Deep Neural Networks Motivated By Ordinary Differential Equations

Machine Learning for Physics and the Physics of Learning

Los Angeles, September, 2019

Lars Ruthotto

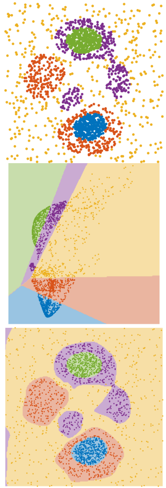
Departments of Mathematics and Computer Science, Emory University

lruthotto@emory.edu

@lruthotto

Agenda: Deep Neural Networks Motivated By ODEs

- **Deep Learning meets Optimal Control**
  - discretize $\rightarrow$ differentiate (or vice versa?)
- **Stability and Generalization**
  - when is deep learning well-posed?
  - stabilizing the forward propagation
- **Numerical Methods**
  - symplectic, reversible neural networks
  - layer-parallel training using multigrid in time
- **DNNs motivated by PDEs (tomorrow)**
  - parabolic/hyberbolic CNNs, IMEX-Net, Lean ResNets,...



Goals: Theoretical insight, mathematically sound architectures, competitive results.

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

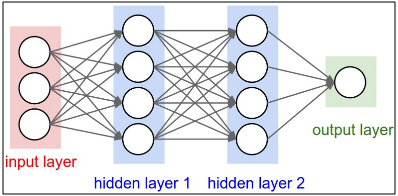
E Holtham et al. *Learning Across Scales.* AAAI, 2018.

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

LR, E Haber *Deep Neural Networks motivated by PDEs.* arXiv, 2018.

Deep Learning meets Optimal Control

Deep Learning Revolution (?)

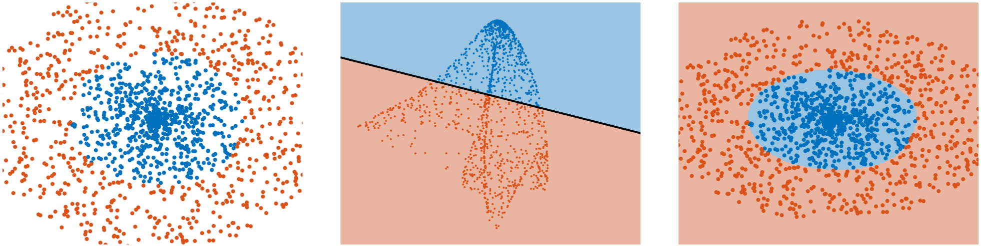


$$\begin{cases} \mathbf{Y}_{j+1} = \sigma(\mathbf{K}_j\mathbf{Y}_j + \mathbf{b}_j) \\ \mathbf{Y}_{j+1} = \mathbf{Y}_j + \sigma(\mathbf{K}_j\mathbf{Y}_j + \mathbf{b}_j) \\ \mathbf{Y}_{j+1} = \mathbf{Y}_j + \sigma\left(\mathbf{K}_{j,2}\sigma(\mathbf{K}_{j,1}\mathbf{Y}_j + \mathbf{b}_{j,1}) + \mathbf{b}_{j,2}\right) \\ \quad\vdots \end{cases}$$

(Notation: $\mathbf{Y}_j$: features, $\mathbf{K}_j, \mathbf{b}_j$: weights, σ : activation)

- 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



training data, $\mathbf{Y}_0, \mathbf{C}$ propagated features, $\mathbf{Y}_N, \mathbf{C}$ classification result

Supervised Deep Learning Problem

Given input features, $\mathbf{Y}_0$, and labels, $\mathbf{C}$, find **network weights ($\mathbf{K}$, $\mathbf{b}$)** and **classification weights ($\mathbf{W}$, μ)** such that the DNN predicts the data-label relationship (and generalizes to new data), by solving

$$\begin{aligned} \text{minimize}_{\mathbf{K},\mathbf{b},\mathbf{W},\boldsymbol{\mu}} \quad & \text{loss}[g(\mathbf{W}\mathbf{Y}_N + \boldsymbol{\mu}), \mathbf{C}] + \text{regularizer}[\mathbf{K}, \mathbf{b}, \mathbf{W}, \boldsymbol{\mu}] \\ \text{subject to} \quad & \mathbf{Y}_{j+1} = \text{activation}(\mathbf{K}_j\mathbf{Y}_j + \mathbf{b}_j), \quad \forall j = 0, \ldots, N-1. \end{aligned}$$

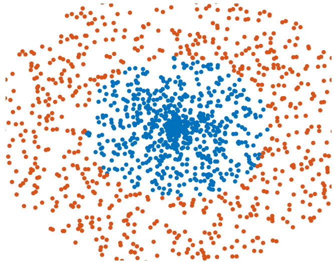
Deep Residual Neural Networks (simplified)

Award-winning forward propagation

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + h\mathbf{K}_{j,2}\sigma(\mathbf{K}_{j,1}\mathbf{Y}_j + \mathbf{b}_j), \quad \forall\, j = 0, 1, \ldots, N-1.$$

ResNet is forward Euler discretization of

$$\partial_t \mathbf{y}(t) = \mathbf{K}_2(t)\sigma\left(\mathbf{K}_1(t)\mathbf{y}(t) + \mathbf{b}(t)\right), \qquad \mathbf{y}(0) = \mathbf{y}_0.$$

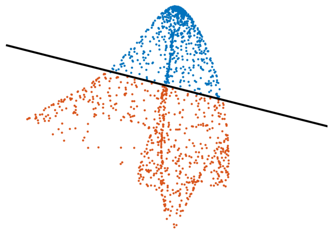


input features, $\mathbf{Y}_0$

Notation: $\boldsymbol{\theta}(t) = (\mathbf{K}_1(t), \mathbf{K}_2(t), \mathbf{b}(t))$ and

$$\partial_t \mathbf{y}(t) = f(\mathbf{y}, \boldsymbol{\theta}(t)), \quad \mathbf{y}(0) = \mathbf{y}_0$$

where $\quad f(\mathbf{y}, \boldsymbol{\theta}) = \mathbf{K}_2(t)\sigma\left(\mathbf{K}_1(t)\mathbf{y}(t) + \mathbf{b}(t)\right).$



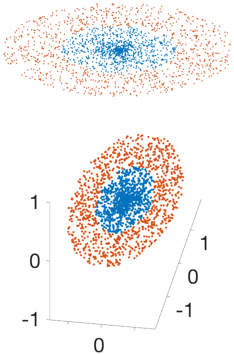
propagated features, $\mathbf{Y}_N$

K. He, X. Zhang, S. Ren, and J. Sun
Deep residual learning for image recognition.
IEEE Conf. on CVPR, 770–778, 2016.

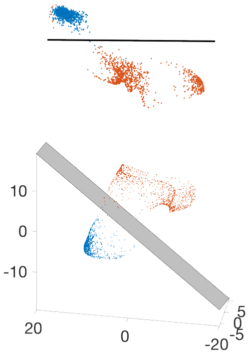
Blessing of Dimensionality (or Width)

Setup: ResNN, 9 fully connected single layers, $\sigma = \tanh$.
Motivated by: E Celledoni, M Erhardt, M Benning(Cambridge)

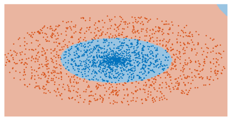
input features + labels



propagated features

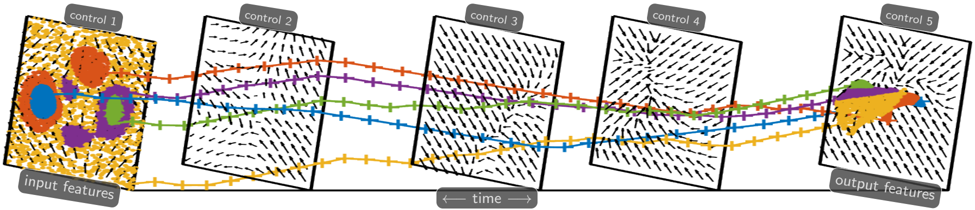


classification result[1]



[1] $\approx$ 100% validation accuracy

Optimal Control Approaches to Deep Learning



Deep Learning meets 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 $\rightsquigarrow$ continuous problem $\rightsquigarrow$ discrete architecture

(Some) Related Work

DNNs as (stochastic) Dynamical Systems

- **Weinan E**, *Proposal on ML via Dynamical Systems*, Commun. Math. Stat., 5(1), 2017.
- **E Haber, LR**, *Stable Architectures for DNNs*, Inverse Problems, 2017.
- **Q. Li, L. Chen, C. Tai, Weinan E**, *Maximum Principle Based Algorithms*, arXiv, 2017.
- **B. Wang, B. Yuan, Z. Shi, S. Osher**, *ResNets Ensemble via the Feynman-Kac Formalism*, arXiv, 2018.

Numerical Time Integrators

- **Y. Lu, A. Zhong, Q. Li, B. Dong**, *Beyond Finite Layer DNNs*, arXiv, 2017.
- **B. Chang, L. Meng, E. Haber, LR, D. Begert, E. Holtham**, *Reversible architectures for DNNs*, AAAI, 2018.
- **T. Chen, Y. Rubanova, J. Bettencourt, D. Duvenaud**, *Neural ODEs*, NeurIPS, 2018.
- **E. Haber, K. Lensink, E. Treister, LR**, *IMEXnet: Forward Stable DNN*. ICML, 2019.

Optimal Control

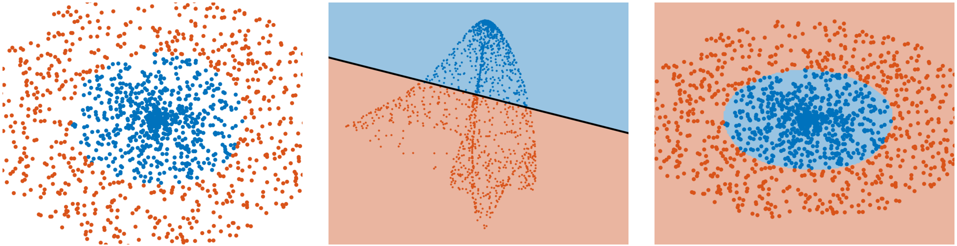
- **S. Günther, LR, J.B. Schroder, E.C. Cyr, N.R. Gauger**, *Layer-parallel training of ResNets*, arXiv, 2018.
- **A. Gholami, K. Keutzer, G. Biros**, *ANODE: Unconditionally Accurate Memory-Efficient Gradients for Neural ODEs*, arXiv, 2019.
- **T. Zhang, Z. Yao, A. Gholami, K. Keutzer, J. Gonzalez, G. Biros, M. Mahoney**, *ANODEV2: A Coupled Neural ODE Evolution Framework*, arXiv, 2019.

PDE-motivated Approaches

- **E. Haber, LR, E. Holtham**, *Learning across scales - Multiscale CNNs*, AAAI, 2018.
- **LR, E. Haber**, *DNNs motivated by PDEs*, arXiv, 2018.

Optimal Control

Optimal Control Framework for Deep Learning



training data, $\mathbf{Y}_0, \mathbf{C}$ | prop. features, $\mathbf{Y}(T), \mathbf{C}$ | classification result

Supervised Deep Learning Problem

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

$$\text{minimize}_{\boldsymbol{\theta},\mathbf{W},\boldsymbol{\mu}} \quad \text{loss}[g(\mathbf{W}+\boldsymbol{\mu}), \mathbf{C}] + \text{regularizer}[\boldsymbol{\theta}, \mathbf{W}, \boldsymbol{\mu}]$$

Optimal Control Background: Diff→Disc vs. Disc→Diff

$$\begin{aligned} \text{minimize}_{\boldsymbol{\theta},\mathbf{W},\boldsymbol{\mu}} \quad & \text{loss}[g(\mathbf{W}\mathbf{Y}(T)+\boldsymbol{\mu}),\mathbf{C}] + \text{regularizer}[\boldsymbol{\theta},\mathbf{W},\boldsymbol{\mu}] \\ \text{subject to} \quad & \partial_t \mathbf{Y}(t) = f\left(\mathbf{Y}(t),\boldsymbol{\theta}(t)\right), \ \ \mathbf{Y}(0)=\mathbf{Y}_0. \end{aligned}$$

- **First-Differentiate-then-Discretize (Diff→Disc)**
  - Keep $\theta, \mathbf{b}, \mathbf{Y}$ continuous in time
  - Euler-Lagrange-Equations $\rightsquigarrow$ adjoint equation ($\approx$ backprop)
  - **flexible choice of ODE solver in forward and adjoint**
  - **gradients only useful if fwd and adjoint solved well**
  - use optimization to obtain discrete solution of ELE
- **First-Discretize-then-Differentiate (Disc→Diff)**
  - Discretize $\theta, \mathbf{b}, \mathbf{Y}$ in time (could use different grids)
  - Differentiate objective (e.g., use automatic differentiation)
  - /**gradients related to adjoints but no choice of solver**
  - **gradients useful even if discretization is inaccurate**
  - use nonlinear optimization tools to approximate minimizer

MD Gunzburger
Perspectives in flow control and optimization.
SIAM, 2013.

TQ Chen et al.,
Neural Ordinary Differential Equations.
NeurIPS, 2018.

A Gholami, K Keutzer, G Biros
ANODE: Unconditionally Accurate Memory-Efficient Gradients for Neural ODEs.
arXiv:1902.10298

Example: The Adjoint Equation

Simplified learning problem: one example $(\mathbf{y}_0, \mathbf{c})$, no weights for classifier, no regularizer

$$\min_{\boldsymbol{\theta}} \mathrm{loss}(\mathbf{y}(1,\boldsymbol{\theta}), \mathbf{c}) \quad \text{subject to} \quad \partial_t \mathbf{y}(t,\boldsymbol{\theta}) = f(\mathbf{y}(t), \boldsymbol{\theta}(t)), \quad \mathbf{y}(0,\boldsymbol{\theta}) = \mathbf{y}_0.$$

Use adjoint method to compute gradient of objective w.r.t. $\boldsymbol{\theta}$

$$\frac{\partial \mathrm{loss}}{\partial \boldsymbol{\theta}}(t) = \left(\frac{\partial f}{\partial \boldsymbol{\theta}}(\mathbf{y}(t,\boldsymbol{\theta}), \boldsymbol{\theta}(t)) \right)^\top \mathbf{z}(t)$$

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

$$-\partial_t \mathbf{z}(t,\boldsymbol{\theta}) = \left(\frac{\partial f}{\partial \mathbf{y}}(\mathbf{y}(t,\boldsymbol{\theta}), \boldsymbol{\theta}(t)) \right)^\top \mathbf{z}(t), \quad \mathbf{z}(1,\boldsymbol{\theta}) = \frac{\partial \mathrm{loss}}{\partial \mathbf{y}}(\mathbf{y}(1,\boldsymbol{\theta}), \mathbf{c}).$$

note: $\mathrm{y}(t)$ needed for solve adjoint equation memory

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.
Nature, 533–536, 1986.

Multilevel Training of ResNets

Note: Training result depends on initial guess for optimization.

Multi-level learning

ResNet with n layers $\rightarrow h = T/n$
$\boldsymbol{\theta}^0, \mathbf{W}^0, \boldsymbol{\mu}^0 \leftarrow$ random initialization
for $\ell = 1:3$ **do**
  train ResNet with initial weights $\boldsymbol{\theta}^0, \mathbf{W}^0, \boldsymbol{\mu}^0$
  obtain $\boldsymbol{\theta}^*, \mathbf{W}^*, \boldsymbol{\mu}^*$
  $(n, h) \leftarrow (2n, h/2)$
  refine ResNet
  $\boldsymbol{\theta}^0 \leftarrow$ prolongate $\boldsymbol{\theta}^*$
  $(\mathbf{W}^0, \boldsymbol{\mu}^0) \leftarrow \mathbf{W}^*, \boldsymbol{\mu}^*$

FA Bornemann, P Deuflhard
The cascadic multigrid method for elliptic problems.
Numerische Mathematik, 1996.

B Chang et al.
Multi-level Residual Networks from Dynamical Systems View.
ICLR, 2018.

Stability and Well-Posedness

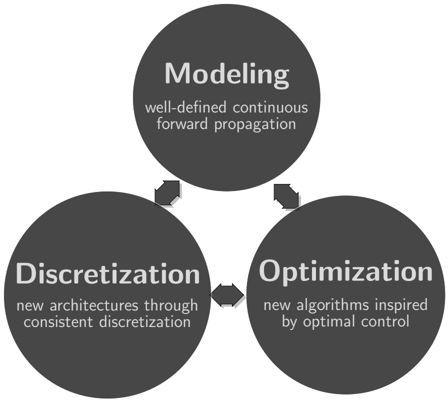
Stability of Deep Neural Networks: Motivation

Goal in learning: Build model that generalizes.

Todo list:

1. model forward dynamic
2. discretize forward dynamic ($\rightsquigarrow$ 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 o forward propagation

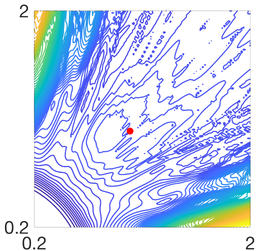
Impact of Network Architecture on Optimization - 1

$$\min_{\theta} \frac{1}{2}\|\mathbf{Y}_N(\theta) - \mathbf{C}\|_F^2 \qquad \mathbf{Y}_{j+1}(\theta) = \mathbf{Y}_j(\theta) + \frac{10}{N}\tanh\left(\mathbf{K}\mathbf{Y}_j(\theta)\right)$$

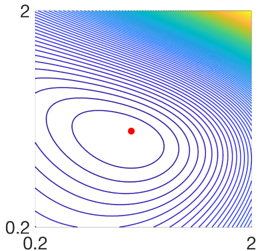
where $\mathbf{C} = \mathbf{Y}_{200}(1,1)$, $\mathbf{Y}_0 \sim \mathcal{N}(0,1)$, and

$$\mathbf{K}(\theta) = \begin{pmatrix} -\theta_1 - \theta_2 & \theta_1 & \theta_2 \\ \theta_2 & -\theta_1 - \theta_2 & \theta_1 \\ \theta_1 & \theta_2 & -\theta_1 - \theta_2 \end{pmatrix}$$

loss, $N = 5$

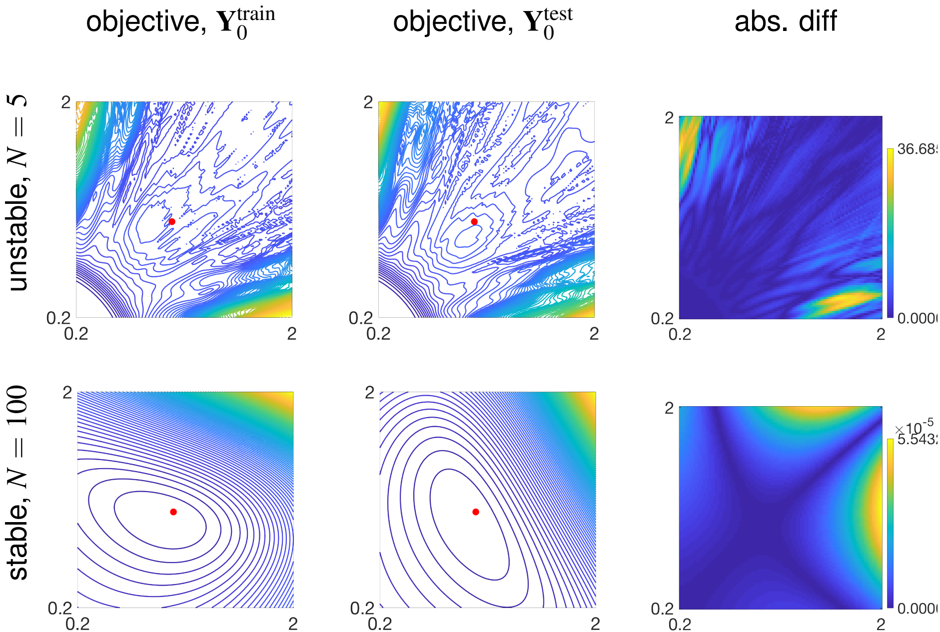


loss, $N = 100$



Next: Compare examples for different inputs $\sim$ generalization

Impact of Network Architecture on Optimization - 2



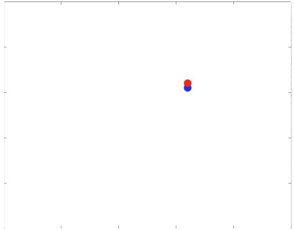
Stability of Continuous Forward Propagation

Interpret ResNet as discretization of initial value problem

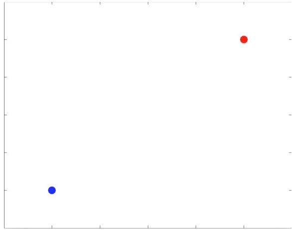
$$\partial_t \mathbf{y}(t,\boldsymbol{\theta},\mathbf{y}_0) = f(\mathbf{y}(t,\boldsymbol{\theta},\mathbf{y}_0),\boldsymbol{\theta}(t))$$
$$\mathbf{y}(0,\boldsymbol{\theta},\mathbf{y}_0) = \mathbf{y}_0.$$

IVP is stable if for any $\mathbf{v} \in \mathbb{R}^n$

$$\|\mathbf{y}(T,\boldsymbol{\theta},\mathbf{y}_0) - \mathbf{y}(T,\boldsymbol{\theta},\mathbf{y}_0+\epsilon\mathbf{v})\|^2 = \mathcal{O}(\epsilon\|\mathbf{v}\|).$$



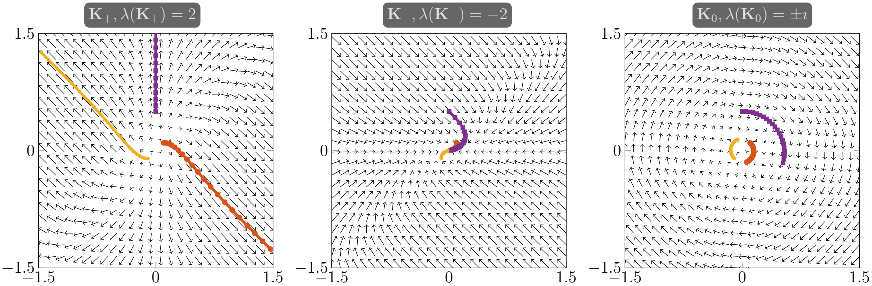
unstable IVP



stable IVP

idea: ensure stability by design / constraints on f and $\boldsymbol{\theta}$

Stability of Forward Propagation



Fact: The ODE $\partial_t \mathbf{y}(t) = f(\mathbf{y})$ is stable if the real parts of the eigenvalues of the Jacobian $\mathbf{J}$ are non-positive.

Example: Consider ResNet with stationary weights

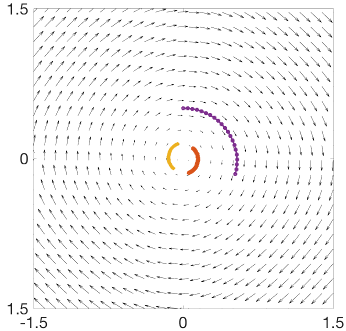
$$\partial_t \mathbf{y}(t) = \sigma\left(\mathbf{K}\mathbf{y}(t) + \mathbf{b}\right) \quad \Rightarrow \quad \mathbf{J}(t) = \mathrm{diag}(\sigma'(\mathbf{K}\mathbf{y}(t) + \mathbf{b}))\mathbf{K}.$$

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

Networks with non-stationary weights require additional arguments (e.g., kinematic eigenvalues) or assumptions ($\mathbf{J}$ changes slowly).

Enforcing Stability: Antisymmetric Transformation

Two examples of more stable networks.

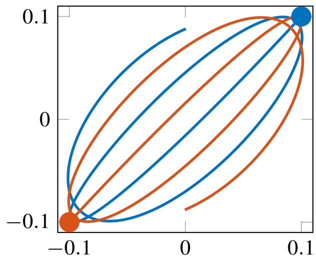


ResNet with antisymmetric transformation matrix

$$\partial_t \mathbf{y}(t) = \sigma((\mathbf{K}(t) - \mathbf{K}(t)^\top)\mathbf{y} + \mathbf{b}(t)).$$

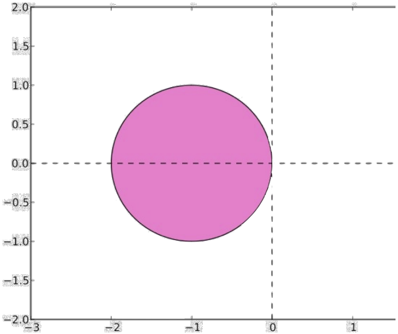
Hamiltonian-like ResNet

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

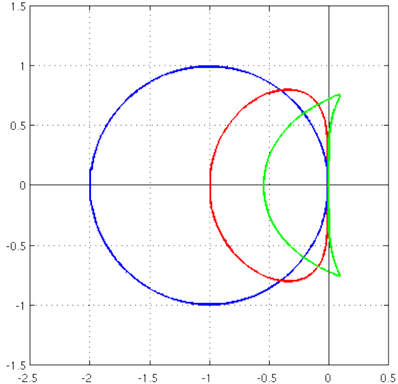


How about the stability of the discrete system?

Stability of Discrete Forward Problem



forward Euler



Bashforth family

ResNet not stable for layer $f_{\text{antisym}}(\mathbf{Y}, \mathbf{K}_j, \mathbf{b}_j) = \sigma\left((\mathbf{K}_j - \mathbf{K}_j^\top)\mathbf{Y} + \mathbf{b}_j\right)$.
Need to replace fwd Euler by, e.g.,

$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + \frac{h}{12}\left(23 f_{\text{antisym}}(\mathbf{Y}_j, \boldsymbol{\theta}_j) - 16 f_{\text{antisym}}(\mathbf{Y}_{j-1}, \boldsymbol{\theta}_{j-1}) + 5 f_{\text{antisym}}(\mathbf{Y}_{j-2}, \boldsymbol{\theta}_{j-2})\right).$$

Verlet Integration for Hamiltonian-inspired NNs

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

$$\mathbf{Z}_{j+\frac{1}{2}} = \mathbf{Z}_{j-\frac{1}{2}} - h\sigma\left(\mathbf{K}_j \mathbf{Y}_j + \mathbf{b}_j\right)$$
$$\mathbf{Y}_{j+1} = \mathbf{Y}_j + h\sigma\left(\mathbf{K}_j^\top \mathbf{Z}_{j+\frac{1}{2}} + \mathbf{b}_j\right)$$

Note that this is reversible. Given $\mathbf{Y}_N$ and $\mathbf{Z}_{N-\frac{1}{2}}$ and $j = N-1, \ldots, 1$ do

$$\mathbf{Y}_j = \mathbf{Y}_{j+1} - h\sigma\left(\mathbf{K}_j^\top \mathbf{Z}_{j+\frac{1}{2}} + \mathbf{b}_j\right)$$
$$\mathbf{Z}_{j-\frac{1}{2}} = \mathbf{Z}_{j+\frac{1}{2}} + h\sigma\left(\mathbf{K}_j \mathbf{Y}_j + \mathbf{b}_j\right)$$

Notes:

- reversibility often exploited in 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
arXiv 1707.04585, 2017.

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 $F(\mathbf{Y}) = \alpha\mathbf{Y}$ and $G(\mathbf{Z}) = \beta\mathbf{Z}$, i.e.,

$$\mathbf{Z}_{j+\frac{1}{2}} = \mathbf{Z}_{j-\frac{1}{2}} - \alpha\mathbf{Y}_j, \quad \text{and} \quad \mathbf{Y}_{j+1} = \mathbf{Y}_j + \beta\mathbf{Z}_{j+\frac{1}{2}}.$$

Combining two time steps in $\mathbf{Y}$

$$\mathbf{Y}_{j+1} - \mathbf{Y}_j = \beta\mathbf{Z}_{j+\frac{1}{2}}, \quad \text{and} \quad \mathbf{Y}_j - \mathbf{Y}_{j-1} = \beta\mathbf{Z}_{j-\frac{1}{2}}$$

Subtracting those two gives

$$\begin{aligned}\mathbf{Y}_{j+1} - 2\mathbf{Y}_j + \mathbf{Y}_{j-1} &= \beta(\mathbf{Z}_{j+\frac{1}{2}} - \mathbf{Z}_{j-\frac{1}{2}}) = \alpha\beta\mathbf{Y}_j \\ \Leftrightarrow \mathbf{Y}_{j+1} - (2+\alpha\beta)\mathbf{Y}_j + \mathbf{Y}_{j-1} &= 0\end{aligned}$$

There is a solution $\mathbf{Y}_j = \xi^j$, i.e., with $a = (2+\alpha\beta)/2$

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

$|\xi|^2 = 1$ (stable) if $a^2 \leq 1$. Otherwise ξ growing!

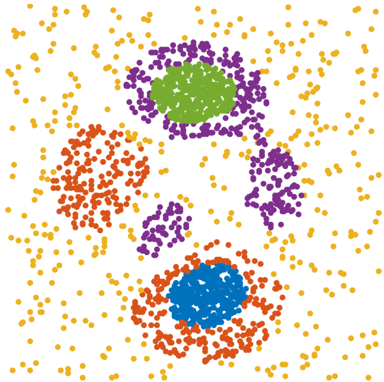
Example: Impact of Discretization on Training

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

features and labels

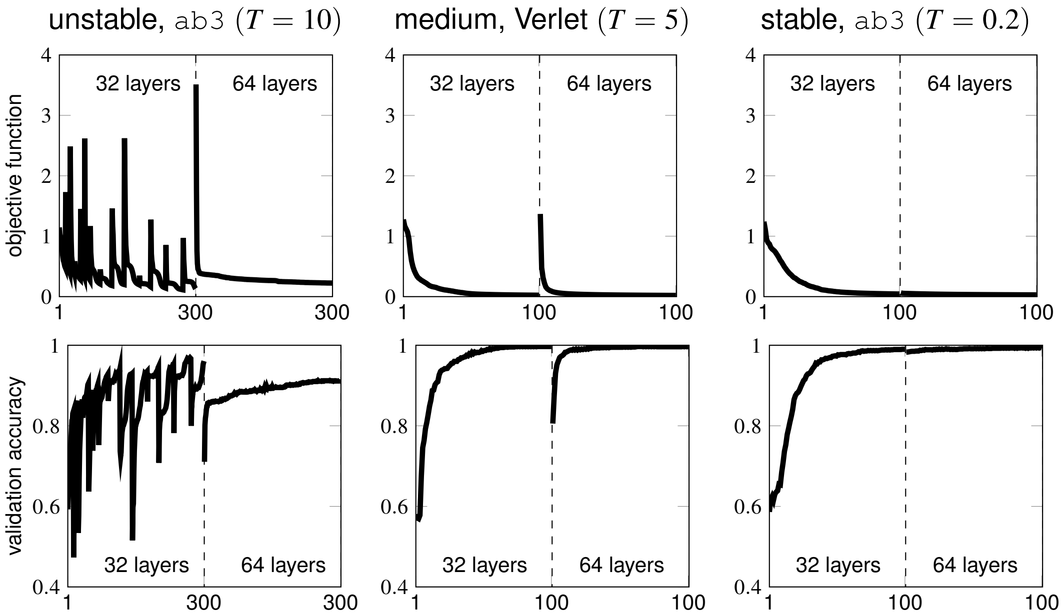


compare three configurations

1. ”unstable”: $T = 10$ (3rd order multistep)
2. ”medium”: $T = 5$ (1st order Verlet)
3. ”stable”: $T = 0.2$ (3rd order multistep)

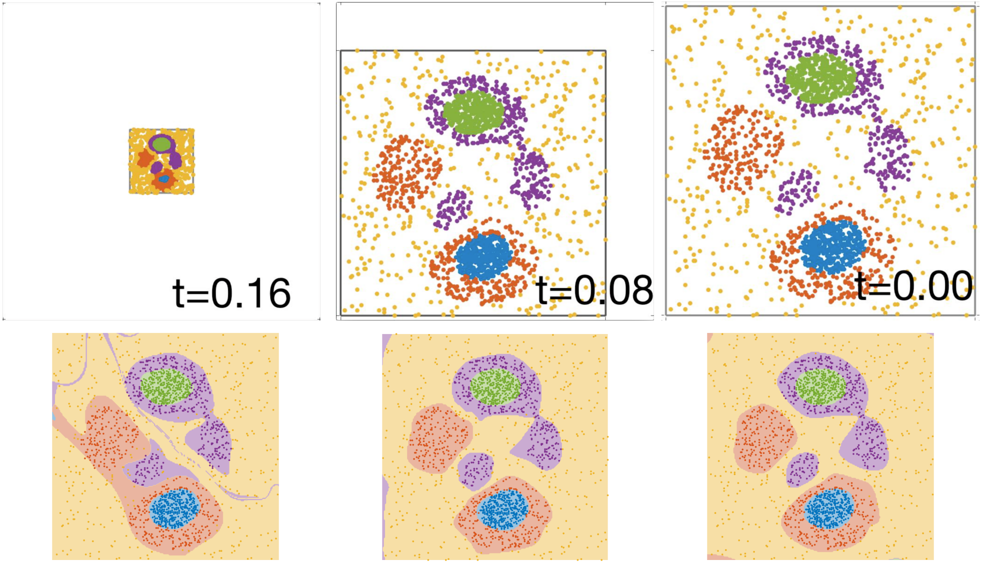
Q: how does learning performance compare?

Example: Impact of ODE Solver - Convergence



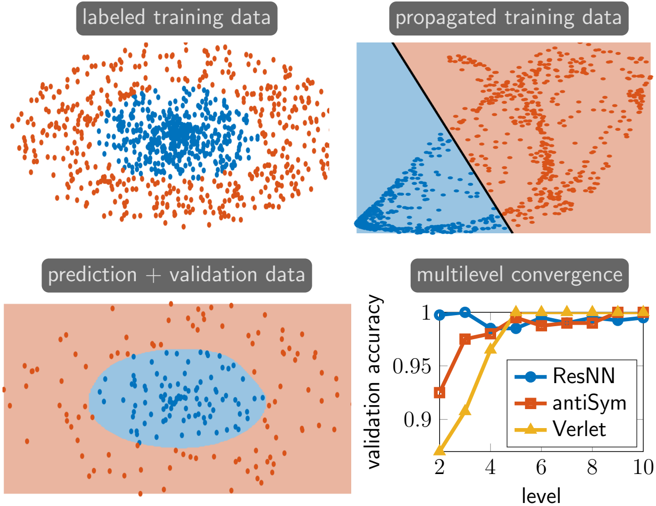
Example: Impact of ODE Solver - Dynamics

unstable, $\texttt{ab3}$ $(T = 10)$ medium, Verlet $(T = 5)$ stable, $\texttt{ab3}$ $(T = 0.2)$



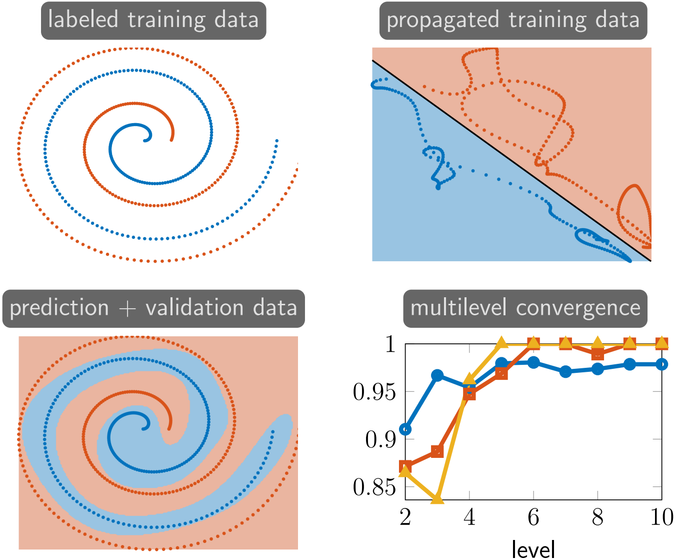
Example (Ellipses): Hamitonian-like network with Verlet

- 2D feature space, concentric ellipses
- 1k training, 2k validation
- multilevel: 2 $\to$ 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 $\rightarrow$ 1024 layers
- optimization: block coordinate descent with Newton-PCG
- weight decay (Tikhonov) regularization
- $\tanh$ activation, width 4



Layer-Parallel Training

Layer-Parallel Training of Deep Residual Neural Networks

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

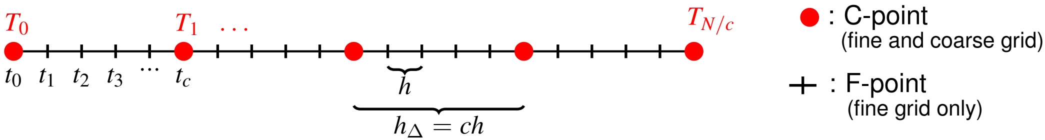
Full-space version of the optimal control formulation of supervised learning

$$\underset{\boldsymbol{\theta}_0,\ldots,\boldsymbol{\theta}_{N-1},\mathbf{W},\mathbf{Y}_1,\ldots,\mathbf{Y}_N}{\text{minimize}} \quad \text{loss}[g(\mathbf{W}\mathbf{Y}_N),\mathbf{C}] + \text{regularizer}[\boldsymbol{\theta},\mathbf{W}]$$

$$\text{subject to} \quad \begin{aligned} \mathbf{Y}_1 &= \mathbf{Y}_0 + hf(\mathbf{Y}_0,\boldsymbol{\theta}_0) \\ \mathbf{Y}_2 &= \mathbf{Y}_1 + hf(\mathbf{Y}_1,\boldsymbol{\theta}_1) \\ &\ \vdots \\ \mathbf{Y}_N &= \mathbf{Y}_{N-1} + hf(\mathbf{Y}_{N-1},\boldsymbol{\theta}_{N-1}) \end{aligned}$$

Recall: Constraints can be eliminated (explicit Euler, sequential).
Goal: Stay in full-space and create parallelism in time

Example: Elimination vs. Iterative Solve

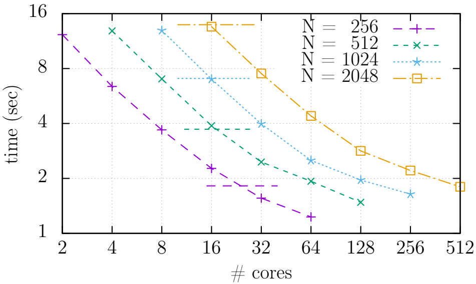


Consider linear dynamics $\partial_t \mathbf{y} = \mathbf{A}\mathbf{y}$. Forward Euler discretization is

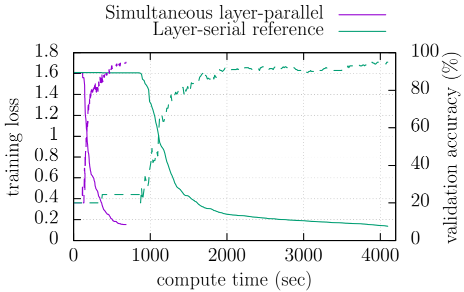
$$\begin{pmatrix} \mathbf{I} & & & & \\ -(\mathbf{I}+h\mathbf{A}) & \mathbf{I} & & & \\ & -(\mathbf{I}+h\mathbf{A}) & \mathbf{I} & & \\ & & \ddots & \ddots & \\ & & & -(\mathbf{I}+h\mathbf{A}) & \mathbf{I} \end{pmatrix} \begin{pmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \\ \mathbf{y}_3 \\ \vdots \\ \mathbf{y}_N \end{pmatrix} = \begin{pmatrix} \mathbf{y}_0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix}$$

- Option 1: forward substitution (optimal complexity but sequential)
- Option 2: use iterative solver higher complexity but parallel)
- Option 3: multigrid in time (MGRIT)

Scalability for Forward and Simultaneous Optimization



strong scaling (fwd + gradient)



simultaneous optimization

- MGRIT provides a new form of parallelism (in addition to data parallelism)
- Use-case 1: Replace forward and backward propagation in SGD
- Use-case 2: simultaneous optimization (more intrusive)

S. Günther, LR, J. B. Schroder, E. C. Cyr, N. R. Gauger
Layer-Parallel Training of Deep Residual Neural Networks.
in revision, SIMODS, 2019.

Conclusion

Σ: Deep Neural Networks motivated by ODEs

Optimal control formulation

- new insights, theory, algorithms

Stability and well-posedness

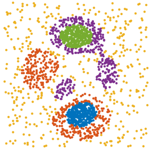
- differentiate-then-discretize vs. **discrete-then-differentiate**
- examples: impact on optimization/generalization

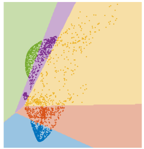
Numerical Methods: Discretize-Optimize

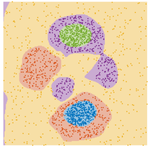
- Verlet: reversible and stable networks (memory-free)
- Parallel-in-Layer: additional option for parallelism

DNNs motivated by PDEs **(tomorrow)**

- parabolic CNNs, hyberbolic CNNs, IMEX-Net, Lean ResNets







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

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

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.
arXiv, 2018.

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: DMS 1522599, DMS 1751636 US-Israel BSF 2018209

Other support: Donation of a TITAN X GPU