Adversarial attacks on neural networks through canonical Riemannian foliations

Eliot Tron, Nicolas Couellan, Stephane Puechmorel

Adversarial attacks on neural networks through canonical Riemannian foliations

Eliot Tron[1*], Nicolas Couellan[1,2] and Stéphane Puechmorel[1,2]

[1*]Ecole Nationale de l'Aviation Civile, 7 Avenue Edouard Belin, Toulouse, 31400, France.
[2]Institut de Mathématiques de Toulouse, UMR 5219, Université de Toulouse, CNRS, UPS, 118 route de Narbonne, Toulouse, F-31062 Cedex 9, France.

*Corresponding author(s). E-mail(s): eliot.tron@enac.fr;
Contributing authors: nicolas.couellan@recherche.enac.fr;
stephane.puechmorel@enac.fr;

Abstract

Deep learning models are known to be vulnerable to adversarial attacks. Adversarial learning is therefore becoming a crucial task. We propose a new vision on neural network robustness using Riemannian geometry and foliation theory. The idea is illustrated by creating a new adversarial attack that takes into account the curvature of the data space. This new adversarial attack, called the *two-step spectral attack*, is a piece-wise linear approximation of a geodesic in the data space. The data space is treated as a (pseudo) Riemannian manifold equipped with the pullback of the Fisher Information Metric (FIM) of the neural network. In most cases, this metric is only semi-definite and its kernel becomes a central object to study. A canonical foliation is derived from this kernel. The curvature of transverse leaves gives the appropriate correction to get a two-step approximation of the geodesic and hence a new efficient adversarial attack. The method is first illustrated on a 2D toy example in order to visualize the neural network foliation and the corresponding attacks. Next, we report numerical results on the `MNIST` and `CIFAR10` datasets with the proposed technique and state of the art attacks presented in [1] (OSSA) and [2] (AutoAttack). The results show that the proposed attack is more efficient at all levels of available budget for the attack (norm of the attack), confirming that the curvature of the transverse neural network FIM foliation plays an important role in the robustness of neural networks. The main objective and interest of this study is to provide a mathematical understanding of the geometrical issues at play in the data space when constructing efficient attacks on neural networks.

Keywords: Neural Networks, Robustness, Fisher Information Metric, Information geometry, Adversarial attacks.

1 Introduction

Lately there has been a growing interest in the analysis of neural network robustness and the sensitivity of such models to input perturbations ([3–6]). Most of these investigations have highlighted their weakness to handle adversarial attacks ([7]) and have proposed some means to increase their robustness. Adversarial attacks are real threats that could slow down or eventually stop the development of neural network models or their applications in contexts where robustness guarantees are needed. For example, in the specific case of aviation safety, immunization of critical systems to adversarial attacks should not only be guaranteed but also certified. Therefore, addressing the robustness of future on-board or air traffic control automated systems based on such models is a main concern.

Adversarial attacks are designed to fool classification models by introducing perturbations in the input data. These perturbations remain small and in the case of images for example, may be undetectable to the human eye. So far, most of the research effort has focused on designing such attacks in order to augment the training dataset with the constructed adversarial samples and expecting that training will be more robust. Among these methods, one can refer to the Fast Gradient Sign methods ([8]), robust optimization methods ([9]), DeepFool ([10]), and others ([11]). There are major drawbacks with these approaches. Most attacks are data dependent and provide no guarantees that all relevant attacks have been considered and added to the training set. Furthermore, training in such manner does not acquire a global knowledge about the weakness of the model towards adversarial threats. It will only gain robustness for attacks that have been added to the dataset. However, crafting adversarial attacks by exploiting the properties of neural network learning is useful to understand the principles at play in the robustness or sensitivity of neural architectures.

Many authors consider neural network attacks and robustness properties in a Euclidean input space. Yet, it is commonly admitted that to learn from high dimensional data, data must lie in a low dimensional manifold ([12]). Such manifold has in general non-zero curvature and Riemannian geometry should therefore be a more appropriate setting to analyze distances and sensitivity from an attack point of view. Furthermore, to analyze neural network model separation capabilities and its robustness, it is critical to understand not only the topology of the decision boundaries in the input space but also the topology of iso-information regions induced by the neural network. Again, there is no reason to believe that these sub-manifolds have zero curvature in general. The Fisher information metric (FIM) is a valid metric for such purpose. Indeed, the network output is seen as a discrete probability that lies on a statistical manifold. The FIM may then be used as a Riemannian metric at the output and the pullback metric of the Fisher information as a metric for the input manifold ([1]). The importance of the FIM in the context of deep neural networks has already

been pointed out by several authors. In [13], it is shown that the FIM defines the landscape of the parameter space and the maximum FIM eigenvalue defines an approximation of the appropriate learning rate for gradient methods. The FIM with respect to data (also called local data matrix) instead of network parameters has also been investigated from a geometric perspective in [14]. The authors have shown that training data are organized on sub-manifolds (leaves) of a foliation of the data domain. A few authors have also tried to exploit this geometric knowledge to construct adversarial attacks or get some form of immunization from them. In [1], the direction of eigenvector corresponding to the maximum eigenvalue of the pullback FIM metric is used as a direction of attack where as in [15], similar developments are proposed to robustify the model by regularizing the neural network loss function by the trace of the FIM. In [16, 17], the authors directly use the geodesic distance associated with the FIM as a regularisation term during the training of the neural network. They show that exploiting the geometry of the FIM makes the network more robust to adversarial attacks. In [17], they also suggest that some notion of curvature of the statistical manifold is linked to the robustness of the network to adversarial attacks, as we will investigate in this article by looking at the foliated input space. More specifically, in [1], the authors have shown that the one step spectral attack (OSSA) they proposed is more efficient than the Fast Gradient Sign and the One Step Target Class methods [18]. This shows that the choice of the FIM metric as a measure of sensitivity is relevant and supports our decision to also use it in our study. However, unlike this previous work, we will investigate its geometrical properties directly in the data space where attacks are constructed. Note that other kinds of geometry have also been investigated, such as Wasserstein geometry in [19] where the authors show that Wasserstein barycenters make good candidates for robust data augmentation.

In this work, we build on the work of [1] and exploit further the geometrical properties of the foliation of the pullback metric of the neural network FIM. More specifically, we show that the curvature of the leaf of the transverse foliation can be utilized to construct a two-step attack procedure referred as *two-step* attack. Given a budget of attack, meaning the Euclidean norm of the attack vector, we first move in the direction of the eigenvector corresponding to the maximum eigenvalue of the FIM as proposed in [1] and then make another move that takes into account the curvature. The two steps could be seen as a discretized move along a geodesic curve. The interest of this procedure is not only to prove that, for a given budget of attack, it is possible to construct worse attacks than those proposed in [1] but also and more importantly to emphasize the role of curvature in the sensitivity of neural network models. Mathematically, this translates into the expression of the quadratic form approaching the Kullback-Leibler divergence between the network output probability distribution at the origin of the attack and the probability distribution at the point reached by the attack. Indeed, we show that this expression makes explicit use of the Riemannian curvature tensor of the foliation in the input space. To better grasp and visualize the effect of curvature on the attacks, we provide details on the calculation of the neural network FIM and the corresponding attacks on a `XOR` toy problem. The small dimension of the problem allows also 2D illustrations of the phenomenon. To demonstrate

experimentally the benefit of exploiting the foliation curvature, experiments are conducted first with the `XOR` toy problem and next with the `MNIST` ([20]) and `CIFAR10` ([21]) dataset. A measure of attack efficiency is defined using the fooling rate that computes the percentage of predictions that are changing class on random data points. The fooling rate is then used to compare the *two-step* attack to OSSA, and the state of the art technique AutoAttack [2]. The result show a substantial increase for the fooling rates when curvature is utilized.
The contribution of this study is two-fold: First and mainly, it proposes a Riemannian geometry framework for analyzing the robustness of neural networks. Gaining mathematical understanding on the efficiency of attacks should give future perspectives on strong defense mechanisms for neural networks. Next, since we use the construction of an attack as a methodology to understand the role of the curvature of the transverse leaves in the efficiency of an attack, we also provide as a by-product a two-step procedure for crafting adversarial attacks. These attacks have proven to be more effective than state of the art attacks on small neural architectures. However, there are limitations in the generalization of our construction technique on larger instances as it involves the computation of the whole Jacobian of the neural network. This computation may turn out to be untractable for very large networks.
The paper is organized as follows. Section 2 details the general mathematical framework of this study and defines precisely the adversarial attacks that are considered. In the first part of Section 3, the construction of the local attack by [1] is recalled. Next, in Section 4, the main developments of this research are detailed. They consist in extending the local attack using the geometry of the problem and constructing the so-called two-step attack. Section 5 is dedicated to the illustration of the method through a simple play-test problem. It details more precisely the required calculations on a simple low dimensional problem and provides illustrations of the resulting foliation at the heart of the technique. Section 6 provides the numerical results and Section 7 concludes the article. Appendices A and B recall some important concepts of Riemannian geometry that are used throughout the article. Appendix C contains the proof of some stated theoretical result.

2 Problem statement

2.1 Setup

In this paper, we are studying the behavior of a neural network $N : \mathcal{X} \to \mathcal{M}$. Its output can be considered as a parameterized probability density function $p_\theta(y \mid x) \in \mathcal{M}$ where $\mathcal{M}$ is a manifold of such probability density functions, $x \in \mathcal{X}$ is the input, $y \in \mathcal{Y}$ is the targeted label and $\theta \in \Theta$ is the parameter of the model (for instance the weights and biases in a perceptron). The geometric study of such probability distributions is part of *Information Geometry* [22].

In this case, $\mathcal{M}$ and $\mathcal{X}$ are two (pseudo) Riemannian Manifolds when equipped with the *Fisher Information Metric* (FIM). Fisher Information is originally a way to measure the variance of a distribution along a parameter. It was used as a Riemannian metric by Jeffrey and Rao (1945-1946), then by Amari, which gave birth to *Information Geometry* [22, 23]. Measuring how the distribution of the predicted labels changes

with input perturbations falls exactly in our usecase. This explains why the FIM is a good candidate to be $\mathcal{X}$ and $\mathcal{M}$'s metric.

Definition 1 (Fisher Information Metric)**.** The Fisher Information Metric (FIM) on the manifold $\mathcal{X}$ at the point x is defined by the following positive semi-definite symmetric matrix:

$$g_{ij}(x) = \mathbb{E}_{y|x,\theta}\left[\partial_{x_i} \ln p(y \mid x, \theta) \partial_{x_j} \ln p(y \mid x, \theta)\right].$$

Remark 1. Note that this definition is not the usual definition of the Fisher Information. Indeed, the one defined by Fisher, used in the work of Amari and many others, differentiate with respect to the parameter θ where we differentiate with respect to the input x. The authors in [14] call this new metric the *local data matrix* to avoid confusion.

It is important to notice that for the probability distribution given by a neural network, the Fisher Information is only a pseudo-Riemmannian metric, meaning that the matrix $G = (g_{ij})_{i,j}$ is not full rank. This is mainly due to the fact that $\dim \mathcal{X} \gg \dim \mathcal{M}$ in most cases of neural networks classifiers.

In this pseudo-Riemmannian geometric setting, one can define the kernel of the metric at each point. This subset of the tangent space $T\mathcal{X}$ is called a distribution and it is integrable under some assumptions, meaning that it gives rise to a *foliation* i.e. a partition of $\mathcal{X}$ into submanifolds called *leaves*. With this canonical foliation, a dual one can be defined using the orthogonal[1] complement of the kernel at each point. It is called the *transverse* foliation. At each point, moving along the FIM kernel leaf will not modify the output of the neural network. The properties of the transverse foliation will thus be of great interest in the setting of adversarial attacks. The reader may find more details on these geometric objects in the appendix A and B.

2.2 Adversarial attacks

One primary goal of this article is to craft the best *adversarial attack* possible, or at least approach it. An adversarial attack aims at disturbing the output of the neural network by adding noise to the original input, with the intention of changing the predicted class and thus fooling the network. The FIM with respect to the input is a good measure of dissimilarity between outputs, given a displacement of the input. In order to change the predicted class after the attack one would want to maximize the dissimilarity between the predicted distribution of probability before and after the attack. In other words, one would like to maximize the FIM's geodesic distance[2] $d(x_o, x_a)$ between the input point x_o and the point x_a after the attack.

Proposition 2.1. *The geodesic distance can be expressed with the Riemannian norm and the logarithm map*[3]*:*

$$d(x_o, x_a)^2 = \left\|\log_{x_o} x_a\right\|_{\mathcal{X}}^2 = g_{x_o}\left(\log_{x_o} x_a, \log_{x_o} x_a\right).$$

[1] Orthogonal with respect to the ambiant Euclidean metric.
[2] see Definition 8 in A
[3] see Definition 11 in A

Otherwise said, if $v = \log_{x_o} x_a$ is the initial velocity of the geodesic between the two points,

$$d(x_o, \exp_{x_o}(v))^2 = \|v\|_{\mathcal{X}}^2 .$$

The optimal solution will thus be the $\hat{x}_a$ maximizing this quantity, with some constraints. Indeed, one of the characteristics of a good adversarial attack is to be as undetectable as possible by usual measurements on the input. A usual measure for that is the Euclidean distance between x_o and x_a. The attacked point will thus be constrained to the Euclidean ball of given (small) radius and centered at x_o. The optimal attacked point is the result of a geodesic move from x_o with initial velocity $v \in T_{x_o}\mathcal{X}$.

Definition 2 (ε-Adversarial Attack Problem)**.** The optimal geodesic attack with a Euclidean budget of $\varepsilon > 0$ and initial velocity v verifies:

$$\max_{v \in T_{x_o}\mathcal{X}} \|v\|_{\mathcal{X}}^2 \text{ subject to } \left\|\exp_{x_o}(v) - x_o\right\|_2^2 \leq \varepsilon^2 \qquad (\varepsilon\text{-AAP})$$

Remark 2. Some other authors [24] are looking for the smallest attack in Euclidean norm that changes the predicted class. We are taking the problem in reverse and looking for an imperceptible attack with respect to a sensor with an activation level of ε (e.g. the human eye, the human ear, camera or some other devices) with no guarantees that there exists a class changing attack. However, in real life context where dimension is large, it has been shown that such class changing attacks exist for small ε with high probability [25].

A usual optimization algorithm could solve this problem, but the amount of calculus would be tremendous since the geodesic requires to solve an ODE with boundary conditions each time. This greedy solution is thus not feasible in practice. To get close to the optimal solution, the geodesic can be approximated by Euclidean steps.

3 A local method

The authors in [1] have proposed a method to approximate the solution of a near-(ε-AAP) problem. Their formulation of the Adversarial Attack Problem is not exactly the same since the criteria they are maximizing is not the geodesic distance, but the Kullback-Leibler divergence. The matter with (KL) divergences is that they are not distances, because often not symmetric. Nevertheless, if G is the matrix associated to the FIM, a second order Taylor approximation gives us that:

$$\begin{aligned} D_{KL}\left(p_\theta\left(y \mid x\right) \| \, p_\theta\left(y \mid x+v\right)\right) &= \frac{1}{2} v^T G_x v + o\left(\|v\|^2\right) \\ &= \frac{1}{2} g_x\left(v, v\right) + o\left(\|v\|^2\right) = \frac{1}{2}\|v\|_{\mathcal{X}}^2 + o\left(\|v\|^2\right). \end{aligned}$$

Then, the Euclidean constraint is taken directly on v meaning that x_o is moved by a Euclidean step in the direction of v. The problem thus solved in [1] is a linear approximation of (ε-AAP). Doing so amounts to approaching the geodesic by its initial velocity. In other words, the first order approximation $\exp_{x_o}(v) \approx x_o + v$ is used for this section.

We reformulate and prove below the main result of [1] in the context of this study.

Proposition 3.1. *The vector $\hat{v}$ maximizing the quadratic form $v \mapsto v^T G_x v$ is an eigenvector of the FIM G_x corresponding to the largest eigenvalue. Its re-normalization by $\frac{\varepsilon^2}{\|\hat{v}\|_2^2}$ gives an approximated solution to (ε-AAP). This method is illustrated by Figure 1.*

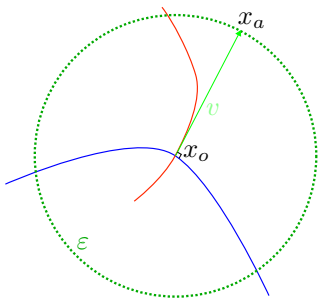


Fig. 1: The one-step spectral attack in action. The circle represents the Euclidean budget. The blue curve represents the leaf of the kernel foliation and the red curve represents the transverse foliation.

The proof of Proposition 3.1 can be found in Appendix C.

Remark 3. The KKT optimality conditions lead to two valid and yet very different solutions: $\hat{v}$ and $-\hat{v}$. To choose between the two, we select the solution decreasing the probability of the original class the most, meaning that we select the attack vector $v \in \{\hat{v}, -\hat{v}\}$ satisfying:

$$p_\theta\left(y_{x_o} \mid x_o - v\right) > p_\theta\left(y_{x_o} \mid x_o + v\right). \tag{1}$$

With this choice, we increase the probability of fooling the network on this point.

This first method is local and does not take into account the curvature of the data. Hence, we propose a new method to improve the performances, especially in regions of $\mathcal{X}$ where this curvature is high.

4 A two-step attack

The idea here is to take a first local step v as in Section 3, and then take a second step to refine the linear approximation. The two-step nature of this attack allows the second step to take into account the curvature of the area around the input point. This method is able to exploit the geometry of the problem to construct a better approximation of the original problem (ε-AAP). This attack will be named the Two Step Spectral Attack (TSSA) .

Let v be the approximated solution of (ε-AAP) using the method of Section 3 but with a budget of $\mu^2 < \varepsilon^2$.

The new problem to solve is the following:

$$\max_{w} \|w\|_{\mathcal{X}}^2 \text{ subject to } \begin{cases} \|w+v\|_2^2 \leq \varepsilon^2 \\ \|v\|_2^2 = \mu^2 < \varepsilon^2 \\ v \text{ eigenvector of } G_x \end{cases} \tag{S2P}$$

In practice, one could solve this problem in the same way as in Section 3 by taking w the eigenvector of G_{x+v} with the largest eigenvalue once again (see Figure 2). By doing that, we linearly approach the geodesic between x_o and x_a by two line segments in the Euclidean sense. Taking more steps would approximate the geodesic better. Using only two steps as proposed is a simple procedure from a computational point of view and still achieves significantly better results than taking just one step as we will see in Section 5.

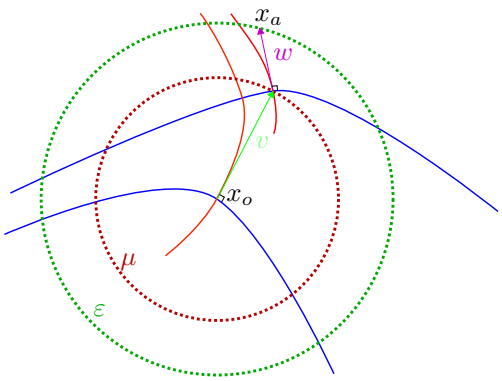


Fig. 2: The two-step attack in action. The two circles represent the Euclidean budget. The blue curves represent the leaves of the kernel foliation and the red curves represent the transverse foliation.

To explicit the action of the curvature at x of the input space on the trajectory of this multi-step attack, we will approximate G_{x+v} by its value at x using normal coordinates[4].

Remark 4. In the sequel of the article, for any vector (or tensor) x, $\bar{x}$ will denote x expressed in terms of normal coordinates. Besides, we are going to use the Einstein summation notation omitting the symbol $\sum$ whenever the index over which the sum should apply repeats.

Proposition 4.1. *If $\overline{x}$ are the normal coordinates at x_o and if R is the Riemannian curvature tensor, then*

$$w^T G_{x+v} w = \frac{\partial x^m}{\partial \overline{x}^i} \frac{\partial x^n}{\partial \overline{x}^j} \left(\delta_{mn} + \frac{1}{3} \overline{R}_{mkln}(x) \overline{v}^k \overline{v}^l \right) w^i w^j + o\left(\|v\|^2 \right) \tag{2}$$

[4] see Definition 10 in A

Proof.

$$\begin{aligned} w^T G_{x+v} w &= \left(G_{x+v}\right)_{ij} w^i w^j \\ &= \frac{\partial x^m}{\partial \overline{x}^i}\frac{\partial x^n}{\partial \overline{x}^j}\left(\overline{G}_{x+v}\right)_{nm} w^i w^j \\ &= \frac{\partial x^m}{\partial \overline{x}^i}\frac{\partial x^n}{\partial \overline{x}^j}\left(\delta_{mn} + \frac{1}{3}\overline{R}_{mkln}(x)\overline{v}^k\overline{v}^l + o\left(\|v\|^2\right)\right) w^i w^j. \end{aligned}$$

The last line is obtained by the second-order Taylor expansion of g in normal coordinates centered at x. The reader can find details of the computations in [26], Section 3.5 Corollary 7. □

Proposition 4.2. *By denoting* $\overline{w} = \left(\frac{\partial x^m}{\partial \overline{x}^i} w^i\right)_m = Pw$ *and* $R_v = \overline{R}_{mkln}(x)\overline{v}^k\overline{v}^l$, *Equation 2 can be rewritten with matrix notation by the following:*

$$\|w\|_{\mathcal{X}}^2 = \|\overline{w}\|_2^2 + \frac{1}{3}\overline{w}^T R_v \overline{w} + o\left(\|v\|^2\right). \tag{3}$$

Remark 5. In what follows, $\|w\|_{\mathcal{X}}^2$ will always be taken at $x+v$, and be approximated by the right hand of Equation 3. Additionally, we compare w and v without taking into account parallel transport since the Christoffel symbols vanish around the origin in normal coordinates.

The transition matrix P is equal to $\left(\frac{\partial x^i}{\partial \overline{x}^j}\right)_{i,j}$ and its inverse to $\left(\frac{\partial \overline{x}^i}{\partial x^j}\right)_{i,j}$. We should have for instance:

$$G_x = P^T \overline{G}_x P = P^T I_n P \text{ and } {P^{-1}}^T G_x P^{-1} = I_n$$

The matrix $P^{-1} = \left[\frac{v_1}{\sqrt{\lambda_1}} \cdots \frac{v_n}{\sqrt{\lambda_n}}\right]$ with v_i the eigenvector of G_x associated with the eigenvalue λ_i satisfies this equation (the family is chosen to be orthonormal for the ambient Euclidean metric: $v_i^T v_j = \delta_{i,j}$). Note that this gives us

$$P = \begin{bmatrix} \sqrt{\lambda_1} v_1^T \\ \vdots \\ \sqrt{\lambda_n} v_n^T \end{bmatrix}.$$

Remark 6. The pullback metric g_x is always degenerate in this problem. Indeed, the dimension of its image is strictly bounded by the number of classes of the given task[5].

To take this into account, if $d = \dim \operatorname{Im} G_x$, one can rewrite the metric in normal coordinates by:

$$\overline{G}_x = \begin{bmatrix} \mathbf{I}_d & \mathbf{0}_{d,n-d} \\ \mathbf{0}_{n-d,d} & \mathbf{0}_{n-d,n-d} \end{bmatrix}.$$

The transition matrix P^{-1} is equal to $\left[\frac{v_1}{\sqrt{\lambda_1}} \cdots \frac{v_d}{\sqrt{\lambda_d}}\ v_{d+1} \cdots v_n\right]$.

[5] See [14] for the proof.

Proposition 4.3. *If w is a solution to S2P, then there exists a scalar $\lambda \geq 0$ such that*

$$\left(P^T BP - \lambda I_n\right) w = \lambda v \tag{4}$$

with $B = I_n + \frac{1}{3}R_v$.

Proof. Using again the KKT conditions but with the constraint $\|v+w\|_2^2 = \left\|v + P^{-1}\overline{w}\right\|_2^2 \leq \varepsilon^2$, it implies that there exists a scalar $\lambda \geq 0$ such that:

$$\begin{aligned} &\nabla_{\overline{w}} \left(\|\overline{w}\|_2^2 + \frac{1}{3}\overline{w}^T R_v \overline{w} \right) \\ &- \lambda \nabla_{\overline{w}} \left(\left\|v + P^{-1}\overline{w}\right\|_2^2 - \varepsilon^2 \right) = 0 \\ \Longrightarrow\ &\overline{w} + \frac{1}{3} R_v \overline{w} - \lambda \left({P^{-1}}^T \left(P^{-1}\overline{w} + v \right) \right) = 0. \end{aligned}$$

Thus

$$\begin{aligned} & P^T \left(I_n + \frac{1}{3} R_v \right) Pw = \lambda \left(w + v \right) \\ \Longleftrightarrow\ & P^T \left(B - \lambda {P^{-1}}^T P^{-1} \right) Pw = \lambda v \\ \Longleftrightarrow\ & \left(P^T BP - \lambda I_n \right) w = \lambda v \end{aligned}$$

where $B = I_n + \frac{1}{3}R_v$. □

Corollary 4.4. *In normal coordinates, Equation 4 rewrites as:*

$$PP^T B\overline{w} = \lambda\left(\overline{w} + \overline{v}\right) \ \ ie\ \ \left(PP^T B - \lambda I_n\right)\overline{w} = \lambda \overline{v}.$$

Remark 7. Note that $PP^T = \mathrm{diag}\left(\lambda_i\right)$.

Corollary 4.5. *Whenever λ is not an eigenvalue of $P^T BP$, $\left(P^T BP - \lambda I_n\right)$ is non-singular and*

$$w = \lambda \left(P^T BP - \lambda I_n\right)^{-1} v. \tag{5}$$

It remains to find a λ such that the constraint $\|w+v\|_2^2 \leq \varepsilon^2$ is satisfied. There are two cases:

1. either $\|w+v\|_2^2 = \varepsilon^2$ and $\lambda > 0$,
2. or $\|w+v\|_2^2 < \varepsilon^2$ and $\lambda = 0$.

4.1 The case $\lambda > 0$

We will use the constraint $\|w+v\|_2^2 = \varepsilon^2$ to get λ. We suppose in what follows that 2λ is not an eigenvalue of B, ie $(2\lambda - 1)$ is not an eigenvalue of $\frac{1}{3}R_v$. The constraint

may be written as:

$$\begin{aligned}\|w+v\|_2^2 &= \left\|\lambda\left(P^TBP-\lambda I_n\right)^{-1}v+v\right\|_2^2\\ &= \left\|\left(\lambda\left(P^TBP-\lambda I_n\right)^{-1}+I_n\right)v\right\|_2^2.\end{aligned}$$

Lemma 4.6.

$$\lambda\left(P^TBP-\lambda I_n\right)^{-1}+I_n=\left(P^TBP-\lambda I_n\right)^{-1}P^TBP.$$

Proof.

$$\begin{aligned}&\left(P^TBP-\lambda I_n\right)\left(\lambda\left(P^TBP-\lambda I_n\right)^{-1}+I_n\right)\\ =&\lambda\left(P^TBP-\lambda I_n\right)\left(P^TBP-\lambda I_n\right)^{-1}+\left(P^TBP-\lambda I_n\right)\\ =&\cancel{\lambda I_n}+P^TBP-\cancel{\lambda I_n}=P^TBP.\end{aligned}$$

□

Thus,

$$\varepsilon^2=\|w+v\|_2^2=\left\|P^TBP\left(P^TBP-\lambda I_n\right)^{-1}v\right\|_2^2. \tag{6}$$

To find λ that satisfies Equation 6, we will study the vanishing points of:

$$\begin{aligned}\varphi:[0,\infty[&\longrightarrow [0,\infty[\\ \lambda &\longmapsto \left\|P^TBP\left(P^TBP-\lambda I_n\right)^{-1}v\right\|_2^2-\varepsilon^2\end{aligned}$$

However, finding the vanishing points of such a function is not an easy task. Several methods may be used. A numerical method such as the Newton's method [27] could be applied.

Alternatively, observe that problem S2P can be simplified by using the triangular inequality to get back to an easier eigenvalue problem. Indeed, consider the following problem:

$$\max_w \|w\|_{\mathcal{X}}^2 \text{ subject to } \begin{cases}\|w\|_2\leq\varepsilon-\mu\\ \|v\|_2^2=\mu^2<\varepsilon^2\\ v \text{ eigenvector of } G_x\end{cases} \tag{S2.2P}$$

This new problem is illustrated on Figure 3. The green circle is the true budget and the two other smaller circles represent the triangular inequality approximation. One can see that the second step w does not reach the green circle but stop before due to the approximation.

Proposition 4.7. *Any solution of S2.2P problem will satisfy the constraint of S2P problem.*

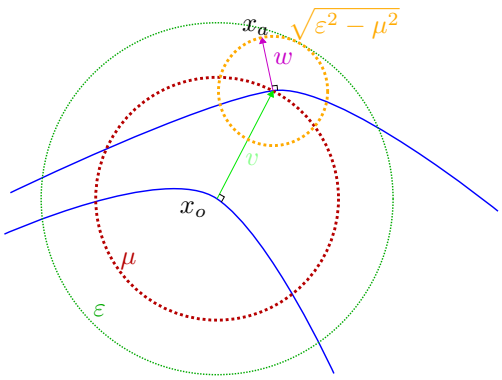


Fig. 3: The two-step attack in action with the triangular inequality simplification. The three circles represent the Euclidean budget. The blue curves represent the leaves of the kernel foliation.

Proof. Indeed, by the triangular inequality, $\|w + v\|_2 \leq \|w\|_2 + \|v\|_2 \leq (\varepsilon - \mu) + \mu = \varepsilon$. □

With this new constraint, KKT's conditions boil down to

$$P^T B P w = \lambda w$$

or in normal coordinates:

$$P P^T B \overline{w} = \lambda \overline{w}.$$

Proposition 4.8. *A solution to S2.2P is to choose $\overline{w}$ to be the eigenvector of PP^TB with the highest eigenvalue λ and with the appropriate Euclidean norm of $\varepsilon - \mu$.*

An algorithm that finds a solution to S2.2P can then be created using Proposition 4.8. It is presented in Algorithm 1. It is then implemented[6] and tested in Section 6.

Complexity analysis of TSSA

Let d be the dimension of the input space of the neural network, and C the number of classes of the classification problem. The runtime analysis of Algorithm 1 can be understood as follow:

- Line 1:

1. Computing G_{x_o}: It requires the computation of the the network's Jacobian matrix with respect to the input, as well as inverting the output of the neural network. Computing such Jacobian matrix requires $O(C)$ calls to the Autograd procedure, PyTorch automatic differentiation package [28], where C is the dimension of the output. The complexity of the Autograd algorithm depends on the neural network architecture, and scales with the number of weights, but also with the dimension of the input. After that, 2 matrix multiplications are needed to get $G_{x_o} = J_N(x)^T \mathrm{diag}\,(N(x))^{-1} J_N(x)$.

[6] The implementation of TSSA is available at https://github.com/eliot-tron/CurvNetAttack.

Algorithm 1: Two Step Spectral Attack (TSSA)

Data: x_o initial point, $\varepsilon > \mu > 0$ euclidean budgets.
Result: x_a attacked point

1 $v \leftarrow$ highest eigenvector of G_{x_o}
2 $v \leftarrow \mu \times \frac{v}{\|v\|_2}$
3 $y_{x_o} \leftarrow \text{argmax}_y\, p_\theta\left(y \mid x_o\right)$
4 **if** $p_\theta\left(y_{x_o} \mid x_o + v\right) > p_\theta\left(y_{x_o} \mid x_o - v\right)$ **then**
5 $\quad v \leftarrow -v$
6 **end**
7 $w \leftarrow$ highest eigenvector of G_{x_o+v}
8 $w \leftarrow (\varepsilon - \mu) \times \frac{w}{\|w\|_2}$
9 **if** $v^T w < 0$ **then**
10 $\quad w \leftarrow -w$
11 **end**
12 $x_a \leftarrow x_o + v + w$
13 **return** x_a

2. Computing the highest eigenvector of this $d \times d$ matrix: It can be solved with a power method[7] in $O\left(d^2\right)$.

- Line 2: $O(d)$.
- Line 3: Evaluation time of the neural network, and then finding the maximum in $O(C)$.
- Line 4: Two evaluations of the neural network.
- Line 5: $O(d)$
- Line 7: Same complexity as Line 1.
- Lines 8-12: $O(d)$.

In practice, Line 1 and 7 are the most expensive ones. This algorithm requires, in total, $O(C)$ calls to Autograd with respect to the input of the network for each given input point x_o, which limits the tests on large datasets requiring large networks. Possible runtime improvements can be developed in future work such as the use of faster power iteration methods (e.g. Lanczos), or the use of the alias method when C is high. See [1] for some details on these aspects. Some other improvements can be developed on the implementation side in PyTorch with batched gradient computation. We leave this for future work and focus here on the effect of curvature on adversarial attacks.

4.2 The case $\lambda = 0$

In that case, w is in the interior of the boundary of the problem. The problem reduces to

$$B\overline{w} = 0 \iff R_v\overline{w} = -3\overline{w}.$$

[7] See Section 7.3, p. 365 of [29].

This means that w is the optimal second step if $\overline{w} \in \operatorname{Ker} B$, ie if w is eigenvector of R_v with eigenvalue -3. However, this case does not produce any interesting KKT admissible point. Indeed,

$$\|w\|_{\mathcal{X}}^2 = \|\overline{w}\|_2^2 + \frac{1}{3}\overline{w}^T R_v \overline{w} = \|\overline{w}\|_2^2 - \overline{w}^T\overline{w} = 0.$$

Therefore, the case $\lambda = 0$ does not lead to useful adversarial attacks when the previous approximations are applied to the problem. The study of this singularity is left as future work.

5 An enlightening play-test example

5.1 Setup

In this section, we focus on a low dimensional example in order to visualize more easily the effect of curvature on the efficiency of the adversarial attack. A simple neural network N_θ with one hidden layer of k neurons and sigmoids as activation functions is used. Its architecture is depicted on Figure 4.

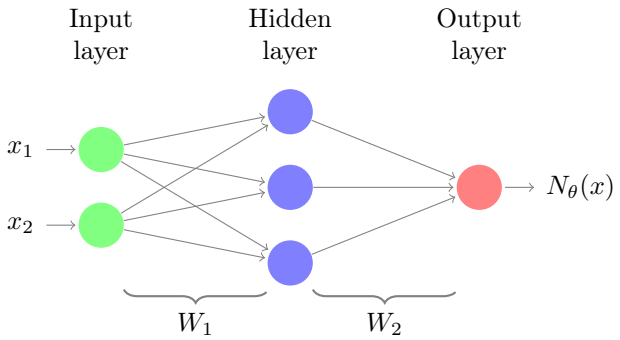


Fig. 4: XorNet N_θ with 3 hidden neurons.

To be more precise, if $\sigma : a \in \mathbb{R}^d \mapsto \left(\frac{1}{1+e^{-a_i}}\right)_{i=1}^d \in \mathbb{R}^d$ is the sigmoid function and if $x \in \mathbb{R}^2$, $W_1 \in \mathcal{M}_{2,k}(\mathbb{R})$, $W_2 \in \mathcal{M}_{k,1}(\mathbb{R})$, we have

$$N_\theta(x) = \sigma\left(W_2\sigma\left(W_1 x + b_1\right) + b_2\right). \tag{7}$$

This neural network is then trained to approximate the very simple function `Xor`: $\{0,1\}^2 \to \{0,1\}$ defined in Table 1.

The output $N_\theta(x)$ is seen as the parameter p of a Bernoulli law and associates to the network the following probability distribution: $p(y \mid x, \theta)$ where θ is the vector containing the weights and biases $(W_i,\ b_i)$, x is the input and y is the true label.

Proposition 5.1. *The random variable $Y \mid X, \theta$ follows Bernoulli's law of parameter $p = N_\theta(x)$.*

Table 1: `Xor` function.

`Xor`	0	1
0	0	1
1	1	0

5.2 Computing the output FIM

The output of the network is the manifold of Bernoulli probability densities parameterized by the open segment $]0,1[$.

Proposition 5.2. *Let $p \in]0,1[$ and G_p the Fisher Information Metric at the point p.*

$$G_p = \frac{1}{p}\frac{1}{1-p} \tag{8}$$

Proof.

$$\begin{aligned}
G_p &= -\mathbb{E}_{y|x,\theta}\left[\partial_p^2 \left(\ln P(y \mid x, \theta)\right)\right] \\
&= -\mathbb{E}\left[\partial_p^2 \left(y \ln p + (1-y)\ln(1-p)\right)\right] \\
&= -\mathbb{E}\left[\partial_p \left(\frac{y}{p} - \frac{(1-y)}{1-p}\right)\right] \\
&= -\mathbb{E}\left[-\frac{y}{p^2} - \frac{(1-y)}{(1-p)^2}\right] \\
&= \frac{1}{p} + \frac{1}{1-p} = \frac{1}{p}\frac{1}{1-p}.
\end{aligned}$$

□

5.3 Computing the pullback metric

Let $x \in \mathcal{M}$ be a point associated with p by the network.

Lemma 5.3. *The Fisher Information Metric G_x on $\mathcal{X}$ is the pullback metric of G_p by the neural network N_θ.*

Corollary 5.4. *If $J = \left[\frac{\partial p}{\partial x_j}\right]_{j=1,2} = \left[\frac{\partial N_\theta(x)}{\partial x_1} \;\; \frac{\partial N_\theta(x)}{\partial x_2}\right]$, then*

$$G_x = J^T G_p J. \tag{9}$$

Remark 8. The Jacobian matrix of a neural network J is not difficult to compute thanks to automatic differentiation available in most neural network training software's packages. Besides, this approach for computing J numerically allows the sequel of the article to stay quite general regarding the architecture of the neural network.

Knowing how to compute the metric at any point unlocks the computation of the local method seen in Section 3 and the two-step method seen in Section 4 when the FIM is recomputed at the intermediary point. Additionally, studying the curvature

of the input space is insightful to understand the behavior of the correction step in the two-step attack, and more generally to understand why adversarial attacks are, in some cases, so efficient.

Nonetheless, the pullback metric being almost always only semi-definite for machine learning tasks, it makes sense to consider its kernel. The curvature will have a decomposition term on the kernel of g and a decomposition term on its orthogonal[8]. To craft the attack, the orthogonal term will be the only curvature component of interest as stated at the end of Section 2.1. Thus in the following subsection, we compute the metric kernel for this neural network.

5.4 The metric kernel foliation

Definition 3. The kernel of a metric G at the point x is defined by

$$\ker_x G = \left\{ X \in T_p\mathcal{M} \mid X^T G_x Y = 0,\ \forall Y \in T_p\mathcal{M} \right\}.$$

This kernel defines an integrable distribution when Frœbenius' condition[9] is satisfied, and with this distribution emerges a Riemannian foliation on the input manifold, foliation defined by the action of the neural network.

Lemma 5.5. $\ker_x G = \ker J$.

Proof. We omit $\frac{1}{p}\frac{1}{1-p}$ during the proof because it is always non-zero.

- Let us prove first that $\ker_x G \subset \ker J$. If $X \in \ker_x G$, then $X^T J^T J X = 0$. Hence $(JX)^T (JX) = 0$, or written otherwise: $\|JX\| = 0$. Hence $JX = 0$ and $X \in \ker J$.
- Then we can prove that $\ker J \subset \ker_x G$. This inclusion is simply due to the fact that if $X \in \ker J$, we have for all $Y \in T_p\mathcal{M}$ that $X^T J^T J Y = \underbrace{(JX)^T}_{=0} JX = 0$.

□

Proposition 5.6. *If at least one of the two components of J is non-zero, the distribution at x is one dimensional and given by $P_x = \mathrm{Span}\left(J_2(x)\partial_1 - J_1(x)\partial_2\right)$.*

Remark 9. If $J = 0$, the leaf at x is singular and is of dimension 2.

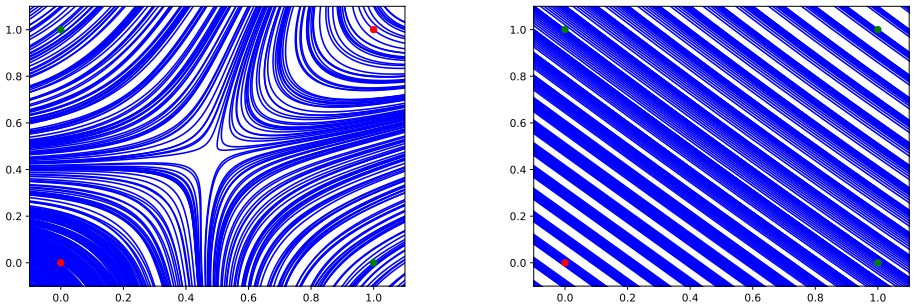
With the `Xor` function, the dimension of the leaves is 1, thus the condition of Frœbenius is trivially satisfied and P is integrable.

Proposition 5.7. *If $\gamma : t \in I \mapsto \gamma(t) \in \mathcal{M}$ is an integral curve for the distribution P, it satisfies the following ODE:*

$$\begin{aligned}\gamma'(t) &= J_2\left(\gamma(t)\right)\partial_1 - J_1\left(\gamma(t)\right)\partial_2 \\ &= J_{\gamma(t)} \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}.\end{aligned}$$

[8] Here, the Euclidean orthogonal is considered.
[9] For more details, see Chapter 1 of [30]

It can be solved numerically quite easily with a finite difference method[10]. Figure 5 provides illustrations of the computed neural network kernel foliation with such a method for the `Xor` function (Figure 5a), and for the `Or` function (Figure 5b).



(a) Task: `Xor` (b) Task: `or`

Fig. 5: Kernel foliation: the leaves are represented by the blue lines, the red dots are the 0 result and the green dots are the 1 results.

Remark 10. For readability, the transverse leaves are not represented but can be easily deduced from figure 5 as Euclidean orthogonal curves to the blue ones.

These results are quite interesting. First of all, we notice that a linearly separable problem such as the `or` function seems to have hyperplanes as leaves. In fact, it is easy to show that if the neural network is replaced by a linear form $x \mapsto \langle n, x \rangle + b$, then the leaves are hyperplanes defined by the normal vector n.

Second of all, one can see that there is a singular point close to the middle point $(0.5, 0.5)$ for the `Xor` task. Clearly, it is also easy to see that around that central point, the curvature of the leaves is the highest. Since the two-step adversarial attack makes use of the curvature, we can conjecture that it is in the central region that the second step will have the most impact. This phenomenon will be confirmed in Section 6.

5.5 Visualizing the curvature

In this section, we are computing the extrinsic curvature of a transverse leaf to see where the two-step attack will differ the most from the one-step attack. But first, we compute the Levi-Civita connection. In the following of the article, we will use $i, j, k, l \in \{1, \dots, \dim \mathcal{X}\}$ as indices and we will use the Einstein summation notation.

Definition 4 (Levi-Civita connection)**.** In coordinates e_i, if Γ_{ij}^k are Chrystoffel's symbols, then the Levi-Civita connection is defined by:

$$\mathring{\nabla}_{e_j} e_i = \Gamma_{ij}^k e_k \quad .$$

[10] See for example [31].

Definition 5 (Riemannian curvature)**.** The *Riemannian Curvature Tensor* R is defined such that for all three tensor fields X, Y, Z,

$$R(X,Y)Z = \nabla_X \nabla_Y Z - \nabla_Y \nabla_X Z - \nabla_{[X,Y]} Z.$$

In local coordinates, this gives:

$$\begin{aligned} R^l_{ijk} &= dx^l \left(R\left(\partial_i, \partial_j\right) \partial_k \right) \\ &= dx^l \left(\nabla_{\partial_i} \nabla_{\partial_j} \partial_k - \nabla_{\partial_j} \nabla_{\partial_i} \partial_k - \nabla_{[\partial_i, \partial_j]} \partial_k \right) \\ &= \frac{\partial \Gamma^l_{kj}}{\partial x^i} - \frac{\partial \Gamma^l_{ki}}{\partial x^j} + \Gamma^\alpha_{kj} \Gamma^l_{\alpha i} - \Gamma^\alpha_{ki} \Gamma^l_{\alpha j}. \end{aligned}$$

Remark 11. In normal coordinates, at the origin point, one has:

$$R_{ijkl} = \frac{1}{2} \left(\partial_i \partial_l g_{jk} + \partial_j \partial_k g_{il} - \partial_i \partial_k g_{jl} - \partial_j \partial_l g_{ik} \right).$$

Finding an explicit form for R is unreasonable, but since we are in the simpler case where the dimension of the transverse leaves is 1, a quick approximation can do the trick. In fact, leaves are embedded submanifolds of the Euclidean space and the extrinsic curvature, i.e. the second fundamental form, can be computed as the rate of rotation of the normal vector.

To approximate the second step w of the two-step attack, we look at the rotation speed of the unit normal to the kernel leaf $\vec{n}$ when moved by an infinitesimal step dx. This infinitesimal step is taken in the direction of the transverse leaf, and is Euclidean. Since the rotation rate is approximated by finite difference, one can expect the angle variation to be very small. To ensure numerical stability, the usual procedure based on its cosine computation using inner product is replaced by one using the cross product. The next lemma gives the expression of the sine of the angle variation between two close positions on curve transverse to the $\ker g$ foliation.

Lemma 5.8. *If $\vec{n}_y$ is the normal to the kernel leaf at $y \in \mathcal{X}$, and if $\cdot \times \cdot$ is the cross product, then the infinitesimal variation of angle is*

$$|d\theta| = \arcsin\left(\|\vec{n}_x \times \vec{n}_{x+dx}\|_2 \right).$$

Please note that in the small angles approximation, the sine can be replaced by the angle itself, thus recovering the usual infinitesimal rotation representation as a cross product (this is in fact a Lie algebra representation in the usual sense).

To approximate the effect of the curvature during the Euclidean step v, one has to compute the rotation matrix R of angle $d\theta$:

$$R = \begin{bmatrix} \cos(d\theta) & -\sin(d\theta) \\ \sin(d\theta) & \cos(d\theta) \end{bmatrix}$$

Proposition 5.9. *If v is the first step, the approximated second step is given by $w = Rv$ and then re-normalized to get $\|w\|^2 + \|v\|^2 = \varepsilon^2$.*

The signed curvature $d\theta$ for the `Xor` problem can then be seen on Figure 6. One can see that the curvature is the highest in the middle, around the point $(0.5, 0.5)$, and also on the diagonals.

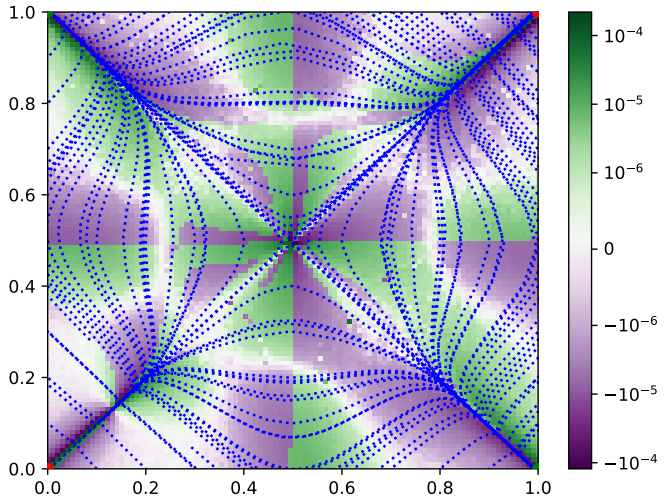


Fig. 6: Extrinsic (signed) curvature of the transverse leaves ($d\theta$) for the task `Xor`, computed with a dx of 10^{-6}, with the transverse leaves in blue.

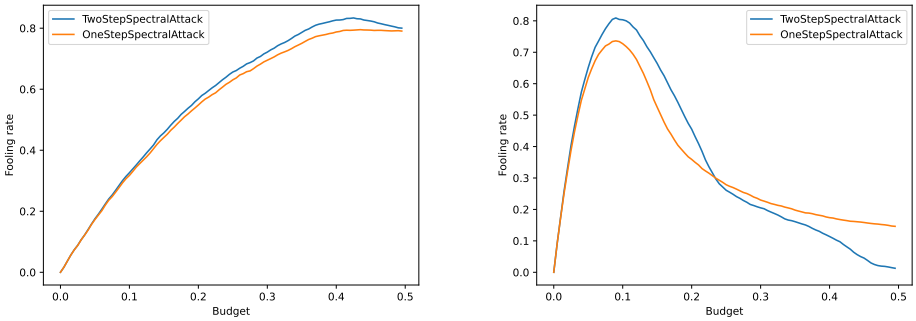
6 Numerical results

In the following subsections, we first provide an illustrative example, and next we report experimental results on two public datasets: MNIST [20] and CIFAR10 [21]. All codes used to produce the following results can be found in [32].

Note that the first step in the Two Step Spectral Attack is set to have a Euclidean budget μ of 60% of the total budget ε in all following experiments.

6.1 Xor dataset

To compare the different attacks, we train a neural network with 8 hidden neurons on random points taken in the square $[0, 1]^2$ until convergence. We then compute the two different attacks which are the One Step Spectral Attack presented in Section 3 and the Two Step Spectral Attack presented in Section 4 with Algorithm 1 where we compute the FIM at x_o and $x_o + v$. These attacks are computed on 5000 random points selected in a square I of variable length. The fooling rate is then computed as the quotient of the number of fooled prediction by the total number of points. The budget is selected between 0 and 0.5. We plot the results on Figure 7.



(a) $I = [0, 1]^2$ (b) $I = [0.4, 0.6]^2$

Fig. 7: Fooling rates with respect to the Euclidean budget with random points taken in I for the task `Xor`.

On each figure, the two-step attack beats the one step attack (for reasonable budgets). The two-step attack is especially strong on the area where the curvature is the strongest: almost one point better for the TSSA at the peak in high curvature zones (see Figure 7b) compared to only half a point for the full space (see Figure 7a). Indeed, as seen on Figure 6, the curvature is really strong at the center of the square $[0, 1]^2$. The eigenvector of the FIM associated with the greatest eigenvalue is always orthogonal to the leaf kernel. Therefore, striking close to the middle region without taking into account the curvature usually results in not changing the output label: $(0, 0) \leftrightarrow (1, 1)$ or $(1, 0) \leftrightarrow (0, 1)$. This is why the TSSA gets better results by taking the curvature into account.

Remark 12. Note that on Figure 7b, the fooling rates collapse shortly after budget $= 0.1$ because such high budgets makes every points leave the initial square I. These high budgets are too big for this area which is very close to the decision boundaries and the second step is not enough to compensate these instabilities, thus leading to such strange results at first glance.

The illustration of the two different attacks can be found on Figure 8.

These results confirm our intuition that using the information of local curvature to craft an adversarial attack is meaningful. It also highlights the role of the leaves of the kernel foliation in the sensitivity of neural networks to attacks.

6.2 MNIST dataset

Next, we train a medium convolutional neural network with 4 layers (2 convolutional and 2 linear), `ReLU` as activation function and a `Softmax` function at the output. The fooling rates for the One Step Spectral Attack and the Two Step Spectral Attack are represented on Figure 9 for different Euclidean budget ranging from 0 to 10. The TSSA performs better than the OSSA, proving that curvature of the network is of interest in the construction of attacks or defenses to these attacks. The difference between the fooling rate of the TSSA and the fooling rate of the OSSA is plotted on Figure 10. It

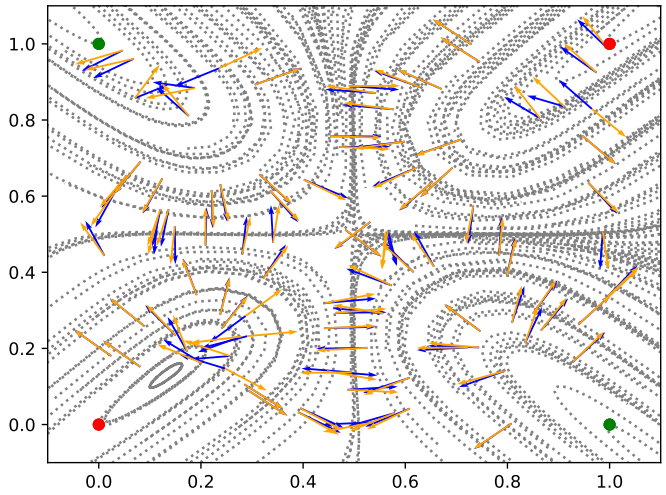


Fig. 8: TSSA (in blue) compared to OSSA (in orange) with $\varepsilon = 0.1$. The kernel foliation is depicted with the grey lines.

represents the advantage given by the curvature. As one can see, there are 3 different regimes:

1. The first one in blue is the region where the budget is small enough for the approximation $\exp_{x_o}(v) \approx x_o + v$ to hold. Taking into account the curvature does not modify enough the attack to fool the network on a lot of additional images compared to the straight line attack (OSSA).
2. The second regime in green is the region where the budget is high enough for the correction due to the curvature to be essential, and yet small enough for this same correction to hold. That is why the TSSA is much better than the OSSA for these budgets.
3. The third and last regime in red corresponds to budgets to big so that the approximations no longer make sense. At these horizons, the manifold is too non-linear.

To evaluate our technique against State Of The Art (SOTA) attacks, we consider the AutoAttack (AA) proposed in [2]. The authors propose a method to create adversarial attacks combining and improving 4 SOTA algorithms, making the AutoAttack algorithm an appropriate candidate to compare performances to. Indeed, it has been designed to benchmark adversarial robustness, enabling the community to have a unified way to compare the performances of their adversarial defenses with others on SOTA adversarial attacks. To implement the method, we used the code available at https://github.com/fra31/auto-attack.git. The settings used for the AutoAttack are the L_2 norm, a varying budget of ε, and the *standard* version of the attack.

The difference between the fooling rate of the TSSA and the fooling rate of the AutoAttack is plotted with respect to the Euclidean budget on Figure 11 for both `ReLU` and `Sigmoid` activation function cases. From the plots, one can see that the AutoAttack achieves better results than our Two Step attack for mid-range budgets. However, by design, the TSSA algorithm only performs two steps when the AutoAttack procedure carries out many moves (for example, 100 iterations for the APGD or FABAttack subroutines used within AutoAttack). To have a fair comparison with TSSA, it would require a 100 steps attack or equivalently solve the geodesic equation with 100 discretization steps. This is computationally expensive at each step due to the calculation of the FIM and its eigenvectors. Nevertheless, the TSSA achieves comparable performances for large budgets, and even better performances for small budgets. Note that the range of small budgets is the one considered in [2]. The results confirm that good adversarial attacks can be efficiently computed (meaning in very few steps) with the use of geometry, which was the primary objective of this study.

Note that with equal Euclidean budget, the infinity norm of the attack is higher for the AutoAttack than for the TSSA or the OSSA most of the time as shown in Figure 12. It means that, in the case of MNIST images, the absolute value of the variation of pixel intensity is higher in the case of AutoAttack for equal Euclidean budget. Therefore, attacks constructed with AutoAttack might be more noticeable to the human eye on the average.

Some examples of attacked images are also included for different budgets in order for the reader to realize that such small budgets are enough to fool the network while us - humans - still recognize very well the true digit. Other examples of attacks produced by the TSSA procedure are included in Figure 13 and in Figure 14 for different budgets.

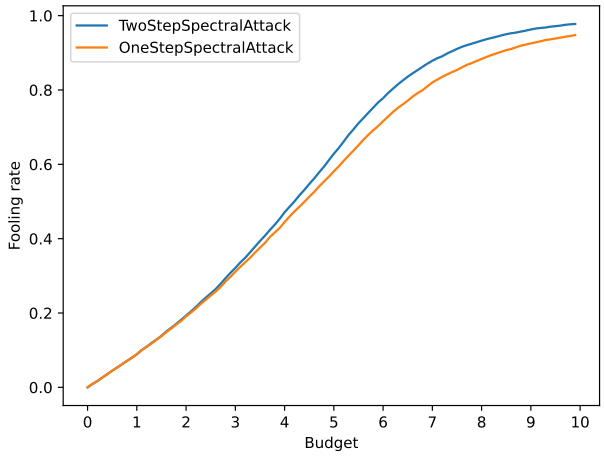


Fig. 9: Fooling rates with respect to the Euclidean budget on all MNIST test-set.

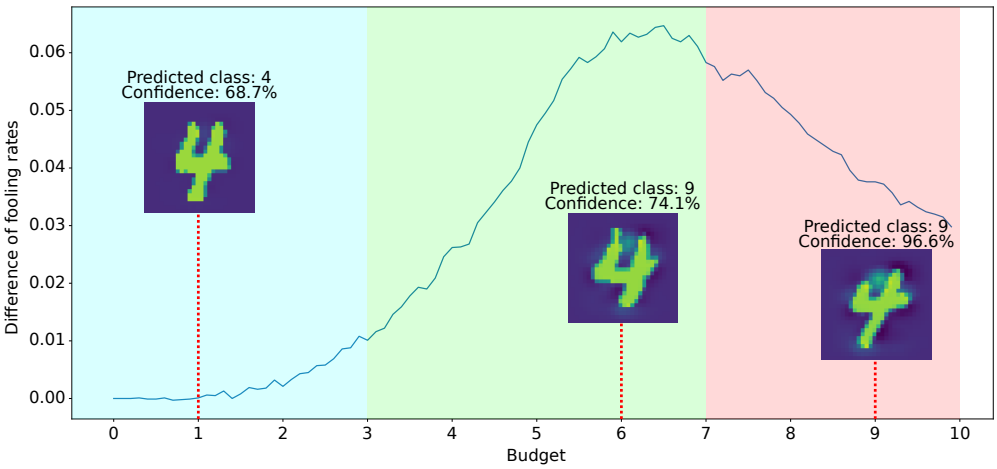


Fig. 10: Difference between the fooling rate of the TSSA ($\mathrm{fr}_{\mathrm{TSSA}}$) and the one of the OSSA ($\mathrm{fr}_{\mathrm{OSSA}}$) with respect to the Euclidean budget ($\mathrm{fr}_{\mathrm{TSSA}} - \mathrm{fr}_{\mathrm{OSSA}}$), and some examples of the attacks it produces.

Table 2: Fooling rates with respect to the Euclidean budget on all MNIST test-set (without input normalization, activation function: Sigmoid) for the 3 attack procedures.

Budget (ε)	0.50	1.00	1.50	2.00	2.50	3.00	3.50	4.00	4.50	5.00
OSSA	0.126	0.326	0.599	0.805	0.915	0.963	0.983	0.992	0.997	0.998
TSSA	0.128	0.339	0.643	0.845	0.945	0.981	0.994	0.998	1.000	1.000
AA	0.110	0.324	0.689	0.906	0.980	0.998	1.000	1.000	1.000	1.000

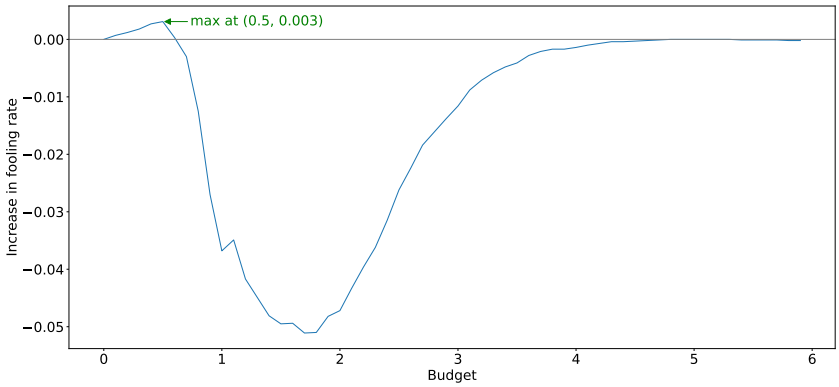
Table 3: Fooling rates with respect to the Euclidean budget on all MNIST test-set (without input normalization, activation function: ReLU) for the 3 attack procedures.

Budget (ε)	0.50	1.00	1.50	2.00	2.50	3.00	3.50	4.00	4.50	5.00
OSSA	0.088	0.309	0.614	0.820	0.923	0.970	0.988	0.996	0.998	0.999
TSSA	0.093	0.349	0.666	0.862	0.953	0.985	0.996	0.999	1.000	1.000
AA	0.090	0.386	0.716	0.909	0.979	0.997	1.000	1.000	1.000	1.000

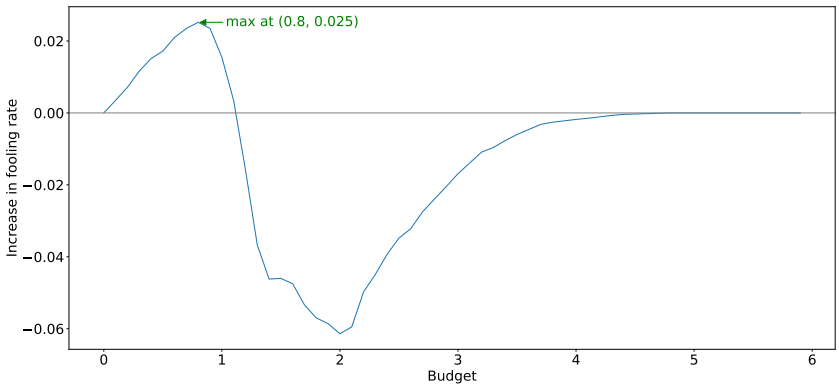
The values of the fooling rates with respect to the Euclidean budget for the 3 examined adversarial attack procedures on MNIST are detailed in Table 2 for the Sigmoid network and in Table 3 for the ReLU network.

6.3 CIFAR10 dataset

We ran similar experiments with the CIFAR10 dataset as in Section 6.2 with the MNIST dataset. The fooling rates for the One Step Spectral Attack and the Two Step Spectral Attack are represented on Figure 15 for different Euclidean budget ranging



(a) Activation function: ReLU



(b) Activation function: Sigmoid

Fig. 11: Difference between the fooling rate of the TSSA ($\mathrm{fr}_{\mathrm{TSSA}}$) and the one of AutoAttack ($\mathrm{fr}_{\mathrm{AA}}$) with respect to the Euclidean budget ($\mathrm{fr}_{\mathrm{TSSA}} - \mathrm{fr}_{\mathrm{AA}}$) on MNIST.

from 0 to 2. Figure 16 and Figure 17 show the difference between the fooling rate of the TSSA and respectively the fooling rate of OSSA and the fooling rate of AutoAttack. Figure 18 provides the comparison between the infinity norm of the attack of TSSA and AutoAttack with respect again to its Euclidean norm.

Similar conclusions to the MNIST case can be drawn from the CIFAR10 results. It strengthens our confidence in the results and their interpretations stated in Section 6.2.

The values of the fooling rates with respect to the Euclidean budget for the 3 examined adversarial attack procedures on CIFAR10 are detailed in Table 4.

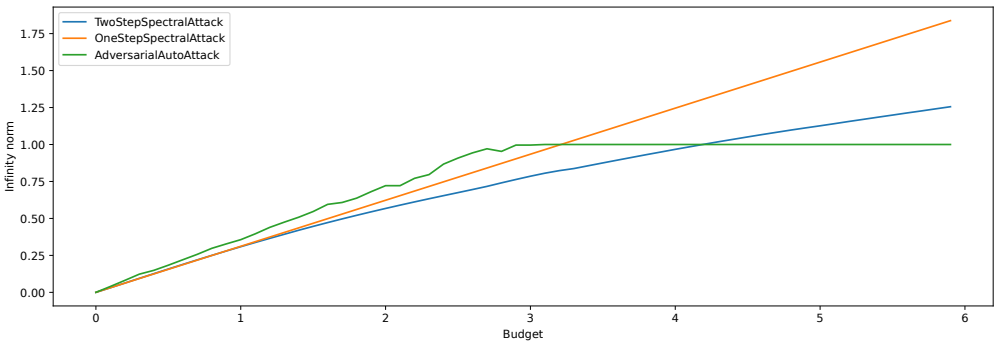


Fig. 12: $\|\cdot\|_\infty$ of the attack with respect to its Euclidean norm $\|\cdot\|_2$ on MNIST.

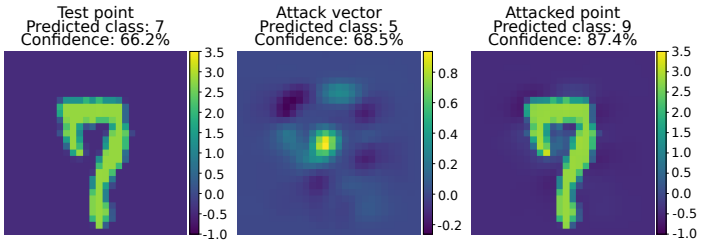


Fig. 13: Result of the TSSA with a budget $\varepsilon = 3$.

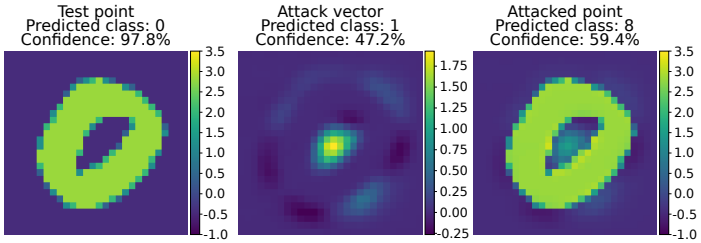


Fig. 14: Result of the TSSA with a budget $\varepsilon = 7$.

7 Conclusion

This paper explores the relationship between adversarial attacks and the curvature of the data space. Using the curvature information, we have proposed a Two Steps Attack that achieves better results than the One Step Spectral Attack presented by [1] on relatively small architectures. As the computation of the attack involves the computation of the whole Jacobian matrix of the neural network, some limitations will be reached on larger architectures. The main goal of this study is the mathematical emphasis on the role of geometrical properties of the data space through the neural network Riemannian foliations rather than providing a new numerical attacking strategy that could outperform existing state-of-art attacks for various neural network architectures. The

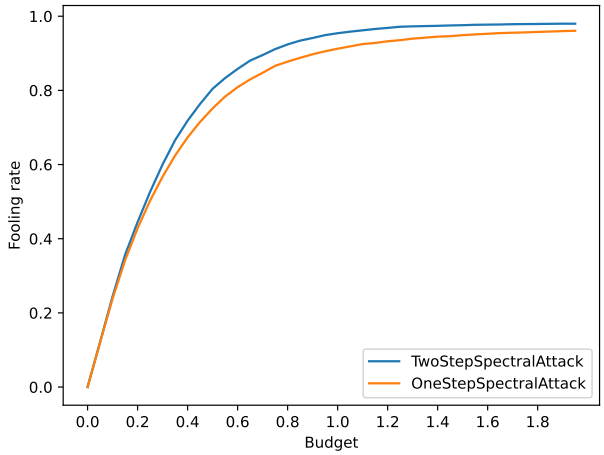


Fig. 15: Fooling rates with respect to the Euclidean budget on all CIFAR10 test-set.

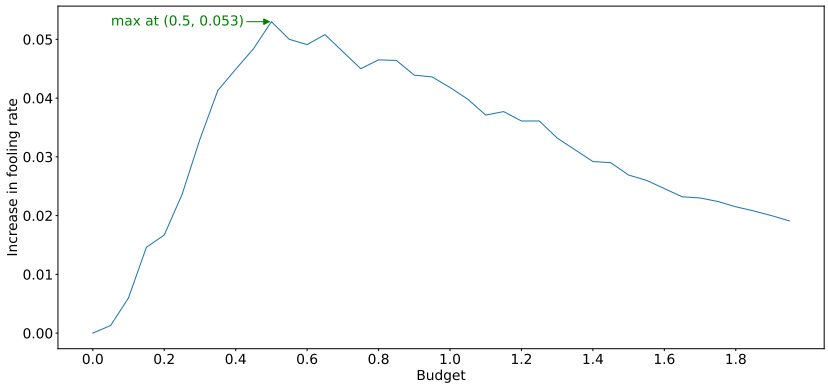


Fig. 16: Difference between the fooling rate of the TSSA (fr_{TSSA}) and the one of the OSSA (fr_{OSSA}) with respect to the Euclidean budget ($\text{fr}_{\text{TSSA}} - \text{fr}_{\text{OSSA}}$) on CIFAR10.

Table 4: Fooling rates with respect to the Euclidean budget on all CIFAR10 test-set for the 3 attack procedures.

Budget (ε)	0.20	0.40	0.60	0.80	1.00	1.20	1.40	1.60	1.80
OSSA	0.428	0.673	0.809	0.877	0.912	0.932	0.945	0.953	0.958
TSSA	0.445	0.718	0.858	0.924	0.954	0.969	0.974	0.977	0.979
AA	0.434	0.737	0.882	0.939	0.962	0.970	0.973	0.973	0.975

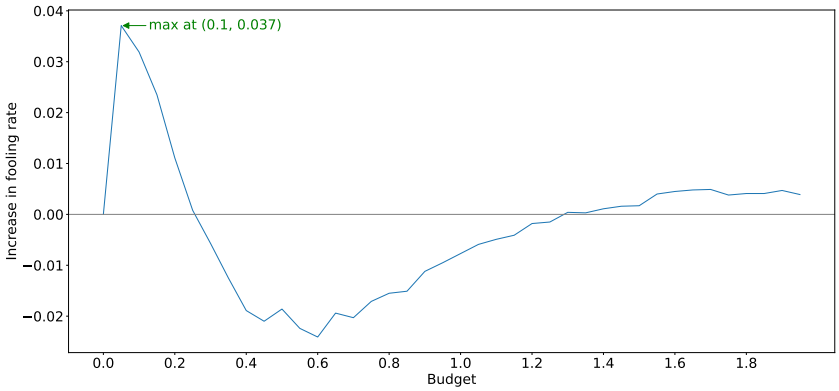


Fig. 17: Difference between the fooling rate of the TSSA ($\mathrm{fr}_{\mathrm{TSSA}}$) and the one of AutoAttack ($\mathrm{fr}_{\mathrm{AA}}$) with respect to the Euclidean budget ($\mathrm{fr}_{\mathrm{TSSA}} - \mathrm{fr}_{\mathrm{AA}}$) on CIFAR10.

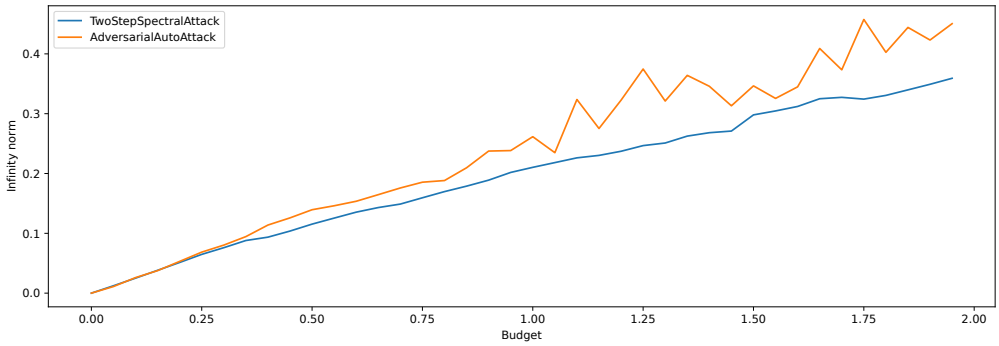


Fig. 18: $\|\cdot\|_\infty$ of the attack with respect to its Euclidean norm $\|\cdot\|_2$ on CIFAR10.

analytical mathematical expression of the proposed attack explicitly uses the curvature tensor of the FIM kernel leaves. This emphasizes the importance of geometry in the construction of an efficient attack. Additionally, with simple experiments on a toy example, we have illustrated and confirmed that exploiting such geometrical information is relevant and actually outperforms a state of the art strategy. The mathematical construction of the proposed method opens also new opportunities for future research. Indeed, it is clear that, in the case of neural networks, the geometrical properties of the leaves of the kernel foliation is related to its robustness as explained above but more generally to its power to separate data points. Therefore, the geometry of the foliation is directly linked to the complexity of the model (i.e. the neural network architecture). A deeper analysis of these aspects should help in gaining knowledge and some explainability of the underlying principles at play in neural network learning and more generally deep learning methods. This will be the focus of our future research.

Appendix A Notions of Riemannian geometry

Let $\mathcal{M} = (M, g)$ be a real Riemannian manifold of dimension n.

Definition 6 (length of a path)**.** Let $\gamma : [0,1] \to M$ be a C^1 path. The *length* of γ, denoted $l(\gamma)$ is defined by

$$l(\gamma) = \int_0^1 g\left(\gamma(t), \gamma'(t), \gamma'(t)\right)^{1/2} dt$$

Definition 7 (geodesic)**.** Let $p,\ q$ be two points of M. The C^1 path $\gamma : [0,1] \to M$ is said to be a *geodesic* between p and q if:

$$\gamma(0) = p,\ \gamma(1) = q$$
$$l(\gamma) = \inf\left\{l(\theta),\ \theta \in C^1\left([0,1),\ M\right),\ \theta(0) = p,\ \theta(1) = q\right\}$$

Definition 8 (geodesic distance)**.** The length of a geodesic between p and q is called the *geodesic distance*, denoted $d(p,q)$.

In what follows, the Levi-Civita connexion of $\mathcal{M}$ will be denoted $\overset{\circ}{\nabla}$.

Definition 9. Let $\gamma : [0,1] \to M$ be a C^2 path. It it said to be a geodesic of $\overset{\circ}{\nabla}$ is, for each $t \in]0,1[$ the following holds:

$$\overset{\circ}{\nabla}_{\dot\gamma(t)}\dot\gamma(t) = 0 \tag{A1}$$

The differential equation A1 translates in local coordinates to:

$$\frac{\partial^2\gamma^k}{\partial t^2}(t) + \Gamma^k_{ij}\left(\gamma(t)\right)\frac{\partial\gamma^i}{\partial t}(t)\frac{\partial\gamma^j}{\partial t}(t) = 0 \tag{A2}$$

For an initial point $p = \gamma(0)$, Cauchy-Lipschitz theorem shows that ther is a unique local solution to Equation A1.

Proposition A.1. *Let $p \in M$. There exist $\varepsilon > 0$ such that for all $v \in T_pM$, $\|v\| < \varepsilon$, there exist a unique geodesic $\gamma : [0,1] \to M$ such that $\gamma(0) = p$, $\gamma'(0) = v$. The function which to such v associates $\gamma(1)$ with γ the geodesic of $\overset{\circ}{\nabla}$ such that $\gamma(0) = p$, $\gamma'(0) = v$ is called the* exponential map *and denoted* $\exp_p$.

Proposition A.2. *Let $v \in T_pM$, $\|v\| < \varepsilon$ and let $q = \exp_p(v)$. Then $\gamma : t \in [0,1] \mapsto \exp_p(tv)$ is a geodesic between p and q. Besides, $l(\gamma) = \|v\|$.*

Remark 13. Note that $\|v\| = g\left(p; v, v\right)^{1/2}$ is the riemannian norm and not the Euclidean norm.

Definition 10. $d\exp_p(0) = Id$ and thus the exponential map is a local diffeomorphism. Around each point, $\exp_p$ defines a chart of M. The local coordinates we get are called the *normal coordinates* at p.

Definition 11 (logarithm map)**.** Let $p \in M$ and $\varepsilon > 0$ such that the exponential map is defined in $B(0,\varepsilon)$. For all $q \in M$ such that $d(p,q) < \varepsilon$ we set:

$$\log_p(q) = v,\ \ \exp_p v = q.$$

Remark 14. One can compute the logarithm by solving the following differential system:

$$\begin{cases} \frac{\partial^2 \gamma^k}{\partial t^2}(t) + \Gamma^k_{ij}\left(\gamma(t)\right) \frac{\partial \gamma^i}{\partial t}(t) \frac{\partial \gamma^j}{\partial t}(t) = 0 \\ \gamma(0) = p,\ \gamma(1) = q \end{cases} \tag{A3}$$

Proposition A.3. *In normal coordinates at $p \in M$, geodesics with origin p are strait lines going through the origin.*

Definition 12 (parallel transport)**.** Let $v \in T_pM$ and $q = \exp_p x$. The geodesic between p and q is $\gamma : t \in [0,1] \mapsto tx$ in normal coordinates. Besides, the linear differential equation:

$$\mathring{\nabla}_{\dot{\gamma}(t)} X(t) = 0,\ X(0) = v$$

has a solution on $[0,1]$ called the *parallel transport* of v.

Parallel transport allows to go from a tangent vector at $q = \exp_p x$ to a tangent vector at p.

Appendix B Fisher Information Metric

An important question arising when dealing with Fisher information metric is to know when going in the converse direction is feasible: given a Riemannian manifold $(\mathcal{X}, g)$, is it possible to find a probability family such that g is exactly its Fisher information? This is exactly what is behind the next definition.

Definition 13 (Statistical model)**.** A statistical model for a Riemannian manifold $(\mathcal{X}, g)$ is a probability space $(\Omega, \mathcal{T}, P)$ such that:

- It exists a family of probabilities p_x, $x \in \mathcal{X}$, absolutely continuous with respect to P.
- For any $x \in X$:

$$g_{ij}(x) = E_{p_x}\left[\partial_i \ln p_x \partial_j \ln p_x\right]$$

Remark 15. When p_x is C^2 with support not depending on x and the conditions for exchanging derivative and expectation are satisfied, then:

$$g_{ij}(x) = -E_{p_x}\left[\partial_{ij} \ln p_x\right]$$

In such a case, the metric g is Hessian.

In nearly all cases considered in machine learning, the metric g is only semi-definite. It thus makes sense to consider its kernel.

Definition 14. Let g be a semi-definite metric on a manifold $\mathcal{X}$. A tangent vector $X \in T_x\mathcal{X}$ is said to belong to the kernel $\ker_x g$ of g_x if for any $Y \in T_x\mathcal{X}$, $g(X,Y) = 0$.

Proposition B.1. *Let $(\mathcal{X}, g)$ be a connected manifold with g a semi-definite metric. If it exists a torsionless connection ∇ on $T\mathcal{X}$ such that $\nabla g = 0$, then the mapping $x \in \mathcal{X} \to \ker_x g$ defines an integrable distribution, denoted by $\ker g$.*

Proof. It is clear that for any $x \in \mathcal{X}$, $\ker_x g$ is a linear subspace of $T_x\mathcal{X}$. Let X, Y, Z be vector fields such that $Y \in \ker g$. Then, since $\nabla g = 0$ by assumption:

$$X\left(g(Y,Z)\right) = g\left(\nabla_X Y, Z\right) + g\left(Y, \nabla_X Z\right)$$

Since $Y \in \ker g$:

$$X\left(g(Y,Z)\right) = 0 = g\left(\nabla_X Y, Z\right)$$

and so, for any X, $\nabla_X Y \in \ker g$. This proves that the parallel transport of a vector in $\ker g$ is a vector in $\ker g$. The dimension of $\ker g$ is thus constant. Now, if $X, Y \in \ker g$, by the above result and since ∇ has vanishing torsion: $[X,Y] = \nabla_X Y - \nabla_Y X \in \ker g$, proving that $\ker g$ is an integrable distribution. □

Remark 16. If the dimension of $\ker g$ is not constant, then no torsionless connection ∇ can be such that $\nabla g = 0$. However, there is still a singular foliation associated with $\ker g$, with a canonical stratification by the dimension of $\ker g$.

Proposition B.2. *Under the assumptions of prop. B.1, g defines a transverse metric for the* $\ker g$ *foliation.*

Proof. This is essentially prop 3.2, p. 78 in [30]. □

The leaves of the $\ker g$ foliation are neutral submanifolds for the fisher information metric, that is moving along them will not modify the output distribution. On the other hand, the transverse metric is a measure of output variation when moving in a direction normal to the leaves.

Appendix C Proofs

Proof of Proposition 3.1. To maximize this quadratic form under the constraint $\|v\|_2^2 = \varepsilon^2$, Karush-Kuhn-Tucker's optimality conditions[11] ensure that there exists a scalar $\lambda \in \mathbb{R}$ such that the optimum attack $\hat{v}$ satisfies:

$$\begin{aligned} &\nabla_v \left(v^T G_x v\right) - \lambda \nabla_v \left(\|v\|_2^2 - \varepsilon^2\right) = 0 \\ \implies &\left(G_x + G_x^T\right)\hat{v} = \lambda 2 \hat{v} \\ \implies &G_x \hat{v} = \lambda \hat{v}. \end{aligned}$$

Otherwise said, the optimum $\hat{v}$ is in the set of G_x's eigenvectors. In this case:

$$\|\hat{v}\|_{\mathcal{X}}^2 = \hat{v}^T G_x \hat{v} = \hat{v}^T \lambda \hat{v} = \lambda \|\hat{v}\|_2^2 = \lambda \varepsilon^2.$$

Thus, $\hat{v}$ corresponding to the largest eigenvalue maximizes the Riemannian norm. □

References

[1] Zhao, C., Fletcher, P.T., Yu, M., Peng, Y., Zhang, G., Shen, C.: The Adversarial Attack and Detection under the Fisher Information Metric. Proceedings of the AAAI Conference on Artificial Intelligence **33**, 5869–5876 (2019)

[2] Croce, F., Hein, M.: Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In: III, H.D., Singh, A. (eds.) Proceedings of

[11] See [33] for more details.

the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 2206–2216. PMLR, Cambridge, MA, USA (2020). https://proceedings.mlr.press/v119/croce20b.html

[3] Fawzi, A., Fawzi, O., Frossard, P.: Analysis of classifiers' robustness to adversarial perturbations. Machine Learning **107**(3), 481–508 (2018)

[4] Shaham, U., Yamada, Y., Negahban, S.: Understanding adversarial training: Increasing local stability of supervised models through robust optimization. Neurocomputing **307**, 195–204 (2018)

[5] Kolter, J.Z., Wong, E.: Provable defenses against adversarial examples via the convex outer adversarial polytope. In: ICML (2018)

[6] Raghunathan, A., Steinhardt, J., Liang, P.: Certified Defenses against Adversarial Examples (2020)

[7] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., al., I.G.: Intriguing properties of neural networks. International Conference on learning representations (ICLR) (2014)

[8] Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015)

[9] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., Vladu, A.: Towards Deep Learning Models Resistant to Adversarial Attacks (2019)

[10] Moosavi-Dezfooli, S., Fawzi, A., Frossard, P.: Deepfool: A simple and accurate method to fool deep neural networks. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2574–2582. IEEE Computer Society, Los Alamitos, CA, USA (2016)

[11] Fawzi, A., Moosavi-Dezfooli, S.-M., Frossard, P.: The robustness of deep networks: A geometrical perspective. IEEE Signal Processing Magazine **34**(6), 50–62 (2017)

[12] Fefferman, C., Mitter, S., Narayanan, H.: Testing the manifold hypothesis. J. Amer. Math. Soc. **29**, 983–1049 (2016)

[13] Karakida, R., Akaho, S., Amari, S.-i.: Universal statistics of fisher information in deep neural networks: Mean field approach. In: Chaudhuri, K., Sugiyama, M. (eds.) Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research, vol. 89, pp. 1032–1041. PMLR, Cambridge, MA, USA (2019)

[14] Grementieri, L., Fioresi, R.: Model-Centric Data Manifold: The Data Through

the Eyes of the Model (2021)

[15] Shen, C., Peng, Y., Zhang, G., Fan, J.: Defending Against Adversarial Attacks by Suppressing the Largest Eigenvalue of Fisher Information Matrix (2019)

[16] Yan, J., Yin, H., Zhao, Z., Ge, W., Zhang, J.: Enhance adversarial robustness via geodesic distance. IEEE Transactions on Artificial Intelligence (2024)

[17] Picot, M., Messina, F., Boudiaf, M., Labeau, F., Ayed, I.B., Piantanida, P.: Adversarial robustness via fisher-rao regularization. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(3), 2698–2710 (2022)

[18] Kurakin, A., Goodfellow, I., Bengio, S.: Adversarial Machine Learning at Scale (2016)

[19] Zhu, J., Qiu, J., Guha, A., Yang, Z., Nguyen, X., Li, B., Zhao, D.: Interpolation for robust learning: data augmentation on wasserstein geodesics. In: International Conference on Machine Learning, pp. 43129–43157 (2023). PMLR

[20] LeCun, Y.: The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/ (1998)

[21] Krizhevsky, A., Nair, V., Hinton, G.: CIFAR-10 (Canadian Institute for Advanced Research). University of Toronto, ON, Canada (2009)

[22] Nielsen, F.: An elementary introduction to information geometry. Entropy **22**(10), 1100 (2020)

[23] Amari, S.-i.: Information Geometry and Its Applications vol. 194. Springer, Tokyo, Japan (2016)

[24] Fawzi, A., Fawzi, H., Fawzi, O.: Adversarial vulnerability for any classifier. Advances in neural information processing systems **31** (2018)

[25] Gilmer, J., Metz, L., Faghri, F., Schoenholz, S.S., Raghu, M., Wattenberg, M., Goodfellow, I.: Adversarial Spheres (2018)

[26] Willmore, T.J.: An Introduction to Differential Geometry. Courier Corporation, USA (2013)

[27] Dennis, J.E., Schnabel, R.B.: Numerical Methods for Unconstrained Optimization and Nonlinear Equations. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (1996)

[28] Baydin, A.G., Pearlmutter, B.A., Radul, A.A., Siskind, J.M.: Automatic differentiation in machine learning: a survey. Journal of Machine Learning Research **18**(153), 1–43 (2018)

[29] Golub, G.H., Van Loan, C.F.: Matrix Computations. JHU press, ??? (2013)

[30] Molino, P.: Riemannian Foliations. Progress in Mathematics. Birkhäuser Boston, MA, USA (1988)

[31] Strikwerda, J.C.: Finite Difference Schemes and Partial Differential Equations, Second Edition. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (2004)

[32] Tron, E.: CurvNetAttack. https://github.com/eliot-tron/CurvNetAttack

[33] H. W. Kuhn, A.W.T.: Nonlinear programming. In: Proceedings of 2nd Berkeley Symposium, vol. 2, pp. 481–492. University of California press, Berkeley, CA, USA (1951)