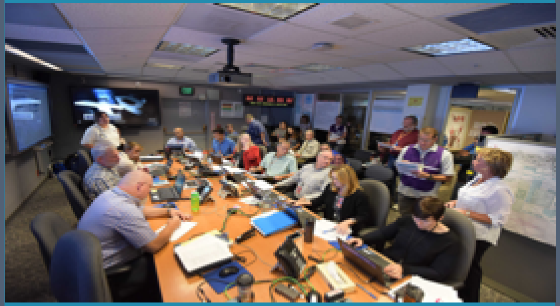
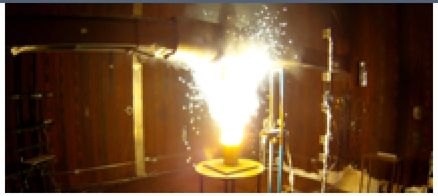
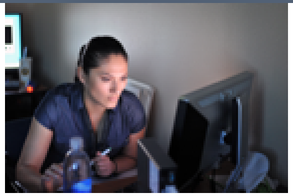
SAND2021-2260C

An Adaptive Basis Perspective to Improve Initialization and Accelerate Training of DNNs







Eric C. Cyr, Sandia National Laboratories

Authors

Mamikon Gulian, Ravi Patel, Mauro Perego, Nat Trask

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Overview

Take a viewpoint and see where it leads

We Adopt an Adaptive Basis Viewpoint of Neural Networks

This perspective leads to:

- A new initialization strategy based on stability analysis
- A hybrid least squares/gradient descent training algorithm for regression
- A hybrid Newton/gradient descent training algorithm for classification

More details can be found in:

- Cyr, Gulian, Patel, Perego, and Trask. "Robust training and initialization of deep neural networks: An adaptive basis viewpoint." In *Mathematical and Scientific Machine Learning*, pp. 512-536. PMLR, 2020.
- Patel, Trask, Gulian, and Cyr. "A block coordinate descent optimizer for classification problems exploiting convexity." *arXiv preprint arXiv:2006.10123* (2020). (Accepted to AAAI!)

Neural Networks

A neural network is a parameterized model:

Neural Network → $$\mathcal{N}\mathcal{N}(x;\Theta) \rightarrow y$$ ← Output

Input Parameters

It is composed of multiple layers*

Feature Vectors

$$u_1 = A_0 x + b_0,$$

$$u_{i+1} = g(u_i;\{A_i, b_i\}) \quad i = 1 \ldots L-1,$$

$$y = A_L u_L;$$

$$\Theta = \{A_i, b_i\}_{i=0}^{L-1} \cup \{A_L\}$$

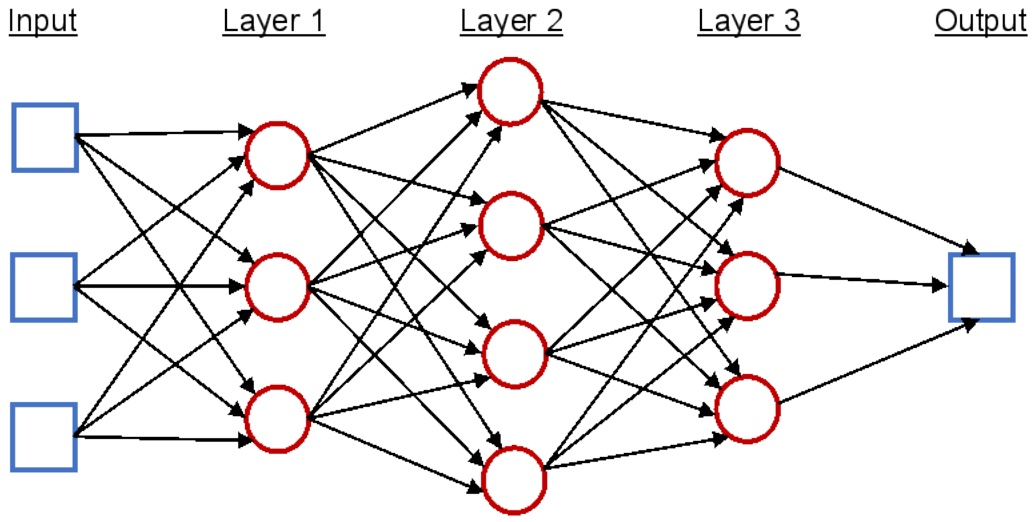
Neural Networks cont…

	Update Rule $g(u; A, b)$
Feed Forward	$u_{i+1} = \sigma(A_i u_i + b_i)$
ResNet	$u_{i+1} = u_i + \sigma(A_i u_i + b_i)$

Activation Function

Bias Vector

Weighting Matrix



Determining the Parameters

Neural network should map data according to the sampled **training set** :

$$\{(x_i, y_i)\}$$

Input → x_i, **Output** → y_i

Find Θ minimizing the loss in the model over the training set:

Parameters → Θ

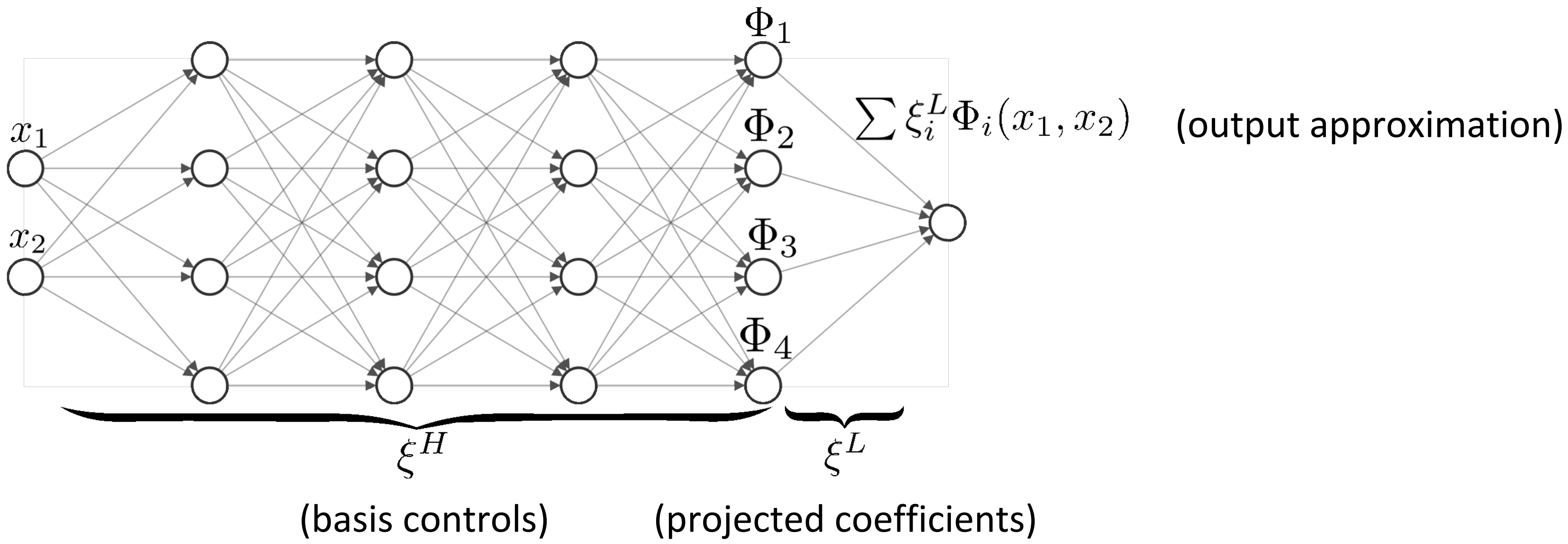
$$\min_{\Theta} \sum_{n=1}^{N} \text{Loss}\left(\mathcal{NN}(x_n;\Theta), y_n\right)$$

Loss function is model/data difference:

- $\text{Loss}(y^{model}, y^{data}) = \|y^{model} - y^{data}\|^2$
- $\text{Loss}(\vec{y}^{\,model}, \vec{y}^{\,data}) = \sum_{c=1}^{N_c} y_c^{data} \log\left(y_c^{model}\right)$

An Adaptive Basis Perspective

View a neural network as producing a “basis” followed by a projection



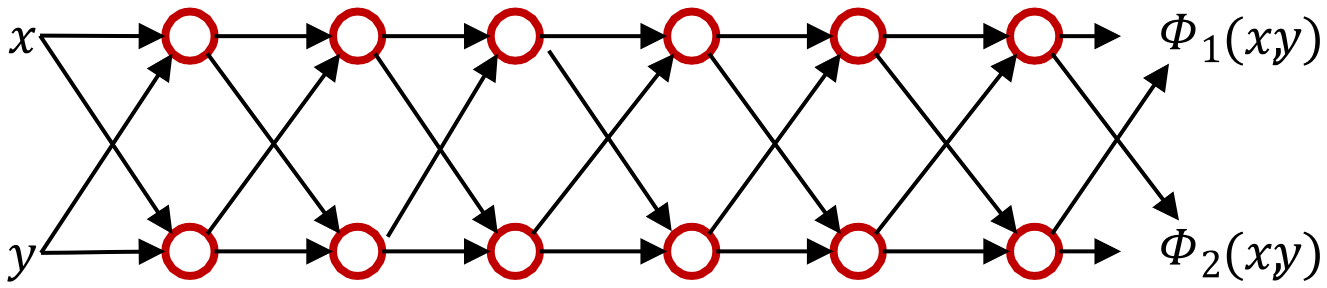
Taking this perspective we will explore:

1. Parameter initialization
2. Training algorithms
   a. Regression
   b. Classification

Parameter Initialization: An Experiment

Experimental setup:

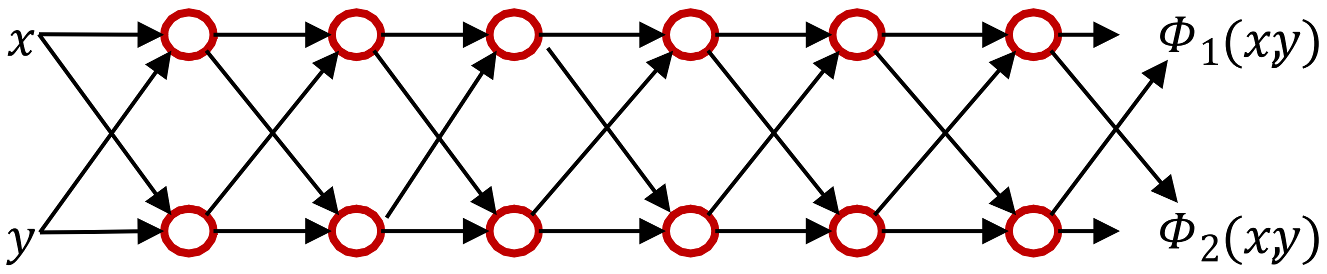
1. Initialize weights and biases
2. Propagate $[0,1]^2$ through the neural network
   - ReLU activations (no batch norm)
   - Feed-forward and ResNet



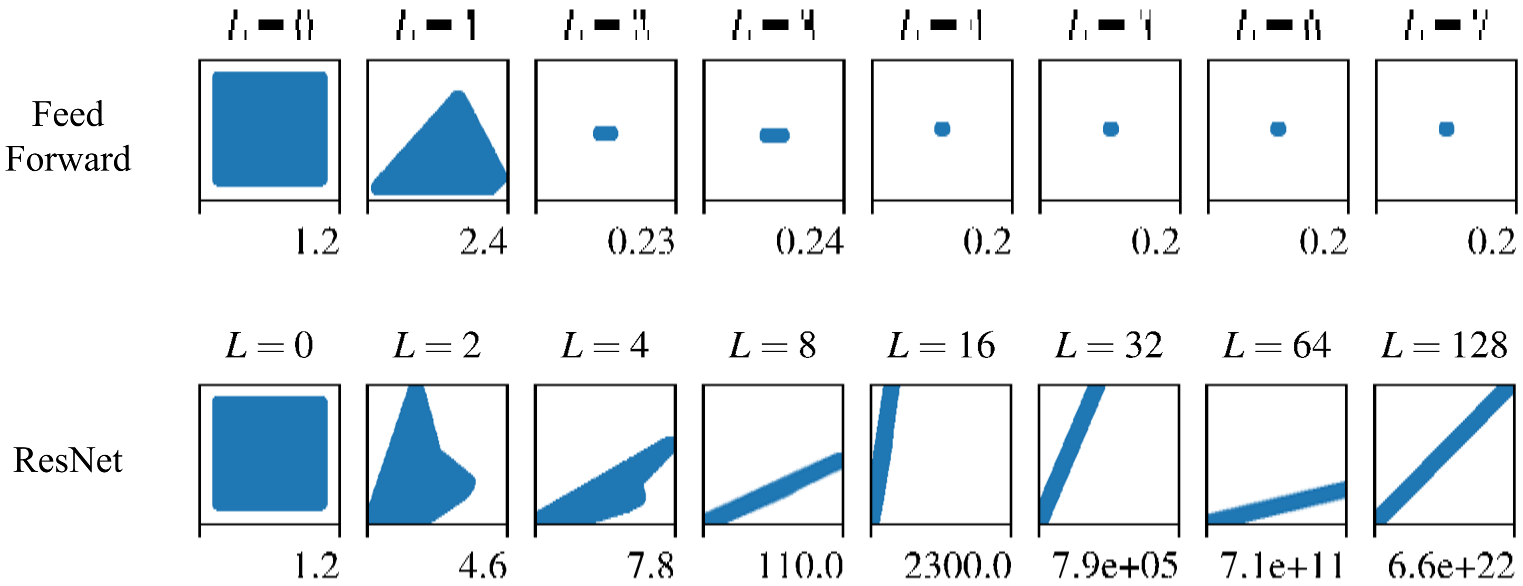
3. What does the basis look like?
   - Is it a good basis?
   - Is this a good place to start training?

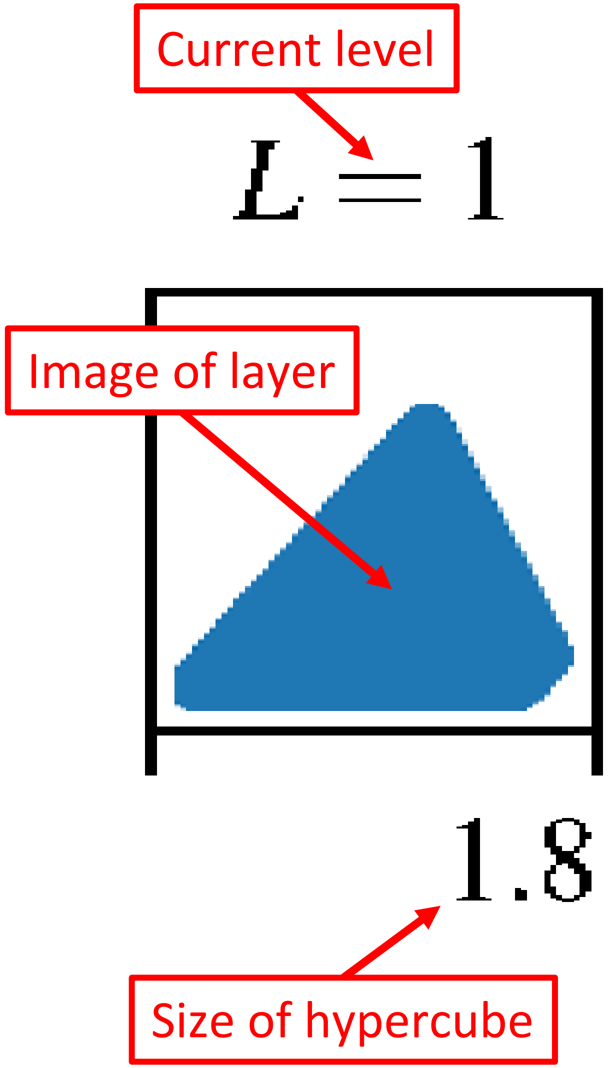
He Initialization

"He*" is a standard technique: for ReLU's with batch norm



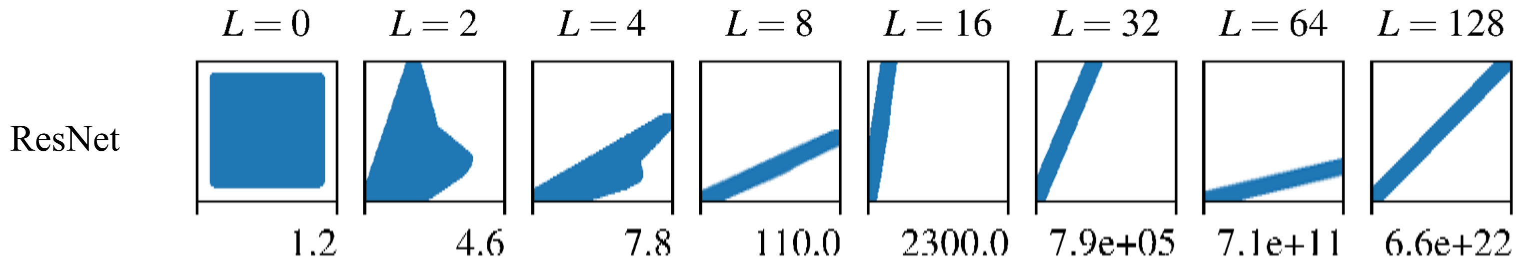
Plot the image of $[0,1]^2$ through all layers





*He, K., Zhang, X., Ren, S., & Sun, J. (2015). In *Proceedings of the IEEE international conference on computer vision* (pp. 1026-1034).

What is happening? ResNets



Each layer update is: $$x_{l+1} = x_l + \sigma(A_l x_l)$$

$$x_{l+1} \sim (I+\lambda)x_l \sim (I+\lambda)^{l+1} x_0$$

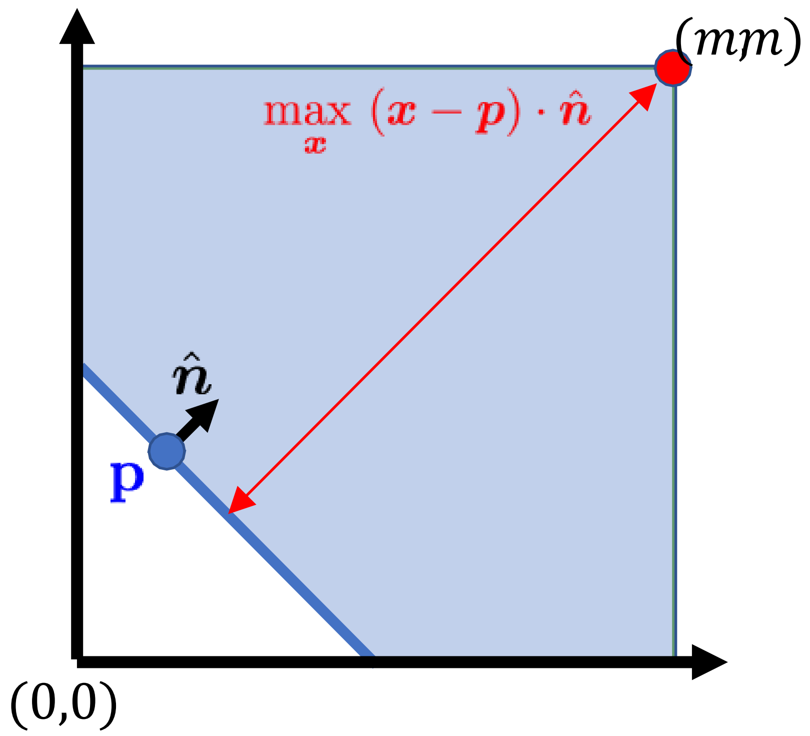
λ is the spectral radius of A_l

Related to exploding/vanishing gradients, if initialized weights are too large inference with DNN will be unstable

Our Approach: “Box” Initialization

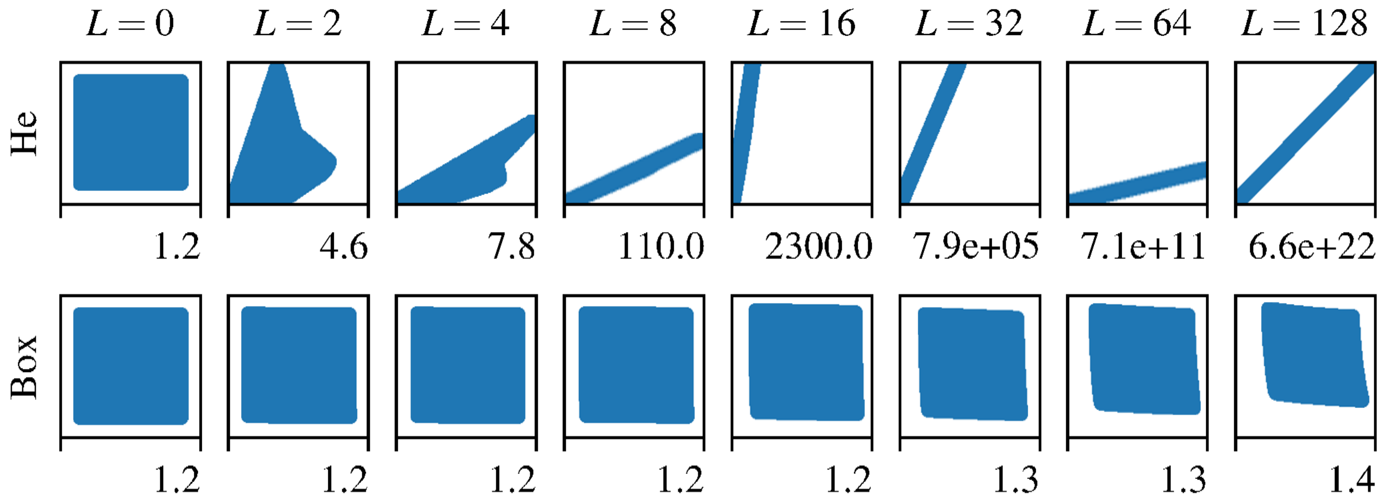
Goals:

- Remain Bounded
- Don’t Collapse: Requires growth of cell size
- Keep cut-plane is in cell at each layer



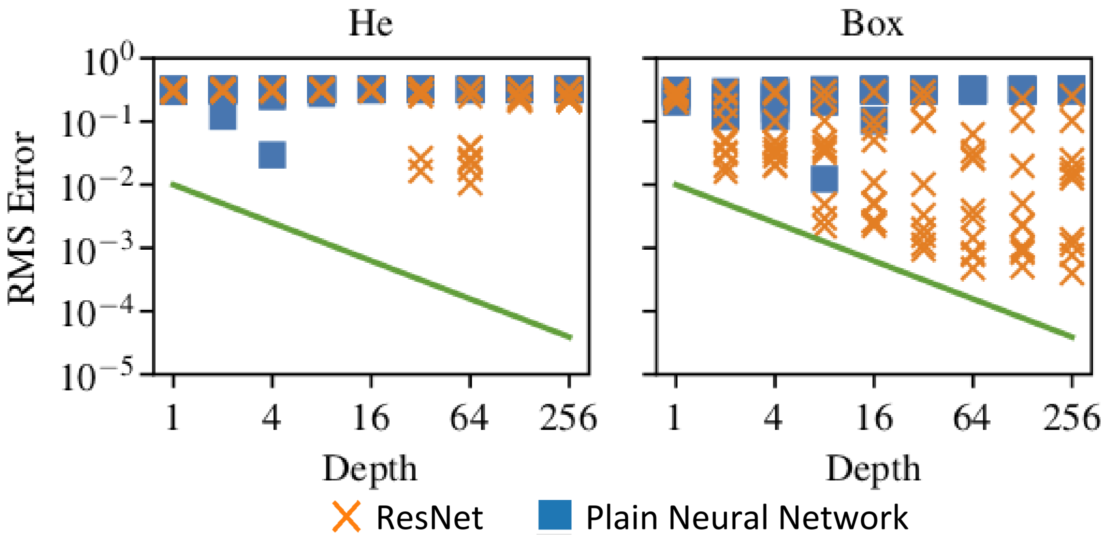
Initialize weights so that $\lambda \leq L^{-1}$ gives:

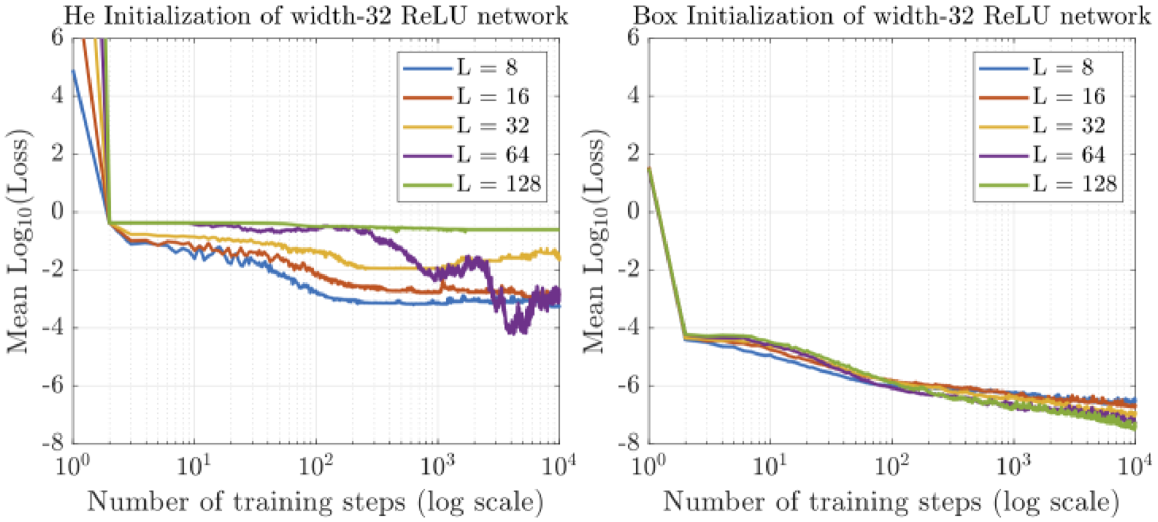
$$x_L \sim (1 + L^{-1})^L x_0 \leq e^1 x_0$$



- “Box” prevents, collapse and exponential growth
- $[0,1]^2$ cube maps to nearly a cube after 128 layers

Experiments: Initialization with Box vs. He





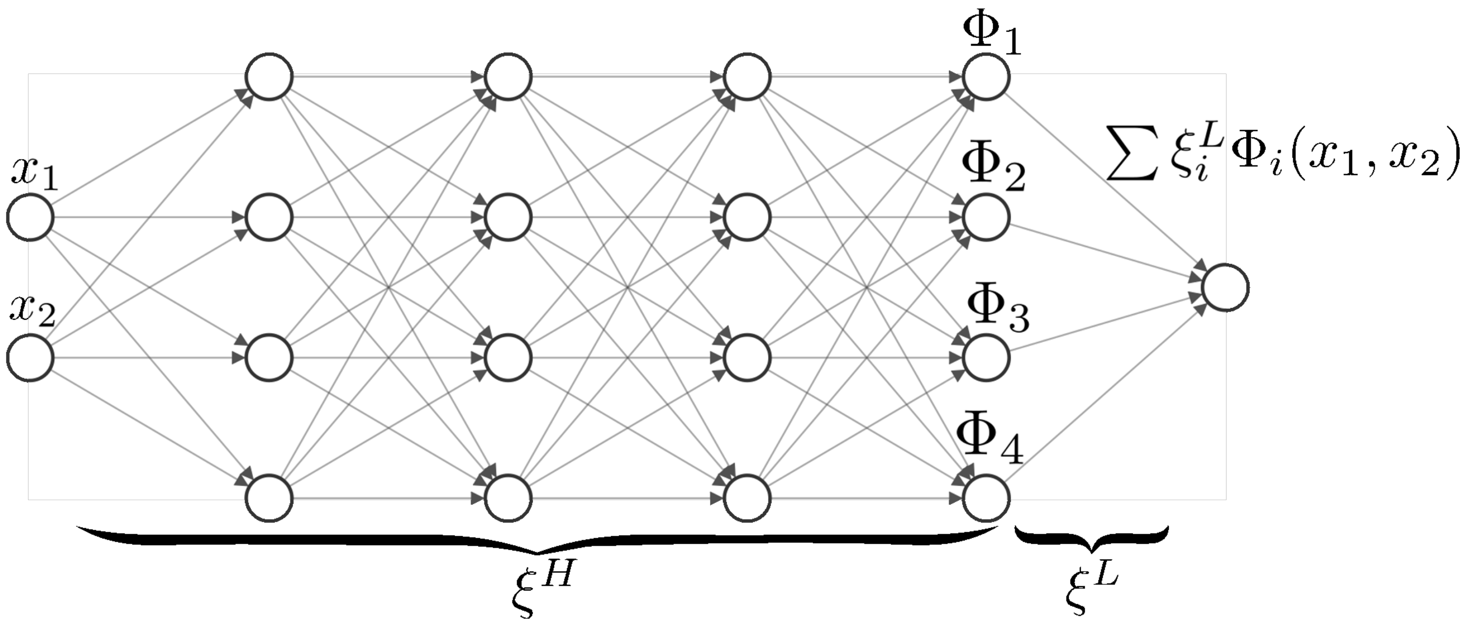
Approximating a discontinuous function composed of two polynomials (network width is 2)

- Only Box with ResNet (orange crosses) works well
- Box does better over multiple samples, more robust achieving some convergence on average

Approximating $\sin(2\pi x)$

- Both He and Box work okay for small numbers of layers
- He suffers for large numbers of layers
- Box leads to smaller errors, with better performance for large numbers of layers

Adaptive Basis Approaches to Training



Adaptive Basis Perspective Suggests a Training Approach

- Split Neural Network Parameters
  - Nonlinear: ξ_H
  - Linear: ξ_L
- Generalized Sketch of Training Approach
  1. Update ξ_H with gradient descent: “Refine” basis
  2. Solve optimization problem for ξ_L: Project onto basis

Regression with Hybrid LS/GD

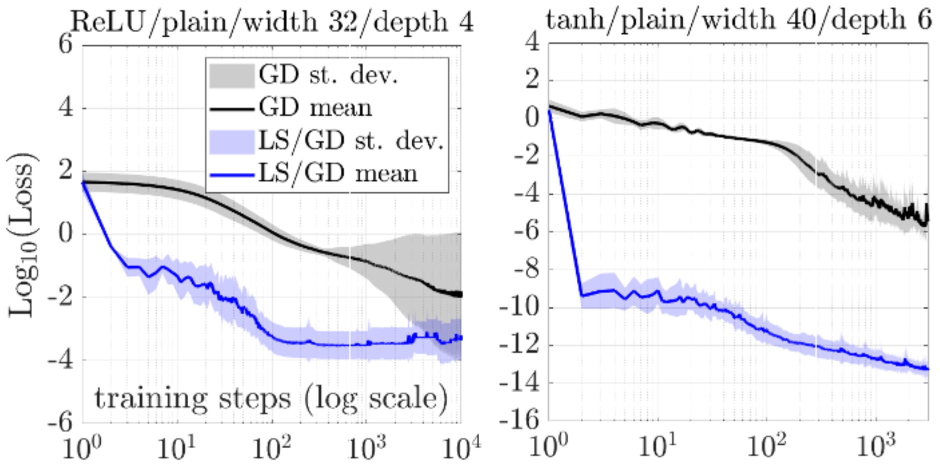
Applying Approach to Regression problems: a Least Squares/Gradient Descent algorithms

$$\underset{\boldsymbol{\xi}^{\mathrm{L}},\,\boldsymbol{\xi}^{\mathrm{H}}}{\operatorname{argmin}} \left\| u - \sum_i \xi_i^{\mathrm{L}} \Phi_i(\boldsymbol{x}, \boldsymbol{\xi}^{\mathrm{H}}) \right\|$$

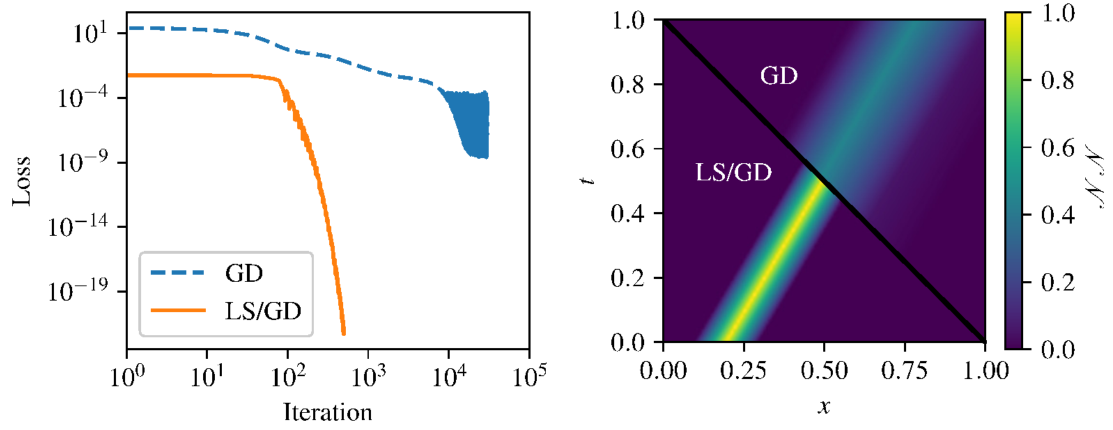
function LSGD($\boldsymbol{\xi}_0^H$)
  $\boldsymbol{\xi}^H = \boldsymbol{\xi}_0^H$
  $\boldsymbol{\xi}^L = LS(\boldsymbol{\xi}^H)$
  for $i = 1 \ldots$ **do**
    $\boldsymbol{\xi}^H = GD(\boldsymbol{\xi})$
    $\boldsymbol{\xi}^L = LS(\boldsymbol{\xi}^H)$
  end for
end function

Examples

Regression for $\sin(2\pi x)$



PINNs for Transport

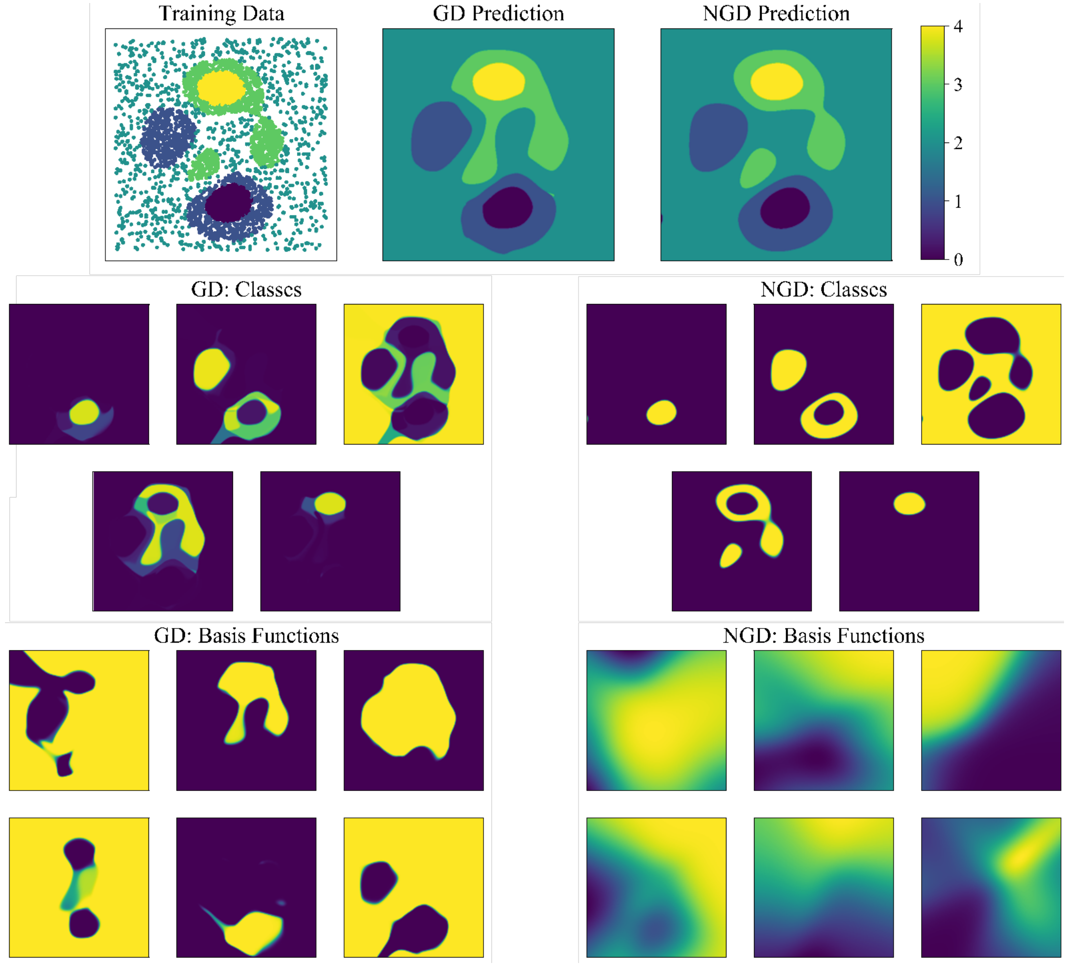


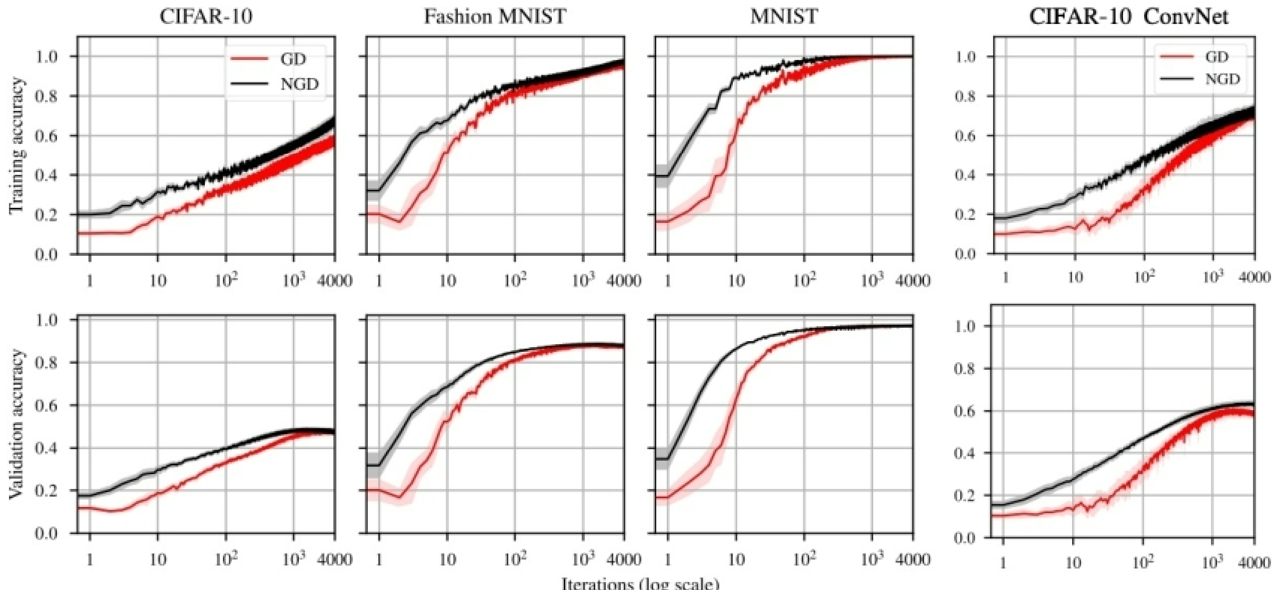
Classification with Newton/GD

Applying Approach to Classification problems:
a Newton/Gradient Descent algorithms

$$\underset{\xi^{\mathrm{L}},\,\xi^{\mathrm{H}}}{\operatorname{argmin}} \sum_{i} \sum_{c=1}^{N_c} y_{i,c} \log\left(\bar{y}_{i,c}\right)$$

where $$\bar{y}_{i,c} \propto \exp\left(\underbrace{\sum_j \xi^L_{c,j} \Phi_j(x_i, \xi^H)}_{\text{Level set approx. for each class}}\right)$$

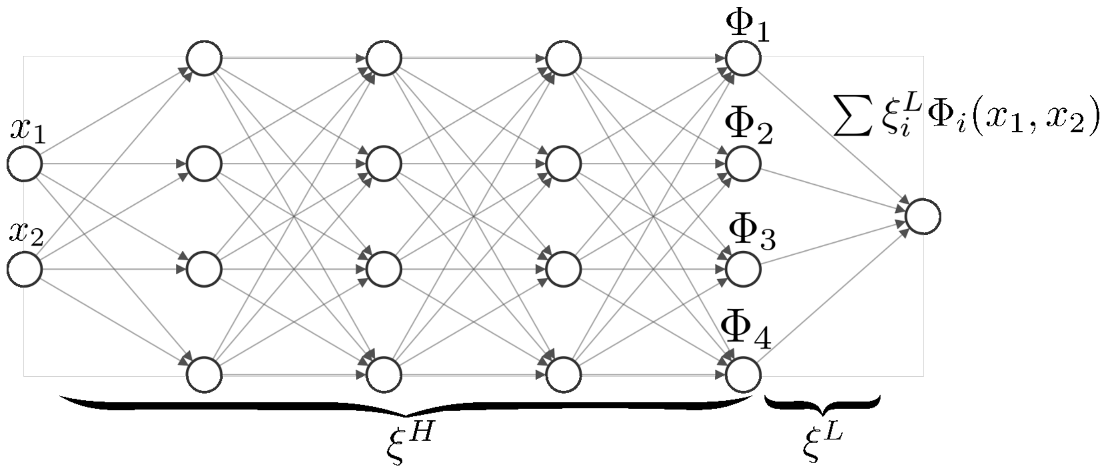




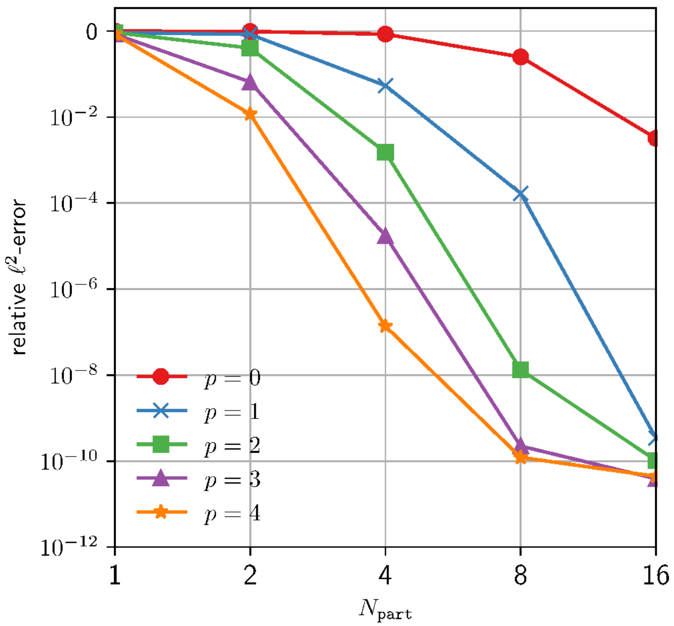
Final Thoughts

The adaptive basis perspective lead to ideas that improved neural network training

- "Box" initialization was developed by understanding how to generate a good basis
- LSGD was developed by splitting coefficients from basis parameters
  - ➢ Taking advantage of the convexity the regression loss
- NGD was developed by splitting coefficients from basis parameters
  - ➢ Taking advantage of the convexity of the classification loss



POU Net convergence



Next Step: Partition of Unity Neural Networks

- https://arxiv.org/abs/2101.11256, accepted to AAAI