: Proceedings of the Twenty-Eight International Joint Conference on Artificial Intelligence (IJCAI-19) (2019)

A Background on Optimal Transport

The Wasserstein space $\mathbb{W}_2(\Omega)$ with Ω a convex and compact subset of $\mathbb{R}^d$ is the space $\mathcal{P}(\Omega)$ of probability measures over Ω, equipped with the distance W_2 given by the solution to the optimal transport problem

$$W_2^2(\alpha,\beta) = \min_{\gamma\in\Pi(\alpha,\beta)} \int_{\Omega\times\Omega} \|x-y\|^2 \,\mathrm{d}\gamma(x,y) \tag{11}$$

where $\Pi(\alpha,\beta)$ is the set of probability distribution over $\Omega\times\Omega$ with first marginal α and second marginal β, i.e. $\Pi(\alpha,\beta) = \{\gamma\in\mathcal{P}(\Omega\times\Omega) \mid \pi_{1\sharp}\gamma = \alpha,\ \pi_{2\sharp}\gamma = \beta\}$ where $\pi_1(x,y) = x$ and $\pi_2(x,y) = y$. The optimal transport problem can be seen as looking for a transportation plan minimizing the cost of displacing some distribution of mass from one configuration to another. This problem indeed has a solution in our setting (see for example [48, 53]). If α is absolutely continuous and $\partial\Omega$ is α-negligible then the problem in (11) (called the Kantorovich problem) has a unique solution and is equivalent to the following problem, called the Monge problem,

$$W_2^2(\alpha,\beta) = \min_{T \text{ s.t. } T_\sharp\alpha=\beta} \int_{\Omega} \|T(x)-x\|^2 \,\mathrm{d}\alpha(x) \tag{12}$$

and this problem has a unique solution $T^\star$ linked to the solution $\gamma^\star$ of (11) through $\gamma^\star = (\mathrm{id}, T^\star)_\sharp\alpha$. Another equivalent formulation of the optimal transport problem in this setting is the dynamical formulation ([5]). Here, instead of directly pushing samples of α to β using T, we can equivalently displace mass, according to a continuous flow with velocity $v_t : \mathbb{R}^d \to \mathbb{R}^d$. This implies that the density α_t at time t satisfies the *continuity equation* $\partial_t\alpha_t + \nabla\cdot(\alpha_t v_t) = 0$, assuming that initial and final conditions are given by $\alpha_0 = \alpha$ and $\alpha_1 = \beta$ respectively. In this case, the optimal displacement is the one that minimizes the total action caused by v :

$$\begin{aligned} W_2^2(\alpha,\beta) = \min_{v} \int_0^1 \|v_t\|^2_{L^2(\alpha_t)} \,\mathrm{d}t \\ \text{s.t. } \partial_t\alpha_t + \nabla\cdot(\alpha_t v_t) = 0,\ \alpha_0 = \alpha, \alpha_1 = \beta \end{aligned} \tag{13}$$

Instead of describing the density's evolution through the continuity equation, we can describe the paths ϕ_t^x taken by particles at position x from α when displaced along the flow v. Here ϕ_t^x is the position at time t of the particle that was at $x\sim\alpha$ at time 0. The continuity equation is then equivalent to $\partial_t\phi_t^x = v_t(\phi_t^x)$. See chapters 4 and 5 of [48] for details. Rewriting the conditions as necessary, Problem (13) becomes

$$\begin{aligned} W_2^2(\alpha,\beta) = \min_{v} \int_0^1 \|v_t\|^2_{L^2((\phi_t^\cdot)_\sharp\alpha)} \,\mathrm{d}t \\ \text{s.t. } \partial_t\phi_t^x = v_t(\phi_t^x),\ \phi_0^\cdot = \mathrm{id}, (\phi_1^\cdot)_\sharp\alpha = \beta \end{aligned} \tag{14}$$

and the optimal transport map $T^\star$ that solves (12) is in fact $T^\star(x) = \phi_1^x$ for ϕ that solves the continuity equation together with the optimal $v^\star$ from (14). The

optimal vector field is related to the optimal map through $v_t^\star = (T^\star - \mathrm{id}) \circ (T_t^\star)^{-1}$, where $T_t^\star = (1-t)\mathrm{id} + tT^\star$ and is invertible. This simply means that the points move in straight lines and with constant speed from x to $T^\star(x)$. The $\mathbb{W}_2(\Omega)$ space is a metric geodesic space, and the geodesic between α and β is the curve α_t found while solving (13). It is also given by $\alpha_t = (\pi_t)_\sharp \gamma^\star = (T_t^\star)_\sharp \alpha$, where $\pi_t(x,y) = (1-t)x + ty$. We refer to Section 5.4 of [48] for these results on optimal transport.

Optimal transport maps have some regularity properties under some boundedness assumptions. We mention the following result from [17]:

Theorem 4. *Suppose there are X, Y, bounded open sets, such that the densities of α and β are null in their respective complements and bounded away from zero and infinity over them respectively.*
Then, if Y is convex, there exists $\eta > 0$ such that the optimal transport map T between α and β is $C^{0,\eta}$ over X.
If Y isn't convex, there exists two relatively closed sets A, B in X, Y respectively such that $T \in C^{0,\eta}(X \setminus A, Y \setminus B)$, where A and B are of null Lebesgue measure. Moreover, if the densities are in $C^{k,\eta}$, then $C^{0,\eta}$ can be replaced by $C^{k+1,\eta}$ in the conclusions above. In particular, if the densities are smooth, then the transport map is a diffeomorphism.

A final result that we mention is the following, which says that the inverse of the optimal transport map between α and β is the optimal transport map from β to α,

Theorem 5. *If α and β are absolutely continuous measures supported respectively on compact subsets X and Y of $\mathbb{R}^d$ with negligible boundaries, then there exists a unique couple (T, S) of functions such that the five following points hold*

- *$T : X \to Y$ and $S : Y \to X$*
- *$T_\# \alpha = \beta$ and $S_\# \beta = \alpha$*
- *T is optimal for the Monge problem from α to β*
- *S is optimal for the Monge problem from β to α*
- *$T \circ S \overset{\beta-a.s.}{=} \mathtt{id}$ and $S \circ T \overset{\alpha-a.s.}{=} \mathtt{id}$*

B Background on Numerical Methods for ODEs

We refer to [45] for this quick background on numerical methods for ODEs. Consider the Cauchy problem $x' = f(t,x)$ with initial condition $x(t_0) = x_0$ and a subdivision $t_0 < t_1 < .. < t_N = t_0 + T$ of $[t_0, t_0 + T]$. Denote the time-steps $h_n := t_{n+1} - t_n$ for $0 \leq n < N$ and define $h_{\max} := \max h_n$. In a one-step method, an approximation of $x(t_n)$ is x_n given by

$$\frac{x_{n+1} - x_n}{h_n} = \phi(t_n, x_n, h_n)$$

For $\phi(t,x,h) = f(t,x)$, we get Euler's method: $x_{n+1} = x_n + h_n f(t_n, x_n)$.

Definition 2. (Consistency and order) *For a one-step method, the consistency errors e_n, for $0 \leq n < N$, are*

$$e_n = \frac{x(t_{n+1}) - \Phi(t_n, x(t_n), h_n)}{h_n} = \frac{x(t_{n+1}) - x(t_n)}{h_n} - \phi(t_n, x(t_n), h_n)$$

where x is solution. The local (truncation) errors are $h_n e_n$. The method is consistent if $\max|e_n|$ goes to zero as h_{max} goes to zero. For $p \in \mathbb{N}^*$, the method has order p if $\max|e_n| \leq Ch_{max}^p$ for a constant C that depends on f, t_0 and T.

Theorem 6. (Consistency criterion) *If f and ϕ are continuous then the one-step method is consistent if and only if $\phi(t, x, 0) = f(t, x)$ for all (t, x).*

Theorem 7. (Order criterion) *If f is $\mathcal{C}^p$ and ϕ is $\mathcal{C}^p$ in h then the one-step method is of order p if and only if $\partial_h^k \phi(t, x, 0) = \frac{1}{k+1} f^{[k]}(t, x)$ for all (t, x) and $0 \leq k < p$ where $f^{[0]} = f$ and $f^{[k]} = \partial_t f^{[k-1]} + f\partial_x f^{[k-1]}$.*

Corollary 1. (Consistency and order of Euler's method) *If f is continuous then Euler's method is consistent. If f is $\mathcal{C}^1$ then Euler's method has order 1.*

Definition 3. (Zero-stability) *A one-step method is zero-stable (or stable) if $\exists\ S > 0$ such that for all $(x_n)_{0 \leq n \leq N}$, $(\tilde{x}_n)_{0 \leq n \leq N}$ and $(\varepsilon_n)_{0 \leq n < N}$ satisfying*

$$\frac{x_{n+1} - x_n}{h_n} = \phi(t_n, x_n, h_n)$$

and

$$\frac{\tilde{x}_{n+1} - \tilde{x}_n}{h_n} = \phi(t_n, \tilde{x}_n, h_n) + \epsilon_n$$

for $0 \leq n < N$, we have

$$\max_n \|\tilde{x}_n - x_n\| \leq S(\|\tilde{x}_0 - x_0\| + T \max_n |\epsilon_n|)$$

, where $\epsilon_n = \varepsilon_n / h_n$. The constant S is the stability constant of the method.

Theorem 8. (Zero-stability criterion) *If ϕ is uniformly L-Lipschitz in its second variable, then the one-step method is stable with constant e^{LT}.*

Corollary 2. (Zero-stability of Euler's method) *If f is Lipschitz in its second variable, then Euler's method is stable.*

Definition 4. (Convergence) *A numerical method converges if its global error $\max_n \|x(t_n) - x_n\|$ goes to zero as h_{max} goes to zero.*

Theorem 9. (Convergence criterion) *If a method is consistent and stable with stability constant S, then it converges and $\max_n \|x(t_n) - x_n\| \leq ST \max|e_n|$. If the method is of order p with constant C, then $\max_n \|x(t_n) - x_n\| \leq STCh_{max}^p$*

Corollary 3. (Convergence of Euler's method) *Euler's method converges if f is $\mathcal{C}^0$ and Lipschitz in x. If f is also $\mathcal{C}^1$ then it converges with speed $O(h_{max})$.*

C Proofs

C.1 Proof of Theorem 1

Proof. A solution v to (5) exists and is linked to an optimal transport map T that is a solution to (4) through $v_t = (T - \mathtt{id}) \circ T_t^{-1}$ where $T_t := (1-t)\mathtt{id} + tT$ which is invertible (see Appendix A).

By Theorem 4 in Appendix A, being an optimal transport map, T is η-Hölder on X. So for all $a, b \in \text{support}(\alpha_t)$ and $t \in [0, 1[$, where $\alpha_t = (\phi_t^{\cdot})_\sharp \alpha = (T_t)_\sharp \alpha$ with ϕ solving (5) with v, we have

$$\|v_t(a) - v_t(b)\| \leq \|T_t^{-1}(a) - T_t^{-1}(b)\| + C\|T_t^{-1}(a) - T_t^{-1}(b)\|^\eta$$

Since $(\alpha_t)_{t=0}^1$ is a geodesic between α and $\beta = \alpha_1 = T_\sharp \alpha$, then $(\alpha_s)_{s=0}^t$ is a geodesic between α and α_t (modulo reparameterization to $[0, 1]$). And since $\alpha_s = (T_s)_\sharp \alpha$, the map T_t is an optimal transport map between α and α_t. Therefore its inverse T_t^{-1} is an optimal transport map (see Theorem 5 in Appendix A) and is η_t-Hölder with $0 < \eta_t \leq 1$ (being a push-forward by T_t, the support of α_t satisfies the conditions of Theorem 4 in Appendix A). Therefore, for all $a, b \in \text{support}(\alpha_t)$

$$\|v_t(a) - v_t(b)\| \leq C_t\|a - b\|^{\eta_t} + CC_t^\eta \|a - b\|^{\eta\eta_t} \tag{15}$$

and for all $a, b \in \text{support}(\alpha_{t_m})$

$$\|r_m(a) - r_m(b)\| \leq \varepsilon + C_{t_m}\|a - b\|^{\eta_{t_m}} + CC_{t_m}^\eta \|a - b\|^{\eta\eta_{t_m}} \tag{16}$$

by the hypothesis on r and the triangle inequality. Let $K := \max_m C_{t_m} + CC_{t_m}^\eta$, $\zeta_1 := \eta \min_m \eta_{t_m}$ and $\zeta_2 := \max_m \eta_{t_m}$. Then, we have the desired result immediately from (16).

Remark 1. If the convexity hypothesis on the support Y of the target distribution β is too strong, we still get the same results almost everywhere. More precisely, if the set Y such that β is bounded away from zero and infinity on Y and is zero on $Y^\complement$ is open and bounded but not convex, then the solution map T is η-Hölder almost everywhere on X (see Appendix A).

Remark 2. If the distributions α and β in Theorem 1 are $\mathcal{C}^{k,\eta}$ (i.e all derivatives up to the k-th derivative are η-Hölder), then the optimal transport map T is $\mathcal{C}^{k+1,\eta}$. This means that the more regular the data, the more regular the network we find.

C.2 Proof of Theorems 2 and 3

Proof. Since $T_t^{-1}(\phi_t^x) = x$, we have for any $a_0, b_0 \in X$ by the triangle inequality

$$\begin{aligned}\|r_m(a_m) - r_m(b_m)\| \leq{}& \|r_m(a_m) - r_m(\phi_{t_m}^{a_0})\| + \|r_m(\phi_{t_m}^{a_0}) - v_{t_m}(\phi_{t_m}^{a_0})\| + \\ &+ \|v_{t_m}(\phi_{t_m}^{a_0}) - v_{t_m}(\phi_{t_m}^{b_0})\| + \|r_m(\phi_{t_m}^{b_0}) - v_{t_m}(\phi_{t_m}^{b_0})\| + \\ &+ \|r_m(b_m) - r_m(\phi_{t_m}^{b_0})\|\end{aligned}$$

So

$$\|r_m(a_m) - r_m(b_m)\| \leq \varepsilon + \|a_0 - b_0\| + C\|a_0 - b_0\|^\eta + \\ + L(\|a_m - \phi_{t_m}^{a_0}\| + \|b_m - \phi_{t_m}^{b_0}\|)$$

where $L = \max_m L_m$ and L_m is the Lipschitz constant of r_m (which is Lipschitz being a composition of matrix multiplications and activations such as ReLU). This the bound in Theorem 2.

In this bound, the term $\|a_m - \phi_{t_m}^{a_0}\|$ (and likewise $\|b_m - \phi_{t_m}^{b_0}\|$) represents the distance between the point a_m we get after m residual blocks (i.e. after m Euler steps using the approximation r of v) and the point $\phi_{t_m}^{a_0}$ we get by following the solution vector field v up to time t_m. By the properties of the Euler method (consistency and zero-stability, see Corollaries 1, 2 and 3 in Appendix B), under more regularity conditions on v, it is possible to bound this term. Indeed, if v is $\mathcal{C}^1$ and M-Lipschitz in x (this is not stronger than the regularity we get on v through our regularization, because we still need to use (15)), we have for constants $R, S > 0$,

$$\|\phi_{t_m}^{a_0} - a_m\| \leq \|\phi_{t_m}^{a_0} - \tilde{a}_m\| + \|\tilde{a}_m - a_m\| \leq S\varepsilon + SRh$$

where $\tilde{a}_m$ comes from the Euler scheme with access to v (i.e. $\tilde{a}_{m+1}{:=}\tilde{a}_m + hv_{t_m}(\tilde{a}_m)$ and $\tilde{a}_0{:=}a_0$), R is the consistency constant of the Euler method and S is its zero-stability constant. Likewise, we get the same bound for $\|b_m - \phi_{t_m}^{b_0}\|$.

If $a_0, b_0 \notin X$, we need to introduce $\hat{a}_0 := \mathrm{Proj}_X(a_o)$ and $\hat{b}_0 := \mathrm{Proj}_X(b_o)$ to apply (15). We now get

$$\|r_m(a_m) - r_m(b_m)\| \leq \varepsilon + \|a_0 - b_0\| + C\|a_0 - b_0\|^\eta + \\ + L(\|a_m - \phi_{t_m}^{\hat{a}_0}\| + \|b_m - \phi_{t_m}^{\hat{b}_0}\|)$$

Bounding the terms $\|a_m - \phi_{t_m}^{\hat{a}_0}\|$ and $\|b_m - \phi_{t_m}^{\hat{b}_0}\|$ now gives

$$\|\phi_{t_m}^{\hat{a}_0} - a_m\| \leq \|a_m - \tilde{a}_m\| + \|\tilde{a}_m - \phi_{t_m}^{\hat{a}_0}\| \leq S(\|a_0 - \hat{a}_0\| + \varepsilon) + SRh$$

where $\tilde{a}_m$ now comes from the Euler scheme with access to v that starts at $\hat{a}_0$ (meaning $\tilde{a}_{m+1}{:=}\tilde{a}_m + hv_{t_m}(\tilde{a}_m)$ and $\tilde{a}_0{:=}\hat{a}_0$). Likewise, we get the same bound for $\|b_m - \phi_{t_m}^{\hat{b}_0}\|$.

Since $\|a_0 - \hat{a}_0\| = \mathrm{dist}(a_0, X)$ and $\|b_0 - \hat{b}_0\| = \mathrm{dist}(b_0, X)$, we get the bound in Theorem 3. Note that if we use the stability of the ODE instead of the Euler method to bound $\|a_m - \phi_{t_m}^{\hat{a}_0}\|$ we get the same result. Indeed, if $\tilde{a}_m$ again comes from the Euler scheme with access to v that starts at a_0 (meaning $\tilde{a}_{m+1}{:=}\tilde{a}_m + hv_{t_m}(\tilde{a}_m)$ and $\tilde{a}_0{:=}a_0$), we can write, for some constant $F > 0$

$$\|\phi_{t_m}^{\hat{a}_0} - a_m\| \leq \|a_m - \tilde{a}_m\| + \|\tilde{a}_m - \phi_{t_m}^{a_0}\| + \|\phi_{t_m}^{a_0} - \phi_{t_m}^{\hat{a}_0}\| \\ \leq S\varepsilon + SRh + F\|a_0 - \hat{a}_0\|$$

since

$$\begin{aligned}\|\phi_{t_m}^{a_0} - \phi_{t_m}^{\hat{a}_0}\| &\leq \|a_0 - \hat{a}_0\| + \int_0^{t_m} \|v_s(\phi_s^{a_0}) - v_s(\phi_s^{\hat{a}_0})\| \, \mathrm{d}s \\ &\leq \|a_0 - \hat{a}_0\| + M \int_0^{t_m} \|\phi_s^{a_0} - \phi_s^{\hat{a}_0}\| \, \mathrm{d}s \leq F\|a_0 - \hat{a}_0\|\end{aligned}$$

where we get the last line by Gronwall's lemma.

D Additional Experiments

D.1 Adversarial Attacks

White-box attacks have access to the network's weights and architecture. The Fast Gradient Method (FGM) [19] takes a perturbation step in the direction of the gradient that maximizes the loss. Projected Gradient Descent (PGD) [37] and the Basic Iterative Method (BIM) [31] are iterative versions of FGM. We use the Auto-PGD-CE [13] variant of PGD which has an adaptive step size. Two slower but more powerful attacks are DeepFool (DF) [40], which iteratively perturbs an input in the direction of the closest decision boundary, and Carlini-Wagner (CW) [8], which solves an optimization problem to find the perturbation. AutoAttack (AA) [13] is a combination of three white-box attacks (two variants of Auto-PGD [13] and the FAB attack of [12]), and of the black-box Square Attack (SA) [4]. Black-box attacks don't have any knowledge about the network and can only query it. We use two such attacks: Hop-Skip-Jump (HSJ) [10], which estimates the gradient direction at the decision boundary, and the Boundary Attack (BA) [7], which starts from a large adversarial input and moves towards the boundary decision to minimize the perturbation. We use a maximal perturbation of ϵ=0.03 for FGM, APGD, BIM and AA. We use the L_2 norm for CW and HSJ and L_∞ for the other attacks. We use ART [43] and its default hyper-parameter values (except those mentioned) to generate the adversarial samples, except for AA for which we use the authors' original code. The number of iterations is 50 for HSJ, 5000 for BA, 10 for CW and 100 for APGD and DF.

D.2 Implementation Details

For ResNeXt50 [59] on CIFAR100 [30], we train for 300 epochs using SGD with a learning rate of 0.1 (divided by ten at epochs 150, 225 and 250), Kaiming initialization, a batch size of 128 and weight decay of 0.0001. For RCE training, the only changes are that the learning rate is 0.05 and the initialization is orthogonal with a gain of 0.05.

For ResNet110 [21] on CIFAR10 [30], we train for 300 epochs using SGD with a learning rate of 0.1 (divided by ten at epochs 150, 225 and 250), orthogonal initialization with a gain of 0.05, a batch size of 256, weight decay of 0.0001 and gradient clipping at 5. For RCE training, the only change is that we don't use gradient clipping.

For WideResNet [60] on TinyImageNet, we train for 300 epochs using SGD with a learning rate of 0.1 (divided by ten at epochs 150, 225 and 250), orthogonal initialization with a gain of 0.1, a batch size of 114 and weight decay of 0.0001.

For the magnitude parameter of the Mahalanobis detector, we try all the values tried in their paper for the magnitude and we report the best results.

D.3 Adversarial detection training data

See Figure 3.

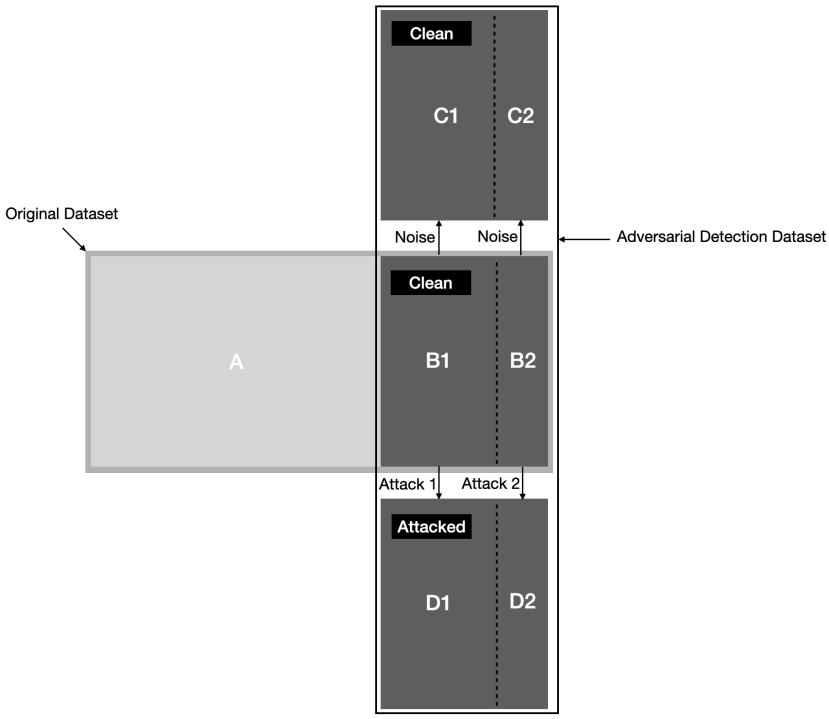

Fig. 3. Adversarial detection dataset creation. A∪B1∪B2 is the original dataset, where A is the training set and B1∪B2 is the test set. We create a noisy version of B1∪B2 by adding random noise to each sample in B1∪B2 to get C1∪C2. Noisy samples are considered clean (i.e. not attacked) in adversarial detection training. We create an attacked version of B1∪B2 by creating an attacked image from each image in B1∪B2 to get D1∪D2. In the case of generalization to unseen attacks, Attack 2 used to create D2 from B2 is different from Attack 1 used to create D1 from B1. Otherwise, Attack 1 and Attack 2 are the same. B1∪C1∪D1 is the adversarial detection training set and B2∪C2∪D2 is the adversarial detection test set.

D.4 Preliminary Experiments

We see in Figure 4 below that training deep ResNets without batch-normalization is near impossible, whereas LAP-ResNets maintain the same performance and stability without ResNets for up to 50 blocks. LAP-ResNets are also compared in this regard to the small step method of [63], which simply adds a small weight h of around 0.1 in front of the residue function to make ResNets more stable. The Least Action Principle has the same improved stability when training without batch-normalization in Figure 4 as this method, while also improving the test accuracy when batch-normalization is used [26] which the small step method does not claim.

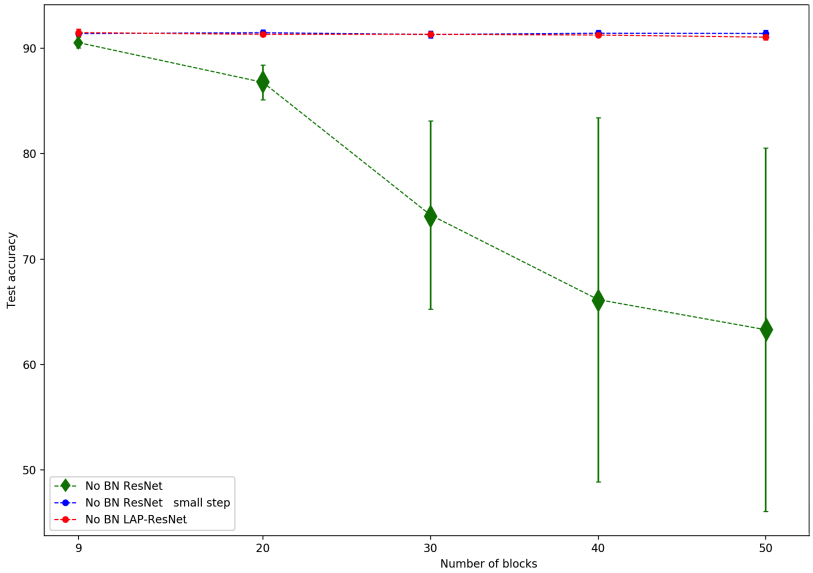

Fig. 4. Test accuracy of ResNets of various depths without batch-normalization on CIFAR10.

D.5 Detection of Seen Attacks

In the tables below, VAN corresponds to detectors trained on a vanilla network, RCE on an RCE-network and LAP on a LAP-network.

Table 3: Average adversarial detection accuracy of seen attacks and standard deviation over 5 runs using ResNet110 on CIFAR10.

	Attack			
Det	FGM	APGD	BIM	DF
VAN T	97.1 ± 0.6	94.1 ± 0.4	97.5 ± 0.5	**100** ± 0.5
RCE T	96.0 ± 0.5	95.3 ± 0.8	96.2 ± 0.6	99.9 ± 0.1
LAP T	**98.7** ± 0.3	**97.5** ± 0.4	**99.3** ± 0.3	99.8 ± 0.1
VAN M	87.8 ± 4.2	82.1 ± 4.0	86.8 ± 4.7	91.5 ± 3.2
RCE M	93.0 ± 0.6	87.9 ± 0.5	92.3 ± 1.0	95.0 ± 0.4
LAP M	95.6 ± 0.6	90.7 ± 0.7	95.4 ± 0.5	96.7 ± 0.7
VAN N	94.6 ± 0.7	94.3 ± 0.5	95.0 ± 0.5	99.8 ± 0.1

Table 4: Average adversarial detection accuracy of seen attacks and standard deviation over 5 runs using ResNet110 on CIFAR10.

	Attack			
Det	CW	AA	HSJ	BA
VAN T	**98.0** ± 0.5	88.9 ± 1.3	**99.9** ± 0.1	96.6 ± 0.6
RCE T	89.4 ± 0.5			
LAP T	98.0 ± 0.4	**94.1** ± 0.8	99.9 ± 0.1	**97.0** ± 0.2
VAN M	85.6 ± 2.6	80.5 ± 2.2	85.5 ± 1.8	80.2 ± 2.1
RCE M	83.4 ± 0.5			
LAP M	93.4 ± 0.6	90.0 ± 0.9	94.6 ± 0.4	89.6 ± 0.4
VAN N	93.9 ± 9.2	88.8 ± 1.4	99.7 ± 0.1	92.1 ± 0.5

Table 5: Average adversarial detection accuracy of seen attacks and standard deviation over 5 runs using ResNeXt50 on CIFAR100.

	Attack			
Det	FGM	PGD	BIM	AA
VAN T	97.3 ± 0.5	96.0 ± 0.5	98.0 ± 0.3	84.9 ± 0.7
RCE T	97.4 ± 0.4	97.0 ± 0.1	97.8 ± 0.2	50.1 ± 0.1
LAP T	**98.3** ± 0.3	**97.8** ± 0.5	**98.9** ± 0.1	**87.6** ± 0.6
VAN M	95.8 ± 0.5	93.9 ± 0.5	96.1 ± 0.6	83.9 ± 0.7
RCE M	96.5 ± 0.4	94.7 ± 0.4	96.6 ± 0.6	50.1 ± 0.1
LAP M	96.8 ± 0.4	94.6 ± 0.7	97.8 ± 0.5	86.6 ± 0.5
VAN N	94.7 ± 0.7	94.2 ± 1.0	94.7 ± 0.6	84.8 ± 0.8

Table 6: Average adversarial detection accuracy of seen attacks and standard deviation over 5 runs using ResNeXt50 on CIFAR100.

	Attack	
Detector	DF	CW
VAN TR	**99.80** ± 0.15	**97.04** ± 0.88
RCE TR	99.04 ± 0.14	92.52 ± 0.34
LAP TR	99.58 ± 0.18	97.80 ± 0.18
VAN MH	97.30 ± 0.45	95.38 ± 0.56
RCE MH	97.64 ± 0.44	88.36 ± 0.62
LAP MH	97.12 ± 0.28	96.42 ± 0.42
VAN NS	99.56 ± 0.21	90.72 ± 1.39

Table 7: Average adversarial detection accuracy of seen attacks and standard deviation over 5 runs using WideResNet on TinyImageNet.

	Attack			
Det	FGM	APGD	BIM	AA
VAN T	**95.4** ± 0.4	**95.2** ± 0.5	**95.3** ± 0.5	**81.4** ± 0.4
LAP T	95.1 ± 0.5	95.2 ± 0.7	95.1 ± 0.7	81.2 ± 0.5
VAN M	81.1 ± 1.1	79.7 ± 1.0	81.2 ± 1.3	78.4 ± 0.7
LAP M	85.3 ± 1.0	85.1 ± 0.6	82.5 ± 1.6	78.4 ± 1.0
VAN N	94.9 ± 0.7	94.9 ± 0.9	95.0 ± 0.6	81.3 ± 0.2

D.6 Detection of Unseen Attacks

Table 8: Average adversarial detection accuracy of unseen attacks after training on FGM and standard deviation over 5 runs using ResNet110 on CIFAR10.

	Attack			
Detector	APGD	BIM	AA	DF
VAN TR	89.3 ± 1.6	96.0 ± 0.7	**85.1** ± 1.1	**91.0** ± 0.9
RCE TR	91.8 ± 1.1	93.6 ± 1.1	50.0 ± 0.1	63.4 ± 1.1
LAP TR	**92.8** ± 0.5	**98.8** ± 0.4	84.2 ± 0.5	75.5 ± 1.2
VAN MH	77.3 ± 4.7	77.2 ± 4.8	72.1 ± 3.1	80.1 ± 3.4
RCE MH	81.5 ± 0.6	82.6 ± 1.3	50.0 ± 0.1	81.2 ± 0.7
LAP MH	87.9 ± 0.8	84.9 ± 0.4	81.9 ± 1.2	81.6 ± 0.7
VAN NS	92.1 ± 0.5	93.9 ± 0.4	51.8 ± 0.6	51.4 ± 0.58

Table 9: Average adversarial detection accuracy of unseen attacks after training on FGM and standard deviation over 5 runs using ResNet110 on CIFAR10.

	Attack		
Detector	CW	HSJ	BA
VAN TR	**93.2** ± 1.0	**93.0** ± 0.9	**90.9** ± 0.6
RCE TR	60.5 ± 0.9	63.9 ± 1.0	52.5 ± 0.5
LAP TR	75.2 ± 1.0	76.8 ± 0.6	75.0 ± 0.4
VAN MH	79.9 ± 3.7	79.7 ± 3.0	79.3 ± 3.0
RCE MH	76.0 ± 0.9	81.6 ± 0.9	68.5 ± 1.2
LAP MH	81.5 ± 0.6	81.5 ± 0.3	81.4 ± 0.8
VAN NS	50.84 ± 1.1	52.1 ± 0.7	59.9 ± 5.4

Table 10: Average adversarial detection accuracy of unseen attacks after training on FGM and standard deviation over 5 runs using ResNeXt50 on CIFAR100.

	Attack			
Detector	APGD	BIM	AA	DF
VAN T	91.9 ± 0.8	95.0 ± 0.5	73.3 ± 1.0	**85.2** ± 0.6
RCE T	87.7 ± 0.5	95.1 ± 0.9	50.0 ± 0.1	72.3 ± 0.4
LAP T	89.3 ± 0.8	**97.7** ± 0.3	74.0 ± 1.3	76.0 ± 1.0
VAN M	90.9 ± 0.8	93.2 ± 0.3	73.1 ± 0.6	82.7 ± 0.9
RCE M	82.0 ± 0.7	88.6 ± 0.8	50.0 ± 0.1	74.1 ± 0.8
LAP M	86.7 ± 0.9	93.9 ± 0.4	**80.0** ± 0.6	79.4 ± 1.6
VAN NS	**92.2** ± 0.4	93.9 ± 1.0	51.3 ± 0.4	51.6 ± 0.5

Table 11: Average adversarial detection accuracy of unseen attacks after training on FGM and standard deviation over 5 runs using ResNeXt50 on CIFAR100.

	Attack		
Detector	CW	HSJ	BA
VAN T	**78.2** ± 1.0	**85.0** ± 0.4	**92.1** ± 4.8
RCE T	61.9 ± 0.5	72.4 ± 0.4	57.7 ± 0.4
LAP T	74.7 ± 1.1	78.1 ± 3.4	71.9 ± 3.9
VAN M	76.4 ± 0.7	82.8 ± 1.2	84.5 ± 2.0
RCE M	63.0 ± 0.6	74.6 ± 0.3	63.2 ± 0.9
LAP M	80.9 ± 2.0	80.5 ± 3.7	78.2 ± 2.0
VAN NS	51.0 ± 0.4	52.0 ± 0.7	57.9 ± 7.5

Table 12: Average adversarial detection accuracy of unseen attacks after training on FGM and standard deviation over 5 runs using WideResNet on TinyImageNet.

	Attack		
Detector	APGD	BIM	AA
VAN TR	93.26 ± 0.60	94.66 ± 0.49	**77.04** ± 0.74
LAP TR	93.48 ± 0.72	**94.80** ± 0.56	76.58 ± 0.48
VAN MH	76.96 ± 0.94	77.02 ± 1.08	60.36 ± 0.62
LAP MH	77.96 ± 0.49	78.00 ± 0.77	61.96 ± 0.89
VAN NS	**94.06** ± 0.61	94.62 ± 0.64	72.82 ± 1.98

Table 13: Average adversarial detection accuracy of unseen attacks after training on FGM and standard deviation over 5 runs using WideResNet on TinyImageNet.

	Attack	
Detector	DF	CW
VAN TR	**90.62** ± 0.60	91.42 ± 1.06
LAP TR	90.12 ± 0.55	**91.52** ± 0.89
VAN MH	73.18 ± 0.59	75.52 ± 0.82
LAP MH	73.98 ± 1.12	76.22 ± 0.83
VAN NS	71.96 ± 4.03	65.60 ± 2.20

D.7 Detection Rate of Successful Adversarial Samples

As in [33], we might be only concerned with detecting adversarial samples that successfully fool the network and that are created from clean samples that are correctly classified. We find that the detection rate of successful adversarial samples is always very high and close to 100% on our detector. On seen attacks, the results are in Table 14. On unseen attacks, the results are in Table 15.

Table 14: Average detection rate of successful adversarial samples from seen attacks over 5 runs on Network/LAP-Network.

Attack	Detector	ResNet110 CIFAR10	ResNeXt50 CIFAR100	WideResNet TinyImageNet
FGM	TR	97.7/**98.6**	98.2/**98.6**	95.4/**95.9**
	MH	88.3/93.9	96.7/97.1	84.0/85.0
	NS	95.9	95.0	94.4
APGD	TR	99.3/**99.4**	97.1/**97.9**	**96.7**/96.4
	MH	85.9/86.8	95.8/92.7	82.7/84.8
	NS	95.9	94.8	94.5
BIM	TR	98.3/**99.6**	98.6/**99.2**	95.0/**96.1**
	MH	88.1/93.8	96.6/98.0	85.8/86.2
	NS	96.4	94.6	94.3
AA	TR	**100**/**100**	**100**/**100**	**100**/**100**
	MH	88.3/95.4	98.8/98.7	95.6/96.5
	NS	99.9	99.9	99.9
DF	TR	**100**/99.9	**99.9**/99.4	
	MH	93.8/98.2	97.6/97.8	
	NS	99.9	99.3	
CW	TR	98.6/98.7	99.9/99.6	
	MH	83.5/93.9	98.1/98.1	
	NS	**99.9**	**100**	
HSJ	TR	**100**/99.9		
	MH	82.3/95.1		
	NS	99.7		

Table 15: Average detection rate of successful adversarial samples from unseen attacks after training on FGM over 5 runs.

Attack	Detector	ResNet110 CIFAR10	ResNeXt50 CIFAR100	WideResNet TinyImageNet
APGD	TR	**96.94**	**98.76**	**100.0**
	MH	79.80	90.86	79.16
	NS	93.02	91.8	93.20
BIM	TR	**98.68**	**98.56**	**100.0**
	MH	78.66	93.38	86.42
	NS	94.24	93.8	93.98
AA	TR	**98.54**	**74.06**	**91.06**
	MH	81.10	73.74	71.70
	NS	10.22	8.32	54.5
DF	TR	**93.42**	**75.14**	**95.00**
	MH	79.96	73.64	68.34
	NS	8.12	8.06	50.80
CW	TR	**92.22**	**72.90**	**96.00**
	MH	78.96	72.34	76.66
	NS	8.76	7.38	51.96
HSJ	TR	**93.22**	**75.42**	
	MH	78.14	73.66	
	NS	9.60	8.94	
BA	TR	**93.38**	**91.04**	
	MH	79.42	76.44	
	NS	25.50	21.90	

D.8 False Positive Rate

We report here the false positive rate on seen (Table 16) and unseen (Table 17) attacks of both detectors.

Table 16: Average FPR of seen attacks over 5 runs on Network/LAP-Network.

Attack	Detector	ResNet110 CIFAR10	ResNeXt50 CIFAR100	WideResNet TinyImageNet
FGM	TR	3.3/**1.5**	3.4/**1.9**	**3.7**/5.2
	MH	13.9/3.5	5.2/3.3	18.1/16.3
	NS	5.4	5.0	4.5
APGD	TR	6.3/**2**	4.6/**2.3**	5.3/4.9
	MH	18.7/4.7	6.3/3.6	17.5/16.9
	NS	4.9	5.0	**4.5**
BIM	TR	2.7/**0.8**	2.5/**1.9**	**3.6**/4.6
	MH	13.6/3.1	4.6/3.3	18.0/15.9
	NS	4.8	4.8	4.2
AA	TR	1.9/**1.5**	4.0/**4.1**	**6.8**/7.0
	MH	13.3/6.4	13.7/10.8	14.8/15.3
	NS	2.9	5.1	7.4
DF	TR	**0.1**/0.2	**0.2**/0.3	
	MH	10.2/4.6	2.9/2.7	
	NS	0.4	**0.2**	
CW	TR	2.6/2.8	**0.3**/0.4	
	MH	12.4/7.4	2.5/2.2	
	NS	**2.3**	2.8	
HSJ	TR	**0.1**/**0.1**		
	MH	11.9/5.8		
	NS	0.3		

Table 17: Average FPR of unseen attacks after training on FGM over 5 runs.

Attack	Detector	ResNet110 CIFAR10	ResNeXt50 CIFAR100	WideResNet TinyImageNet
APGD	TR	**6.70**	8.96	**9.44**
	MH	21.36	**7.28**	14.48
BIM	TR	**6.02**	7.10	**9.44**
	MH	21.46	**7.08**	14.54
AA	TR	**8.06**	**7.82**	**9.48**
	MH	23.36	7.90	14.48
DF	TR	**6.34**	**3.74**	**9.44**
	MH	18.48	7.78	14.54
CW	TR	**5.64**	**3.74**	**9.44**
	MH	18.86	7.68	14.82
HSJ	TR	**6.70**	**4.86**	
	MH	18.72	7.56	
BA	TR	**6.70**	**5.74**	
	MH	19.10	7.90	

D.9 AUROC

We report in Table 18 the AUROC of seen attacks, and in Table 19 the AUROC of unseen attacks. Note that the AUROC is computed on the class-agnostic random forest detector, not on the ensemble of the class-agnostic and the class-conditional detectors.

Table 18: Average AUROC of seen attacks on Network/LAP-Network.

Attack	Detector	ResNet110 CIFAR10	ResNeXt50 CIFAR100	WideResNet TinyImageNet
AA	TR	94.91/**99.95**	**99.86**/99.70	82.58/**82.95**
	MH	81.94/94.17	87.13/94.32	70.85/71.36
DF	TR	**99.94**/99.92	**99.84**/99.58	
	MH	89.60/94.17	86.88/91.82	
CW	TR	98.77/**99.96**	**99.85**/99.61	
	MH	88.33/94.31	86.33/91.36	
HSJ	TR	**99.95**/99.94		
	MH	86.01/93.45		

Table 19: Average AUROC of unseen attacks after training on FGM over 5 runs.

Attack	Detector	ResNet110 CIFAR10	ResNeXt50 CIFAR100	WideResNet TinyImageNet
AA	TR	**82.70**	**71.53**	**9.48**
	MH	76.46	66.60	61.61
DF	TR	**90.09**	**72.61**	**9.44**
	MH	82.25	66.00	71.72
CW	TR	**88.84**	**71.89**	**9.44**
	MH	81.59	66.06	71.57
HSJ	TR	**87.30**	**74.85**	
	MH	75.89	67.07	

D.10 Time Comparison

With a ResNeXt50 on CIFAR100 and a Tesla V100 GPU, it takes our method (including the time to generate FGM attacks) 66 seconds to extract its features from both the clean and the adversarial samples, while it takes the Mahalanobis method 110 seconds. Mahalanobis also extracts some statistics from the training set prior to adversarial detection training, which takes an additional 35 seconds. Our feature vector is of size 32, compared to 5 for the Mahalanobis detector. So our random forest takes only 4 more seconds to train than the Mahalanobis one (7 vs 3 seconds). Computation of the features our detector uses (norms and cosines) is in $O(MD)$, where M is the number of residual blocks and D is the largest embedding dimension inside the network.

D.11 Detection of Out-Of-Distribution Samples

Since our analysis applies to all out-of-distribution (OOD) samples, we test detection of OOD samples in a similar setting to the Mahalanobis paper [33]. We use the same ResNet110 and ResNeXt50 models trained on CIFAR10 and CIFAR100 respectively. Since the detectors need to be trained, we are in the OOD setting where we have a first dataset for training the network (CIFAR10 in Tables 20 and 21 and CIFAR100 in Tables 26 and 27) and a second dataset from another distribution that is not the test OOD distribution to train the detector on. This could be another dataset (CIFAR100 in Table 20), some images found in the wild, or a perturbation of our dataset that we generate using an adversarial attack (CW on CIFAR10 in Table 21, and AA and CW on CIFAR100 in Tables 26 and 27 respectively). Detectors can then be used by training them to distinguish between these first two datasets, and then testing them on distinguishing between the first dataset and a third unseen dataset (SVHN [41] in both tables). The accuracy is in Tables 20 to 27. The AUROC is in Tables 22 to 29. The false positive rate (FPR) at a fixed true positive rate (TPR) of 95% is in Tables 24 to 31. Our detector performs very well and better than the MH detector in three of

the four experiments, and in the fourth case, the MH detector benefits from LAP training by 8 percentage points (Table 26). Without any extra data available, using the CW adversarial attack allows to detect OOD samples from an unseen distribution with more than 90% accuracy and an FPR of less than 10% at a fixed TPR of 95%. The choice of the attack is also important, as CW allows for much better detection of unseen samples from SVHN than AA.

Table 20: Average OOD detection accuracy and standard deviation over 5 runs using ResNet110 trained on CIFAR10.

Detector	CIFAR100 (seen)	SVHN (unseen)
VAN TR	98.30 ± 0.46	97.46 ± 0.49
RCE TR	**98.42** ± 0.40	98.20 ± 0.39
LAP TR	98.30 ± 0.22	**98.50** ± 0.47
VAN MH	86.88 ± 1.52	91.28 ± 0.92
RCE MH	94.82 ± 0.45	92.16 ± 0.57
LAP MH	94.84 ± 0.41	90.46 ± 1.45

Table 21: Average OOD detection accuracy and standard deviation over 5 runs using ResNet110 trained on CIFAR10.

Detector	CW-CIFAR10 (seen)	SVHN (unseen)
VAN TR	**97.42** ± 0.57	**91.38** ± 0.95
RCE TR	91.54 ± 6.06	77.58 ± 6.72
LAP TR	97.28 ± 0.62	85.46 ± 2.64
VAN MH	81.80 ± 1.96	83.76 ± 1.13
RCE MH	76.74 ± 2.75	54.24 ± 3.46
LAP MH	89.68 ± 0.65	76.72 ± 1.73

Table 22: Average OOD detection AUROC and standard deviation over 5 runs using ResNet110 trained on CIFAR10.

Detector	CIFAR100 (seen)	SVHN (unseen)
VAN TR	99.64 ± 0.13	98.74 ± 0.47
RCE TR	99.61 ± 0.09	99.01 ± 0.25
LAP TR	**99.73** ± 0.09	**99.43** ± 0.28
VAN MH	92.74 ± 1.50	97.00 ± 0.80
RCE MH	97.97 ± 0.25	96.26 ± 0.40
LAP MH	98.06 ± 0.38	96.31 ± 0.75

Table 23: Average OOD detection AUROC and standard deviation over 5 runs using ResNet110 trained on CIFAR10.

Detector	CW-CIFAR10 (seen)	SVHN (unseen)
VAN TR	**99.32** ± 0.14	**96.29** ± 0.71
RCE TR	96.53 ± 0.58	86.34 ± 3.98
LAP TR	99.31 ± 0.07	96.12 ± 0.59
VAN MH	88.38 ± 2.94	88.04 ± 4.03
RCE MH	88.28 ± 2.22	79.53 ± 3.78
LAP MH	95.22 ± 0.90	86.54 ± 3.60

Table 24: Average OOD detection FPR at 95% TPR and standard deviation over 5 runs using ResNet110 trained on CIFAR10.

Detector	CIFAR100 (seen)	SVHN (unseen)
VAN TR	1.16 ± 0.66	2.68 ± 1.05
RCE TR	1.16 ± 0.41	1.94 ± 0.86
LAP TR	**1.02** ± 0.44	**1.54** ± 0.56
VAN MH	36.42 ± 4.29	15.66 ± 1.69
RCE MH	7.34 ± 0.90	20.30 ± 4.45
LAP MH	6.98 ± 2.35	17.56 ± 5.76

Table 25: Average OOD detection FPR at 95% TPR and standard deviation over 5 runs using ResNet110 trained on CIFAR10.

Detector	CW-CIFAR10 (seen)	SVHN (unseen)
VAN TR	2.71 ± 1.01	6.68 ± 0.98
RCE TR	19.2 ± 2.35	24.98 ± 5.43
LAP TR	**2.70** ± 0.64	**5.68** ± 1.12
VAN MH	49.46 ± 4.80	36.78 ± 6.97
RCE MH	41.94 ± 6.44	52.72 ± 7.71
LAP MH	23.06 ± 6.29	58.40 ± 14.42

Table 26: Average OOD detection accuracy and standard deviation over 5 runs using ResNeXt50 trained on CIFAR100.

Detector	AA-CIFAR100 (seen)	SVHN (unseen)
VAN TR	84.48 ± 0.59	75.32 ± 0.62
RCE TR	50.04 ± 0.07	55.44 ± 5.76
LAP TR	**87.10** ± 0.12	72.98 ± 2.72
VAN MH	83.44 ± 0.48	78.82 ± 0.48
RCE MH	50.04 ± 0.07	58.74 ± 2.36
LAP MH	86.04 ± 0.31	**86.84** ± 0.68

Table 27: Average OOD detection accuracy and standard deviation over 5 runs using ResNeXt50 trained on CIFAR100.

Detector	CW-CIFAR100 (seen)	SVHN (unseen)
VAN TR	95.82 ± 0.67	**92.92** ± 1.36
RCE TR	76.48 ± 0.75	75.66 ± 0.68
LAP TR	**95.94** ± 0.57	85.94 ± 2.88
VAN MH	94.96 ± 0.81	85.10 ± 1.50
RCE MH	76.20 ± 0.72	72.20 ± 1.47
LAP MH	94.82 ± 0.34	88.92 ± 1.26

Table 28: Average OOD detection AUROC and standard deviation over 5 runs using ResNeXt50 trained on CIFAR100.

Detector	AA-CIFAR100 (seen)	SVHN (unseen)
VAN TR	94.96 ± 0.38	78.77 ± 1.15
RCE TR	50.12 ± 0.05	50.30 ± 8.13
LAP TR	**96.45** ± 0.08	76.28 ± 0.73
VAN MH	93.32 ± 0.41	85.02 ± 0.97
RCE MH	50.15 ± 0.08	58.04 ± 3.56
LAP MH	94.87 ± 0.24	**92.76** ± 0.39

Table 29: Average OOD detection AUROC and standard deviation over 5 runs using ResNeXt50 trained on CIFAR100.

Detector	CW-CIFAR100 (seen)	SVHN (unseen)
VAN TR	99.00 ± 0.13	95.17 ± 0.31
RCE TR	87.50 ± 1.65	79.99 ± 3.62
LAP TR	**99.16** ± 0.28	94.84 ± 1.63
VAN MH	98.24 ± 0.32	93.07 ± 0.33
RCE MH	86.73 ± 2.03	77.42 ± 3.38
LAP MH	97.84 ± 0.20	**95.92** ± 0.64

Table 30: Average OOD detection FPR at 95% TPR and standard deviation over 5 runs using ResNeXt50 trained on CIFAR100.

Detector	AA-CIFAR100 (seen)	SVHN (unseen)
VAN TR	27.68 ± 1.84	32.66 ± 1.41
RCE TR	96.12 ± 0.96	94.72 ± 3.10
LAP TR	**21.62** ± 0.30	**29.47** ± 0.59
VAN MH	29.75 ± 1.09	39.06 ± 0.96
RCE MH	95.28 ± 0.27	91.42 ± 1.26
LAP MH	24.74 ± 0.69	34.54 ± 1.16

Table 31: Average OOD detection FPR at 95% TPR and standard deviation over 5 runs using ResNeXt50 trained on CIFAR100.

Detector	CW-CIFAR100 (seen)	SVHN (unseen)
VAN TR	5.42 ± 0.82	**8.90** ± 1.20
RCE TR	44.12 ± 3.31	45.68 ± 2.01
LAP TR	**4.90** ± 0.92	10.52 ± 3.61
VAN MH	8.02 ± 0.81	16.38 ± 0.82
RCE MH	45.12 ± 3.76	48.45 ± 4.55
LAP MH	8.10 ± 0.46	9.80 ± 1.21

D.12 Attacking the Detector

We consider here the case where the attacker also attacks the detector (adaptive attacks). We try two such attacks on the TR and MH detectors on ResNet110 trained on CIFAR10. Both attacks are white-box with respect to the network. The first is black-box with respect to the detector. It only knows if an adversarial sample has been detected or not. The second has some knowledge about the detector. It knows what features it uses and can attack it directly to find adversarial features. We test these attacks by looking at the percentage of detected successful adversarial samples that they turn into undetected successful adversarial samples that fool both the network and the detector.

The first attack proceeds as follows. A strong white-box attack (CW) is used on the network on image x that has label y. If it finds a successful adversarial image $\tilde{x}$ that fools the network into predicting $\tilde{y} \neq y$ but is detected by the detector, the attacker will attempt to modify this image $\tilde{x}$ so that the network and the detector are both fooled. For this, the image $\tilde{x}$ is used as the initialization for an attack (HSJ with a budget of 50 iterations and 10000 evaluations) on a black-box Network-Detector system. The attacker considers that the Network-Detector behaves as follows: it outputs the class prediction of the network if the detector does not detect an attack and outputs an additional 'detected' class if the detector detects an attack. The attacker attacks this Network-Detector on image $\tilde{x}$ targeting the $\tilde{y}$ label. This way the network makes a mistake and the 'detected' class is avoided. On the vanilla ResNet110, this attack turns 16.5% of 1700 detected successful adversarial samples $\tilde{x}$ into undetected successful adversarial samples on our detector, compared to 25.7% on the Mahalanobis detector. These percentages are lower on the LAP-ResNet110 as they drop to 6.8% on our detector and 12.9% on the Mahalanobis detector. This shows that LAP training improves the robustness of both adversarial detectors to being attacked themselves, and that the Transport detector is more robust than the MH detector.

The second attack is very similar to the adaptive attack used in [9] to break the Kernel Density detector of [16]. It proceeds as follows. A strong white-box attack (CW) is used on the network on image x that has label y. If it finds a successful adversarial image $\tilde{x}$ that fools the network but is detected by the

detector, the detection features $\tilde{z}$ that $\tilde{x}$ generates when run through the network are used as the initialization for a black-box attack (HSJ with a budget of 50 iterations and 10000 evaluations) on the detector. If successful adversarial detection features z^* that fool the detector are found, the attacker has to find an adversarial perturbation of x that still fools the network and that generates these features z^* (or close features that also fool the detector) when run through the network. We do this as in [9] by solving the following optimization problem:

$$\min_{x^*} -L(N(x^*), y) + c_1\|D(x^*) - z^*\| + c_2\|x^* - x\| \tag{17}$$

where L is the cross-entropy loss, N is the network, and D is the (differentiable) function that returns the detection features of its input. This optimization problem is differentiable and we try differentiable optimization algorithms such as BFGS and NR to solve it. The initial detected successful adversarial image $\tilde{x}$ is used as initialization as in [9]. This attack turns 14% of detected successful adversarial samples $\tilde{x}$ into undetected successful adversarial samples on our detector on the LAP-ResNet110.

Given that initial detection rates of successful adversarial samples are almost 100% (see Appendix D.7), this shows that adaptive attacks do not (at least not easily) circumvent the detector, as detection rates drop to 85% at worst. Obviously, the second attack is stronger than the first one, but it can probably still be improved by using a white-box attack that is specific to random forests for attacking the detector such as [25] or [61], or a different loss than cross-entropy such as the one used in the CW attack. However, the difficulty of combining the attack on the network with that on the detector remains. It is the non-differentiability of the random forest that forces either this separate treatment of network and detector then the use of a proxy differentiable term for the detector (here $\|D(x^*) - z^*\|$ in (17)) to combine both, or the use of a black-box method as in the first attack. Also, we did not consider here the ensemble of the class-conditional detector and the general detector, which is the best performing version of the detector (see Section 5.2), and should be even more robust to adaptive attacks, as the attacker will have to fool two random forest detectors at once and target a particular label, constraining further the optimization problem he solves.