Deep Neural Networks Motivated By Ordinary Differential Equations

MS: Theoretical Foundations of Deep Learning
ICIAM @ Valencia, July 17, 2019

Lars Ruthotto
Departments of Mathematics and Computer Science, Emory University
lruhottot@emory.edu
@lruhottot
Team and Acknowledgments

Lili Meng  Bo Chang  Elliot Holtham  Eldad Haber  Samy Wu Fung
Eran Treister  Keegan Lensik  Jacob Schroder  Eric Cyr  Stefanie Günther

Funding: NSF DMS 1522599, DMS 1751636  US-Israel BSF 2018209
Other support: Donation of a TITAN X GPU
Agenda: Deep Neural Networks Motivated By ODEs

- Deep Learning meets Optimal Control
- Stability and Generalization
  - when is deep learning well-posed?
  - stabilizing the forward propagation
- Numerical Methods
  - simplectic, reversible neural networks
  - layer-parallel training using multigrid in time
- DNNs motivated by PDEs Fri, 3pm @ A1-1-3
  - parabolic CNNs, hyperbolic CNNs, IMEX-Net

**Goals:** Gain theoretical insight and(!) obtain mathematically sound network models that achieve competitive results.

- E Haber, LR
  *Stable Architectures for DNNs. Inverse Problems, 2017.*

- E Holtham et al.
  *Learning Across Scales. AAAI, 2018.*

- B Chang et al.,

- LR, E Haber
  *Deep Neural Networks motivated by PDEs. arXiv, 2018.*
Deep Learning meets Optimal Control
Deep Learning Revolution (?)

- deep learning: use neural networks (from $\approx$ 1950's) with many hidden layers
- able to "learn" complicated patterns from data
- applications: image classification, face recognition, segmentation, driverless cars, . . .
- recent success fueled by: massive data sets, computing power
- A few recent references:
  - A radical new neural network design could overcome big challenges in AI, MIT Tech Review ’18
  - Data Scientist: Sexiest Job of the 21st Century, Harvard Business Rev ’17
Supervised Learning using Deep Neural Networks

**Supervised Deep Learning Problem**

Given input features, $Y_0$, and labels, $C$, find **network weights** $(K, b)$ and **classification weights** $(W, \mu)$ such that the DNN predicts the data-label relationship (and generalizes to new data), by solving

$$\text{minimize}_{K, b, W, \mu} \quad \text{loss}[g(WY_N + \mu), C] + \text{regularizer}[K, b, W, \mu]$$

subject to

$$Y_{j+1} = \text{activation}(K_j Y_j + b_j), \quad \forall j = 0, \ldots, N - 1.$$
Deep Residual Neural Networks (simplified)

Award-winning forward propagation

\[ Y_{j+1} = Y_j + h K_{j,2} \sigma(K_{j,1} Y_j + b_j), \quad \forall j = 0, 1, \ldots, N - 1. \]

ResNet is forward Euler discretization of

\[ \partial_t y(t) = K_2(t) \sigma(K_1(t) y(t) + b(t)), \quad y(0) = y_0. \]

Notation: \( \theta(t) = (K_1(t), K_2(t), b(t)) \) and

\[ \partial_t y(t) = f(y, \theta(t)), \quad y(0) = y_0 \]

where \( f(y, \theta) = K_2(t) \sigma(K_1(t) y(t) + b(t)) \).

K. He, X. Zhang, S. Ren, and J. Sun

*Deep residual learning for image recognition.*

Optimal Control Framework for Deep Learning

Supervised Deep Learning Problem

Given training data, $Y_0$, and labels, $C$, find network parameters $\theta$ and classification weights $W, \mu$ such that the DNN predicts the data-label relationship (and generalizes to new data), i.e., solve

$$\minimize_{\theta, W, \mu} \text{loss}[g(W + \mu), C] + \text{regularizer}[	heta, W, \mu]$$
Review: Adjoint Method

Simplified learning problem: one example \((y_0, c)\), no weights for classifier, no regularizer

\[
\min_\theta \text{loss}(y(1, \theta), c) \quad \text{subject to} \quad \partial_t y(t, \theta) = f(y(t), \theta(t)), \quad y(0, \theta) = y_0.
\]

Use adjoint method to compute gradient of objective w.r.t. \(\theta\)

\[
\frac{\partial \text{loss}}{\partial \theta}(t) = \left( \frac{\partial f}{\partial \theta}(y(t), \theta(t)) \right)^\top z(t)
\]

where \(z\) satisfies the adjoint method \((-\partial_t \rightsquigarrow \text{backward in time})\)

\[
-\partial_t z(t, \theta) = \left( \frac{\partial f}{\partial y}(y(t, \theta), \theta(t)) \right)^\top z(t), \quad z(T, \theta) = \frac{\partial \text{loss}}{\partial y}(y(1, \theta), c).
\]

**note:** \(y(t)\) needed for solve adjoint equation  

---

G. A. Bliss  
*The use of adjoint systems in the problem of differential corrections for trajectories.*  
JUS Artillery, 51:296–311, 1919

D.E. Rumelhart, G.E. Hinton, R.J. Williams  
*Learning representations by back-propagating errors.*  
Deep Learning ↔ optimal control / parameter estimation.

- new ways to analyze and design neural networks
- expose similarities to trajectory problem, optimal transport, image registration, . . .
- training algorithms motivated by (robust) optimal control
- discrete ResNet ↩ continuous problem ↩ discrete architecture
(Some) Related Work

DNNs as (stochastic) Dynamical Systems


Optimal Control


Numerical Time Integrators


PDE-motivated Approaches

Stability and Well-Posedness
Stability of Deep Neural Networks: Motivation

**Goal in learning:** Build model that generalizes.

**Todo list:**
1. model forward dynamic
2. discretize forward dynamic (\(\rightarrow\) architecture)
3. train network by minimizing regularized loss

**Expectation:** tasks are related

Analogy: Recall the ingredients of a well-posed inverse problem
1. well-posed forward problem
2. bounded inverse

**Next:** study properties of forward propagation

---

**Modeling**
well-defined continuous forward propagation

**Discretization**
new architectures through consistent discretization

**Optimization**
new algorithms inspired by optimal control
Impact of Network Architecture on Optimization - 1

\[
\min_{\theta} \frac{1}{2} \| Y_N(\theta) - C \|_F^2 \quad Y_{j+1}(\theta) = Y_j(\theta) + \frac{10}{N} \tanh(KY_j(\theta))
\]

where \( C = Y_{200}(1, 1), \ Y_0 \sim \mathcal{N}(0, 1), \) and

\[
K(\theta) = \begin{pmatrix}
-\theta_1 - \theta_2 & \theta_1 & \theta_2 \\
\theta_2 & -\theta_1 - \theta_2 & \theta_1 \\
\theta_1 & \theta_2 & -\theta_1 - \theta_2
\end{pmatrix}
\]

loss, \( N = 5 \)

loss, \( N = 100 \)

Next: Compare examples for different inputs \( \sim \) generalization
Impact of Network Architecture on Optimization - 2

objective, $Y_0^{\text{train}}$

objective, $Y_0^{\text{test}}$

abs. diff

unstable, $N = 5$

stable, $N = 100$

|$\Sigma$|

| Title | Intro | Stab | New |

| 15 |
Stability of Continuous Forward Propagation

Interpret ResNet as discretization of initial value problem

\[
\frac{\partial}{\partial t} y(t, \theta, y_0) = f(y(t, \theta, y_0), \theta(t))
\]

\[y(0, \theta, y_0) = y_0.\]

IVP is stable if for any \(v \in \mathbb{R}^n\)

\[
\|y(T, \theta, y_0) - y(T, \theta, y_0 + \epsilon v)\|^2 = O(\epsilon \|v\|).
\]

idea: ensure stability by design / constraints on \(f\) and \(\theta\)
Fact: The ODE \( \partial_t y(t) = f(y) \) is stable if the real parts of the eigenvalues of the Jacobian \( J \) are non-positive.

Example: Consider ResNet with stationary weights

\[
\partial_t y(t) = \sigma \left( K y(t) + b \right) \quad \Rightarrow \quad J(t) = \text{diag}(\sigma'(Ky(t) + b))K.
\]

In general, one cannot assume that forward propagation is stable.

Networks with non-stationary weights require additional arguments (e.g., kinematic eigenvalues) or assumptions (\( J \) changes slowly).
Enforcing Stability: Antisymmetric Transformation

Two examples of more stable networks.

ResNet with antisymmetric transformation matrix

\[ \partial_t y(t) = \sigma((K(t) - K(t)^\top)y + b(t)). \]

Hamiltonian-like ResNet

\[ \frac{d}{dt} \begin{pmatrix} y \\ z \end{pmatrix}(t) = \sigma \begin{pmatrix} 0 & K(t) \\ -K(t)^\top & 0 \end{pmatrix} \begin{pmatrix} y \\ z \end{pmatrix} + b(t). \]

How about the stability of the discrete system?
Stability of Discrete Forward Problem

ResNet not stable for layer \( f_{\text{antisym}}(Y, \theta_j) = \sigma \left( (K_j - K_j^\top)Y + b_j \right) \).

Need to replace fwd Euler by, e.g.,

\[
Y_{j+1} = Y_j + \frac{h}{12} \left( 23f_{\text{antisym}}(Y_j, \theta_j) - 16f_{\text{antisym}}(Y_{j-1}, \theta_{j-1}) + 5f_{\text{antisym}}(Y_{j-2}, \theta_{j-2}) \right).
\]
Verlet Integration for Hamiltonian-inspired NNs

Forward propagation: For $Z_{-\frac{1}{2}} = 0$ and $j = 0, \ldots, N - 1$ do

$$Z_{j+\frac{1}{2}} = Z_{j-\frac{1}{2}} - h\sigma (K_j Y_j + b_j)$$

$$Y_{j+1} = Y_j + h\sigma \left( K_j^T Z_{j+\frac{1}{2}} + b_j \right)$$

Note that this is reversible. Given $Y_N$ and $Z_{N-\frac{1}{2}}$ and $j = N - 1, \ldots, 1$ do

$$Y_j = Y_{j+1} - h\sigma \left( K_j^T Z_{j+\frac{1}{2}} + b_j \right)$$

$$Z_{j-\frac{1}{2}} = Z_{j+\frac{1}{2}} + h\sigma (K_j Y_j + b_j)$$

Notes:

- reversibility well-known from hyperbolic PDE-constrained optimization
- this network is a special (in particular, stable) case of ’RevNet’

A. Gomez, M. Ren, R. Urtasun, R. Grosse
*The Reversible Residual Network: Backpropagation Without Storing Activations*

B. Chang, L. Meng, E. Haber, LR, D. Begert, E. Holtham
*Reversible architectures for arbitrarily deep residual neural networks*
32nd AAAI, 1–8, 2018.
Limitations of Reversibility

Q: Is any algebraically reversible network reversible in practice? For $\alpha, \beta \in \mathbb{R}$ consider original RevNet with $F(Y) = \alpha Y$ and $G(Z) = \beta Z$, i.e.,

$$Z_{j+\frac{1}{2}} = Z_{j-\frac{1}{2}} - \alpha Y_j, \quad \text{and} \quad Y_{j+1} = Y_j + \beta Z_{j+\frac{1}{2}}.$$ 

Combining two time steps in $Y$

$$Y_{j+1} - Y_j = \beta Z_{j+\frac{1}{2}}, \quad \text{and} \quad Y_j - Y_{j-1} = \beta Z_{j-\frac{1}{2}}$$

Subtracting those two gives

$$Y_{j+1} - 2Y_j + Y_{j-1} = \beta (Z_{j+\frac{1}{2}} - Z_{j-\frac{1}{2}}) = \alpha \beta Y_j$$

$$\Leftrightarrow Y_{j+1} - (2 + \alpha \beta)Y_j + Y_{j-1} = 0$$

There is a solution $Y_j = \xi^j$, i.e., with $a = (2 + \alpha \beta)/2$

$$\xi^2 - 2a\xi + 1 = 0 \quad \Rightarrow \quad \xi = a \pm \sqrt{a^2 - 1}$$

$$|\xi|^2 = 1 \text{ (stable) if } a^2 \leq 1. \text{ Otherwise } \xi \text{ growing!}$$
Example: Impact of ODE Solver on Training Performance

classification problem generated from peaks in MATLAB®

data setup

- 2,000 points in 2D, 5 classes
- Residual Neural Network
- \( \text{tanh} \) activation, softmax classifier
- multilevel: 32 layers \( \rightarrow \) 64 layers

compare three configurations
1. "unstable": \( T = 10 \) (3rd order multistep)
2. "medium": \( T = 5 \) (1st order Verlet)
3. "stable": \( T = 0.2 \) (3rd order multistep)

Q: how does learning performance compare?
Example: Impact of ODE Solver - Convergence

- **unstable, ab3 ($T = 10$)**
  - Objective function
  - Validation accuracy

- **medium, Verlet ($T = 5$)**
  - Objective function
  - Validation accuracy

- **stable, ab3 ($T = 0.2$)**
  - Objective function
  - Validation accuracy

Graphs show the behavior of DNNs with different ODE solvers and layer counts over time.
Example: Impact of ODE Solver - Dynamics

unstable, $\text{ab}3$ ($T = 10$)  

medium, Verlet ($T = 5$)  

stable, $\text{ab}3$ ($T = 0.2$)
Example (Ellipses): Hamitonian-like network with Verlet

- 2D feature space, concentric ellipses
- 1k training, 2k validation
- multilevel: 2 → 1024 layers
- optimization: block coordinate descent with Newton-PCG
- weight decay (Tikhonov) regularization
- \( \tanh \) activation, width 2
Example (Swiss Roll): Hamitonian-like network with Verlet

- 2D feature space, swiss roll
- 256 training, 256 validation
- multilevel: 2 → 1024 layers
- optimization: block coordinate descent with Newton-PCG
- weight decay (Tikhonov) regularization
- tanh activation, width 4
Numerical Methods
Layer-Parallel Training of Deep Residual Neural Networks

with S. Günther, J. B. Schroder, E. C. Cyr, N. R. Gauger

strong scaling (fwd + gradient)  simultaneous optimization

- forward and backward propagation are optimal but sequential
- replace with suboptimal but parallel iterative solver
- use multigrid in time (MGRIT) and one-shot optimization

S. Günther, LR, J. B. Schroder, E. C. Cyr, N. R. Gauger

*Layer-Parallel Training of Deep Residual Neural Networks.*
in revision, SIMODS, 2019.
course launched Spring 18 at Emory and UBC
slides + simple MATLAB codes available (pyTorch to come)
next offerings: Fall ’19 at UBC and Spring ’20 at Emory

check it out: https://github.com/IPAIopen
Meganet - A Scientific Computing Approach to DL

- **Meganet.m**
  - A fresh approach to deep learning written in MATLAB
  - [Meganet.m](http://www.github.com/XtractOpen/)

- **Meganet.jl**
  - A fresh approach to deep learning written in Julia
  - [Meganet.jl](http://www.github.com/XtractOpen/)

- **Purpose**: enable scientific computing research in deep learning
- **Status**: under heavy development.
- **Goal**: better models and algorithms \( \rightarrow \) scientific machine learning
- **Launched**: January 2018 / MATLAB and Julia / open access license

http://www.github.com/XtractOpen/
European Journal of Applied Mathematics
Editors-in-Chief: Martin Burger, Michael Ward, John King

More info at
https://www.cambridge.org/core/journals/european-journal-of-applied-mathematics

To submit your paper: https://www.editorialmanager.com/ejam

Read the EJAM blog at
https://www.cambridge.org/core/blog/

Call for Papers: Connections between Deep learning and Partial Differential Equations

Conclusion
Σ: Deep Neural Networks motivated by ODEs

Optimal control formulation
- new insights, theory, algorithms

Stability and well-posedness
- continuous (constraints/design) vs. discrete
- examples: impact on optimization/generalization

Numerical Methods: Discretize-Optimize
- Verlet: reversible and stable networks (memory-free)
- Parallel-in-Layer: additional option for parallelism

DNNs motivated by PDEs Fri, 3pm @ A1-1-3
- parabolic CNNs, hyperbolic CNNs, IMEX-Net

Lots to do/explore/contribute for computational and applied mathematicians…

E Haber, LR
Stable Architectures for DNNs.

E Holtham et al.
Learning Across Scales.
AAAI, 2018.

B Chang et al.,
Reversible Architectures for Deep ResNNs.
AAAI, 2018.

LR, E Haber
Deep Neural Networks motivated by PDEs.