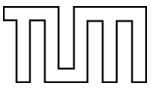

Technische Universität München

Department of Mathematics

Master's Thesis

# Adversarial Deformations for Neural Ordinary Differential Equations

Shpresim Sadiku

Supervisor: Prof. Dr. Michael Wolf

Advisor: Prof. Dr. Michael Wolf

Submission Date: May 11, 2020

*Babait tim*
*To my father*

I assure the single handed composition of this master's thesis only supported by declared resources.

Garching,

**Abstract**

Neural networks have celebrated impressive success in practice in many recognition and classification tasks. Mathematically, yet, their inner workings are poorly understood. Neural network architectures for object classification have been shown to be unstable to the so-called adversarial attacks, achieved by small perturbations of the correctly classified image, imperceptible for the human eye. This questions the usage of neural networks in safety and security critical applications. In this context, the thesis examines classical and modern results in neural networks partitioned into two key fragments.

The first part exploits the algorithmic stability in a general setting while inspecting functions with good stability properties. Next, the emphasis is given to function classes represented by neural networks and their density within different function spaces under various assumptions on the activation function, to then motivate the efficiency of deep neural networks over the shallow neural networks.

In the second part, instead of specifying a discrete sequence of hidden layers, the derivative of the hidden state is parameterized using a neural network. We present a concise optimal control optimization approach to such continuous-depth models by discussing ideas and algorithms derived from the optimality conditions of the powerful Pontryagin's Maximum Principle. The new emerging field of constant memory cost models, however, is vulnerable to adversarial attacks. Apart from highlighting the inconsistency of neural networks theoretically, we experiment with adversarial deformations for neural ordinary differential equations on MNIST and compare our results to convolutional neural network-based architectures.

# Acknowledgments

I express my sincere gratitude to my supervisor, Prof. Michael Wolf, for the guidance and encouragement throughout my work. I am especially grateful for the generous recommendation to elite doctoral programs. Also, I would like to thank Prof. Donna Ankerst and Prof. Felix Krahmer for playing key roles in shaping my path to a PhD in mathematics. I am indebted to my friends for their support at all times.
Faleminderit dy prindërve të mi. Faleminderit, Liri dhe Ilir.

# Notation

## Probability Theory

Expectations with respect to some probability measure $P$ and $P^n$ over a sequence (of $n$ elements) of input-output pairs $S$ will be denoted by $\mathbb{E}[.]$ and $\mathbb{E}_S[.]$, respectively. We denote $\mathbb{E}_{S\sim P^n}$ to emphasize that $S$ is distributed according to $P^n$. $\mathbb{P}_S[A]$ denotes the probability of an event $A$ with respect to $P^n$.

## General

Let $t$ denote a real number, which will be called time. Let $\mathbb{1}_X$ denote the indicator function on a set $X$. For $n \in \mathbb{N}$ let $[n] = \{1, ..., n\}$. A bold lowercase variable, $\mathbf{x}$, denotes a vector which is an element of Euclidean space $\mathbb{R}^n$ for some $1 \leq n < \infty$. The transpose of the vector $\mathbf{x}$ and matrix $\mathbf{X}$ are denoted by $\mathbf{x}', \mathbf{X}'$. The inner product of two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ will be written as $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x} \cdot \mathbf{y} = \mathbf{x}'\mathbf{y}$. We denote by $\|\mathbf{x}\|_2$ the ordinary Euclidean norm of a vector $\mathbf{x} \in \mathbb{R}^n$ . Thus

$$\|\mathbf{x}\|_2 = \left( \sum_{i=1}^{n} (x_i)^2 \right)^{\frac{1}{2}}.$$

In general, denote the $p$-(quasi)norm, i.e. $\ell_p$ norm, of $\mathbf{x} \in \mathbb{R}^n$ for $0 < p < \infty$ by $\|\mathbf{x}\|_p = (\sum_{i=1}^{n} |x_i|^p)^{1/p}$, which is a quasi-norm for $p \in (0, 1)$ and a norm otherwise. The infinity norm of a vector is denoted by $\|\mathbf{x}\|_\infty = \max_{i\in[n]} |x_i|$. Denote by $\|\mathbf{x}\|_0 = \{i \in [n] : x_i \neq 0\}$ the $\ell_0$ norm (even though it is not a norm). $\mathcal{S}^{n-1}$ denotes the surface of the $\ell_2$-unit norm ball in $\mathbb{R}^n$, i.e. $\mathcal{S}^{n-1} = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_2 = 1\}$. Let $B_r^p(\mathbf{x}) = \{\mathbf{y} \in \mathbb{R}^n : \|\mathbf{y} - \mathbf{x}\|_p < r\}$ be the open $\ell_p$-ball of radius $r$ around $\mathbf{x}$ in $\mathbb{R}^n$. Let $\partial K$ and $\text{int}(K)$ denote the boundary and the interior of a set $K \subset \mathbb{R}^n$.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be a differentiable function at $\mathbf{x} \in \mathbb{R}^n$. The gradient of $f$ at $\mathbf{x}$ is a vector-valued function

$$\frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}) = \nabla f(\mathbf{x}) = \begin{bmatrix} \partial_1 f(\mathbf{x}) \\ \vdots \\ \partial_n f(\mathbf{x}) \end{bmatrix} \in \mathbb{R}^n$$

whose components

$$\partial_i f : \mathbb{R}^n \to \mathbb{R}, \quad \mathbf{x} \mapsto \frac{\partial f}{\partial x_i}(\mathbf{x}), \quad \text{for} \quad i \in [n]$$

denote its partial derivatives. As a special case, differentiation with respect to a subset $\mathbf{y}$ of the variables $x_1, ..., x_n$ is denoted by $\nabla_{\mathbf{y}} f(\mathbf{x})$, and contains the partial derivatives of $f$

with respect to the $x_i$'s which are in $\mathbf{y}$.
Let $F = (F_1, ..., F_m) : \mathbb{R}^n \to \mathbb{R}^m$ be a differentiable function at $\mathbf{x} \in \mathbb{R}^n$. The derivative of $F$ at $\mathbf{x}$ is denoted by $\frac{\partial F}{\partial \mathbf{x}}(\mathbf{x})$, or simply $\mathrm{D}_{\mathbf{x}} F$. It is a linear mapping $\mathbb{R}^n \to \mathbb{R}^m$ represented by the Jacobian matrix

$$\begin{bmatrix} \frac{\partial F_1}{\partial \mathbf{x}}(\mathbf{x})' \\ \vdots \\ \frac{\partial F_m}{\partial \mathbf{x}}(\mathbf{x})' \end{bmatrix} = \begin{bmatrix} \partial_1 F_1(\mathbf{x}) & \cdots & \partial_n F_1(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ \partial_1 F_m(\mathbf{x}) & \cdots & \partial_n F_m(\mathbf{x}) \end{bmatrix}.$$

Differentiation with respect to a subset $\mathbf{y}$ of the variables $x_1, ..., x_n$ is denoted by $\frac{\partial F}{\partial \mathbf{y}}(\mathbf{x})$, and contains the partial derivatives of $F$ with respect to the $x_i$'s which are in $\mathbf{y}$. For a function $f$ denote by $\mathrm{supp}(f)$ the set $\mathrm{supp}(f) = \overline{\{\mathbf{x} | f(\mathbf{x}) \neq 0\}}$.

## Function Spaces

$L^p(\nu)$ denotes the space of all measurable functions $f : \mathbb{R}^n \to \mathbb{R}$ such that

$$\|f\|_p = \left( \int_{\mathbb{R}^n} |f(\mathbf{x})|^p d\nu(\mathbf{x}) \right)^{1/p} < \infty.$$

Abbreviate almost everywhere with a.e. Similarly, $L^\infty(X)$ denotes the space of all measurable functions $f : \mathbb{R}^n \to \mathbb{R}$ defined a.e. with respect to Lebesgue measure $\nu$ on a measurable set $X \subset \mathbb{R}^n$, which are essentially bounded on $X$, i.e. $|f(\mathbf{x})|$ is bounded a.e. on $X$. We denote $f \in L^\infty(X)$ with the norm

$$\|f\|_\infty = \inf\{\lambda \mid \nu\{\mathbf{x} : |f(\mathbf{x})| \geq \lambda\} = 0\} = \operatorname*{ess\,sup}_{\mathbf{x} \in X} |f(\mathbf{x})|.$$

$L^\infty_{loc}(\mathbb{R}^n)$ denotes the space of locally essentially bounded functions. A function $f$ defined a.e. with respect to Lebesgue measure on $\mathbb{R}^n$ is said to be locally essentially bounded on $\mathbb{R}^n$, denoted as $f \in L^\infty_{loc}(\mathbb{R}^n)$, if for every compact set $K \subset \mathbb{R}^n$, $f \in L^\infty(K)$. $C(X), C(K)$ denote the space of continuous functions defined on the space $X$ or the compact set $K$ respectively. $C^k(X)$ denotes the space of continuous functions which have continuous partial derivatives up to order $k$. $C^\infty(X)$ denotes the space of smooth functions, that is, continuous functions which have continuous partial derivatives of all orders. Let $C^k_0(X)$ and $C^\infty_0(X)$ denote the subsets of $C^k(X)$ and $C^\infty(X)$ containing functions with compact support.

# Contents

# 1 Introduction

Supervised learning using deep neural networks has tremendously facilitated the progress of modern machine learning applications [GBC16]. Despite the practical success of deep neural networks, we still lack a theoretical framework for understanding them. Moreover, neural network architectures for object classification have been shown to be unstable to the so-called adversarial attacks, achieved by small perturbations of the correctly classified image, imperceptible for the human eye [Sze+13]. A vast number of algorithms have been proposed over the years to efficiently compute perturbations that fool neural networks. Figure (1) shows such an adversarial example.

We start by exploiting the algorithmic stability in a general setting in chapter (2), as we look for functions with good stability properties. The motivation for such an analysis has always been to design algorithms that will not be affected by noise corrupting the inputs. The theory presented on this chapter serves as a basement for setting up the framework for supervised learning and algorithmic stability.

Chapter (3) is devoted to the approximation theory of shallow neural networks and representation benefits of deep neural networks. First, we discuss fundamental results on density of the single hidden layer perceptron model as we show universality of all non-polynomial activation functions. Then we introduce more hidden layers to the neural network and examine recent results which prove that deep neural networks are exponentially more efficient than the shallow ones, in terms of the number of parameters used.

Theoretical efficiency and practical success of deep neural networks point to the need for efficient training in applications. The most commonly applied training technique is stochastic gradient descent [Bot10], where incremental updates to the trainable parameters are performed using gradient information computed via backpropagation [Kel60]. While efficient to implement, the incremental updates to the parameter tend to be slow, especially in the initial stages of the training. Moreover, other than the computation of gradients through backpropagation, the specific structure of deep neural networks is not utilized. In response to such concerns, in chapter (4) we discuss an alternative training approach by exploring the optimal control viewpoint of deep learning, introduced by E [Wei17]. Interpreting residual networks [He+16a] as discretized ordinary differential equations, we consider the first context in which deep neural networks were replaced by

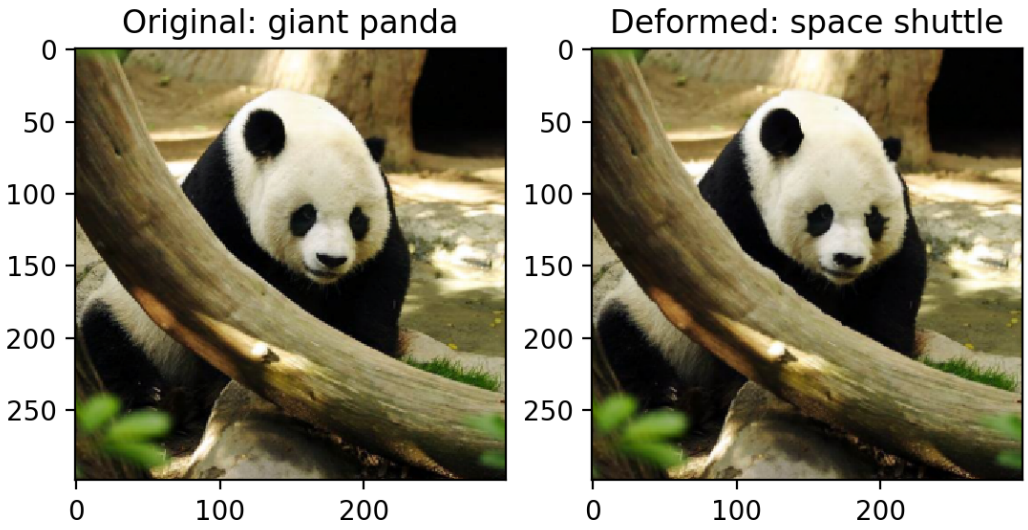


Figure 1: An adversarial example for a pre-trained Inception-v3 model [Sze+16] produced by ADef [AAG18]. The left image, taken from the ILSVRC2012 validation set [Rus+15], is correctly classified as "giant panda" while the image on the right, a slightly perturbed version of the original image, is classified as "space shuttle".

continuous dynamical systems. The focus will be on ideas derived from the optimality conditions of the powerful Pontryagin's Maximum Principle [Pon+62] to find efficient training schemes. We discuss the structure and general properties of control problems with respect to several performance criteria, which provide a buffer between the theoretical material of optimal control and the design problems in deep learning. Then, we examine the recent approach of Neural Ordinary Differential Equations (Neural ODEs) of Chen et al. [Che+18] for training deep learning models.

The new emerging field of continuous-depth models of constant memory cost, however, is vulnerable to adversarial attacks. In this context, chapter (5) is a discussion on stability of Neural ODEs. First, we highlight the inconsistency of neural networks theoretically while discussing the phenomenon of adversarial attacks for object classification problems and then we review the robustness of Neural ODEs with respect to several adversarial attacks. Finally, we experiment with adversarial deformations, achieved by small deformations to the image found through a gradient descent step. We apply such attacks to Neural ODEs on MNIST and compare our results to convolutional neural networks.

# 2 Basic Learning Theory and Algorithmic Stability

In this chapter we discuss classical results in learning theory and algorithmic stability. We begin by setting up the framework for supervised learning in section (2.1), which seeks to find a function based on some given data, and then in section (2.2) we derive three key errors involved in such a task. Concretely, the chapter examines the difference between the error of this function to the true underlying distribution of data and the empirical error when finding the function from the given inputs. In section (2.3) we provide a bound on this difference based on the uniform stability property of the function we want to find. Finally, we study the Tikhonov regularization scheme as a means to guarantee uniform stability.

## 2.1 Statistical framework

Let's start by giving some theoretical background relevant to the analysis of algorithmic stability [Wol18], [SB14], [MRT12]. First, we define a learning algorithm, whose input is a finite sequence of pairs $S$ in $\mathcal{X} \times \mathcal{Y}$, for $\mathcal{X}$ the domain set and $\mathcal{Y}$ the label set.

Output of the learning algorithm is a hypothesis $h : \mathcal{X} \to \mathcal{Y}$, $h \in \mathcal{Y}^{\mathcal{X}}$, assumed to be a Borel function, that aims to predict $y \in \mathcal{Y}$ for arbitrary $x \in \mathcal{X}$. Hence, a learning algorithm can be seen as a map $\mathcal{A} : \bigcup_{n\in\mathbb{N}}(\mathcal{X} \times \mathcal{Y})^n \to \mathcal{Y}^{\mathcal{X}}$, whose range is denoted by $\mathcal{F}$.[1] Data instances are assumed to be identically and independently distributed according to some probability measure $P$ over $\mathcal{X} \times \mathcal{Y}$, and the corresponding $\sigma$-algebra is assumed to be a product of Borel $\sigma$-algebras with respect to the usual topologies.

Goal of the learning algorithm is to find a good hypothesis $h$ with respect to a suitably chosen loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. Common loss function choices include the quadratic loss $L(y, h(x)) = |y - h(x)|^2$ (for continuous $\mathcal{Y}$), or the 0-1 loss $L(y, h(x)) = \mathbb{1}_{y\neq h(x)}$ (for discrete $\mathcal{Y}$). The smaller the average loss, called risk and given by

$$R(h) := \int_{\mathcal{X}\times\mathcal{Y}} L(y, h(x))dP(x, y),$$

the better the hypothesis. But, the problem is that we actually do not know $P$.

[1] Starting from chapter (3) and onwards we will restrict our study to function classes $\mathcal{F}$ represented by neural networks.

## 2.2 Error decomposition

In order to cope with the problem of section (2.1), there are two helping schemes. First, the prior knowledge, for instance hidden in the choice of $\mathcal{F}$ and the way the learning algorithm chooses a hypothesis from this class. Second, we know the performance of the hypothesis on the training data, i.e. we have the empirical risk

$$\hat{R}(h) := \frac{1}{n}\sum_{i=1}^{n} L(y_i, h(x_i)). \tag{2.1}$$

The approach of minimizing $\hat{R}$ is called empirical risk minimization (ERM). [2]

Empirical risk immediately helps us to find bounds for the risk of a hypothesis $h$, $R(h)$. Such bounds include the Clopper-Pearson bound [Wol18, Theorem 1.3], which states that with a high probability (of at least $1-\delta$, for a fixed confidence parameter $\delta \in (0,1]$) over an i.i.d. draw of a test set (i.e. the set of labeled data points not used for training), the risk of the hypothesis $R(h)$ is bounded by the empirical risk evaluated on the test set plus a constant that depends on $\delta$, and the number of test samples.

For a given a distribution $P$ over $\mathcal{X} \times \mathcal{Y}$, the Bayes risk is defined as the infimum of the error achieved over all measurable functions $h : \mathcal{X} \to \mathcal{Y}$, i.e. $R^* = \inf_h R(h)$. Now, let $R_{\mathcal{F}} := \inf_{h\in\mathcal{F}} R(h)$ quantify the optimal performance of a learning algorithm with range $\mathcal{F}$. Assume further that a hypothesis $\hat{h} \in \mathcal{F}$ minimizes the empirical risk, i.e. $\hat{R}(\hat{h}) \leq \hat{R}(h), \forall h \in \mathcal{F}$.

We want a small difference between the risk of a hypothesis $h$ and the optimal Bayes risk, which can be decomposed as

$$R(h) - R^* = \underbrace{(R(h) - R(\hat{h}))}_{\textit{optimization error}} + \underbrace{(R(\hat{h}) - R_{\mathcal{F}})}_{\textit{estimation error}} + \underbrace{(R_{\mathcal{F}} - R^*)}_{\textit{approximation error}}.$$

In subsequent chapters we will make frequent use of these three (error) concepts, which are the key to any good machine learning algorithm. Mathematical fields included in studying such errors include optimization theory, statistics/stochastics and approximation theory.

[2] If $|\mathcal{Y}| < \infty$, then there always exists a minimizer $\hat{h} \in \mathcal{F}$ that attains $\inf_{h\in\mathcal{F}} \hat{R}(h) = \hat{R}(\hat{h})$ since the functions are only evaluated at a finite number of points, which restricts $\mathcal{F}$ to a finite space.

In this chapter, we consider the estimation error, which can be bounded by

$$\begin{aligned} R(\hat{h}) - R_{\mathcal{F}} &= R(\hat{h}) - \hat{R}(\hat{h}) + \hat{R}(\hat{h}) - R(h_{\mathcal{F}}) \\ &\leq R(\hat{h}) - \hat{R}(\hat{h}) + \hat{R}(h_{\mathcal{F}}) - R(h_{\mathcal{F}}) \\ &\leq 2 \sup_{h \in \mathcal{F}} \left| \hat{R}(h) - R(h) \right|. \end{aligned} \tag{2.2}$$

We used $R_{\mathcal{F}} = R(h_{\mathcal{F}})$ in equation (2.2), for some $h_{\mathcal{F}}$ in the function class $\mathcal{F}$. Bounds on difference between the risk and empirical risk are called generalization bounds. They quantify how well the hypothesis generalizes from the observed data to unseen cases.

Taking into account that we are considering random training data, we should account for the possibility of an unfair sample of the underlying distribution. Hence, generalization bounds have to be probabilistic. We can reasonably hope that we obtain guarantees of the form

$$\mathbb{P}_S \left[ \left| \hat{R}(h) - R(h) \right| \geq \epsilon \right] \leq \delta. \tag{2.3}$$

Bounds of the form (2.3) are the center of the Probably Approximately Correct (PAC) learning framework.

Many of the PAC learning bounds then rely on the Hoeffding's inequality (see appendix (A.1)). Generalization bounds deriving from Hoeffding's inequality and union bound for finite function classes $\mathcal{F}$ include [Cas18, pp. 70-71], which states that, with a high probability (greater than $1 - \delta$), the true risk for all $f \in \mathcal{F}$ is bounded by the empirical risk of $f$ plus a constant that depends on $\delta > 0$, the number of training samples $n$, and the size of the class $\mathcal{F}$.

Other generalization bounds are based on different measures of complexity of the function class $\mathcal{F}$, including the growth function (finite label set) and the VC-dimension (binary label set) [Vap98], or covering numbers [Alo+97] to fix the cardinality problem of finite label sets. These bounds of different nature and applied in different situations have in common the fact that they hold uniformly for all the hypotheses in some fixed function class $\mathcal{F}$, i.e. $\mathbb{P}_S \left[ \sup_{h \in \mathcal{F}} \mid \hat{R}(h) - R(h) \mid \geq \epsilon \right] \leq \delta$. But, we may not have a way to describe this function class and assess its size.

In the next section, we will take into account the process by which a learning algorithm chooses a hypothesis and we will see generalization bounds that apply to specific hypotheses with good stability properties. Shalev-Shwartz et al. [Sha+10] define a hypothesis

to be learnable if it admits a consistent learning algorithm. They continue by defining on-average-replace-one-example algorithmic stability, which they prove to be the key necessary and sufficient condition for learnability.

## 2.3 Algorithmic stability

A learning algorithm is said to be stable if slight perturbations in the training data result in small changes of the hypotheses at the output of the algorithm and these changes vanish as the data set grows bigger and bigger [BS13]. A small change in the input typically means the change or omission of a single data point in the training data set. The most common way to quantify changes in the hypothesis is by means of the loss function. Let us recall the definition of uniform stability:

**Definition 2.1** (Uniform Stability)**.** *[BE01, p. 3]*
*Consider a loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$. A learning algorithm that maps $S \in (\mathcal{X} \times \mathcal{Y})^n$ to a hypothesis $h_S$ is said to be uniformly stable with rate $\epsilon : \mathbb{N} \to \mathbb{R}$ if for all $n \in \mathbb{N}, i \in [n], (x, y) \in \mathcal{X} \times \mathcal{Y}$ the following inequality holds for all $S, S' \in (\mathcal{X} \times \mathcal{Y})^n$ that differ in only one element*

$$|L(y, h_S(x)) - L(y, h_{S'}(x))| \leq \epsilon(n). \tag{2.4}$$

The next theorem gives a PAC-type generalization bound based on uniform stability rate.

**Theorem 2.2** (PAC bound from stability)**.** *[BE01, Theorem 2]*
*Consider a loss function with range in $[-c, c]$ and any learning algorithm $S \mapsto h_S$ that is uniformly stable with rate $\epsilon_1 : \mathbb{N} \to \mathbb{R}$. Then the following holds with respect to repeated sampling of training data sets of size $n$. For all $\epsilon > 0$ and all probability measures over $\mathcal{X} \times \mathcal{Y}$*

$$\mathbb{P}_S \left[ \mid \hat{R}(h_S) - R(h_S) \mid \geq \epsilon + \epsilon_1(n) \right] \leq 2\exp\left[ -\frac{n\epsilon^2}{2(n\epsilon_1(n) + c)^2} \right].$$

*Proof.* Consider $\varphi(S) := \hat{R}(h_S) - R(h_S)$ as a function of $n$ i.i.d. random variables. In order to apply McDiarmid's inequality (see appendix (A.1)), we have to bound the expectation of $\varphi(Z)$. We begin with a useful lemma.

**Lemma 2.3.** *[BE01, Lemma 1]*
*For any learning algorithm $\mathcal{A} : S \mapsto h_S$ and any probability measure $P$ on $(\mathcal{X} \times \mathcal{Y})$*

$$\mathbb{E}_S \left[ R(h_S) - \hat{R}(h_S) \right] = \mathbb{E}_S \mathbb{E}_{(x,y)} \mathbb{E}_i \left[ L(y_i, h_{S^i}(x_i)) - L(y_i, h_S(x_i)) \right] \tag{2.5}$$

*where $S = ((x_i, y_i))_{i=1}^n$, $S^i$ is obtained from $S$ by replacing the $i$'th element $(x_i, y_i)$ with $(x, y)$ and $E_i$ denotes the expectation with respect to a uniform distribution $i \in [n]$.*

*Proof.* Note that

$$\mathbb{E}\left[R(h_S)\right] = \mathbb{E}_S\mathbb{E}_{(x,y)}\left[L(y, h_S(x))\right] = \mathbb{E}_S\mathbb{E}_{(x,y)}\left[L(y_i, h_{S^i}(x_i))\right]$$

by the fact that risk function is the average loss and the i.i.d. assumption on $(x_i, y_i)$ and $(x, y)$. Since this holds for all $i$ we may, in addition, take the expectation value $\mathbb{E}_i$ on the right hand side. At the same time, we have

$$\mathbb{E}_S\left[\hat{R}(h_S)\right] = \mathbb{E}_S\mathbb{E}_i\left[L(y_i, h_S(x_i))\right] = \mathbb{E}_S\mathbb{E}_{(x,y)}\mathbb{E}_i\left[L(y_i, h_S(x_i))\right]$$

so that by subtracting the two identities we get equation (2.5). □

Now, using the fact that uniform stability (2.4) implies the so-called on-average stability (as given below) with the same rate, i.e.

$$\mid \mathbb{E}_{S\sim P^n}\mathbb{E}_{(x,y)\sim P}\mathbb{E}_i\left[L(y_i, h_{S^i}(x_i)) - L(y_i, h_S(x_i))\right] \mid \leq \epsilon_1(n)$$

we have from equation (2.5) that $\mid \mathbb{E}\left[\varphi(S)\right] \mid \leq \epsilon_1(n)$. We observe that $\mid \varphi(S) \mid \geq \epsilon + \mid \mathbb{E}\left[\varphi(S)\right] \mid$, which implies $\mid \varphi(S) - \mathbb{E}\left[\varphi(S)\right] \mid \geq \epsilon$. Hence

$$\begin{aligned}\mathbb{P}_S\left[\mid \hat{R}(h_S) - R(h_S) \mid \geq \epsilon + \epsilon_1(n)\right] &\leq \mathbb{P}_S\left[\mid \varphi(S) - \mathbb{E}\left[\varphi(S)\right] \mid \geq \epsilon\right] \\ &\leq 2\exp\left[-\frac{2\epsilon^2}{n\nu^2}\right]\end{aligned}$$

where the second step is McDiarmid's inequality with $\nu$ an upper bound on $\mid \varphi(S) - \varphi(S^i) \mid$ that is yet to be determined. This can be done by again applying the assumed stability to the inequality

$$\begin{aligned}\mid \varphi(S) - \varphi(S^i) \mid &\leq \mid \hat{R}(h_S) - \hat{R}(h_{S^i}) \mid + \mid R(h_S) - R(h_{S^i}) \mid \\ &\leq \frac{1}{n}\sum_{j\neq i} \mid L(y_j, h_S(x_j)) - L(y_j, h_{S^i}(x_j)) \mid + \frac{2c}{n} + \mid R(h_S) - R(h_{S^i}) \mid .\end{aligned}$$

We can bound the sum of the r.h.s by $\epsilon_1(n)$ and, similarly,

$$\mid R(h_S) - R(h_{S^i}) \mid = \mid \mathbb{E}_{(X,Y)}\left[L(Y, h_S(X)) - L(Y, h_{S^i}(X))\right] \mid \leq \epsilon_1(n).$$

The claim then follows with $\nu := 2(\epsilon_1(n) + c/n)$. □

In the remaining part, let's analyze the use of regularization as a means to guarantee uniform stability, where a single hypothesis class $\mathcal{F}$ is chosen together with a regularizer, i.e. a complexity penalizing function $\varrho : \mathcal{F} \to \mathbb{R}_+$, and one minimizes the regularized empirical risk $\hat{R}(h) + \varrho(h)$. If $\mathcal{F}$ is embedded in a normed space, a common scheme is

Tikhonov regularization, where $\varrho(h) := \|Ah\|^2$ for some linear map $A$, which is often simply a multiple of the identity. The remaining free parameter is then chosen for instance by cross-validation [Koh+95].

A learning algorithm $S \mapsto h_S$ is said to suffer from overfitting if the difference between the true risk of its output $R(h_S)$, and the empirical risk of its output $\hat{R}(h_S)$, is large. Lemma (2.3) tells us that a learning algorithm does not overfit iff it is on-average stable. Of course, a learning algorithm that does not overfit is not necessarily a good learning algorithm - take, for example, an algorithm $\mathcal{A}$ that always outputs the same hypothesis. A useful algorithm should find a hypothesis that on one hand fits the training set (i.e. has a low empirical risk) and on the other hand does not overfit. Hence, the algorithm should both fit the training set and at the same time be stable. The parameter $\lambda$ of the Tikhonov functional in theorem (2.6) achieves this balance.

To continue, we need the following notion from convex optimization:

**Definition 2.4** (Strong convexity)**.** *Let $\mathcal{F}$ be a convex subset of a real inner product space and $\alpha > 0$ a real number. A function $\Phi : \mathcal{F} \to \mathbb{R}$ is called $\alpha$-strongly convex, if for all $g, h \in \mathcal{F}$ and $\alpha \in [0, 1]$ we have*

$$\lambda\Phi(h) + (1-\lambda)\Phi(g) \geq \Phi(\lambda h + (1-\lambda)g) + \frac{\alpha}{2}\lambda(1-\lambda)\|h-g\|_2^2.$$

In order to prove uniform stability from regularization, we need the following lemma:

**Lemma 2.5.** *[SB14, Lemma 13.5]*
*If $\Phi : \mathcal{F} \to \mathbb{R}$ is $\alpha$-strongly convex and $g$ is a minimizer of $\Phi$, then, for any $h \in \mathcal{F}$*

$$\Phi(h) \geq \Phi(g) + \frac{\alpha}{2}\|h-g\|_2^2.$$

*Proof.* We start by dividing the definition of strong convexity by $\lambda$ and rearrange terms to get that

$$\frac{\Phi(g+\lambda(h-g)) - \Phi(g)}{\lambda} \leq \Phi(h) - \Phi(g) - \frac{\alpha}{2}(1-\lambda)\|h-g\|_2^2.$$

Taking the limit $\lambda \to 0$ we obtain that the right hand side converges to $\Phi(h) - \Phi(g) - \frac{\alpha}{2}\|h-g\|_2^2$. Conversely, the left-hand side becomes the derivative of the function $\varphi(\lambda) = \Phi(g+\lambda(h-g))$ at $\lambda = 0$. Since $g$ is a minimizer of $\Phi$, it follows that $\lambda = 0$ is a minimizer of $\varphi$, and therefore the left-hand side of the preceding goes to zero in the limit $\alpha \to 0$. □

**Theorem 2.6** (Uniform stability from regularization)**.** *[Wol18, Theorem 1.23]*
*Let $\lambda > 0$ and $\mathcal{F} \subseteq \mathcal{Y}^{\mathcal{X}}$ be a convex subset of an inner product space. If for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$ the map $h \mapsto L(y, h(x))$ is convex and $l$–Lipschitz on $\mathcal{F}$, then the learning algorithm that minimizes Tikhonov functional $f_S(h) := \hat{R}(h) + \lambda\langle h, h\rangle$ is uniformly stable with rate $\frac{2l^2}{\lambda n}$.*

*Proof.* As $f_S$ is $2\lambda$-strongly convex (as a sum of an average convex loss and a parabola with parameter $\lambda$ [SB14, Lemma 13.5]), and $h$ minimizes $f_S$, we can exploit lemma (2.5) to obtain

$$\lambda\|h^{'} - h\|_2^2 \leq f_S(h^{'}) - f_S(h). \tag{2.6}$$

On the other hand, with $h := h_S, h^{'} := h_{S^i}$ and the norm being the one induced by the inner product we can write

$$\begin{aligned} f_S(h^{'}) - f_S(h) &= \hat{R}_S(h^{'}) - \hat{R}_S(h) + \lambda(\|h^{'}\|_2^2 - \|h\|_2^2) \\ &= \hat{R}_{S^i}(h^{'}) - \hat{R}_{S^i}(h) + \lambda(\|h^{'}\|_2^2 - \|h\|_2^2) \\ &+ \frac{1}{n}\left[L(y_i, h^{'}(x_i)) - L(y_i, h(x_i)) + L(y, h(x)) - L(y, h^{'}(x))\right]. \end{aligned} \tag{2.7}$$

Using the fact that $h^{'}$ minimizes $f_{S^i}$, in equation (2.7), we obtain that

$$f_S(h^{'}) - f_S(h) \leq \frac{1}{n}\left[L(y_i, h^{'}(x_i)) - L(y_i, h(x_i)) + L(y, h(x)) - L(y, h^{'}(x))\right]. \tag{2.8}$$

Combining relations (2.8) and (2.6) we obtain that

$$\lambda\|h^{'} - h\|_2 \leq \frac{1}{n}\left[L(y_i, h^{'}(x_i)) - L(y_i, h(x_i)) + L(y, h(x)) - L(y, h^{'}(x))\right]. \tag{2.9}$$

Since the loss function, $L(y, h(x))$, is $l$-Lipschitz, then by definition of Lipschitzness

$$L(y_i, h^{'}(x_i)) - L(y_i, h(x_i)) \leq l\|h_{S^i} - h_S\|_2.$$

Similarly,

$$L(y, h(x)) - L(y, h^{'}(x)) \leq l\|h_{S^i} - h_S\|_2. \tag{2.10}$$

Plugging these inequalities into (2.9) leads to $\|h^{'} - h\|_2 \leq 2l/(\lambda n)$. Plugging the preceding back into equation (2.10) we conclude that

$$|\, L(y, h(x)) - L(y, h^{'}(x))\, | \leq l\|h - h^{'}\|_2 \leq \frac{2l^2}{\lambda n}.$$

□

Let's now rewrite the expected risk of a learning algorithm $S \mapsto h_S$ as

$$\mathbb{E}_S\left[R(h_S)\right] = \mathbb{E}_S\left[\hat{R}(h_S)\right] + \mathbb{E}_S\left[R(h_S) - \hat{R}(h_S)\right] \tag{2.11}$$

The first term reflects how well $h_S$ fits the training set while the second term reflects the difference between the true and empirical risks of $h_S$. As it was shown in lemma (2.3), the second term is equivalent to the stability of the learning algorithm $\mathcal{A}$. Since our goal is to minimize the risk of the algorithm, we need that the sum of both terms will be small. From lemma (2.3) and theorem (2.6) we have that

$$\mathbb{E}_{S\sim P^n}\left[R(h_S) - \hat{R}(h_S)\right] \leq \frac{2l^2}{\lambda n} \tag{2.12}$$

which shows that the second term in the equation (2.11) decreases as the regularization parameter $\lambda$ increases. At the same time, the empirical risk increases with $\lambda$. We therefore face a trade-off between fitting and overfitting. The following corollary aims at formalizing this trade-off:

**Corollary 2.7** (Regularization trade-off)**.** *[Wol18, Corollary 1.29]*
*Let $\lambda$ and $\mathcal{F} \subseteq \mathcal{Y}^X$ be a convex subset of an inner product space. Assume that for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$ the map $h \mapsto L(y, h(x))$ is convex and $l$-Lipschitz on $\mathcal{F}$ and define $h^* := \arg\min_{h\in\mathcal{F}} R(h)$. Then the learning algorithm $S \mapsto h_S$ that minimizes the functional $f_S(h) := \hat{R}(h) + \lambda\|h\|_2^2$ satisfies*

$$\mathbb{E}_S\left[R(h_S)\right] \leq R(h^*) + \lambda\|h^*\|_2^2 + \frac{2l^2}{\lambda n}. \tag{2.13}$$

*Proof.* Since $h_S$ minimizes $f_S$, we have

$$\mathbb{E}_S\left[\hat{R}(h_S)\right] \leq \mathbb{E}_S\left[f_S(h_S)\right] \leq \mathbb{E}_S\left[f_S(h^*)\right] = R(h^*) + \lambda\|h^*\|_2^2.$$

Substituting this bound and the bound (2.12) in equation (2.11) leads to the claimed result

$$\mathbb{E}_S\left[\hat{R}(h_S)\right] \leq R(h^*) + \lambda\|h^*\|_2^2 + \frac{2l^2}{\lambda n}.$$

□

The optimal value for $\lambda$ that minimizes the right hand side of equation (2.13) is then

$$\lambda_{opt} = \frac{l}{\|h^*\|_2}\sqrt{\frac{2}{n}} \implies \mathbb{E}_S\left[R(h_S)\right] \leq R(h^*) + 2l\|h^*\|_2\sqrt{\frac{2}{n}}.$$

This tells us that, in expectation, the risk converges asymptotically to the optimum. However, in practice, the norm $\|h^*\|_2$ is not known, but one can try to estimate it, for example with the help of a validation set (the set which is used to give an estimate of the learning algorithm while tuning its free parameters), and then use the estimate instead.

# 3 On Neural Networks

Supervised learning (2) using deep neural networks has tremendously facilitated the progress of modern machine learning applications [GBC16]. Starting from this chapter we focus on function classes $\mathcal{F}$ represented by neural networks as we examine the approximation theory of neural networks and representation benefits of deep neural networks. Concretely, in section (3.1) we discuss classical and fundamental results on the density of the single hidden layer perceptron model, then in section (3.2) we introduce more hidden layers to the neural network and examine recent results which prove that deep neural networks are exponentially more efficient than the shallow ones in terms of the number of parameters used.

## 3.1 Approximation Theory

In this section on approximation theory of the single hidden layer perceptron model we provide a comprehensive proof on universality of all non-polynomial activation functions. This result embeds, as special cases, almost all the previous results that were reported on density literature. However, we note that density does not imply an efficient scheme for approximation. Moreover, there is a lower bound on the degree to which one can approximate using single hidden layer neural networks, which is independent of the activation function. Nevertheless, density is a natural starting point for the analysis of neural networks and a lack of density precludes any approximation scheme from being in the least useful.

### 3.1.1 Density in $C(X)$

Let's start by considering density questions associated with the set of single hidden layer neural networks, or the so-called single hidden layer perceptron model. Concretely, we consider the set

$$\sum_n = \text{span}\{\sigma(\mathbf{w} \cdot \mathbf{x} + \vartheta) : \mathbf{w} \in \mathbb{R}^n, \vartheta \in \mathbb{R}\} \tag{3.1}$$

with univariate activation function $\sigma : \mathbb{R} \to \mathbb{R}$, vector of weights $\mathbf{w} \in \mathbb{R}^n$ and threshold value $\vartheta \in \mathbb{R}$. The central focus of this section is to find for which $\sigma$ it is true that, for any $f \in C(\mathbb{R}^n)$, any compact subset $K$ of $\mathbb{R}^n$, and any $\epsilon > 0$, there exists a $g \in \sum_n$ such that

$$\sup_{\mathbf{x} \in K} |f(\mathbf{x}) - g(\mathbf{x})| < \epsilon.$$

In other words, we want to know for which $\sigma$ we have the density of the linear space $\sum_n$ in the space of continuous functions $C(\mathbb{R}^n)$, with respect to the uniform norm $\|f\|_{K,\infty} = \sup_{x\in K} |f(\mathbf{x})|$ for $f \in C(K)$ for any compact set $K$ of $\mathbb{R}^n$.

Density is a first step before finding efficient schemes for approximation. A lack of density means that it is impossible to approximate a large class of functions, and this excludes any scheme based thereon from being in the least useful. That's why we settle for density as opposed to exact representations in this section. However, from a practical perspective it should be noted that the density of $\sum_n$ in $C(\mathbb{R}^n)$ does not guarantee that one can approximate well to every function $f \in C(\mathbb{R}^n)$ from

$$\left\{ \sum_{i=1}^{r} c_i \sigma(\mathbf{w}_i \cdot \mathbf{x} + \vartheta_i) : c_i, \vartheta_i \in \mathbb{R}, \mathbf{w}_i \in \mathbb{R}^n \right\}, \tag{3.2}$$

i.e. the set of single hidden layer neural networks restricted to have not more than $r$ units in the hidden layer, for some fixed $r$. Moreover, Maiorov et al. [MP99] give a lower bound (for any reasonable set of functions) on the degree to which one can approximate using the set (3.2), independent of the choice of $\sigma$.

A variety of techniques using slightly different assumptions on the activation function were used to prove density results of the single hidden neural networks in the space of continuous functions. Cybenko [Cyb89] proves density in the space of continuous functions with support in the unit hypercube, i.e. $C([0,1]^n)$, for any continuous sigmoidal activation function. The term sigmoidal is used for the class of activation functions $\sigma : \mathbb{R} \to \mathbb{R}$ satisfying $\lim_{t\to-\infty} \sigma(t) = 0$ and $\lim_{t\to\infty} \sigma(t) = 1$ [Cyb89, p. 306]. The main building blocks in his proof are the Hahn-Banach Theorem, the Riesz Representation Theorem and the Lebesgue Bounded Convergence Theorem. Funahashi [Fun89] proves density in $C(K)$, for any non-constant, bounded and monotonously increasing continuous activation function $\sigma$. Similar results were proved by Hornik et al. [HSW+89], who show the density for monotonic sigmoidal activation functions and potentially discontinuous at countably many points. They obtain density by first applying the Stone-Weierstrass Theorem to the sums and products of activation functions. Then they prove the desired result by using the fact that the products of cosine functions can be approximated arbitrarily well on a bounded set by linear combinations of cosine functions [GW88].

However, most of the above mentioned results on proving that the set of single hidden neural networks can approximate any continuous function to any degree of accuracy as-

sume some degree of continuity on the activation function $\sigma$. Leshno et al. [Les+93] prove that we can actually drop the continuous assumption and impose minimal conditions on the activation function. Namely, only non-polynomiality of $\sigma$ is required. This also embeds, as special cases, almost all the activation functions that were reported on density literature prior to this result. Note that the density results for discontinuous activation functions is more of theoretical interest, as the commonly used activation functions in practice (for instance sigmoid function, ReLU) are continuous.

### 3.1.2 Density in $C(K)$ with discontinuous activation functions

For the following theorem on proving the density in $C(K)$ for possibly discontinuous activation functions $\sigma$, denote by $L^{\infty}_{loc}(\mathbb{R})$ the locally essentially bounded functions on $\mathbb{R}$. Leshno et al. [Les+93] then consider the set of functions $M$ which are in $L^{\infty}_{loc}(\mathbb{R})$ and have the property that the closure of the set of points of discontinuity is of zero Lebesgue measure.

**Theorem 3.1** (Universality of all non-polynomial activation functions)**.** *[Les+93, Theorem 1]*
*Let $\sigma \in M$. Then $\sum_n$ is dense in $C(\mathbb{R}^n)$ iff $\sigma$ is not a polynomial (a.e.).*
*As a simple consequence of this theorem and by [VK61], we can restrict the weights* $\mathbf{w}$ *to lie in some $W \subseteq \mathbb{R}^n$. $W$ must have the property that no non-trivial homogeneous polynomial vanishes on $W$.*

*Proof.* It is immediate that density cannot hold if $\sigma$ is a polynomial, as $\sum_n$ would consist of polynomials of a fixed degree which are not dense in $C(\mathbb{R}^n)$.
For the converse result, we need to combine some more engaged steps presented in the form of following lemmas.
The first lemma enables us to reduce our analysis from $\mathbb{R}^n$ to the more tractable univariate $\mathbb{R}$.

**Lemma 3.2** (Dimensionality reduction)**.** *[Les+93, p. 4, Step 2]*
*If $\sum_1$ is dense in $C(\mathbb{R})$, then $\sum_n$ is dense in $C(\mathbb{R}^n)$.*

*Proof.* By [VK61] the space of the so-called ridge functions given by

$$\operatorname{span}\{g(\mathbf{w} \cdot \mathbf{x}) | \mathbf{w} \in \mathbb{R}^n, g \in C(\mathbb{R})\} \tag{3.3}$$

is dense in $C(\mathbb{R}^n)$. Now, let $f \in C(K)$ for some compact set $K \subset \mathbb{R}^n$. Since the space

(3.3) is dense in $C(K)$, given $\epsilon > 0$ there exist $g_i \in C(\mathbb{R})$ and $\mathbf{w}_i \in \mathbb{R}^n, i \in [r]$, such that

$$\left| f(\mathbf{x}) - \sum_{i=1}^{r} g_i(\mathbf{w}_i \cdot \mathbf{x}) \right| < \frac{\epsilon}{2}$$

for all $\mathbf{x} \in K$. Since $K$ is compact, $\{\mathbf{w}_i \cdot \mathbf{x} : \mathbf{x} \in K\} \subseteq [\alpha_i, \beta_i]$ for some finite interval $[\alpha_i, \beta_i], i \in [r]$. Because $\sum_1$ is dense in $C([\alpha_i, \beta_i]), i \in [r]$, there exist constants $c_{ij}, w_{ij}, \vartheta_{ij} \in \mathbb{R}, j \in [m_i], i \in [r]$, such that

$$\left| g_i(y) - \sum_{j=1}^{m_i} c_{ij}\sigma(w_{ij}y + \vartheta_{ij}) \right| < \frac{\epsilon}{2r}$$

for all $y \in [\alpha_i, \beta_i]$. Thus

$$\begin{aligned} \left| f(\mathbf{x}) - \sum_{i=1}^{r}\sum_{j=1}^{m_i} c_{ij}\sigma(w_{ij}(\mathbf{w}_i \cdot \mathbf{x}) + \vartheta_{ij}) \right| &< \left| f(\mathbf{x}) - \sum_{i=1}^{r} g_i(\mathbf{w}_i \cdot \mathbf{x}) \right| \\ &+ \sum_{i=1}^{r} \left| g_i(y) - \sum_{j=1}^{m_i} c_{ij}\sigma(w_{ij}y + \vartheta_{ij}) \right| \\ &< \epsilon \end{aligned}$$

for all $\mathbf{x} \in K$. □

Using the homogeneity of the directions of weights $\mathbf{w}$ and by [VK61], Pinkus [Pin99, p. 155] proves that we can restrict $\mathbf{w}$ to lie in $\mu \times W$ for some $\mu \in \mathbb{R}$ and $W \subseteq \mathcal{S}^{n-1}$, where $W$ has the property that no non-trivial homogeneous polynomial vanishes on $W$. The next proposition then proves density for the restricted class of smooth activation functions.

**Proposition 3.3** (Density for smooth $\sigma$)**.** *[Les+93, p.4, Step 3]*
*If $\sigma \in C^\infty$ is not a polynomial, then $\sum_1$ is dense in $C(\mathbb{R})$.*

*Proof.* Since $\sigma \in C^\infty$, and $[\sigma((w+h)x + \vartheta) - \sigma(wx + \vartheta)]/h \in \sum_1$ for every $w, \vartheta \in \mathbb{R}$ and $h \neq 0$, it follows that $(d/dw)\sigma(wx + \vartheta)$ is contained in the closure of $\sum_1$, $\overline{\sum_1}$. Similarly, $(d^k/dw^k)\sigma(wx + \vartheta) \in \overline{\sum_1}$ for all $k \in \mathbb{N}$ and all $w, \vartheta \in \mathbb{R}$. But $(d^k/dw^k)\sigma(wx + \vartheta) = x^k\sigma^{(k)}(wx+\vartheta)$, where $\sigma^{(k)}$ denotes the $k$th derivative of $\sigma$. Now, since $\sigma$ is not a polynomial there exists a $\vartheta_k$ such that $\sigma^{(k)}(\vartheta_k) \neq 0$ (this is a non-trivial consequence of Baire's Category Theorem, cf. [BB96]). Hence $\overline{\sum_1}$ contains all monomials

$$x^k\sigma^{(k)}(\vartheta_k) = \frac{\partial^k}{\partial w^k}\sigma(wx + \vartheta)\bigg|_{w=0,\vartheta=\vartheta_k},$$

and thus all polynomials. By the Weierstrass Approximation Theorem it follows that $\sum_1$ is dense in $C(\mathbb{R})$. □

In a more general form, Proposition 3.3 can be restated by requiring $w$ to lie in a set containing a sequence of values tending to zero and $\vartheta$ to lie on an open interval on which $\sigma$ is not a polynomial [Pin99, Corollary 3.5].

The following lemma gives a very important step on generalizing from $\sigma \in C^\infty$ to $\sigma \in M$, where $M$ contains locally essentially bounded activation functions that have the property that the closure of the set of points of discontinuity is of zero Lebesgue measure (that is, are Riemann-integrable).

**Lemma 3.4** (Dropping the smoothness assumption on $\sigma$)**.** *[Les+93, p.4, Step 4]*
*For each $\sigma \in M$ not a polynomial and $\varphi \in C_0^\infty$ ($C^\infty$ with compact support), the convolution $\sigma * \varphi \in \overline{\sum_1}$.*

The proof continues in a constructive manner by showing that we can approximate the convolution

$$(\sigma * \varphi)(x) = \int \sigma(x - y)\varphi(y)dy$$

uniformly by

$$\sum_{i=1}^{m} \sigma(x - y_i)\varphi(y_i)\Delta y_i$$

on any compact set $[-\alpha, \alpha]$, such that $\operatorname{supp} \varphi \subseteq [-\alpha, \alpha]$, where $y_i = -\alpha + 2i\alpha/m$ and $\Delta y_i = 2\alpha/m$ for $i \in [m]$ [Les+93, p.4, Step 4].

From lemma (3.4) it follows that $\sigma * \varphi \in \overline{\sum_1}$. Thus,

$$(\sigma * \varphi)(wx + \vartheta) = \int \sigma(wx + \vartheta - y)\varphi(y)dy$$

is also in $\overline{\sum_1}$, for each $w, \vartheta \in \mathbb{R}$. Now, for $\sigma \in M$ not a polynomial and any $\varphi \in C_0^\infty$, we have that $\sigma * \varphi \in C^\infty$ [AF03]. Hence, from proposition (3.3), if $\sigma * \varphi$ is not a polynomial then $\sum_1$ is dense in $C(\mathbb{R})$.
The remaining part of the proof is done by contradiction, that is, by assuming that $\sigma * \varphi$ is a polynomial for all $\varphi \in C_0^\infty$ and concluding from this fact that $\sigma$ is itself a polynomial (a.e.).

**Lemma 3.5.** *[Les+93, pp. 5-6, Steps 6, 7]*
*For each $\sigma \in M$ not a polynomial and $\varphi \in C_0^\infty$ assume that the convolution $\sigma * \varphi$ is a polynomial, then there exists an $m \in \mathbb{N}$ such that $\sigma * \varphi$ is a polynomial of degree at most $m$ for all $\varphi \in C_0^\infty$. As a consequence, $\sigma$ is also a polynomial of degree at most $m$ (a.e.).*

The proof starts by considering the set of all $C_0^\infty$ functions with support in $[a,b]$ and the fact that $C_0^\infty([a,b])$ is a complete metric vector space - Fréchet space (see appendix (A.2)). By applying Baire's Category Theorem and noting that $\{\varphi \in C_0^\infty([a,b])|\ \text{degree}(\sigma * \varphi) \leq k\}$ for $k \in \mathbb{N}$ is a vector space, there exists an $m \in \mathbb{N}$ such that

$$C_0^\infty([a,b]) = \{\varphi \in C_0^\infty([a,b])|\ \text{degree}(\sigma * \varphi) \leq m\}\,.$$

For the general case of $\varphi \in C_0^\infty(\mathbb{R})$, the proof that $\text{degree}(\sigma * \varphi) \leq m$ follows by simply noting that the number $m$ does not depend on the interval $[a,b]$, but it depends at most on the length of the interval.
To prove the second part of the lemma, we use the fact that there exists a sequence $(\varphi_n)_{n\in\mathbb{N}} \in C_0^\infty(\mathbb{R})$ such that the convolution $\sigma * \varphi_n$ converges to $\sigma$ in $L^p$ for $1 \leq p < \infty$. For $\varphi_n$ we can take, for example, mollifiers [AF03]. Thus, if $\sigma * \varphi_n$ is a polynomial of degree at most $m$ so is $\sigma$ (a.e.). This contradicts our assumption. □

**Remark 3.6.** *[Les+93, Remark 4]*
*If we were to consider continuous activation functions, $\sigma \in C(\mathbb{R})$, we could complete the proof of theorem (3.1) immediately after proposition (3.3) by applying the theory of mean-periodic functions [Sch47]. A consequence is that if $\sigma \in C(\mathbb{R})$ is not a polynomial then the closure of $\text{span}\{\sigma(x+\vartheta) : \vartheta \in \mathbb{R}\}$ contains a function of the form $e^{\lambda x}\cos(\nu x)$ for some $\lambda, \nu \in \mathbb{R}\backslash\{0\}$. The proof then follows by noting that any such function is in $C^\infty$.*

Similarly to the proof of lemma (3.5), Pinkus [Pin99, Proposition 3.8] extends the result on density in $C(\mathbb{R}^n)$ for discontinuous activation functions. Namely, he proves that density holds for any $\sigma$ that is bounded and Riemann-integrable on every finite interval.

Until now, we have seen density in $C(\mathbb{R}^n)$. But, if $\mu$ is any non-negative finite Borel measure, with support in some compact set $K$, then $C(K)$ is dense in the space of $p$th power integrable functions $L^p(\mu)$ defined on the compact set $K$ for any $1 \leq p < \infty$. Hence, we can extend these results to these spaces as well.

**Theorem 3.7** (Density in $L^p(\mu)$)**.** *[Les+93, Proposition 1]*
*Assume $\mu$ is a non-negative finite measure on $\mathbb{R}^n$ with compact support which is absolutely continuous with respect to the Lebesgue measure. Then $\sum_n$ is dense in $L^p(\mu), 1 \leq p < \infty$, iff $\sigma$ is not a polynomial (a.e.).*

*Proof.* Clearly, if $\sigma$ is a polynomial, then $\sum_n$ is contained in the set of polynomials of bounded degree. Hence, density in $L^p(\mu)$ for $1 \leq p < \infty$ cannot possibly hold.

For the converse result, we let $K$ denote the support of $\mu$. By [AF03], $C(K)$ is dense in $L^p(\mu)$. Thus, given $f \in L^p(\mu)$ and $\epsilon > 0$ there exists a $g \in C(K)$ such that

$$\|f - g\|_p \leq \epsilon/2.$$

Now, since $\sum_n$ is dense in $C(K)$ in the uniform norm, for this given $g \in C(K)$ there exists an $h \in \sum_n$, such that

$$\|g - h\|_\infty \leq \frac{\epsilon}{2c}$$

where $c = \mu^{1/p}(K)$. Thus $\|g - h\|_p \leq \epsilon/2$, and

$$\|f - h\|_p \leq \|f - g\|_p + \|g - h\|_p \leq \epsilon.$$

□

## 3.2 Exponential Benefits of Depth in Neural Networks

In this section, we allow for neural networks to have more than one hidden layer as we consider the so-called deep neural networks. For many years, relatively little was known and many authors saw little theoretical gain in considering more than one hidden layer. Viewing deep neural networks as a generalization of shallow neural networks, it is natural to think that also deep neural networks satisfy the density. Then there arises the question on why deep neural networks are so widespread and successful in practice.

The first results in this direction can be seen by exploiting the advantages of the two hidden layer neural networks as opposed to the single hidden layer ones when it comes to the order of approximation. There is a lower bound on the degree to which the set of single layer neural networks with $r$ units in the hidden layer can approximate any function.[3] It is given by the extent to which a linear combination of $r$ ridge functions (3.3) can approximate this function [Mai99]. Thus, it is natural to think on why it would be in the least useful to consider the set of single hidden layer neural networks. Pinkus et al. [MP99, Proposition 1], [Pin99, Corollary 6.4] show that the lower bound for this set can be attained using activation functions that are in $C^\infty$, sigmoidal and strictly increasing.

While there is a lower bound on the degree of approximation for the single hidden layer model which depends on the number of units used, this is not the case in the two hidden

[3] From the results of previous section (3.1), it can be easily derived that the perfect approximation to any function is achieved if the number of units in the hidden layer tends to infinity [Pin99, p. 167]. This in turn implies no bound on the width of the network.

layer model. Maiorov et al. [MP99] show that there is no theoretical lower bound on the error of approximation if we permit two hidden layers, with a $\sigma$ which is analytic (not only $C^\infty$), strictly increasing and sigmoidal. We mention a special case of this result.

**Theorem 3.8.** *[Pin99, Theorem 7.1]*
*There exists an activation function $\sigma$ which is $C^\infty$, strictly increasing and sigmoidal, and has the following property. For any $f \in C([0,1]^n)$ and $\epsilon > 0$, there exist real constants $d_i, c_{ij}, \vartheta_{ij}, \gamma_i$, and vectors $\mathbf{w}_{ij} \in \mathbb{R}^n$ for which*

$$\left| f(\mathbf{x}) - \sum_{i=1}^{4n+3} d_i \sigma \left( \sum_{j=1}^{2n+1} c_{ij} \sigma(\mathbf{w}_{ij} \cdot \mathbf{x} + \vartheta_{ij}) + \gamma_i \right) \right| < \epsilon,$$

*for all $\mathbf{x} \in [0,1]^n$.*

The proof makes use of an improved version of the Kolmogorov Superposition Theorem (3.9). Kolmogorov disproved Hilbert's 13th conjecture that a solution of the general equation of degree seven cannot be expressed as a finite superposition of continuous functions of two variables by showing that every continuous multivariate function can be represented as a finite superposition of continuous functions of only one variable.

**Theorem 3.9** (The Kolmogorov Superposition Theorem)**.** *[LGM96]*
*For every $n \in \mathbb{N}$ there exist strictly increasing functions $\phi_i \in C([0,1]), i \in [2n+1]$ and constants $\lambda_j, j \in [n], \sum_{j=1}^n \lambda_j \leq 1$ such that for every $f \in C([0,1]^n)$ there exists $g \in C([0,1])$ so that*

$$f(x_1, ..., x_n) = \sum_{i=1}^{2n+1} g \left( \sum_{j=1}^{n} \lambda_j \phi_i(x_j) \right). \tag{3.4}$$

We note that the theorem (3.9) is about representing and not approximating functions. From the viewpoint of neural networks, equation (3.4) can be interpreted as a two hidden layer neural network where the first hidden layer contains $n(2n+1)$ units, which use the $\phi_i$'s as activation functions, the second hidden layer contains $2n+1$ units with activation function $g$, and the output unit uses a linear activation function $\sigma(x) = x$. Hence, equation (3.4) provides an exact representation using only finitely many units, but at the cost of having an activation function $g$ that depends on $f$.
The proof of theorem (3.8), apart from applying theorem (3.9), uses an activation function $\sigma$ which has the desired properties (it is in $C^\infty$, strictly increasing and sigmoidal), but is artificially constructed and is only of theoretical interest, so we abstain from presenting the proof. Nonetheless, from an approximation-theoretic point of view, this theorem allows us to conjecture that the two hidden layer model may be significantly more promising

than the single hidden layer model.

In the remaining part of this section, let's focus on the reverse approach to the one discussed in section (3.1). Namely, we will try to identify a countable family of functions for which a good approximation is possible. Telgarsky [Tel15], [Tel16] in his works identifies functions which facilitate an exponential order of approximation with shallow neural networks, linear order with deep neural networks and constant order with recurrent neural networks [RHW86]. We note that in no sense this family of functions will satisfy the density property as in the previous section (3.1).

First, we generalize the concept of a neural network as a function whose evaluation is defined by a graph in the following way. Root nodes, or units, compute $x \mapsto \sigma(\mathbf{w} \cdot \mathbf{x} + \vartheta)$ with $\sigma, \mathbf{w}, \vartheta$ defined as before in (3.1). Internal nodes' input vector is the collective output of their parents. The choices of $\vartheta$ and $\mathbf{w}$ may vary from node to node, and the set of functions obtained by varying these parameters gives the function class of neural networks $\mathcal{F}(\sigma; m, l)$ which has $l$ layers each with at most $m$ units.
We restrict our discussion to classification problems. That is, given a function $f : \mathbb{R}^n \to \mathbb{R}$ we define $\tilde{f} : \mathbb{R}^n \to \{0, 1\}$ with $\tilde{f} := \mathbb{1}_{f(\mathbf{x}) \geq 1/2}$. Then, given a sequence of points $((\mathbf{x}_i, y_i))_{i=1}^n$, define $\hat{R}(f) = 1/n \sum_{i=1}^n \mathbb{1}_{y_i \neq \tilde{f}(\mathbf{x}_i)}$ (2.1). If we focus on functions defined on the univariate $\mathbb{R}$, we have the following theorem.

**Theorem 3.10.** *[Tel15, Theorem 1.1]*
*Let positive integer $k$, number of layers $l$, and number of units per layer $m$ be given with $m \leq 2^{(k-3)/l-1}$. Then there exists a collection of $n := 2^k$ points $((x_i, y_i))_{i=1}^n$ with $x_i \in [0, 1]$ and $y_i \in \{0, 1\}$ such that*

*i)* $\min_{f \in \mathcal{F}(\mathrm{ReLU}; 2, 2k)} \hat{R}(f) = 0.$

*ii)* $\min_{g \in \mathcal{F}(\mathrm{ReLU}; m, l)} \hat{R}(g) \geq 1/6.$

We have denoted by ReLU [NH10] the non-linear activation function given by

$$\mathrm{ReLU}(x) = \max(0, x).$$

This result provides a glimpse on why and when deep neural networks are more efficient than the shallow ones in terms of the number of parameters used. If we fix a particular function, we see that we cannot approximate the function well unless the number of units per layer is exponential (in $k$), whereas for a deeper network with the same activation function, the total number of units required is linear.

Consequently, Telgarsky [Tel15] makes three refinements: the classification problem is specified to be the one given by the $n$-ap (3.11), the network in condition *i)* of theorem (3.10) will be an even simpler recurrent network, and the ReLU activation function in condition *ii)* will be substituted by a general class of piecewise affine activation functions (3.12). These refinements will eventually lead to theorem (3.13).

**Definition 3.11** ($n$-ap)**.** *[Tel15, pp. 1-2]*
*Let $n$-ap (the $n$-alternating-point problem) denote the set of $n := 2^k$ uniformly spaced points within $[0, 1 - 2^{-k}]$ with alternating labels, as depicted in figure (2), that is, points $((x_i, y_i))_{i=0}^{2^k-1}$ with $x_i = i2^{-k}$ and $y_i = 0$ when $i$ is even, and otherwise $y_i = 1$.*

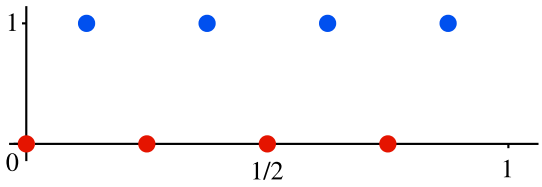


Figure 2: This figure has been reproduced from [Tel15, Figure 1] and depicts the $2^3$-ap.

Let $\mathcal{F}(\sigma; m, l; k)$ denote $k$ iterations of a recurrent network with $l$ layers of at most $m$ units each, such that every $f \in \mathcal{F}(\sigma; m, l; k)$ consists of some fixed network $g \in \mathcal{F}(\sigma; m, l)$ applied $k$ times

$$f(x) = g^k(x) = (\underbrace{g \circ \cdots \circ g}_{k \text{ times}})(x).$$

Consequently, $\mathcal{F}(\sigma; m, l; k) \subseteq \mathcal{F}(\sigma; m, lk)$ [Tel15], but the former has $\mathcal{O}(ml)$ parameters whereas the latter has $\mathcal{O}(mlk)$ parameters.

**Definition 3.12** ($t$-sawtooth activation function)**.** *[Tel15, p. 2]*
*$\sigma : \mathbb{R} \to \mathbb{R}$ is $t$-sawtooth if it is piecewise affine with $t$ pieces.*

**Theorem 3.13.** *[Tel15, Theorem 1.2]*
*Let positive integer $k$, number of layers $l$, and number of units per layer $m$ be given. Given a $t$-sawtooth $\sigma : \mathbb{R} \to \mathbb{R}$ and $n := 2^k$ points as specified by the $n$-ap, then*

*i)* $\min_{f \in \mathcal{F}(\mathrm{ReLU}; 2, 2; k)} \hat{R}(f) = 0.$

*ii)* $\min_{g \in \mathcal{F}(\sigma; m, l)} \hat{R}(g) \geq (n - 4(tm)^l)/(3n).$

In order to prove theorem (3.13) we need to take the following steps which are instructive in themselves and are presented as lemmas. Theorem (3.10) will then be a simple consequence of theorem (3.13).

**Lemma 3.14.** *[Tel15, Lemma 2.3]*
*Let $f, g : \mathbb{R} \to \mathbb{R}$ be respectively $k$- and $l$-sawtooth. Then $f + g$ is $(k + l)$-sawtooth, and $f \circ g$ is $kl$-sawtooth.*

Consider a partition $\mathcal{I}_f$, respectively $\mathcal{I}_g$, of $\mathbb{R}$ corresponding to $f$, respectively $g$, on each of which they are piecewise affine with $k$, respectively $l$, pieces, and note that adding together sawtooth functions grows the number of regions very slowly, whereas composition grows the number of regions very quickly, an early sign of the benefits of depth.

**Lemma 3.15.** *[Tel15, Lemma 2.1]*
*If $\sigma$ is $t$-sawtooth, then every $f \in \mathcal{F}(\sigma; m, l)$ with $f : \mathbb{R} \to \mathbb{R}$ is $(tm)^l$-sawtooth.*

*Proof.* Argue by induction over the layers, that is, prove that the output of each unit in layer $i$ is $(tm)^i$-sawtooth as a function of the neural network input. For the first layer this is obvious, since each unit computes $x \mapsto \sigma(wx + \vartheta)$, which by lemma (3.14) is $t$-sawtooth, since $wx + \vartheta$ is itself affine. Consequently, the input to layer $i$ with $i > 1$ is a collection of functions $(g_1, ..., g_{m'})$ with $m' \leq m$ and $g_j$ being $(tm)^{i-1}$-sawtooth by inductive hypothesis. Then, by applying lemma (3.14) twice, $x \mapsto \sigma(\sum_j w_j g_j(x) + \vartheta)$ is $(tm)^i$-sawtooth. □

**Proposition 3.16.** *[Tel15, Lemma 2.2]*
*Let $((x_i, y_i))_{i=1}^n$ be given according to the $n$-ap Then every $t$-sawtooth function $f : \mathbb{R} \to \mathbb{R}$ satisfies $\hat{R}(f) \geq (n - 4t)/(3n)$.*

*Proof.* Since $f$ is piecewise monotonic with a corresponding partition $\mathbb{R}$ having at most $t$ pieces, then $f$ has at most $2t - 1$ crossings of $1/2$, that is, at most one within each interval of the partition, and at most 1 at the right endpoint of all but the last interval. Thereafter, $\tilde{f}$ is piecewise constant, where the corresponding partition of $\mathbb{R}$ is into at most $2t$ intervals. Thus, $n$ points with alternating labels of $n$-ap must lie in $2t$ intervals, hence the total number of points lying in intervals with at least three points is at least $n - 4t$. Since signs must alternate within any such interval, at least a third of the points in any of these intervals are labeled incorrectly by $\tilde{f}$. □

Proposition (3.16) and lemma (3.15) immediately enable us to deduce condition *ii)* of theorem (3.13), which in turn implies condition *ii)* of theorem (3.10). Indeed, by noting that ReLU is 2-sawtooth and $\mathcal{F}(\text{ReLU}; 2, 2; k) \subseteq \mathcal{F}(\text{ReLU}; 2, 2k)$, the condition $m \leq 2^{(k-3)/l-1}$ implies

$$\frac{n - 4(2m)^l}{3n} = \frac{1}{3} - (2m)^l 2^{-k} \left(\frac{4}{3}\right) \geq \frac{1}{3} - 2^{k-3} 2^{-k} \left(\frac{4}{3}\right) = \frac{1}{6}.$$

For proving condition $i$) of theorem (3.13), Telgarsky starts by constructing a function $f_m : \mathbb{R} \to \mathbb{R}$, depicted in figure (3), and defined as

$$f_m(x) := \begin{cases} 2x & \text{for } 0 \leq x \leq 1/2 \\ 2(1-x) & \text{for } 1/2 < x \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

and noting that $f_m \in \mathcal{F}(\text{ReLU}; 2, 2)$ since, for example, $f_m(x) = \text{ReLU}(2\text{ReLU}(x) - 4\text{ReLU}(x-1/2))$. Applying $f_m(x)$ to itself leads to $f_m^2$ and $f_m^3$ in figure (3). Note that peaks and troughs match the $2^2$-ap and $2^3$-ap. In a summarized approach, these compositions may be written as follows.

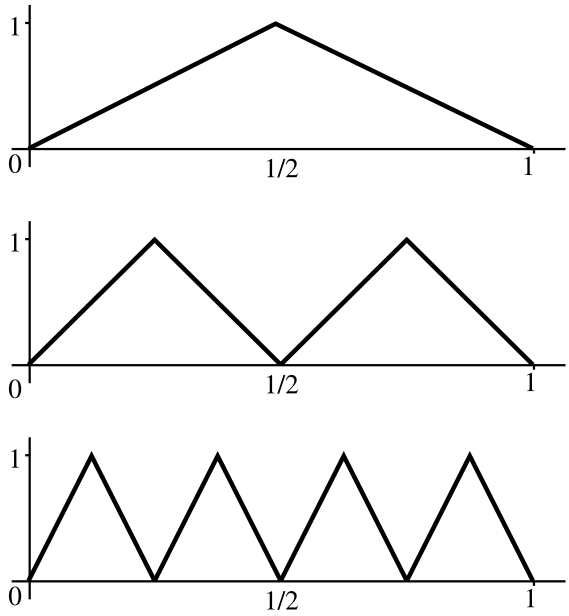


Figure 3: This figure has been reproduced from [Tel15, Figure 2] and depicts $f_m$, $f_m^2$, and $f_m^3$.

**Lemma 3.17.** *[Tel15, Lemma 2.4]*
*Let real $x \in [0, 1]$ and positive integer $k$ be given, and choose the unique non-negative integer $i_k \in \{0, ..., 2^{k-1}\}$ and real $x_k \in [0, 1)$ so that $x = (i_k + x_k)2^{1-k}$. Then*

$$f_m^k(x) = \begin{cases} 2x_k & \text{for } 0 \leq x_k \leq 1/2 \\ 2(1-x_k) & \text{for } 1/2 < x_k < 1. \end{cases}$$

*Proof.* Argue by induction on the number of compositions $l$. For $l = 1$, the result is immediate. For the inductive step, we make use of the mirroring property of pre-composition

with $f_m$ and the symmetry of $f_m^l$ (inductive step), which imply that for every $x \in [0, 1/2]$

$$(f_m^l \circ f_m)(x) = (f_m^l \circ f_m)(1 - x) = (f_m^l \circ f_m)(x + 1/2).$$

Thus, it suffices to consider $x \in [0, 1/2]$, which by the mirroring property means $(f_m^l \circ f_m)(x) = f_m^l(2x)$. Now, the unique non-negative integer $i_{l+1}$ and real $x_{l+1} \in [0, 1)$ satisfy $2x = 2(i_{l+1} + x_{l+1})2^{-l-1} = (i_{l+1} + x_{l+1})2^{-l}$. Applying the inductive hypothesis to $2x$ gives

$$(f_m^l \circ f_m)(x) = f_m^l(2x) = \begin{cases} 2x_{l+1} & \text{for } 0 \leq x_{l+1} \leq 1/2 \\ 2(1 - x_{l+1}) & \text{for } 1/2 < x_{l+1} < 1. \end{cases}$$

□

Condition $i)$ of theorem (3.13) then follows by noting that $f_m^k \in \mathcal{F}(\text{ReLU}; 2, 2; k) \subseteq \mathcal{F}(\text{ReLU}; 2, 2k)$ by construction. Moreover, by lemma (3.17) $f_m^k(x_i) = \tilde{f}_m^k(x_i) = y_i$ on every $(x_i, y_i)$ in the $n$-ap. This also implies condition $i)$ of theorem (3.10).
The more refined result of theorem (3.13) can say, for example, that on the $2^k$-ap one needs exponentially (in $k$) many parameters with linear combinations of decision trees [DC96], linearly many parameters with a deep network and constantly many parameters with a recurrent network [Tel15].

# 4 Neural Ordinary Differential Equations

Motivated by theoretical efficiency (3) and practical success of deep neural networks, in this chapter we exploit residual networks and examine efficient training schemes for practical applications. The most commonly used training technique is stochastic gradient descent [Bot10], where incremental updates to the trainable parameters are performed using gradient information computed via backpropagation [Kel60]. While efficient to implement, the incremental updates to the parameter tend to be slow, especially in the initial stages of the training. Moreover, other than the computation of gradients through backpropagation, the specific structure of deep neural networks is not utilized. In response to such concerns, in this chapter we discuss an alternative training approach by exploring the optimal control viewpoint of deep learning, introduced by E [Wei17].

We start by interpreting residual networks [He+16a] as discretized ordinary differential equations and consider the first context in which deep neural networks were replaced by continuous dynamical systems. In order to provide efficient training schemes, we focus on ideas and algorithms derived from the optimality conditions of the powerful Pontryagin's Maximum Principle [Pon+62] and its variants. We discuss the structure and general properties of control problems with respect to several performance criteria, which provide a buffer between the theoretical material of optimal control and the design problems in deep learning. Next, as a special case, we examine the recent approach of Neural Ordinary Differential Equations (Neural ODEs) of Chen et al. [Che+18] for training deep learning models, to then study their approximation capabilities towards the end of the chapter.

## 4.1 From Deep Residual Networks to Neural Ordinary Differential Equations

Deep residual networks, introduced by He et al. [He+16a], have emerged as a family of extremely deep architectures showing compelling accuracy and good convergence rates in image recognition amongst other tasks. In a general form, such models build complicated transformations by composing a sequence of transformations to a hidden state

$$\begin{aligned} \mathbf{y}_t &= e(\mathbf{x}_t) + f(\mathbf{x}_t, \theta_t) \\ \mathbf{x}_{t+1} &= g(\mathbf{y}_t) \end{aligned} \tag{4.1}$$

where $t \in \{t_0, ..., t_N\}$; $\mathbf{x}_t,\ \mathbf{x}_{t+1} \in \mathbb{R}^n$ are the input and output of the hidden state at layer $t$ which depend on some parameters $\theta_t \in \mathbb{R}^m$, $f : \mathbb{R}^n \to \mathbb{R}^n$ is some differentiable

function which preserves the dimension of $\mathbf{x}_t$ (usually a convolutional neural network - CNN [LKF10]), $e$ and $g$ are some mappings which could in principle be non-linear. From this general form, we see that deep residual networks are described in form of a discrete dynamical system [Wei17].

In particular, He et al. [He+16a] consider the case when $e(\mathbf{x}_t) = \mathbf{x}_t$ is an identity mapping and $g$ is a ReLU function. However, a main result of [He+16b], found through numerous numerical experiments, shows that for very deep networks (hundreds of thousands of layers), training is the easiest if both $g$ and $e$ are the identity map. We refer to [Wei17, pp. 5-6] for a dynamical systems viewpoint to such a phenomenon. Now, if we let $g$ and $e$ both be identity maps, system (4.1) becomes

$$\mathbf{x}_{t+1} = \mathbf{x}_t + f(\mathbf{x}_t, \theta_t). \tag{4.2}$$

Not only in practice, the new family of deep residual networks promise good approximation capabilities in theory as well. Lin et al. [LJ18] demonstrate that a very deep residual network[4] with hidden states that have one unit per hidden layer and ReLU activation functions can uniformly approximate any Lebesgue-integrable[5] function in $n$ dimensions, i.e. $L^1(\mathbb{R}^n)$. Concretely, Lin et al. show that for any Lebesgue-integrable function $h : \mathbb{R}^n \to \mathbb{R}$, for any $\epsilon > 0$, there exists a residual network $R$ such that

$$\int_{\mathbb{R}^n} |h(\mathbf{x}) - R(\mathbf{x})| d\mathbf{x} \leq \epsilon.$$

The residual network in their construction is of the following form:

$$R(\mathbf{x}) = \mathcal{L} \circ (\mathrm{Id} + \mathcal{T}_N) \circ (\mathrm{Id} + \mathcal{T}_{N-1}) \circ \cdots \circ (\mathrm{Id} + \mathcal{T}_0)(\mathbf{x})$$

where $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ is a linear operator and $\mathcal{T}_i$ are basic one-unit residual blocks, i.e. functions $\mathcal{T}_{U,V,u}$ from $\mathbb{R}^n$ to $\mathbb{R}^n$ defined by

$$\mathcal{T}_{U,V,u}(\mathbf{x}) = V\mathrm{ReLU}(U\mathbf{x} + u)$$

where $U \in \mathbb{R}^{1\times n}, V \in \mathbb{R}^{n\times 1}, u \in \mathbb{R}$.

A significant work has been done recently to underpin the relationship between neural networks and differential equations [Lu+17], [HR17], [Li+17], [RH18]. In particular, the

[4]Note that the result of Lin et al. [LJ18] does not give a bound on the depth of the network, as opposed to the results of chapter (3).

[5]A function $h : \mathbb{R}^n \to \mathbb{R}$ is Lebesgue-integrable if $\int_{\mathbb{R}^n} \mid h(\mathbf{x}) \mid d\mathbf{x} < \infty$.

iterative updates of residual networks (4.2) can be seen as an Euler discretization of a continuous transformation [Lu+17], [HR17], [RH18], i.e. the difference $\mathbf{x}_{t+1} - \mathbf{x}_t$ can be interpreted as a discretization of the derivative $d\mathbf{x}(t)/dt$ with timestep $\Delta t = 1$. Letting $\Delta t \to 0$ gives rise to a family of models called Neural Ordinary Differential Equations (Neural ODEs) [Che+18], where the continuous dynamics of hidden units is parameterized by an ordinary differential equation (ODE) specified by a neural network

$$\lim_{\Delta t \to 0} \frac{\mathbf{x}_{t+\Delta t} - \mathbf{x}_t}{\Delta t} = \frac{d\mathbf{x}(t)}{dt} = f(\mathbf{x}(t), \theta, t)$$

or, simply,

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \theta, t)^{6}.$$

Starting from the input layer $\mathbf{x}(t_0)$, we can define the output layer $\mathbf{x}(t_N)$ to be the solution to this ODE initial value problem at some time $t_N$. The hidden state at time $t_N$, i.e. $\mathbf{x}(t_N)$, corresponds to the features learned by the model. In other words, in Neural ODEs we map $\mathbf{x}(t_0)$ to an output $\mathbf{x}(t_N)$ by solving an ODE starting from $\mathbf{x}(t_0)$. We denote this map by $\phi$. We can then adjust the dynamics of the system (encoded by $f$) such that the ODE transforms $\mathbf{x}(t_0)$ to a $\mathbf{x}(t_N)$ which is close to the true output vector.

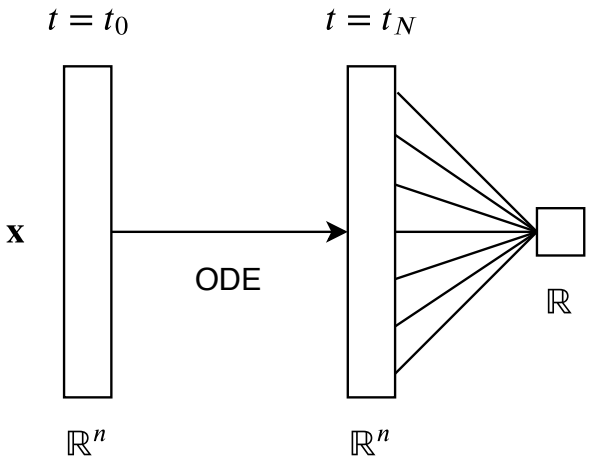


Figure 4: This figure has been reproduced from [DDT19, Figure 2] and depicts a diagram of Neural ODE architecture composed of an ODE layer and a linear layer.

However, we are often interested in learning functions from $\mathbb{R}^n$ to $\mathbb{R}$[7], e.g. for regression or classification. To define a model from $\mathbb{R}^n$ to $\mathbb{R}$, we follow the construction given by

[6] Surprisingly, Chen et al. [Che+18] in this step drop the direct dependence of $\theta$ on time $t$.

[7] We simplified here to functions from $\mathbb{R}^n$ to $\mathbb{R}$. Neural ODEs for image classification problems learn function from $\mathbb{R}^n$ to the discrete label set $\mathcal{Y}$.

Lin et al. [LJ18] for residual networks. We define the Neural ODE $g : \mathbb{R}^n \to \mathbb{R}$ as $g(\mathbf{x}) = \mathcal{L}(\phi(\mathbf{x}(t_0))) = \mathcal{L}(\mathbf{x}(t_N))$ where $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ is a linear map and $\phi : \mathbb{R}^n \to \mathbb{R}^n$ is a mapping from data to features. As shown in figure (4), this is a simple model architecture: an ODE layer, followed by a linear layer. Neural ODE models have shown great promise on a number of tasks including modeling continuous time data and building normalizing flows with low computational cost [Che+18], [Gra+18]. Their desirable properties, such as invertibility and parameter efficiency, have led to more research on Neural ODEs from the perspectives of optimization techniques [Pas+17], [Che+18], [Qua+19], approximation capabilities [DDT19] and generalization [Liu+19], [Yan+19].

Concerning the optimization of Neural ODEs, reverse-mode differentiation (backpropagation) through the ODE solver can be performed in order to train such continuous-depth networks [Pas+17]. However, this training procedure is computationally and memory inefficient [Che+18]. To overcome this issue, Chen et al. [Che+18] propose to compute gradients using the adjoint sensitivity method [Pon+62]. The first rigorous version of this method appears in the Mathematical Theory of Optimal Processes textbook [Pon+62] under the name Pontryagin's Maximum Principle (PMP). This classical version was then improved by other authors, such as the versions appearing in [AF66], [Ber74], [Lib12], which show that the PMP approach computes gradients by solving a second, augmented ODE backwards in time and is applicable to all ODE solvers. In the next sections we try to develop a rather formal optimal control approach to find efficient optimization techniques for training continuous-depth deep learning networks, from which Neural ODEs are derived as a special case, by stating precisely the assumptions under which the PMP holds.

## 4.2 Function Approximation by Dynamical Systems

Let's start with a description of the (continuous) dynamical systems approach to machine learning introduced by E [Wei17]. As usual, we want to approximate some function

$$h : \mathcal{X} \to \mathcal{Y}$$

which maps inputs in $\mathcal{X} \subset \mathbb{R}^n$ (e.g. images, time-series) to labels in $\mathcal{Y}$ (categories, numerical predictions). Given a collection of $d$ sample input-label pairs $\{\mathbf{x}^i, y^i = h(\mathbf{x}^i)\}_{i=1}^d$, we wish to approximate $h$ using these data points. In the dynamical systems framework, the given inputs are considered as the initial condition, i.e. $\mathbf{x}_0 = (\mathbf{x}^1, ..., \mathbf{x}^d) \in \mathbb{R}^{n \times d}$, of a system of ordinary differential equations

$$\dot{\mathbf{x}}^i(t) = f(\mathbf{x}^i(t), \theta(t), t), \quad \mathbf{x}^i(t_0) = \mathbf{x}_0^i, \quad t_0 \leq t \leq t_N, \tag{4.3}$$

where $\theta : [t_0, t_N] \to \Theta \subset \mathbb{R}^m$ represents the control (training) parameters and $\mathbf{x}(t) = (\mathbf{x}^1(t), ..., \mathbf{x}^d(t)) \in \mathbb{R}^{n \times d}$ for all $t \in [t_0, t_N]$. The form of $f$ is chosen as part of the deep learning model and it is typically the composition of a linear transformation and a component-wise non-linear activation function. For the solution to (4.3) to exist for any $\theta$, it is assumed that $f$ and $\partial f / \partial \mathbf{x}$ are continuous in $\mathbf{x}, \theta, t$. For the $i^{\text{th}}$ input sample, the prediction of the network is a deterministic transformation of the terminal state $\mathcal{L}(\mathbf{x}^i(t_N))$ for some $\mathcal{L} : \mathbb{R}^n \to \mathcal{Y}$, which we can view collectively as a function of the initial state (input) $\mathbf{x}_0^i$ and the control parameters (weights) $\theta$. The results from the optimal control theory of the next section (4.3) allow us to consider quite a general space of controls, such as the set of all Lebesgue measurable controls

$$\mathcal{U} := \{\theta : [t_0, t_N] \to \Theta : \theta \text{ is Lebesgue measurable}\}.$$

The goal is to choose $\theta$ from $\mathcal{U}$ so that $\mathcal{L}(\mathbf{x}^i(t_N))$ most closely resembles $y^i$ for $i \in [d]$. We define a loss function $L : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ which is minimized when its arguments are equal, and we consider minimizing $\sum_i L[\mathcal{L}(\mathbf{x}^i(t_N)), y^i]$. Since $\mathcal{L}$ is fixed, Li et al. [Li+17] absorb it into the definition of the loss function by defining $L_i(\cdot) := L(\cdot, y^i)$. Then, they formulate the problem in form of the following optimization problem:

$$\begin{aligned} &\min_{\theta \in \mathcal{U}} \sum_{i=1}^{d} L_i(\mathbf{x}^i(t_N)) + \int_{t_0}^{t_N} R(\theta(t)) dt \\ &\dot{\mathbf{x}}^i(t) = f(\mathbf{x}^i(t), \theta(t), t), \quad \mathbf{x}^i(t_0) = \mathbf{x}_0^i, \quad t_0 \leq t \leq t_N, \quad i \in [d] \end{aligned} \tag{4.4}$$

where $R : \Theta \to \mathbb{R}$ is a running cost, or the regularizer. Considering the fact that most current machine learning models do not regularize the states, Li et al. [Li+17] omit the general case of $R$ depending on $\mathbf{x}(t)$.

Problem (4.4) is a special case of a class of general optimal control problem for ordinary differential equations [AF66]. This formulation allows for deriving the optimality conditions of (4.4) entirely in continuous time and find numerical algorithms that can subsequently be discretized. Note that this approach of first optimizing and then discretizing is an alternative approach to deep learning [Li+17]. In E [Wei17], deep residual networks [He+16a] were considered as the forward Euler discretization of the continuous

approach described above. Hence, the algorithms presented later can also be formulated in the context of deep residual networks.

For simplicity of notation, Li et al. [Li+17] set $d = 1$ and drop the scripts $i$ on all functions, noting that similar results can be obtained in the general case since the dynamics and loss functions are decoupled across samples. In other words, this can be thought as effectively concatenating all $d$ sample inputs into a single input vector of dimension $n \times d$ and redefining the dynamics accordingly. Hence, all results remain valid by performing full-batch training [Li+17] and problem (4.4) can be reformulated as follows:

$$\begin{aligned} &\min_{\theta \in \mathcal{U}} L(\mathbf{x}(t_N)) + \int_{t_0}^{t_N} R(\theta(t))dt \\ &\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \theta(t), t), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \quad t_0 \leq t \leq t_N. \end{aligned} \tag{4.5}$$

Next sections are devoted to finding conditions for optimal solutions of (4.5).

## 4.3 Optimal Control Theory

We dedicate this section to studying a generalization of the celebrated Pontryagin's Minimum Principle (PMP)[8] in optimal control given by Athans et al. [AF66]. The proof of the PMP and its variants can be found in any of the optimal control theory textbooks [Pon+62], [AF66], [Lib12]. We will follow the comprehensive study of Athans et al. [AF66], who present a general version of the PMP, from which, as a special case, Li et al. [Li+17] introduce a set of necessary conditions for optimal solutions of (4.5).

Athans et al. [AF66] start by stating a very general control problem (4.2) and then they carefully delineate a special case of that control problem. Next they give a precise statement of the necessary conditions for optimality, first presented by Pontryagin et al. [Pon+62], which constitute the minimum principle in the special case of the control problem. By various changes of variable, Athans et al. [AF66] show that the minimum principle for the general problem can be obtained from the results for the special problem.

[8] The difference between the notation Pontryagin's Maximum Principle and Pontryagin's Minimum Principle will be clarified in remark (4.9). Since they both refer to the same problem we use the common notation PMP.

### 4.3.1 Admissible Controls

The vector space $\mathbb{R}^n$ of the vector variable $\mathbf{x}$ is the phase space of the object under consideration, whose state at any instant of time is characterized by $x_1, x_2, ..., x_n$. It is assumed that the object is equipped with certain controllers (parameters), characterized by points $\theta$, of a certain control region $\Theta$, an arbitrary set in $\mathbb{R}^m$.

Pontryagin et al. [Pon+62, p. 75] give the first and the most precise definition of a control and the class of admissible controls. Every function $\theta = \theta(t)$, defined on some time interval $t_0 \leq t \leq t_N$, with range in $\Theta$, is called a control. Since $\Theta$ is a set in the space of the control parameters $\theta_1, \theta_2, ..., \theta_m$, each control

$$\theta(t) = (\theta_1(t), \theta_2(t), ..., \theta_m(t))$$

is a vector function (given for $t_0 \leq t \leq t_N$) whose range is in $\Theta$. The control $\theta(t), t_0 \leq t \leq t_N$, is called measurable if the set of all $t$ for which $\theta(t) \in O$ is measurable (in the sense of ordinary Lebesgue measure) in the interval $t_0 \leq t \leq t_N$, for every open set $O \subset \mathbb{R}^m$; and it is called bounded if the set of points $\theta(t), t_0 \leq t \leq t_N$, has a compact closure in $\mathbb{R}^m$. Pontryagin et al. [Pon+62, p. 75] then define a class $\mathcal{U}$ of controls, satisfying the conditions:

1. All the controls $\theta(t), t_0 \leq t \leq t_N$, which belong to the class $\mathcal{U}$ are measurable and bounded.

2. If $\theta(t), t_0 \leq t \leq t_N$, is an admissible control, $\eta$ is an arbitrary point of $\Theta$, and $t'$ and $t''$ are numbers such that $t_0 \leq t' \leq t'' \leq t_N$, the control $\theta_1(t), t_0 \leq t \leq t_N$, defined by the formula

$$\theta_1(t) = \begin{cases} \eta & \text{for} \quad t' \leq t \leq t'' \\ \theta(t) & \text{for} \quad t_0 \leq t \leq t' \quad \text{or} \quad t'' \leq t \leq t_N \end{cases}$$

   is also admisible.

3. If the interval $t_0 \leq t \leq t_N$ is broken up by means of subdivision points into a finite number of subintervals, on each of which the control $\theta(t)$ is admissible, then this control is also admissible on the entire interval $t_0 \leq t \leq t_N$. An admissible control considered on a subinterval is also admissible. A control obtained from an admissible control $\theta(t), t_0 \leq t \leq t_N$, by a translation in time (i.e. the control $\theta_1(t) = \theta(t - \alpha),\ t_0 + \alpha \leq t \leq t_N + \alpha$) is also admissible.

Controls belonging to the class $\mathcal{U}$ are called admissible. For the class of admissible controls one may, for instance, take the set of all measurable[9] and bounded controls, which contains every other class of admissible controls as a subclass [Pon+62, p. 76].

Another example is the set of all piecewise continuous controls (with range in $\Theta$), i.e. controls $\theta = \theta(t)$ which are continuous for all $t$ under consideration, with the exception of only a finite number of $t$, at which $\theta(t)$ may have discontinuities of the first kind. From the definition of discontinuities of the first kind, it is assumed the existence of the finite limits [Pon+62, p. 10]

$$\theta(\tau - 0) = \lim_{\substack{t \to \tau \\ t < \tau}} \theta(t), \qquad \theta(\tau + 0) = \lim_{\substack{t \to \tau \\ t > \tau}} \theta(t)$$

at a point of discontinuity, $\tau$. In particular, it therefore follows that every control $\theta(t)$ is bounded (even if $\Theta$ is not). For completeness, it is assumed that at each point of discontinuity, $\tau$, the value of $\theta(t)$ is equal to its left-hand limit

$$\theta(\tau) = \theta(\tau - 0)$$

and that each control $\theta(t)$ under consideration is continuous at the endpoints of the interval $t_0 \leq t \leq t_N$, on which it is given. [10]

Being the most interesting class for the practical applications, we will fix for the rest of this chapter the class of admissible controls $\mathcal{U}$ to be the set of all bounded piecewise continuous functions $\theta(t)$ on $[t_0, t_N]$. This class of admissible controls corresponds to the assumption of inertialess controllers, since the values of the function $\theta(t)$ may jump (at an instant of discontinuity) instantaneously from one point of the control region to another.

Pontryagin et al. [Pon+62, Chapter V] show that the control problem in optimal control theory, to be formulated in section (4.3.2), is a (highly non-trivial) generalization of the problem of Lagrange in the calculus of variations, and is equivalent to the latter in the case where the control region $\Theta$ is an open set in $\mathbb{R}^m$. Thus, an essential advantage of the minimum principle over the classical theorems in the calculus of variations is the fact that this principle is applicable for any (in particular, closed) set $\Theta \subset \mathbb{R}^m$. This extension

[9] The class of measurable controls is obtained from that of piecewise continuous controls by taking the closure with respect to the a.e. convergence [Lib12, p. 86].

[10] This assumption is made in order to avoid having to add the phrase "except possibly on a countable subset" to many of the statements.

of the class of possible control regions $\Theta$ is highly essential in engineering applications of the theory. That's why we follow the results presented in the optimal control theory.

### 4.3.2 Statement of the Control Problem

We assume that the object's law of motion can be written in the form of a $n$th-order system of differential equations

$$\dot{\mathbf{x}} = f(\mathbf{x}, \theta, t). \tag{4.6}$$

Let $R(\mathbf{x}, \theta, t)$ be a real-valued function on $\mathbb{R}^n \times \mathbb{R}^m \times (T_0, T_N)$, and let $L(\mathbf{x}, t)$ be a real-valued function on $\mathbb{R}^n \times (T_0, T_N)$. Further, we suppose that $S$ is a given subset of $\mathbb{R}^n \times (T_0, T_N)$ so that the elements of $S$ are pairs $(\mathbf{x}, t)$ consisting of a state $\mathbf{x}$ and a point $t$ in the interval of definition of the system.
We now make the following assumptions:

A1) If $f_1(\mathbf{x}, \theta, t), f_2(\mathbf{x}, \theta, t), ..., f_n(\mathbf{x}, \theta, t)$ denote the components of $f(\mathbf{x}, \theta, t)$, then we assume that the functions

$$f_i(\mathbf{x}, \theta, t), \quad \frac{\partial f_i}{\partial \mathbf{x}}(\mathbf{x}, \theta, t), \quad \frac{\partial f_i}{\partial t}(\mathbf{x}, \theta, t) \qquad i = 1, 2, ..., n$$

and the functions

$$R(\mathbf{x}, \theta, t), \quad \frac{\partial R}{\partial \mathbf{x}}(\mathbf{x}, \theta, t), \quad \frac{\partial R}{\partial t}(\mathbf{x}, \theta, t)$$

are continuous on $\mathbb{R}^n \times \overline{\Theta} \times (T_0, T_N)$, where $\overline{\Theta}$ is the closure of $\Theta$ in $\mathbb{R}^m$.

A2) We assume that $S$ is of the form $S = \mathbb{R}^n \times (T_0, T_N)$.

A3) If $L(\mathbf{x}, t)$ is the given real-valued function defined on $\mathbb{R}^n \times (T_0, T_N)$ and if we assume that $S$ is of the form of A2), then $L(\mathbf{x}, t)$ is independent of $t$, i.e. $L(\mathbf{x}, t) = L(\mathbf{x})$, and the functions $L(\mathbf{x})$, $\frac{\partial L}{\partial \mathbf{x}}(\mathbf{x})$, $\frac{\partial^2 L}{\partial \mathbf{x}^2}(\mathbf{x})$ are continuous.

Under these assumptions, the set $S$ is called the target set and $L$ the terminal cost (loss) function.
Then, for all $\mathbf{x}_0$ in $\mathbb{R}^n$, all $t_0$ and $t$ in $(T_0, T_N)$ with $t \geq t_0$, and all $\theta_{(t_0,t]}$ with $\theta$ in $\mathcal{U}$, there is a unique solution $\mathbf{x}(\tau)$ [AF66, Theorem 3-14] of the vector differential equation (4.6)

$$\dot{\mathbf{x}}(\tau) = f\left(\mathbf{x}(\tau), \theta(\tau), \tau\right)$$

defined on the interval $[t_0, t]$ with

$$\mathbf{x}(t_0) = \mathbf{x}_0.$$

We write

$$\mathbf{x}(t) = \varrho(t; \theta_{(t_0,t]}, \mathbf{x}_0)$$

to represent this solution, called the state equation.
$\varrho$ is a continuous function of all its arguments which satisfies the following conditions [AF66, p. 169]:

1. $\mathbf{x}_0 = \varrho(t_0; \theta_{(t_0,t]}, \mathbf{x}_0)$.
2. $\varrho(t; \theta_{(t_0,t]}, \mathbf{x}_0) = \varrho(t; \theta_{(\hat{t},t]}, \varrho(\hat{t}; \theta_{(t_0,\hat{t}]}, \mathbf{x}_0))$ for all $\hat{t}$ in $[t_0, t]$.
3. If $\theta = \eta$ on $(t_0, t]$, then, for all $\tau$ in $[t_0, t]$,

$$\varrho(\tau; \theta_{(t_0,\tau]}, \mathbf{x}_0) = \varrho(\tau; \eta_{(t_0,\tau]}, \mathbf{x}_0).$$

Let's now state the definitions leading to the formal statement of the control problem.

**Definition 4.1.** *[AF66, Definition 5-8]*
*Let $t_0$ be an element of $(T_0, T_N)$ and let $\mathbf{x}_0$ be an element of $\mathbb{R}^n$. Then the admissible control $\theta$ takes $(\mathbf{x}_0, t_0)$ to $S$, where $S$ is the target set, if the set*

$$\left\{(\varrho(t; \theta_{(t_0,t]}, \mathbf{x}_0), t) : t \geq t_0\right\}$$

*intersects $S$. If $\theta$ takes $\mathbf{x}_0$ to $S$ and if $t_N$ is the first instant of time after $t_0$ when the motion $\mathbf{x}(t) = \varrho(t; \theta_{(t_0,t]}, \mathbf{x}_0)$ enters $S$ and if*

$$\mathbf{x}_N = \mathbf{x}(t_N) = \varrho(t_N; \theta_{(t_0,t_N]}, \mathbf{x}_0)$$

*then the well-defined number $J(\mathbf{x}_0, \theta, t_0)$ given by*

$$\begin{aligned} J(\mathbf{x}_0, \theta, t_0) &= L(\mathbf{x}_N, t_N) + \int_{t_0}^{t_N} R(\mathbf{x}(t), \theta(t), t)dt \\ &= L[\varrho(t_N; \theta_{(t_0,t_N]}, \mathbf{x}_0), t_N] + \int_{t_0}^{t_N} R[\varrho(t; \theta_{(t_0,t]}, \mathbf{x}_0), \theta(t), t]dt \end{aligned}$$

*is called the cost functional of the control $\theta$. $t_N$ is called the terminal time and $\mathbf{x}_N$ the terminal state.*

**Definition 4.2** (The Control Problem)**.** *[AF66, Definition 5-9]*
*The control problem for the system equation (4.6) under assumptions A1) - A3) and with respect to target set $S$, the cost functional $J(\mathbf{x}_0, \theta, t_0)$, the set $\mathcal{U}$ of admissible controls, and the initial value $\mathbf{x}_0$ at the initial time $t_0$ is: Determine the control $\theta$ in $\mathcal{U}$ which minimizes the cost functional $J(\mathbf{x}_0, \theta, t_0)$, or simply $J(\theta)$.*

### 4.3.3 Pontryagin's Minimum Principle

Having stated the general control problem, Athans et al. [AF66] delineate a special case of it by listing additional assumptions:

A4) The system equation (4.6) does not depend on $t$ explicitly, i.e. the system equation is of the form

$$\dot{\mathbf{x}} = f(\mathbf{x}, \theta). \tag{4.7}$$

A5) The function $R$ does not depend on $t$ explicitly, and the function $L$ is identically zero (that is, $L(\mathbf{x}, t) \equiv 0$), so that the cost functional $J(\theta)$ is given by

$$J(\theta) = \int_{t_0}^{t_N} R(\mathbf{x}(t), \theta(t))dt. \tag{4.8}$$

Hence, we can reformulate the control problem (4.2) together with the additional assumptions A4) and A5) in the following way:

$$\begin{aligned} &\min_{\theta \in \mathcal{U}} \int_{t_0}^{t_N} R\left(\mathbf{x}(t), \theta(t)\right) dt \\ &\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \theta(t)), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \quad t_0 \leq t \leq t_N. \end{aligned} \tag{4.9}$$

Before giving a set of necessary conditions for optimal solutions of (4.9), Athans et al. [AF66] introduce some additional terminology and notation as follows:

**Definition 4.3** (Hamiltonian function)**.** *[AF66, Definition 5-10]*
*Let $H(\mathbf{x}, \mathbf{p}, \theta)$ denote the real-valued function of the $n$ vector $\mathbf{x}$, the $n$ vector $\mathbf{p}$, and the $m$ vector $\theta$, given by:*

$$H(\mathbf{x}, \mathbf{p}, \theta) = R(\mathbf{x}, \theta) + \langle \mathbf{p}, f(\mathbf{x}, \theta) \rangle$$

*where $f(\mathbf{x}, \theta)$ is the function which determines our system (i.e. the right hand side of the state equation (4.7)) and $R(\mathbf{x}, \theta)$ is the integrand of the cost functional. $H(\mathbf{x}, \mathbf{p}, \theta)$ is said to be the Hamiltonian of our problem and $\mathbf{p}$ is called a costate vector.*

Note that, in view of the assumption A1), the functions $H(\mathbf{x}, \mathbf{p}, \theta)$ and $\frac{\partial H}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{p}, \theta)$ are continuous on $\mathbb{R}^n \times \mathbb{R}^n \times \overline{\Theta}$. Moreover, we observe that $\frac{\partial H}{\partial \mathbf{p}}(\mathbf{x}, \mathbf{p}, \theta)$ is well defined and is given by

$$\frac{\partial H}{\partial \mathbf{p}}(\mathbf{x}, \mathbf{p}, \theta) = f(\mathbf{x}, \theta).$$

Now, if we assume that $\mathbf{x}_0$ is our initial state and that $t_0$ is our initial time, and if $\hat{\mathbf{x}}(t)$ denotes the trajectory of our system starting from $\mathbf{x}_0 = \hat{\mathbf{x}}(t_0)$ generated by $\hat{\theta}(t)$, then, for any function $\mathbf{p}(t)$,

$$\dot{\hat{\mathbf{x}}}(t) = \frac{\partial H}{\partial \mathbf{p}}(\hat{\mathbf{x}}(t), \mathbf{p}(t), \hat{\theta}(t)) = f(\hat{\mathbf{x}}(t), \hat{\theta}(t)).$$

In addition, if $\pi$ is any $n$ vector, then the linear differential equation

$$\begin{aligned} \dot{\mathbf{p}}(t) &= -\frac{\partial H}{\partial \mathbf{x}}(\hat{\mathbf{x}}(t), \mathbf{p}(t), \hat{\theta}(t)) \\ &= -\frac{\partial R}{\partial \mathbf{x}}(\hat{\mathbf{x}}(t), \hat{\theta}(t)) - \left(\frac{\partial f}{\partial \mathbf{x}}(\hat{\mathbf{x}}(t), \hat{\theta}(t))\right)' \mathbf{p}(t) \end{aligned} \tag{4.10}$$

has a unique solution $\mathbf{p}(t, \pi)$ satisfying the initial condition

$$\mathbf{p}(t_0, \pi) = \pi.$$

These observations lead to the following definition:

**Definition 4.4** (Hamiltonian system)**.** *[AF66, Definition 5-11]*
*The 2nth-order system of differential equations*

$$\begin{aligned} \dot{\mathbf{x}} &= \frac{\partial H}{\partial \mathbf{p}}(\mathbf{x}, \mathbf{p}, \theta) = f(\mathbf{x}, \theta) \\ \dot{\mathbf{p}} &= -\frac{\partial H}{\partial \mathbf{x}}(\mathbf{x}, \mathbf{p}, \theta) = -\frac{\partial R}{\partial \mathbf{x}}(\mathbf{x}, \theta) - \left(\frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}, \theta)\right)' \mathbf{p} \end{aligned}$$

*is called the Hamiltonian system associated with our problem. If $\hat{\theta}(t)$ is an admissible control with $\hat{\mathbf{x}}(t)$ as corresponding trajectory, then any solution $\hat{\mathbf{p}}(t)$ of the system (4.10) corresponds to $\hat{\theta}(t)$ and $\hat{\mathbf{x}}(t)$. In other words, $\hat{\mathbf{p}}(t)$ corresponds to $\hat{\theta}(t)$ if $\hat{\mathbf{p}}(t)$ and $\hat{\mathbf{x}}(t)$ are a solution of the Hamiltonian system.*

The following theorem then gives the necessary optimal conditions for the control problem (4.2) satisfying A1) - A5), or equivalently, the necessary optimal conditions for the problem (4.9).

**Theorem 4.5** (Pontryagin's Minimum Principle)**.** *[AF66, Theorem 5-6]*
*Let $\theta^*(t)$ be an admissible control which transfers $(\mathbf{x}_0, t_0)$ to $S = \mathbb{R}^n \times (T_0, T_N)$. Let $\mathbf{x}^*(t)$ be the trajectory of equation (4.7) corresponding to $\theta^*(t)$, originating at $(\mathbf{x}_0, t_0)$, and meeting $S$ (for the first time) at $t_N$ (that is, $\mathbf{x}^*(t_N) \in \mathbb{R}^n$). In order that $\theta^*(t)$ be optimal for the cost functional (4.8), it is necessary that there exist a vector function $\mathbf{p}^*(t)$ such that:*

*i)* $\mathbf{p}^*(t)$ *corresponds to* $\theta^*(t)$ *and* $\mathbf{x}^*(t)$*, so that* $\mathbf{p}^*(t)$ *and* $\mathbf{x}^*(t)$ *are a solution of the Hamiltonian system*

$$\begin{aligned}\dot{\mathbf{x}}^*(t) &= \frac{\partial H}{\partial \mathbf{p}}(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t)) \\ \dot{\mathbf{p}}^*(t) &= -\frac{\partial H}{\partial \mathbf{x}}(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t))\end{aligned}$$

*satisfying the boundary conditions*

$$\mathbf{x}^*(t_0) = \mathbf{x}_0, \quad \mathbf{x}^*(t_N) \in \mathbb{R}^n.$$

*ii) The function* $H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta)$ *has an absolute minimum* [11] *as a function of* $\theta$ *over* $\Theta$ *at* $\theta = \theta^*(t)$ *for* $t$ *in* $[t_0, t_N]$*, that is,*

$$\min_{\theta \in \Theta} H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta) = H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t))$$

*or, equivalently,*

$$H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t)) \leq H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta), \quad \textit{for all} \quad \theta \in \Theta.$$

*iii) The function* $H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t)) = 0$ *for* $t \in [t_0, t_N]$.

*iv) The vector* $\mathbf{p}^*(t_N)$ *is the zero vector, that is,* $\mathbf{p}^*(t_N) = \mathbf{0}$.

The proof of the theorem (4.5), first presented in [Pon+62], makes frequent use of geometric ideas which help us to get a more intuitive way of understanding its necessary optimality conditions.
The proof starts by introducing an auxiliary variable $x_0$ such that

$$\dot{x}_0(t) = R(\mathbf{x}(t), \theta(t)), \quad x_0(t_0) = 0.$$

This immediately gives

$$\begin{aligned}J(\mathbf{x}_0, \theta, t_0) &= \int_{t_0}^{t_N} R(\mathbf{x}(t), \theta(t)) dt \\ &= \int_{t_0}^{t_N} \dot{x}_0(t) dt = x_0(t_N).\end{aligned}$$

[11] A point $\mathbf{x}^*$ in $X$ is said to be an absolute minimum (maximum) of a real-valued function $g$ of a vector $\mathbf{x}$ if $g(\mathbf{x}^*) \leq g(\mathbf{x})$, respectively $g(\mathbf{x}^*) \geq g(\mathbf{x})$, for all $\mathbf{x}$ in $X \subseteq \mathbb{R}^n$.

Considering the $(n+1)$st-order system

$$\begin{aligned} \dot{x}_0(t) &= R(\mathbf{x}(t), \theta(t)) \\ \dot{\mathbf{x}}(t) &= f(\mathbf{x}(t), \theta(t)) \end{aligned} \tag{4.11}$$

then the special problem (general control problem (4.2) together with the assumptions A4) and A5) of the section (4.3.2)) may be rephrased as follows:

Given the $(n+1)$st-order system (4.11); given the initial time $t_0$ and the initial condition $(0, \mathbf{x}_0)$, i.e. $x_0(t_0) = 0$ in $\mathbb{R}^{n+1}$. Let $S = S' \times (T_0, T_N)$ be the target set (in $\mathbb{R}^{n+1} \times (T_0, T_N)$) where $S'$ is the line in $\mathbb{R}^{n+1}$ through $(0, \mathbf{x}_N)$ and parallel to the $x_0$ axis. $S'$ is given by the equations

$$x_1 - x_{N1} = 0, x_2 - x_{N2} = 0, ..., x_n - x_{Nn} = 0$$

where the $x_{Nj}$ are the components of the vector $\mathbf{x}_N$. Then determine the $\theta$ which transfers $\left( \begin{bmatrix} 0 \\ \mathbf{x}_0 \end{bmatrix}, t_0 \right)$ to $S' \times (T_0, T_N)$ and which, in so doing, minimizes $x_0$ at the terminal time.

In other words, we want to find the trajectory of the system (4.11), starting from the point $(0, \mathbf{x}_0)$, which intersects the line $S'$ in a point with smallest $x_0$ coordinate.
The geometric situation is illustrated in figure (5). The top part of the figure exhibits an optimal $(^*)$ trajectory in $\mathbb{R}^{n+1}$. Often the state space is indicated as the $x_i x_j$ plane, and the cost axis $(x_0)$ is viewed as pointing upward. In the bottom part of figure (5) the state space has been compressed into a single axis (the $x$ axis). The arrows indicate the motion of a point (in $\mathbb{R}^n$ or $\mathbb{R}^{n+1}$) as time increases. $t^*$ denotes the terminal time, so that $\mathbf{x}^*(t^*) = \mathbf{x}_N$.

### 4.3.4 Pontryagin's Minimum Principle: Change of Variable

Athans et al. [AF66, pp. 291-303] show, by various changes of variable, how the minimum principle for the general control problem (4.2) can be obtained from theorem (4.5). Keeping the assumptions of sections (4.3.2), (4.3.3) in force, they start by removing the restriction of time independence for the system and the cost functional, and then they treat the problem in which there is a terminal cost function. In both cases the problem is reduced to the special case (general control problem (4.2) together with the assumptions A4) and A5) of the section (4.3.2)).

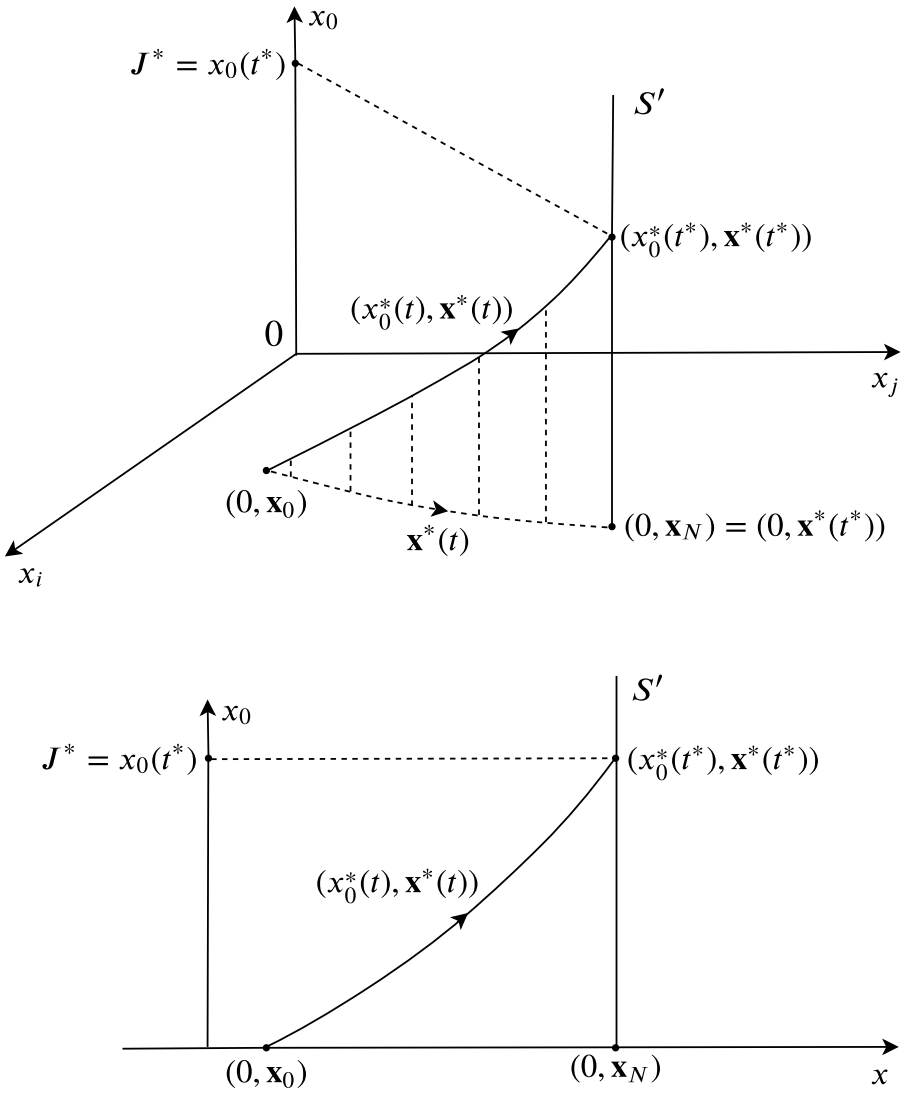


Figure 5: This figure has been reproduced from [AF66, Figure 5-16] and depicts the geometric interpretation of the control problem (4.2) under assumptions A4) and A5).

#### 4.3.4.1 Time-dependent System and Cost Functional

Let's suppose that the system equation is of the form

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \theta(t), t) \tag{4.12}$$

and let's assume that the target set $S$ is of the form

$$S = \mathbb{R}^n \times (T_0, T_N).$$

We also suppose that $R$ is of the form

$$R(\mathbf{x}, \theta, t)$$

and that the cost functional $J(\theta)$ is given by

$$J(\theta) = \int_{t_0}^{t_N} R(\mathbf{x}(t), \theta(t), t)dt, \quad t_N \text{ free.} \tag{4.13}$$

As usual, we let $\mathbf{x}_0$ be the initial state and $t_0$ be the initial time.
The aim is to reduce the control problem for time-dependent system (4.12) with cost functional (4.13) to the special problem of section (4.3.2). To achieve that, Athans et al. [AF66] first let $x_{n+1}$ denote an auxiliary variable and consider the $(n+1)$st-order system

$$\begin{aligned} \dot{\mathbf{x}}(t) &= f(\mathbf{x}(t), \theta(t), x_{n+1}) \\ \dot{x}_{n+1}(t) &= 1. \end{aligned} \tag{4.14}$$

Then they examine the following (auxiliary) control problem for the system (4.14):
Given the initial time $t_0$ and the initial condition $\begin{bmatrix} \mathbf{x}_0 \\ t_0 \end{bmatrix}$ (that is, $x_{n+1}(t_0) = t_0$) in $\mathbb{R}^{n+1}$. Let $S = \mathbb{R}^{n+1} \times (T_0, T_N)$ be the target set. Then determine the admissible control $\theta$ which transfers $\left( \begin{bmatrix} \mathbf{x}_0 \\ t_0 \end{bmatrix}, t_0 \right)$ to $\mathbb{R}^{n+1} \times (T_0, T_N)$ and which, in so doing, minimizes the cost functional $J_1(\theta)$, given by

$$J_1(\theta) = \int_{t_0}^{t_N} R(\mathbf{x}(t), \theta(t), x_{n+1}(t))dt, \quad t_N \text{ free.}$$

We observe that this problem is equivalent to our original problem (for the system (4.12)) in the sense that a control is optimal for one iff it is optimal for the other. Furthermore, this new problem is precisely of the form of special problem of section (4.3.2). Thus, theorem (4.5) may be applied to deduce that if $\theta^*(t)$ is optimal, then there are a function $\mathbf{p}^*(t)$ and a function $p^*_{n+1}(t)$ such that the following conditions are satisfied:

1.

$$\begin{aligned} \dot{\mathbf{x}}^*(t) &= \frac{\partial H_1}{\partial \mathbf{p}}(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), x^*_{n+1}(t), p^*_{n+1}(t)) \\ \dot{\mathbf{p}}^*(t) &= -\frac{\partial H_1}{\partial \mathbf{x}}(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), x^*_{n+1}(t), p^*_{n+1}(t)) \\ \dot{x}^*_{n+1}(t) &= \frac{\partial H_1}{\partial p_{n+1}}(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), x^*_{n+1}(t), p^*_{n+1}(t)) = 1 \\ \dot{p}^*_{n+1}(t) &= -\frac{\partial H_1}{\partial x_{n+1}}(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), x^*_{n+1}(t), p^*_{n+1}(t)) \end{aligned}$$

where $H_1(\mathbf{x}, \mathbf{p}, \theta, x_{n+1}, p_{n+1})$ is given by the equation

$$H_1(\mathbf{x}, \mathbf{p}, \theta, x_{n+1}, p_{n+1}) = R(\mathbf{x}, \theta, x_{n+1}) + \langle \mathbf{p}, f(\mathbf{x}, \theta, x_{n+1}) \rangle + p_{n+1} \cdot 1$$

and where

$$\begin{aligned} \mathbf{x}^*(t_0) &= \mathbf{x}_0, \qquad & x^*_{n+1}(t_0) &= t_0 \\ \mathbf{x}^*(t_N) &\in \mathbb{R}^n, \qquad & x^*_{n+1}(t_N) &= t_N. \end{aligned}$$

2. The function $H_1(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta, x^*_{n+1}(t), p^*_{n+1}(t))$ has an absolute minimum as a function of $\theta$ over $\Theta$ at $\theta = \theta^*(t)$ for $t$ in $[t_0, t_N]$.

3. $H_1(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), x^*_{n+1}(t), p^*_{n+1}(t)) = 0$ for $t$ in $[t_0, t_N]$.

4. The $n+1$ vector

$$\begin{bmatrix} \mathbf{p}^*(t_N) \\ p^*_{n+1}(t_N) \end{bmatrix}$$

is the zero vector in $\mathbb{R}^{n+1}$, that is,

$$\mathbf{p}^*(t_N) = \mathbf{0}, \quad p^*_{n+1}(t_N) = 0.$$

Now, these results can be applied to our original problem (for the system (4.12)) by noting that the auxiliary variable $x_{n+1}$ is, in fact, the time $t$. Thus, for our original problem, we define a function $H(\mathbf{x}, \mathbf{p}, \theta, t)$ of the $n$ vector $\mathbf{x}$, the $n$ vector $\mathbf{p}$, the $m$ vector $\theta$, and the time $t$ by setting

$$H(\mathbf{x}, \mathbf{p}, \theta, t) = R(\mathbf{x}, \theta, t) + \langle \mathbf{p}, f(\mathbf{x}, \theta, t) \rangle. \tag{4.15}$$

$H$ is called the Hamiltonian of our problem and $\mathbf{p}$ the costate variable. We then have:

**Theorem 4.6** (PMP for the Time-dependent Problem with Moving Target Set)**.** *[AF66, Theorem 5-9]*
*Let $\theta^*(t)$ be an admissible control which transfers $(\mathbf{x}_0, t_0)$ to the target set $S = \mathbb{R}^n \times (T_0, T_N)$. Let $\mathbf{x}^*(t)$ be the trajectory of equation (4.12) corresponding to $\theta^*(t)$, originating at $(\mathbf{x}_0, t_0)$, and meeting $S$ at $t_N$. In order that $\theta^*(t)$ be optimal for the cost functional (4.13), it is necessary that there exist a function $\mathbf{p}^*(t)$ such that:*

*i)* $\mathbf{p}^*(t)$ *corresponds to* $\theta^*(t)$ *and* $\mathbf{x}^*(t)$ *so that* $\mathbf{p}^*(t)$ *and* $\mathbf{x}^*(t)$ *are a solution of the Hamiltonian system*

$$\begin{aligned}\dot{\mathbf{x}}^*(t) &= \tfrac{\partial H}{\partial \mathbf{p}}(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), t)\\ \dot{\mathbf{p}}^*(t) &= -\tfrac{\partial H}{\partial \mathbf{x}}(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), t)\end{aligned}$$

*satisfying the boundary conditions*

$$\mathbf{x}^*(t_0) = \mathbf{x}_0, \quad (\mathbf{x}^*(t_N), t_N) \in S.$$

*ii) The function* $H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta, t)$ *has an absolute minimum as a function of* $\theta$ *over* $\Theta$ *at* $\theta = \theta^*(t)$ *for* $t$ *in* $[t_0, t_N]$*, that is,*

$$\min_{\theta\in\Theta} H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta, t) = H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), t)$$

*or, equivalently,*

$$H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), t) \le H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta, t), \quad \textit{for all } \theta \textit{ in } \Theta.$$

*iii) The function* $H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), t)$ *satisfies the relations*

$$\begin{aligned}H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), t) &= -\int_t^{t_N} \frac{\partial H}{\partial t}(\mathbf{x}^*(\tau), \mathbf{p}^*(\tau), \theta^*(\tau), \tau)d\tau \quad (4.16)\\ H(\mathbf{x}^*(t_N), \mathbf{p}^*(t_N), \theta^*(t_N), t_N) &= 0.\end{aligned}$$

*iv)* $\mathbf{p}^*(t_N)$ *is the zero vector, that is,*

$$\mathbf{p}^*(t_N) = \mathbf{0}.$$

Athans et al. [AF66] note that the main difference between theorems (4.5) and (4.6) is the difference in the behavior of the Hamiltonians along the optimal trajectory. In the time-independent problem of theorem (4.5), the Hamiltonian is zero along the optimal path, while in the time-dependent problem at hand, the Hamiltonian varies with time in accordance with equation (4.16) along the optimal path. We next observe that, in view of the fact that $x_{n+1}(t) = t$, the crucial relation which allows us to deduce theorem (4.6) from conditions 1 to 4, satisfied by the optimum in the new (or auxiliary) problem, is

$$H_1(\mathbf{x}, \mathbf{p}, \theta, t, p_{n+1}) = H(\mathbf{x}, \mathbf{p}, \theta, t) + p_{n+1}.$$

This relation implies that, for given values of $\mathbf{x}, \mathbf{p}$, and $t$, the value of $\theta$ in $\Theta$ which minimizes $H$ will also minimize $H_1$ (for any value of $p_{n+1}$), and vice versa. Moreover, this relation combined with the fact that

$$\dot{p}^*_{n+1}(t) = -\frac{\partial H}{\partial t}(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), t), \quad p^*_{n+1}(t_N) = 0,$$

as $\partial H_1/\partial x_{n+1} = \partial H(\mathbf{x}, \mathbf{p}, \theta, x_{n+1})/\partial x_{n+1}$, enables us to deduce condition *iii)* of theorem (4.6).

#### 4.3.4.2 Terminal Cost Function Present

Until now, it was assumed that our cost functional did not involve a terminal cost (loss) term (i.e. $L(\mathbf{x}, t) \equiv 0$). We will now remove that restriction. We once again suppose that the system equation is of the form

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \theta(t), t) \tag{4.17}$$

and that $\mathbf{x}_0$ is our initial state, with $t_0$ as our initial time. We assume that the target set $S$ is of the form

$$S = \mathbb{R}^n \times (T_0, T_N) \tag{4.18}$$

We suppose that $L(\mathbf{x}, t)$ is a real-valued function on $\mathbb{R}^n \times (T_0, T_N)$ such that

$$L(\mathbf{x}, t), \frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}, t), \frac{\partial L}{\partial t}(\mathbf{x}, t), \frac{\partial^2 L}{\partial \mathbf{x}^2}(\mathbf{x}, t), \frac{\partial^2 L}{\partial \mathbf{x} \partial t}(\mathbf{x}, t), \frac{\partial^2 L}{\partial t^2}(\mathbf{x}, t)$$

are all continuous. We assume then that our cost functional $J(\theta)$ is given by

$$J(\theta) = L(\mathbf{x}(t_N), t_N) + \int_{t_0}^{t_N} R(\mathbf{x}(t), \theta(t), t) dt. \tag{4.19}$$

Athans et al. [AF66] then reduce the control problem for the system (4.17) with cost functional (4.19) and target set $S$ of equation (4.18) to a problem in which there is no terminal cost function, which in turn allows to apply theorem (4.6) to deduce the desired form of the minimum principle.

Since $\mathbf{x}(t_0) = \mathbf{x}_0$, we observe that

$$L(\mathbf{x}(t_N), t_N) = L(\mathbf{x}_0, t_0) + \int_{t_0}^{t_N} \frac{d}{dt}[L(\mathbf{x}(t), t)] dt.$$

However, we know that

$$\begin{aligned} \frac{d}{dt}[L(\mathbf{x}(t), t)] &= \left\langle \frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}(t), t), \dot{\mathbf{x}}(t) \right\rangle + \frac{\partial L}{\partial t}(\mathbf{x}(t), t) \\ &= \left\langle \frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}(t), t), f(\mathbf{x}(t), \theta(t), t) \right\rangle + \frac{\partial L}{\partial t}(\mathbf{x}(t), t). \end{aligned} \tag{4.20}$$

It follows that the cost functional $J(\theta)$ can also be given by

$$J(\theta) = L(\mathbf{x}_0, t_0) + \int_{t_0}^{t_N} \left[ R(\mathbf{x}(t), \theta(t), t) + \left\langle \frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}(t), t), f(\mathbf{x}(t), \theta(t), t) \right\rangle + \frac{\partial L}{\partial t}(\mathbf{x}(t), t) \right] dt.$$

Since $\mathbf{x}_0$ and $t_0$ are given, $L(\mathbf{x}_0, t_0)$ is a known constant, hence it is obvious that the problem of transferring $(\mathbf{x}_0, t_0)$ to $S$ along a trajectory of equation (4.17) in such a way as to minimize the cost functional $J_1(\theta)$, given by

$$J_1(\theta) = \int_{t_0}^{t_N} \left[ R(\mathbf{x}(t), \theta(t), t) + \left\langle \frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}(t), t), f(\mathbf{x}(t), \theta(t), t) \right\rangle + \frac{\partial L}{\partial t}(\mathbf{x}(t), t) \right] dt$$

is equivalent to our original problem, in the sense that a control is optimal for one iff it is optimal for the other.

By applying theorem (4.6) to this new problem, we may deduce that, if $\theta^*(t)$ is optimal, then there is a function $\mathbf{p}_1^*(t)$ such that:

1.

$$\begin{aligned} \dot{\mathbf{p}}_1^*(t) &= -\frac{\partial H_1}{\partial \mathbf{x}}(\mathbf{x}^*(t), \mathbf{p}_1^*(t), \theta^*(t), t) \\ &= -\frac{\partial R}{\partial \mathbf{x}}(\mathbf{x}^*(t), \theta^*(t), t) - \left[ \frac{\partial}{\partial \mathbf{x}} \left\langle \frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}, t), f(\mathbf{x}, \theta, t) \right\rangle + \frac{\partial^2 L}{\partial \mathbf{x} \partial t}(\mathbf{x}, t) \right] \Big|_{(\mathbf{x}^*(t), \theta^*(t), t)} \\ &\quad - \left( \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}^*(t), \theta^*(t), t) \right)' \mathbf{p}_1^*(t). \end{aligned} \tag{4.21}$$

2.

$$\mathbf{p}_1^*(t_N) = \mathbf{0}. \tag{4.22}$$

3.

$$\min_{\theta \in \Theta} H_1(\mathbf{x}^*(t), \mathbf{p}_1^*(t), \theta, t) = H_1(\mathbf{x}^*(t), \mathbf{p}_1^*(t), \theta^*(t), t) \tag{4.23}$$

where $H_1(\mathbf{x}, \mathbf{p}_1, \theta, t)$ is given by

$$\begin{aligned} H_1(\mathbf{x}, \mathbf{p}_1, \theta, t) &= \left[ R(\mathbf{x}, \theta, t) + \left\langle \frac{\partial L}{\partial \mathbf{x}}, f(\mathbf{x}, \theta, t) \right\rangle + \frac{\partial L}{\partial t}(\mathbf{x}, t) \right] + \langle \mathbf{p}_1, f(\mathbf{x}, \theta, t) \rangle \\ &= R(\mathbf{x}, \theta, t) + \left\langle \mathbf{p}_1 + \frac{\partial L}{\partial \mathbf{x}}, f(\mathbf{x}, \theta, t) \right\rangle + \frac{\partial L}{\partial t}(\mathbf{x}, t). \end{aligned}$$

Now, if we consider the function $H(\mathbf{x}, \mathbf{p}, \theta, t)$, given by

$$H(\mathbf{x}, \mathbf{p}, \theta, t) = R(\mathbf{x}, \theta, t) + \langle \mathbf{p}, f(\mathbf{x}, \theta, t) \rangle$$

and if we set

$$\mathbf{p}^*(t) = \mathbf{p}_1^*(t) + \frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}^*(t), t) \tag{4.24}$$

then we have, from equations (4.21) and (4.20),

$$\begin{aligned}
\dot{\mathbf{p}}^*(t) &= -\frac{\partial R}{\partial \mathbf{x}}(\mathbf{x}^*(t),\theta^*(t),t) - \left[\frac{\partial^2 L}{\partial \mathbf{x}^2}(\mathbf{x}^*(t),t) f(\mathbf{x}^*(t),\theta^*(t),t)\right. \\
&\quad \left. + \left(\frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}^*(t),\theta^*(t),t)\right)' \frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}^*(t),t) + \frac{\partial^2 L}{\partial \mathbf{x}\partial t}(\mathbf{x}^*(t),t)\right] \\
&\quad - \left(\frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}^*(t),\theta^*(t),t)\right)' \left[\mathbf{p}^*(t) - \frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}^*(t),t)\right] \\
&\quad + \frac{\partial^2 L}{\partial \mathbf{x}^2}(\mathbf{x}^*(t),t) f(\mathbf{x}^*(t),\theta^*(t),t) + \frac{\partial^2 L}{\partial \mathbf{x}\partial t}(\mathbf{x}^*(t),t) \\
&= -\frac{\partial R}{\partial \mathbf{x}}(\mathbf{x}^*(t),\theta^*(t),t) - \left(\frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}^*(t),\theta^*(t),t)\right)' \mathbf{p}^*(t) \\
&= -\frac{\partial H}{\partial \mathbf{x}}(\mathbf{x}^*(t),\mathbf{p}^*(t),\theta^*(t),t).
\end{aligned}$$

Moreover, since $(\partial L/\partial t)(\mathbf{x}^*(t),t)$ does not depend explicitly on $\theta$, from equation (4.23) it follows that

$$\min_{\theta\in\Theta} H(\mathbf{x}^*(t),\mathbf{p}^*(t),\theta,t) = H(\mathbf{x}^*(t),\mathbf{p}^*(t),\theta^*(t),t).$$

Finally, from equations (4.22) and (4.24), it follows that

$$\mathbf{p}^*(t_N) = \frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}^*(t_N),t_N).$$

We summarize these results in the following theorem:

**Theorem 4.7** (PMP for the Terminal-cost Problem)**.** *[AF66, Theorem 5-11]*
*Let $\theta^*(t)$ be an admissible control which transfers $(\mathbf{x}_0,t_0)$ to the target set $S = \mathbb{R}^n \times (T_0,T_N)$. Let $\mathbf{x}^*(t)$ be the trajectory of equation (4.17) corresponding to $\theta^*(t)$, originating at $(\mathbf{x}_0,t_0)$ and meeting $S$ at $t_N$. In order that $\theta^*(t)$ be optimal for the cost functional (4.19), it is necessary that there exist a function $\mathbf{p}^*(t)$ such that:*

*i) $\mathbf{p}^*(t)$ corresponds to $\theta^*(t)$ and $\mathbf{x}^*(t)$ so that $\mathbf{p}^*(t)$ and $\mathbf{x}^*(t)$ are a solution of the Hamiltonian system*

$$\begin{aligned}
\dot{\mathbf{x}}^*(t) &= \tfrac{\partial H}{\partial \mathbf{p}}(\mathbf{x}^*(t),\mathbf{p}^*(t),\theta^*(t),t) \\
\dot{\mathbf{p}}^*(t) &= -\tfrac{\partial H}{\partial \mathbf{x}}(\mathbf{x}^*(t),\mathbf{p}^*(t),\theta^*(t),t)
\end{aligned}$$

*satisfying the boundary conditions*

$$\mathbf{x}^*(t_0) = \mathbf{x}_0, \quad \mathbf{x}^*(t_N) \text{ free.}$$

*ii)* *The function* $H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta, t)$ *has an absolute minimum as a function of* $\theta$ *over* $\Theta$ *at* $\theta = \theta^*(t)$ *for* $t$ *in* $[t_0, t_N]$, *that is,*

$$\min_{\theta \in \Theta} H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta, t) = H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), t)$$

*or, equivalently,*

$$H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), t) \leq H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta, t), \quad \textit{for all } \theta \textit{ in } \Theta.$$

*iii)* *The function* $H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), t)$ *satisfies the relations*

$$\begin{aligned} H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), t) &= -\frac{\partial L}{\partial t}(\mathbf{x}^*(t_N), t_N) \\ &\quad - \int_t^{t_N} \left[ \frac{\partial H}{\partial t}(\mathbf{x}^*(\tau), \mathbf{p}^*(\tau), \theta^*(\tau), \tau) + \frac{\partial^2 L}{\partial t^2}(\mathbf{x}^*(\tau), \tau) \right] d\tau \\ H(\mathbf{x}^*(t_N), \mathbf{p}^*(t_N), \theta^*(t_N), t_N) &= -\frac{\partial L}{\partial t}(\mathbf{x}^*(t_N), t_N). \end{aligned}$$

*iv)* *The vector* $\mathbf{p}^*(t_N)$ *satisfies*

$$\mathbf{p}^*(t_N) = \frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}^*(t_N), t_N). \tag{4.25}$$

In other words, theorem (4.7) gives us the set of necessary conditions for optimal solutions of the following control problem:

$$\begin{aligned} &\min_{\theta \in \mathcal{U}} L(\mathbf{x}(t_N), t_N) + \int_{t_0}^{t_N} R(\mathbf{x}(t), \theta(t), t) dt \\ &\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \theta(t), t), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \quad t_0 \leq t \leq t_N. \end{aligned} \tag{4.26}$$

Now it is obvious that the problem (4.5) of section (4.2), where we first studied the continuous dynamical systems approach to machine learning, is a special problem of the general problem (4.26). We derive the PMP version formulated in [Li+17] which gives the set of necessary conditions for optimal solutions of problem (4.5) in two steps. First, we do a refinement on the terminal time $t_N$ of the problems considered so far. Next, we redefine the Hamiltonian, which in turn allows us to arrive to a maximum principle.

**Remark 4.8.** (Fixed Terminal Time)
*If we were to consider problems in which the terminal time* $t_N$ *is a fixed element of* $(T_0, T_N)$ *with* $t_N > t_0$, *that is, our target set* $S$ *is of the form*

$$S = \mathbb{R}^n \times \{t_N\},$$

*then the PMP is formulated in the same way as in theorem (4.7) with the condition iii) being automatically satisfied and providing no additional information, and $L = L(\mathbf{x}(t_N))$ with no direct dependence on the terminal time [AF66, Theorem 5.10]. Similarly to section (4.3.4.1), the proof follows by simply introducing an auxiliary variable $x_{n+1} = t$ which in turn allows reducing the problem of fixed terminal time to the special problem of section (4.3.2).*

**Remark 4.9** (Hamiltonian maximization)**.** *[Lib12, pp. 96-97]*
*If we would define the Hamiltonian and the costate variable of equation (4.15) using a different sign convention*

$$\hat{H}(\mathbf{x}, \mathbf{p}, \theta, t) = -R(\mathbf{x}, \theta, t) + \langle \mathbf{p}, f(\mathbf{x}, \theta, t) \rangle, \quad \hat{\mathbf{p}} = -\mathbf{p}^*$$

*then the function*

$$\begin{aligned} \hat{H}(\mathbf{x}^*(t), \hat{\mathbf{p}}(t), \theta, t) &= -R(\mathbf{x}^*(t), \theta, t) - \langle \mathbf{p}^*(t), f(\mathbf{x}^*(t), \theta, t) \rangle \\ &= -H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta, t) \end{aligned}$$

*would have an absolute maximum at $\theta^*(t)$ while $\mathbf{x}^*, \hat{\mathbf{p}}$ would again be a solution of the Hamiltonian system with respect to $\hat{H}$*

$$\dot{\mathbf{x}}^* = f(\mathbf{x}^*, \theta^*, t) = \frac{\partial \hat{H}}{\partial \mathbf{p}}(\mathbf{x}^*, \theta^*, t)$$

*and*

$$\begin{aligned} \dot{\hat{\mathbf{p}}} &= -\dot{\mathbf{p}}^* = \frac{\partial H}{\partial \mathbf{x}}(\mathbf{x}^*, \mathbf{p}^*, \theta^*, t) = \frac{\partial R}{\partial \mathbf{x}}(\mathbf{x}^*, \theta^*, t) + \left( \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}^*, \theta^*, t) \right)' \mathbf{p}^* \\ &= \frac{\partial R}{\partial \mathbf{x}}(\mathbf{x}^*, \theta^*, t) - \left( \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}^*, \theta^*, t) \right)' \hat{\mathbf{p}} = -\frac{\partial \hat{H}}{\partial \mathbf{x}}(\mathbf{x}^*, \hat{\mathbf{p}}, \theta^*, t). \end{aligned}$$

*This formulation, in terms of Hamiltonian maximization is equivalent to the maximization for the cost functional $-J$ (instead of minimization of the cost functional $J$) in our control problem considered before. Hence, whether we arrive at a minimum principle or a maximum principle is determined only by the sign convention, and has nothing to do with whether the cost functional $J$ is being minimized or maximized.*

Finally, from the fact that problem (4.5) is a special problem of the general control problem (4.26), theorem (4.7) and remarks (4.8), (4.9) we can formulate the PMP version given by Li et al. [Li+17, Theorem 1] which gives the set of necessary conditions for optimal solutions of problem (4.5). Obviously, the Hamiltonian for this special case is defined as

$$H(\mathbf{x}, \mathbf{p}, \theta, t) = -R(\mathbf{x}) + \langle \mathbf{p}, f(\mathbf{x}, \theta, t) \rangle$$

whereas our system equation is as before

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \theta(t), t) \tag{4.27}$$

and our cost functional is given by

$$J(\theta) = L(\mathbf{x}(t_N)) + \int_{t_0}^{t_N} R(\theta(t))dt. \tag{4.28}$$

Then we can give the following theorem.

**Theorem 4.10** (Pontryagin's Maximum Principle for the Terminal-cost Problem with Fixed-terminal-time ). *[Li+17, Theorem 1]*
*Let $\theta^*(t)$ be an admissible control which transfers $(\mathbf{x}_0, t_0)$ to the target set $S = \mathbb{R}^n \times \{t_N\}$. Let $\mathbf{x}^*(t)$ be the trajectory of equation (4.27) corresponding to $\theta^*(t)$, originating at $(\mathbf{x}_0, t_0)$ and meeting $S$ at $t_N$. In order that $\theta^*(t)$ be optimal for the cost functional (4.28)[12], it is necessary that there exist a function $\mathbf{p}^*(t)$ such that:*

*i) $\mathbf{p}^*(t)$ corresponds to $\theta^*(t)$ and $\mathbf{x}^*(t)$ so that $\mathbf{p}^*(t)$ and $\mathbf{x}^*(t)$ are a solution of the Hamiltonian system*

$$\dot{\mathbf{x}}^*(t) = \frac{\partial H}{\partial \mathbf{p}}(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), t) \tag{4.29}$$

$$\dot{\mathbf{p}}^*(t) = -\frac{\partial H}{\partial \mathbf{x}}(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), t) \tag{4.30}$$

*satisfying the boundary conditions*

$$\mathbf{x}^*(t_0) = \mathbf{x}_0, \quad \mathbf{x}^*(t_N) \text{ free.} \tag{4.31}$$

*ii) The function $H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta, t)$ has an absolute maximum as a function of $\theta$ over $\Theta$ at $\theta = \theta^*(t)$ for $t$ in $[t_0, t_N]$, that is,*

$$\max_{\theta \in \Theta} H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta, t) = H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), t)$$

*or, equivalently,*

$$H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta^*(t), t) \geq H(\mathbf{x}^*(t), \mathbf{p}^*(t), \theta, t), \quad \text{for all } \theta \text{ in } \Theta. \tag{4.32}$$

*iii) The vector $\mathbf{p}^*(t_N)$ satisfies*

$$\mathbf{p}^*(t_N) = -\frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}^*(t_N)). \tag{4.33}$$

[12]Note that theorem (4.10) is true, in general, for a cost functional where the regularization term $R = R(\mathbf{x}(t), \theta(t), t)$.

From equations (4.29), (4.30), (4.31), (4.32), (4.33) of theorem (4.10) we can solve for the unknowns $\mathbf{x}^*, \mathbf{p}^*, \theta^*$ simultaneously as a function of $t$. Thus, the resulting optimal control $\theta^*$ is open-loop, i.e. $\theta^*(t) = \theta^*(\mathbf{x}^*(t_0), t)$, and is not in a feed-back form $\theta^*(t) = \theta^*(\mathbf{x}^*(t), t)$ [Li+17]. The latter is of closed-loop type and are usually obtained from dynamic programming and the Hamilton-Jacobi-Bellman formalism [Bel57]. From this viewpoint, the PMP gives a weaker control. However, open-loop solutions are sufficient for neural network applications, where the trained weights and treshold values are fixed and only depend on the layer number and not the inputs [Li+17].
Note that the PMP is only a necessary condition, hence the solution to it is not always globally optimal for (4.26). However, in applications the PMP is often strong enough to give good solution candidates, and when certain convexity assumptions are satisfied the PMP becomes sufficient [BP07]. In the following, let's discuss an algorithm to solve the PMP numerically.

#### 4.3.4.3 Basic Method of Successive Approximations

We review a simple numerical algorithm for training (4.26) via solving the PMP (equations (4.29), (4.30), (4.31), (4.32), (4.33)). Among many algorithms (cf. the survey [Rao09]), we present the method of successive approximations (MSA) [CL82], which is an iterative method based on alternating propagation and optimization steps, known to scale well to modern deep learning problems with huge number of state and control variables [Li+17].

Note that (4.29) is simply the state equation

$$\dot{\mathbf{x}}^*(t) = f(\mathbf{x}^*(t), \theta^*(t), t)$$

which is independent of the function $\mathbf{p}^*$. Hence, the algorithm presented by Li et al. [Li+17] proceeds in the following way. First, we make an initial guess of the optimal control $\theta^0 \in \mathcal{U}$. For each $k = 0, 1, 2, ...$, we solve (4.29)

$$\dot{\mathbf{x}}^{\theta^k}(t) = f(\mathbf{x}^{\theta^k}(t), \theta^k(t), t), \quad \mathbf{x}^{\theta^k}(t_0) = \mathbf{x}_0$$

for $\mathbf{x}^{\theta^k}$, which in turn allows us to solve (4.30)

$$\dot{\mathbf{p}}^{\theta^k}(t) = -\frac{\partial H}{\partial \mathbf{x}}(\mathbf{x}^{\theta^k}(t), \mathbf{p}^{\theta^k}(t), \theta^k(t), t), \quad \mathbf{p}^{\theta^k}(t_N) = -\frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}^{\theta^k}(t_N))$$

to get $\mathbf{p}^{\theta^k}$. Finally, using the maximization condition (4.32), we can update

$$\theta^{k+1}(t) = \arg\max_{\theta \in \Theta} H(\mathbf{x}^{\theta^k}(t), \mathbf{p}^{\theta^k}(t), \theta, t)$$

**Algorithm 1** Basic MSA, [Li+17, Algorithm 1]

1: Initialize: $\theta^0 \in \mathcal{U}$
2: **for** $k = 0$ to $\#Iterations$ **do**
3: Solve $\dot{\mathbf{x}}^{\theta^k}(t) = f(\mathbf{x}^{\theta^k}(t), \theta^k(t), t), \quad \mathbf{x}^{\theta^k}(t_0) = \mathbf{x}_0$
4: Solve $\dot{\mathbf{p}}^{\theta^k}(t) = -\frac{\partial H}{\partial \mathbf{x}}(\mathbf{x}^{\theta^k}(t), \mathbf{p}^{\theta^k}(t), \theta^k(t), t), \quad \mathbf{p}^{\theta^k}(t_N) = -\frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}^{\theta^k}(t_N))$
5: Set $\theta^{k+1}(t) = \arg\max_{\theta \in \Theta} H(\mathbf{x}^{\theta^k}(t), \mathbf{p}^{\theta^k}(t), \theta, t)$ for each $t \in [t_0, t_N]$
6: **end for**

for $t \in [t_0, t_N]$. This procedure is summarized in algorithm (1).
Like for the PMP, the two main components of MSA are the forward-backward Hamiltonian dynamics and the maximization for the optimal parameters at each time. Note that the Hamiltonian maximization step in MSA is decoupled for each $t \in [t_0, t_N]$. From the viewpoint of deep learning, the optimization step is decoupled for different network layers and only the Hamiltonian ODEs, i.e. steps 3, 4 of algorithm (1), involve propagation through the layers. This enables the parallelization of the maximization step, which is usually the most time-consuming step [Li+17].

However, as was shown by Aleksandrov et al. [Ale68], the basic MSA converges for a restricted class of linear quadratic regularization terms, but in general it tends to diverge, particularly when a bad initial $\theta^0$ is chosen. Rather than modifying the basic MSA to control its divergent behavior, we will follow the approach presented by Chen et al. [Che+18], who take the crucial step on deriving the maximum principle by not regularizing the network, i.e. not considering the second part of the cost functional in the control problem (4.26), but only considering the terminal cost (loss) function. In the optimal control literature this is known as the Mayer problem [Lib12, p. 87].

## 4.4 Neural Ordinary Differential Equations: Continuous Backpropagation

The most recent (continuous) dynamical systems approach to machine learning presented by Chen et al. [Che+18] considers optimizing the following control problem when finding efficient schemes for training Neural ODEs:

$$\begin{aligned} &\min_{\theta \in \mathcal{U}} L(\mathbf{x}(t_N)) \\ &\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \theta, t), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \quad t_0 \le t \le t_N. \end{aligned} \tag{4.34}$$

Note that this optimization problem is formulated in accordance with the simplifications of the section (4.2). We have summarized the control problems considered throughout chapter (4) in table (1).

| Cost functional | Corresponding Control Problem |
|---|---|
| General cost functional | $\min_{\theta \in \mathcal{U}} L(\mathbf{x}(t_N), t_N) + \int_{t_0}^{t_N} R(\mathbf{x}(t), \theta(t), t)dt$ <br> $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \theta(t), t), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \quad t_0 \leq t \leq t_N$ (4.26) |
| No terminal cost term $L$ | $\min_{\theta \in \mathcal{U}} \int_{t_0}^{t_N} R(\mathbf{x}(t), \theta(t))dt$ <br> $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \theta(t)), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \quad t_0 \leq t \leq t_N$ (4.9) |
| No regularization term $R$ | $\min_{\theta \in \mathcal{U}} L(\mathbf{x}(t_N))$ <br> $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \theta, t), \quad \mathbf{x}(t_0) = \mathbf{x}_0, \quad t_0 \leq t \leq t_N$ (4.34) |

Table 1: Versions of the cost functional and the corresponding control problem. Note that control problem (4.5) is also contained in the first control problem (4.26) as a special case, with its cost functional including both the terminal cost and the regularization term.

In order to give the necessary optimality conditions for control problem (4.34), Chen et al. [Che+18] start by defining a function

$$\mathbf{p}(t) = \frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}(t))$$

for the costate vector $\mathbf{p}$ (cf. equation (4.25) of theorem (4.7)). Since we are not considering the regularization term in the cost functional of (4.34), that is, $R(\mathbf{x}, \theta, t) \equiv 0$, the Hamiltonian corresponding to problem (4.7) is

$$H(\mathbf{x}, \mathbf{p}, \theta, t) = \langle \mathbf{p}, f(\mathbf{x}, \theta, t) \rangle.$$

Then, from the optimal control theory, we should expect that the dynamics of the function $\mathbf{p}(t)$ are given by the following differential equation:

$$\dot{\mathbf{p}}(t) = -\frac{\partial H}{\partial \mathbf{x}}(\mathbf{x}(t), \mathbf{p}(t), \theta, t) = -\left(\frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}(t), \theta, t)\right)' \mathbf{p}(t). \tag{4.35}$$

Chen et al. [Che+18] present the most succinct proof to (4.35) by noting that the costate vector is the gradient of the loss function with respect to the hidden state at a specified time $t$, and that in standard neural networks, the gradient of a hidden layer $\mathbf{x}_t$ depends on the gradient from the next layer $\mathbf{x}_{t+1}$ by chain rule

$$\frac{\partial L}{\partial \mathbf{x}_t} = \left(\frac{\partial \mathbf{x}_{t+1}}{\partial \mathbf{x}_t}\right)' \frac{\partial L}{\partial \mathbf{x}_{t+1}}.$$

Now, the transformation after an $\epsilon$ change in time for a continuous hidden state can be written as

$$\mathbf{x}(t+\epsilon) = \mathbf{x}(t) + \int_t^{t+\epsilon} f(\mathbf{x}(t), \theta, t)dt = T_\epsilon(\mathbf{x}(t), t)$$

and then by applying the chain rule

$$\frac{\partial L}{\partial \mathbf{x}(t)} = \left(\frac{\partial \mathbf{x}(t+\epsilon)}{\partial \mathbf{x}(t)}\right)' \frac{\partial L}{\partial \mathbf{x}(t+\epsilon)} \quad \text{or} \quad \mathbf{p}(t) = \left(\frac{\partial T_\epsilon}{\partial \mathbf{x}}(\mathbf{x}(t), t)\right)' \mathbf{p}(t+\epsilon). \tag{4.36}$$

Then the proof of (4.35) below follows simply from the definition of the derivative and equation (4.36) on the first line, and Taylor series around $\mathbf{x}(t)$ on the second line

$$\begin{aligned}
\dot{\mathbf{p}}(t) &= \lim_{\epsilon \to 0^+} \frac{\mathbf{p}(t+\epsilon) - \mathbf{p}(t)}{\epsilon} = \lim_{\epsilon \to 0^+} \frac{\mathbf{p}(t+\epsilon) - \left(\frac{\partial T_\epsilon}{\partial \mathbf{x}}(\mathbf{x}(t), t)\right)' \mathbf{p}(t+\epsilon)}{\epsilon} \\
&= \lim_{\epsilon \to 0^+} \frac{\mathbf{p}(t+\epsilon) - \left[\frac{\partial}{\partial \mathbf{x}}(\mathbf{x}(t) + \epsilon f(\mathbf{x}(t), \theta, t) + \mathcal{O}(\epsilon^2))\right]' \mathbf{p}(t+\epsilon)}{\epsilon} \\
&= \lim_{\epsilon \to 0^+} \frac{\mathbf{p}(t+\epsilon) - \left(I + \epsilon \frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}(t), \theta, t) + \mathcal{O}(\epsilon^2)\right)' \mathbf{p}(t+\epsilon)}{\epsilon} \\
&= \lim_{\epsilon \to 0^+} \frac{-\epsilon \left(\frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}(t), \theta, t)\right)' \mathbf{p}(t+\epsilon) + \mathcal{O}(\epsilon^2)}{\epsilon} \\
&= \lim_{\epsilon \to 0^+} -\left(\frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}(t), \theta, t)\right)' \mathbf{p}(t+\epsilon) + \mathcal{O}(\epsilon) = -\left(\frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}(t), \theta, t)\right)' \mathbf{p}(t).
\end{aligned}$$

Equation (4.36) pinpoints the similarity between the PMP and backpropagation. Hence, similarly to backpropagation, the ODE for the costate function $\mathbf{p}(t)$ needs to be solved backwards in time. By specifying the constraint on the terminal time point, which is simply the gradient of the loss function with respect to the terminal state, we can obtain

the gradients with respect to the hidden state at any time, including the initial state

$$\begin{aligned}\mathbf{p}(t_N) &= \frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}(t_N)) \\ \mathbf{p}(t_0) &= \mathbf{p}(t_N) + \int_{t_N}^{t_0} \dot{\mathbf{p}}(t)dt = \mathbf{p}(t_N) - \int_{t_N}^{t_0} \left(\frac{\partial f}{\partial \mathbf{x}}(\mathbf{x}(t), \theta, t)\right)' \mathbf{p}(t)dt.\end{aligned} \tag{4.37}$$

Throughout it was assumed that the loss function $L$ depends only on the last time point $t_N$. If function $L$ depends also on intermediate time points $t_1, t_2, ..., t_{N-1}$, the procedure (4.37) can be applied for each of the intervals $[t_{N-1}, t_N], [t_{N-2}, t_{N-1}]$ in the backward order and then the obtained gradients are summed up [Che+18].

### 4.4.1 Gradients with respect to $\theta$ and $t$

Chen et al. [Che+18] generalize (4.35) to obtain gradients of the loss function with respect to $\theta$, and the initial and end times, $t_0$ and $t_N$.[13] Since they do not assume direct dependence of $\theta$ on $t$, they formulate the variables $\theta$ and $t$ as states with constant differential equations

$$\frac{\partial \theta(t)}{\partial t} = \mathbf{0}, \quad \frac{dt(t)}{dt} = 1.$$

Then they adjoin these additional states to the state $\mathbf{x}$ to form an augmented state with corresponding differential equation and costate function

$$\begin{aligned}\frac{d}{dt}\begin{bmatrix}\mathbf{x}\\ \theta \\ t\end{bmatrix}(t) = f_{aug}([\mathbf{x}, \theta, t]) := \begin{bmatrix} f([\mathbf{x}, \theta, t]) \\ \mathbf{0} \\ 1 \end{bmatrix} \\ \mathbf{p}_{aug} := \begin{bmatrix}\mathbf{p}\\ \mathbf{p}_\theta \\ \mathbf{p}_t\end{bmatrix}, \mathbf{p}_\theta(t) := \frac{\partial L}{\partial \theta(t)}, \mathbf{p}_t(t) := \frac{\partial L}{\partial t(t)}.\end{aligned} \tag{4.38}$$

The augmented ODE formulated in this way is as an autonomous (time-invariant) ODE, but the derivations of the previous section (4.4) are again true since this is a special case of a time-variant ODE [Che+18]. The Jacobian matrix of $f$ has the form

$$\frac{\partial f_{aug}}{\partial [\mathbf{x}, \theta, t]} = \begin{bmatrix} \left(\frac{\partial f}{\partial \mathbf{x}}\right)' & \left(\frac{\partial f}{\partial \theta}\right)' & \left(\frac{\partial f}{\partial t}\right)' \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix}(t). \tag{4.39}$$

[13] Note that $L$ does not depend on $\theta, t_0$ and $t_N$ directly but through the trajectory $\mathbf{x}(t)$ that the control $\theta$ generates. Hence, all derivatives appearing in subsequent derivations are assumed to exist.

Plugging this into (4.35) gives

$$\dot{\mathbf{p}_{aug}}(t) = -\left(\frac{\partial f_{aug}}{\partial[\mathbf{x}, \theta, t]}(t)\right)' \begin{bmatrix}\mathbf{p}(t) & \mathbf{p}_\theta(t) & \mathbf{p}_t(t)\end{bmatrix}' = -\begin{bmatrix}\left(\frac{\partial f}{\partial \mathbf{x}}\right)'\mathbf{p} & \left(\frac{\partial f}{\partial \theta}\right)'\mathbf{p} & \left(\frac{\partial f}{\partial t}\right)'\mathbf{p}\end{bmatrix}'(t).$$

Note that the first element is the differential equation (4.35), as expected. Then, Chen et al. [Che+18] use the second element (cf. condition *ii)* in theorem (4.10)) to obtain the gradient of the loss with respect to the parameters $\theta$, by setting $\mathbf{p}_\theta(t_N) = \mathbf{0}$ and integrating over the full interval

$$\frac{\partial L}{\partial \theta} = \mathbf{p}_\theta(t_0) = -\int_{t_N}^{t_0}\left(\frac{\partial f}{\partial \theta}(\mathbf{x}(t), \theta, t)\right)'\mathbf{p}(t)dt. \tag{4.40}$$

Finally, they also get gradients with respect to $t_0$ and $t_N$

$$\frac{\partial L}{\partial t_N} = f(\mathbf{x}(t_N), \theta, t_N)'\mathbf{p}(t_N), \quad \frac{\partial L}{\partial t_0} = \mathbf{p}_t(t_N) - \int_{t_N}^{t_0}\left(\frac{\partial f}{\partial t}(\mathbf{x}(t), \theta, t)\right)'\mathbf{p}(t)dt. \tag{4.41}$$

Equations (4.35), (4.37), (4.40) and (4.41) give the gradients of the loss with respect to all possible inputs to an ODE solver, without backpropagating through the operations of the solver. These results are summarized in algorithm (2). Not storing any intermediate quantities of the forward pass allows us to train models with constant memory cost as a function of depth, a major bottleneck of training deep models [Che+18].

**Algorithm 2** Complete reverse-mode derivative of an ODE initial value problem, [Che+18, Algorithm 2]

**Input:** dynamics parameters $\theta$, start time $t_0$, stop time $t_1$, final state $\mathbf{x}(t_1)$, terminal cost gradient $\frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}(t_1))$

$\frac{\partial L}{\partial t_1} = f(\mathbf{x}(t_1), \theta, t_1)'\frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}(t_1))$

$s_0 = \left[\mathbf{x}(t_1), \frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}(t_1)), \mathbf{0}_{|\theta|}, -\frac{\partial L}{\partial t_1}\right]$ ▷ Define initial augmented state

**def** aug_dynamics($[\mathbf{x}(t), \mathbf{p}(t), \cdot, \cdot], \theta, t$): ▷ Define dynamics on augmented state

  **return** $\left[f(\mathbf{x}(t), \theta, t), -\left(\frac{\partial f}{\partial \mathbf{x}}\right)'\mathbf{p}(t), -\left(\frac{\partial f}{\partial \theta}\right)'\mathbf{p}(t), -\left(\frac{\partial f}{\partial t}\right)'\mathbf{p}(t)\right]$ ▷ Compute vector-Jacobian products

$\left[\mathbf{x}(t_0), \frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}(t_0)), \frac{\partial L}{\partial \theta}, \frac{\partial L}{\partial t_0}\right] = \text{ODESolve}(s_0, \text{aug\_dynamics}, t_1, t_0, \theta)$ ▷ Solve reverse-time ODE

**return** $\frac{\partial L}{\partial \mathbf{x}}(\mathbf{x}(t_0)), \frac{\partial L}{\partial \theta}, \frac{\partial L}{\partial t_0}, \frac{\partial L}{\partial t_1}$

## 4.5 Approximation limitations of Neural Ordinary Differential Equations

So far we have studied the optimal control theory to find optimization methods that can successfully train continuous-depth models. We tried to draw concise mathematical links between such models and optimal control theory, and this approach was finalized with the application of the maximum principle for training Neural Ordinary Differential Equations. In this section, we focus on the approximation capabilities of the novel architecture of Neural ODEs. Dupont et al. [DDT19] were the first to highlight their limitations in approximation capabilities, which they attribute to the preserving of input topology. Using tools from ODE theory and basic topology, they find simple classes of functions which Neural ODEs cannot represent.

Dupont et al. [DDT19] consider the case of no controls[14] in the system of ordinary differential equations which we considered in the previous sections

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), t), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{4.42}$$

To guarantee existence and uniqueness of solutions for (4.42), they weaken the assumptions of the previous sections by requiring $f$ to be continuous in $t$ and globally Lipschitz continuous in $\mathbf{x}$ [AA15, Theorem 2.4.5], that is, there exists a constant $L \geq 0$ such that

$$\|f(\mathbf{x}_2(t), t) - f(\mathbf{x}_1(t), t)\|_2 \leq L \|\mathbf{x}_2(t) - \mathbf{x}_1(t)\|_2$$

for all $t \in \mathbb{R}$. Then they define the flow $\phi_t$ associated to the vector field $f(\mathbf{x}(t), t)$ as the solution at time $t$ of the ODE starting from the initial condition $\mathbf{x}(t_0) = \mathbf{x}_0$. The flow at the final time $t_N$ to which the ODE is solved is defined as features $\phi(\mathbf{x}_0) = \phi_{t_N}(\mathbf{x}_0)$. Finally, the Neural ODE model is defined by applying a linear map $\mathcal{L} : \mathbb{R}^n \to \mathbb{R}$ to the feature function $\phi$. Next Dupont et al. [DDT19] state a useful property of the flow $\phi_t$.

**Lemma 4.11.** *[You10, Theorem C.7]*
*For all $t \in [t_0, t_N], \phi_t : \mathbb{R}^n \to \mathbb{R}^n$ is a homeomorphism.*

In order for $\phi_t$ to be a homeomorphism, it must be a continuous bijection, whose inverse $\phi_t^{-1}$ is continuous. The proof that $\phi_t$ and $\phi_t^{-1}$ are continuous relies on the Gronwall's Lemma (see appendix (A.3)). Bijection on the other hand follows from the fact that the solution of the ODE initial value problem is unique [AA15, Theorem 2.4.5].

[14] The parameters $\theta$ are encoded within the function $f$.

Lemma (4.11) implies that $\phi(\mathbf{x}_0) = \phi_{t_N}(\mathbf{x}_0)$ is a homeomorphism as well. Since homeomorphims preserve topological properties, then Neural ODEs learn only features which have the same topology as the input space by continuously deforming the input space and cannot for instance tear a connected region apart. Concretely, Dupont et al. [DDT19] construct a simple function $g$ which Neural ODEs cannot represent.
Denote by $\phi(S) = \{\mathbf{y} \in \mathbb{R}^n : \mathbf{y} = \phi(\mathbf{x}), \mathbf{x} \in S\}$ the feature transformation of a set $S \subset \mathbb{R}^n$. Denote by $A = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_2 \leq r_1\}$ a sphere and by $B = \{\mathbf{x} \in \mathbb{R}^n : r_2 \leq \|\mathbf{x}\|_2 \leq r_3\}$ an annulus region in $\mathbb{R}^n$.

**Proposition 4.12.** *[DDT19, Proposition2]*
*Neural ODEs cannot represent a function $g : \mathbb{R}^n \to \mathbb{R}$ given by*

$$\begin{cases} g(\mathbf{x}) = -1 & \text{if } \mathbf{x} \in A \\ g(\mathbf{x}) = 1 & \text{if } \mathbf{x} \in B \end{cases}$$

*where $A$ and $B$ are the sphere and the annulus defined above for $0 < r_1 < r_2 < r_3$.*

*Proof.* First note that the sphere $A$ in enclosed by the annulus region $B$. Next assume that there exist a Neural ODE such that the linear map $\mathcal{L}$ maps $\phi(A)$ to $-1$ and $\phi(B)$ to $+1$, i.e. $\phi(A)$ and $\phi(B)$ are linearly separable. Define a disk $D \subset \mathbb{R}^n$ by $D = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\|_2 \leq r_2\}$. Obviously, $A \subset \text{int}(D), A \cap \partial D = \emptyset$ and $\partial D \subset B$. Hence, if $\phi(\text{int}(D))$ and $\phi(\partial D)$ are not linearly separable, then neither are $\phi(A)$ or $\phi(B)$.
Since the feature transformation $\phi$ is a homeomorphism, then $\phi(\text{int}(D)) = \text{int}(\phi(D))$ and $\phi(\partial D) = \partial(\phi(D))$ [Arm13]. Denote by $\overline{D} = \phi(D)$ and assume that $\text{int}(\overline{D})$ and $\partial\overline{D}$ are linearly separated. That is, there exists a linear function $\mathcal{L}(\mathbf{x}) = \mathbf{w}'\mathbf{x}$ and a constant $C$ such that $\mathcal{L}(\mathbf{x}) > C$ for all $\mathbf{x} \in \partial\overline{D}$, and $\mathcal{L}(\mathbf{x}) < C$ for all $\mathbf{x} \in \text{int}(\overline{D})$.
Now, since $\overline{D}$ is a connected subset of $\mathbb{R}^n$ ($D$ is connected and $\phi$ is a homeomorphism), then every point $\mathbf{x} \in \text{int}(\overline{D})$ can be expressed as a convex combination of points on the boundary $\partial\overline{D}$, that is, any $\mathbf{x} \in \text{int}(\overline{D})$

$$\mathbf{x} = \lambda\mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2$$

for some $\mathbf{x}_1, \mathbf{x}_2 \in \partial\overline{D}$ and $0 < \lambda < 1$. But,

$$\begin{aligned} \mathcal{L}(\mathbf{x}) &= \mathbf{w}'\mathbf{x} = \mathbf{w}'(\lambda\mathbf{x}_1 + (1 - \lambda)\mathbf{x}_2) \\ &= \lambda\mathbf{w}'\mathbf{x}_1 + (1 - \lambda)\mathbf{w}'\mathbf{x}_2 \\ &\geq \lambda C + (1 - \lambda)C = C, \end{aligned}$$

which means that all points of the interior are on the same side of the hyperplane as points on the boundary. Thus $\phi(A)$ and $\phi(B)$ being linearly separable cannot possibly hold. □

Dupont et al. [DDT19] then experiment with training Neural ODEs to approximate function $g$ in dimensions 1 and 2. Note that when optimizing Neural ODEs we actually train on discrete points which are sampled from the continuous regions, and not the whole region. Hence it is still possible for Neural ODEs to learn good approximations of function $g$ since the flow could squeeze through the gaps. However, through experiments, they show that such problems usually lead to ill-posed ODE problems that are numerically expensive to solve [DDT19].

In order to deal with such ill-posed problems, Dupont et al. [DDT19] introduce Augmented Neural ODEs (ANODEs), as a straightforward extension of Neural ODEs. They instead augment the learning space and solve the ODE from $\mathbb{R}^n$ to $\mathbb{R}^{n+p}$, which allows the ODE flow to lift points into the additional dimensions such that linear separability is easily satisfied. Concretely, by denoting $\mathbf{a}(t) \in \mathbb{R}^p$ a point in the augmented part of space, they formulate the augmented ODE problem in the following way:

$$\frac{d}{dt}\begin{bmatrix}\mathbf{x}(t)\\ \mathbf{a}(t)\end{bmatrix}(t) = f(\begin{bmatrix}\mathbf{x}(t)\\ \mathbf{a}(t)\end{bmatrix}, t), \qquad \begin{bmatrix}\mathbf{x}(t_0)\\ \mathbf{a}(t_0)\end{bmatrix} = \begin{bmatrix}\mathbf{x}_0\\ \mathbf{0}\end{bmatrix}.$$

Dupont et al. [DDT19] achieve lower losses, lower computational cost and better generalization than the regular Neural ODEs, both on learning the previous function $g$ and on image datasets such as MNIST [LeC98] and CIFAR-10 [KNH10]. Their experiments follow the architecture proposed by Chen et al. [Che+18] of modelling $f$ as a CNN [LKF10] or an MLP [Pin99] with weights that are not a function of time, and instead the time dependency is encoded by passing a concatenated tensor $(\mathbf{x}(t), t)$ as input to the network.

# 5 Stability of Neural Ordinary Differential Equations

In this chapter we start by highlighting the inconsistency of neural networks theoretically, then we discuss the robustness of the Neural ODE model when compared to other conventional neural network architectures such as CNNs.

In the first section (5.1), we review studies on the phenomenon of adversarial examples for neural networks. Despite the success of deep neural networks in image classification, an observation first made by Szegedy et al. [Sze+13] shows that deep neural networks exhibit unstable behavior to small changes in the input. In this context, it is possible to perturb input images in a manner that the changes are indistinguishable for the human eye, but the label assigned by the network is very different. This sheds doubt on the neural network generalization capabilities and the ability to learn the true underlying concepts that determine the correct output label. Next, in the second section (5.2), we report the empirical success of various adversarial examples for Neural ODE models. We examine a property of ODE trajectories which ensures intrinsic regularization in Neural ODE networks. Finally, in section (5.3), we experiment with adversarial deformations, achieved by small deformations to the image found through a gradient descent step. We apply such attacks to Neural ODEs on MNIST and compare our results to CNNs.

## 5.1 Adversarial Examples

In image classification problems a neural network finds a mapping $\mathcal{K} : \mathbb{R}^n \to \{1, \cdots, L\}$, which maps image pixel value vectors to a discrete label set $\mathcal{Y}$ containing $L$ integer classes. An adversarial example for an image classifier $\mathcal{K}$ with respect to a correctly classified image $\mathbf{x} \in \mathbb{R}^n$ is an image $\tilde{\mathbf{x}} \in \mathbb{R}^n$ that is imperceptible from $\mathbf{x}$ but is classified differently, $\mathcal{K}(\mathbf{x}) \neq \mathcal{K}(\tilde{\mathbf{x}})$. Proximity in $\ell_p$-norm (for $1 \leq p \leq \infty$) is considered to be a sufficient substitute for imperceptibility.

Much research has since gone into developing algorithms to construct adversarial examples (or so-called adversarial attacks) [GSS14], [KGB16], [MFF16], [CW17], [Hua+17], [Mad+18] and the recent surveys [Pap+16a], [AM18]. These results suggest that classifiers based on deep learning techniques, even those who obtain excellent performance on the test set, might not have understood concepts in the way humans would. Even though adversarial examples are very unlikely to occur in practice [Sze+13], [Dhi+18], [Ily+19],

their existence is very relevant and crucial for deep learning to achieve the maturity to enter into safety and security critical applications.

### 5.1.1 Adversarial Perturbations

Szegedy et al. [Sze+13] were the first to highlight the instability of neural networks by constructing adversarial examples through corruption of training samples with additive noise. They formulate the problem of finding adversarial examples, or so-called adversarial perturbations, in form of the following optimization problem:

$$\begin{aligned} &\min \|\mathbf{r}\|_2 \\ &\mathcal{K}(\mathbf{x}+\mathbf{r}) = l \\ &\mathbf{x}+\mathbf{r} \in [0,1]^n \end{aligned} \tag{5.1}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a given image, whose true label is different from $l$. The optimization problem (5.1) is non-convex and, in general, its exact computation is a hard problem, because the constraint $\mathcal{K}(\mathbf{x}+\mathbf{r}) = l$ is highly non-linear. Instead, Szegedy et al. approximate it by using a box-constrained L-BFGS [LN89] to find the minimum $c > 0$ for which the minimizer $\mathbf{r}^*$ of the following optimization problem satisfies $\mathcal{K}(\mathbf{x}+\mathbf{r}^*) = l$

$$\begin{aligned} &\min c \cdot \|\mathbf{r}\|_2 + J(\theta, \mathbf{x}+\mathbf{r}, l) \\ &\mathbf{x}+\mathbf{r} \in [0,1]^n \end{aligned}$$

where $J(\theta, \mathbf{x}, l)$ denotes a continuous loss function of a neural network with parameters $\theta$ evaluated at an input $\mathbf{x}$ and a label $l$.

Based on the ideas of Szegedy et al. [Sze+13], Carlini et al. [CW17] construct adversarial examples by extending the optimization problem (5.1) for either $\ell_0, \ell_2$ or $\ell_\infty$-norms. Their aim is to express the optimization problem in a different form which is better suited for gradient-based methods that do not support box constraints. To achieve that, they define an objective function $f$ such that

$$\mathcal{K}(\mathbf{x}+\mathbf{r}) = l \text{ iff } f(\mathbf{x}+\mathbf{r}) \leq 0. \tag{5.2}$$

One of the many choices that the authors propose for $f$ is

$$f(\mathbf{x}+\mathbf{r}) = -J(\theta, \mathbf{x}+\mathbf{r}, l) + 1$$

where $J$ denotes the cross-entropy loss. For other choices of $f$ and testing the respective condition (5.2), we refer to the original work [CW17]. Having defined the function $f$, problem (5.1) can alternatively be formulated as

$$\begin{aligned} &\min \|\mathbf{r}\|_p + c \cdot f(\mathbf{x}+\mathbf{r}) \\ &\mathbf{x}+\mathbf{r} \in [0,1]^n \end{aligned} \tag{5.3}$$

where $c > 0$ is a suitable chosen constant. Carlini et al. use three methods, namely: projected gradient descent, clipped gradient descent and change of variables to avoid the box constraints in problem (5.3). Then they use the Adam optimizer [KB14] to find adversarial examples which are very successful even on distilled neural networks [Pap+16b] (i.e. neural networks that have undergone defensive distillation strategy to increase their robustness towards adversarial examples which were reported in literature prior to the work of Carlini et al. [CW17]).

Note that until now we considered targeted adversarial examples, that is, given an input image $\mathbf{x} \in \mathbb{R}^n$ and a target $l \in \mathcal{Y}$ different from the true label, such attacks try to find an image $\tilde{\mathbf{x}} \in \mathbb{R}^n$ such that $\mathcal{K}(\tilde{\mathbf{x}}) = l$ while $\mathbf{x}$ and $\tilde{\mathbf{x}}$ are close according to $\ell_p$-norm distance. Another method of constructing adversarial examples discussed in literature is via untargeted adversarial examples, whose objective is to construct an image $\tilde{\mathbf{x}} \in \mathbb{R}^n$ such that $\mathcal{K}(\mathbf{x}) \neq \mathcal{K}(\tilde{\mathbf{x}})$ and $\mathbf{x}, \tilde{\mathbf{x}}$ are close.

### 5.1.2 Fast Gradient Sign Method

Goodfellow et al. [GSS14] propose the fast gradient sign method (FGSM) to compute adversarial examples when maximizing the difference between the classifier's output with a fixed $\ell_\infty$-norm. We note that this method supports both targeted adversarial examples and untargeted ones. First, they linearize the cost function $J$ around the value of parameters $\theta$ and formulate the problem of finding adversarial examples as

$$\mathbf{r} = \underset{\|\mathbf{r}\|_\infty < \epsilon}{\arg\max}\ J(\theta, \mathbf{x}+\mathbf{r}, t) \tag{5.4}$$

for an image $\mathbf{x}$ of true label $t$ and an $\epsilon > 0$ which controls the magnitude of the perturbation. Then they use a first order approximation for the non-convex problem (5.4) [GSS14]

$$\mathbf{r} = \underset{\|\mathbf{r}\|_\infty < \epsilon}{\arg\max}\ J(\theta, \mathbf{x}, t) + \mathbf{r}'\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, t) = \underset{\|\mathbf{r}\|_\infty < \epsilon}{\arg\max}\ \mathbf{r}'\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, t)$$

which gives the maximal perturbation

$$\mathbf{r} = \epsilon\, \mathrm{sign}(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, t)).$$

In case of a targeted attack, FGSM performs a single gradient descent step

$$\tilde{\mathbf{x}} = \mathbf{x} - \epsilon \operatorname{sign}(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, l)).$$

We must note that FGSM was primarily designed to be fast since the gradient $\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, t)$ can be efficiently computed via backpropagation. As for optimality, FGSM is not meant to always produce the minimal adversarial perturbations. Among many conclusions in [GSS14] is also the non-uniqueness of adversarial examples, that is, an example generated for one model is often misclassified by other models, even when they have different architectures or were trained on disjoint training sets. They attribute this property to the classifiers being of linear nature.

### 5.1.3 Projected Gradient Descent

As a straightforward extension of FGSM, Kurakin et al. [KGB16] propose projected gradient descent (PGD). This more powerful, multi-step approach applies FGSM multiple times with smaller step size $\alpha$, and clips pixel values of intermediate results after each step to ensure they are in an $\epsilon$-$\ell_\infty$-neighbourhood of the original image [Mad+18]

$$\tilde{\mathbf{x}}_0 = \mathbf{x}, \quad \tilde{\mathbf{x}}_{n+1} = \operatorname{Clip}_{B_\epsilon^\infty(\mathbf{x})}\{\tilde{\mathbf{x}}_n + \alpha \operatorname{sign}(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, t))\}$$

where

$$\left(\operatorname{Clip}_{B_\epsilon^\infty(\mathbf{x})}\{\tilde{\mathbf{x}}\}\right)_i = \begin{cases} x_i - \epsilon & \text{for } \tilde{x}_i < x_i - \epsilon \\ x_i + \epsilon & \text{for } \tilde{x}_i > x_i + \epsilon \\ \tilde{x}_i & \text{otherwise} \end{cases}$$

for every $i \in [n]$. PGD was found to show more promising results than FGSM [KGB16].

### 5.1.4 DeepFool

DeepFool [MFF16] is an untargeted attack technique (which can be easily modified for the targeted setting) developed by Dezfooli et al. with the aim of finding an efficient yet accurate algorithm for searching minimum perturbations with respect to the $\ell_p$-norm ($1 \leq p \leq \infty$). For simplicity, we will discuss the algorithm for the case $p = 2$.

Dezfooli et al. [MFF16] develop the algorithm by imagining that neural networks are totally linear, with a hyperplane separating each class from another. From this, they construct adversarial perturbations as the orthogonal projection onto the closest hyperplane.

Then, since neural networks are not actually linear, they propose an iterative procedure where on each step the orthogonal projection onto the first-order approximation of these hyperplanes is computed.

For some space of images $\mathcal{X}$ let $F = (F_1, \cdots, F_L) : \mathcal{X} \to \mathbb{R}^L$ be the underlying model for the classifier $\mathcal{K} : \mathcal{X} \to \{1, \cdots, L\}$, such that

$$\mathcal{K}(\mathbf{x}) = \arg\max_{k \in [L]} F_k(\mathbf{x}).$$

If we fix an image $\mathbf{x} \in \mathcal{X}$ of true label $t$, a target label $l$, and set $f_l := F_l - F_t$, then our goal is to find a small perturbation $\mathbf{r} \in \mathcal{X}$ such that $f_l(\mathbf{x} + \mathbf{r}) \geq 0$. Assuming that $F$ is differentiable at $\mathbf{x}$, we can approximate $f_l$ around $\mathbf{x}$ with

$$f_l(\mathbf{x} + \mathbf{r}) \approx f_l(\mathbf{x}) + \mathbf{r}' \nabla f_l(\mathbf{x}).$$

Assuming $\nabla f_l(\mathbf{x}) \neq \mathbf{0}$, then the orthogonal projection onto the hyperplane $\{\mathbf{x} : f_l(\mathbf{x}) = 0\}$ is given by

$$\mathbf{r} = -\frac{f_l(\mathbf{x})}{\|\nabla f_l(\mathbf{x})\|_2^2} \nabla f_l(\mathbf{x}). \tag{5.5}$$

We note that such an $\mathbf{r}$ satisfies $\mathbf{r}' \nabla f_l(\mathbf{x}) = -f_l(\mathbf{x})$. Hence, for $\mathbf{r} \in \mathcal{X}$ given by (5.5) such that $\|\mathbf{r}\|_2 = f_l(\mathbf{x}) / \|\nabla f_l(\mathbf{x})\|_2$ is small enough we have

$$f_l(\mathbf{x} + \mathbf{r}) \approx 0.$$

This means that the classifier $\mathcal{K}$ has approximately equal confidence for the perturbed image $\mathbf{x} + \mathbf{r}$ to have either label $t$ or $l$.
The DeepFool algorithm [MFF16] then is formulated in the iterative fashion

$$\mathbf{x}_0 := \mathbf{x}, \;\; \mathbf{x}_n = \mathbf{x} + \mathbf{r}_1 + ... + \mathbf{r}_n$$

for $\mathbf{r}$'s updated according to (5.5) and in every step the target label is chosen to minimize the $\|\mathbf{r}\|_2$, that is

$$l_n = \arg\min_{k \neq t} \frac{|f_k(\mathbf{x}_{n-1})|}{\|\nabla f_k(\mathbf{x}_{n-1})\|_2^2}.$$

The algorithm terminates if $\arg\max_k F_k(\mathbf{x}^n) \neq t$ and outputs $\tilde{\mathbf{x}} = \mathbf{x}_n$. If it converges to a point $\tilde{\mathbf{x}}$ on a decision boundary hyperplane of $F$, then the total perturbation $\tilde{\mathbf{r}} = \mathbf{r}_1 + ... + \mathbf{r}_n$ is multiplied by a constant $1 + \eta$, so that the final image $\tilde{\mathbf{x}} = \mathbf{x} + (1 + \eta)\tilde{\mathbf{r}}$ is moved away

from the decision boundary hyperplane.

In [MFF16], DeepFool shows superior results to FGSM [GSS14] when finding minimal perturbations (the average perturbation obtained using DeepFool is as much as 5 times lower than the one estimated with FGSM), which makes DeepFool more accurate in detecting directions that can potentially fool neural networks. However, the authors note that the algorithm operates in a greedy way and is not guaranteed to converge to the minimal perturbation. Furthermore, the DeepFool algorithm does not impose any box constraints which may result in a perturbed image of invalid pixel values, as opposed to FGSM or PGD who overcome this with constrained search of perturbations. In the next section we will see a more recent approach of deformation attacks introduced by Alaifari et al. [AAG18] which naturally overcomes this problem.

### 5.1.5 Adversarial Deformations

Alaifari et al. [AAG18] extend the DeepFool algorithm by proposing the ADef algorithm, which instead of using an additive perturbation iteratively deforms an input image using gradient descent steps.

Considering an input image $\xi : [0,1]^2 \to \mathbb{R}^c$ (with $c = 1$ or $c = 3$ for grayscale or color images), Alaifari et al. [AAG18] model a deformation with respect to a vector field $\tau : [0,1]^2 \to \mathbb{R}^2$ as

$$\xi^\tau(u) = \xi(u + \tau(u)), \quad \forall u \in [0,1]^2$$

where $\xi$ is set to zero outside of $[0,1]^2$. This transformation embeds as special cases many other image transformations. For instance, translations are deformations with respect to the constant vector field $\tau = v \in \mathbb{R}$. The deformation $\xi - \xi^\tau$ is not small in general in $\ell_p$ norm even if the two images look visually indistinguishable [AAG18]. This questions the construction of adversarial examples and the design of defense strategies which has been conducted so far in the context of small additive noise $\mathbf{r}$ measured in the $\ell_p$-norm.

Alaifari et al. [AAG18] consider the discrete setting where the deformations are implemented in the following way. An image $\mathbf{x}$ in some space of images $\mathcal{X}$, is represented as a function $\xi : [0,1]^2 \to \mathbb{R}^c$ which is evaluated on a regular grid $\{1/(W+1), ..., W/(W+1)\}^2 \subseteq [0,1]^2$. Such a function $\xi$ can be obtained by interpolation from $\mathbf{x}$, that is

$$\mathbf{x}_{k,i,j} = \xi_k(i/(W+1), j/(W+1))$$

for $k \in [c]; i, j \in [W]$. Further, $\tau : [0,1]^2 \to \mathbb{R}^2$ is evaluated on the same grid. For simplicity, only vector fields that do not move points on the grid $\{1, ..., W\}^2$ outside of $[1, W]^2$ are considered

$$T := \{\tau : \{1, ..., W\}^2 \to \mathbb{R}^2 \mid \tau(i,j) + (i,j) \in [1, W]^2,\ \forall i, j \in [W]\}.$$

The deformation of $\mathbf{x}$ with respect to $\tau$ is defined as

$$\mathbf{x}_k^\tau = \xi_k \circ \left(\frac{\mathrm{Id} + \tau}{W+1}\right) \quad \text{or} \quad \mathbf{x}_{k,i,j}^\tau = \xi_k \left(\frac{(i,j) + \tau(i,j)}{W+1}\right) \tag{5.6}$$

for $k \in [c]; i, j \in [W]$. The following $T$-norm is used as a proxy to capture the visual difference between the original and the deformed image

$$\|\tau\|_T := \max_{i,j \in [W]} \|\tau(i,j)\|_2.$$

Similarly to section (5.1.4), for a given image $\mathbf{x}$ in some space of images $\mathcal{X}$, with true label $t$ and target label $l \in [L]$, we define the function $f_l = F_l - F_t : \mathcal{X} \to \mathbb{R}$. Assuming that $\mathbf{x}$ does not lie on a decision boundary hyperplane, then $f_l(\mathbf{x}) < 0$. Next, define a function $g : T \to \mathbb{R},\ \tau \mapsto f_l(\mathbf{x}^\tau)$. Note that,

$$g(0) = f_l(\mathbf{x}) < 0.$$

Alaifari et al. [AAG18] seek to find a small vector field $\tau \in T$ such that $g(\tau) \geq 0$. For that, they use a linear approximation of $g$ in a neighbourhood of 0 as

$$g(\tau) \approx g(0) + (\mathrm{D}_0 g)\tau$$

where $\mathrm{D}_0 g : T \to \mathbb{R}$ is the derivative of $g$ evaluated at $\tau = 0$. Hence, if $\tau$ is small in the $T$-norm and it satisfies $(\mathrm{D}_0 g)\tau = -g(0)$, then classifier $\mathcal{K}$ has approximately equal confidence for the deformed image $\mathbf{x}^\tau$ to have either label $t$ or $l$. The authors then employ least-squares to solve the equation

$$(\mathrm{D}_0 g)\tau = -g(0) \tag{5.7}$$

whose solution is given by [AAG18]

$$\tau = -\frac{f_l(\mathbf{x})}{\sum_{i,j=1}^W |\mathrm{D}_0 g(i,j)|^2} \mathrm{D}_0 g \tag{5.8}$$

where $\mathrm{D}_0 g$ is the discrete vector field given by

$$\mathrm{D}_0 g(i,j) = \frac{1}{W+1} \sum_{k=1}^c \left(\nabla f_l(\mathbf{x})\right)_{k,i,j} \cdot \nabla \xi_k \left(\frac{(i,j)}{W+1}\right).$$

Finally, the deformed image $\mathbf{x}^\tau \in \mathcal{X}$ is defined using equation (5.6). Moreover, Alaifari et al. [AAG18] impose some degree of smoothness on the deforming vector field. For this, they search in the range of a smoothing operator $\mathcal{S} : T \to T$. Concretely, they apply a componentwise two-dimensional Gaussian filter (of any standard deviation) to the entries of $\mathrm{D}_0 g$ in equation (5.8). The new vector field will again satisfy equation (5.7) since $\mathcal{S}$ is self-adjoint. This results in a smooth deformation of the image $\mathbf{x}$. For a rigorous derivation of the model we refer to [Gau17].

As in DeepFool [MFF16], the ADef algorithm iterates the deformation process by setting

$$\mathbf{x}_0 := \mathbf{x}, \;\; \mathbf{x}_n = \mathbf{x}_{n-1} \circ (\mathrm{Id} + \tau_n)$$

where for $n \geq 1$, the vector field $\tau_n$ is given by (5.8) for $\mathbf{x}_{n-1}$. At every step the target label $l$ is chosen among a candidate set of $m$ labels according to

$$l_n = \operatorname*{arg\,min}_{k \in [m]} \|\tau_k\|_T.$$

The algorithm terminates when the deformed image is misclassified and outputs $\tilde{\mathbf{x}} = \mathbf{x}_n$. The iteration also terminates if $\tilde{\mathbf{x}}$ lies on a decision boundary hyperplane of $\mathcal{K}$, in which case the total deforming vector field is approximated by $\tilde{\mathbf{r}} = \mathbf{r}_1 + ... + \mathbf{r}_n$ and then multiplied by a constant $1 + \eta$ so that the final image $\tilde{\mathbf{x}} = \mathbf{x} \circ (\mathrm{Id} + (1 + \eta)\tilde{\mathbf{r}})$ is moved away from the decision boundary hyperplane.

Interestingly, the ADef algorithm [AAG18] concentrates the deforming vector field $\tau$ on the edges of the image $\mathbf{x}$, due to $\tau$ taking small values wherever $\xi$ has a small derivative, provided that $\nabla f$ is moderate. Further, the result of a deformation is always an image of valid pixel values as opposed to DeepFool. The adversarial deformations of Alaifari et al. [AAG18] fool convolutional neural networks [LKF10] on MNIST [LeC98] and a pre-trained Inception-v3 (or Resnet-101) [Sze+16], [He+16a] on ImageNet [Rus+15] with very high success rate.

## 5.2 On Robustness of Neural Ordinary Differential Equations

So far in chapter (5) we have examined theoretically the inconsistency of neural networks in general. We presented several attack strategies for fooling neural networks, grouped into two general categories: adversarial perturbations and adversarial deformations. In this section we review empirical and theoretical results on robustness of Neural ODEs [Che+18] for image classification tasks with respect to adversarial perturbations. The

results of this section and the next section (5.3) on generalization of Neural ODEs are a continuity of optimization perspectives and approximation capabilities of Neural ODEs which we studied earlier in chapter (4). Concretely, we report the results of Yan et al. [Yan+19], who expose Neural ODEs to inputs of various types of perturbations and then measure the sensitivity of the corresponding outputs.

Yan et al. [Yan+19] experiment with random Gaussian perturbations [Sze+13], FGSM adversarial examples [GSS14] and PGD adversarial examples [Mad+18]. First, such methods are examined on Neural ODE [Che+18] and CNN [LKF10] models trained only on original, non-perturbed images, and their success is defined as a percentage of perturbed images, whose original image is correctly classified. In order to have a fair comparison, they experiment with Neural ODEs and CNNs which share the same architecture, with the exception of the former model containing an additional channel which represents the time $t$. This in turn implies that the number of parameters in each model is close to each other. In this setting, Yan et al. [Yan+19] achieve superior robustness of Neural ODEs when compared to CNNs both on MNIST [LeC98], a subset of ImageNet [Rus+15] and SVHN [Net+11] datasets.

In the second setting, Yan et al. [Yan+19] train independent copies of the networks using the adversarial training procedure proposed by Madry et al. [Mad+18]. That is, input images are perturbed with random Gaussian noise before each step of the training process. Training a model on original images together with their perturbed versions provides increased robustness against adversarial perturbations. This training technique builds on the ideas of Goodfellow et al. [GSS14], who hypothesise a linear behavior of neural networks. By noting that linear models cannot be constant near the input samples and at the same time assign different labels to different inputs, Goodfellow et al. [GSS14] point the need to tweak the training rather than the architecture of neural networks for better robustness. As a defence strategy against adversarial perturbations, they propose training based on the following loss

$$\tilde{J}(\theta, \mathbf{x}, t) = \alpha J(\theta, \mathbf{x}, t) + (1 - \alpha) J(\theta, \mathbf{x} + \epsilon \operatorname{sign}(\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, t)), t)$$

for $\alpha \in (0, 1)$. We note that, in general, this procedure is computationally expensive. Moreover, there is a trade-off between the adversarial robustness and the accuracy on original non-perturbed images [YGZ18], [Tsi+18].
However, considering the fact Gaussian noise is usually assumed to be present in images and synthesizing it is computationally easy, Yan et al. [Yan+19] perturb the original

images with Gaussian noise. Next, they follow the adversarial training procedure both on Neural ODE and CNN models. Again, Neural ODEs show by far more robustness than the counterpart of CNNs. These observations are probably one step further to strengthen the conjecture of Goodfellow et al. [GSS14], who propose designing more powerful optimization methods that are able to train models whose behavior is more locally stable.

Finally, Yan et al. [Yan+19] provide an intuitive understanding of the robustness of Neural ODEs based on the following theorem from ODE theory.

**Theorem 5.1** (ODE trajectories do not intersect)**.** *[You10, Proposition C.6]*
*Let* $\mathbf{x}_1(t)$ *and* $\mathbf{x}_2(t)$ *be two solutions of the system of ordinary differential equations*

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), t), \quad \mathbf{x}(t_0) = \mathbf{x}_0 \tag{5.9}$$

*with different initial conditions* $\mathbf{x}_1(t_0) \neq \mathbf{x}_2(t_0)$*. Then, for all* $t \in [0, \infty)$*,* $\mathbf{x}_1(t) \neq \mathbf{x}_2(t)$*.*

For simplicity, Yan et al. [Yan+19] demonstrate theorem (5.1) in the 1-dimensional case. Denote by $x_i(t)$ the solution of equation (5.9) starting from some point $A_i = (t_0, x_i(t_0))$ for $i \in [3]$ and assume that $A_1$ is between $A_2$ and $A_3$. Theorem (5.1) implies that the trajectory $x_1(t)$ lies always between the trajectories $x_2(t)$ and $x_3(t)$.
Next, let $\epsilon < \min\{|x_2(t_0) - x_1(t_0)|, |x_3(t_0) - x_1(t_0)|\}$ and denote by $\tilde{x}_1(t)$ a solution of equation (5.9) starting from the point $\tilde{A}_1 = (t_0, \tilde{x}_1(t_0))$, which is in an $\epsilon$-neighbourhood of $A_1$ with $|\tilde{x}_1(t_0) - x_1(t_0)| < \epsilon$. Applying theorem (5.1) gives

$$|\tilde{x}_1(t_N) - x_1(t_N)| \leq |x_3(t_N) - x_2(t_N)|.$$

For Neural ODEs this means that if a correctly classified input image is slightly perturbed, the trajectory associated to its perturbed version would not change too much from the original image. Therefore, the perturbed input could still be correctly classified. For a CNN model on the other hand there is no such bound on the deviation from the original output [Yan+19]. In other words, there exists intrinsic robustness regularization in Neural ODE networks, which is not present for CNN networks.

## 5.3 Adversarial Deformations for Neural Ordinary Differential Equations

In this section, we experiment with adversarial deformations [AAG18] for classification tasks which have undergone Neural ODE architecture for training [Che+18]. We report, for the first time, the ADef success rate for Neural ODE networks trained on original, non-perturbed images of the MNIST [LeC98] test set.

### 5.3.1 Experimental settings

Neural ODEs preserve the dimension of the input, however, in image classification tasks, we model maps that transform high-dimensional inputs into a categorical output. For that, the ODE-Net architecture of Chen et al. [Che+18] consists of three key parts. First, we map a high-dimensional input to a multi-channel feature map. Next, the Neural ODE serves as the non-linear representation mapping. Finally, the vector consisting of predictions is generated via applying a fully-connected classifier to the output of the Neural ODE.

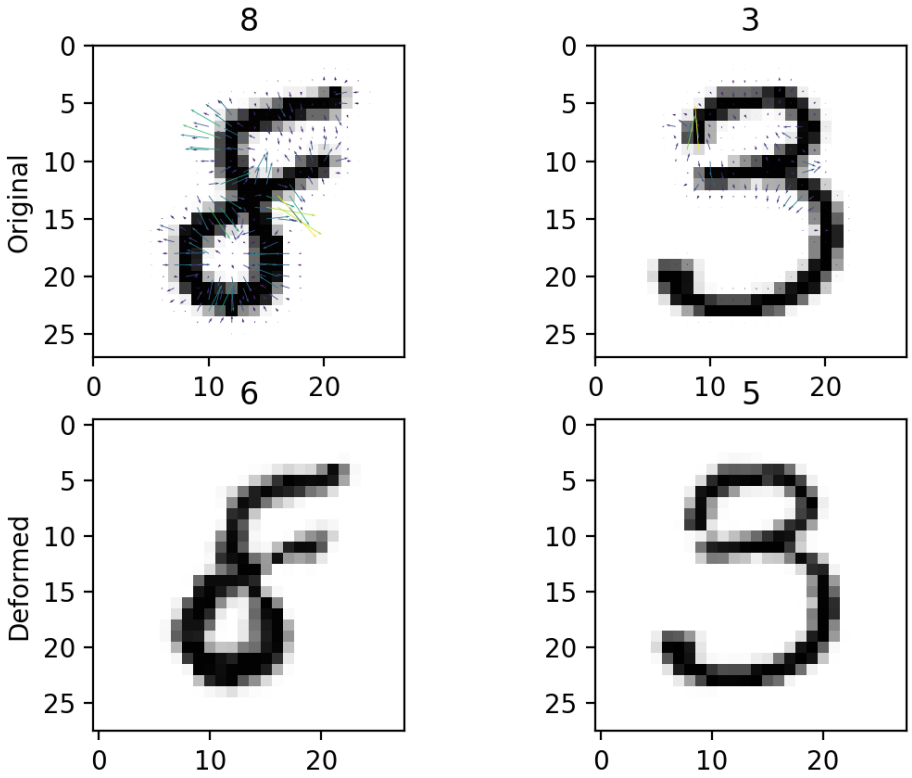


Figure 6: Adversarial deformations for ODE-Net. First row: Original images from the MNIST test set. Second row: The deformed images.

Concretely, we have considered the following experimental settings:

**Dataset:** We conduct experiments to evaluate the robustness of ODE-Net models with respect to adversarial deformations on MNIST test set.

**Architecture:** The ODE-Net model is a sequential model consisting of four convolutional layers (two downsampling layers and two feature layers) and one fully-connected layer

at the end. In practice, Chen et al. [Che+18] propose solving the Neural ODE with various numerical solvers such as the Euler method and the Runge-Kutta methods. In our experiments we use the easily-implemented Euler method with step size 0.1. Our implementation builds on the open-source Neural ODE codes: https://github.com/rtqichen/torchdiffeq.

**Training:** We train the model only on original, non-perturbed images. Training the model on original images together with their deformed version turned out to be computationally very inefficient, as opposed to adversarial perturbation training. We wish to study this phenomenon and experiment with adversarial deformation training in future work.

**Deformations:** We use the ADef algorithm, as coded in https://gitlab.math.ethz.ch/tandrig/ADef, to produce adversarial deformations for the original images, such that the ODE-Net model misclassifies the deformed images. The algorithm is configured to produce any label other than the correct label, hence the set of candidate labels includes all incorrect labels. The algorithm performs smoothing by a Gaussian filter of standard deviation 1/2, uses bilinear interpolation to obtain intermediate pixel intensities, and applies an overshoot factor of 1.2 whenever it converges to a decision boundary hyperplane.

In figure (6), we first trained the network using the ODE-Net architecture on MNIST test set, and then we used the ADef algorithm to find the needed deformations such that the same network actually misclassifies the deformed handwritten digits. The vector field corresponding to the deformation is shown on the original image.

### 5.3.2 ADef success for ODE-Net

We report, for the first time, an ADef success rate (7) of 92.4% for ODE-Net models trained on original images of the MNIST test set. This shows, once again, superior robustness results of Neural ODEs compared to CNNs. Indeed, the authors in [AAG18] achieve a success rate of ADef for conventional CNN models on MNIST which is always above 95%.

We motivate the interested reader, in presence of sufficient compute power, to train Neural ODE models with an increased number of epochs for more precision, and whose number of parameters are close to that of their counterpart CNN models.

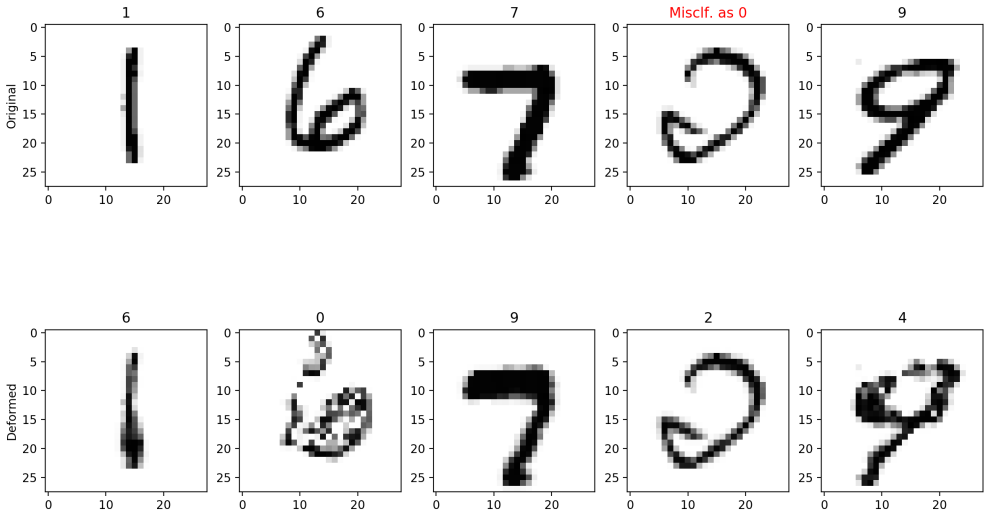


Figure 7: The success rate of ADef for ODE-Net is defined as a percentage of the correctly classified inputs.

The promising results of this chapter further strengthen the conjecture that Neural ODEs are more robust towards adversarial attacks in general compared to CNNs. Achieving good robustness results is of great relevance for Neural ODEs to reach the maturity to enter into real-life applications.

# 6 Conclusion

In the algorithmic stability section we have analyzed the Tikhonov regularization scheme as a means to guarantee uniform stability which in turn implies the ability to learn a function from the given data that generalizes well to unseen cases. Further, we have given emphasis to function classes represented by neural networks. We have learnt that shallow neural networks and thus neural networks in general are universal approximators for different function classes under minimal assumptions on the activation function such as non-polynomiality and local essential boundedness. We have observed that, for certain problems, deep neural networks perform exponentially better over shallow neural networks, in terms of the number of parameters used.

Considering very deep neural networks and parameterizing the derivative of the hidden state using a neural network, we have exploited the first context in which deep neural networks were replaced by continuous dynamical systems. Concretely, we have examined in details the novel model of Neural Ordinary Differential Equations from the perspectives of optimization techniques, approximation capabilities and generalization. We have seen that optimal control theory can be used to obtain optimization algorithms for training continuous-depth models. With our work we have contributed theoretically to a concise derivation of the exact variant, amongst many, of the maximum principle for training continuous-depth models, from which we have obtained the Neural ODE training algorithm as a special case. We have inspected that the property of preserving the input topology is a key aspect to the approximation limitations of Neural ODEs.

From the section of stability we have learnt that neural networks are very brittle to adversarial attacks. We have contrasted various techniques which are used to fool neural networks. However, the promising results of chapter (5) show that Neural ODEs are more robust towards adversarial attacks compared to CNNs. We met our expectations and strengthened this conjecture by testing adversarial deformations for Neural ODEs on MNIST. In future work, it would be interesting to prove theoretically what we discovered in the tests, other than simply exploiting the non-intersecting property of ODE trajectories. Obtaining good robustness results on deep learning architectures is of great relevance for many real-life applications, especially in safety and security critical systems. Robustness combined with other desirable properties such as invertibility and parameter efficiency show great premises for Neural ODEs to new extensions in practice.

# A Appendices

## A.1 Probability Theory Basics

**Lemma A.1** (Hoeffding's inequality). *[Hoe63]*
*Let $Z_1, ..., Z_n$ be real independent random variables whose values are contained in intervals $[a_i, b_i] \supseteq \text{range}[Z_i.]$ Then for every $\epsilon > 0$ it holds that*

$$\mathbb{P}\left[\sum_{i=1}^{n} Z_i - \mathbb{E}[Z_i] \geq \epsilon\right] \leq \exp\left[-\frac{2\epsilon^2}{\sum_{i=1}^{n}(a_i - b_i)^2}\right].$$

**Lemma A.2** (McDiarmid's inequality). *[McD89]*
*Let $(Z_1, ..., Z_n) = Z$ be a finite sequence of independent random variables, each with values in $\mathcal{Z}$ and $\varphi : \mathcal{Z}^n \to \mathbb{R}$ a measurable function such that $\mid \varphi(z) - \varphi(z') \mid \leq \nu_i$ whenever $z$ and $z'$ only differ in the $i$'th coordinate. Then for every $\epsilon > 0$*

$$\mathbb{P}[\varphi(Z) - \mathbb{E}[\varphi(Z)] \geq \epsilon] \leq \exp\left[-\frac{2\epsilon^2}{\sum_{i=1}^{n}\nu_i^2}\right].$$

## A.2 $C_0^\infty([a, b])$

In lemma (3.5) of section (3.1.2) we used the fact that $C_0^\infty([a, b])$ is a complete metric vector space for any $a < b$ in $\mathbb{R}$. For completeness, we provide a metric and a norm for which this is true.
Define a metric $\rho$ on $C_0^\infty([a, b])$ by

$$\rho(\varphi_1, \varphi_2) = \sum_{n=0}^{\infty} 2^{-n} \frac{\|\varphi_1 - \varphi_2\|_n}{1 + \|\varphi_1 - \varphi_2\|_n}$$

where $\|\varphi\|_n = \sum_{j=0}^{n} \sup_{x\in[a,b]} |\varphi^{(j)}(x)|$. Then, $(C_0^\infty([a, b]), \rho)$ is a complete metric space.

## A.3 Gronwall's Lemma

**Theorem A.3** (Gronwall's inequality). *[How98]*
*Let $U \subset \mathbb{R}^n$ be an open set. Let $f : U \times [t_0, t_N] \to \mathbb{R}^n$ be a continuous function and let $\mathbf{x}_1, \mathbf{x}_2 : [t_0, t_N] \to U$ satisfy the systems of ordinary differential equations:*

$$\dot{\mathbf{x}}_1(t) = f(\mathbf{x}_1(t), t), \quad \mathbf{x}_1(t_0) = \mathbf{x}_0$$
$$\dot{\mathbf{x}}_2(t) = f(\mathbf{x}_2(t), t), \quad \mathbf{x}_2(t_0) = \mathbf{x}_0.$$

*Assume there is a constant $L \geq 0$ such that, for all $t \in [t_0, t_N]$,*

$$\|f(\mathbf{x}_2(t), t) - f(\mathbf{x}_1(t), t)\|_2 \leq L \|\mathbf{x}_2(t) - \mathbf{x}_1(t)\|_2.$$

*Then, for any* $t \in [t_0, t_N]$,

$$\|\mathbf{x}_2(t) - \mathbf{x}_1(t)\|_2 \leq e^{Lt} \|\mathbf{x}_2 - \mathbf{x}_1\|_2 .$$

# References

[Ale68] Vladimir V Aleksandrov. *On the accumulation of perturbations in the linear systems with two coordinates.* Vestnik MGU, 3, 1968.

[Sch47] Laurent Schwartz. "Théorie générale des fonctions moyenne-périodiques". In: *Annals of Mathematics*. 1947, pp. 857–929.

[Bel57] Richard Bellman. *Dynamic programming*. NJ: Univ. Press, Princeton, 1957.

[Kel60] Henry J Kelley. "Gradient theory of optimal flight paths". In: *Ars Journal* 30.10 (1960), pp. 947–954.

[VK61] BA Vostretsov and Mikhail Aleksandrovich Kreines. "Approximation of continuous functions by superpositions of plane waves". In: *Doklady Akademii Nauk*. Vol. 140. 6. Russian Academy of Sciences. 1961, pp. 1237–1240.

[Pon+62] Lev Semenovich Pontryagin et al. *The mathematical theory of optimal processes*. New York, NY: Wiley, 1962. URL: http://cds.cern.ch/record/234445.

[Hoe63] Wassily Hoeffding. "Probability Inequalities for Sums of Bounded Random Variables". In: *Journal of the American Statistical Association* 58.301 (Mar. 1963), pp. 13–30.

[AF66] M Athans and PL Falb. *Optimal control: An introduction to the theory and its applications*. New York [u.a.]: McGraw-Hill, 1966.

[Ber74] Leonard David Berkovitz. *Optimal control theory*. Springer Verlag, New York, NY, 1974.

[CL82] Felix L Chernousko and Alexey A Lyubushin. *Method of successive approximations for solution of optimal control problems.* Optimal Control Applications and Methods 3(2):101-114, 1982.

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536.

[GW88] A Ronald Gallant and Halbert White. "There exists a neural network that does not make avoidable mistakes". In: *Proc. of the International Conference on Neural Networks, San Diego*. 1988.

[Cyb89] George Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of control, signals and systems* 2.4 (1989), pp. 303–314.

[Fun89] Ken-Ichi Funahashi. "On the approximate realization of continuous mappings by neural networks". In: *Neural networks* 2.3 (1989), pp. 183–192.

[HSW+89] Kurt Hornik, Maxwell Stinchcombe, Halbert White, et al. "Multilayer feedforward networks are universal approximators." In: *Neural networks* 2.5 (1989), pp. 359–366.

[LN89] DC Liu and J Nocedal. "On the limited memory BFGS method for large scale optimization". In: *Mathematical Programming* 45.3 (1989), pp. 503–528. DOI: 10.1007/BF01589116.

[McD89] Colin McDiarmid. "On the method of bounded differences". In: *Surveys in combinatorics* 141.1 (1989), pp. 148–188.

[Les+93] Moshe Leshno et al. "Multilayer feedforward networks with a nonpolynomial activation function can approximate any function". In: *Neural networks* 6.6 (1993), pp. 861–867.

[Koh+95] Ron Kohavi et al. "A study of cross-validation and bootstrap for accuracy estimation and model selection". In: *Ijcai*. Vol. 14. 2. Montreal, Canada. 1995, pp. 1137–1145.

[BB96] Ralph P Boas and Harold P Boas. *A Primer of Real Functions*. Cambridge University Press, 1996.

[DC96] Harris Drucker and Corinna Cortes. "Boosting decision trees". In: *Advances in neural information processing systems*. 1996, pp. 479–485.

[LGM96] GG Lorentz, M von Golitschek, and Y Makovoz. "Constructive Approximation: Advanced Problems". In: *Grundlehren*. Vol. 304. Springer, Berlin. 1996.

[Alo+97] Noga Alon et al. "Scale-sensitive dimensions, uniform convergence, and learnability". In: *Journal of the ACM (JACM)* 44.4 (1997), pp. 615–631.

[How98] Ralph Howard. "The Gronwall inequality". In: (1998). URL: http://people.math.sc.edu/howard/Notes/gronwall.pdf.

[LeC98] Yann LeCun. "The MNIST database of handwritten digits". In: *http://yann.lecun. com/exdb/mnist/* (1998).

[Vap98] Vladimir N Vapnik. *Statistical learning theory*. New York [u.a.], 1998.

[Mai99] VE Maiorov. "On best approximation by ridge functions". In: *Journal of Approximation Theory* 99.1 (1999), pp. 68–94.

[MP99] Vitaly Maiorov and Allan Pinkus. "Lower bounds for approximation by MLP neural networks". In: *Neurocomputing* 25.1-3 (1999), pp. 81–91.

[Pin99] Allan Pinkus. "Approximation theory of the MLP model in neural networks". In: *Acta numerica* 8 (1999), pp. 143–195.

[BE01] Olivier Bousquet and André Elisseeff. "Algorithmic stability and generalization performance". In: *Advances in Neural Information Processing Systems*. 2001, pp. 196–202.

[AF03] Robert A Adams and John JF Fournier. *Sobolev spaces*. Vol. 140. Elsevier, 2003.

[BP07] Alberto Bressan and Benedetto Piccoli. *Introduction to mathematical control theory*. Philadelphia: AIMS series on applied mathematics, 2007.

[Rao09] Anil V Rao. "A survey of numerical methods for optimal control". In: *Advances in the Astronautical Sciences* 135.1 (2009), pp. 497–528.

[Bot10] Léon Bottou. "Large-scale machine learning with stochastic gradient descent". In: *Proceedings of COMPSTAT'2010*. Springer, 2010, pp. 177–186.

[KNH10] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. "CIFAR-10 (Canadian Institute for Advanced Research)". In: (2010). URL: http://www.cs.toronto.edu/~kriz/cifar.html.

[LKF10] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. "Convolutional networks and applications in vision". In: *Proceedings of 2010 IEEE international symposium on circuits and systems*. IEEE. 2010, pp. 253–256.

[NH10] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[Sha+10] Shai Shalev-Shwartz et al. "Learnability, stability and uniform convergence". In: *Journal of Machine Learning Research* 11.Oct (2010), pp. 2635–2670.

[You10] Laurent Younes. *Shapes and diffeomorphisms*. Vol. 171. Springer, 2010.

[Net+11] Yuval Netzer et al. "Reading digits in natural images with unsupervised feature learning". In: (2011).

[Lib12] Daniel Liberzon. *Calculus of Variations and Optimal Control Theory: A Concise Introduction*. Princeton University Press, 2012. ISBN: 978-0-691-15187-8.

[MRT12] Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012. ISBN: 026201825X, 9780262018258.

[Arm13] Mark Anthony Armstrong. *Basic topology*. Springer Science & Business Media, 2013.

[BS13] J Frédéric Bonnans and Alexander Shapiro. *Perturbation analysis of optimization problems*. Springer Science & Business Media, 2013.

[Sze+13] Christian Szegedy et al. "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199* (2013).

[GSS14] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *arXiv preprint arXiv:1412.6572* (2014).

[KB14] D Kingma and J Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[SB14] Shai Shalev-Shwartz and Shai Ben-David. *Understanding Machine Learning: From Theory to Algorithms*. New York, NY, USA: Cambridge University Press, 2014. ISBN: 1107057132, 9781107057135.

[AA15] Shair Ahmad and Antonio Ambrosetti. *A textbook on ordinary differential equations*. Vol. 88. Springer, 2015.

[Rus+15] Olga Russakovsky et al. "ImageNet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision (IJCV)* 115.3 (2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[Tel15] Matus Telgarsky. "Representation benefits of deep feedforward networks". In: *arXiv preprint arXiv:1509.08101* (2015).

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[He+16a] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[He+16b] Kaiming He et al. "Identity mappings in deep residual networks". In: *European conference on computer vision*. Springer. 2016, pp. 630–645.

[KGB16] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. "Adversarial examples in the physical world". In: *arXiv preprint arXiv:1607.02533* (2016).

[MFF16] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2574–2582.

[Pap+16a] Nicolas Papernot et al. "The limitations of deep learning in adversarial settings". In: *2016 IEEE European symposium on security and privacy (EuroS&P)*. IEEE. 2016, pp. 372–387.

[Pap+16b] N Papernot et al. "Distillation as a defense to adversarial perturbations against deep neural networks". In: *IEEE Symposium on Security and Privacy* (2016).

[Sze+16] Christian Szegedy et al. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[Tel16] Matus Telgarsky. "Benefits of depth in neural networks". In: *arXiv preprint arXiv:1602.04485* (2016).

[CW17] Nicholas Carlini and David Wagner. "Towards evaluating the robustness of neural networks". In: *2017 ieee symposium on security and privacy (sp)*. IEEE. 2017, pp. 39–57.

[Gau17] Tandri Gauksson. "Adversarial perturbations and deformations for convolutional neural networks". MA thesis. ETH Zurich, 2017.

[HR17] Eldad Haber and Lars Ruthotto. "Stable architectures for deep neural networks". In: *Inverse Problems* 34.1 (2017), p. 014004.

[Hua+17] Xiaowei Huang et al. "Safety verification of deep neural networks". In: *International Conference on Computer Aided Verification*. Springer. 2017, pp. 3–29.

[Li+17] Qianxiao Li et al. "Maximum principle based algorithms for deep learning". In: *The Journal of Machine Learning Research* 18.1 (2017), pp. 5998–6026.

[Lu+17] Yiping Lu et al. "Beyond finite layer neural networks: Bridging deep architectures and numerical differential equations". In: *arXiv preprint arXiv:1710.10121* (2017).

[Pas+17] Adam Paszke et al. "Automatic differentiation in pytorch". In: (2017).

[Wei17] E Weinan. "A proposal on machine learning via dynamical systems". In: *Communications in Mathematics and Statistics* 5.1 (2017), pp. 1–11.

[AM18] Naveed Akhtar and Ajmal Mian. "Threat of adversarial attacks on deep learning in computer vision: A survey". In: *IEEE Access* 6 (2018), pp. 14410–14430.

[AAG18] Rima Alaifari, Giovanni S Alberti, and Tandri Gauksson. "ADef: an iterative algorithm to construct adversarial deformations". In: *arXiv preprint arXiv:1804.07729* (2018).

[Cas18] Rui Castro. *2di70-statistical learning theory lecture notes*. 2018.

[Che+18] Tian Qi Chen et al. "Neural ordinary differential equations". In: *Advances in neural information processing systems*. 2018, pp. 6571–6583.

[Dhi+18] Guneet S Dhillon et al. "Stochastic activation pruning for robust adversarial defense". In: *arXiv preprint arXiv:1803.01442* (2018).

[Gra+18] Will Grathwohl et al. "Ffjord: Free-form continuous dynamics for scalable reversible generative models". In: *arXiv preprint arXiv:1810.01367* (2018).

[LJ18] Hongzhou Lin and Stefanie Jegelka. "Resnet with one-neuron hidden layers is a universal approximator". In: *Advances in Neural Information Processing Systems*. 2018, pp. 6169–6178.

[Mad+18] Aleksander Madry et al. "Towards deep learning models resistant to adversarial attacks". In: *International Conference on Learning Representations* (2018). URL: https://openreview.net/forum?id=rJzIBfZAb.

[RH18] Lars Ruthotto and Eldad Haber. "Deep neural networks motivated by partial differential equations". In: *arXiv preprint arXiv:1804.04272* (2018).

[Tsi+18] Dimitris Tsipras et al. "Robustness may be at odds with accuracy". In: *arXiv preprint arXiv:1805.12152* (2018).

[Wol18] Michael M Wolf. *Mathematical foundations of supervised learning*. 2018.

[YGZ18] Ziang Yan, Yiwen Guo, and Changshui Zhang. "Deep defense: Training dnns with improved adversarial robustness". In: *Advances in Neural Information Processing Systems*. 2018, pp. 419–428.

[DDT19] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. "Augmented neural odes". In: *arXiv preprint arXiv:1904.01681* (2019).

[Ily+19] Andrew Ilyas et al. “Adversarial examples are not bugs, they are features”. In: *arXiv preprint arXiv:1905.02175* (2019).

[Liu+19] Xuanqing Liu et al. “Neural SDE: Stabilizing Neural ODE Networks with Stochastic Noise”. In: *arXiv preprint arXiv:1906.02355* (2019).

[Qua+19] Alessio Quaglino et al. “Accelerating neural odes with spectral elements”. In: *arXiv preprint arXiv:1906.07038* (2019).

[Yan+19] Hanshu Yan et al. “On Robustness of Neural Ordinary Differential Equations”. In: *arXiv preprint arXiv:1910.05513* (2019).