Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Theoretically Motivated Neural ODE Architectures for Stable Adjoint Behavior

Research in Data Science

F. Nestaas

May 6, 2022

Advisors: Prof. Dr. A. R. Krause, L. Treven

Department of Computer Science, ETH Zürich

Abstract

Exploding and vanishing gradients are infamous phenomena in the deep learning community. Here, we explore exploding and vanishing gradients in neural ordinary differential equations (neural ODEs, Chen et al. [3]) both from a theoretical and practical view. We prove that enforcing that the behavior of the adjoint, the relevant quantity for exploding and vanishing gradients, in one sense is as stable as possible limits the space of available models greatly. Pragmatically, we introduce restrictions to regular neural ODEs, inspired by the theoretical results, which empirically show more stable adjoint behavior than other neural ODE implementations.

Contents

Chapter 1

Introduction

Exploding and vanishing gradients are infamous phenomena when training neural networks, particularly prominent in e.g. recurrent nerual networks. When talking about exploding gradients, one generally means the adjoint of the system, i.e. how the loss depends on a state z_t at layer t in a neural network. While numerous papers propose architectures which empirically exhibit nice adjoint behavior (Chang et al. [2], Helfrich et al. [5]), theoretical bounds are often exponential (Lin and Wang [8]) and thus have limited applicability.

Recent work (Chang et al. [2]) explores the connections between differential equations and neural networks, in particular by viewing residual recurrent neural networks as discretizations of initial value problems. Adding to this, Chen et al. [3] takes this analogy even further by explicitly learning a differential equation and solving an initial value problem using numerical differential equation solvers, obtaining an output which can be seen as a continuous-time equivalent of a residual recurrent neural network. This architecture is called a neural ODE (ordinary differential equation), and here, we work exlusively with such systems.

In this work, we aim to explore the dynamics of neural ODEs where the adjoint has constant norm, i.e. designing architectures where the adjoint norm is "as stable as possible". We show that this is a hard restriction, but implement architectures motivated by our theoretical results which empirically still exhibit stable adjoint behaviors. In doing so, we find ourselves implementing system dynamics in a form which should be equivalent to regular neural networks, but allowing to impose restrictions e.g. stabilizing the adjoint behavior.

Chapter 2

Theoretical Foundations

Definition 2.1 *Given a neural ordinary differential equation*

$$\begin{aligned}&f_\theta : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d\\ &\dot{z}(t) = f_\theta(z(t), t)\\ &t \in [t_0, t_1]\end{aligned}$$

with loss function L and parameters θ, the adjoint is $a(t) := \frac{dL(z(t_1))}{dz(t)} \in \mathbb{R}^{1\times d}$.

Mostly when people talk about exploding or vanishing gradients, they are actually referring to the adjoint of the system in question, i.e. $\left\| \frac{dL}{dz_t} \right\|^2$ for a state z_t rather than e.g. $\left\| \frac{dL}{d\theta} \right\|^2$. Often, this is in a discrete setting, whereas neural ordinary differential equations reside in continuous time. In the following, we will explore the adjoint for neural ordinary differential equations.

Proposition 2.2 *The time derivative of the adjoint is*

$$\frac{d}{dt}a(t) = -a(t)\frac{\partial}{\partial z}f_\theta(z(t), t).$$

Refer to Chen et al. [3] for a proof of Proposition 2.2.

Lemma 2.3 *If $\frac{\partial}{\partial z} f_\theta(z(t), t)$ is skew-symmetric, then the L_2-norm of the adjoint is constant in time.*

Proof

$$\frac{d}{dt}\frac{1}{2}\left\|a(t)\right\|^2 = \dot{a}(t)a(t)^T = -a(t)\frac{\partial}{\partial z}f_\theta(z(t), t)a(t)^T$$

$$\frac{d}{dt}\frac{1}{2}\left\|a(t)\right\|^2 \in \mathbb{R} \Rightarrow \frac{d}{dt}\frac{1}{2}\left\|a(t)\right\|^2 = \left(\frac{d}{dt}\frac{1}{2}\left\|a(t)\right\|^2\right)^T$$

If $\frac{\partial}{\partial z} f_\theta$ is skew-symmetric, then $\frac{\partial}{\partial z} f_\theta(z(t), t) = -\left(\frac{\partial}{\partial z} f_\theta(z(t), t)\right)^T$. Hence $\forall t \in (t_0, t_1)$

$$\frac{d}{dt}\frac{1}{2}\left\|a(t)\right\|^2 = -\frac{d}{dt}\frac{1}{2}\left\|a(t)\right\|^2 = 0. \qquad \square$$

Lemma 2.3 is sufficient for $\left\|a(t)\right\|^2$ to be constant in time. If we allow $a(t)$ to be any vector in $\mathbb{R}^{1\times d}$, it is also necessary.

Lemma 2.4 *Suppose $f : \mathbb{R}^d \to \mathbb{R}^d$ is a differentiable function such that $\frac{df(x)}{dx} = A(x)$. Then $f(x) = f(0) + B(x)x$, where*

$$B : \mathbb{R}^d \to \mathbb{R}^{d\times d},\ B(x) = \int_0^1 A(sx)ds.$$

Proof Let $G : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d, (s, x) \mapsto f(sx)$. Then G is differentiable in both of its arguments, since f is differentiable, and we obtain

$$\frac{d}{ds}G(s, x) = \left(\left[\frac{d}{dz}f(z)\right]_{z=sx}\right)x = A(sx)x$$
$$\Rightarrow f(x) = G(1, x) = G(0, x) + \int_0^1 \frac{d}{ds}G(s, x)ds = f(0) + \int_0^1 A(sx)xds.$$

Since we do not integrate with respect to any variables on which x depends, we can simplify the integral to $\int_0^1 A(sx)xds = B(x)x$, where $B_{i,j}(x) = \int_0^1 A_{i,j}(sx)ds$

$\square$

From Lemma 2.4 we obtain the following corollary, which will actually play an important role in this work.

Corollary 2.5 *We see that if $A(x)$ is skew-symmetric for every $x \in \mathbb{R}^d$, then $B(x)$ is also skew-symmetric;*

$$B_{i,j}(x) + B_{j,i}(x) = \int_0^1 \underbrace{A_{i,j}(sx) + A_{j,i}(sx)}_{=0} ds = 0.$$

For completeness, this also works in the other direction;

Lemma 2.6 *B is a differentiable skew symmetric matrix $\iff$ $\exists Q : \mathbb{R}^d \to \mathbb{R}^{d\times d}$ where Q is skew-symmetric and $B(x) = \int_0^1 Q(sx)ds$.*

Proof ($\Leftarrow$): This is a direct consequence of Corollary (2.5).

($\Rightarrow$): Given $B : \mathbb{R}^d \to \mathbb{R}^{d\times d}$, $B(x)$ skew-symmetric for any $x \in \mathbb{R}^d$, fix x and define $M_x(t) := B(tx)$. This is skew-symmetric for any t, x, and so is $\frac{d}{dt}M_x(t)$;

$$\begin{aligned}\frac{d}{dt}\left[M_x(t)\right]_{i,j} &= \frac{d}{dt}B_{i,j}(tx)\\ &= \sum_k \left(\frac{d}{dx_k}B_{i,j}(tx)\right)x_k\\ &= -\sum_k \left(\frac{d}{dx_k}B_{j,i}(tx)\right)x_k\\ &= -\frac{d}{dt}(M_x(t))_{j,i}\end{aligned}$$

where in the second line we get the sum since we are taking the total derivative with respect to t. Then we can apply the fundamental theorem of calculus;

$$B(x) - B(0) = \int_0^1 \frac{d}{ds}M_x(sx)ds$$
$$\Updownarrow$$
$$B(x) = \int_0^1 \left(\frac{d}{ds}M_x(sx) + B(0)\right)ds.$$

We see that $\frac{d}{ds}M_x(sx) + B(0)$ is the sum of skew symmetric matrices, and therefore also skew symmetric, which finishes the proof. $\square$

Example 2.7 *It is important that f exists; not every skew-symmetric matrix $A(x)$ has an anti-derivative. Take for example $d = 2$ and*

$$A(x) = \begin{pmatrix} 0 & x_1 \\ -x_1 & 0 \end{pmatrix}.$$

This matrix is skew-symmetric. If it is the jacobian of some function $f : \mathbb{R}^2 \to \mathbb{R}^2$, then

$$\begin{aligned}A_{1,1}(x) &= \frac{d}{dx_1}f_1(x) = 0 \Rightarrow f_1(x) = c(x_2)\\ A_{1,2}(x) &= \frac{d}{dx_2}f_1(x) = x_1 \Rightarrow \frac{d}{dx_2}c(x_2) = x_1\end{aligned}$$

where $c(x_2)$ does not depend on x_1. However, this is not possible in light of the second line, since $\frac{d}{dx_2}c(x_2)$ would have to depend on x_1.

Example 2.8 *Such dynamics also arise rather naturally if we consider recurrent neural networks of the form*

$$z_{t+1} = z_t + hf_\theta(z_t, t).$$

Here, it is useful to think of h as a step-size in a forward-euler discretization of the differential equation $\dot{z}(t) = f_\theta(z(t), t)$. In this case, we define the adjoint as

$$a_t := \frac{dL}{dz_t}$$

and obtain the recursion

$$a_t = a_{t+1}\frac{dz_{t+1}}{dz_t}.$$

Let $A_t := \frac{\partial f_\theta(z_t,t)}{\partial z_t}$. Then the squared norm of the adjoint is

$$\begin{aligned}\|a_t\|^2 &= a_{t+1}\frac{dz_{t+1}}{dz_t}\left(\frac{dz_{t+1}}{dz_t}\right)^T a_{t+1}^T \\ &= a_{t+1}\left(I + hA_t\right)\left(I + hA_t^T\right)a_{t+1}^T \\ &= \|a_{t+1}\|^2 + ha_{t+1}(A_t + A_t^T)a_{t+1}^T + h^2\|a_{t+1}A_t\|^2.\end{aligned}$$

To ensure that the change in norm is small, we could eliminate the terms which are linear in h. In that case, we would choose A_t to be skew-symmetric and obtain that $\{\|a_t\|^2\}_{t=t_0}^{t_1}$ is a decreasing sequence where

$$\|a_t\|^2 = \|a_0\|^2 - h^2\sum_{\tau=0}^{t-1}\|a_{\tau+1}A_\tau\|^2.$$

While this might look like an attractive class of functions, in the following we will see that the class of functions where A_t is skew symmetric is limited.

Lemma 2.9 *Suppose $f : \mathbb{R}^d \to \mathbb{R}^d$ is a twice-differentiable function such that $\frac{df(x)}{dx} = A(x)$ is a skew-symmetric matrix. Then $f(x) = f(0) + Cx$, where $C \in \mathbb{R}^{d\times d}$ is constant and skew-symmetric.*

Proof We have by Lemma 2.4 that $f(x) = f(0) + B(x)x$ where $B(x)$ is skew-symmetric and $\frac{d}{dx_k}(B(x)x)_i = A_{i,k}(x)$. Let $a_{ijk} := \frac{d}{dx_k}A_{i,j}(x)$. Then $a_{ijk} = -a_{jik}(i)$ since A is skew-symmetric for any x, and $a_{ijk} = a_{ikj}(ii)$, since

$$\frac{d}{dx_k}A_{i,j}(x) = \frac{d}{dx_k}\frac{d}{dx_j}(B(x)x)_i = \frac{d}{dx_j}\frac{d}{dx_k}(B(x)x)_i = \frac{d}{dx_j}A_{i,k}(x).$$

We obtain

$$a_{ijk} \overset{(ii)}{=} a_{ikj} \overset{(i)}{=} -a_{kij} \overset{(ii)}{=} -a_{kji}.$$

On the other hand,

$$a_{ijk} \overset{(i)}{=} -a_{jik} \overset{(ii)}{=} -a_{jki} \overset{(i)}{=} a_{kji}.$$

Hence, $a_{ijk} = -a_{ijk} = 0$ for any i, j, k. Therefore, $A_{i,j}$ is constant in x_k for any i, j, k. As such, $B(x)x = Cx$ for a constant $C \in \mathbb{R}^{d\times d}$, and furthermore, $A_{i,j}(x) = C_{i,j}$, so C is skew-symmetric.

□

Example 2.10 *We note at this point that this does not mean that $B(x)$ is necessarily constant. While this may seem counter-intuitive, consider*

$$B(x) = \begin{pmatrix} 0 & x_3 & -x_2 \\ -x_3 & 0 & x_1 \\ x_2 & -x_1 & 0 \end{pmatrix}$$

We see that $B(x)x = 0$ for every $x \in \mathbb{R}^d$, so $\frac{d}{dx}(B(x)x) = 0$, which is skew-symmetric. However, $B(x)$ itself is not constant. Lemma 2.9 shows that there exists a constant matrix C such that for every x, $B(x)x = Cx$. In the present case, we could have chosen $C = 0$ to obtain the same f as we do using $B(x)$.

We can extend Lemma 2.9 to apply to functions of the form $f_\theta(z(t), t)$ by replacing the full derivative $\frac{d}{dx}f(x)$ by the partial derivative $\frac{\partial}{\partial z}f_\theta(z(t), t)$. We can do this since we do not need to consider θ or t when taking partial derivatives with respect to z, and hence treat them as parameters in the above computations. Hence, for any fixed θ and $t \in [t_0, t_1]$, we see that if $\frac{\partial}{\partial z}f_\theta(z(t), t)$ is skew-symmetric, we can write

$$f_\theta(z(t), t) = f_\theta(0, t) + C_\theta(t)z(t).$$

The significance of this result is that C_t does not depend on the state $z(t)$; the trajectory of C_θ is the same regardless of what $z(t)$ is. One implication of this is that if we want a system as described above, we cannot use a neural network that predicts $C_\theta(t)$ based on the current state $z(t)$.

Chapter 3

Methods

3.1 Model Architecture

While in chapter 2 we show that only simple neural ODEs can ensure that the L_2-norm of the adjoint stays constant in time, we might be inspired by Corollary (2.5) to experiment with dynamics of the form

$$\dot{z}(t) = f(z(t), t) = f(0, t) + A(z(t), t)z(t) \tag{3.1}$$

where A is skew-symmetric, and investigate their performance. To further motivate this, we recall from Lemma (2.4) that if we do not restrict A, then we can represent any function $f : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ in the form (3.1). Adding to this, consider the following;

Lemma 3.1 *Given any differentiable $g : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, let $J_g = \frac{\partial g}{\partial z}$, and recall from Lemma 2.4 that we can write $g(z, t) = g(0, t) + B(z, t)z$ for some matrix B. Suppose that $\mathbb{E}\left[J_g(z, t)\right] = 0\ \forall z$, where we take the expectation over model parameters, and that for each i, j, $B_{i,j}$ is i.i.d. and independent of $\frac{\partial}{\partial z_k} B_{i,j}$ for any k. If $A(z, t) = \frac{1}{\sqrt{2}} \int_0^1 \left(J_g(sz, t) - J_g^T(sz, t)\right) ds = \frac{B(z,t) - B^T(z,t)}{\sqrt{2}}$, and $f(z, t) = f(0, t) + A(z, t)z$ then*

$$\mathbb{E}\left[\left(\frac{1}{2}\left(\left[J_f(z, t)\right]_{i,j} + \left[J_f(z, t)\right]_{j,i}\right)\right)^2\right] \leq \mathbb{E}\left[\left(\frac{1}{2}\left(\left[J_g(z, t)\right]_{i,j} + \left[J_g(z, t)\right]_{j,i}\right)\right)^2\right]$$

where we take the expectation over model parameters.

The significance of this is that $v^T Q v = v^T Sym(Q) v$, where $Sym(Q) = \frac{1}{2}\left(Q + Q^T\right)$ is the symmetric part of Q, and here, we bound a matrix which influences the behavior of the adjoint. The lemma shows that in expectation over model parameters, we can decrease the Frobenius norm of $Sym\left(J_{f_\theta}(sz, t)\right)$, which directly influences the adjoint, using particular choices of A in (3.1). The proof of Lemma 3.1 is in the Appendix A.1.1.

Thus, Lemma 3.1 might give us some hope that the adjoint varies less in time if we restrict A to be skew symmetric for dynamics of the form (3.1), than if we do not. A question that is still to be answered, however, is whether such restrictions impact performance in machine learning tasks.

3.1.1 Implementation

We implement all functionality using the JAX framework Bradbury et al. [1]. In particular, for neural networks we use the Equinox framework by Kidger and Garcia [7] and for solving differential equations we use the Diffrax framework by Kidger [6].

Parameter Initialization

When implementing functions of the form 3.1, we need to be careful about parameter initialization. In our implementation, we use a neural network to predict d^2 parameters and reshape the output to a matrix in $\mathbb{R}^{d\times d}$. However, in expectation, the output of the model will have larger variance than if we predict the output vector directly, using d output parameters. To see this, suppose that we predict the matrix $A(z,t) \in \mathbb{R}^{d\times d}$ and compute $A(z,t)z$. Then

$$(A(z,t)z)_i = \sum_j A(z,t)_{i,j} z_j$$

In our implementation, $\mathbb{E}\left[A(z,t)\right] = 0$ as we use the Equinox library (Kidger and Garcia [7]) where the parameter initialization of such neural networks is uniform over a range $[-a,a], a > 0$. Since $A(z,t)_{i,j}$ are i.i.d. upon initialization (since they are initialized in the same way), and if z is non-random, the variance of $(A(z,t)z)_i$ is

$$\mathbb{E}\left[(A(z,t)z)_i^2\right] = \mathbb{E}\left[\sum_j A(z,t)_{i,j}^2 z_j^2\right] = \mathbb{E}\left[A(z,t)_{1,1}^2\right] \sum_j z_j^2 \tag{3.2}$$

On the other hand, had we predicted the output directly, its variance would have been $\mathbb{E}\left[A(z,t)_{1,1}^2\right]$, which is less than in 3.2. In the next chapter, we conduct different experiments, and in the regression experiment in 4.1, we use an implementation where we divide the parameters of the final layer in the neural network predicting $A(z,t)$ by $w\sqrt{d}$, where w is the width of the second to final layer, and the parameters of $f(0,t)$ in 3.1 by $\sqrt{dw}$. While these normalizations are arbitrary, they work in practice. In the classification experiment in 4.2, however, we use a different normalization, and a different activation function, as we will get back to. There, we divide the parameters of the final layer for predicting $A(z,t)$ by $\sqrt{2d}$, since the output is a sum over d variables, which we differentiate, and therefore expect the magnitude of the gradients to be approximately a factor $\sqrt{d}$ larger during back-propagation. Further, this is only half of the output magnitude, since we also predict $f(0,t)$, hence the factor $\sqrt{2}$. As for $f(0,t)$ we divide by $\sqrt{2w}$, where w is the width of the final layer for predicting A, as we can view $f(0,t)$ as a bias (it does not vary with z) and this then corresponds to half of the magnitude of the bias normally in Equinox.

Activation Function

In the two experiments in the next chapter we use different activation functions. In the regression experiment in 4.1, we simply use the identity activation to compute $A(z,t)$ from the final neural network layer. On the other hand, in the classification experiment in 4.2 we use the tanh activation. The reason for this is that in the regression experiment, we could successfully run the experiments on all implementations using the identity activation, while in the classification experiment, using the form 3.1 and not restricting the form of A to be skew-symmetric would sometimes result in a failure. In particular, the ODEs would be too stiff to solve and require too many function evaluations for the Diffrax (Kidger [6]) implementation we use, but using a bounded activation function alleviates this problem[1].

3.1.2 Metrics

We are mainly interested in the adjoint behavior in the following experiments. Thus, we decided to summarize this behavior using the metric $\frac{\hat{q}_{.95}(\|a\|)}{\hat{q}_{.05}(\|a\|)}$, where $\hat{q}_s(\|a\|)$ is the s-quantile of the norm of the adjoint, $\|a\|$, over 100 time points. We use this metric since it is scale-independent, and is large when the maximum and minimum adjoint norms are very different, but less sensitive to outliers than if we had simply divided the maximum and minimum values. Further, we note that the best possible score using this metric is 1, which indicates that, at least within the 0.05 and 0.95 time-quantiles, the adjoint norm is constant.

We also track the number of funciton evaluations (NFE) required to solve the dynamics of the neural ODE during training, and the loss on a validation set in every experiment. We track the NFE one one hand because if this number is large, it could indicate that the neural ODE is more prone to fail

[1]Note that we did not change the regression experiments due to lack of time. However, using the identity activation is still an implementation which one could use to solve regression problems, and therefore, we decided to keep the experiments.

in an experiment by requiring too many function evaluations to solve. Recall from 3.1.1 that this was a problem for neural ODEs of the form 3.1 if when not restricting A. But on the other hand, a large NFE can also indicate that the network is "learning to be deep", as described by Chen et al. [3]. The validation error gives us an idea of what kind of performance to expect.

Chapter 4

Experiments

We test three different models, which are the “Skew Symmetric”, “Unrestricted” and “Regular” models. The first two of these are of the form 3.1, where A is skew-symmetric or an unrestricted (arbitrary) matrix, respectively. The “Regular” model predicts the dynamics directly instead of computing the matrix-vector product in the function form 3.1, which is how one normally implements neural ODEs.

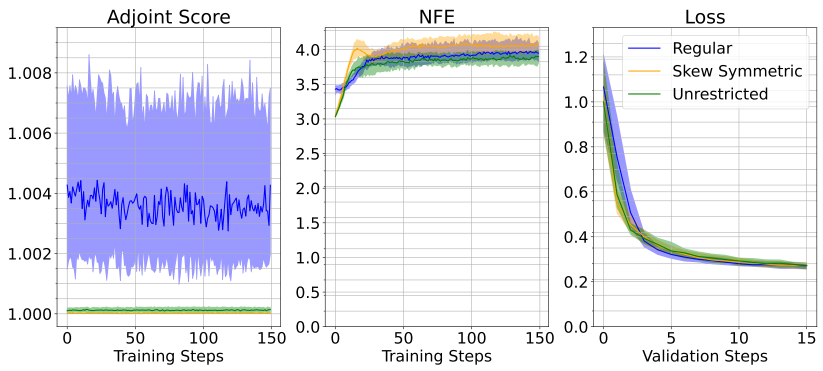

Figure 4.1: Results from the regression experiment. “Regular”, “Unrestricted” and “Skew Symmetric” indicate respectively using a regular neural ODE, where we use a neural network to predict the state dynamics, and using the form (3.1) with aribitrary or skew-symmetric matrices. The plots show the mean of the relevant metric over the batch size in each step. The solid line is the median of this mean over 50 random seeds and the shaded regions regions indicate 25% and 75% quantiles. (Right) Score of the adjoint norm, as described in 3.1.2. (Middle) Number of function evaluations (NFE) required for forward propagation during training. (Left) Loss on the validation set.

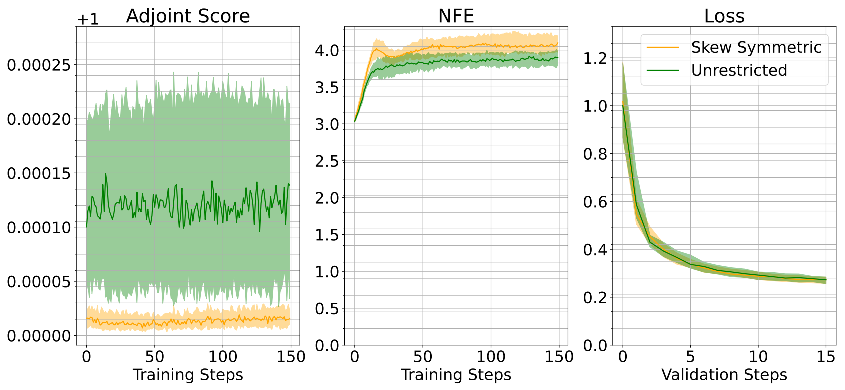

Figure 4.2: Same as figure 4.1 but only comparing "Unrestricted" and "Skew Symmetric".

4.1 Regression Experiment

As a first test, we run regression on a data set with 15 features to get an idea of how architectures of the form (3.1) compare for different choices of the matrix A, and to regular neural ODE architectures. We use the same data set as Zhang et al. [9], taken from Dua and Graff [4], but modify the goal to predict the CO concentration from the other numerical covariates, in order to obtain a regression data set.

Figure 4.1 illustrates the results. We see that all of the tested architectures have similar validation loss, which indicates that restricting the matrix A in (3.1) does not impact performance greatly on this data set. However, the adjoint seems to behave more nicely when choosing this matrix to be skew symmetric, both when compared to choosing any matrix ("Unrestricted") and when not using the form (3.1) at all ("Regular"). In figure 4.2 we compare only the "Unrestricted" and "Skew Symmetric" architectures, and still, the adjoint behaves more nicely in the "Skew Symmetric" case.

It is interesting that "Unrestricted" architecture does not behave the same as the "Regular" architecture, as in light of Lemma 2.4, they should be able to learn the same dynamics. We will see this again in the next experiment, in section 4.2, and we believe that the difference comes from the way we implement functions of the form (3.1). In particular, that the class of realizable functions in our implementation is not the same as the class of functions required to recover the "Regular" dynamics. Since they behave differently, we should also be somewhat careful when comparing the "Regular" and "Skew Symmetric" architectures, as it is not the case that the skew symmetric matrix is directly related to the jacobian of the "Regular" dynamics, which we would need for Lemma 2.4.

However, when comparing only the "Skew Symmetric" and "Unrestricted"

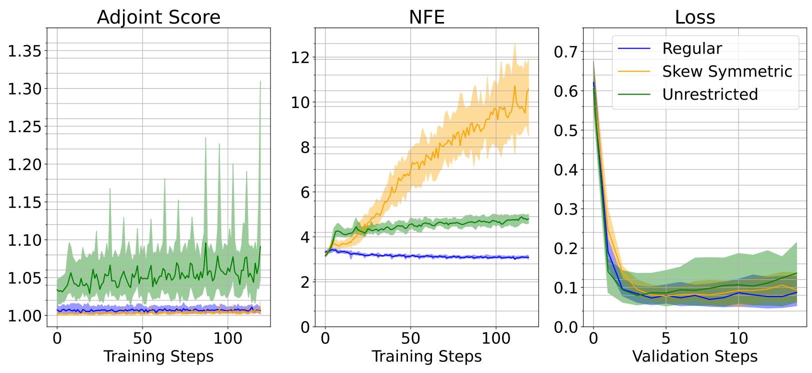

Figure 4.3: Results from the classification experiment. “Regular” and “Skew Symmetric” indicate respectively using a regular neural ODE, where we use a neural network to predict the state dynamics, and using the form (3.1) with aribitrary or skew-symmetric matrices. The plots show the mean of the relevant metric over the batch size in each step. The solid line is the median of this mean over 50 random seeds and the shaded regions indicate 25% and 75% quantiles. (Right) Score of the adjoint norm, as described in 3.1.2. (Middle) Number of function evaluations (NFE) required for forward propagation during training. (Left) Loss on the validation set.

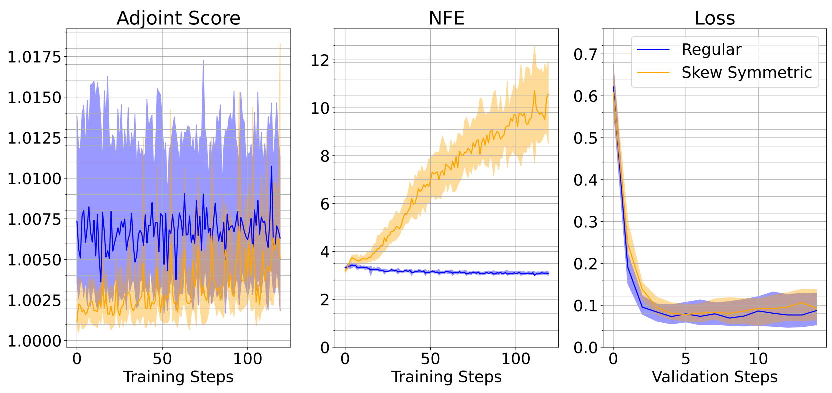

Figure 4.4: Same as figure 4.3 but only comparing “Regular” and “Skew Symmetric”.

architectures in figure 4.2, we see that restricting the matrix A for dynamics of the form (3.1) still results in better adjoint behavior, while not impacting performance significantly. Granted, in this case, the adjoint also does not behave particularly poorly for the “Unrestricted” architecture as the score is still close to 1, indicating a nearly constant adjoint norm, but the “Skew Symmetric” architecture exhibits a better adjoint behavior nonetheless.

4.2 Classification Experiment

We also run a classification experiment on the *Breast Cancer Wisconsin (Diagnostic) Data Set* taken from Dua and Graff [4]. The data set contains 30 features (in addition to the diagnosis, *Malignant* or *Benign* as well as a patient identifier), and we predict a binary label corresponding to the relevant patient's diagnosis. In these experiments, we use the tanh-activation in order to make the "Unrestricted" architecture sufficiently stable (see 3.1.1).

Figure 4.3 shows the results. We see that the adjoint score is best for the skew symmetric architecture. It also seems that using this architecture results in more function evaluations, which might indicate that the way we implement dynamics of the form (3.1) is prone to producing stiff differential equations, which require many function evaluations to solve (see 3.1.2). In terms of the loss, we see that both of the tested architectures achieve similar minimum losses, again indicating that functions of the form 3.1 have comparable performance to the "Regular" architecture.

Interestingly, the "Regular" architecture requires very few, even a decreasing number, of function evaluations to solve in this setting, and the adjoint score is similar to that of the "Skew Symmetric" implementation (see figure 4.4). It is possible that the hyperparameters in this test are particularly favorable for the "Regular" architecture in this setting, especially when considering the results of the regression experiment in 4.1, where all architectures behave more similarly.

Chapter 5

Discussion

We introduce a new architecture for neural ODEs which is theoretically motivated to avoid exploding and vanishing gradients; phenomena well known to the deep learning community. It turns out that restricting the system dynamics f to have the form $f(z,t) = f(0,t) + A(z,t)z$, where A is skew-symmetric, improves the stability of the adjoint norm, a key metric for exploding and vanishing gradients.

While in theory, we should be able to recover any dynamics using this matrix form, in our experiments, there is a discrepancy between using a regular neural ODE and using the matrix form. A possible reason for this is that when we predict the matrix for a matrix-vector multiplication, the resulting functions need not bein the same class of functions as when predicting the vector itself directly. Lemma 2.4 gives us a relationship between the predicted and theoretically correct matrix, but we did not manage to implement an initialization scheme where the matrices behave in the same way. We believe further research is due on how to make neural ODEs in the matrix form behave the same as regular neural ODEs, since imposing the skew-symmetric restriction improved adjoint behavior and did not seem to impact performance greatly, an observation which may still hold in a different implementation and allowing for more stable adjoint behavior.

Another possible direction is making recurrent neural networks based on the approaches taken in this report. We could do this either by using these dynamics for neural controlled differential equations (in fact, we are currently looking into this), or move to the discrete time setting and make recurrent neural networks which employ the same dynamics that we use here. One could argue that such recurrent neural networks more or less exist already, e.g. the so-called OrthoRNN by Helfrich et al. [5]. However, as we show in the Appendix A.1.2, while OrthoRNN and our approach agree when considering one-layer neural networks with element-wise activation functions, our approach allows for more general function classes. This is because we do not assume that the dynamics are of the form $f(z, x) = \sigma(Wz + Qx)$, but rather compositions of such functions. When implementing such recurrent architectures, one could also experiment with different discretization schemes instead of a simple forward-Euler discretization.

Chapter 6

Conclusion

Exploding and vanishing gradients are infamous problems, well known to the deep learning community. It is possible to extend this notion to neural ODEs, and here, we aim to make the behavior of the adjoint, which is the relevant quantity when considering exploding and vanishing gradients, as stable as possible. We prove that restricting the adjoint norm to be constant in time implies simple dynamics but develop a new architecture, inspired by the gained theoretical insights. The architecture performs comparably to other neural ODE implementations, but exhibits more stable adjoint behavior.

Appendix A

Appendix

A.1 Further Theoretical Results

A.1.1 Proof of Lemma 3.1

Lemma A.1 *Given any differentiable $g : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$, let $J_g = \frac{\partial g}{\partial z}$, and recall from Lemma 2.4 that we can write $g(z,t) = g(0,t) + B(z,t)z$ for some matrix B. Suppose that $\mathbb{E}\left[J_g(z,t)\right] = 0 \; \forall z$, where we take the expectation over model parameters, and that for each i, j, $B_{i,j}$ is i.i.d. and independent of $\frac{\partial}{\partial z_k} B_{i,j}$ for any k. If $A(z,t) = \frac{1}{\sqrt{2}} \int_0^1 \left(J_g(sz,t) - J_g^T(sz,t)\right) ds = \frac{1}{\sqrt{2}} \left(B(z,t) - B^T(z,t)\right)$, and $f(z,t) = f(0,t) + A(z,t)z$ then*

$$\mathbb{E}\left[\left(\frac{1}{2}\left(\left[J_f(z,t)\right]_{i,j} + \left[J_f(z,t)\right]_{j,i}\right)\right)^2\right] \leq \mathbb{E}\left[\left(\frac{1}{2}\left(\left[J_g(z,t)\right]_{i,j} + \left[J_g(z,t)\right]_{j,i}\right)\right)^2\right]$$

where we take the expectation over model parameters.

Proof We can compare the function $f(x) = A(x)x$ to $g(x) = B(x)x$ where A is skew symmetric and $B(x) = \int_0^1 J_g(sx)ds$, $J_g(x) = \frac{d}{dx}g(x)$ (i.e. g is some function with jacobian J_g). This is possible by Lemma 2.4. A simple way to do so is to let $A(x) = \alpha(B(x) - B^T(x))$ for $\alpha \in \mathbb{R}$. Then, we can compute the symmetric part of $J_f(x) := \frac{d}{dx}f(x)$ and $J_g(x)$ at indices i, k as

$$\begin{aligned}
\frac{1}{2}\left(\left[J_f(x)\right]_{i,k} + \left[J_f(x)\right]_{k,i}\right) &= \frac{1}{2}\left(\left[A(x)\right]_{i,k} + \left[A(x)\right]_{k,i}\right) + \frac{1}{2}\sum_j (a_{ijk} + a_{kji})x_j \\
&= \frac{1}{2}\sum_j (a_{ijk} + a_{kji})x_j \\
\frac{1}{2}\left(\left[J_g(x)\right]_{i,k} + \left[J_g(x)\right]_{k,i}\right) &= \frac{1}{2}\left(\left[B(x)\right]_{i,k} + \left[B(x)\right]_{k,i}\right) + \frac{1}{2}\sum_j (b_{ijk} + b_{kji})x_j
\end{aligned}$$

where $a_{ijk} = \frac{d}{dx_k} A_{i,j}$ and $b_{ijk} = \frac{d}{dx_k} B_{i,j}$.

Since $\mathbb{E}\left[A_{i,j}(x)\right] = \mathbb{E}\left[B_{i,j}(x)\right] = 0$, also $\mathbb{E}\left[a_{ijk}\right] = \mathbb{E}\left[b_{ijk}\right] = 0$ since x is not random and we can exchange expectations and differentiation. Then also $\mathbb{E}\left[\frac{1}{2}\left(\left[J_g(x)\right]_{i,j} + \left[J_g(x)\right]_{j,i}\right)\right] = \mathbb{E}\left[\frac{1}{2}\left(\left[J_f(x)\right]_{i,j} + \left[J_f(x)\right]_{j,i}\right)\right] = 0$ for any x. Further, as $A_{i,j}$ and hence a_{ijk} are independent for every x, we can then compare the expected magnitudes by considering

$$\begin{aligned}\mathbb{E}\left[\left(\frac{1}{2}\left(\left[J_g(x)\right]_{i,j} + \left[J_g(x)\right]_{j,i}\right)\right)^2\right] &= \mathbb{E}\left[\left(\frac{1}{2}\left([B(x)]_{i,k} + [B(x)]_{k,i}\right) + \frac{1}{2}\sum_j (b_{ijk} + b_{kji})x_j\right)^2\right] \\ &= \mathbb{E}\left[\left(\frac{1}{2}\left([B(x)]_{i,k} + [B(x)]_{k,i}\right)\right)^2\right] + \mathbb{E}\left[\left(\frac{1}{2}\sum_j (b_{ijk} + b_{kji})x_j\right)^2\right].\end{aligned}$$

The analysis for J_f will be the same, but the first term in the above equation is 0 since $[A(x)]_{i,k} + [A(x)]_{k,i} = 0$. We have

$$\sum_j (a_{ijk} + a_{kji})x_j = \alpha \sum_j (b_{ijk} - b_{jik} + b_{kji} - b_{kji}).$$

So if $i \neq k$ and b_{ijk} are i.i.d.

$$\begin{aligned}\mathbb{E}\left[\left(\sum_j (a_{ijk} + a_{kji})x_j\right)^2\right] &= \alpha^2 \mathbb{E}\left[\left(\sum_j (b_{ijk} - b_{jik} + b_{kji} - b_{kji})x_j\right)^2\right] \\ &\overset{(i)}{=} 2\alpha^2 \mathbb{E}\left[\left(\sum_j (b_{ijk} - b_{jik})x_j\right)^2\right] \\ &= 2\alpha^2\left(\mathbb{E}\left[\sum_{j\neq l} (b_{ijk} - b_{jik})(b_{ilk} - b_{lik})x_j x_l\right] + \mathbb{E}\left[\sum_j (b_{ijk} - b_{jik})^2 x_j^2\right]\right) \\ &\overset{(ii)}{=} 2\alpha^2\left(-\mathbb{E}\left[\sum_{j\neq l} (b_{jik}b_{ilk} + b_{ijk}b_{lik})x_j x_l\right] + \mathbb{E}\left[\sum_j (b_{ijk} - b_{jik})^2 x_j^2\right]\right).\end{aligned}$$

In (i) we used that $\mathbb{E}\left[(b_{ijk} - b_{jik})(b_{kji} - b_{kji})\right] = \mathbb{E}\left[b_{ijk} - b_{jik}\right]\mathbb{E}\left[b_{kji} - b_{kji}\right] = 0$ due to the independence of b_{ijk} for any i, j, k, and in (ii) we used that $\mathbb{E}\left[b_{ijk}b_{ilk}\right] = 0$, since $j \neq l$ in the relevant sum. We also have that $b_{jik} \neq b_{ilk}$

since if that were true, then $l = i = j$, which is excluded from the sum, so $\sum_{j \neq l} \mathbb{E}\left[(b_{jik}b_{ilk} + b_{ijk}b_{lik})\right] x_j x_l = 0$. Hence,

$$\begin{aligned}\mathbb{E}\left[\left(\sum_j (a_{ijk} + a_{kji})x_j\right)^2\right] &= 2\alpha^2 \mathbb{E}\left[\sum_j (b_{ijk} - b_{jik})^2 x_j^2\right] \\ &= 2\alpha^2 \mathbb{E}\left[2\sum_{j \neq i} b_{ijk}^2 x_j^2\right].\end{aligned}$$

On the other hand, considering J_g,

$$\begin{aligned}\mathbb{E}\left[\left(\sum_j (b_{ijk} + b_{kji})x_j\right)^2\right] &= \mathbb{E}\left[\sum_{j \neq l} (b_{jik}b_{ilk} + b_{ijk}b_{lik})x_j x_l\right] + \mathbb{E}\left[\sum_j (b_{ijk} + b_{jik})^2 x_j^2\right] \\ &= \mathbb{E}\left[\sum_j (b_{ijk} + b_{jik})^2 x_j^2\right] \\ &= 2\mathbb{E}\left[\sum_j b_{ijk}^2 x_j^2\right].\end{aligned}$$

Hence if we set $\alpha = \frac{1}{\sqrt{2}}$, then the magnitude of the symmetric part of J_f is less or equal to that of J_g.

If on the other hand $i = k$, then

$$\mathbb{E}\left[\left(\sum_j (a_{ijk} + a_{kji})x_j\right)^2\right] = 4\mathbb{E}\left[\left(\sum_j a_{iji}x_j\right)^2\right]$$

and

$$\begin{aligned}\mathbb{E}\left[\left(\sum_j a_{iji}x_j\right)^2\right] &= \alpha^2 \mathbb{E}\left[\left(\sum_j (b_{iji} - b_{jii})x_j\right)^2\right] \\ &= \alpha^2 \mathbb{E}\left[\left(\sum_{j \neq i} (b_{iji} - b_{jii})x_j\right)^2\right] \\ &= 2\alpha^2 \mathbb{E}\left[\sum_{j \neq i} b_{iji}^2 x_j^2\right]\end{aligned}$$

As for J_g, we have

$$\mathbb{E}\left[\left(\sum_j b_{iji}x_j\right)^2\right] = \mathbb{E}\left[\sum_j b_{iji}^2 x_j^2\right] \geq \mathbb{E}\left[\sum_{j\neq i} b_{iji}^2 x_j^2\right]$$

This shows that for any function $g(x)$ there exists an $f(x)$ of the form $f(x) = f(0) + A(x)x$ such that the expected magnitude of the symmetric part of the jacobian is smaller than that of g. We obtain an explicit choice of f by choosing $A(x) = \frac{1}{\sqrt{2}}\int_0^1 \left(J_g(sx) - J_g^T(sx)\right) ds$. $\square$

A.1.2 Relation to OrthoRNNs

Helfrich et. al. [5] proposes to implement recurrent neural networks of the form

$$z_{t+1} = \sigma\left(Wz_t + Qx_{t+1} + b\right)$$

where z_t is the hidden state, x_t is the input at time t, W is guaranteed to be orthogonal, and $\sigma : \mathbb{R}^d \to \mathbb{R}$ is a function applied element-wise.

If we choose σ to be the identity function, compatibe with the bounds presented in Helfrich et. al. [8], and set $Q = 0, b = hb_0$ (which is the setting we consider in neural ODEs), the limit where the time unit goes to 0, this architecture becomes a special case of dynamics of the form (3.1) for a proper choice of W. This choice is setting $W = I + hA + o(h)$ where I is the identity matrix:

$$\begin{aligned} z_{t+h} &= Wz_t + b \\ &= z_t + h\left(\frac{W - I}{h} z_t + \frac{b}{h}\right) \\ &= z_t + h(Az_t + b_0) + o(h) \end{aligned}$$

Enforcing that W is orthogonal means that A is skew symmetric, since

$$I = W^T W = (I + hA)(I + hA^T) + o(h) = I + h(A + A^T) + o(h).$$

Hence, this is in agreement with our result that to preserve a constant adjoint norm, we need $\dot{z}(t) = \lim_{h\to 0} \frac{z(t+h)-z(t)}{h} = Az(t) + b_0$ where A is skew symmetric. However, in our analysis, we did not assume that $z(t+h) = \sigma\left(Wz(t) + Qx_{t+1} + b\right)$, and thus, we present a more general result.

Bibliography

[1] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018.

[2] Bo Chang, Minmin Chen, Eldad Haber, and Ed H. Chi. Antisymmetricrnn: A dynamical system view on recurrent neural networks, 2019.

[3] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations, 2018.

[4] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[5] Kyle Helfrich, Devin Willmott, and Qiang Ye. Orthogonal recurrent neural networks with scaled cayley transform, 2017.

[6] Patrick Kidger. *On Neural Differential Equations*. PhD thesis, University of Oxford, 2021.

[7] Patrick Kidger and Cristian Garcia. Equinox: neural networks in JAX via callable PyTrees and filtered transformations. *Differentiable Programming workshop at Neural Information Processing Systems 2021*, 2021.

[8] David Kewei Lin. urnn : An approach to bounded gradients. 2019.

[9] Shuyi Zhang, Bin Guo, Anlan Dong, Jing He, Ziping Xu, and Song Xi Chen. Cautionary tales on air-quality improvement in beijing. *Proceedings. Mathematical, physical, and engineering sciences*, 473(2205):20170457–20170457, Sep 2017. 28989318[pmid].

ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.