Robustness of Deep Learning Models and Differential Geometry

u1705902

University of Warwick, Department of Physics

22nd March 2021

Contents

Abstract

The security of Deep Learning models in safety-critical systems has been a concern for some time. It is an active area of research, both from a theoretical and practical perspective concerning methods of attacks, as well as providing guarantees on the safety of these systems which we call robustness. In this project we will aim to bridge the gap between some of the experimental and theoretical notions of robustness, as well as formulate it from a more fundamental perspective. Specifically, we will show that this new interpretation is more versatile and it also relates to the ideas present in the current theory. The different notions of robustness both theoretically and experimentally all seem to be closely related, which raises the following question. Is it possible to formulate a more fundamental theory that brings together the ideas present in the current literature? We will show that the curvature of the decision boundary plays a key role in the robustness of Deep Learning models for classification problems, and that under certain conditions random perturbations can make these algorithms more robust to targeted attacks. Finally, we will derive an explicit upper bound on the probability of misclassification, which directly depends on curvature.

1 Introduction

Machine Learning and Deep Learning algorithms are all around us. From simple e-mail spam detection or user tailored ad recommendation systems to airport facial recognition and autonomous vehicles, these algorithms have been integrated into our modern lives. There are many interesting applications of Deep Learning such as solving a Rubik's cube, text-to-speech synthesis or music generation, as seen in [26, 14, 5]. These models are able to learn complex representations from the data given to them depending on the application, and make predictions or draw conclusions from previously unseen information. However, what happens if the algorithm supervising an autonomous vehicle makes the wrong decision? The failure of self-driving cars can have catastrophic consequences, as discussed in [17]. Deep Learning algorithms are amazing for a variety of applications and it could also be argued that they attempt to simulate the human brain in a simplistic manner, but still they can be fooled. One analogy that comes into my mind is an optical illusion. Fig.1 clearly illustrates that its possible to confuse our brains with certain paintings. Optical illusions provide an insight into how our brains perceive information through vision and perception and how easy it is to confuse ourselves. If we can be fooled so easily, what chance do Deep Learning algorithms have?

The realization that neural networks may fail to correctly classify data that has been perturbed only slightly has led to a flurry of research on adversarial perturbations in deep learning, as seen

in [30, 9]. This project focuses on the problem of robustness from a theoretical perspective, but we will pay special attention to Sequential models. Examples include recurrent neural networks such as Long Short-Term Memory (LSTM) networks. Applications are found in natural language processing, financial time series predictions, or time-varying image data.

We will explore the differences between adversarial and random Perturbations in depth from a theoretical perspective. Previous work on robustness of Machine Learning and Deep Learning models have been done in the past from a theoretical perspective [23, 7] as well as tackling the problem of robustness by Jacobian and Hessian regularization [13, 25] which proposes a more practical approach. We will see that these ideas appear quite naturally when robustness is analysed from a theoretical perspective. The ultimate goal of the project is to be able to quantify how robust a certain Sequential Deep Learning model is to small and random perturbations and provide a new and more fundamental insight from which robustness can be defined. From a theoretical perspective, the probability that a certain perturbation will cause misclassification will be analysed as well as the discussion of possible defenses against targeted attacks.

As it turns out, the theoretical research in the literature is quite inconsistent and the notion of robustness is defined in a variety of different ways. This gives motivation for a more general framework of robustness, which will be defined using Differential Geometry, linking the previous ideas to Weyl's tube formula [19].

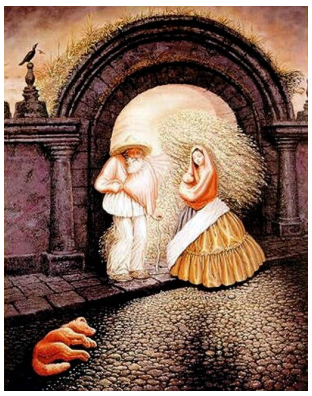

Figure 1: What do you see? An old man or two people standing in front of a gate?

2 Theory

The theory necessary for the analysis brings together a wide variety of areas in the literature. Robustness is analysed from a deterministic, probabilistic and differential-geometric interpretation. The theoretical results proposed in Section 3 aim to draw a parallel between known results in the literature and set up a more fundamental framework in which robustness can be defined as well

as provide a new insight from a differential-geometric interpretation.

2.1 Preliminaries

For the majority of this report the theoretical analysis does not assume a particular type of architecture, but some examples will be based on the multilayer perceptron or recurrent neural network (RNN) sequential models which are introduced here. Informally, a neural network is an algorithm that attempts to learn the representation and relationship between sets of data. It loosely inspired by the connections between brain cells inside the human brain in a simplified manner, making them versatile for a variety of applications. We call these connections weights and they are represented using matrices. During training we measure how well the algorithm is doing using a function called the loss function. The output of the network is then compared to a ground truth label and the weights are updated accordingly to minimise the loss. The following definitions can be found in [6, 8].

Definition 2.1 (Multilayer Perceptron Forward Propagation)**.** The forward propagation of a multilayer perceptron is defined as follows. Let M be the number of layers, where each layer has J_m, $m = 1, \dots, M$ nodes. The weight matrices from the $(m-1)$ to the m-th layer is defined by $\boldsymbol{W}^{(m-1)}$. We consider an input $\boldsymbol{x} \in \mathbb{R}^d$ and an output $\hat{\boldsymbol{y}} \in \mathbb{R}^L$. Denote the bias, output,and activation of the i-th neuron in the m-th layer by $\theta_i^{(m)}, o_i^{(m)}, \varphi_i^{(m)}(.)$ respectively. The forward propagation through the network for $m = 2, \dots, M$ is then given by,

$$\begin{aligned} \hat{\boldsymbol{y}} &= \boldsymbol{o}^{(M)},\ \boldsymbol{o}_p^{(1)} = \boldsymbol{x}\,, \\ \boldsymbol{net}^{(m)} &= [\boldsymbol{W}^{(m-1)}]^T \boldsymbol{o}^{(m-1)} + \boldsymbol{\theta}^{(m)}\,, \\ \boldsymbol{o}^{(m)} &= \boldsymbol{\varphi}^{(m)}(\boldsymbol{net}^{(m)})\,. \end{aligned}$$

where $\boldsymbol{W}^{(m-1)}$ is a $J_{m-1} \times J_m$ matrix and $\boldsymbol{net}^{(m)}$ is the pre-activation output of the m-th layer.

The main difference between a sequential model such as an RNN and a multilayer perceptron is the way information is presented to the network. In the case of a perceptron, the input $\boldsymbol{x}$ is fed into the algorithm at the same time, therefore the order in which the data is presented does not matter. On the other hand, for a recurrent neural network or (RNN for short) the input data enters the algorithm over different time steps and the output of the network at a given time step τ^* depends on the previous inputs for times $t \leq \tau^*$. This property of sequential models makes them more versatile for applications involving time-varying data and natural language processing such as machine translation.

Definition 2.2 (RNN Forward Propagation)**.** For an RNN block let $\boldsymbol{h}^{<0>}$ be the initial state at time $t = 0$, then for time $t = 1$ to $t = \tau$ we apply the following updates to a sequence of input

vectors $\boldsymbol{x}^{<t>} \in \mathbb{R}^d$ for $t \in [1, \tau]$

$$\begin{aligned}
\boldsymbol{a}^{<t>} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{<t>}\,, \\
\boldsymbol{h}^{<t>} &= \sigma_1(\boldsymbol{a}^{<t>})\,, \\
\boldsymbol{o}^{<t>} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{<t>}\,, \\
\hat{\boldsymbol{y}}^{<t>} &= \sigma_2(\boldsymbol{o}^{<t>})\,.
\end{aligned}$$

Where σ_i for $i = 1, 2$ is a general activation function, usually $\tanh(x)$ or Relu(x) = $\max(x, 0)$. $\boldsymbol{U}$,$\boldsymbol{W}$,$\boldsymbol{V}$ and $\boldsymbol{b}$,$\boldsymbol{c}$ are the weight matrices and bias vectors respectively. Note that here the superscript t here refers to the t-th input and output and not to the layer, as above.

We denote the loss for the i-th ouput by $\mathcal{L}(\hat{y}_i, y_i)$. During training the algorithm computes the sum of individual losses over all the training data and aims to minimize the overall loss. If the loss is close to 0, it means that the classifier is doing well on the training data. This is not always the case, but the algorithm always attempts to minimise this function.

Example 2.3 (Common loss functions)**.** For the i-th output $\hat{y}_i$ and ground truth label y_i, the $0-1$, quadratic and negative log-likelihood loss functions are defined as,

$$\begin{aligned}
\mathcal{L}(\hat{y}_i, y_i)_{0-1} &= \begin{cases} 1 & \hat{y}_i \neq y_i \\ 0 & \hat{y}_i = y_i \end{cases}, \\
\mathcal{L}(\hat{y}_i, y_i)_{\text{quad}} &= A(\hat{y}_i - y_i)^2\,. \\
\mathcal{L}(\hat{y}_i, y_i)_{\log} &= -y_i \log(\hat{y}_i) + (1 - y_i) log(1 - \hat{y}_i)\,.
\end{aligned}$$

where $\hat{y}$ is the output of the algorithm and y represents the ground truth.

In the case of a binary classification both the ground truth label y and the output of the network $\hat{y}$ is a probability distribution, in the range $(0, 1)$. For the majority of the project we assume a trained model and we will pay particular attention to the forward propagation output of the algorithm.

2.2 Robustness of classifiers

The following subsection introduces the deterministic interpretation of robustness of an arbitrary classifier, which can be formulated as a Lagrangian optimization problem. It illustrates why the robustness of classifiers is an important area of research and introduces one of the more common and straightforward ways to quantify it. For linear classifiers, the decision boundary is piece wise linear, which simplifies the calculations. This allows us to compute explicit bounds on robustness for simple architectures. The material presented here will follow the lecture notes of

Dr. Martin Lotz, [20] closely. Let $\mathcal{X}$ be a data space and consider a continuous map

$$f\colon \mathcal{X} \to \mathbb{R}^L,$$

where the function $f = (f_1, \ldots, f_L)$. A classifier can be interpreted as the map f, with $f_i(\boldsymbol{x})$ representing the probability of the class i. A common example of the input data space is $\mathcal{X} = \mathbb{R}^d$. This could consist of 128×128 pixel images of handwritten digits between 0 and 9, and therefore the output would have dimensions $L = 10$. The point $\boldsymbol{x} \in \mathcal{X}$ is assigned to class i if $f_i(\boldsymbol{x}) > f_j(\boldsymbol{x})$ for all $i \neq j$. We define the output difference between two classes i,j as the function,

$$g_{ij} = f_i - f_j, \qquad i \neq j.$$

This map f divides the data space $\mathcal{X}$ into L regions

$$C_j = \{\boldsymbol{x} \in \mathcal{X} \mid g_{ij} \geq 0,\ \forall\, i \neq j\}.$$

We denote the interior $\operatorname{int}(C_j)$ as the set of $\boldsymbol{x}$ which are assigned to class j. We can assume for simplicity that the data space $\mathcal{X} = \mathbb{R}^d$ and that $\forall\ j$, each C_j has non-empty interior. The boundary of the region C_j and the decision boundary can be written as,

$$\Sigma_j = \bigcup_{i \neq j} (C_j \cap C_i), \qquad \Sigma = \bigcup_j \Sigma_j.$$

We assume that the function f is continuous, therefore it follows that the g_{ij} are continuous as well. The boundary Σ divides the input space into connected components. Each $\mathbb{R}^d \backslash \Sigma$ corresponds to one class, but the C_j are not necessarily connected. For each j and $\boldsymbol{x} \in \mathbb{R}^d$ define the distance

$$\delta_j(\boldsymbol{x}) = \inf\{\|\boldsymbol{x} - \boldsymbol{y}\|\colon \boldsymbol{y} \notin C_j\} =: \operatorname{dist}(\boldsymbol{x}, \mathbb{R}^d \backslash C_j).$$

Note that unless stated otherwise, we assume for simplicity that the norm $||.||$ is the Eucledian norm. It follows that $\delta_j(\boldsymbol{x}) = 0$ if and only if $\boldsymbol{x} \in \mathbb{R}^d \backslash C_j$, and for $\boldsymbol{x} \in C_j$, $\delta_j(\boldsymbol{x})$ is the distance to misclassification. Given a point $\boldsymbol{x}$, we can denote the distance to misclassification as,

$$\delta(\boldsymbol{x}) := \sum_{i=1}^{L} \delta_i(\boldsymbol{x}). \tag{2.1}$$

This can also be defined alternatively as,

$$\delta(\boldsymbol{x}) = \inf\{\|\boldsymbol{x} - \boldsymbol{y}\|\colon \boldsymbol{y} \in \Sigma\} = \operatorname{dist}(\boldsymbol{x}, \Sigma).$$

From a theoretical perspective, any classifier can be interpreted as an arbitrary function which takes a datapoint $\boldsymbol{x} \in \mathbb{R}^d$ and outputs a vector of classes $\hat{\boldsymbol{y}} \in \mathbb{R}^L$, where the entry with the highest value corresponds to the predicted class. During training, the algorithm optimizes itself to learn the match between $\boldsymbol{x} \in \mathbb{R}^d$ and the correct class $y^* = \underset{i=1,\ldots,L}{\arg\max}\, y_i$ where $\boldsymbol{y} = (y_1, y_2, \ldots, y_L) \in \mathbb{R}^L$ is the ground truth vector of classes. Here the vector $\boldsymbol{y}$ is usually an indicator vector with 1 in

a single entry and 0 everywhere else. A data point perturbed in the correct way would cause the algorithm to make the wrong prediction, hence making the adversarial attack successful against the classifier.

2.2.1 Guarantees on robustness

For a fixed point $\boldsymbol{x} \in C_j \subset \mathbb{R}^d$, we can formulate the problem of finding the distance to misclassification as a quadratic optimization problem

$$\underset{\boldsymbol{y}}{\text{minimize}} \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|^2 \quad \text{subject to} \quad \boldsymbol{y} \in \mathbb{R}^d \backslash C_j. \tag{2.2}$$

Let $\boldsymbol{y}$ be a solution to the problem above, such that $\boldsymbol{y} \in C_j \cap C_i$ and $g = g_{ij}$. This way, calculating the distance to misclassification reduces to the following problem.

$$\underset{\boldsymbol{y}}{\text{minimize}} \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|^2 \quad \text{subject to} \quad g(\boldsymbol{y}) \leq 0. \tag{2.3}$$

We can write the Lagrangian of the (2.3) as

$$\mathcal{L}(\boldsymbol{y}, \eta) = \frac{1}{2}\|\boldsymbol{x}-\boldsymbol{y}\|^2 + \eta g(\boldsymbol{y}).$$

We assume the Lagrangian $\mathcal{L}(\boldsymbol{y}, \eta)$ as a function of $\boldsymbol{y}$ is convex for some η. The goal is to minimize $\mathcal{L}(\boldsymbol{y}, \eta)$, which can be done by computing the gradient and setting it to zero

$$\nabla \mathcal{L}(\boldsymbol{y}, \eta) = \boldsymbol{y} - \boldsymbol{x} + \eta \nabla g(\boldsymbol{y}) = 0.$$

From this we obtain the following

$$\langle \boldsymbol{y}-\boldsymbol{x}, \nabla g(\boldsymbol{y})\rangle + \eta\|\nabla g(\boldsymbol{y})\|^2 = 0, \qquad \|\boldsymbol{y}-\boldsymbol{x}\| = \eta\|\nabla g(\boldsymbol{y})\|.$$

Rearranging the first equation for η and plugging it into the second leads to,

$$\|\boldsymbol{y}-\boldsymbol{x}\| = \frac{\langle \boldsymbol{x}-\boldsymbol{y}, \nabla g(\boldsymbol{y})\rangle}{\|\nabla g(\boldsymbol{y})\|}. \tag{2.4}$$

Example 2.4 (Piecewise linear boundary)**.** Assume we have a linear classifier, with $f_i(\boldsymbol{x}) = \boldsymbol{w}_i^\top \boldsymbol{x} + b_i$. Each set C_j is then a polyhedron P, given as the intersection of the half-spaces $g_{ij} \geq 0$ for $i \neq j$ and supporting hyperplanes $H_i = \{\boldsymbol{x} \colon g_{ij}(\boldsymbol{x}) = 0\}$. The distance $\delta(\boldsymbol{x})$ is then the radius of the largest ball around $\boldsymbol{x}$ contained in P. Fix a class j, $\boldsymbol{x} \in P := C_j$, and set $\boldsymbol{w} = \boldsymbol{w}_j - \boldsymbol{w}_i$, $b = b_j - b_i$, where i is such that $\boldsymbol{y} \in C_i$ for a point $\boldsymbol{y}$ that minimizes the Lagrangian. We need to solve (2.3) with $g(\boldsymbol{y}) = \boldsymbol{w}^\top \boldsymbol{y} + b$. It is not a difficult taks to determine the distance $\boldsymbol{x}$ to the hyperplane defined by $g(\boldsymbol{y}) = 0$ and get

$$\delta(\boldsymbol{x}) = \delta_g(\boldsymbol{x}) = \frac{|\boldsymbol{w}^\top \boldsymbol{x} + b|}{\|\boldsymbol{w}\|} = \frac{|g(\boldsymbol{x})|}{\|\nabla g(\boldsymbol{x})\|}.$$

The piecewise linear case is relevant, as it applies to neural networks with ReLU activation. It is clear that if the attacker has full control of the architecture, it is not a difficult problem to generate adversarial perturbations. This is the basis of the DeepFool classifier [24].

2.2.2 Jacobian Regularization, Curvature and Robustness

We now turn to a slightly more general bound on robustness for an arbitrary L class classifier. The following theorem defines the adversarial distance, which is exactly the same definition of $\delta(\boldsymbol{x})$ as above. This is a deterministic bound on robustness and it involves the Cross-Lipschitz constant as in [12], which is defined as the norm of the gradient of the output difference $g_{cj} = f_c - f_j$, where f_c is the output of the desired class.

Theorem 2.5. *Let $\boldsymbol{x} \in \mathbb{R}^d$ and $f : \mathbb{R}^d \longrightarrow \mathbb{R}^L$ be a multi-class classifier with continuously differentiable components and let $i = \arg\max_{l=1,\dots,L} f_l(x)$ be the class which f predicts for $\boldsymbol{x}$ s.t. $\boldsymbol{x} \in int(C_i)$. Then,*

$$||\boldsymbol{\delta}(\boldsymbol{x})||_2 \geq \max_{R>0} \min \left\{ \min_{j \neq i} \frac{g_{ij}(\boldsymbol{x})}{\max\limits_{\boldsymbol{y} \in \mathbb{B}(\boldsymbol{x},R)} ||\nabla g_{ij}(\boldsymbol{y})||_2}, R \right\} := \alpha$$

Proof. Let $\boldsymbol{\delta} \in \mathbb{R}^d$ be a perturbation such that $\boldsymbol{x} + \boldsymbol{\delta} \in C_j \; j \neq i$. Then

$$g_{ij}(\boldsymbol{x} + \boldsymbol{\delta}) - g_{ij}(\boldsymbol{x}) = \int_0^1 \langle \nabla g_{ij}(\boldsymbol{x} + t\boldsymbol{\delta}), \boldsymbol{\delta} \rangle dt \,,$$

noticing that $g_{ij}(\boldsymbol{x}) > 0, g_{ij}(\boldsymbol{x} + \boldsymbol{\delta}) < 0$ and $g_{ij} = -g_{ji}$ we obtain,

$$0 \leq g_{ij}(\boldsymbol{x}) \leq \int_0^1 \langle \nabla g_{ji}(\boldsymbol{x} + t\boldsymbol{\delta}), \boldsymbol{\delta} \rangle dt \leq ||\delta|| \int_0^1 ||\nabla g_{ij}(\boldsymbol{x} + t\boldsymbol{\delta})|| dt \,,$$

Then we obtain that $\boldsymbol{\delta} \in \mathbb{R}^d$ satisifes,

$$||\boldsymbol{\delta}|| \geq \frac{g_{ij}}{\int_0^1 ||\nabla g_{ij}(\boldsymbol{x} + t\boldsymbol{\delta})|| dt} \,,$$

if we fix some radius $R > 0$ s.t $||\boldsymbol{\delta}|| \leq R$,

$$\int_0^1 ||\nabla g_{ij}(\boldsymbol{x} + t\boldsymbol{\delta})|| dt \leq \max_{\boldsymbol{y} \in \mathbb{B}(\boldsymbol{x},R)} ||\nabla g_{ij}(\boldsymbol{y})||_2 \,, \tag{2.5}$$

therefore it follows that $||\boldsymbol{\delta}||$ can be written as

$$||\boldsymbol{\delta}|| \geq \min \left\{ \frac{g_{ij}}{\max\limits_{\boldsymbol{y} \in \mathbb{B}(\boldsymbol{x},R)} ||\nabla g_{ij}(\boldsymbol{y})||_2}, R \right\} \,,$$

such that the classifier decision changes from class $i \to j$. This is because if $||\boldsymbol{\delta}(\boldsymbol{x})|| \leq R$, then the original bound holds because of (2.5) and if $||\boldsymbol{\delta}(\boldsymbol{x})|| > R$ it holds automatically. □

We notice that it is required that the norm of the gradients $||\nabla g_{ij}(\boldsymbol{y})||$ are bounded for this inequality to make sense. More precisely, if $||\nabla g_{ij}(\boldsymbol{y})||$ is bounded above, it would place a lower bound on $||\boldsymbol{\delta}||$. A natural idea that comes into mind is to include the Cross-Lipshcitz terms as a regularizer during training, which is presented in [12] and has the form,

$$\frac{1}{nL} \sum_{l=1}^{n} \sum_{i=1}^{L} ||\nabla g_{ij}(\boldsymbol{x}_l)||^2 \,.$$

This idea is closely related to Jacobian regularization as seen in [13], which considers a more practical perspective. Define the Jacobian regularization by considering the functions f_i and the term that appears in the loss function takes the form,

$$\frac{\lambda_{JR}}{2} \sum_{l=1}^{n} \sum_{i=1}^{L} ||\nabla f_i(\boldsymbol{x}_l)||^2 \,. \tag{2.6}$$

Here the regularizer considers the gradients $||\nabla f_i||$ and not the Cross-Lipshchitz terms $||\nabla g_{ij}||$. These two notions are closely related. The term λ_{JR} is a tuning parameter, it should be interpreted as a measure of how much the term $||\nabla f_i(\boldsymbol{x}_l)||^2$ is penalized during training.

2.3 Differential-geometric interpretation

Some of the theoretical results borrow tools from Differential Geometry so this section introduces the necessary theory needed to understand the analysis. When considering a classifier we will refer to it as a map $f : \mathbb{R}^d \longrightarrow \mathbb{R}$ and assume that it is smooth, i.e. $f \in C^\infty(\mathbb{R}^d)$. For an arbitrary classifiers we will take the decision boundary as a hypersurface in d-dimensional euclidean space $\mathbb{R}^d$. The general references for the background material presented in this section can found in [10, 32, 15, 16, 29].

Definition 2.6. The differential of a map $f : \mathbb{R}^d \longrightarrow \mathbb{R}$ is defined as $\mathrm{d}f_{\boldsymbol{p}} : \mathbb{R}^d \longrightarrow \mathbb{R}$. It is characterized by $f(\boldsymbol{x} + \boldsymbol{v}) = f(\boldsymbol{x}) + \mathrm{d}f_{\boldsymbol{p}} \cdot \boldsymbol{v} + \varphi(\boldsymbol{v})$, where

$$\mathrm{d}f_{\boldsymbol{p}}(\boldsymbol{v}) = \langle \nabla f(\boldsymbol{p}), \boldsymbol{v} \rangle \tag{2.7}$$

Definition 2.7. Consider a map $f : \mathbb{R}^d \longrightarrow \mathbb{R}$. The point $\boldsymbol{p} \in \mathbb{R}^d$ is called a regular point if $\nabla f(\boldsymbol{p}) \neq 0$, and $c \in \mathbb{R}$ is a regular value if every $\boldsymbol{p} \in f^{-1}(c)$ is a regular point.

Definition 2.8. A hypersurface in $\mathbb{R}^d$ is a set $\mathcal{M} = f^{-1}(c)$ where c is the regular value of the map $f : \mathbb{R}^d \longrightarrow \mathbb{R}$. The tangent space $T_{\boldsymbol{p}}\mathcal{M}$ at $\boldsymbol{p} \in \mathcal{M}$ can be defined as,

$$T_{\boldsymbol{p}}\mathcal{M} = \ker \mathrm{d}f_{\boldsymbol{p}} = \{\boldsymbol{v} \in \mathbb{R}^d | \langle \nabla f(\boldsymbol{p}), \boldsymbol{v} \rangle = 0\}$$

This is a linear subspace of $\mathbb{R}^d$ of dimensions $(d-1)$ or equivalently, codimension 1. The hypersurfaces we will consider are assumed to be regular, therefore we will just refer to them as hypersurfaces.

A regular hypersurface on $\mathbb{R}^d$ can be parameterized using local coordinates around a point $\boldsymbol{p} \in \mathcal{M}$. $\forall\, \boldsymbol{p} \in \mathcal{M}\ \exists$ open neighborhood $U \subset \mathbb{R}^{d-1}, V \subset \mathcal{M}$ and a homeomorphism $\varphi : U \longrightarrow V$ s.t. $\varphi(\boldsymbol{0}) = \boldsymbol{p}$. This is called a local parameterization. The most straightforward way to parameterize the surface desrcibing the decision boundary is given by the Implicit Function Theorem,

Theorem 2.9 (Implicit Function Theorem)**.** *Given an open neighborhood $U \subset \mathbb{R}^d$, a smooth map $f : \mathbb{R}^d \longrightarrow \mathbb{R}$ and a point $\boldsymbol{p} = (\boldsymbol{p}', q) \in \mathbb{R}^{d-1} \times \mathbb{R}$, where $\boldsymbol{p} \in U$ is a point for which,*

$$f(\boldsymbol{p}) = 0 \quad \text{and} \quad \frac{\partial f}{\partial x_d}(\boldsymbol{p}) \neq 0$$

then there exists and open set $U' \subset \mathbb{R}^{d-1}$ with the point $\mathbf{p}' \in U'$ and a unique continuously differentiable function $F : U' \longrightarrow \mathbb{R}$ such that $q = F(\boldsymbol{p}')$ and

$$f(\tilde{\boldsymbol{x}}, F(\tilde{\boldsymbol{x}})) = 0$$

holds for $\tilde{\boldsymbol{x}} \in U'$.

Conveniently, it is possible to describe $T_{\boldsymbol{p}}\mathcal{M}$ as the image $T_{\boldsymbol{p}}\mathcal{M} =$im $d\varphi_{\mathbf{0}}(\mathbb{R}^{d-1})$ with basis vectors,

$$\frac{\partial}{\partial x^i} = d\varphi_{\mathbf{0}}(\boldsymbol{e}_i) = \left.\frac{\partial \varphi}{\partial x_i}\right|_{\boldsymbol{x}=\mathbf{0}}$$

Here, the $\boldsymbol{e}_i$ denotes the standard basis vectors on $\mathbb{R}^d$.

Definition 2.10 (Directional derivative)**.** Suppose $X_{\boldsymbol{p}} = \sum a^i \left.\frac{\partial}{\partial x^i}\right|_{\boldsymbol{p}}$ is a tangent vector at a point $\boldsymbol{p} = (p^1, \ldots, p^d) \in \mathbb{R}^d$ and $f(x^1, \ldots, x^d) \in C^\infty$ in a neighborhood of $p \in \mathbb{R}^d$. First we write down a set of parametric equations for the line through p in the direction $X_{\boldsymbol{p}}$:

$$x^i = p^i + ta^i, i = 1, \ldots, n$$

The directional derivative at p in the direction $X_{\boldsymbol{p}}$ is defined by,

$$D_{X_{\boldsymbol{p}}} f = \lim_{t \longrightarrow 0} \frac{f(p + ta) - f(p)}{t}$$

We can easily define the second fundamental form for a hypersurface in $\mathbb{R}^3$, however we will see later that this idea generalises nicely in higher dimensions as well. A normal vector at a point $\boldsymbol{p} \in \mathcal{M}$ is a vector $\boldsymbol{n}$ such that $\langle \boldsymbol{n}, \boldsymbol{v} \rangle = 0, \ \forall\, \boldsymbol{v} \in T_{\boldsymbol{p}}\mathcal{M}$. We denote the set of normal vectors at the point $\boldsymbol{p} \in \mathcal{M}$ as $N_{\boldsymbol{p}}\mathcal{M}$

Definition 2.11 (Second fundamental form)**.** Let $\varphi = \varphi(u, v)$ be a parameterization of a surface in $\mathbb{R}^3$. Given a field of unit normal vectors $\boldsymbol{n} = \frac{\boldsymbol{n}_u \times \boldsymbol{n}_v}{|\boldsymbol{n}_u \times \boldsymbol{n}_v|}$ where $\boldsymbol{n}_u$ denotes the partial derivative of φ w.r.t u define the second fundamental form as $\mathbb{I} = Ldu^2 + 2Mdudv + Ndv^2$ where the coefficients can be calculated from $L = \varphi_{uu} \cdot \boldsymbol{n}$, $M = \varphi_{uv} \cdot \boldsymbol{n}$, $N = \varphi_{vv} \cdot \boldsymbol{n}$.

Definition 2.12 (Shape Operator)**.** Let p be a point on a surface $\mathcal{M} \in \mathbb{R}^d$ and let $\mathcal{N}$ be a C^∞ unit normal vector field on $\mathcal{M}$. For any tangent vector $X_{\boldsymbol{p}} \in T_{\boldsymbol{p}}\mathcal{M}$ define $L_{\boldsymbol{p}}(X_{\boldsymbol{p}}) = -D_{X_{\boldsymbol{p}}}\mathcal{N}$. The linear map $L_{\boldsymbol{p}} : T_{\boldsymbol{p}}\mathcal{M} \longrightarrow T_{\boldsymbol{p}}\mathcal{M}$ is called the shape operator of the surface M at p. Where $D_{X_{\boldsymbol{p}}}$ is the directional derivative along $X_{\boldsymbol{p}}$. Here C^∞ denotes the space of smooth functions.

This is convenient, as now we can define the components of the second fundamental form w.r.t. an orthonormal bases in terms of the shape operator.

Definition 2.13. At each point p of an oriented hypersurface $\mathcal{M} \subset \mathbb{R}^d$ the second fundamental form is defined as,

$$\mathbb{I}(X_{\boldsymbol{p}}, Y_{\boldsymbol{p}}) = \langle L(X_{\boldsymbol{p}}), Y_{\boldsymbol{p}} \rangle$$

where $Y_{\boldsymbol{p}}, X_{\boldsymbol{p}} \in T_{\boldsymbol{p}}\mathcal{M}$ is a tangent vector field at the point p.

Let $E_1, \ldots, E_{d-1}$ be an orthonormal frame field. This means that the E_i for $1 \leq i \leq d-1$ is a tangent vector field and the $\{E_1, \ldots, E_{d-1}\}$ form an orthonormal basis of $T_{\boldsymbol{p}}\mathcal{M}$ at the point $\boldsymbol{p}$. The goal is to express the entries of the shape operator in terms of a function $f : \mathbb{R}^d \longrightarrow \mathbb{R}$ which defines the hypersurface $\mathcal{M}$. We make use of the fact that for $\{E_1, \ldots, E_{d-1}\}$ there exist local coordinates $(x^1, \ldots, x^{d-1})$ around the point $\boldsymbol{p}$ such that $E_i = \frac{\partial}{\partial x^i}$. We can represent the entries of the shape operator w.r.t. a suitable choice of basis vectors $\frac{\partial}{\partial x^i}$ at the point $\boldsymbol{p}$. This can be interpreted as a matrix $S_{\boldsymbol{p}}$ with entries,

$$(S_{\boldsymbol{p}})_{i,j} = \mathbb{I}\left(\frac{\partial}{\partial x^i}, \frac{\partial}{\partial x^j}\right)$$

The eigenvalues of $S_{\boldsymbol{p}}$ are denoted by $\kappa_1, \ldots, \kappa_{d-1}$ and these are the principal curvatures of $\mathcal{M}$ at the point $\boldsymbol{p}$.

Example 2.14. If $x^1, \ldots, x^{d-1}$ are the standard coordinates on $\mathbb{R}^{d-1}$ and $p \in \mathbb{R}^d$ then the partial derivatives

$$\left.\frac{\partial}{\partial x^1}\right|_{\boldsymbol{p}}, \ldots, \left.\frac{\partial}{\partial x^d}\right|_{\boldsymbol{p}}$$

are tangent vectors at p that form a basis for the tangent space $T_{\boldsymbol{p}}(\mathbb{R}^d)$

It is now possible to relate the hessian to the components of the second fundamental form, using a variation of the following Lemma from [33]. For further information please see [16] Ch. VII, Ex. 3.3.

Lemma 2.15. *Let $\varphi : U \longrightarrow V \subset \mathcal{M}$ be a local parameterization at the point $\boldsymbol{p} \in V$ with normal coordinates $(x^1, \ldots, x^{d-1})$. The entries of the shape operator w.r.t a unit normal vector field $\boldsymbol{n}$ are given by,*

$$\mathbb{I}\left(\frac{\partial}{\partial x^i}, \frac{\partial}{\partial x^j}\right) = \left\langle \frac{\partial^2 \varphi}{\partial x^i \partial x^j}, \boldsymbol{n} \right\rangle$$

Proof. We begin by noticing that the $\frac{\partial}{\partial x_i} = d\varphi_{\mathbf{0}}(e_i) = \frac{\partial \varphi}{\partial x^i}$ form an orthonormal basis of $T_{\boldsymbol{p}}\mathcal{M}$. Then,

$$\frac{\partial}{\partial x^i}\left\langle \boldsymbol{n}, \frac{\partial \varphi}{\partial x^j} \right\rangle = \left\langle \frac{\partial \boldsymbol{n}}{\partial x^i}, \frac{\partial \varphi}{\partial x^j} \right\rangle + \left\langle \boldsymbol{n}, \frac{\partial^2 \varphi}{\partial x^i \partial x^j} \right\rangle = 0$$

where the first term on the right hand side is the negative of the Shape operator and the sum is equal to 0 by construction. The result then follows immediately. □

We interpret the second order partial derivatives as a Hessian matrix for a function $f : \mathbb{R}^d \longrightarrow \mathbb{R}$. We denote the entries of the Hessian matrix of f at the point $\boldsymbol{p}$ as,

$$H_{\boldsymbol{p}}(f) = \left(\frac{\partial^2 f}{\partial x^i \partial x^j} \right), \ \forall\, 1 \leq i, j \leq d$$

Corollary 2.16. *Let $\mathcal{M} = f^{-1}(c)$ be a smooth hypersurface. The entries of the shape operator w.r.t. an orthonormal frame $E_1, \ldots, E_{d-1}$ at the point $\boldsymbol{p} \in \mathcal{M}$ is written as,*

$$\mathbb{I}(E_i, E_j) = -\frac{1}{||\nabla f(\boldsymbol{p})||} E_i^T H_{\boldsymbol{p}}(f) E_j. \tag{2.8}$$

Proof. Recall that the local coordinates in $\mathbb{R}^d$ are written with lower indices x_i. Noting that $f(\varphi(\boldsymbol{x})) = c$ and letting $\boldsymbol{p} = \varphi(\boldsymbol{x})$ leads to,

$$\begin{aligned} 0 = \frac{\partial^2 f(\varphi(\boldsymbol{x}))}{\partial x^i \partial x^j} &= \sum_l \left(\sum_k \frac{\partial^2 f(\boldsymbol{p})}{\partial x_l \partial x_k} \frac{\partial \varphi_l}{\partial x^i} \frac{\partial \varphi_k}{\partial x^j} + \frac{\partial f(\boldsymbol{p})}{\partial x_l} \frac{\partial^2 \varphi_l}{\partial x^i \partial x^j} \right), \\ &= \left\langle \frac{\partial^2 \varphi}{\partial x^i \partial x^j}, \nabla f(\boldsymbol{p}) \right\rangle + \left(\frac{\partial \varphi}{\partial x^i} \right)^T H_{\boldsymbol{p}}(f) \left(\frac{\partial \varphi}{\partial x^j} \right). \end{aligned}$$

We finish the proof by dividing by the norm $||\nabla f(\boldsymbol{p})||$ and the result follows immediately. □

From [18] establishing the following lemma will be useful as it leads us to the notion of planar curvature. In 2-dimensions the decision boundary can be described by the function $f(x, y) = 0$.

Lemma 2.17. *The planar curvature of a decision boundary in 2-D input space is given by,*

$$k_r = -\frac{f_{xx} f_y^2 - 2 f_x f_y f_{yx} + f_x^2 f_{yy}}{f_y^3} \left(1 + \frac{f_x^2}{f_y^2} \right)^{\frac{3}{2}}$$

where $f = f(x, y)$ the output of the neural network and $f_x = \frac{\partial f}{\partial x}$ and so on.

2.4 Volumes of tubular neighborhoods, a more general idea of robustness

In this section we outline an expression for the volume of tubular neighborhoods around a hypersurface, from H.Weyl [11]. The material presented in this section follows the same setting as seen in [19]. We can write the characteristic polynomial of the shape operator as,

$$det(\mathbf{1} - tS_{\boldsymbol{p}}) = \sum_{i=0}^{d-1} t^i \psi_i(\kappa_1, \ldots, \kappa_{d-1})$$

where the ψ_i are up to sign, the elementary symmetric functions of the principal curvatures.

$$\psi_i(\kappa_1, \ldots, \kappa_{d-1}) = (-1)^i \sigma_i(\kappa_1, \ldots, \kappa_{d-1})$$

and the $\sigma_i(\kappa_1, \ldots, \kappa_{d-1})$ are elementary symmetric functions are given by the following definition.

Definition 2.18. Given a polynomial of n variables $x_1, \ldots, x_n$ we define the n elementary symmetric polynomials as,

$$\begin{aligned}
\sigma_0 &= 1\,, \\
\sigma_1 &= x_1 + \cdots + x_n\,, \\
\sigma_2 &= x_1x_2 + x_1x_3 + \cdots + x_{n-1}x_n = \sum_{i<j} x_ix_j\,, \\
\sigma_3 &= \sum_{i<j<k} x_ix_jx_k\,, \\
&\vdots \\
\sigma_n &= x_1x_2\ldots x_n\,.
\end{aligned}$$

Define the integrals of absolute curvature as follows, for more information see [15].

Definition 2.19. Let $\mathcal{M}$ be a compact hypersurface of $\mathbb{R}^d$. The integrals of absolute curvature can be defined as,

$$|K_i|(\mathcal{M}) = 2\int_{\mathcal{M}} |\psi_i(\kappa_1, \ldots, \kappa_{d-1})| d\mathcal{M}$$

Here $d\mathcal{M}$ is the natural Riemannian volume form on $\mathcal{M}$. Furthermore, note that the factor of 2 comes from the fact that we are considering a two-sided tubular neighborhood. This factor disappears in the setting where only a one-sided tubular neighborhood is considered.

We can define the closed tube of radius ε around a compact hypersurface $\mathcal{M} \in \mathbb{R}^d$ as the set,

$$T(\mathcal{M}, \varepsilon) = \left\{p \in \mathbb{R}^d \mid \exists \text{ line of length} \leq \varepsilon \text{ from } p \text{ meeting } \mathcal{M} \text{ orthogonally}\right\}$$

We denote the volume of this tubular region as $\operatorname{vol} T(\mathcal{M}, \varepsilon)$, which is just the Lebesgue measure. The following Theorem is a variation of [19] (Theorem 3.1)

Theorem 2.20. *Let $\mathcal{M}$ be a compact hypersurface. Then $\forall\, \varepsilon > 0$,*

$$\operatorname{vol} T(\mathcal{M}, \varepsilon) \leq \sum_{i=0}^{d-1} \frac{1}{1+i} |K_i|(\mathcal{M}) \varepsilon^{i+1}$$

We can formulate a probabilistic interpretation of Theorem 2.20 in the following way. Assume that X is uniformly distributed around a point $\boldsymbol{x} \in \mathbb{R}^d$ in a ball $B(\boldsymbol{x}, \sigma)$ of radius σ. We consider the tubular region around $\mathcal{M}$ such that $T(\mathcal{M}, \varepsilon) \subset \mathbb{B}(\boldsymbol{x}, \sigma)$, then the probability that X gets ε close to the tubular neighborhood can be written as,

$$\mathbb{P}\{\operatorname{dist}(X, \mathcal{M}) \leq \varepsilon\} = \frac{\operatorname{vol} T(\mathcal{M}, \varepsilon)}{\operatorname{vol} \mathbb{B}(\boldsymbol{x}, \sigma)}\,.$$

From this point onwards, we can consider an open subset $\mathcal{M}'$ such that,

$$T(\mathcal{M}, \varepsilon) \cap \mathbb{B}(\boldsymbol{x}, \sigma) \subseteq T(\mathcal{M}', \varepsilon)\,.$$

Note that the set $\mathcal{M}'$ is not compact, therefore the result of the following theorem is not completely obvious. See [19] for more detail.

Theorem 2.21. *Let $\mathcal{M}'$ be an open subset such that $T(\mathcal{M}, \varepsilon) \cap \mathbb{B}(\boldsymbol{x}, \sigma) \subseteq T(\mathcal{M}', \varepsilon)$. Then $\forall\, \varepsilon > 0$ the following holds,*

$$\mathbb{P}\{\operatorname{dist}(X, \mathcal{M}) \leq \varepsilon\} \leq \frac{1}{\sigma^d \operatorname{vol} \mathbb{B}(\mathbf{0}, 1)} \sum_{i=0}^{d-1} \frac{1}{i+1} |K_i|(\mathcal{M}') \varepsilon^{i+1} \,. \tag{2.9}$$

We remember that the tube $T(\mathcal{M}, \varepsilon)$ is the orthogonal tube around $\mathcal{M}$. We can write it as the set of points $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{v}$ for $\boldsymbol{x} \in \mathcal{M}$. The vector $\boldsymbol{v}$ is a normal vector with length $||\boldsymbol{v}|| \leq \varepsilon$. A hypersurface $\{\boldsymbol{x} \mid g(\boldsymbol{x}) = 0\}$ is orientable and the two normal vector fields can be written as,

$$\boldsymbol{n}(\boldsymbol{z})_+ = \frac{\nabla g(\boldsymbol{x})}{||\nabla g(\boldsymbol{x})||}, \qquad \boldsymbol{n}(\boldsymbol{z})_- = -\frac{\nabla g(\boldsymbol{x})}{||\nabla g(\boldsymbol{x})||} \,.$$

The one sided tube can be defined as,

$$T_{+/-}(\mathcal{M}, \varepsilon) = \{\boldsymbol{z} + t\boldsymbol{n}(\boldsymbol{z})_{+/-} \mid t \in [0, \varepsilon]\} \,.$$

3 Results

This section presents the theoretical results and proofs. An intuitive explanation of the implications of the proposed theorems will be given as well. The theoretical analysis draws a parallel between existing results in an attempt to clear up the inconsistencies of the notion of Robustness as well as provide a new interpretation and insight behind it. It is convenient to define a more general setting as seen in [23]. The space $\mathcal{X}$ introduced earlier will be taken as $\mathbb{R}^d$.

Definition 3.1 (Adversarial Perturbation)**.** Consider an arbitrary L-class classifier $f : \mathbb{R}^d \longrightarrow \mathbb{R}^L$ where d is the dimensions of the input $\boldsymbol{x}$ and L is the number of classes. Given a point $\boldsymbol{x} \in \mathbb{R}^d$ we define $\hat{k}(\boldsymbol{x}) = \arg\max_k f_k(\boldsymbol{x})$ to be the correct label corresponding to class k. An adversarial perturbation is defined by the following optimization problem. Find $\boldsymbol{\delta}(\boldsymbol{x}) = \arg\min_{r \in \mathbb{R}} ||\boldsymbol{r}(\boldsymbol{x})||_2$ subject to $\hat{k}(\boldsymbol{x} + \boldsymbol{r}) \neq \hat{k}(\boldsymbol{x})$. In other words, it is the smallest distance the data point $\boldsymbol{x}$ needs to be perturbed such that it will be misclassified.

3.1 Jacobian Regularization, the Hessian and worst case robustness

The idea of Jacobian and Cross-Lipschitz regularization is closely related. If there is a bound on the norm of the Jacobian of f at any point, then the norm $||\nabla g_{cj}||$ is bounded as well.

Lemma 3.2. *Consider a map $f : \mathbb{R}^d \longrightarrow \mathbb{R}^L$. Assume that $\exists\, M \in \mathbb{R}$, $M > 0$ such that $||J(\boldsymbol{x})||_F \leq M$ for any point $\boldsymbol{x} \in \mathbb{R}^d$, then*

$$||\nabla g_{cj}||_2 \leq M \,.$$

Proof. Begin by writing out the frobenius norm of the Jacobian matrix explicitly,

$$||J(\boldsymbol{x})||_F^2 = \sum_i \sum_j \left| \frac{\partial f_i}{\partial x_j}(\boldsymbol{x}) \right|^2 \leq M^2 \tag{3.1}$$

Then notice that,

$$||\nabla g_{cj}||_2^2 = ||\nabla f_c - \nabla f_j||_2^2 = \sum_l \left| \frac{\partial f_c}{\partial x_l}(\boldsymbol{x}) - \frac{\partial f_j}{\partial x_l}(\boldsymbol{x}) \right|^2 \leq \sum_l \left| \frac{\partial f_c}{\partial x_l}(\boldsymbol{x}) \right|^2 + \left| \frac{\partial f_j}{\partial x_l}(\boldsymbol{x}) \right|^2$$

From Equation 3.1 pick $i \in \{c, j\}$ the result follows immediately,

$$\begin{aligned} ||\nabla g_{cj}||_2^2 &\leq M^2 \,. \\ ||\nabla g_{cj}||_2 &\leq M \,. \end{aligned}$$

□

It is clear that if Jacobian Regularization is integrated into the loss function, the entries $J_{i,j}(\boldsymbol{x}) = \frac{\partial f_i}{\partial x_j}(\boldsymbol{x})$ will not be too large. This ensures that the Cross-Lipschitz term from Theorem 2.5 has an upper bound for all points $\boldsymbol{x}$, which then places a minimum lower bound on $||\boldsymbol{\delta}(\boldsymbol{x})||_2$. As seen in [13], Jacobian Regularization pushes the decision surfaces outwards w.r.t. each datapoint $\boldsymbol{x} \in \mathbb{R}^d$, reducing the instability and rough edges of the decision surface. This is consistent with Lemma 3.2 because if the decision boundary is pushed outwards, the classifier will be immune to perturbations of a certain size. This is the same as saying $\exists\ \delta^* \in \mathbb{R}$ s.t. $||\boldsymbol{\delta}(\boldsymbol{x})||_2 \geq \delta^*$ for all points $\boldsymbol{x} \in \mathbb{R}^d$. This lower bound does not depend on direction, therefore it also naturally comes out that generally the boundary would become smoother as well.

3.2 Shifting of the decision boundary

The shifting of the decision boundary of an RNN comes out naturally when considering the planar curvature in 2-dimensions. Here we consider a two dimensional input $\boldsymbol{x}$ for a simplistic RNN architecture of the following form $f^{<t>}(\boldsymbol{x}) = (W_{ay})^T \sigma(W_{aa}\boldsymbol{a}^{<t-1>} + W_{ax}\boldsymbol{x}^{<t>})$, which is similar to Definition 2.2. For this particular example assume that the map $f^{<t>} : \mathbb{R}^2 \longrightarrow \mathbb{R}^L$ is the output of the network at time t. Then, W_{aa}, W_{ax} are $(L \times 2)$ weight matrices and $W_{ay} \in \mathbb{R}^L$ is a $(L \times 1)$ vector. Using Lemma 2.17, the following theorem is proposed.

Theorem 3.3. *Assume an L-class classification task given by $f^{<t>} : \mathbb{R}^2 \longrightarrow \mathbb{R}^L$ for the activation at time step t such that $f^{<t>}(\boldsymbol{x}) = (W_{ay})^T \sigma(W_{aa}\boldsymbol{a}^{<t-1>} + W_{ax}\boldsymbol{x}^{<t>})$ and $\boldsymbol{x}^{<t>} \in \mathbb{R}^2$. If the planar curvature of the decision boundary at time t is given by $k_r^{<t>}$ and the derivatives are taken w.r.t $\boldsymbol{x}^{<t>}$ then, for all time steps $t, t' \in \mathbb{N}_{\geq 0}, t \neq t'$,*

$$k_r^{<t>} = k_r^{<t'>} \,.$$

Furthermore the conditions on the weights such that the decision boundary is flat $(k_r^{<t>} = 0)$ *for* $i, j \in \{1, \dots, L\}$ *is given by,*

$$W_{ay}^i W_{ax}^{i1} = 0 \quad or \quad W_{ay}^i W_{ax}^{i2} = 0\,,$$
$$W_{ay}^i W_{ay}^j (W_{ax}^{i1} W_{ax}^{j2} - W_{ax}^{i2} W_{ax}^{j1}) = 0\,.$$

where W_{ax}^{i1} *denotes the* $(i, 1)$ *entry of the weight matrix* W_{ax}*,* W_{ay}^i *is the* i*-th entry of the vector* W_{ay} *and so on.*

Proof. The condition on the weights can be obtained in the same way as in p.23 of [18], because the planar curvature $k_r^{<t>}$ is calculated w.r.t. $\boldsymbol{x}^{<t>}$ and $\boldsymbol{a}^{<t-1>}$ is a function of the form $\boldsymbol{a}^{<t-1>} = g(\boldsymbol{x}^{<t-1>}, \boldsymbol{x}^{<t-2>}, \dots, \boldsymbol{x}^{<0>}) \implies \frac{\partial \boldsymbol{a}^{<t-1>}}{\partial x_i^{<t>}} = 0$ for $i \in \{1, 2\}$, which reduces the problem to solving for an ANN architecture. As a consequence, it follows that,

$$\begin{aligned} \frac{\partial f^{<t>}(\boldsymbol{x})}{\partial x_i^{<t>}} &= \frac{\partial}{\partial x_i^{<t>}} \left(W_{ay} \sigma (W_{aa} \boldsymbol{a}^{<t-1>} + W_{ax} \boldsymbol{x}^{<t>}) \right)\,, \\ \frac{\partial f^{<t>}(\boldsymbol{x})}{\partial x_i^{<t>}} &= \frac{\partial}{\partial x_i^{<t>}} \left(W_{ay} \sigma (W_{ax} \boldsymbol{x}^{<t>}) \right)\,. \end{aligned}$$

The last equality uses the fact that the derivatives $\frac{\partial \boldsymbol{a}^{<t-1>}}{\partial x_i^{<t>}}$ vanish accordingly. It is now easy to see that for all $t,\ t' \in \mathbb{N}_{\geq 0},\ t \neq t'$ we have $k_r^{<t>} = k_r^{<t'>}$. Here the term $W_{aa} \boldsymbol{a}^{<t-1>}$ acts as a variable bias. □

The main insight from this theorem is that finding conditions such that the RNN decision boundary is flat is the same as for a fully connected ANN, but as a consequence it comes out naturally that the decision boundary must shift as we iterate over time. This is because the planar curvature $k_r^{<t>}$ at each time step t stays the same. If there exists $t \neq t'$ s.t. the output $f^{<t>} \neq f^{<t'>}$, it must mean that over time the decision boundary shifts. To the best of my knowledge, [4] is the only paper that considers the shifting of the decision boundary but it is not analysed from a theoretical perspective. They propose that the $\boldsymbol{a}^{<t>}$ term is a variable bias which changes the decision of the model based on the order the training data is presented. This also gives an intuitive answer to how sequential models differ from other architectures. The decision of the network depends on the order the input data $\boldsymbol{x}^{<t>} \in \mathbb{R}^d$ at each time step t.

3.3 The Probabilistic Setting, Tubular Neighborhoods and Differential Geometry

In the literature of robustness, random perturbations are considered as a form of attack that have the potential to fool an algorithm. This subsection formulates a theoretical bound on the probability that a randomly perturbed point will land outside the decision boundary, causing

misclassification. Furthermore, it will be shown that under certain circumstances random perturbations can make a classifier more robust to adversarial attacks of a certain size. For the remainder of this section we consider an arbitrary L-class classifier $f : \mathbb{R}^d \longrightarrow \mathbb{R}^L$ and follow a similar setting as [23]. We denote the decision boundary by Σ that divides $\mathbb{R}^d$ into regions $C_k = \{\boldsymbol{x} \in \mathbb{R}^d \mid f_k \geq f_i,\ \forall i \neq k\}$. Consider a datapoint $\boldsymbol{x} \in \mathbb{R}^d$ such that $\boldsymbol{x} \in \mathbb{R}^d \setminus \Sigma$ is not on the decision boundary. We define $\hat{k}(\boldsymbol{x}) = \arg\max_k f_k(\boldsymbol{x})$ to be the correct label corresponding to class k. Assume that $\exists$ a unique closest point for $\boldsymbol{x} \in \mathbb{R}^d \setminus \Sigma$ on Σ and define $\boldsymbol{r}(\boldsymbol{x}) = \arg\min_{\boldsymbol{r}}\{||\boldsymbol{r}|| \mid \boldsymbol{x} + \boldsymbol{r} \in \Sigma\}$ as the closest point from $\boldsymbol{x}$ to the boundary Σ. Furthermore define $\rho > 0$ s.t if $||\boldsymbol{r}(\boldsymbol{x})|| \leq \rho$, then given $\boldsymbol{v} \in \mathbb{B}(\boldsymbol{x}, \rho) \implies \exists \boldsymbol{v} \in \mathbb{B}(\boldsymbol{x}, \rho)$ s.t. $\hat{k}(\boldsymbol{x}+\boldsymbol{v}) \neq \hat{k}(\boldsymbol{x})$. This leads us to a similar definition, as seen in [7].

Definition 3.4 (uniform ε-robustness)**.** Given $\varepsilon \in [0, 1]$, let $\Delta_{unif,\varepsilon} = \max_{\rho \geq 0} \rho$, such that $\mathbb{P}_{v \in \mathbb{B}}(\hat{k}(\boldsymbol{x}+\rho\boldsymbol{v}) \neq \hat{k}(\boldsymbol{x})) \leq \varepsilon$ where $\mathbb{B}$ denotes the unit ball centred at the origin with radius 1 in $\mathbb{R}^d$.

This definition states that $\Delta_{unif,\varepsilon}$ is the radius of the ball centred at $\boldsymbol{x}$ such that perturbed points sampled uniformly will be misclassified with a small probability (ε).

The differential-geometric interpretation aims to provide a new and more fundamental formulation of robustness. We will see that the new insight provided by this interpretation improves on a known regularization method as well as giving a theoretical upper bound on the probability of misclassification for random perturbations. This will be investigated experimentally. The motivation for using Theorem 2.21 is that by bounding the decision boundary with the eigenvalues of the Hessian matrix, the theorem proposes an upper bound on the likelihood that a point perturbed within a sphere will land close, (ε close) to it. This would establish a stronger condition on robustness, as the ε neighbourhood sets a threshold around the decision boundary. If the probability $\mathbb{P}\{\text{dist}(X, \Sigma) \leq \varepsilon\}$ is small then the algorithm can be called robust in the setting where misclassification is allowed. Furthermore, this also implies that if $\mathbb{P}\{\text{dist}(X, \Sigma) \leq \varepsilon\}$ is small, then the perturbed point will land away from the boundary with high probability.

Theorem 3.5. *Assume the result from Theorem 2.21 then,*

$$\mathbb{P}\{\text{dist}(X, \Sigma) \leq \varepsilon\} \leq \frac{2\,\text{vol}(\mathcal{M}')}{\kappa^* \,\text{vol}(\mathbb{B}(\mathbf{0}, 1))\sigma^d d}\left[(1 + \kappa^*\varepsilon)^d - 1\right],$$

where $\kappa^* = \max_i(\kappa_i)$ is the largest eigenvalue of the shape operator or largest sectional curvature at a point in $\mathcal{M}'$. The open subset $\mathcal{M}'$ is such that $\mathcal{M}' \subseteq \Sigma$ and $T(\Sigma, \varepsilon) \cap \mathbb{B}(\boldsymbol{x}, \sigma) \subseteq T(\mathcal{M}', \varepsilon)$ and $\text{vol}(\mathcal{M}')$ is the Lebesgue measure of $\mathcal{M}'$.

Proof. From Theorem 2.21 the following inequality holds,

$$\mathbb{P}\{\text{dist}(X, \Sigma) \leq \varepsilon\} \leq \frac{1}{\sigma^d \,\text{vol}\,\mathbb{B}(\mathbf{0}, 1)} \sum_{i=0}^{d-1} \frac{1}{i+1} |K_i|(\mathcal{M}') \left(\frac{\varepsilon}{\sigma}\right)^{i+1} \tag{3.2}$$

We know from Definition 2.18 that $\sigma_i(\kappa_1, \kappa_2, \ldots, \kappa_d)$ are elementary symmetric polynomials in the eigenvalues of the second fundamental form $\mathbb{I}(\frac{\partial}{\partial x^i}, \frac{\partial}{\partial x^j})$. Suppose $\sigma_i(\kappa_1, \kappa_2, \ldots, \kappa_d) = \sum_{l_1<l_2<\cdots<l_i} \kappa_{l_1} \ldots \kappa_{l_i}$ where $\{1, \ldots, i\}$ is an index set. Let $\kappa^* = \max_i\{\kappa_i\}$. We notice that,

$$\begin{aligned}
\sigma_i(\kappa_1, \ldots, \kappa_d) &\leq \sum_{l_1<l_2<\cdots<l_i} (\kappa^*)^i \,, \\
\sigma_i(\kappa_1, \ldots, \kappa_d) &\leq \binom{d-1}{i} (\kappa^*)^i \,.
\end{aligned}$$

Where for the last inequality to get rid of the summation, we notice that there are $\binom{d-1}{i}$ ways to pick $\kappa_{l_1} \ldots \kappa_{l_i}$. Finally bound $|K_i|(\mathcal{M}')$ using,

$$\begin{aligned}
|K_i|(\mathcal{M}') &= 2\int_{\mathcal{M}'} |\sigma_i(\kappa_1, \kappa_2, \ldots, \kappa_d)| d\mathcal{M} \,, \\
|K_i|(\mathcal{M}') &\leq 2\int_{\mathcal{M}'} \binom{d-1}{i} (\kappa^*)^i \, d\mathcal{M} \,, \\
|K_i|(\mathcal{M}') &\leq 2\,\mathrm{vol}(\mathcal{M}')\binom{d-1}{i} (\kappa^*)^i \,.
\end{aligned}$$

Now, (3.2) can be rewritten as

$$\begin{aligned}
\mathbb{P}\{\mathrm{dist}(X, \mathcal{M}) \leq \varepsilon\} &\leq \frac{2\,\mathrm{vol}(\mathcal{M}')}{\sigma^d \,\mathrm{vol}\,\mathbb{B}(\mathbf{0}, 1)} \sum_{i=0}^{d-1} \frac{1}{i+1}\binom{d-1}{i} (\kappa^*)^i \, \varepsilon^{i+1} \,, \\
\mathbb{P}\{\mathrm{dist}(X, \mathcal{M}) \leq \varepsilon\} &\leq \frac{2\,\mathrm{vol}(\mathcal{M}')}{\kappa^*\sigma^d \,\mathrm{vol}\,\mathbb{B}(\mathbf{0}, 1)} \sum_{i=0}^{d-1} \frac{1}{i+1}\binom{d-1}{i} (\kappa^*\varepsilon)^{i+1} \,,
\end{aligned}$$

We notice that,

$$\frac{1}{i+1}\binom{d-1}{i} = \frac{1}{d}\binom{d}{i+1} .$$

Therefore we can write,

$$\begin{aligned}
\mathbb{P}\{\mathrm{dist}(X, \mathcal{M}) \leq \varepsilon\} &\leq \frac{2\,\mathrm{vol}(\mathcal{M}')}{\kappa^*\sigma^d \,\mathrm{vol}\,\mathbb{B}(\mathbf{0}, 1)} \sum_{i=0}^{d-1} \frac{1}{d}\binom{d}{i+1} (\kappa^*\varepsilon)^{i+1} \,, \\
\mathbb{P}\{\mathrm{dist}(X, \mathcal{M}) \leq \varepsilon\} &\leq \frac{2\,\mathrm{vol}(\mathcal{M}')}{\kappa^*\sigma^d \,\mathrm{vol}\,\mathbb{B}(\mathbf{0}, 1)} \sum_{i=1}^{d} \frac{1}{d}\binom{d}{i} (\kappa^*\varepsilon)^{i} \,, \\
\mathbb{P}\{\mathrm{dist}(X, \mathcal{M}) \leq \varepsilon\} &\leq \frac{2\,\mathrm{vol}(\mathcal{M}')}{\kappa^* \,\mathrm{vol}(\mathbb{B}(\mathbf{0}, 1))\sigma^d d} \left[(1 + \kappa^*\varepsilon)^d - 1\right] \,,
\end{aligned}$$

where $\mathrm{vol}(\mathcal{M}')$ denotes the Lebesgue measure on $\mathcal{M}'$ □

The intuition behind this inequality is that higher curvatures lead to larger eigenvalues which increase the probability that a randomly perturbed point will land ε close to the boundary. This means that if we are in the ε tubular neighborhood of the boundary, the algorithm will be vulnerable to an adversarial attack of size ε. By using hessian regularization to reduce the curvature, the corresponding probability will be smaller as well. This implies that randomly perturbing a point will push the point away from the boundary with a higher probability, therefore

increasing its robustness. This leads to the conclusion that Random perturbations can make classifiers more robust w.r.t adversarial attacks of a certain size. This might seem counter intuitive at first because usually, random perturbations have the potential to fool a classifier. This gives motivation for incorporating random training with Hessian regularization, which will be explored in the Experiments Section.

3.4 Probability of misclassification using Tubular covers

The motivation for the following section is a special case of Theorem 3.5. As before, consider a datapoint $\boldsymbol{p} \in \mathbb{R}^d$ such that $\boldsymbol{p} \in \mathbb{R}^d \setminus \Sigma$ is not on the decision boundary. Define $\hat{k}(\boldsymbol{p}) = \arg\max_k f_k(\boldsymbol{p})$ to be the correct label corresponding to class k. We let the function $g = f_k - f_j$ and use the convention that if $g(\boldsymbol{p}) > 0$ then the point $\boldsymbol{p}$ is assigned class k, otherwise $g(\boldsymbol{p}) < 0$. Set $\boldsymbol{r} = \boldsymbol{r}(\boldsymbol{p})$, $\boldsymbol{z} = \boldsymbol{p} + \boldsymbol{r}$ and $H_{\boldsymbol{z}}(g)$ is the Hessian of the function g evaluated at the decision boundary. Let $\delta = \delta(\boldsymbol{p}) = ||\boldsymbol{r}(\boldsymbol{x})||$ be the adversarial distance to the boundary. Recall that the decision boundary for class k is given by $\Sigma_k = \bigcup_{i \neq k}(C_k \cap C_i)$. We assume there exists an open subset $\mathcal{M}' \subseteq \Sigma_k$ of the decision boundary such that $\boldsymbol{z} \in \mathcal{M}'$, together with

$$T(\Sigma_k, \varepsilon) \cap \mathbb{B}(\boldsymbol{p}, \sigma) \subseteq T(\mathcal{M}'\varepsilon)\,.$$

Theorem 3.6. *Assume the same setting as seen in Theorem 3.5, Let X be uniformly distributed in a ball $\mathbb{B}(\mathbf{0}, \sigma)$ of radius $\sigma > \delta$ around a datapoint $\boldsymbol{p} \in \mathbb{R}^d$. The probability of misclassification is then given by,*

$$\mathbb{P}(\hat{k}(\boldsymbol{p} + X) \neq \hat{k}(\boldsymbol{p})) \leq \frac{\operatorname{vol}(\mathcal{M}')}{\lambda^* \operatorname{vol}(\mathbb{B}(\mathbf{0}, 1))d\sigma^d} \left[(1 + \lambda^*(\sigma - \delta))^d - 1\right]\,,$$

where,

$$\lambda^* = \max_{\boldsymbol{q} \in \mathcal{M}'} \frac{|\lambda_{\max}(H_{\boldsymbol{q}}(g))}{||\nabla g(\boldsymbol{q})||}\,.$$

Proof. Recall that at the point $\boldsymbol{z}$, $g(\boldsymbol{z}) = 0$ and $g(\boldsymbol{p}) > 0$. Notice that the gradient $\nabla g(\boldsymbol{z})$ points away from the boundary. Set $\varepsilon = \sigma - \delta$ and conveniently. we can use the one sided tube $T_+(\mathcal{M}, \varepsilon)$ to obtain the following,

$$\mathbb{P}(\hat{k}(\boldsymbol{p} + X) \neq \hat{k}(\boldsymbol{p})) = \frac{\operatorname{vol}(T_+(\Sigma_k, \varepsilon) \cap \mathbb{B}(\boldsymbol{p}, \sigma))}{\operatorname{vol}(\mathbb{B}(\boldsymbol{p}, \sigma))} \leq \frac{\operatorname{vol}(T_+(\mathcal{M}', \varepsilon))}{\sigma^d \operatorname{vol}(\mathbb{B}(\mathbf{0}, 1))}\,.$$

Let $\boldsymbol{q} \in \mathcal{M}'$. Then there exists a local parameterization around every $\boldsymbol{q}$ given by $\varphi : U \longrightarrow V$ such that $V \subseteq \mathcal{M}'$ together with $\varphi(\mathbf{0}) = \boldsymbol{q}$. Further assume that the coordinates of this local parameterization are normal coordinates, therefore $\frac{\partial \varphi}{\partial x^i}\big|_{\boldsymbol{q}}$ form an orthonormal basis of the tangent space $T_{\boldsymbol{q}}\mathcal{M}$. Using the result of Lemma 2.15 we can write the entries of the shape operator w.r.t. to the chosen coordinates as,

$$\mathbb{I}\left(\frac{\partial}{\partial x^i}, \frac{\partial}{\partial x^j}\right) = \left\langle \frac{\partial^2 \varphi}{\partial x^i \partial x^j}, \frac{\nabla g(\boldsymbol{q})}{||\nabla g(\boldsymbol{q})||} \right\rangle \tag{3.3}$$

We write this in terms of the function g by computing,

$$0 = \frac{\partial^2 g(\varphi(\boldsymbol{x}))}{\partial x^i \partial x^j} = \sum_l \left(\sum_k \frac{\partial^2 g(\boldsymbol{q})}{\partial x_l \partial x_k} \frac{\partial \varphi_l}{\partial x^i} \frac{\partial \varphi_k}{\partial x^j} + \frac{\partial g(\boldsymbol{q})}{\partial x_l} \frac{\partial^2 \varphi_l}{\partial x^i \partial x^j} \right) ,$$
$$= \left\langle \frac{\partial^2 \varphi}{\partial x^i \partial x^j}, \nabla g(\boldsymbol{q}) \right\rangle + \left(\frac{\partial \varphi}{\partial x^i} \right)^T H_{\boldsymbol{p}}(g) \left(\frac{\partial \varphi}{\partial x^j} \right) .$$

After normalising with $||\nabla g(\boldsymbol{q})||$ and rearranging, we can write Eq. (3.3) as,

$$\mathbb{I}\left(\frac{\partial}{\partial x^i}, \frac{\partial}{\partial x^j} \right) = -\frac{1}{||\nabla g(\boldsymbol{q})||} \left(\frac{\partial \varphi}{\partial x^i} \right)^T H_{\boldsymbol{q}}(g) \left(\frac{\partial \varphi}{\partial x^j} \right) .$$

Define $\boldsymbol{A}$ to be the orthogonal matrix with rows $\frac{\partial \varphi}{\partial x^i}$ then we obtain the shape operator as a matrix,

$$S_{\boldsymbol{q}} = -\frac{1}{||\nabla g(\boldsymbol{q})||} \boldsymbol{A}^T H_{\boldsymbol{q}}(g) \boldsymbol{A} .$$

Recall that $\kappa_i(\boldsymbol{q})$ are the eigenvalues of $S_{\boldsymbol{p}}$. To bound $\kappa_i(\boldsymbol{q})$ in absolute values, we have

$$|\kappa_i(\boldsymbol{q})| \leq \frac{1}{||\nabla g(\boldsymbol{q})||} \max_{\boldsymbol{m} \in \mathbb{S}^{d-1}} |\boldsymbol{m}^T \boldsymbol{A}^T H_{\boldsymbol{q}}(g) \boldsymbol{A} \boldsymbol{m}| \leq \lambda^* .$$

Finally, we notice that $\mathrm{vol}(T_+(\mathcal{M}, \varepsilon) = \frac{1}{2} \mathrm{vol}(T(\mathcal{M}, \varepsilon))$ and apply the result of Theorem 3.5 to obtain,

$$\mathbb{P}(\hat{k}(\boldsymbol{p} + X) \neq \hat{k}(\boldsymbol{p})) \leq \frac{\mathrm{vol}(\mathcal{M}')}{\lambda^* \, \mathrm{vol}(\mathbb{B}(\boldsymbol{0}, 1)) d\sigma^d} \left[(1 + \lambda^*(\sigma - \delta))^d - 1 \right] ,$$

□

The statement of the theorem is illustrated by Fig.2.

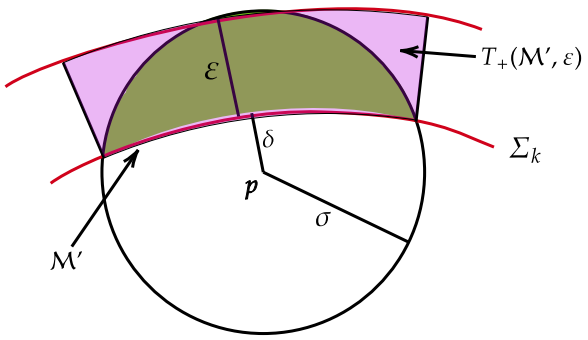

Figure 2: The tube $T_+(\mathcal{M}', \varepsilon)$ around the true decision boundary Σ_k with the point X perturbed in a ball $\mathbb{B}(\boldsymbol{p}, \sigma)$. Probability of misclassification is shown as the green shaded area.

4 Experiments

4.1 Random Hessian regularization

One of the most common defenses against adversarial attacks is adversarial training, as can be seen in [31, 21]. We will perform experiments on one of the most popular dataset, called MNIST [1]. It consists of 10 classes of handwritten digits $(0-9)$ inclusive, where each handwritten image is (28×28) pixels. The model architecture is a multiclass classifier sequence model which is able to correctly classify handwritten digits. The implementation consists of a vanilla RNN network, similar to Definition 2.2. Projected gradient descent or (PGD) is a method of solving a constrained optimization problem. The PGD algorithm can be used to craft adversarial examples. This can be formulated by finding the perturbation $\boldsymbol{\delta}(\boldsymbol{x})$ that maximizes the loss function for a point $\boldsymbol{x}$, while keeping the size of the perturbation $||\boldsymbol{\delta}(\boldsymbol{x})|| \leq \varepsilon$ for some $\varepsilon > 0$.

In the case of the PGD attack, the adversary has full knowledge of the model architecture, therefore it can perturb the datapoints in such a way which will lead to misclassification. This is known as a white-box attack. As Deep Learning is integrated into our modern lives, making sure that these models are secure against targeted attacks is an important area of research. This can be seen in [27], which considers a variety of different attacks and defenses against adversarial attacks, one of the most common being adversarial training. Unfortunately [25] has proposed the idea of Hessian regularization a month before starting my Masters' thesis. Their experimental results indicate that Hessian regularization reduces the curvature of the decision boundary and increases the robustness of classifiers against adversarial manipulations.

One of the interpretations of the theoretical result of Theorem 3.5 motivates the idea of using random training when Hessian regularization has been used. The size of the perturbation for random training is denoted by δ_p. The important thing is that after perturbing the training data the model is trained on the perturbed points, while keeping the output labels the same. Fig. 3 illustrates the data the algorithm produces during the training phase of the RNN. The perturbed image is assigned the same label as the original datapoint, in this case it has class 5. This justifies the choice of δ_p for this experiment, since a human observer can still tell which handwritten digit the perturbed image resembles. The proposed method is tested against a PGD attack and compared with the defenses proposed by [13, 25].

Fig.4 shows the performance of Hessian and Jacobian Regularization against random Hessian regularization. As a baseline, the PGD attack has been performed on a RNN model with no regularization. The parameters for the PGD attack are $\varepsilon \in [1, 30], \alpha = 2$ and it is performed for 30 steps. Note that the parameters are normalised in the range $(0, 1)$. In this specific setting,

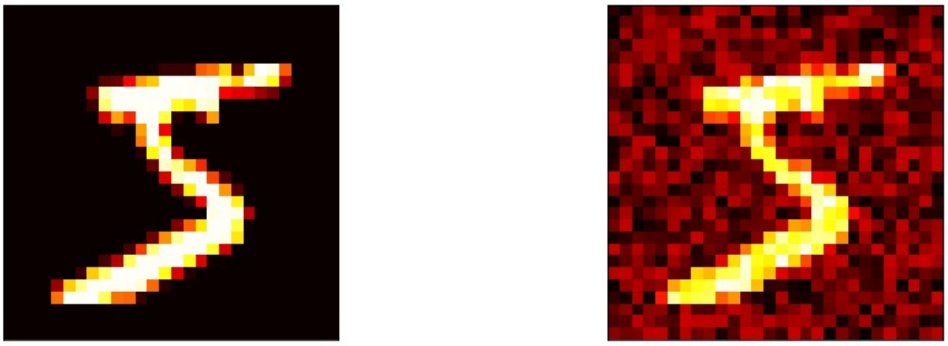

Figure 3: One example of the MNIST handwritten digit dataset before and after a perturbation in the range $(-\delta_p, \delta_p)$ shown on the left and right respectively. For this specific example $\delta_p = 0.2$.

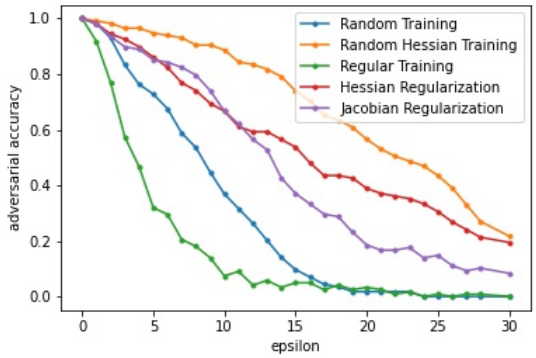

Figure 4: Adversarial accuracy of a 30 step PGD attack for 5 different training methods.

random Hessian regularization outperforms the previous two defenses. It can be seen that the baseline model performs poorly against a PGD attack as ε increases the adversarial accuracy drops to 0. On the other hand, the other 5 training methods have a similar performance for smaller values of ε, but they diverge as it increases. It is important to notice that Random Hessian regularization does not guarantee robustness against all attacks. We observe that random training on its own does not perform very well when compared to the two regularizers. This is because for random training to be effective against adversarial attacks, the curvature of the decision boundary has to be small, and Hessian regularization ensures this. A similar method called Gaussian data augmentation as seen in [28] aims to improve the robustness of NN architectures by adding adversarial Gaussian noise to the training data. This also confirms why the random training on its own performs poorly against a PGD attack. Theorem 3.5 only provides guarantees on robustness against adversarial attacks of a certain size with high probability under lower curvatures. Fig. 5 provides a visual explanation of how Hessian regularization affects the shape and decision boundary. It can be seen that if the Hessian matrix is regularized, the curvature of the boundary

becomes smoother as well. During regular training, clusters of classes seems to form more often and the boundary has a larger curvature.

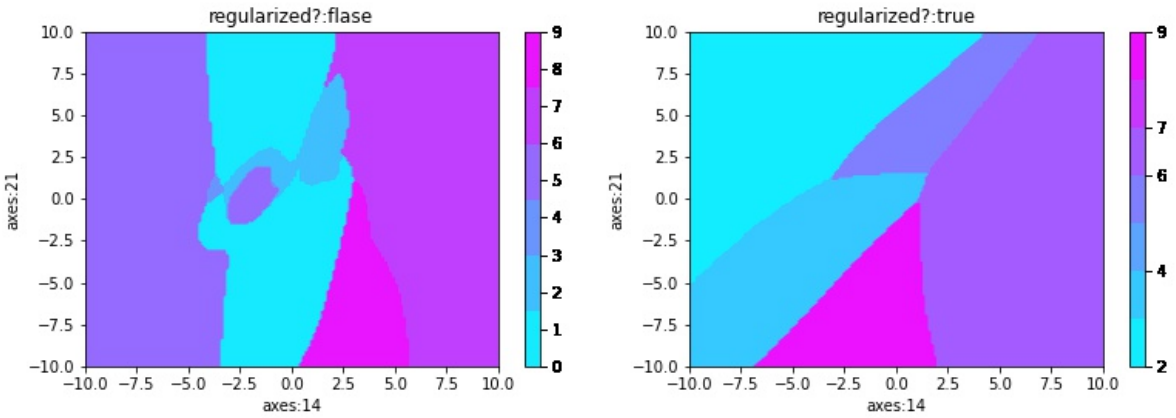

Figure 5: A 2-dimensional slice of the decision boundary for the MNIST data set along 2 randomly picked axes with and without Hessian regularization implemented is shown on the (right) and (left) images respectively.

4.2 Case Study:Exoplanets and Random Hessian Training.

The Kepler Time series data [2] contains the change in flux for stars and the ground truth labels consist of a binary label whether the star has a confirmed exoplanet or not. In comparison with the MNIST dataset, the Kepler dataset provides a more realistic scenario where the robustness of Deep Learning models become important. Machine Learning models are becoming more popular in Astrophysics, as seen in [22, 3], therefore the robustness of these models to corruption cannot be ignored when the algorithm has to make accurate predictions.

We perform the same PGD attack on the RNN classifier, as seen in Subsection 44.1. The RNN classifier is trained using the five different training methods as seen previously and the performance is compared against various attack sizes.

The PGD algorithm is performed over 50 steps, which is sufficient to fool the classifiers when the perturbation radius ϵ is in the range $(0, 10)$. Fig. 6 illustrates that Random Hessian training still outperforms the other 4 methods. In the case of this dataset, random training performs in a similar way to Jacobian and Hessian regularization, unlike the performance on the MNIST dataset. Note that the dataset is not preprocessed, which means the RNN is trained on the raw lightcurve data. The mean value of flux in the dataset is of the order 10^2, therefore it is convenient to set the perturbation size to $\delta_p = \pm 100$. As the value of the perturbation ε increases, the performance of all 5 training methods drop significantly, when $\varepsilon = 6$ the adversarial accuracy goes below 50%.

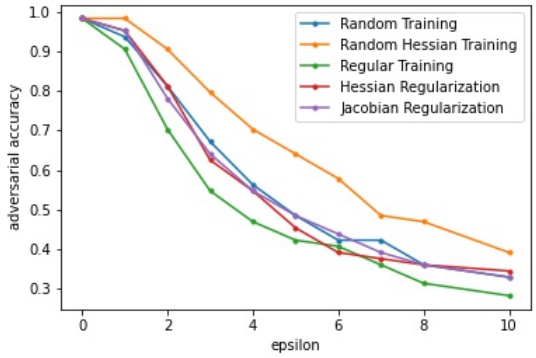

Figure 6: Adversarial accuracy of a 50 step PGD attack for 5 different training methods on the Kepler Time Series data.

4.3 Probability of misclassification

To experimentally verify Theorem 3.6, we set up a simple 1-hidden layer feedforward ANN architecture. The dataset is the blobs dataset, which is illustrated on Fig.7. The classifier attempts to learn the true decision boundary which separates the datapoints. The reason for using a simple feedforward architecture is because it is possible to explicitly calculate the 2nd order approximation to the boundary and evaluate the hessian of the output. To approximate the volume of misclassification, we perform a monte carlo simulation. Pick a radius of perturbation σ, then sample points within the circle and classify them. Calculating the fraction of points outside the boundary gives a good enough approximation for the volume of misclassification. It is clear from Fig.8 that the theoretical bound is close to the Monte Carlo simulation for the blobs dataset. At the perturbation radius $\sigma \approx 0.15$ the two lines cross and the theoretical bound predicts a lower probability of misclassification than the simulation. The decision boundary for this data set is nicely[1] curved. On the other hand, this is not the exactly the case for the moons dataset. Fig.10 illustrates the difference between the theoretical upper bound and the simulation.

Remark 4.1. It is important to note that the bound from Theorem 3.6 should be taken as a theoretical result. This is due to the fact that the quantities $\mathrm{vol}(\mathcal{M}')$ and λ^* are quite difficult to calculate. For illustration purposes however, it is possible to approximate these quantities. It provides an insight into how the probability of misclassification changes as a function of perturbation radius and curvature when compared to a Monte Carlo simulation.

[1] by "nice" we refer to a decision boundary with lower curvatures and no sharp edges or turns.

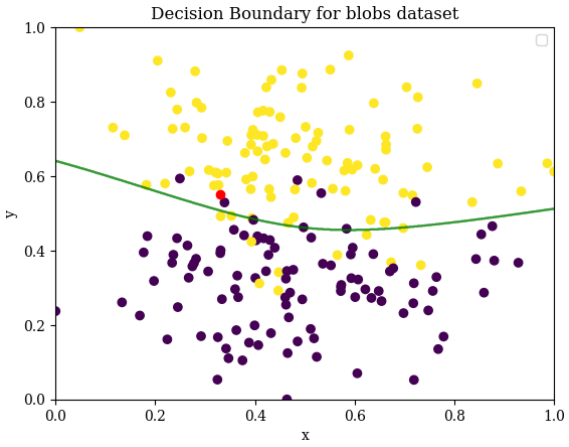

Figure 7: blobs dataset, where the decision boundary is shown in green and the perturbed point $\boldsymbol{x}$ is shown in red.

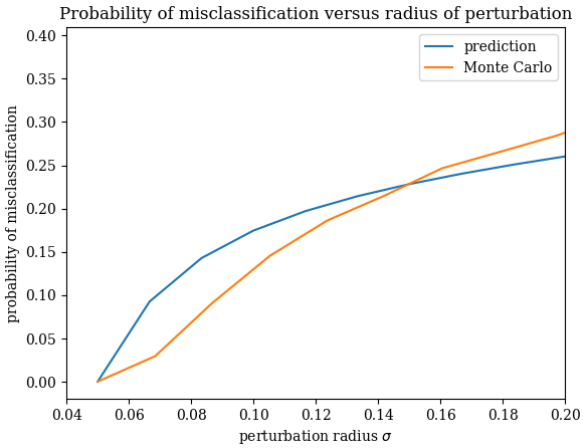

Figure 8: The theoretical probability of misclassification in blue, monte carlo simulation shown in orange.

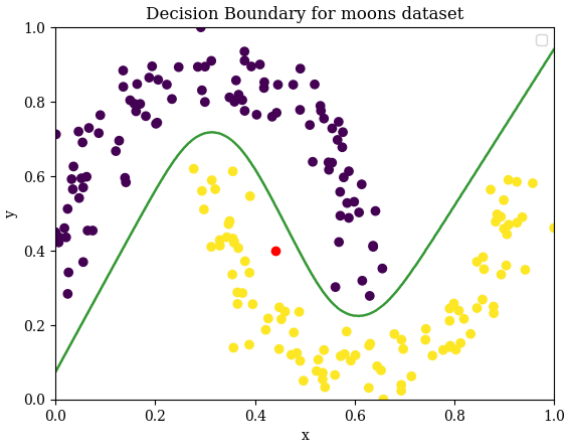

Figure 9: moons dataset, where the decision boundary is shown in green and the perturbed point $\boldsymbol{x}$ is shown in red.

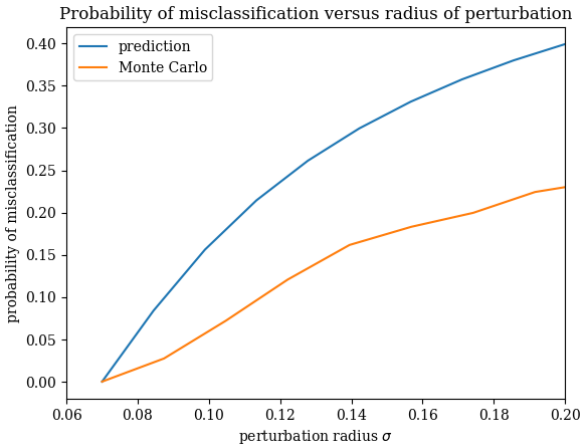

Figure 10: The theoretical probability of misclassification in blue, monte carlo simulation shown in orange.

5 Conclusions

We have seen that the differential-geometric interpretation provides a new and more versatile insight in which robustness can be defined. Specifically, both the curvature and the gradient of the decision boundary appears quite naturally and it plays a key role in determining the robustness of classifiers. Furthermore, it is also present in known defenses such as Jacobian and Hessian regularization as well as in the theoretical formulation of robustness. Moreover, when considering the planar curvature of a sequential model such as an RNN architecture, the shifting of the decision boundary also appears naturally. It provides an insight into how they differ from a regular feed-forward architecture when analysing curvature. This further motivates the development of a more fundamental theory, because different ideas and formulations are in

fact closely related. On one hand, random perturbations can have catastrophic consequences in safety critical systems, but on the other hand they can also defend classifiers against adversarial attacks when the curvature of the decision boundary is lower. This also supports the idea of random Hessian training, as it seems to improve on the performance of Hessian regularization against adversarial attacks of a certain size. It was shown that formulating the probability of misclassification using the idea of tubular neighborhoods leads to an explicit upper bound in terms of the perturbation radius and adversarial distance to the decision boundary. Furthermore, this upper bound directly depends on the eigenvalues of the Hessian matrix and the gradient of the decision boundary, which appear naturally as well. Even though there were experiments performed on the probability of misclassification, the upper bound should still be interpreted as a theoretical result, because the dependence on curvature is a difficult task to compute. I still believe that the differential-geometric interpretation provides a more fundamental insight into the theory of robustness and I hope that it will inspire the future development of new theory to keep classifiers secure against adversarial manipulations.

References

[1] *MNIST handwritten digit dataset*.

[2] Exoplanet Hunting in Deep Space, Kepler labelled time series data.

[3] David J. Armstrong, Jevgenij Gamper, and Theodoros Damoulas. Exoplanet Validation with Machine Learning: 50 new validated Kepler planets, 2020.

[4] T. L. Burrows and M. Niranjan. THE USE OF RECURRENT NEURAL NETWORKS FOR CLASSIFICATION.

[5] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A Generative Model for Music, 2020.

[6] Ke-Lin Du and M. N. S. Swamy. *Multilayer Perceptrons: Architecture and Error Back-propagation*, pages 97–141. Springer London, London, 2019.

[7] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Analysis of classifiers' robustness to adversarial perturbations. *CoRR*, abs/1502.02590, 2015.

[8] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[9] Ian Goodfellow, Patrick McDaniel, and Nicolas Papernot. Making machine learning robust against adversarial inputs. *Communications of the ACM*, 61(7):56–66, 2018.

[10] Alfred Gray. *Differential Geometry of Curves and Surfaces*, page 163. Prentice-Hall, Inc, Englewood Cliffs, New Jersey, 1976.

[11] Alfred Gray. *An Introduction to Weyl's Tube Formula*, pages 1–12. Birkhäuser Basel, Basel, 2004.

[12] Matthias Hein and Maksym Andriushchenko. Formal Guarantees on the Robustness of a Classifier against Adversarial Manipulation. *CoRR*, abs/1705.08475, 2017.

[13] Judy Hoffman, Daniel A. Roberts, and Sho Yaida. Robust Learning with Jacobian Regularization, 2019.

[14] Ye Jia, Yu Zhang, Ron J. Weiss, Quan Wang, Jonathan Shen, Fei Ren, Zhifeng Chen, Patrick Nguyen, Ruoming Pang, Ignacio Lopez Moreno, and Yonghui Wu. Transfer Learning from Speaker Verification to Multispeaker Text-To-Speech Synthesis, 2019.

[15] Klaus Jänich. *Vector Analysis*, pages 1–26. Springer, New York, NY, 2012.

[16] S. Kobayashi and K. Nomizu. *Foundations of differential geometry*, page 16. John Wiley Sons, Inc., 1969.

[17] Sam Levin and Julia Carrie Wong. Self-driving Uber kills Arizona woman in first fatal crash involving pedestria. *The Guardian*, 2018.

[18] Bo Liu. Geometry and Topology of Deep Neural Networks' Decision Boundaries, 2020.

[19] Martin Lotz. On the volume of tubular neighborhoods of real algebraic varieties, 2013.

[20] Martin Lotz. *Mathematics of Machine Learning MA3K1*, pages 143–145. University of Warwick, Department of Mathematics, 2020. Accessed: 13/03/2021.

[21] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards Deep Learning Models Resistant to Adversarial Attacks, 2019.

[22] Abhishek Malik, Ben Moster, and Christian Obermeier. Exoplanet Detection using Machine Learning, 2020.

[23] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, Pascal Frossard, and Stefano Soatto. Robustness of Classifiers to Universal Perturbations: A Geometric Perspective. In *International Conference on Learning Representations*, 2018.

[24] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. DeepFool: a simple and accurate method to fool deep neural networks, 2016.

[25] Waleed Mustafa, Robert A. Vandermeulen, and Marius Kloft. Input Hessian Regularization of Neural Networks, 2020.

[26] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving Rubik's Cube with a Robot Hand, 2019.

[27] Kui Ren, Tianhang Zheng, Zhan Qin, and Xue Liu. Adversarial Attacks and Defenses in Deep Learning. *Engineering*, 6(3):346–360, 2020.

[28] Evgenia Rusak, Lukas Schott, Roland S. Zimmermann, Julian Bitterwolf, Oliver Bringmann, Matthias Bethge, and Wieland Brendel. A simple way to make neural networks robust against diverse image corruptions, 2020.

[29] Harold R. Parks Steven G. Krantz. *The Implicit Function Theorem*, page 36. Birkhäuser, New York, NY, 2013.

[30] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations*, 2014.

[31] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. Ensemble Adversarial Training: Attacks and Defenses, 2020.

[32] Loring W. Tu. *Differential Geometry*, page 30. Springer-Verlag, Berlin, 2017.

[33] Javier Álvarez Vizoso, Michael Kirby, and Chris Peterson. Integral Invariants from Covariance Analysis of Embedded Riemannian Manifolds, 2018.