Deep Learning $\iff$ Partial Differential Equations

SIAM/CAIMS Annual Meeting, 2020

slides:
mathcs.emory.edu/~lruthotto/talks/2020-SIAMCAIMS.pdf

Lars Ruthotto
Departments of Mathematics and Computer Science, Emory University
lruthotto@emory.edu
@lruthotto
Core of Science: Understanding the World Through Models and Data

Deep Learning (DL)
1. model: deep neural network
2. data: ImageNet $\geq 14$M images
3. key: generalize beyond data

Partial Differential Equations (PDE)
1. models: elasticity, Navier stokes, …
2. data: 1-3 images of patient
3. key: analyze, discretize, solve PDE

Today: Connecting DL and PDEs

(Deng et al. 2009; Krizhevsky et al. 2012; He et al. 2015)

(Lin et al. 2017; Lefieux et al. 2020; Viguerie and Veneziani 2019)
Deep Learning in a Nutshell

- DL := machine learning (ML) with deep neural networks (DNN)
- DNN := neural network with many layers
- AI research activity follows waves, starting \( \approx 1950s \)
- new surge due to massive datasets and computing power
- excellent references: Goodfellow et al. 2016; Higham and Higham 2018
Computational and Applied Mathematicians’ Role in DL

An (almost perfectly) true statement

\[ \text{lots of data} + \text{back propagation} + \text{GPU} + \{ \text{TensorFlow, Caffe, Torch} \} \Rightarrow \text{success!} \]

So, why study the mathematics of deep learning?
Fundamental Questions and Recent Mathematical Advances

**Expressibility, Approximation Properties**
- why does the DNN succeed (or fail) to approximate a function/operator?
- recent works: Poggio et al. 2017; Bölcskei et al. 2019; Lu et al. 2019a

**Learning, Optimization, Generalization**
- why do some optimization algorithms lead to better generalization than others?
- recent works: Bottou et al. 2018; Zhang et al. 2018; Chizat and Bach 2020; Osher et al. 2018
- Tue, 5PM MT1: Generalization Theory in Machine Learning (A. Oberman)

**Explainability, Interpretability**
- why does the neural network choose a certain prediction?
- recent works: Samek et al. 2017; Montavon et al. 2018; Adebayo et al. 2018

**Robustness, Adversarial Attacks, Stability**
- why can DNNs be easily fooled by perturbing input features or training data?
- recent works: Madry et al. 2017; Shafahi et al. 2018; Wang et al. 2018; Etmann et al. 2019

**Fairness**
- does my DNN discriminate based on sensitive features (e.g., gender, ethnicity)?
- recent works: Friedler et al. 2016; Kleinberg et al. 2016; Bellamy et al. 2018

**Scientific Use of Machine Learning**
- recent works: Lusch et al. 2018; Raissi et al. 2019; Han et al. 2018; Arridge et al. 2019
- July 20 – July 24: Mathematical and Scientific Machine Learning (MSML 2020)
Main goal: highlight connections between PDEs, ODEs, and DL.

← insight, efficiency, robustness ←

part 1

DL \rightleftharpoons ODE \rightleftharpoons PDE

part 2

→ tackle curse of dimensionality →

part 3

Collaborators and Funding

Funding:
DMS 1751636, 1522599  BSF 2018209  MLP 2019
Up next: ODE $\rightarrow$ DL
Example: Supervised Classification with a DNN
ResNet: Residual Neural Networks (He et al. 2016)

Training data \(\{(y^{(1)}, c^{(1)}), (y^{(2)}, c^{(2)}), \ldots\} \subset \mathbb{R}^2 \times \{0, 1\}\).

Forward propagation of input \(y\) through simple ResNet

\[
\begin{align*}
\mathbf{u}_0 &= K_{\text{in}} y \\
\mathbf{u}_1 &= \mathbf{u}_0 + h \sigma(K_0 \mathbf{u}_0 + b_0) \\
& \vdots \\
\mathbf{u}_N &= \mathbf{u}_{N-1} + h \sigma(K_{N-1} \mathbf{u}_{N-1} + b_{N-1}) \\
\mathbf{z} &= s(K_{\text{out}} \mathbf{u}_N + b_{\text{out}}),
\end{align*}
\]

with \(h > 0\), \(\theta_i^{\text{Res}} := (K_i, b_i)\).

Let \(F(y, \theta) := \mathbf{z}\), weights \(\theta := (\theta_0^{\text{Res}}, \ldots, \theta_{N-1}^{\text{Res}}, K_{\text{out}}, b_{\text{out}})\).

Train weights by solving (Bottou et al. 2018)

\[
\min_{\theta} \mathbb{E} [\ell(F(y, \theta), c)] + \frac{\alpha}{2} \|\theta\|_2^2,
\]

with cross entropy loss \(\ell(z, c) = -c \log(z) - (1 - c) \log(1 - z)\).
ResNet: Discussion

In ResNet, \( u_N \) is forward Euler approximation of \( u(T) \),

\[
\partial_t u(t) = f(u(t), \theta^\text{ODE}(t)), \quad t \in (0, T], \quad u(0) = u_0;
\]

see (E 2017; Haber and Ruthotto 2017) (\( \approx \) same time).

Advantages over other Architectures

1. ResNets often improve with depth
2. state-of-the-art results for many tasks
3. easy to train and easy to add depth

Remarks

1. \( f(u, \theta^\text{ODE}(t)) = \sigma(K(t)u + b(t)) \) gives ResNet on previous slide
2. in practice: more complicated layer \( f \), concatenate ResNets to change width or image resolution
3. similar continuous networks, extensions to PDEs, and implicit time integrators in (Rico-Martínez et al. 1992; González-García et al. 1998).
Stable Architectures for DNNs (Haber and Ruthotto 2017)

When is forward propagation stable? That is, when \( \exists M > 0 \) such that

\[
\| F(y + \epsilon, \theta) - F(y, \theta) \| \leq M \| \epsilon \| \quad (\epsilon \text{ input perturbation})
\]

Motivation: well-posed training problem, adversarial attacks, efficient optimization, …

Main Findings and Contributions
1. \( \partial_t u(t) = \sigma(K(t)u(t) + b(t)) \) not stable for all \( K(\cdot), b(\cdot) \)
2. alternative \( f \): antisymmetric \( K \), Hamiltonian-inspired networks
3. symplectic integrators to obtain stable architecture (\( \neq \) ResNet)
4. stable DNNs perform competitively (on simple tasks)

Improvements and Related Works
1. more expressive architectures (Chang et al. 2018), multilevel training (Chang et al. 2017)
2. improved stability results (Ruthotto and Haber 2020; Celledoni et al. 2020)
3. analysis: convergence (Thorpe and Gennip 2018), opt. conditions (Benning et al. 2019)
4. multi-step and other time integrators (Lu et al. 2017)
5. discrete weights (Li and Hao 2018), maximum principles (Li et al. 2017)
Neural ODEs: Neural Ordinary Differential Equations (Chen et al. 2018)

Main Novelties and Contributions
1. apply adaptive time integrator to continuous ResNet
2. compute gradients of loss function using adjoint equation
   \[-\partial_t p(t) = \nabla f(u(t), \theta(t))^\top p(t), \quad p(T) = \nabla u_N \ell(F(y), c)\]
3. save memory by re-computing $u(t)$ backward in time with $p$.
4. popularized continuous models in ML community

Improvements and Related Works
1. example for numerical instability of item 3 above (Gholami et al. 2019) and alternative using checkpointing
2. invertible ResNet (Behrmann et al. 2019), generative modeling (Grathwohl et al. 2018; Chen et al. 2019)
3. augmented, more expressive models (Dupont et al. 2019)
Optimize-Discretize vs. Discretize-Optimize (Gholami et al. 2019)

Compare optimal control approaches for learning problems like

\[
\min_{\theta} \mathbb{E} [\ell(F(y, \theta), c)] + \frac{\alpha}{2} \|\theta\|_2^2,
\]

where \(u_N\) in \(F\) is approximately equal to \(u(T)\) given by

\[
\partial_t u(t) = f(u(t), \theta^{\text{ODE}}(t)), \quad t \in (0, T], \quad u(0) = u_0.
\]

**O \rightarrow D: Optimize-Discretize (Neural ODE)**

1. keep \(\theta^{\text{ODE}}, u\) continuous in time
2. Euler-Lagrange-Equations \(\Rightarrow\) adjoint equation
3. use adaptive time integrators in optimization

**D \rightarrow O: Discretize-Optimize (ANODE)**

1. discretize \(\theta^{\text{ODE}}, u\) in time (could use different grids)
2. differentiate discrete problem \(\Rightarrow\) backpropagation
3. keep discretization fixed during optimization

**My advice:** use \(D \rightarrow O\) (● accurate gradients, ● fixed costs, ● convergence, ● generalization)
Layer-Parallel Training of Deep ResNets (Günther et al. 2020)

**Idea:** Train ResNet with parallel-in-time methods from time-dependent control (Falgout et al. 2014)

1. replace (sequential) forward and backward propagation with (parallel) nonlinear multigrid iteration
2. simultaneously iterate on $\theta$ and $u, p$.

**Findings and Contributions (Indian Pines Example):**

1. parallel multigrid faster when using $\geq 16$ cores
2. simultaneous optimization speeds up training by $4.4 \times$ with 128 cores ($\approx 3.5\%$ efficiency)
3. layer-parallelism is new way to parallelize and distribute ResNet training (in addition to data parallelism)

**Improvements and Related Works:**

1. nested iterations increase performance (Cyr et al. 2019)
2. more efficient parallel-in-time (Parpas and Muir 2019)
Up next: PDE → DL
Convolutional Neural Networks (CNNs) for Speech, Image, Video Data

Example: digit recognition (LeCun et al. 1990)

Key Challenges (Segmentation Example):
1. input: images consist of $\approx 700K$ pixels
2. output: label each pixel into one of 32 classes
3. convolutions act locally $\Rightarrow$ need many layers
4. need efficient and robust prediction

$\Rightarrow$ need computing power, storage, new ideas
Lessons from PDE-Based Image Processing

A few seminal works

- optical flow (Horn and Schunck 1981)
- elasticity for image registration (Broit 1981)
- variational methods for image segmentation (Mumford and Shah 1989)
- total variation for edge-preserving denoising (Rudin et al. 1992)
- nonlinear diffusion (Perona et al. 1994; Scherzer and Weickert 2000; Weickert 2009)

Common thread: Replace discrete images and operators with functions and PDE $\Rightarrow$ better understanding, improved robustness, higher efficiency.

see: SIAM Bookstore, SIAM IS20 (e.g., IP2 by Thomas Pock, MS 53, MS 70)
Deep Neural Networks Motivated by PDEs (Ruthotto and Haber 2020)

Idea: design CNNs that inherit properties of PDEs.

Main Findings and Contributions
1. stability result for non-autonomous \textbf{parabolic CNNs}
2. first-order and second-order \textbf{hyperbolic CNNs}
3. hyperbolic: symplectic integrators \(\sim\) \(\downarrow\) memory
4. different PDEs lead to competitive performance

Improvements and Related Works
1. reversibility first used in (Chang et al. 2018) to train 1202-layer ResNet on one GPU
2. overcome field-of-view problem with semi-implicit time-stepping (Haber et al. 2019)
3. reduce massive number of CNN weights \(\mathcal{O}(10^6)\) using lean operators (Ephrath et al. 2020a) and multigrid-in-channel (Ephrath et al. 2020b)
4. (Lensink et al. 2019) added wavelets to alleviate all memory requirements
5. related: parameter estimation (González-García et al. 1998), reaction diffusion for denoising (Chen and Pock 2017)
Up next: DL $\rightarrow$ PDE
Example: Deep Learning for High-Dimensional PDEs

Consider this PDE problem

\[-\Delta \Phi(x) = g(x), \quad x \in \Omega \subset \mathbb{R}^d, \quad \Phi(s) = 0, \quad s \in \partial\Omega.\]

Idea: Parameterize \(\Phi(\cdot)\) with a neural network \(F(\cdot, \theta)\) and solve

\[
\min_{\theta} \int_{\Omega} \frac{1}{2} \|\nabla F(x, \theta)\|^2 - F(x, \theta)g(x)dx + \frac{\lambda}{2} \int_{\partial\Omega} F(s, \theta)^2 ds.
\]

Short discussion

- DNNs are mesh-free and scale to high-dimensions
- use PDE (no training data needed)
- linear PDE \(\Rightarrow\) non-convex optimization problem

A few (of many) recent works in this area

2. Nonlinear Black-Scholes, Hamilton-Jacobi Bellman, Allen-Cahn (Han et al. 2018)
4. theoretical results (Kutyniok et al. 2019; Shin et al. 2020)
ML for High-Dimensional Mean Field Games (Ruthotto et al. 2020)

**Idea:** Use DNNs for optimal transport and mean field games.
1. variational approach, Hamilton-Jacobi-Bellman (HJB) penalty
2. simulate population density with Lagrangian PDE solver
3. tailored neural network, fast gradient and Laplacian computations

**Main Findings and Contributions**
1. $d = 2$: tie with convex solver (Haber and Horesh 2015)
2. synthetic transport and crowd motion problems up to $d = 100$
3. HJB penalty: more efficiency and accuracy

**Improvements and Related Work**
1. ML applications: (Yang and Karniadakis 2019; Finlay et al. 2020; Onken et al. 2020)
2. stochastic mean field games (Lin et al. 2020)
3. CP 1: Derek Onken’s poster
4. Wed 3:30 PM: Levon Nurbekyan’s talk in MS 18
Up next: Summary
Deep Learning and PDEs: Related Activities

SIAM/CAIMS Annual Meeting

- CP1 / MS34: Contributed and Poster
- MS 8: Reinforcement Learning and Behavioral Modeling
- MS 9: Tutorial on Emerging Research Areas
- MT1: Generalization Theory (Adam Oberman)
- JP1: Optimal Transport Theory for Machine Learning (Gabriel Peyré)
- MS 16 (3 parts): Developments in Machine Learning
- MS 18 (3 parts): Intersection of Optimal Control and ML
- MS 68: Sparse Recovery and ML
- IP 13: Solving Eigenvalue Problems in High Dimension (Jianfeng Lu)

SIAM Imaging Sciences 2020

- IP 2: Variational Networks (Thomas Pock)
- IP 5: Deep Internal Learning (Michael Irani)
- IP 6: Deep Learning in Wave-based Imaging and Inverse Problems (Maarten V. de Hoop)
- MT 1: Model-Versus Learning-Based Approaches to Image Reconstruction (Moeller and Cremers)
- MS 8 (Learning and Processing of Geometric Structures), MS 12 (Semi-Supervised Learning), MS 15 (Learning Priors), MS 32 (Learning Imaging Operators), MS 47
Lectures and Seminars Available On-Demand

SIAM MDS 2020

- Eldad Haber, *Deep Neural Nets meet ODE/PDEs* (65 mins)
- Christoph Reisinger, *DL for Optimal Control and PDEs* (56 mins)
- Weinan E, *A Mathematical Perspective of ML* (60 mins)
- **MS119: Advances in Optimal Control for and with ML** (3 talks)
- **MS130: Advances in Optimal Control for and with ML** (4 talks)

IPAM

- Levon Nurbekyan, *Computational Methods for Mean Field Games* (2 parts)
- Lars Ruthotto, *Deep Neural Networks as ODE/PDE* (2 parts)
- Lars Ruthotto, *Numerical Analysis Perspective on DNNs* (56 mins)

Other Talks

- Samy Wu Fung, *APAC Net - DL for Stochastic Mean Field Games* (45 mins)
- Lars Ruthotto, *ML ↔ Optimal Transport* (48 mins)
Deep Learning $\iff$ Partial Differential Equations

Let us correct our statement from above:

$$\text{data} + \text{back propagation} + \text{GPU} + \left\{ \begin{array}{c} \text{TensorFlow} \\ \text{Caffe} \\ \text{Torch} \\ \vdots \end{array} \right\} + \text{mathematics} \Rightarrow \text{success}$$

What we covered:
- **PDE $\rightarrow$ DL**: insight, efficiency, robustness
- **DL $\rightarrow$ PDE**: tackle curse of dimensionality

What we did not cover:
- unsupervised, semi-supervised, reinforcement, active learning
- **DL $+$ PDE**: combine data and models

Short Q&A now! More at emory.zoom.us/my/lruthotto
Acknowledgements

Mentors, Colleagues, Collaborators:

Michele Benzi
Martin Burger
Chen Greif
Joseph Hart
James Herring
James Nagy
Jan Modersitzki
Alessandro Veneziani
Bart Van Bloemen Waanders
Yuanzhe Xi

Thanks to SIAM / CAIMS!

Funding:
DMS 1751636, 1522599
BSF 2018209
MLP 2019


References (cont.)


Chen, Tian Qi et al. (June 2018). “Neural Ordinary Differential Equations”. In: NeurIPS.


References (cont.)


Finlay, Chris et al. (2020). How to train your neural ODE: the world of Jacobian and kinetic regularization. arXiv:.

Friedler, Sorelle A. et al. (2016). On the (im)possibility of fairness. arXiv:.


References (cont.)


He, K et al. (2015). “Delving deep into rectifiers: Surpassing human-level performance on imagenet classification”. In: Proceedings of the IEEE ICCV.


References (cont.)

- Lensink, Keegan et al. (2019). *Fully Hyperbolic Convolutional Neural Networks*. arXiv:
- Lin, Alex Tong et al. (2020). *APAC-Net: Alternating the Population and Agent Control via Two Neural Networks to Solve High-Dimensional Stochastic Mean Field Games*. arXiv:
- Lu, Lu et al. (July 2019b). “DeepXDE: A deep learning library for solving differential equations”. In: *arXiv:*


References (cont.)


Wang, Bao et al. (2018). ResNets Ensemble via the Feynman-Kac Formalism to Improve Natural and Robust Accuracies. arXiv:.

