Fast and Stable Algorithms for Deep Learning

Brenda Gonzalez and Andreas Mang

Department of Mathematics, University of Houston, Houston, TX, USA

Background

Task: The development, implementation and assessment of efficient and stable numerical methods for the forward propagation process in deep residual neuronal networks.

Deep Learning Deep learning has evolved to a key technology with numerous applications in the applied sciences, especially in computer vision. It can be used for image classification, face recognition, or image segmentation (see Fig. 1 for an example). The latter will be our primary area of application. Among the most successful architectures in deep learning are deep residual networks.

Residual Networks and Differential Equations There are few rigorous results that provide a solid foundation and guideline of how to train and design deep neuronal networks. Recently, there has appeared interesting work that tries to establish a connection between control theory, the theory of differential equations (both fields that are well established in applied mathematics [5, 6, 8]) and deep learning [1, 2, 3]. This interpretation allows us to establish a ***rigorous mathematical framework*** for the design of deep neural networks. In the present work we will explore this connection.

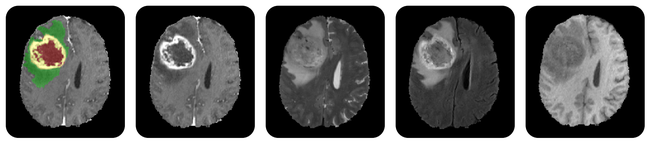


Figure 1*: Exemplary training data for an image segmentation problem. The leftmost image shows the segmentation of a brain tumor (abnormal tissue); red corresponds to the tumor core (necrotic tissue), yellow to contrast enhancement, and green represents the edema. The four images to the right are different magnetic resonance imaging (MRI) datasets (from left to right: T1-weighted MRI with contrast enhancement, fluid attenuation inversion recovery, T2-weighted MRI, and T1-weighted MRI) [7].*

Problem Formulation

Optimal Control Formulation The training problem can be formulated as follows: Given training data consisting of features (or images), $\mathbf{Y}_0$, and the associated labels, $\mathbf{L}$, we seek transformation parameters ($\mathbf{K}$, $\mathbf{u}$) and classification weights ($\mathbf{W}$, $\mathbf{v}$) such that the network predicts the data-label relationship (and generalizes to new data). This problem can be formulated as an optimization problem with a dynamical system as a constraint [2, 3]:

$$\min_{\boldsymbol{\Phi}} \ \text{loss}(g(\mathbf{W}\mathbf{Y}_n + \mathbf{v}), \mathbf{L}) + \text{regularizer}[\boldsymbol{\Phi}] \tag{1a}$$

$$\text{subject to} \ \ \mathbf{Y}_{j+1} = \text{activation}(\mathbf{K}_j\mathbf{Y}_j + \mathbf{u}_j) \tag{1b}$$

with $\boldsymbol{\Phi} := (\mathbf{K}, \mathbf{W}, \mathbf{u}, \mathbf{v})$. The first term, the so-called loss function, measures the discrepancy between the predicted labels $g(\mathbf{W}\mathbf{Y}_n + \mathbf{v})$ and the provided labels $\mathbf{L}$. The second term is a so-called regularization model that is introduced to ensure stability of the estimation of the unknown transformation parameters $(\mathbf{K}, \mathbf{u})$ and classification weights $(\mathbf{W}, \mathbf{v})$ [4]. The equation $\mathbf{Y}_{j+1} = \text{activation}(\mathbf{K}_j\mathbf{Y}_j + \mathbf{u})$ is the constraint of our optimization problem and is typically referred to as the *forward propagation*. We will focus on the forward propagation step.

Forward Propagation For a residual neural network we can represent the forward propagation as

$$\mathbf{Y}_{j+1} = \text{activation}(\mathbf{Y}_j\mathbf{K}_j + \mathbf{u}_j) = \mathbf{Y}_j + f(\mathbf{Y}_j\mathbf{K}_j + \mathbf{u}_j), \quad j = 0, 1, \ldots, n-1.$$

We can interpret this scheme as the numerical time integration of a differential equation [2]. The only modification we have to make is to introduce a time step size $h > 0$:

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + hf(\mathbf{Y}_j\mathbf{K}_j + \mathbf{u}_j), \ \ j = 0, 1, \ldots, n-1. \tag{2}$$

Accordingly, we can view the forward propagation of the residual neural network as an explicit Euler time integration of the following ordinary differential equation (**ODE**)

$$\mathrm{d}_t\mathbf{y}(t) = f(\mathbf{K}^\mathsf{T}(t)\mathbf{y}(t) + \mathbf{u}(t)) \qquad \forall t \in [0,1], \qquad \mathbf{y}(t=0) = \mathbf{y}_0. \tag{3}$$

Methodology

We implemented and tested three different methods for the forward propagation. We replicate test problems available in the literature [2] in order to explore and understand the connection between time integration and forward propagation, and its implications on stability. We present these methods next.

Explicit Euler Method The Euler method corresponds to the discretization in (2) of the ODE in (3). A simple way to stabilize the forward propagation is to use antisymmetric weight matrices [2]. This leads to the forward propagation

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + hf\Big(\frac{1}{2}\mathbf{Y}_j(\mathbf{K}_j + \mathbf{K}_j^\mathsf{T} - \gamma\mathbf{I}) + \mathbf{u}_j\Big), \quad j = 0, 1, \ldots, n-1,$$

which corresponds to the ODE

$$\mathrm{d}_t\mathbf{y}(t) = f(-\frac{1}{2}(\mathbf{K}(t) + \mathbf{K}^\mathsf{T}(t) - \gamma\mathbf{I})\mathbf{y}(t) + \mathbf{u}(t)) \text{ for all } t \in [0,1].$$

Verlet Method Restricting the parameter space to antisymmetric kernels is only one way to obtain a stable forward propagation. Alternatively, we can recast the forward propagation as a Hamiltonian system [2]. With this approach we arrive at a forward propagation step of the form

$$\mathbf{z}_{j+\frac{1}{2}} = \mathbf{z}_{j-\frac{1}{2}} - hf(\mathbf{K}_j^\mathsf{T}\mathbf{y}_j + \mathbf{u}_j) \quad \text{and} \quad \mathbf{y}_{j+1} = \mathbf{y}_j + hf(\mathbf{K}_j\mathbf{z}_{j+\frac{1}{2}} + \mathbf{u}_j)$$

for $j = 0, 1, \ldots, n-1$, which corresponds to the coupled system of ODEs

$$\mathrm{d}_t\mathbf{y}(t) = f(\mathbf{K}(t)\mathbf{z}(t) + \mathbf{u}(t)) \text{ and } \mathrm{d}_t\mathbf{z}(t) = -f(\mathbf{K}^\mathsf{T}(t)\mathbf{y}(t) + \mathbf{u}(t))$$

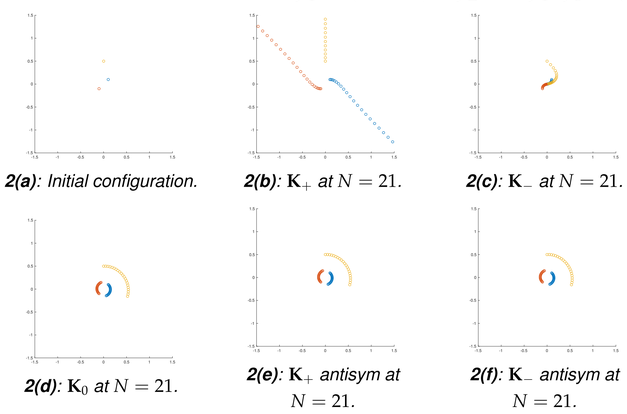
for all $t \in [0,1]$.

Results

Explicit Euler method We consider three networks consisting of $N = 21$ identical layers, i.e., on each layer we use the activation function $f = \tanh$, $h = 0.1$, $b = 0$, and a constant weight matrix. We consider three residual networks parametrized by

$$\mathbf{K}_+ = \begin{pmatrix} 2 & -2 \\ 0 & 2 \end{pmatrix}, \quad \mathbf{K}_- = \begin{pmatrix} -2 & 0 \\ 2 & -2 \end{pmatrix}, \quad \text{and} \quad \mathbf{K}_0 = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$

We consider the feature vectors $\mathbf{y}_1 = (0.1, 0.1)^\mathsf{T}$, $\mathbf{y}_2 = -\mathbf{y}_1$, $\mathbf{y}_3 = (0, 0.5)^\mathsf{T}$.



2(a)*: Initial configuration.* ***2(b)****:* $\mathbf{K}_+$ *at* $N = 21$. ***2(c)****:* $\mathbf{K}_-$ *at* $N = 21$.

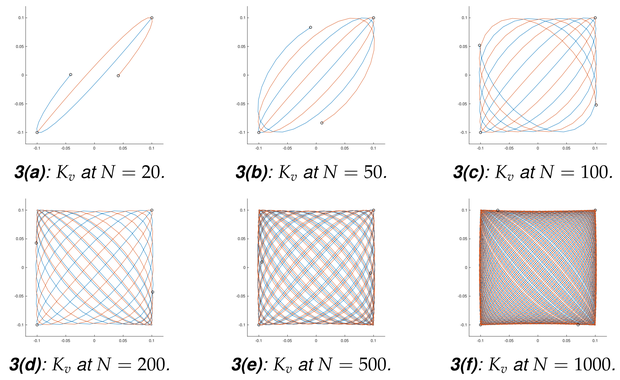
2(d)*:* $\mathbf{K}_0$ *at* $N = 21$. ***2(e)****:* $\mathbf{K}_+$ *antisym at* $N = 21$. ***2(f)****:* $\mathbf{K}_-$ *antisym at* $N = 21$.

Figure 2*: Phase plane diagrams for* $N = 21$ *identical layers using an Euler time integration for the forward propagation. In 2(a) we show the initial configuration of the features* $\mathbf{y}_1$ *(blue),* $\mathbf{y}_2$ *(orange), and* $\mathbf{y}_3$ *(yellow). We show results for the matrices* $\mathbf{K}_+$ *in 2(b),* $\mathbf{K}_-$ *in 2(c), and* $\mathbf{K}_0$ *in 2(d) at layer* $N = 21$*. We also show results for an antisymmetric discretization for* $\mathbf{K}_+$ *in 2(e) and* $\mathbf{K}_-$ *in 2(f).*

Observations: We show the initial configuration of the features in Fig. 2(a) ($\mathbf{y}_1$: blue, $\mathbf{y}_2$: orange, $\mathbf{y}_3$: yellow). In Fig. 2(b) we can see that the features diverge from the origin and one another if we consider $\mathbf{K}_+$. In particular, $\mathbf{y}_1$ and $\mathbf{y}_2$, which were initially near to one another, diverge into opposite directions. This behavior represents an unstable forward propagation, which won't generalize well. In Fig. 2(c), we can see that the features accumulate in the center. These results are for $\mathbf{K}_-$. The differences between the features are annihilated, which leads to difficulties in the learning (the learning problem becomes ill-posed). In Fig. 2(d), the features remain well separated and distances between them are preserved. These results are for the operator $\mathbf{K}_0$. This leads to a stable forward propagation and a well-posed learning problem.

The last two experiments are for an antisymmetric discretization. We can see that we attain a stable behavior for the forward propagation for the operators $\mathbf{K}_+$ and $\mathbf{K}_-$. This demonstrates that an antisymmetric discretization represents a straight forward way to stabilize the forward propagation.

Verlet Method We present results for the forward propagation using the Verlet method. We consider networks consisting of $N = 20, 50, 100, 200, 500, 1000$ identical layers, i.e., on each layer we use the activation function $f = \tanh$, $h = 0.1$, $b = 0$, and $\mathbf{K}_v = \begin{pmatrix}\mathbf{k}_1 & \mathbf{k}_2\end{pmatrix}$ with $\mathbf{k}_1 = (2, -1, 0)^\mathsf{T}$ and $\mathbf{k}_2 = (1, 2, 1)^\mathsf{T}$.



3(a)*:* K_v *at* $N = 20$. ***3(b)****:* K_v *at* $N = 50$. ***3(c)****:* K_v *at* $N = 100$.

3(d)*:* K_v *at* $N = 200$. ***3(e)****:* K_v *at* $N = 500$. ***3(f)****:* K_v *at* $N = 1000$.

Figure 3*: Phase space diagrams for the forward propagation using the Verlet method. We show the behavior of the network for* $N = 20, 50, 100, 200, 500, 1000$ *identical layers.*

Observations: We can see that the forward propagation yields a stable transformation of the features for short term ($N = 20$) and non-trivial long term characteristics ($N = 1000$). We can see that although we use identical layers with constant kernels, the features do not explode or vanish asymptotically. The network remains stable.

Acknowledgements This work was supported by the 2018 Summer Undergraduate Research Fellowship from the Office of Undergraduate Research at the University of Houston.

References

[1] K. He, X. Zhang, S. Ren and J. Sun. Deep residual learning for image recognition. Proc IEEE Conference on Computer Vision and Pattern Recognition, 770–778 2016.

[2] E. Haber and L. Ruthotto. Stable architectures for deep neural networks. Inverse Problems 34(1):014004.

[3] E. Haber and L. Ruthotto. Deep neural networks motivated by partial differential equations. arXiv:1804.04272. 2018.

[4] H. Engl, M. Hanke and A. Neubauer. Regularization of inverse problems. Kluwer Academic Publishers, Dordecht, NL. 1996.

[5] A. Mang and G. Biros. An inexact Newton–Krylov algorithm for constrained diffeomorphic image registration. SIAM Journal on Imaging Sciences, 8:1030–1069, 2015.

[6] A. Mang and L. Ruthotto. A Lagrangian Gauss–Newton–Krylov solver for mass- and intensity-preserving diffeomorphic image registration. SIAM Journal on Scientific Computing, 39:B860–B885, 2017.

[7] A. Mang, S. Tharakan, A. Gholami, et al. SIBIA-GIS: Scalable biophysics-based image analysis for glioma segmentation. In Proc BraTS 2017 Workshop, pages 197–204, 2017.

[8] A. Mang, A. Gholami, C. Davatzikos and G.Biros. PDE-constrained optimization in medical image analysis. Optimization and Engineering, 19:765–812, 2018.