An Optimal Control Framework for Efficient Training of Deep Neural Networks

CoSIP Intense Course on Deep Learning, Berlin
December 1, 2017

Lars Ruthotto
Department of Mathematics and Computer Science, Emory University
lruthotto@emory.edu
Fundamental (Open) Questions

**Expressibility**
- how to find neural network that can approximate function of interest?
- successes: approximation theorems, optimal sparsity, . . .
- communities: harmonic analysis, approximation theory, . . .

**Learning**
- how to (efficiently) train neural network?
- successes: stochastic gradient and zoo of variants (including ADAM, second-order, . . .)
- community: mainly optimization and optimal control

**Generalization**
- does the neural network generalize?
- successes: VC dimensions, bias/variance dilemma, regularization, . . .
- community: mainly statistics

**Computing**
- how to design network that is expressive and generalizes well and which method will train it efficiently?
- successes: hardware, . . .
- community: scientific computing
Team and Acknowledgements

Joint work: Emory ↔ Xtract Tech. ↔ University of British Columbia

Lili Meng  Bo Chang  Elliot Holtham  Eldad Haber  Seong Hwan Jun

Funding:

- ☀️ This work is supported in part by NSF award DMS 1522599
- Thanks to NVIDIA Corp for donation of a TITAN X GPU
Agenda: Optimal Control Framework for Deep Learning

- Deep Learning meets Optimal Control
- Stability and Generalization
  - when is deep learning well-posed?
  - stabilizing the forward propagation
- Convolution Neural Networks as PDE
  - continuity in feature space
  - allows to interpret and categorize CNN
- Multiscale Parabolic CNNs
  - image classification across scales
  - shallow-to-deep training
- Reversible Hyperbolic CNNs
  - memory-efficient + stable → arbitrarily deep

E Haber, LR
*Stable Architectures for Deep Neural Networks.*

E Holtham, E Haber, LR
*Learning Across Scales.*
AAAI, 2018.

B Chang, L Meng, E Holtham, E Haber, LR, D Begert
*Reversible Architectures for Arbitrarily Deep ResNNs.*
AAAI, 2018.
Deep Learning meets Optimal Control
Deep Learning Revolution (\(?)\)

- Neural Networks with a particular (deep) architecture
- invented in the 1950’s
- 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 quotes:
  - Apple Is Bringing the AI Revolution to Your iPhone, WIRED ’16
  - Data Scientist: Sexiest Job of the 21st Century, Harvard Business Rev ’17

\[ Y_{j+1} = \sigma(K_j Y_j + b_j) \]
Supervised Learning using Deep Neural Networks

Supervised Deep Learning Problem

Given training data, \( Y_0 \), and labels, \( C \), find transformation parameters \((K, b)\) and classification weights \((W, \mu)\) such that the DNN predicts the data-label relationship (and generalizes to new data), by solving

\[
\begin{align*}
\text{minimize}_{K,b,W,\mu} & \quad \text{loss}[g(WY_N + \mu), C] + \text{regularizer}[K, b, W, \mu] \\
\text{subject to} & \quad Y_{j+1} = \text{activation}(K_j Y_j + b_j), \quad \forall j = 0, \ldots, N - 1
\end{align*}
\]
Deep Residual Neural Networks

Award-winning forward propagation

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

ResNet is forward Euler discretization of

\[
\begin{align*}
\partial_t y(t, K, b, y_0) &= K_2(t) \sigma (K_1(t)y(t, K, b, y_0) + b(t)), \\
y(0, K, b, y_0) &= y_0.
\end{align*}
\]

deep learning ↔ trajectory problem, image registration, mass transport, . . .

In short, write ResNets as

\[
\begin{align*}
\partial_t y(t, \theta(t), y_0) &= f(y, \theta(t)), \\
y(0, \theta, y_0) &= y_0
\end{align*}
\]

K. He, X. Zhang, S. Ren, and J. Sun
Deep residual learning for image recognition.
Optimal Control Framework for Deep Learning

Given training data, $\mathbf{Y}_0$, and labels, $\mathbf{C}$, find network parameters $\theta$ and classification weights $\mathbf{W}$, $\mu$ such that the DNN predicts the data-label relationship (and generalizes to new data), i.e., solve

$$\min_{\theta, W, \mu} \text{loss} \left[ g(\mathbf{WY}(T, \theta, \mathbf{Y}_0) + \mu), \mathbf{C} \right] + \text{regularizer}[\theta, \mathbf{W}, \mu]$$
Optimal Control Approaches to Deep Learning

Deep Learning ↔ trajectory problem.
- use for analysis and new algorithms
- invent your own architecture

E. Haber, LR
*Stable Architectures for Deep Neural Networks.*
Inverse Problems, accepted 2017.

Weinan E
*A Proposal on Machine Learning via Dynamical Systems.*
Blessing of Dimensionality (or Width)

Setup: ResNN, 9 fully connected single layers, $\sigma = \tanh$

---

1. $\approx 100\%$ validation accuracy
Stability and Well-Posedness
Stability of DNN: Goals

Our goals:
- predict a priori if architecture can generalize
- design architectures that generalize
- impose regularization to find solutions that generalize

main ingredients of well-posed inverse problems:
1. well-posed forward problem
2. bounded inverse
Stability of Continuous Forward Propagation

Interpret ResNet as discretization of initial value problem

\[ \partial_t y(t, K, b, y) = \sigma(K(t)y(t, K, b, y) + b(t)) \]
\[ y(0, K, b, y) = y. \]

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

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

idea: ensure stability by design / constraints on \( K, b \)
Fact: The ODE $y'(t) = f(y)$ is stable if the real parts of the eigenvalues of the Jacobian $J$ are non-positive.

For non-autonomous ODEs we also need that $J$ changes slowly in time. Rigorous argument using framework of kinematic eigenvalues.

For the ResNet

$$J(t) = \text{diag}(\sigma'(K(t)Y(t) + b(t)))K(t).$$

If activation is monotonically increasing, $\text{real}(\text{eig}(K(t))) \leq 0$ sufficient.
Enforcing Stability: Antisymmetric Transformation

Two examples of unconditionally stable networks.

ResNet with antisymmetric transformation matrix

\[
\frac{d}{dt} y(t) = \sigma((K(t) - K(t)^T)y + b(t)).
\]

ResNet with auxiliary variable and antisymmetric matrix

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

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

Note: ResNet (fwd Euler) not stable for antisymmetric transformations. Better:

$$Y_{j+1} = Y_j + \frac{h}{12} \left( 23\sigma(K_j Y_j + b_j) - 16\sigma(K_{j-1} Y_{j-1} + b_{j-1}) + 5\sigma(K_{j-2} Y_{j-2} + b_{j-2}) \right).$$
Symplectic Integration

Hamiltonian-inspired neural networks (Verlet integration)

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

\[
Y_{j+1} = Y_j + h\sigma (K_j^\top Z_{j+\frac{1}{2}} + b_j)
\]

Necessary Conditions for Generalization

- continuous forward propagation stable when \(\text{real}(\text{eig}(K)) \leq 0\)
- learning problem well-posed when \(\text{real}(\text{eig}(K)) \approx 0\)
- stable scheme for discrete forward propagation and \(h\) ”small enough”

Simple explanation of (and cure for) exploding and vanishing gradients!

Y. Bengio, P. Simard, P. Frasconi

Learning Long-Term Dependencies with Gradient Descent Is Difficult
Example: Impact of Network Depth

classification problem generated from peaks in MATLAB®

data setup

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

compare three configurations

1. "deep": \( T = 5 \) (3rd order multistep)
2. "medium": \( T = 2 \) (1st order Verlet)
3. "shallow": \( T = 0.2 \) (3rd order multistep)

Q: how does learning performance compare?
Example: Impact of Network Depth - Convergence

- **deep, \( ab^3 \) (\( T = 5 \))**
  - Objective function: 
    - 32 layers
    - 64 layers
  - Validation accuracy:
    - 32 layers
    - 64 layers

- **medium, Verlet (\( T = 2 \))**
  - Objective function:
    - 32 layers
    - 64 layers
  - Validation accuracy:
    - 32 layers
    - 64 layers

- **shallow, \( ab^3 \) (\( T = 0.2 \))**
  - Objective function:
    - 32 layers
    - 64 layers
  - Validation accuracy:
    - 32 layers
    - 64 layers
Example: Impact of Network Depth - Dynamics

deep, ab3 \((T = 5)\)  
medium, Verlet \((T = 2)\)  
shallow, ab3 \((T = 0.2)\)

t=0.16  
t=0.08  
t=0.00
PDE-Interpretation of Convolution Neural Networks
Convolutions and PDEs

Let \( y \) be of 1D grid function \( y \leftrightarrow y \) (grid: \( n \) cells of width \( h_x = 1/n \))

\[
K(\theta)y = [\theta_1 \theta_2 \theta_3] \ast y = \left( \frac{\beta_4}{4} [1 \ 2 \ 1] + \frac{\beta_2}{2h_x} [-1 \ 0 \ 1] + \frac{\beta_3}{h_x^2} [-1 \ 2 \ -1] \right) \ast y
\]

where the coefficients \( \beta_1, \beta_2, \beta_3 \) satisfy

\[
\begin{pmatrix}
1/4 & -1/(2h_x) & -1/h_x^2 \\
1/2 & 0 & 2/h_x^2 \\
1/4 & 1/(2h_x) & -1/h_x^2
\end{pmatrix}
\begin{pmatrix}
\beta_1 \\
\beta_2 \\
\beta_3
\end{pmatrix}
=
\begin{pmatrix}
\theta_1 \\
\theta_2 \\
\theta_3
\end{pmatrix}.
\]

In the limit \( h_x \to 0 \) this gives

\[
K(\theta(t)) = \beta_1(t) + \beta_2(t) \partial_x + \beta_3(t) \partial_x^2.
\]

Convolution operator \( K \) is linear combination of differential operators.
Parabolic CNN

In original Residual Net choose $\mathbf{K}_2 = -\mathbf{K}_1^T = \mathbf{K}^T$. This gives parabolic PDE

$$\mathbf{Y}_t = -\mathbf{K}(t)^\top \sigma(\mathbf{K}(t)\mathbf{Y} + \mathbf{b}(t)), \quad \mathbf{Y}(0) = \mathbf{Y}_0$$

Jacobian

$$\mathbf{J}(t) = -\mathbf{K}(t)^\top \text{diag} \left( \sigma'(\mathbf{K}(t)\mathbf{Y} + \mathbf{b}(t)) \right) \mathbf{K}(t)$$

is symmetric and negative definite ($\sigma' \geq 0$) $\Rightarrow$ stable if $\mathbf{K}, \mathbf{b}$ do not change too quickly. Use forward Euler discretization with $h$ small enough

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j - h\mathbf{K}_j^\top \sigma(\mathbf{K}_j\mathbf{Y} + \mathbf{b}_j), \quad j = 0, 1, \ldots, N - 1$$

Similar to anisotropic diffusion (popular in image processing)

Y. Chen, T. Pock

*Trainable Nonlinear Reaction Diffusion.*
Hamiltonian CNN

Introducing auxiliary variable $Z$, consider dynamics

\[
\begin{align*}
\frac{\partial}{\partial t} Y(t) &= K_1^T(t) \sigma(K_1(t)Z(t) + b_1(t)), \\
\frac{\partial}{\partial t} Z(t) &= -K_2^T(t) \sigma(K_2(t)Y(t) + b_2(t)).
\end{align*}
\]

In matrix form this is

\[
\begin{pmatrix}
\frac{\partial}{\partial t} Y \\
\frac{\partial}{\partial t} Z
\end{pmatrix} = 
\begin{pmatrix}
K_1^T & 0 \\
0 & -K_2^T
\end{pmatrix} \sigma
\begin{pmatrix}
0 & K_1 \\
K_2 & 0
\end{pmatrix}
\begin{pmatrix}
Y \\
Z
\end{pmatrix} +
\begin{pmatrix}
b_1 \\
b_2
\end{pmatrix}.
\]

(Can be shown that eigenvalues of Jacobian are all imaginary $\Rightarrow$ stability when $K_1, K_2, b_1, b_2$ change slowly in time) Discretize using Verlet method

\[
\begin{align*}
Y_{j+1} &= Y_j + hK_{j1}^T \sigma(K_{j1}Z_j + b_{j1}), \\
Z_{j+1} &= Z_j - hK_{j2}^T \sigma(K_{j2}Y_{j+1} + b_{j2}).
\end{align*}
\]
Second-Order Network

Consider second-order forward dynamics

\[ \partial_{tt} Y = -K(t)^{\top} \sigma(K(t)Y + b(t)) \]

And their Leapfrog discretization

\[ Y_{j+1} = 2Y_j - Y_{j-1} - h^2 K_j^{\top} \sigma(K_jY + b_j) \]

Similar to: Full Waveform Inversion, Ultrasound, ...
Loss Function in CNN

Let $C$ be some values, $Y$ features, and $W$ classification weights.

Regression:

$$\text{loss}(W, Y) = \frac{1}{2} \| WY - C \|^2$$

If $Y$ and $W$ are discretizations of $Y$ and $W$, loss is discretization of

$$\ell(W, Y) = \left( \int_{\Omega} W(x)Y(x)dx - C \right)^2$$

$\Rightarrow$ add $h_x^2$ and model $W$ as image, e.g., enforce smoothness

$$R(W) = \int_{\Omega} \| \nabla W(x) \|^2 dx.$$ 

Classification: Similar, since hypothesis functions can be written as

$$C_{\text{pred}}(W, Y) = g \left( h_x^2 WY \right)$$
Some Challenges in CNN

Computations
- note that networks have a widths and depth
- Toy example: width 16, depth 20 (time steps).
  \textit{Forward propagation: 5,120 2D-convs/image}

Memory Consumption
- adjoint equations (backpropagation) need intermediate states (hidden features)
- Toy example (continued): training data is 5k images with $32 \times 32$ pixels.
  \textit{Storage (just features): 130 GB (double) or 65 GB (single).}

Architecture Design & Interpretation
- CNN should be easy to train and generalize well
- CNN should be difficult to fool (adversarial)
- Can we understand the reasoning of a CNN?

Pei et al., \textit{DeepXplore}, 2017
Parabolic Networks
Parabolic Residual Neural Networks

Recall the decay property of heat equation. Example:

$$\partial_t y(t, x) = -\partial_{xx} y(t, x), \quad + \text{initial + boundary cond.}$$

Some consequences for learning

- forward propagation is asymptotically stable (if kernels constant in time)
- network is robust against perturbation of inputs (adversarial)
- learning problem ill-posed ($\leadsto$ inverse heat equation)
- numerical methods for parabolic include multiscale, multilevel, ROM, ...
Multi-Resolution Learning

Restrict the images $n$ times

$\theta^0, W^0,\mu^0 \leftarrow$ random initialization

for $j = 1 : n$ do

optimize with data on level $k$ starting from $\theta^{j-1}, W^{j-1}, \mu^{k-1}$

obtain $\theta^*, W^*,\mu^*$

$\theta^j \leftarrow$ prolongate $\theta^*$

$W^j \leftarrow$ interpolate $W^*$

How to prolongate the kernels?
Galerkin Projection of Convolution Kernels

\[ K_H = RK_hP, \]

where

- \( K_h \) fine mesh operator (given)
- \( R \) restriction (e.g., averaging)
- \( P \) prolongation (e.g., interpolation)

Remarks:

- Galerkin: \( R = \gamma P^\top \)
- Coarse \( \rightarrow \) Fine: unique if kernel size constant.
- only small linear solve required
Example: Multiresolution Learning

data
- 60,000 gray-scale images $18 \times 18$
- Residual Neural Network, width 6
- 2D convolution layers, fully connected
- $\tanh$ activation, softmax classifier

multilevel experiments
1. train on fine $\rightarrow$ classify coarse:
   - 84.1% vs. 94.9%
   - (no restriction) (with restriction)
2. train on coarse $\rightarrow$ classify fine:
   - 61.0% vs. 91.0%
   - (no prolongation) (with prolongation)
Example: Shallow-to-Deep MNIST

6 level strategy / random initialization vs. multilevel learning.

![Graph showing validation accuracy and objective function for different layer counts with random initialization and multilevel prolongation.]
Example: Multilevel Learning ImageNet-10

ImageNet-10

- 13k natural images
  - 224 × 224
- 10 sub-categories
- Residual Neural Network, width 64, depth 34
- 2D convolution layers, fully connected

![Classification Error vs Epochs](image)

<table>
<thead>
<tr>
<th></th>
<th>Fine-scale only</th>
<th>Coarse-to-Fine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Runtime [sec]</td>
<td>59,122±7,540</td>
<td>43,882±3,476</td>
</tr>
<tr>
<td>Validation Acc.</td>
<td>76.47±0.93%</td>
<td>82.67±0.93%</td>
</tr>
</tbody>
</table>
Hyperbolic Networks
Recall the reversibility of hyperbolic equations. Example:

$$\partial_{tt} y(t, x) = \partial_{xx} y(t, x), \quad + \text{initial + boundary cond.}$$

Similar property recently discovered for residual networks

$$y_{k+1} = y_k + F(x_k) \quad \rightarrow \quad x_{k-1} = x_k - G(y_k)$$

$$x_{k+1} = x_k + G(y_k) \quad \rightarrow \quad y_{k-1} = y_k - F(x_k)$$

useful for adjoint computations (backpropagation)

**Reversible ResNets:** ↓↓↓ memory ↑ computation

---

B.D. Nguyen, G.A. McMechan
*Five ways to avoid storing source wavefield snapshots in 2D elastic prestack reverse time migration.*

A.N. Gomez, M. Ren, R. Urtasun, R.B. Grosse
*The Reversible Residual Network: Backpropagation Without Storing Activations.*
Limitations of Reversibility

Q: Is any algebraically reversible network reversible in practice?

For $\alpha, \beta \in \mathbb{R}$ consider original RevNet for $F(Z) = \alpha Z$ and $G(Y) = \beta Z$

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

Combining two time steps in $Y$

$$Y_{j+1} - Y_j = \alpha Z_j, \quad \text{and} \quad Y_j - Y_{j-1} = \alpha Z_{j-1}$$

Subtracting those two gives

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

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

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

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

$$|\xi|^2 = 1 \quad \text{(stable) if} \quad (1 + \alpha \beta/2)^2 \leq 1. \quad \text{Otherwise} \quad \xi \quad \text{growing!}$$
Reversible Hamiltonian Neural Networks

Forward propagation in double-layer Hamiltonian

\[ Y_{j+1} = Y_j + hK_{j1}^T \sigma(K_{j1}Z_j + b_{j1}), \]
\[ Z_{j+1} = Z_j - hK_{j2}^T \sigma(K_{j2}Y_{j+1} + b_{j2}). \]

Recall: Antisymmetric structure gives stability (when parameters change slowly).

Clearly, given \( Y_N, Y_{N-1} \) and \( Z_N, Z_{N-1} \) dynamics can be computed backwards:

\[ Z_j = Z_{j+1} + hK_{j2}^T \sigma(K_{j2}Y_{j+1} + b_{j2}) \]
\[ Y_j = Y_{j+1} - hK_{j1}^T \sigma(K_{j1}Z_j + b_{j1}), \]

Possible to recompute weights, ↑50% computation costs, but large memory savings + stability

A. Mahendran, A Vedaldi
*Understanding deep image representations by inverting them.*
CVPR, 2015.

B. Chang, L Meng, E. Holtham, E. Haber, LR, D Begert
*Reversible Architectures for Arbitrarily Deep ResNNs.*
Hamiltonian Reversible STL-10

Data:
- color images, $96 \times 96$
- 10 classes
- 5k training images
- 8k test images

Test Accuracy for our networks:
- Hamiltonian: 85.5%
- Midpoint: 84.6%
- Second Order: 83.7%

Some Benchmark Results:

<table>
<thead>
<tr>
<th></th>
<th>Google</th>
<th>Intel</th>
<th>Chinese UHK</th>
<th>Nvidia</th>
<th>Facebook</th>
<th>Xtract</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>2013</td>
<td>2014</td>
<td>2015</td>
<td>2015</td>
<td>2016</td>
<td>2017</td>
</tr>
<tr>
<td>Accuracy</td>
<td>70.2%</td>
<td>72.8%</td>
<td>73.15%</td>
<td>74.10%</td>
<td>74.33%</td>
<td>85.5%</td>
</tr>
</tbody>
</table>
Experiment: Stability and Generalization

Goal: Compare efficiency of reversible Hamiltonian CNN to ResNN

CIFAR10 (total: 50k images)  
STL10 (total: 5k images)

stability leads to improved generalization
Conclusion
Numerical Methods for Deep Learning

An (almost perfectly) true statement

\[
\text{backpropagation} + \text{GPU} + \left\{
\begin{array}{l}
\text{TensorFlow} \\
\text{Caffe} \\
\text{Torch} \\
\vdots
\end{array}
\right\} \implies \text{success}
\]

So, why study numeric methods for deep learning?

Transfer Learning
- DL is similar to path planning, optimal control, differential equations . . .

Do More With Less
- Better modeling and algorithms \(\rightsquigarrow\) process more data, use less resources
- How about 3D images and videos?

Power Of Abstraction
- Use continuous interpretation to design/relate architectures

It’s fun!
- new interdisciplinary courses in computer science + mathematics
ι: Optimal Control Framework for Deep Learning

Optimal control formulation

- new insights, theory, algorithms

Stability and well-posedness

- required for generalization
- continuous propagation \(\leadsto\) Hamiltonian systems
- discrete propagation (Verlet, Adams-Bashforth, \ldots)
- example: impact on convergence (no batch normalization!)

Parabolic CNNs

- multiscale and shallow-to-deep training

Hyperbolic CNNs

- preserve high-frequency features
- can be made reversible

E. Haber, LR
Stable Architectures for Deep Neural Networks.

E. Holtham, E. Haber, LR
Learning Across Scales.
AAAI, 2018.

B. Chang, L Meng, E. Holtham, E. Haber, LR, D Begert
Reversible Architectures for Arbitrarily Deep ResNNs.
AAAI, 2018.