ADVERSARIAL DETECTION THROUGH TRANSPORT STATISTICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Adversarial attacks are perturbations to the input that don't change its class for a human observer, but fool a neural network into changing its prediction. In this paper, we propose a detector of adversarial attacks that is based on the interpretation of residual networks as discrete dynamic systems. The detector tells clean samples from abnormal ones by comparing the discrete vector fields they follow before the final classification layer. We compare this detector favorably to other detectors on seen and unseen attacks. We also show that regularizing the aforementioned vector field during training makes the network more regular on the support of the input distribution, thus making the network's activations on clean samples more distinguishable from those of abnormal samples. This makes adversarial detectors, ours and others, perform better, while improving the test accuracy of the network.

1 INTRODUCTION

Neural networks have significantly improved performances on many machine learning tasks, including computer vision tasks such as image classification (He et al. (2016a); Krizhevsky et al. (2012); Wightman et al. (2021)). They are however vulnerable to so-called adversarial attacks (Szegedy et al. (2013); Goodfellow et al. (2015); Carlini & Wagner (2017b)). These attacks modify an image in a way that is imperceptible to the human eye but that fools the network into wrongly classifying the modified image. These attacks can be carried out in the physical word (e.g. causing autonomous cars to misclassify road signs (Eykholt et al. (2018))), can generalize between networks (Moosavi-Dezfooli et al. (2017); Liu et al. (2017)), and can be generated without any access to the network (Liu et al. (2017)). Developing networks that are robust to such attacks or accompanied by detectors that can detect them is indispensable to deploying them safely in the real word (Amodei et al. (2016)).

In this paper, we focus on detecting these adversarial attacks. Neural classifiers trained a softmax classifier are not well-calibrated and produce highly overconfident predictions even for attacked and out-of-distribution inputs (Nguyen et al. (2015)). This makes it difficult to detect such inputs simply through the softmax outputs of the network, and explains the need for a separate adversarial detector. A detector is a separate system capable of predicting whether an input at test time has been adversarially modified or not. Detectors are typically trained separately after training the network on a dataset made up of clean and attacked images. While simply training the detector on these inputs (or their Fourier features for example) without running them through the network has been tried, the same papers find that using the intermediate representations of the inputs works better (Harder et al. (2021); Metzen et al. (2017)). Detection methods then vary essentially by which network activations to extract and how to statistically process them to extract the features that are fed to the classifier that distinguishes between clean and attacked images (Aldahdooh et al. (2022)). An important property is therefore the dimension of these extracted features, as simply using all activations is prohibitive.

We make two contributions in this paper. First, we propose a detection method for adversarial attacks that is based on the dynamical viewpoint of neural networks as dynamic systems that move the input points in space, time being represented by depth, to separate them before applying a final linear classification layer (Weinan (2017); Lu et al. (2018)). Our detector follows the trajectory of samples in space through time to differentiate clean and attacked images. This means that the statistics that we extract are the positions of the successive embeddings in space represented by their norms and cosines to a fixed vector. Given their resemblance to an Euler scheme for an ordinary differential equation, residual networks (ResNets) (He et al. (2016a;b); Weinan (2017)) are particularly amenable

to this analysis and method, but they can be applied to any architecture. Furthermore, Wu et al. (2020) show an increased vulnerability of residual-type architectures with skip connections to transferable adversarial attacks, precisely because of the presence of skip connections. Since skip connections are a basic building block in many architectures, and ResNets themselves and their variants such as WideResNet (Zagoruyko & Komodakis (2016)) and ResNeXt (Xie et al. (2017)) remain competitive (Wightman et al. (2021)), a detector that is well adapted to residual-type architecture with skip connections is welcome. We test our detector on seven different attacks, three datasets and three architectures, comparing it favorably to the reference Mahalanobis detector (Lee et al. (2018)).

As a second contribution, we regularize the trajectory of the samples in space by reprising the regularization of the transport cost of residual networks proposed in Karkar et al. (2020). This regularization can be applied on any architecture on layers that have the same input and output dimension, and was shown to improve generalization and training stability. We further show experimentally that it increases the detectability of adversarial attacks, meaning that when used during training it improves both the classification test accuracy and the accuracy of adversarial detection systems trained later, for our detection method and also for other detectors. We prove that the regularization achieves this by making the network more regular on the support of the input distribution, thus making the activations of abnormal samples outside of this distribution more distinguishable from those of clean samples.

2 Related Work

An adversarial attack aims at slightly modifying an input at test time by no more than ε in a certain metric (an L_p norm usually) so that the perturbation is undetectable to the human and that the perturbed image is misclassified by the network. White-box attacks are attacks where the adversary has access to the network's architecture and weights. We use five of these attacks. The Fast Gradient Method (FGM) (Goodfellow et al. (2015)) takes a perturbation step in the direction of the gradient that maximizes the loss. Projected Gradient Descent (PGD) (Madry et al. (2018)) and the Basic Iterative Method (BIM) (Kurakin et al. (2017)) are iterative versions of FGM. Two slower but more effective attacks are DeepFool (DF) (Moosavi-Dezfooli et al. (2016)), which iteratively perturbs an input in the direction of the closest decision boundary, and Carlini-Wagner (CW) (Carlini & Wagner (2017a)), which solves an optimization problem to find the perturbation. Black-box attacks don't have any knowledge about the the network. We use two of these attacks: Hop-Skip-Jump (HSJ) (Chen et al. (2020)), which estimates the gradient direction at the decision boundary, and the Boundary Attack (BA) (Brendel et al. (2018)), which starts from a large adversarial sample and moves towards the boundary decision to minimize the perturbation. Two main defense mechanisms exist against adversarial attacks: adversarial robustness and adversarial detection.

Adversarial robustness means training a network that is not easily fooled by adversarial attacks. This has been attempted through adversarial training (augmenting the training data with adversarial samples (Szegedy et al. (2013); Goodfellow et al. (2015))), regularization (Gu & Rigazio (2015)), distillation (Papernot et al. (2016)) and denoising (Meng & Chen (2017)). These defense mechanisms can however generally be avoided by newer attacks (Carlini & Wagner (2017b)), and a theoretical debate exists over a trade-off that is observed in practice between accuracy and adversarial robustness (Yang et al. (2020); Raghunathan et al. (2020); Schmidt et al. (2018); Dohmatob (2019)).

Given these limitations of adversarial robustness, a detector capable of detecting adversarial attacks is an important addition to a network in deployment. An advantage to having a detector is that it can, at least in principle, be used to also detect out-of-distribution samples, for which the notion of robustness does not make sense. An early idea was to use a second neural network as a detector (Metzen et al. (2017)). However, this network can also be adversarially attacked. More popular recent methods using statistical approaches include LID (Ma et al. (2018)), which trains the detector on the local intrinsic dimensionality of activations, and the Mahalanobis detector (Lee et al. (2018)), which trains the detector on the Mahalanobis distances between the activations and a Gaussian fitted to them during training. Detectors trained in the Fourier domain of activations have also been proposed in Harder et al. (2021). See Aldahdooh et al. (2022) for a review on adversarial detection.

Our second contribution is to regularize the network in a way that makes it Hölder-continuous (a weaker version of Lipschitz continuity), but only on the support of the data distribution. Estimation of the Lipschitz constant of a network has been used as estimates of its robustness to attacks in Weng et al. (2018); Szegedy et al. (2013); Virmaux & Scaman (2018); Hein & Andriushchenko (2017), 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 to attacks in Hein & Andriushchenko (2017); Cisse et al. (2017). These regularizations (Cisse et al. (2017); Gu & Rigazio (2015)) often work directly on the weights of the network, therefore making the network more regular on the entire input space. The difference with our method is that we only endue the network with regularity on the support of the clean samples. This won't make it more robust to attacks, but it will make the activations of the clean samples closer to each other and further from those of abnormal samples, thus making adversarial detection easier.

Modeling neural networks, especially residual ones as discrete dynamical systems (Weinan (2017)) has led to new architectures that imitate numerical schemes (Lu et al. (2018)), the neural ODE model as a continuous limit (Chen et al. (2018)) and to regularizations inspired by the dynamic formulation of optimal transport (Finlay et al. (2020)). In the context of adversarial defense, the view of residual networks as transport systems was used in Wang et al. (2019) to relate injecting noise into residual networks and ensembling them to approximating the Feynman-Kac formula for convection-diffusion, which makes the level sets more regular and the network more robust to adversarial attacks.

3 BACKGROUND

Here, we present the notions of optimal transport that we will use. We then present the dynamical viewpoint of ResNets and the transport regularization we use to improve adversarial detection.

3.1 OPTIMAL TRANSPORT

Let α and β be absolutely continuous distributions supported on a compact set $\Omega \subset \mathbb{R}^d$. We consider the Monge problem of looking for a transport map $T : \mathbb{R}^d \to \mathbb{R}^d$ transporting α to β, i.e. such as $T_\sharp \alpha = \beta$, with minimal total Euclidean transport cost:

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

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 as necessary, Problem (1) is equivalent to

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

and $T^\star(x) = \phi_1^x$ for ϕ that solves $\partial_t \phi_t^x = v_t^\star(\phi_t^x)$ with $v^\star$ that minimizes (2). More background on optimal transport is in Appendix (A) and we refer to Villani (2008); Santambrogio (2015).

3.2 LEAST ACTION PRINCIPLE RESIDUAL NETWORKS

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

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

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 disentangle them, the depth representing time, before classification by a linear layer. Karkar et al. (2020) look for a network that solves the task while having minimal transport cost:

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

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. The corresponding dynamical version of (4) is

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

Karkar et al. (2020) show that (4) and (5) are equivalent and have a solution such that T (or the associated vector field v) 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

$$\begin{aligned} \min_{\theta} \quad & \mathcal{C}(\theta) = \sum_{x \in \mathcal{D}} \sum_{m=0}^{M-1} \|r_m(\varphi_m^x)\|^2 \\ \text{subject to} \quad & \varphi_{m+1}^x = \varphi_m^x + h r_m(\varphi_m^x), \;\; \varphi_0^x = x \text{ for all } x \in \mathcal{X}, \;\; \mathcal{L}(\theta) = 0 \end{aligned} \tag{6}$$

4 METHOD

We take the view that a residual network moves its inputs according to a discrete vector field to separate them. Points in each class follow similar trajectories. Heuristically, for a successful adversarial sample that lies close to clean samples, the vector field it follows has to be different at some step from that of the clean samples (in its original class and in other classes), so that it can join the trajectory of the points in another class. In Section 4.1, we present how to detect adversarial attacks by looking at these trajectories. In Section 4.2, we discuss the link between the regularity of neural networks and their robustness to adversarial attacks and use the transport regularization of Karkar et al. (2020) presented in Section 3.2 to improve detectability of adversarial attacks.

4.1 DETECTION

Given a residual network that applies $x_{m+1} = x_m + h r_m(x_m)$ 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 cosines with a constant vector of ones as features to train our adversarial detector on. The embeddings x_m and the residues $r_m(x_m)$ can equivalently describe the trajectory of x_0 in space through the blocks, but in practice we find that the residues $r_m(x_m)$ give slightly better detection performance. We train different classifiers (logistic regression, random forest and support vector machines from scikit-learn (Pedregosa et al. (2011))) on these features. We find that a random forest always works best.

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 (Lee et al. (2018)) and LID (Ma et al. (2018)) that extract one feature per residual stage (a residual stage is a group of residual blocks that keep the same dimension). And even for common large architectures, twice the number of residual blocks is still a small number of features for training a binary classifier. More importantly, the features we extract from the activations (norms and cosines) are simple and quick to calculate, whereas those of other methods require more involved statistical computations on the activations. We include in Section 5.4 a favorable time comparison of our detector to the Mahalanobis detector. Another advantage of our method is that it does not have a hyper-parameter to tune using cross-validation or another technique as opposed to the Mahalanobis and LID detectors.

We tried augmenting the clean samples dataset by adding randomly perturbed samples to be considered as clean as in Lee et al. (2018), with the same noise amplitude as they. But we find that this does not improve detection accuracy in most cases. We also tried a class-conditional version where the features are grouped by the predicted class 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 trained on all samples regardless of the predicted class is also trained to be used in case a particular class is never targeted by adversarial attacks (which might be concentrated on certain classes). We also tried ensembling the class-conditional detector with the class-agnostic one. This works better in some cases, and we used it also for the methods we compare our detector to. Finally, we tried ensembling a detector trained

only on the norms with one trained only on the cosines and one trained on both. We test two ways of ensembling: either a sample is considered an attack if at least one of the detectors detects an attack, or the detectors vote. In the experimental results, we report the results of the best performing version, which is often the first kind of ensembling of a class-conditional detector and of a class-agnostic detector, without noisy samples added during detection training.

4.2 REGULARIZATION

Regularity of neural networks (typically Lipschitz continuity) has been used as a measure of their robustness to adversarial attacks Weng et al. (2018); Szegedy et al. (2013); Virmaux & Scaman (2018); Hein & Andriushchenko (2017); Cisse et al. (2017). 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 therefore more robust has therefore been tried in Hein & Andriushchenko (2017); Cisse et al. (2017). 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 regularize the weights of the network's matrices.

We propose here an alternate 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 $f \in \mathcal{C}^{0,\eta}(X)$). If this result does not apply around the input distribution's support, particularly in the adversarial zones, 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. While the theory only gives a rigorous result on the support of the clean data, 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 4.2.1 and prove the regularity it endues the network with in Section 4.2.2.

4.2.1 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. The residual function r_m we find are then our approximation of the vector field v that solves the continuous version (5) of the problem.

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

$$\begin{cases} \theta_{i+1} = \arg\min\limits_{\theta}\ \mathcal{C}(\theta) + \lambda_i\, \mathcal{L}(\theta) \\ \lambda_{i+1} = \lambda_i + \tau\, \mathcal{L}(\theta_{i+1}) \end{cases} \tag{7}$$

We use SGD for $s > 0$ steps (i.e. batches) for the minimization in the first line of (7), starting from the previous θ_i. The hyper-parameters λ_0, τ and s of the transport regularization are chosen in the experiments to maximize test accuracy, not adversarial detection. When using a ResNeXt network, where a residual block applies $x_{m+1} = \text{ReLU}(x_m + r_m(x_m))$, we regularize the norms of the true residues $x_{m+1} - x_m$ instead of $r_m(x_m)$. The true residue $x_{m+1} - x_m$ can be used when using any non-residual architecture on layers that have the same input and output dimension, allowing to apply the method to any network with many such layers.

4.2.2 THEORETICAL ANALYSIS

We take $\Omega \subset \mathbb{R}^d$ convex and compact and the data distribution $\alpha \in \mathcal{P}(\Omega)$ absolutely continuous and 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 Karkar et al. (2020), we know that 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 it, i.e. $\|r_m - v_{t_m}\|_\infty \leq \varepsilon/2$ for all $0 \leq m < M$, with $t_m = m/M$. The first regularity result is:

Theorem 4.1. *For $a, b \in$ support(α_{t_m}), where $\alpha_t := (\phi_t^{\cdot})_\sharp \alpha$ with ϕ solving (5) along with v,*

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

and

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

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

Proof. A solution v to (5) exists and is linked to an optimal transport map T that is a solution to (4) by $v_t = (T - \mathtt{id}) \circ T_t^{-1}$ where $T_t := (1-t)\mathtt{id} + tT$ which is invertible (see Appendix A). By Theorem A.1 in Appendix A, being an optimal transport map, T is η-Hölder on X (i.e. $\forall\, a, b \in X$ we have $\|T(a) - T(b)\| \leq C\|a-b\|^\eta$ for some constants $C>0, 1\geq\eta>0$, and we denote this $T \in \mathcal{C}^{0,\eta}(X)$). Therefore, for all $a, b \in$ support(α_t) for any fixed $t \in [0,1]$ where $\alpha_t = (\phi_t^{\cdot})_\sharp \alpha = (T_t)_\sharp \alpha$ with ϕ solving (5) along with v, we have

$$\|v_t(a) - v_t(b)\| \leq \|T_t^{-1}(a) - T_t^{-1}(b)\| + C\|T_t^{-1}(a) - T_t^{-1}(b)\|^\eta \tag{8}$$

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 $\alpha_t = (T_t)_\sharp \alpha$, 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 A.2 in Appendix A) and is η_t-Hölder with $0<\eta_t\leq 1$ (it is easy to see that the support of α_t satisfies the conditions of Theorem A.1 in Appendix A). Therefore, for all $a, b \in$ support(α_t)

$$\|v_t(a) - v_t(b)\| \leq C_t\|a-b\|^{\eta_t} + CC_t^\eta\|a-b\|^{\eta\eta_t} \tag{9}$$

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

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

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 (10). □

Remark 4.2. 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 4.3. If the distributions α and β in Theorem 4.1 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.

We now want to look at the activations for network inputs in practice. 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 we (and other methods) use to compute features for adversarial detection are $r_m(a_m)$ and $r_m(b_m)$. We therefore want to bound $\|r_m(a_m) - r_m(b_m)\|$ as a function of the input variation $\|a_0 - b_0\|$. Bound (8) allows us to do this without multiplying the Hölder or the Lipschitz constants of the residual blocks up to depth m (or using more complex methods to estimate the Lipschitz constant of a neural network such as Virmaux & Scaman (2018); Latorre et al. (2020); Bhowmick et al. (2021)). We have the following result:

Theorem 4.4. *For any a_0, b_0 we have*

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

The term $\|a_m - \phi_{t_m}^{a_0}\|$ (likewise $\|b_m - \phi_{t_m}^{b_0}\|$) 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 vector field v up to time t_m. Under more regularity conditions on v, it is possible to bound this term. If v is $\mathcal{C}^1$ and Lipschitz in x (this is not stronger than the regularity we get on v through our regularization, because it does not give bound (8)), we have, for constants $R, S > 0$,

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

The proof is in Appendix C.

5 EXPERIMENTS

5.1 PRELIMINARY EXPERIMENTS

The regularization was shown in Karkar et al. (2020) to improve generalization. We further show that it improves training stability in Figure 2 in Appendix D.2. In Figure 3 in Appendix D.2, we show that it does indeed make the transport costs of clean and attacked samples more different from each other.

5.2 DETECTION OF SEEN ATTACKS

We test our method on five white-box attacks: FGM (Goodfellow et al. (2015)), PGD (Madry et al. (2018)), BIM (Kurakin et al. (2017)), DF (Moosavi-Dezfooli et al. (2016)) and CW Carlini & Wagner (2017a), and two black-box attacks BA (Brendel et al. (2018)) and HSJ (Chen et al. (2020)). For FGM, PGD and BIM, we use $\epsilon = 0.03$. For CW, we use the L_2 version. We use the adversarial robustness toolbox (ART) (Nicolae et al. (2018)) Python package for generating the attacks. We compare our method to the popular Mahalanobis detector (Lee et al. (2018)), whose code we use. For the magnitude parameter, we try the values tried in their paper and we report the best results. Code is available at `github.com/advadvadvadvadv/adv`.

Our experimental protocol is as follows. For each architecture-dataset pair, the architecture is trained with the same hyper-parameters on the training set with and without the transport regularization. The parameters of the regularization are chosen to maximize test accuracy, not adversarial detection. A network trained with the regularization is called a LAP-network (for Least Action Principle).

The rest of the data (the provided test set) is split in $0.9/0.1$ proportions into two datasets, B1 and B2. Dataset B2 is used to evaluate the test accuracy of the network. For each image in B1 (respectively B2), an adversarial attack is generated and a balanced detection training set (respectively a detection test set) is created. We try augmenting the dataset with a randomly perturbed version of each image, to be considered as clean during detection training as in Lee et al. (2018), but we find that this does not improve detection accuracy. This dataset creation protocol is standard and is depicted in Figure 1 in Appendix D.1. We try limiting the dataset to successfully attacked images only (i.e. images that are classified correctly but whose adversarial versions fool the network) as in Lee et al. (2018), but we don't find a significant improvement in detection (even on successfully attacked images only).

Samples in the detection training set are fed through the network and the features for each detection method are extracted. The classifiers (logistic regression, random forest and support vector machines from scikit-learn (Pedregosa et al. (2011))) are trained on these features for each detection method and tested on the features extracted from the samples in detection test set, and a random forest always has the best results. This is done for both the vanilla version of the network and its LAP version.

We report here the detection accuracy of each detection method on the detection test set for both the vanilla version of the network and its LAP version in Table 1 below. In each cell, the first number corresponds to the vanilla network and the second to the regularized network. Our results show two things. First, our detection method performs better than the other methods, with or without the regularization. Second, all detection methods work better on the LAP versions of the networks most times. The Mahalanobis method benefits more from the regularization, which is expected as our method already uses the quantities that are being regularized.

5.2.1 WHITE-BOX ATTACKS

The first experiment is using a ResNeXt50 (Xie et al. (2017)) on CIFAR100 (Krizhevsky (2009)). 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 LAP training we use algorithm (7) with $\tau = 1$, $s = 1$ and $\lambda_0 = 1$. Without the transport regularization the network has a test accuracy of 76.4%, while the LAP-ResNeXt50 has a test accuracy of 79%. Results are in the first row of Table 1. In this experiment, our detector outperforms the Mahalanobis detector by 1 to 5 percentage points depending on the attack on the vanilla ResNeXt50. The transport regularization improves the detection accuracy of our detector and of the Mahalanobis detector by 0.7 percentage points on average. On all attacks, the best detector is the transport detector (on the LAP-ResNeXt50 for FGM, PGD, BIM and CW and on the vanilla ResNeXt50 for DF). These results are averaged over 5 runs and the standard deviations are in Table 4 in Appendix D.3.

The second experiment is using a ResNet110 (He et al. (2016a)) on CIFAR10 (Krizhevsky (2009)). 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 (Saxe et al. (2014)) with a gain of 0.05, a batch size of 256, weight decay of 0.0001 and gradient clipping at 5. For LAP training we use algorithm (7) with $\tau = 1$, $s = 1$ and $\lambda_0 = 1$. Without the transport regularization the network has a test accuracy of 92.7%, while the LAP-ResNet110 has a test accuracy of 93.3%. Results are in the second row of Table 1. In this experiment, our detector outperforms the Mahalanobis detector by 9 to 13 percentage points depending on the attack on the vanilla ResNet110. The transport regularization improves the detection accuracy of our detector by 1.3 percentage points on average and that of the Mahalanobis detector by a substantial 8 percentage points on average. On all attacks, the best detector is the transport detector (on the LAP-ResNet110 for FGM, PGD, BIM and CW and on the vanilla ResNet110 for DF). These results are averaged over 5 runs and the standard deviations are in Table 5 in Appendix D.3.

The third experiment is using a WideResNet (Zagoruyko & Komodakis (2016)) 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 (Saxe et al. (2014)) with a gain of 0.1, a batch size of 114 and weight decay of 0.0001. For LAP training we use algorithm (7) with $\tau = 1$, $s = 1$ and $\lambda_0 = 1$. Without the transport regularization the network has a test accuracy of 65.14%, while the LAP-WideResNet has a test accuracy of 65.34%. Results are in the third row of Table 1. In this experiment, our detector outperforms the Mahalanobis detector by 11 to 12 percentage points depending on the attack on the vanilla WideResNet. The transport regularization does not change the detection accuracy of our detector and improves that of the Mahalanobis detector 0.8 percentage points on average. On all attacks, the best detector is the transport detector. These results are averaged over 5 runs and the standard deviations are in Table 6 in Appendix D.3.

Table 1: Average detection accuracy (5 runs) of seen white-box attacks using Network/LAP-Network.

		Attack				
Network/Data	Detector	FGM	PGD	BIM	DF	CW
ResNeXt50	Transport (ours)	97.2/**98.4**	96.1/**97.9**	97.9/**98.9**	**91.9**/91.2	95.7/**95.8**
CIFAR100	Mahalanobis	95.7/96.8	93.9/94.7	96.0/97.7	87.1/86.8	94.7/94.7
ResNet110	Transport (ours)	97.1/**98.7**	94.3/**97.4**	97.4/**99.2**	**98.1**/98.0	98.0/**98.1**
CIFAR10	Mahalanobis	87.5/95.5	81.8/90.4	86.6/95.1	88.9/95.7	85.4/93.2
WideResNet	Transport (ours)	**95.9**/95.9	96.0/**96.2**	96.6/**96.6**		
TinyImageNet	Mahalanobis	84.6/85.3	84.2/85.1	84.5/85.2		

5.2.2 Black-box attacks

Since they are much slower to generate, we test detection of seen black-box attacks only on the network-data pair of ResNet110-CIFAR10 in Table 2 below. The experimental protocol and the ResNet110 and LAP-ResNet110 networks are the same as above. Our proposed detector outperforms the Mahalanobis detector by 13 percentage points on HSJ and by 16 percentage points on BA. The regularization improves the detection accuracy of the Mahalanobis detector by 8 percentage points on HSJ and by 9 percentage points on HSJ, and that of the transport detector slightly. The transport detector is still the best detector by 5 percentage points on HSJ and by 7 percentage points on BA.

Table 2: Average detection accuracy and standard deviation (5 runs) of seen black-box attacks using ResNet110/LAP-ResNet110 on CIFAR10.

	Attack	
Detector	HSJ	BA
Transport (ours)	97.38 ± 0.60 / **97.38** ± 0.43	96.56 ± 0.56 / **97.02** ± 0.18
Mahalanobis	84.14 ± 1.07 / 92.30 ± 0.46	80.20 ± 2.11 / 89.62 ± 0.43

5.3 DETECTION OF UNSEEN ATTACKS

An important setting is when we do not know which attack might be used at test time or we only have time to train our detector on a fast attack. We still want our detector to generalize well to detecting other unseen attacks. To test this, we use the same trained network as above, on the same datasets. The detectors are trained on the detection training set created using the simplest and quickest attack (FGM) on the detection training sets and tested on the detection test sets created using the six other attacks (PGD, BIM, DF, CW, HSJ and BA). We use only a tenth of the detection test sets for the slower black-box attacks HSJ and BA. Results are in Table 3 below. We see that our detection method has very good detection generalization to unseen attacks, even those very different from FGM, comfortably better than the Mahalanobis detector, by up to 18 percentage points in some cases. These results are averaged over 5 runs and the standard deviations are in Tables 7, 8 and 9 in Appendix D.4.

Table 3: Average detection accuracy (5 runs) of unseen attacks after training on FGM.

		Attack					
Network/Data	Detector	PGD	BIM	DF	CW	HSJ	BA
ResNeXt50	Transport (ours)	**90.82**	**96.12**	**87.60**	**87.54**	**89.90**	**81.70**
CIFAR100	Mahalanobis	86.90	90.92	83.08	84.02	84.10	78.70
ResNet110	Transport (ours)	**89.58**	**96.28**	**91.04**	**93.24**	**93.10**	**92.30**
CIFAR10	Mahalanobis	76.88	76.56	79.70	79.54	79.10	79.90
WideResNet	Transport (ours)	**93.28**	**95.06**	**90.72**	**91.56**		
TinyImageNet	Mahalanobis	77.00	77.00	72.58	74.90		

However, we see in Tables 7, 8 and 9 in Appendix D.4 that the regularization does not improve detection accuracy as much anymore, and reduces it in some cases. It improves it for the Mahalanobis detector on all attacks on the ResNet110-CIFAR10 and WideResNet-TinyImageNet experiments (by up to 10 percentage points), but not on most attacks on the ResNeXt50-CIFAR100. It often reduces detection generalization on our detector. We claim this is because the regularization reduces the variance of features extracted during training on the seen attack, thus harming generalization to unseen attacks. This explains why detection of PGD and BIM, which are variants of FGM, still often improves. Fixing this could be an area a future research.

5.4 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 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).

6 CONCLUSION

We proposed a method for detecting abnormal samples, based on dynamical viewpoint of neural networks. The method examines the discrete vector field moving the inputs to distinguish clean and adversarial samples. The detector requires minimal processing and computation to extract the features it needs for detection and achieves state-of-the-art detection performances on 3 different architectures and datasets, both on seen and unseen attacks. We have also used a transport regularization that was previously shown to improve test accuracy to improve the detection accuracy of adversarial detectors. This regularization makes the embeddings closer to each other on the support of the input distribution, thus making them more distinguishable from those of out-of-distribution samples.

REFERENCES

Ahmed Aldahdooh, Wassim Hamidouche, Sid Ahmed Fezza, and Olivier Déforges. Adversarial example detection for dnn models: a review and experimental comparison. *Artificial Intelligence Review*, 2022.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety, 2016. URL https://arxiv.org/abs/1606.06565.

J.D. Benamou and Y. Brenier. A computational fluid mechanics solution to the monge-kantorovich mass transfer problem. *Numerische Mathematik*, 2000.

Aritra Bhowmick, Meenakshi D'Souza, and G. Srinivasa Raghavan. Lipbab: Computing exact lipschitz constant of relu networks. *CoRR*, abs/2105.05495, 2021. URL https://arxiv.org/abs/2105.05495.

Wieland Brendel, Jonas Rauber, and Matthias Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=SyZI0GWCZ.

N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 39–57, Los Alamitos, CA, USA, may 2017a. IEEE Computer Society. doi: 10.1109/SP.2017.49. URL https://doi.ieeecomputersociety.org/10.1109/SP.2017.49.

Nicholas Carlini and David Wagner. *Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods*, pp. 3–14. Association for Computing Machinery, New York, NY, USA, 2017b. ISBN 9781450352024. URL https://doi.org/10.1145/3128572.3140444.

Jianbo Chen, Michael I. Jordan, and Martin J. Wainwright. Hopskipjumpattack: A query-efficient decision-based attack. In *2020 IEEE Symposium on Security and Privacy*, pp. 1277–1294. IEEE, 2020. URL https://doi.org/10.1109/SP40000.2020.00045.

R. T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. Neural ordinary differential equations. In *NeurIPS*, 2018.

Moustapha Cisse, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. Parseval networks: Improving robustness to adversarial examples. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 854–863. PMLR, 2017.

Elvis Dohmatob. Limitations of adversarial robustness: Strong no free lunch theorem. In *Proceedings of the 36th International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2019.

K. Eykholt, I. Evtimov, E. Fernandes, B. Li, A. Rahmati, C. Xiao, A. Prakash, T. Kohno, and D. Song. Robust physical-world attacks on deep learning visual classification. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1625–1634, Los Alamitos, CA, USA, jun 2018. IEEE Computer Society. doi: 10.1109/CVPR.2018.00175. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00175.

A. Figalli. *The Monge-Ampere Equation and Its Applications*. Zurich lectures in advanced mathematics. European Mathematical Society, 2017.

Chris Finlay, Jorn-Henrik Jacobsen, Levon Nurbekyan, and Adam M Oberman. How to train your neural ode. In *ICML*. PMLR, 2020.

Ian J. Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Shixiang Gu and Luca Rigazio. Towards deep neural network architectures robust to adversarial examples. In *ICLR (Workshop)*, 2015.

Paula Harder, Franz-Josef Pfreundt, Margret Keuper, and Janis Keuper. Spectraldefense: Detecting adversarial attacks on cnns in the fourier domain, 2021. URL https://arxiv.org/abs/2103.03000.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016a.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *ECCV*, 2016b.

Matthias Hein and Maksym Andriushchenko. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *Advances in Neural Information Processing Systems*, pp. 2263–2273, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Skander Karkar, Ibrahim Ayed, Emmanuel de Bézenac, and Patrick Gallinari. A principle of least action for the training of neural networks. In *ECML-PKDD*, 2020.

Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto Technical Report*, 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C.J. Burges, L. Bottou, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial examples in the physical world. In *ICLR (Workshop)*, 2017.

Fabian Latorre, Paul Rolland, and Volkan Cevher. Lipschitz constant estimation of neural networks via sparse polynomial optimization. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rJe4_xSFDB.

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/abdeb6f575ac5c6676b747bca8d09cc2-Paper.pdf.

Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. Delving into transferable adversarial examples and black-box attacks. In *International Conference on Learning Representations*. OpenReview.net, 2017. URL https://openreview.net/forum?id=Sys6GJqxl.

Yiping Lu, Aoxiao Zhong, Quanzheng Li, and Bin Dong. Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations. In *Proceedings of the International Conference on Machine Learning (ICML 2018)*, 2018.

Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Dawn Song, Michael E. Houle, and James Bailey. Characterizing adversarial subspaces using local intrinsic dimensionality. In *ICLR*, 2018.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018. URL https://openreview.net/forum?id=rJzIBfZAb.

Dongyu Meng and Hao Chen. Magnet: A two-pronged defense against adversarial examples. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, CCS '17, pp. 135–147, New York, NY, USA, 2017. Association for Computing Machinery. ISBN 9781450349468. doi: 10.1145/3133956.3134057. URL https://doi.org/10.1145/3133956.3134057.

Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. On detecting adversarial perturbations. In *ICLR*, 2017.

S. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard. Universal adversarial perturbations. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 86–94, Los Alamitos, CA, USA, 2017. IEEE Computer Society. doi: 10.1109/CVPR.2017.17. URL https://doi.ieeecomputersociety.org/10.1109/CVPR.2017.17.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 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.

Anh Nguyen, Jason Yosinski, and Jeff Clune. 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. doi: 10.1109/CVPR.2015.7298640.

Maria-Irina Nicolae, Mathieu Sinn, Minh Ngoc Tran, Beat Buesser, Ambrish Rawat, Martin Wistuba, Valentina Zantedeschi, Nathalie Baracaldo, Bryant Chen, Heiko Ludwig, Ian M. Molloy, and Ben Edwards. Adversarial robustness toolbox v1.0.0, 2018. URL https://arxiv.org/abs/1807.01069.

Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE Symposium on Security and Privacy (SP)*, pp. 582–597, 2016. doi: 10.1109/SP.2016.41.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Alfio Quarteroni, Riccardo Sacco, and Fausto Saleri. *Numerical Mathematics*. Springer Berlin, Heidelberg, 2007.

Aditi Raghunathan, Sang Michael Xie, Fanny Yang, John Duchi, and Percy Liang. Understanding and mitigating the tradeoff between robustness and accuracy. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7909–7919. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/raghunathan20a.html.

F. Santambrogio. *Optimal Transport for Applied Mathematicians*. Birkhäuser, 2015.

Andrew M. Saxe, James L. Mcclelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural network. In *International Conference on Learning Representations (ICLR 2014)*, 2014.

Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. Adversarially robust generalization requires more data. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/f708f064faaf32a43e4d3c784e6af9ea-Paper.pdf.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks, 2013. URL https://arxiv.org/abs/1312.6199.

Cédric Villani. *Optimal Transport: Old and New*. Springer-Verlag, 2008.

Aladin Virmaux and Kevin Scaman. Lipschitz regularity of deep neural networks: analysis and efficient estimation. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/d54e99a6c03704e95e6965532dec148b-Paper.pdf.

Bao Wang, Zuoqiang Shi, and Stanley Osher. Resnets ensemble via the feynman-kac formalism to improve natural and robust accuracies. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/file/d2ed45a52bc0edfa11c2064e9edee8bf-Paper.pdf.

E Weinan. A proposal on machine learning via dynamical systems. *Commun. Math. Stat*, 2017.

Tsui-Wei Weng, Huan Zhang, Pin-Yu Chen, Jinfeng Yi, Dong Su, Yupeng Gao, Cho-Jui Hsieh, and Luca Daniel. Evaluating the robustness of neural networks: An extreme value theory approach. In *International Conference on Learning Representations*, 2018.

Ross Wightman, Hugo Touvron, and Herve Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv*, 2021.

Dongxian Wu, Yisen Wang, Shu-Tao Xia, James Bailey, and Xingjun Ma. Skip connections matter: On the transferability of adversarial examples generated with resnets. In *International Conference on Learning Representations*, 2020.

Saining Xie et al. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.

Yao-Yuan Yang, Cyrus Rashtchian, Hongyang Zhang, Russ R Salakhutdinov, and Kamalika Chaudhuri. A closer look at accuracy vs. robustness. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 8588–8601. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper/2020/file/61d77652c97ef636343742fc3dcf3ba9-Paper.pdf.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016.

Jingfeng Zhang 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 Santambrogio (2015); Villani (2008)). 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 (Benamou & Brenier (2000)). 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 Santambrogio (2015) 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 Santambrogio (2015) for these results on optimal transport.

Optimal transport maps have some regularity properties under some boundedness assumptions. We mention the following result from Figalli (2017):

Theorem A.1. *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 A.2. *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*
- *$T \circ S \overset{\beta-a.s.}{=} \texttt{id}$ and $S \circ T \overset{\alpha-a.s.}{=} \texttt{id}$*

B Background on numerical methods for ODEs

We refer to Quarteroni et al. (2007) for this quick background on the 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 $h_{\text{max}} := \max h_n$. 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 B.1. *(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 $\max |e_n| \leq C h_{\text{max}}^p$ for a constant C that depends on f, t_0 and T.

Theorem B.2. (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 B.3. (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 B.4. (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 B.5. *(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

$$\frac{x_{n+1} - x_n}{h_n} = \phi(t_n, x_n, h_n) \text{ and } \frac{\tilde{x}_{n+1} - \tilde{x}_n}{h_n} = \phi(t_n, \tilde{x}_n, h_n) + \epsilon_n \tag{15}$$

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 B.6. (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 B.7. (Zero-stability of Euler's method) *If f is Lipschitz in its second variable, then Euler's method is stable.*

Definition B.8. *(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 B.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 B.10. (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 PROOF OF THEOREM 4.4

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

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

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 is the first bound in the theorem.

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 B.4, B.7 and B.10 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 (8)), 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 $\hat{a}_0 := \text{Proj}_X(a_o)$ and $\hat{b}_0 := \text{Proj}_X(b_o)$ to apply (8). 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 $\|a_0 - \hat{a}_0\| = \text{dist}(a_0, X)$ and $\|b_0 - \hat{b}_0\| = \text{dist}(b_0, X)$, we get the second bound in the theorem. 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 DETECTION TRAINING PROTOCOL

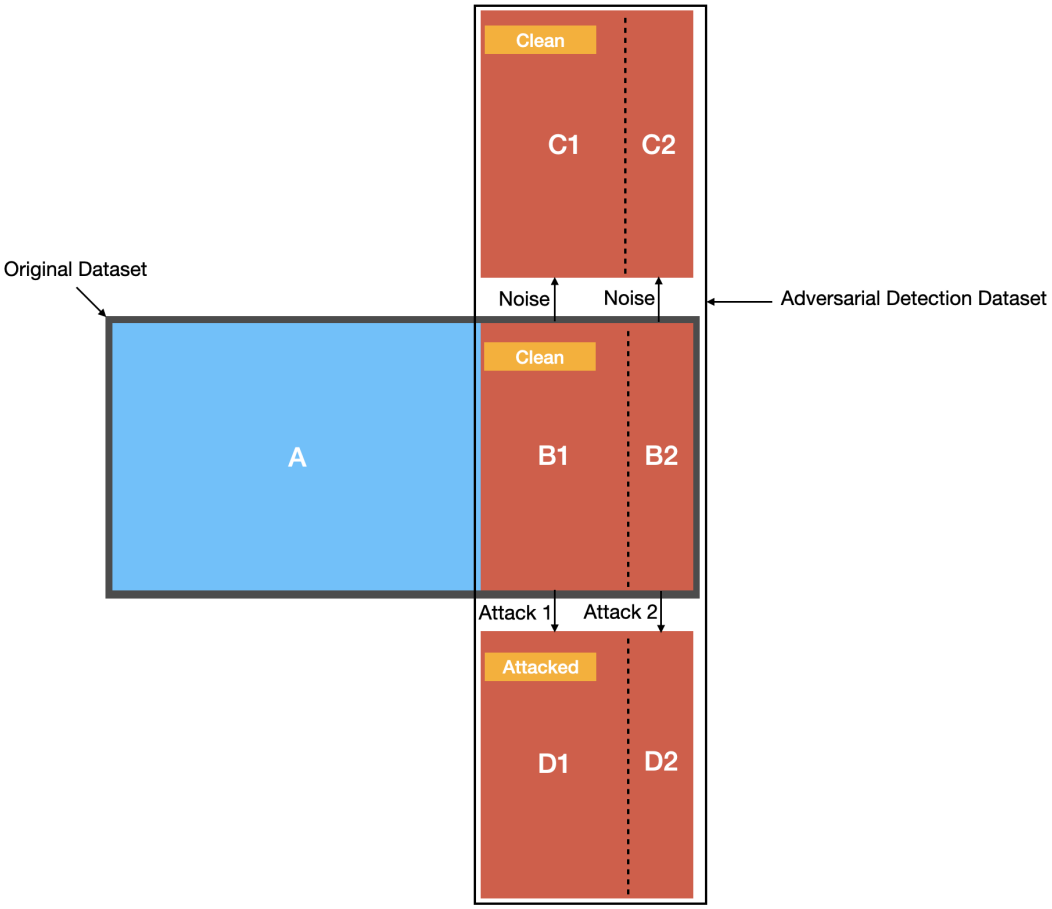

Figure 1: Adversarial detection dataset creation. A+B1+B2 is the original dataset, where A is the given training set and B1+B2 is the given test set. We create a noisy version of B1+B2 by adding random noise to each sample in B1+B2 to get C1+C2. Noisy samples are considered clean (i.e. not attacked) 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.2 PRELIMINARY EXPERIMENTS

We see in Figure 2 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 Zhang et al. (2019), 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 2 as this method, while also improving the test accuracy (Karkar et al. (2020)) which the small step method does not claim.

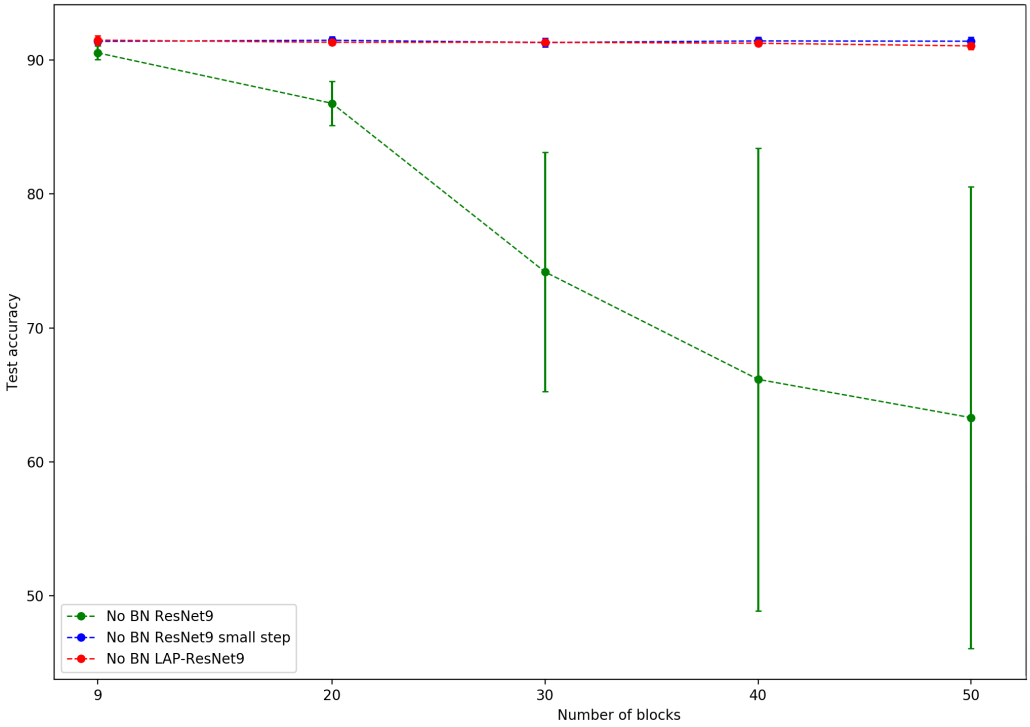

Figure 2: Test accuracy of ResNets of various depths without batch-normalization on CIFAR10.

We see in Figure 3 below that regularizing the transport cost makes this quantity more distinguishable between clean and attacked samples. In fact, we see that using this one quantity without any training by considering its empirical quantiles on clean samples allows then to detect some attacks with near perfect recall and a chosen fixed false positive rate.

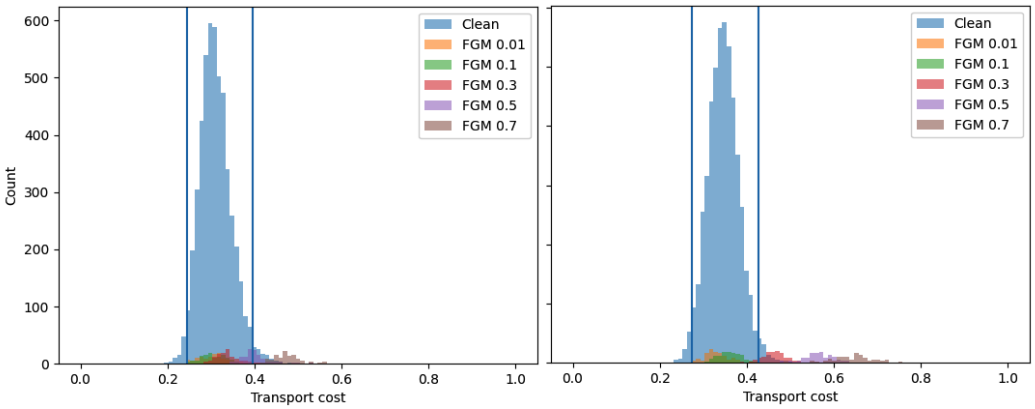

Figure 3: Histogram of transport cost for clean and attacked test samples on CIFAR100. The vertical lines represents the 0.02 and 0.98 empirical quantiles of the transport cost of the clean samples. Left: ResNeXt50. Right: LAP-ResNeXt50

D.3 DETECTION OF SEEN ATTACKS

In the tables below, VAN corresponds to detectors trained on a vanilla network and LAP to detectors trained on a transport-regularized network. 'Maha' refers to the Mahalanobis detector and 'Tra' or 'Transport' refers to our detector.

Table 4: Average adversarial detection accuracy and standard deviation over 5 runs using ResNeXt50 on CIFAR100.

	Attack				
Detector	FGM	PGD	BIM	DF	CW
VAN Transport	97.20 $\pm$ 0.46	96.06 $\pm$ 0.37	97.90 $\pm$ 0.42	**91.92** $\pm$ 0.65	95.66 $\pm$ 0.43
LAP Transport	**98.42** $\pm$ 0.42	**97.92** $\pm$ 0.60	**98.92** $\pm$ 0.18	91.24 $\pm$ 1.14	**95.78** $\pm$ 0.48
VAN Maha	95.72 $\pm$ 0.38	93.92 $\pm$ 0.67	95.96 $\pm$ 0.51	87.14 $\pm$ 0.59	94.66 $\pm$ 0.71
LAP Maha	96.82 $\pm$ 0.38	94.66 $\pm$ 0.69	97.74 $\pm$ 0.43	86.82 $\pm$ 0.78	94.74 $\pm$ 0.34

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

	Attack				
Detector	FGM	PGD	BIM	DF	CW
VAN Transport	97.12 $\pm$ 0.65	94.34 $\pm$ 0.65	97.42 $\pm$ 0.32	**98.06** $\pm$ 0.44	98.04 $\pm$ 0.44
LAP Transport	**98.74** $\pm$ 0.26	**97.36** $\pm$ 0.48	**99.24** $\pm$ 0.21	97.98 $\pm$ 0.32	**98.06** $\pm$ 0.37
VAN Maha	87.46 $\pm$ 3.65	81.80 $\pm$ 3.26	86.62 $\pm$ 3.28	88.94 $\pm$ 2.36	85.44 $\pm$ 2.70
LAP Maha	95.46 $\pm$ 0.46	90.38 $\pm$ 0.38	95.10 $\pm$ 0.58	95.74 $\pm$ 0.55	93.22 $\pm$ 0.64

Table 6: Average adversarial detection accuracy and standard deviation over 5 runs using WideResNet on TinyImageNet.

	Attack		
Detector	FGM	PGD	BIM
VAN Transport	**95.92** $\pm$ 0.10	96.04 $\pm$ 0.14	96.56 $\pm$ 0.07
LAP Transport	95.86 $\pm$ 0.11	**96.16** $\pm$ 0.07	**96.56** $\pm$ 0.07
VAN Maha	84.62 $\pm$ 1.32	84.20 $\pm$ 1.32	84.46 $\pm$ 1.29
LAP Maha	85.26 $\pm$ 0.96	85.10 $\pm$ 0.60	85.24 $\pm$ 1.11

D.4 DETECTION OF UNSEEN ATTACKS

Table 7: Average adversarial detection accuracy of unseen attacks after training on FGM and standard deviation over 5 runs using ResNeXt50 on CIFAR100.

	Attack					
Detector	PGD	BIM	DF	CW	HSJ	BA
VAN Tra	**90.8** $\pm$ 0.5	96.1 $\pm$ 0.6	**87.6** $\pm$ 0.2	**87.5** $\pm$ 0.4	**89.9** $\pm$ 2.2	**81.7** $\pm$ 1.6
LAP Tra	89.3 $\pm$ 0.8	**97.7** $\pm$ 0.3	76.0 $\pm$ 1.0	74.7 $\pm$ 1.1	78.1 $\pm$ 3.4	71.9 $\pm$ 3.9
VAN Maha	86.9 $\pm$ 0.1	90.9 $\pm$ 1.0	83.1 $\pm$ 0.7	84.0 $\pm$ 1.1	84.1 $\pm$ 3.4	78.7 $\pm$ 1.4
LAP Maha	86.7 $\pm$ 0.9	93.9 $\pm$ 0.4	79.4 $\pm$ 1.6	80.9 $\pm$ 2.0	80.5 $\pm$ 3.7	78.2 $\pm$ 2.0

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

	Attack					
Detector	PGD	BIM	DF	CW	HSJ	BA
VAN Tra	89.6 ± 1.3	96.3 ± 0.5	$\mathbf{91.0} \pm 1.0$	$\mathbf{93.2} \pm 1.2$	$\mathbf{93.1} \pm 2.5$	$\mathbf{92.3} \pm 3.2$
LAP Tra	$\mathbf{93.0} \pm 0.4$	$\mathbf{98.7} \pm 0.2$	75.4 ± 0.3	75.2 ± 0.6	77.2 ± 3.4	74.7 ± 2.1
VAN Maha	76.9 ± 2.4	76.6 ± 2.1	79.7 ± 3.0	79.5 ± 2.5	79.1 ± 4.2	79.9 ± 2.6
LAP Maha	87.6 ± 1.3	84.3 ± 1.0	80.5 ± 0.9	81.0 ± 0.8	81.1 ± 2.5	82.3 ± 3.3

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

	Attack			
Detector	PGD	BIM	DF	CW
VAN Tra	93.3 ± 0.1	$\mathbf{95.1} \pm 0.1$	$\mathbf{90.7} \pm 0.04$	91.6 ± 0.1
LAP Tra	$\mathbf{93.4} \pm 0.1$	94.9 ± 0.2	90.6 ± 0.2	$\mathbf{91.7} \pm 0.1$
VAN Maha	77.0 ± 1.0	77.0 ± 1.0	72.6 ± 1.1	74.9 ± 1.1
LAP Maha	77.4 ± 0.6	77.7 ± 0.9	73.3 ± 0.7	75.6 ± 0.8