Train Like a (Var)Pro: Efficient Neural Network Training with Variable Projection

Elizabeth Newman[1] Lars Ruthotto[1]
Joseph Hart[2] Bart van Bloemen Waanders[2]

[1]Emory University

[2]Sandia National Laboratories

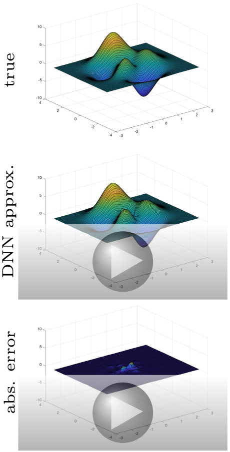
February 12, 2021

Funding Acknowledgements:

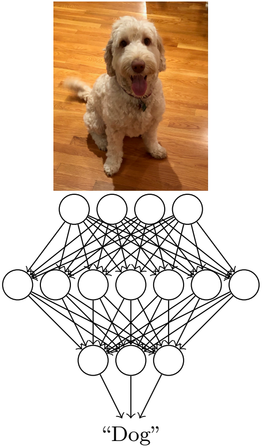
DE-NA-0003525. SAND2020-7339 FA9550-20-1-0372 ASCR 20-023231

A Shallow Look at Deep Neural Networks

Function Approximation



Classification



Goodfellow, Bengio, and Courville 2016; Raghu and Schmidt 2020

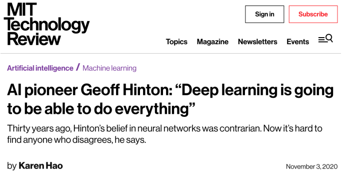
The DNN Buzz

The Hype

→ expressibility (Cybenko 1989; Poggio et al. 2017)

→ efficient approximators (Tripathy and Bilionis 2018; Han, Jentzen, and Weinan 2018)

→ versatility



Applications, Applications, Applications

→ computer vision (He et al. 2016; Krizhevsky, Sutskever, and Hinton 2012)

→ speech recognition (Hinton et al. 2012; Song 2015)

→ scientific applications (Han, Jentzen, and E 2018; Raissi, Perdikaris, and Karniadakis 2019)

The Challenges

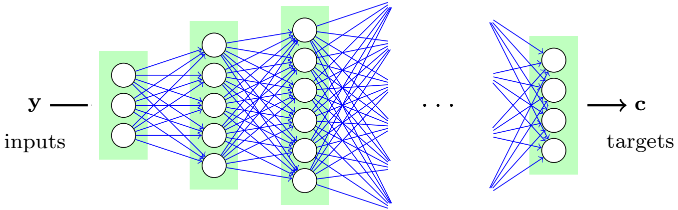
→ generalizability (Keskar et al. 2017; Papernot et al. 2016; Moosavi-Dezfooli, Fawzi, and Frossard 2016)

→ explainability (Samek et al. 2015; Montavon, Samek, and Mller 2018; Adebayo et al. 2020)

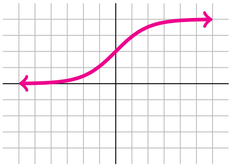
→ **inefficient training** (Li et al. 2018)

The Anatomy of a Neural Network

Features **Weights** **Nonlinearity**

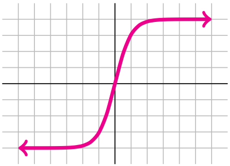


sigmoid



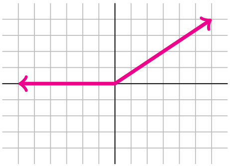
$\sigma(x) = \frac{1}{1+e^{-x}}$

tanh



$\sigma(x) = \tanh(x)$

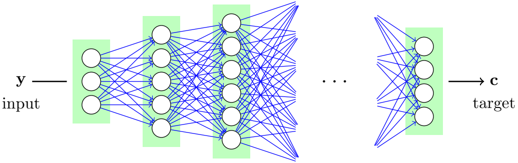
ReLU



$\sigma(x) = \max(x, 0)$

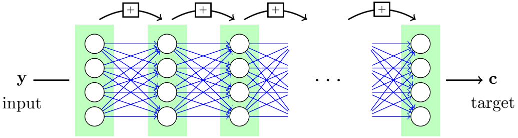
The Anatomy of a Neural Network

Feedforward Network



$$\begin{aligned}\mathbf{u}_1 &= \sigma(\mathbf{K}_{\text{in}}\mathbf{y} + \mathbf{b}_{\text{in}})\\ \mathbf{u}_2 &= \sigma(\mathbf{K}_1\mathbf{u}_1 + \mathbf{b}_1)\\ &\vdots\\ \mathbf{u}_d &= \sigma(\mathbf{K}_{d-1}\mathbf{u}_{d-1} + \mathbf{b}_{d-1})\\ \mathbf{x} &= s(\mathbf{K}_{\text{out}}\mathbf{u}_d + \mathbf{b}_{\text{out}})\end{aligned}$$

Residual Network (He et al. 2016; Ruthotto and Haber 2019; Weinan and Yu 2018)

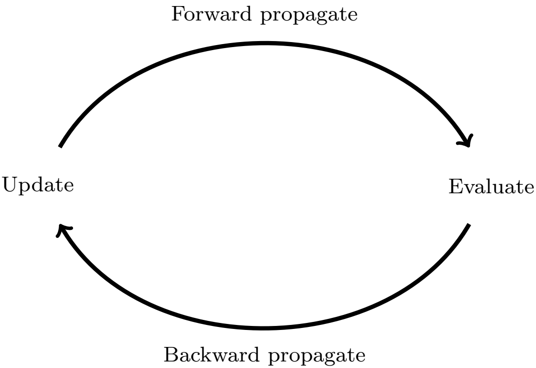


$$\begin{aligned}\mathbf{u}_1 &= \sigma(\mathbf{K}_{\text{in}}\mathbf{y} + \mathbf{b}_{\text{in}})\\ \mathbf{u}_2 &= \mathbf{u}_1 + h\sigma(\mathbf{K}_1\mathbf{u}_1 + \mathbf{b}_1)\\ &\vdots\\ \mathbf{u}_d &= \mathbf{u}_{d-1} + h\sigma(\mathbf{K}_{d-1}\mathbf{u}_{d-1} + \mathbf{b}_{d-1})\\ \mathbf{x} &= s(\mathbf{K}_{\text{out}}\mathbf{u}_d + \mathbf{b}_{\text{out}})\end{aligned}$$

$$\mathbf{x} = \text{DNN}(\mathbf{y}, \bar{\boldsymbol{\theta}}) \quad \text{where} \quad \bar{\boldsymbol{\theta}} = (\mathbf{K}_{\text{in}}, \mathbf{b}_{\text{in}}, \mathbf{K}_1, \mathbf{b}_1, \ldots, \mathbf{K}_{\text{out}}, \mathbf{b}_{\text{out}})$$

How to Train Your Network

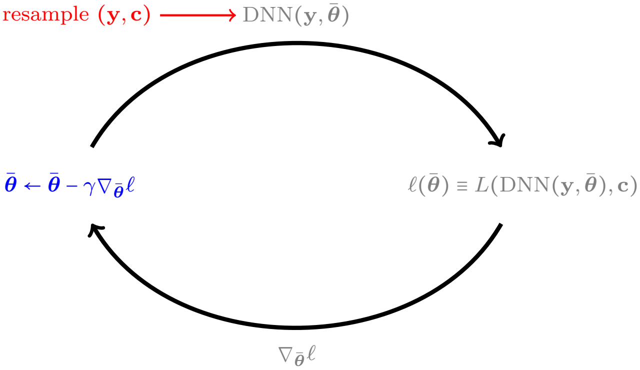
For one input-target pair $(\mathbf{y}, \mathbf{c})$,



Nielsen 2017

How to Train Your Network

For one input-target pair $(\mathbf{y}, \mathbf{c})$,



Nielsen 2017

Two Schools of Training

Stochastic Approximation (SA)
Minimize expected loss

$$\min_{\bar{\boldsymbol{\theta}}} \mathbb{E}[L(\text{DNN}(\mathbf{y}, \bar{\boldsymbol{\theta}}), \mathbf{c})]$$

- ☺ Memory-efficient
- ☺ Generalization
- ☹ Sensitive to hyperparameters
- ☹ Slow to converge

Common methods are SGD variants such as ADAM (Kingma and Ba 2014)

Sample Average Approximation (SAA)
Minimize approximated expected loss

$$\min_{\bar{\boldsymbol{\theta}}} \frac{1}{|\mathcal{T}|} \sum_{(\mathbf{y},\mathbf{c})\in\mathcal{T}} L(\text{DNN}(\mathbf{y}, \bar{\boldsymbol{\theta}}), \mathbf{c})$$

- ☺ Deterministic
- 😐 Dependent on large samples
  - ☺ Potentially parallelizable
  - ☹ Expensive memory-wise

Amenable to, e.g., Newton-Krylov schemes (O'Leary-Roseberry, Alger, and Ghattas 2019)

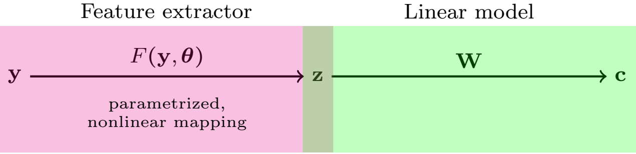
Improving Training

→ employ second-order methods (O'Leary-Roseberry, Alger, and Ghattas 2019; Yao et al. 2020; Bollapragada, Byrd, and Nocedal 2018)

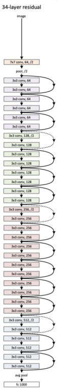
→ **choose optimal network weights** (Cyr et al. 2019; Sjöberg and Viberg 1997)

Kleywegt, Shapiro, and Mello 2002; Linderoth, Shapiro, and Wright 2006; Nemirovski et al. 2009

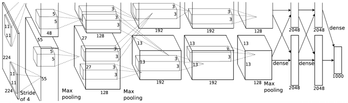
A Classy Way to DNNs



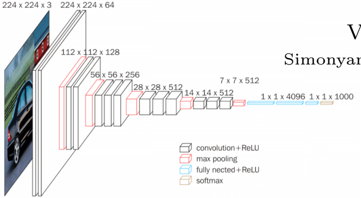
ResNet
He et al., 2016



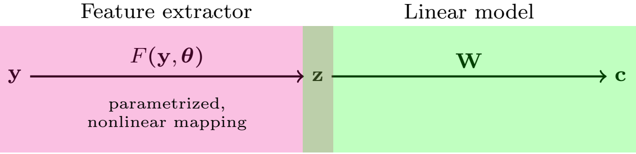
AlexNet
Krizhevsky et al., 2012



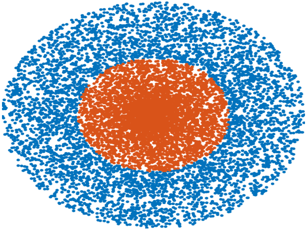
VGG
Simonyan et al., 2015



A Classy Way to DNNs



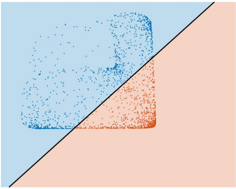
Inputs
$\{\mathbf{y}^{(1)},\mathbf{y}^{(2)},\dots\} \subset \mathbb{R}^2$



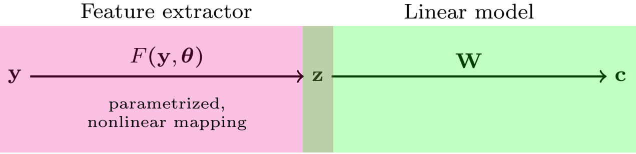
Outputs
$\{\mathbf{z}^{(1)},\mathbf{z}^{(2)},\dots\} \subset \mathbb{R}^2$

$$\begin{aligned}
\mathbf{u}_1 &= \sigma(\mathbf{K}_{\text{in}}\mathbf{y} + \mathbf{b}_{\text{in}}) \in \mathbb{R}^4\\
\mathbf{u}_2 &= \sigma(\mathbf{K}_1\mathbf{u}_1 + \mathbf{b}_1) \in \mathbb{R}^4\\
\mathbf{z} &= \sigma(\mathbf{K}_2\mathbf{u}_2 + \mathbf{b}_2) \in \mathbb{R}^2\\
x &= s(\mathbf{W}\mathbf{z}) \in \mathbb{R}
\end{aligned}$$

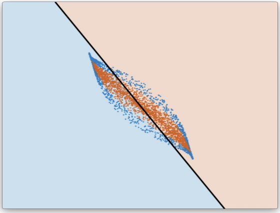
Targets
$\{c^{(1)},c^{(2)},\dots\} \subset \{0,1\}$



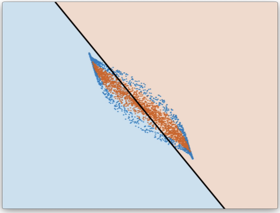
A Classy Way to DNNs



Network weights $\mathbf{W}$



Optimal $\mathbf{W}$



Variable Projection (VarPro)

The Supervised Learning Problem

Given training dataset $\mathcal{T}$, find the network weights $\mathbf{W}$ and $\boldsymbol{\theta}$ by solving

$$\min_{\mathbf{W},\boldsymbol{\theta}} \Phi(\mathbf{W},\boldsymbol{\theta}) \equiv \underbrace{\frac{1}{|\mathcal{T}|}\sum_{(\mathbf{y},\mathbf{c})\in\mathcal{T}} L(\mathbf{W}F(\mathbf{y},\boldsymbol{\theta}),\mathbf{c})}_{\text{loss}} + \underbrace{R(\boldsymbol{\theta}) + S(\mathbf{W})}_{\text{regularization}}$$

We consider loss functions and regularizers such that Φ is

→ smooth

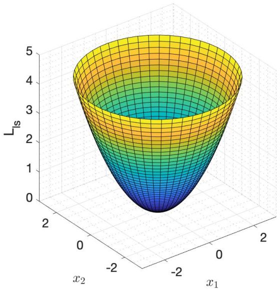
→ strictly convex in the first argument

Some Winning Loss Functions

Let $\mathbf{x} = \mathbf{W}F(\mathbf{y}, \boldsymbol{\theta})$ be the DNN approximation. Let t be the number of targets.

Least-Squares (Function Approximation)

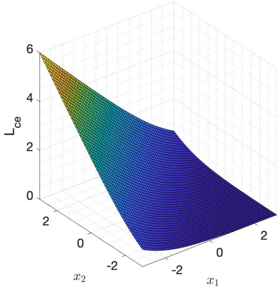
$$L_{\text{ls}}(\mathbf{x}, \mathbf{c}) = \tfrac{1}{2}\|\mathbf{x} - \mathbf{c}\|_2^2$$
$$L_{\text{ls}} : \mathbb{R}^t \times \mathbb{R}^t \to \mathbb{R}$$



$\mathbf{c} = (0, 0)^\top$

Cross-Entropy (Classification)

$$L_{\text{ce}}(\mathbf{x}, \mathbf{c}) = -\mathbf{c}^\top \log\left(\frac{\exp(\mathbf{x})}{\mathbf{e}^\top \exp(\mathbf{x})}\right)$$
$$L_{\text{ce}} : \mathbb{R}^t \times \Delta^t \to \mathbb{R}$$



$\mathbf{c} = (1, 0)^\top$

Variable Projection (VarPro)

The Supervised Learning Problem

Given training dataset $\mathcal{T}$, find the network weights $\mathbf{W}$ and $\boldsymbol{\theta}$ by solving

$$\min_{\mathbf{W},\boldsymbol{\theta}} \Phi(\mathbf{W},\boldsymbol{\theta}) \equiv \underbrace{\frac{1}{|\mathcal{T}|}\sum_{(\mathbf{y},\mathbf{c})\in\mathcal{T}} L(\mathbf{W}F(\mathbf{y},\boldsymbol{\theta}),\mathbf{c})}_{\text{loss}} + \underbrace{R(\boldsymbol{\theta}) + S(\mathbf{W})}_{\text{regularization}}$$

Main Idea: eliminate $\mathbf{W}$ to exploit coupling of $\boldsymbol{\theta}$ and $\mathbf{W}$ → accelerate convergence

The Reduced Optimization Problem

$$\underbrace{\min_{\boldsymbol{\theta}} \Phi_{\text{red}}(\boldsymbol{\theta}) \equiv \Phi(\mathbf{W}(\boldsymbol{\theta}),\boldsymbol{\theta})}_{\text{outer optimization}} \quad \text{s.t.} \quad \underbrace{\mathbf{W}(\boldsymbol{\theta}) = \arg\min_{\mathbf{W}} \Phi(\mathbf{W},\boldsymbol{\theta})}_{\text{inner optimization}}$$

Connection between VarPro and No VarPro:

$$\nabla_{\mathbf{W}}\Phi(\mathbf{W}(\boldsymbol{\theta}),\boldsymbol{\theta}) = \mathbf{0} \quad \Longrightarrow \quad \nabla_{\boldsymbol{\theta}}\Phi_{\text{red}}(\boldsymbol{\theta}) = \nabla_{\boldsymbol{\theta}}\Phi(\mathbf{W}(\boldsymbol{\theta}),\boldsymbol{\theta})$$

Golub and Pereyra 1973; Sjöberg and Viberg 1997

"Trustworthy" Optimization of $\mathbf{W}(\boldsymbol{\theta})$

Assume $\Phi(\mathbf{W}, \boldsymbol{\theta})$ is smooth and strictly convex in the first argument and solve

$$\mathbf{W}(\boldsymbol{\theta}) = \arg\min_{\mathbf{W}} \Phi(\mathbf{W}, \boldsymbol{\theta})$$

- ☺ Optimization problem is modest in size
- ☺ Independent of feature extractor
- ☹ No closed-form solution
- ☹ Solve efficiently to high accuracy

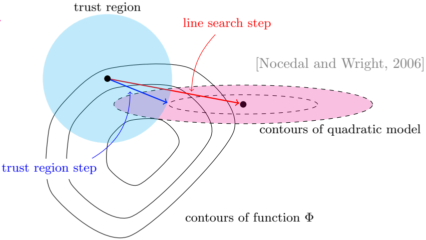
Newton-Krylov Trust Region Method: Update $\mathbf{W}_{\text{trial}} = \mathbf{W}^{(j)} + \delta\mathbf{W}$

$$\min_{\delta\mathbf{w}} \nabla_{\mathbf{w}}\Phi(\mathbf{W}^{(j)}, \boldsymbol{\theta})^\top \delta\mathbf{w} + \tfrac{1}{2}\delta\mathbf{w}^\top \nabla^2_{\mathbf{w}}\Phi(\mathbf{W}^{(j)}, \boldsymbol{\theta})\delta\mathbf{w}$$

subj. to $\quad \|\delta\mathbf{w}\| \leq \Delta^{(j)}$

Efficient: low-rank approx. to Hessian via Krylov method

Accurate: use double-precision



For separable, nonlinear least squares, see, e.g., Golub and Pereyra 1973.

Optimizing $\boldsymbol{\theta}$: Gauss-Newton-Krylov VarPro (GNvpro)

The Reduced Optimization Problem

$$\min_{\boldsymbol{\theta}} \Phi_{\text{red}}(\boldsymbol{\theta}) \equiv \underbrace{\frac{1}{|\mathcal{T}|}\sum_{(\mathbf{y},\mathbf{c})\in\mathcal{T}} L(\mathbf{W}(\boldsymbol{\theta})F(\mathbf{y},\boldsymbol{\theta}),\mathbf{c})}_{\text{loss}} + \underbrace{R(\boldsymbol{\theta}) + S(\mathbf{W}(\boldsymbol{\theta}))}_{\text{regularization}}$$

- ☺ Accelerates convergence to high accuracy
- ☹ Requires careful Jacobian implementation

Gauss-Newton-Krylov Trust Region Method: Update $\boldsymbol{\theta}_{\text{trial}} = \boldsymbol{\theta}^{(k)} + \delta\boldsymbol{\theta}$

$$\min_{\delta\boldsymbol{\theta}} \nabla_{\boldsymbol{\theta}}\Phi_{\text{red}}(\boldsymbol{\theta}^{(k)})^\top \delta\boldsymbol{\theta} + \tfrac{1}{2}\delta\boldsymbol{\theta}^\top \nabla_{\boldsymbol{\theta}}^2\Phi_{\text{red}}(\boldsymbol{\theta}^{(k)})\delta\boldsymbol{\theta} \quad \text{subj. to} \quad \|\delta\boldsymbol{\theta}\| \le \Delta^{(k)}$$

Approximate $\nabla_{\boldsymbol{\theta}}^2\Phi_{\text{red}}(\boldsymbol{\theta}^{(k)})$ via

$$\nabla_{\boldsymbol{\theta}}^2\Phi_{\text{red}}(\boldsymbol{\theta}^{(k)}) \approx J_{\boldsymbol{\theta}}(\mathbf{W}(\boldsymbol{\theta})F(\mathbf{y},\boldsymbol{\theta}))^\top \nabla^2 L J_{\boldsymbol{\theta}}(\mathbf{W}(\boldsymbol{\theta})F(\mathbf{y},\boldsymbol{\theta})) + \nabla^2 R$$

O'Leary and Rust 2013

The GNvpro Jacobian

Expand the Jacobian

$$J_{\boldsymbol{\theta}}(\mathbf{W}(\boldsymbol{\theta})F(\mathbf{y},\boldsymbol{\theta})) = \mathbf{W}(\boldsymbol{\theta})J_{\boldsymbol{\theta}}F(\mathbf{y},\boldsymbol{\theta}) + (F(\mathbf{y},\boldsymbol{\theta})^\top \otimes \mathbf{I})J_{\boldsymbol{\theta}}\mathbf{w}(\boldsymbol{\theta})$$

Solve for $J_{\boldsymbol{\theta}}\mathbf{w}(\boldsymbol{\theta})$ implicitly

$$\nabla^2_{\mathbf{w}}\Phi(\mathbf{W}(\boldsymbol{\theta}),\boldsymbol{\theta})J_{\boldsymbol{\theta}}\mathbf{w}(\boldsymbol{\theta}) = -J_{\boldsymbol{\theta}}\nabla_{\mathbf{w}}\Phi(\mathbf{W}(\boldsymbol{\theta}),\boldsymbol{\theta})$$

Least-Squares

$$\begin{aligned}\min_{\boldsymbol{\theta}} \Phi_{\text{ls,red}}(\boldsymbol{\theta}) &\equiv \frac{1}{2|\mathcal{T}|}\|\mathbf{W}(\boldsymbol{\theta})F(\mathbf{Y},\boldsymbol{\theta}) - \mathbf{C}\|_F^2 \\ &+ \frac{\alpha_1}{2}\|\boldsymbol{\theta}\|_2^2 + \frac{\alpha_2}{2}\|\mathbf{W}(\boldsymbol{\theta})\|_F^2\end{aligned}$$

- → Solve for $\mathbf{W}(\boldsymbol{\theta})$ using the SVD of $F(\mathbf{Y},\boldsymbol{\theta})$
- → Form $\nabla^2_{\mathbf{w}}\Phi_{\text{ls}}(\mathbf{W}(\boldsymbol{\theta}),\boldsymbol{\theta})$ re-using the SVD of $F(\mathbf{Y},\boldsymbol{\theta})$

Cross-Entropy

$$\begin{aligned}\min_{\boldsymbol{\theta}} \Phi_{\text{ce,red}}(\boldsymbol{\theta}) &\equiv \frac{1}{|\mathcal{T}|}\sum_{(\mathbf{y},\mathbf{c})\in\mathcal{T}} -\mathbf{c}^\top \log\left(\frac{\exp(\mathbf{W}(\boldsymbol{\theta})F(\mathbf{y},\boldsymbol{\theta}))}{\mathbf{e}^\top\exp(\mathbf{W}(\boldsymbol{\theta})F(\mathbf{y},\boldsymbol{\theta}))}\right) \\ &+ R(\boldsymbol{\theta}) + S(\mathbf{W}(\boldsymbol{\theta}))\end{aligned}$$

- → Solve for $\mathbf{W}(\boldsymbol{\theta})$ with Newton-Krylov Trust Region Method
- → Approx. $\nabla^2_{\mathbf{w}}\Phi_{\text{ce}}(\mathbf{W}(\boldsymbol{\theta}),\boldsymbol{\theta})$ using low-rank factorization from Krylov method

PDE Surrogate Modeling

Problem Setup:

$$\mathbf{c} = \mathcal{P}u \quad \text{subject to} \quad \mathcal{A}(u;\mathbf{y}) = 0$$

- → $\mathbf{y}$: parameters
- → u: solution
- → $\mathcal{P}$: discretize solution
- → $\mathbf{c}$: observables
- → $\mathcal{A}$: PDE operator

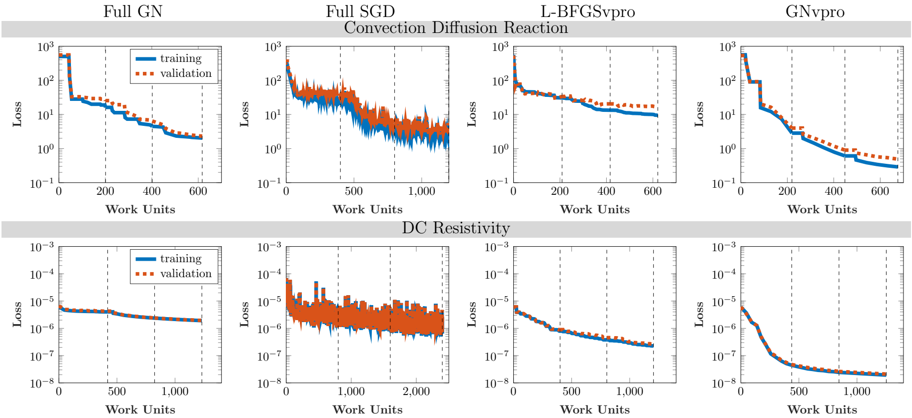
Goal: map parameters to observables and avoid expensive PDE solves

Loss Function: Least-Squares

PDEs:

- Convection Diffusion Reaction: $\mathbf{y} \in \mathbb{R}^{55}$, $\mathbf{c} \in \mathbb{R}^{72}$ (Grasso and Innocente 2018; Choquet and Comte 2017)
- Direct Current Resistivity: $\mathbf{y} \in \mathbb{R}^{3}$, $\mathbf{c} \in \mathbb{R}^{882}$ (Seidel and Lange 2007; Dey and Morrison 1979)

Surrogate Modeling Convergence



Work Units = number of forward and backward passes through network

SGD: 2 work units per epoch (1 forward pass, 1 backward pass)

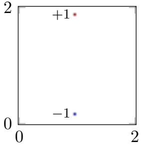
GNvpro: 2 works units + $2r$ work units for rank-r approx. to $\nabla^2_{\boldsymbol{\theta}} \Phi_{\text{red}}$ per iteration

DCR Visualization

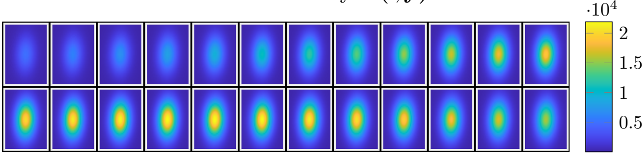
PDE: parameters $\mathbf{y} \in \mathbb{R}^3$ correspond to depth, volume, and rotation on x_1-x_2 plane

$$-\nabla \cdot (m(\cdot, \mathbf{y})\nabla u) = q \qquad \text{on } \Omega$$
$$\nabla u \cdot \mathbf{n} = 0 \qquad \text{on } \partial\Omega$$

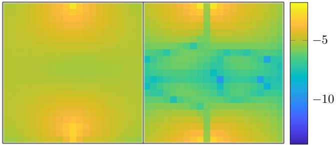
Source q



Conductivity $m(\cdot, \mathbf{y})$



Observations: $\mathbf{c} \in \mathbb{R}^{882}$ generated by measuring differences in u at surface in x_1 and x_2 directions



DCR Visualization

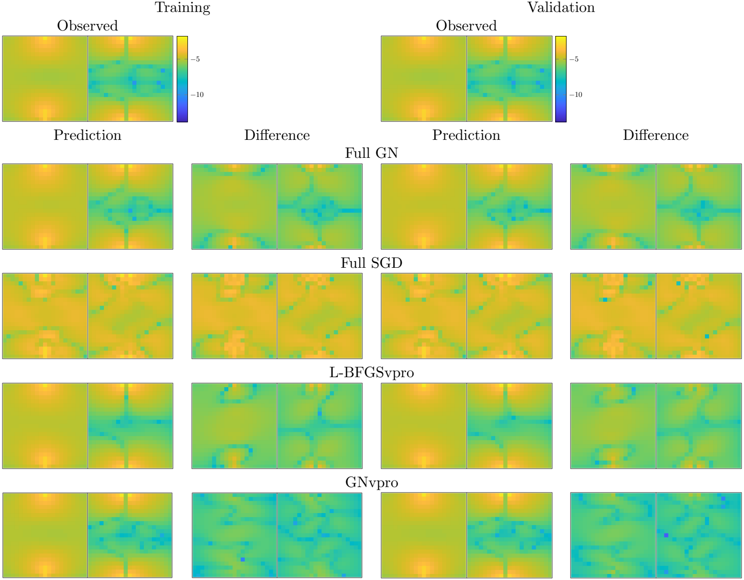
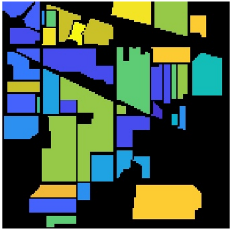


Image Segmentation

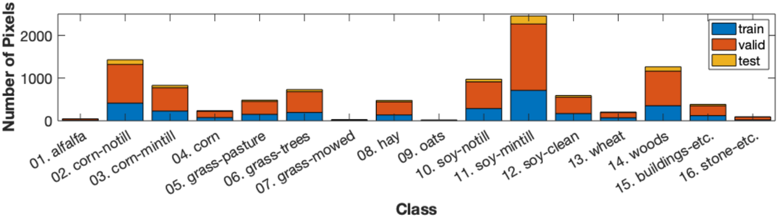


Goal: partition the pixels in an image into classes

Indian Pines Hyperspectral Dataset:

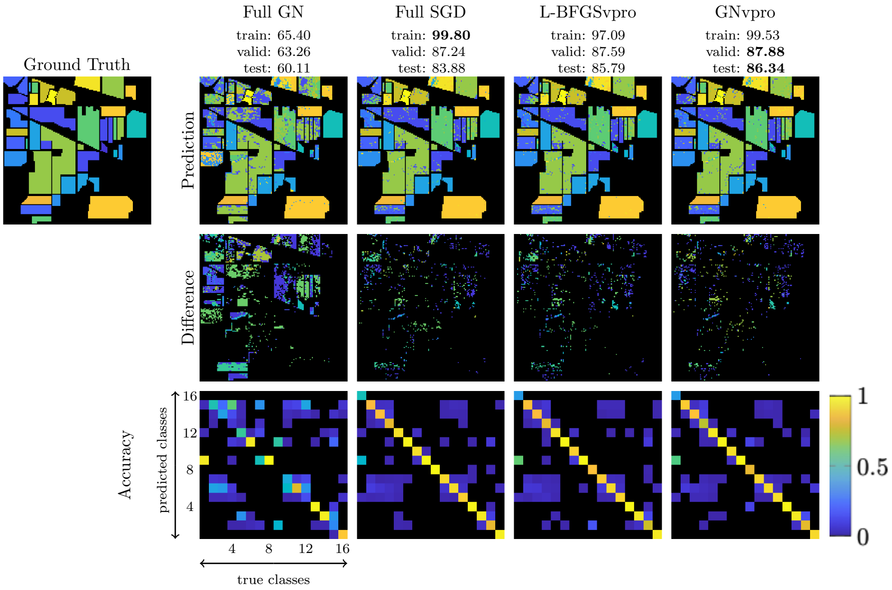
- → $\mathbf{y} \in \mathbb{R}^{220}$: pixels,
- → $\mathbf{c} \in \Delta^{16}$: one-hot labels

Loss Function: Cross-Entropy



Baumgardner, Biehl, and Landgrebe 2015

Image Segmentation Visualization



Summary

GNvpro: *"The \ of neural networks"*
(not quite yet)

GNvpro...

- ...accelerates training of DNNs to a high accuracy
- ...can be applied to non-quadratic objective functions
- ...is independent of nonlinear feature extractor

Future Work:

- Apply GNvpro to a wider range of learning problems (e.g., image classification)
- Implement GNvpro in other machine learning frameworks (e.g., Pytorch)
- Extend VarPro for stochastic approximation schemes (e.g., SGDvpro)

Thanks for listening!

Check out our paper on arXiv and our code Meganet.m on github.

E. Newman, L. Ruthotto, J. Hart, and B. van Bloemen Waanders. *Train Like a (Var)Pro: Efficient Training of Neural Networks with Variable Projection*, (2020) arXiv:2007.13171.

`https://github.com/XtractOpen/Meganet.m`

References I

Adebayo, Julius et al. (2020). *Sanity Checks for Saliency Maps.* arXiv: 1810.03292 [cs.CV].

Baumgardner, Marion F., Larry L. Biehl, and David A. Landgrebe (2015). *220 Band AVIRIS Hyperspectral Image Data Set: June 12, 1992 Indian Pine Test Site 3.* DOI: doi:/10.4231/R7RX991C. URL: https://purr.purdue.edu/publications/1947/1.

Bollapragada, Raghu, Richard H Byrd, and Jorge Nocedal (2018). "Exact and inexact subsampled Newton methods for optimization". In: *IMA Journal of Numerical Analysis* 39.2, pp. 545–578. ISSN: 1464-3642. DOI: 10.1093/imanum/dry009. URL: http://dx.doi.org/10.1093/imanum/dry009.

Choquet, EmmanuelleAugeraud-Vèronand Catherine and Èloïsese Comte (2017). "Optimal Control for a Groundwater Pollution Ruled by a ConvectionDiffusionReaction Problem". In: *Journal of Optimization Theory and Applications.*

Cybenko, G. (1989). "Approximations by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals, and Systems* 2, pp. 303–314.

Cyr, Eric C. et al. (2019). *Robust Training and Initialization of Deep Neural Networks: An Adaptive Basis Viewpoint.* arXiv: 1912.04862 [cs.LG].

Dey, A. and H.F. Morrison (1979). "Resistivity modeling for arbitrarily shaped three dimensional structures". In: *Geophysics* 44, pp. 753–780.

Golub, G.H. and V. Pereyra (1973). "The Differentiation of Pseudo-Inverses and Nonlinear Least Squares Problems whose Variables Separate". In: *SIAM Journal on Numerical Analysis* 10.2, pp. 413–432.

Goodfellow, Ian, Yoshua Bengio, and Aaron Courville (2016). *Deep Learning.* MIT Press.

References II

Grasso, Paolo and Mauro S. Innocente (2018). *Advances in Forest Fire Research: A two-dimensional reaction-advection-diffusion model of the spread of fire in wildlands*. Imprensa da Universidade de Coimbra.

Han, Jiequn, Arnulf Jentzen, and Weinan E (2018). "Solving high-dimensional partial differential equations using deep learning". In: *Proceedings of the National Academy of Sciences* 115.34, pp. 8505–8510. ISSN: 0027-8424. DOI: 10.1073/pnas.1718942115. eprint: https://www.pnas.org/content/115/34/8505.full.pdf. URL: https://www.pnas.org/content/115/34/8505.

Han, Jiequn, Arnulf Jentzen, and E Weinan (2018). "Solving high-dimensional partial differential equations using deep learning". In: *Proceedings of the National Academy of Sciences* 115.34, pp. 8505–8510.

He, Kaiming et al. (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778.

Hinton, G. et al. (2012). "Deep Neural Networks for Acoustic Modeling in Speech Recognition: The Shared Views of Four Research Groups". In: *IEEE Signal Processing Magazine* 29.6, pp. 82–97.

Keskar, Nitish Shirish et al. (2017). *On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima*. arXiv: 1609.04836 [cs.LG].

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*.

References III

Kleywegt, Anton J., Alexander Shapiro, and Tito Homem-de Mello (2002). "The Sample Average Approximation Method for Stochastic Discrete Optimization". In: *SIAM Journal on Optimization* 12.2, pp. 479–502. DOI: 10.1137/S1052623499363220. eprint: https://doi.org/10.1137/S1052623499363220. URL: https://doi.org/10.1137/S1052623499363220.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *NIPS*.

Li, Hao et al. (2018). "Visualizing the Loss Landscape of Neural Networks". In: *32nd Conference on Neural Information Processing Systems*.

Linderoth, Jeff, Alexander Shapiro, and Stephen Wright (2006). "The empirical behavior of sampling methods for stochastic programming". In: *Annals of Operations Research* 142.1, pp. 215–241. DOI: 10.1007/s10479-006-6169-8. URL: https://doi.org/10.1007/s10479-006-6169-8.

Montavon, Grgoire, Wojciech Samek, and Klaus-Robert Mller (2018). "Methods for interpreting and understanding deep neural networks". In: *Digital Signal Processing* 73, 115. ISSN: 1051-2004. DOI: 10.1016/j.dsp.2017.10.011. URL: http://dx.doi.org/10.1016/j.dsp.2017.10.011.

Moosavi-Dezfooli, Seyed-Mohsen, Alhussein Fawzi, and Pascal Frossard (2016). *DeepFool: a simple and accurate method to fool deep neural networks.* arXiv: 1511.04599 [cs.LG].

Nemirovski, A. et al. (2009). "Robust Stochastic Approximation Approach to Stochastic Programming". In: *SIAM Journal on Optimization* 19.4, pp. 1574–1609. DOI: 10.1137/070704277. eprint: https://doi.org/10.1137/070704277. URL: https://doi.org/10.1137/070704277.

Nielsen, Michael (2017). *Neural Networks and Deep Learning.* http://neuralnetworksanddeeplearning.com/.

References IV

O'Leary, Dianne P and Bert W Rust (2013). "Variable projection for nonlinear least squares problems". In: *Computational Optimization and Applications. An International Journal* 54.3, pp. 579–593.

O'Leary-Roseberry, Thomas, Nick Alger, and Omar Ghattas (2019). "Inexact Newton Methods for Stochastic Non-Convex Optimization with Applications to Neural Network Training". In: *arXiv*. arXiv: 1905.06738.

Papernot, Nicolas et al. (2016). "The Limitations of Deep Learning in Adversarial Settings". In: *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. DOI: 10.1109/eurosp.2016.36. URL: http://dx.doi.org/10.1109/EuroSP.2016.36.

Poggio, Tomaso et al. (2017). *Why and When Can Deep – but Not Shallow – Networks Avoid the Curse of Dimensionality: a Review.* arXiv: 1611.00740 [cs.LG].

Raghu, Maithra and Eric Schmidt (2020). *A Survey of Deep Learning for Scientific Discovery.* arXiv: 2003.11755 [cs.LG].

Raissi, M, P Perdikaris, and GE Karniadakis (2019). "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations". In: *Journal of Computational Physics, Elsevier* 378, pp. 686–707.

Ruthotto, Lars and Eldad Haber (2019). "Deep Neural Networks Motivated by Partial Differential Equations". In: *Journal of Mathematical Imaging and Vision* 62.3, 352364. ISSN: 1573-7683. DOI: 10.1007/s10851-019-00903-1. URL: http://dx.doi.org/10.1007/s10851-019-00903-1.

Samek, Wojciech et al. (2015). "Evaluating the visualization of what a Deep Neural Network has learned". In: *arXiv.org*. arXiv: 1509.06321v1 [cs.CV].

References V

Seidel, Knut and Gerhard Lange (2007). "Direct Current Resistivity Methods". In: *Environmental Geology: Handbook of Field Methods and Case Studies*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 205–237. ISBN: 978-3-540-74671-3. DOI: 10.1007/978-3-540-74671-3_8. URL: https://doi.org/10.1007/978-3-540-74671-3_8.

Sjöberg, J and Mats Viberg (1997). "Separable non-linear least-squares minimization-possible improvements for neural net fitting". In: *Neural Networks for Signal Processing VII. Proceedings of the 1997 IEEE Signal Processing Society Workshop.*

Song, William (2015). "End-to-End Deep Neural Network for Automatic Speech Recognition". In:

Tripathy, Rohit K and Ilias Bilionis (2018). "Deep UQ: Learning deep neural network surrogate models for high dimensional uncertainty quantification". In: *Journal of Computational Physics* 375, pp. 565–588.

Weinan, E and Bing Yu (2018). "The deep Ritz method: a deep learning-based numerical algorithm for solving variational problems". In: *Communications in Mathematics and Statistics* 6.1, pp. 1–12.

Yao, Zhewei et al. (2020). *ADAHESSIAN: An Adaptive Second Order Optimizer for Machine Learning.* arXiv: 2006.00719 [cs.LG].

Newton-Krylov Trust Region Method

Initialize $\mathbf{W}^{(0)} \equiv \mathbf{0}$ and $\Delta^{(0)}$.

$$\min_{\delta\mathbf{w}} \nabla_{\mathbf{w}}\Phi(\mathbf{W}^{(j)}, \boldsymbol{\theta})^\top \delta\mathbf{w} + \tfrac{1}{2}\delta\mathbf{w}^\top \nabla^2\Phi(\mathbf{W}^{(j)}, \boldsymbol{\theta})\delta\mathbf{w}$$
$$\text{subj. to } \|\delta\mathbf{w}\| \le \Delta^{(j)}$$

Project onto Krylov Subspace: $\mathcal{K}_r(\nabla^2\Phi(\mathbf{W}^{(j)}, \boldsymbol{\theta}), \nabla_{\mathbf{w}}\Phi(\mathbf{W}^{(j)}, \boldsymbol{\theta}))$

$$\mathbf{Q}_{r+1}\mathbf{H}_r = \nabla^2\Phi(\mathbf{W}^{(j)}, \boldsymbol{\theta})\mathbf{Q}_r$$

Solve for Approximate Step: $\delta\mathbf{w} = \mathbf{Q}_r\mathbf{z}^*(\lambda)$

$$\mathbf{z}^*(\lambda) = \arg\min_{\mathbf{z}\in\mathbb{R}^r} \tfrac{1}{2}\|\mathbf{H}_r\mathbf{z} - \beta\mathbf{e}_1\|^2 + \tfrac{\lambda}{2}\|\mathbf{z}\|^2$$

where $\beta = \|\nabla_{\mathbf{w}}\Phi(\mathbf{W}^{(j)}, \boldsymbol{\theta})\|$.

Approximate Inverse Hessian: $\nabla^2\Phi(\mathbf{W}(\boldsymbol{\theta}), \boldsymbol{\theta})^{-1} \approx \mathbf{Q}_r\mathbf{H}_r^\dagger\mathbf{Q}_{r+1}^\top$.

Numerical Experiments DNN Setup

Architecture: Neural ODE of the form

$$\mathbf{u}(t) = f(\mathbf{u}(t), \mathbf{K}(t), \mathbf{b}(t)) \quad \text{for} \quad t \in (0, T], \quad \mathbf{u}(0) = \sigma(\mathbf{K}_{\text{in}}\mathbf{y} + \mathbf{b}_{\text{in}}).$$

To promote stable dynamics, use an antisymmetric layer

$$f(\mathbf{u}, \mathbf{K}, \mathbf{b}) = \sigma((\mathbf{K} - \mathbf{K}^\top - \gamma\mathbf{I})\mathbf{u} + \mathbf{b})$$

with $\gamma = 10^{-4}$.

Training:

- Discretize the features and weights on nodes of an equidistant grid $[0, T]$ with d cells (d = depth)
- Optimize with a fourth-order Runge-Kutta scheme
- Multilevel: increase d and prolongate weights

Image Segmentation Convergence