# Deep Learning Tutorial

Michael Lutter

Darmstadt, 24. August 2017

# Agenda

- Neural Network History
- Neural Network Basics
  - Computational Graph
  - Neurons
  - Parameter Learning
- Optimization for Deep Neural Networks
  - Non-Convex Cost Functions
  - Weight Initialization
  - Stochastic Gradient Descent (SGD)
  - Plugins for SGD

# Agenda

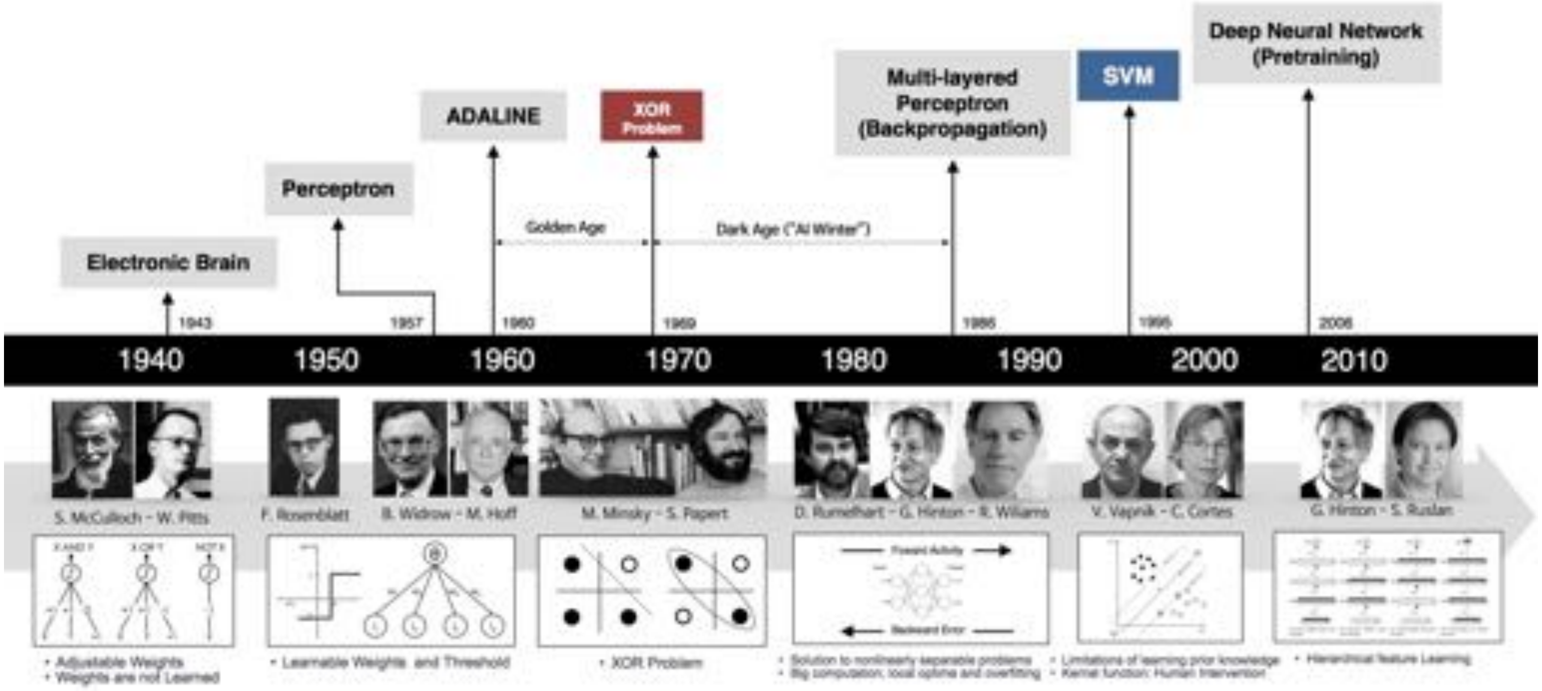
- **Neural Network History**
- Neural Network Basics
  - Computational Graph
  - Neurons
  - Parameter Learning
- Optimization for Deep Neural Networks
  - Non-Convex Cost Functions
  - Weight Initialization
  - Stochastic Gradient Descent (SGD)
  - Plugins for SGD

# The Human Brain

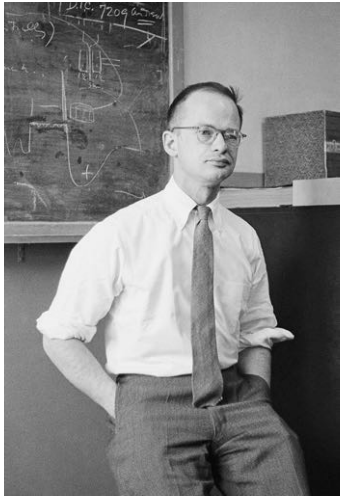
| | Brain | Deep Learning |
|---|---|---|
| Mass: | 1.3 Kg | - |
| Power: | 20 W | n x 250 W |
| Frequency: | 10 Hz | 1531 MHz |
| Neurons: | $10^{11}$ | $10^{7}$ |
| Synapses[1]: | $10^{4}$ | $10^{4}$ |

[1] Synapses per Neuron

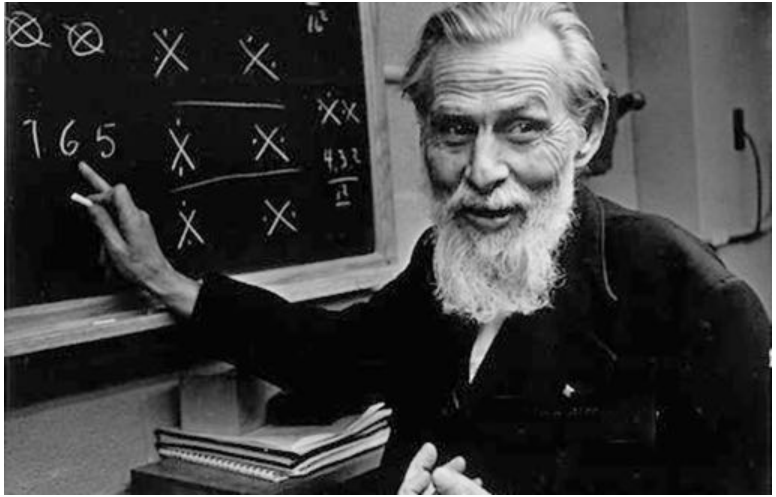
# History of Neural Networks



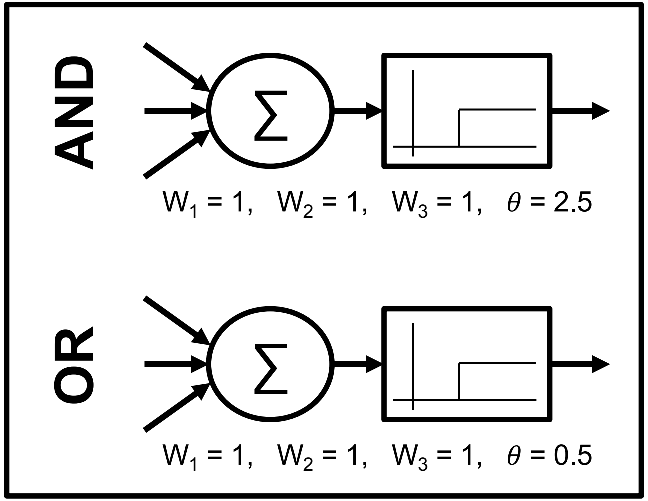
# 1943 – The Electronic Brain



Walter Pitts

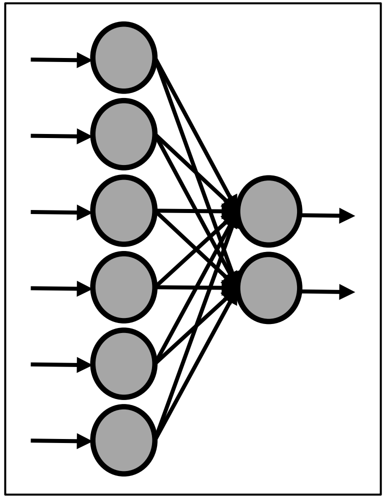


Warren McCulloch



# 1957 - The Perceptron

Frank Rosenblatt

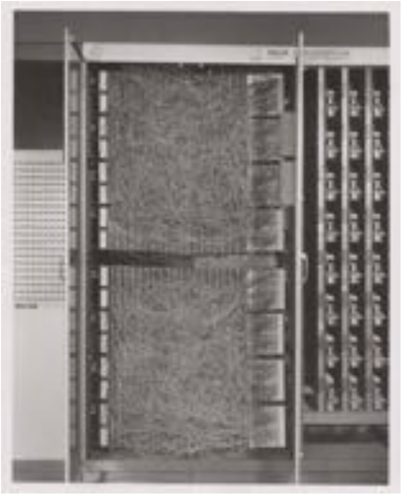


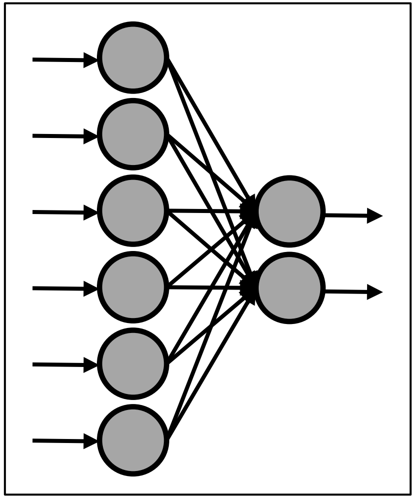
“The embryo of an electronic computer that [the Navy] expects will be able to walk, talk, see, write, reproduce itself and be conscious of its existence.”

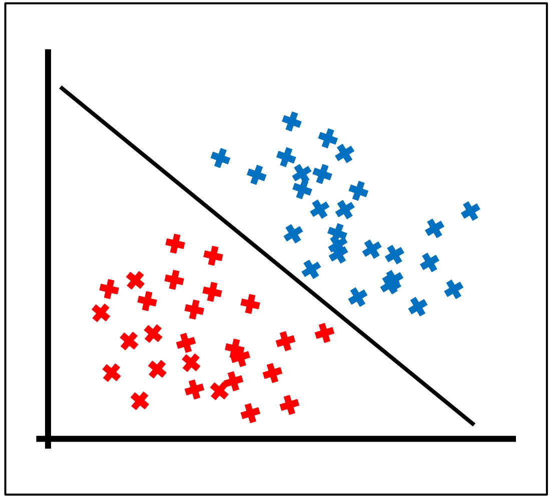
1958 New York Times

# 1957 - The Perceptron

Frank Rosenblatt

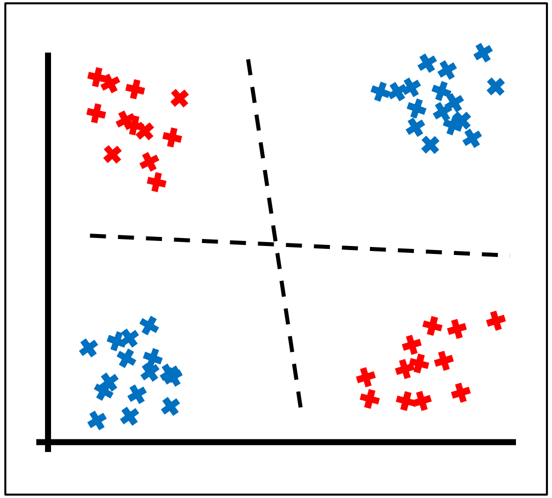




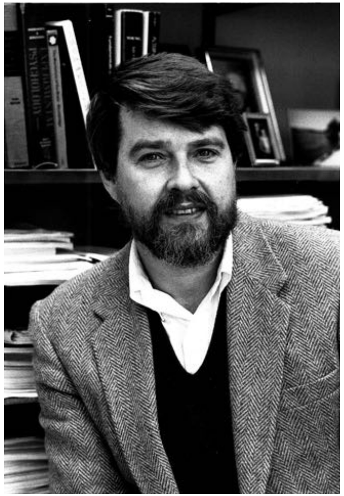


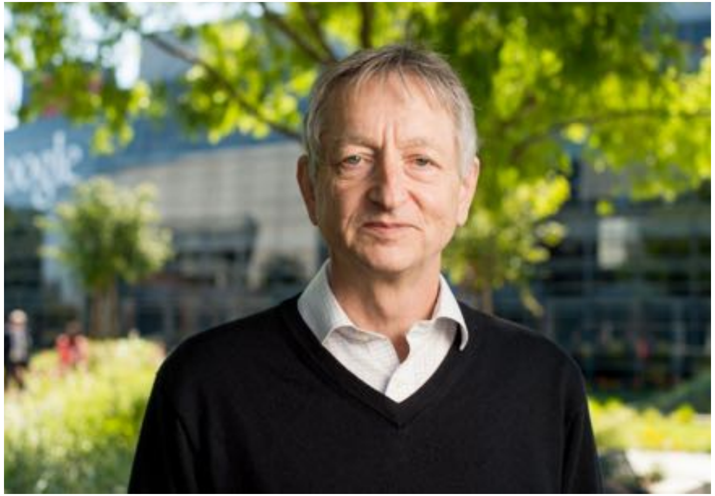
# 1969 – The 1st Neural Network Winter

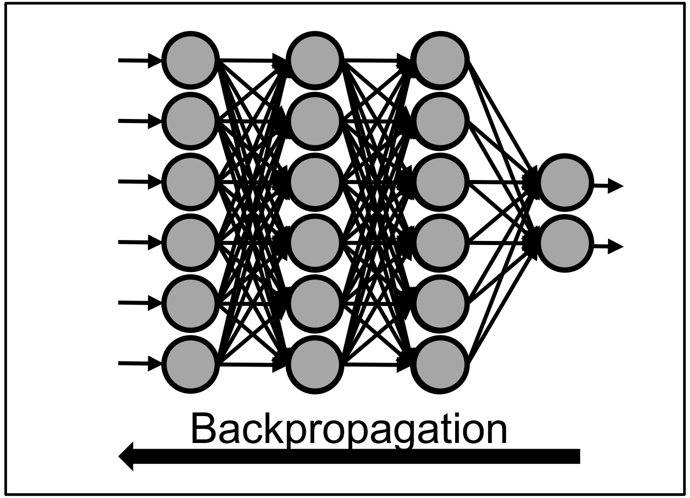
Marvin Minsky Seymour Papert



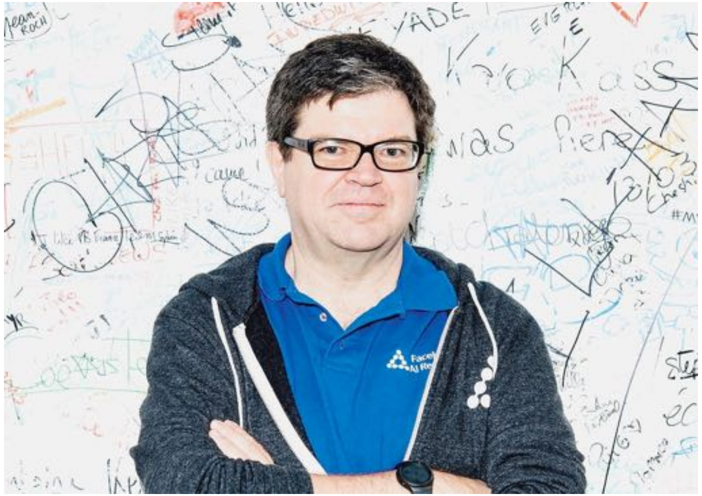
# 1987 – Emergence of Backpropagation


David Rumelhart


Geoffrey Hinton



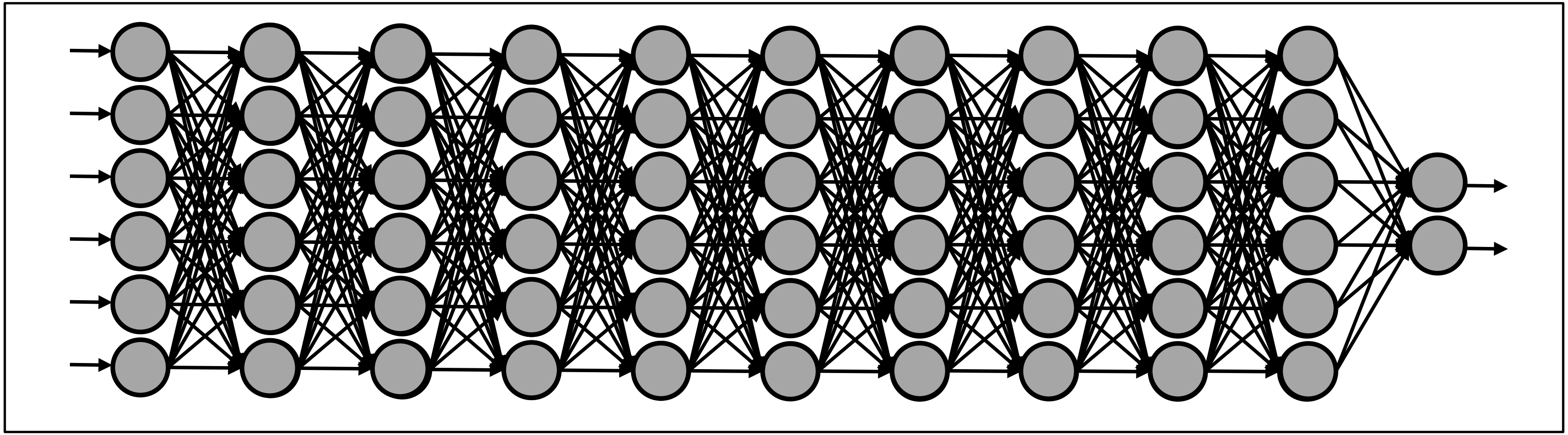
# 2000 – The 2nd Neural Network Winter



Yann LeCun

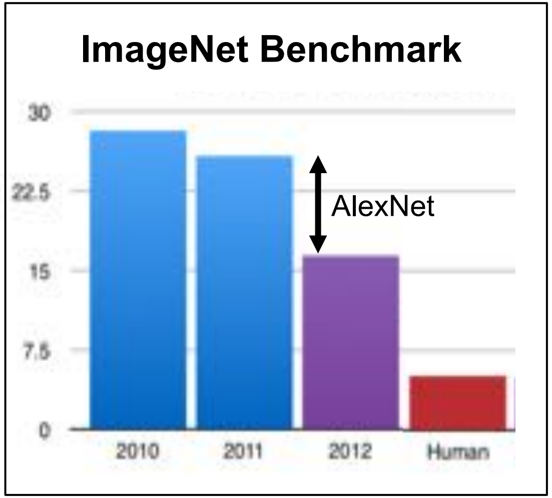
“The view was that he [Yann LeCun] was carrying on doing things that had been promising in the 80’s but he should have got over it” Geoffrey Hinton

# 2006 – Deep Learning



# 2012 – The Breakthrough

Ilya Sutskever /Alex Krizhevsky / Geoffrey Hinton

# Computer Vision - Segmentation

Kaiming He et. al. “Mask R-CNN”, 2017

# Status Quo – Image Classification

## MNIST

| | |
|---|---|
| 10 | classes |
| 70k | Images |
| 0.20 % | Human Performance |
| **0.21 %** | **Best Performance** |

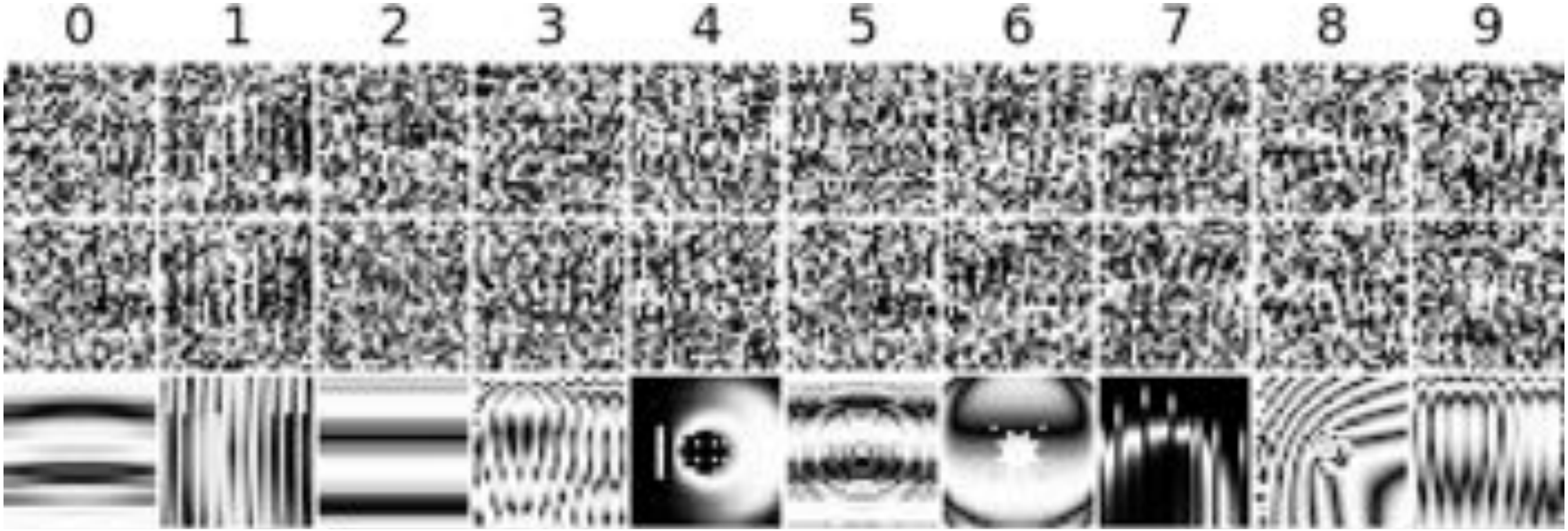
## CIFAR 10

| | |
|---|---|
| 10 | classes |
| 60k | Images |
| 6.00 % | Human Performance |
| **4.41 %** | **Best Performance** |

## Imagenet

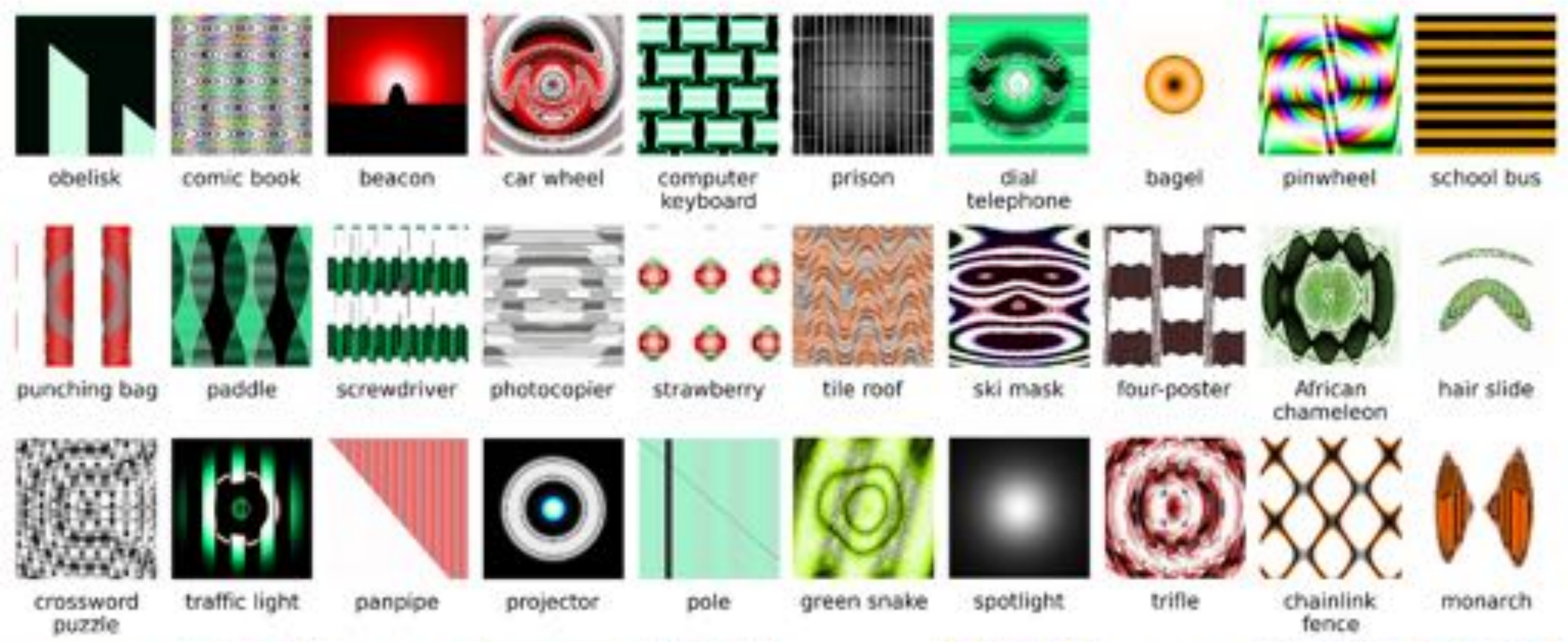
| | |
|---|---|
| 1000 | classes |
| 1200k | Images |
| 5.10 % | Human Performance |
| **4.80 %** | **Best Performance** |

# Status Quo – Image Classification




Anh Nguyen et.al., “Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images”, 2015

# Status Quo – Image Classification

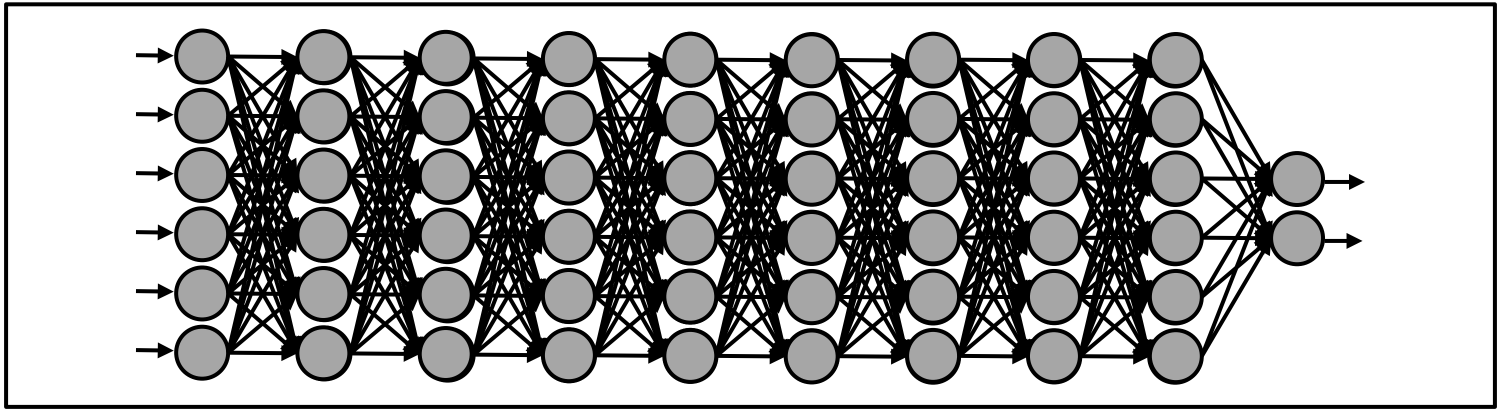



Anh Nguyen et.al., “Deep Neural Networks are Easily Fooled: High Confidence Predictions for Unrecognizable Images”, 2015

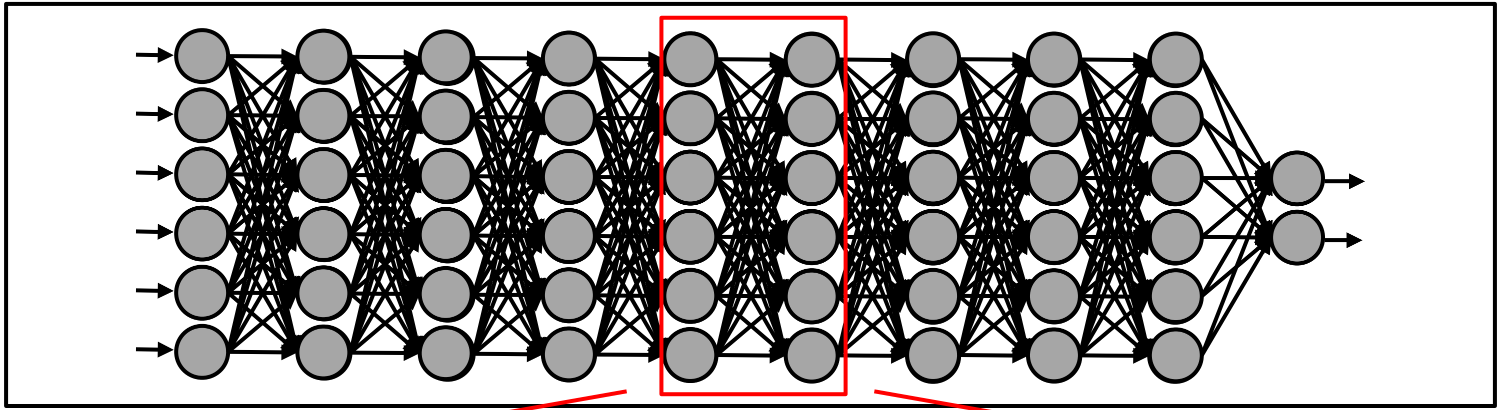
# Agenda

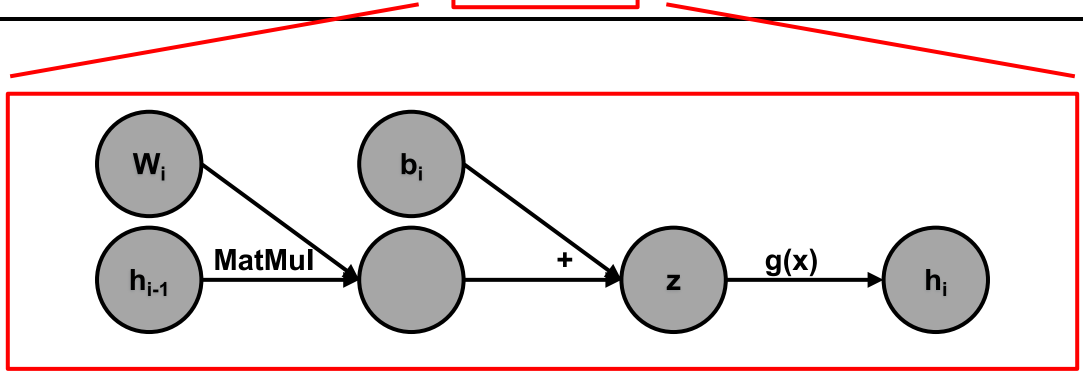
- Neural Network History
- **Neural Network Basics**
  - Computational Graph
  - Neurons
  - Parameter Learning
- Optimization for Deep Neural Networks
  - Non-Convex Cost Functions
  - Weight Initialization
  - Stochastic Gradient Descent (SGD)
  - Plugins for SGD

# Computational Graph



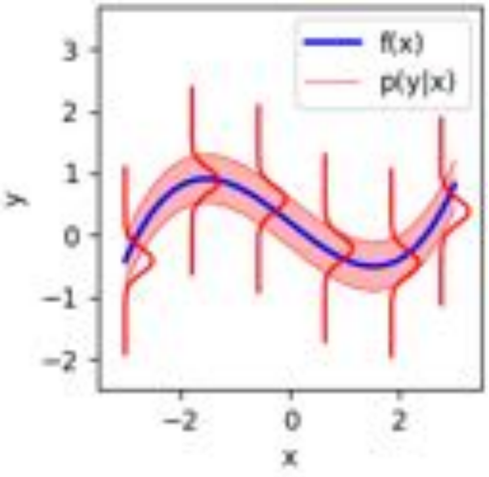
# Computational Graphs





# Output Neuron Types

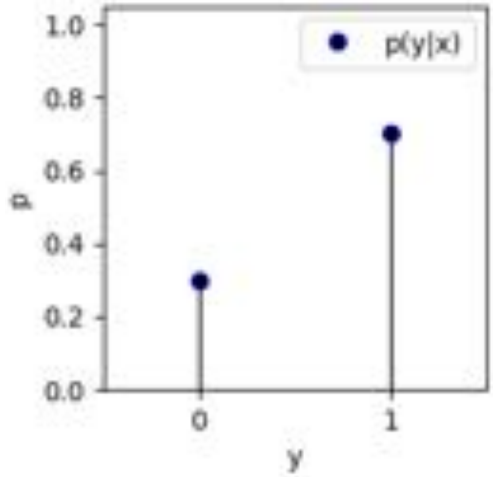
## Linear Neuron




$$g(\boldsymbol{z}_i) = \boldsymbol{z}_i$$

$$p(\boldsymbol{y} \mid \boldsymbol{z}) = N(\boldsymbol{y} - \boldsymbol{z}, \boldsymbol{I})$$

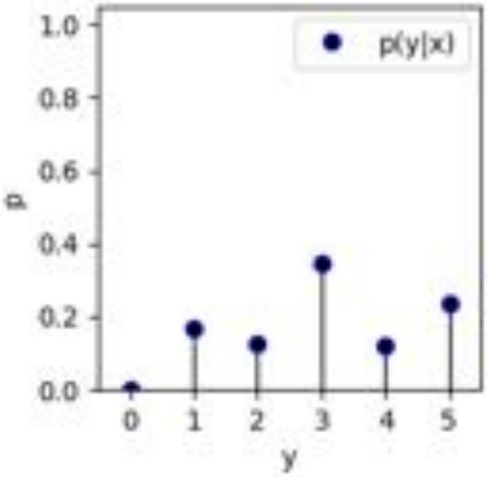
## Sigmoid Neuron




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

$$p(y \mid z) = \sigma((2y - 1)z)$$

## Softmax Neuron

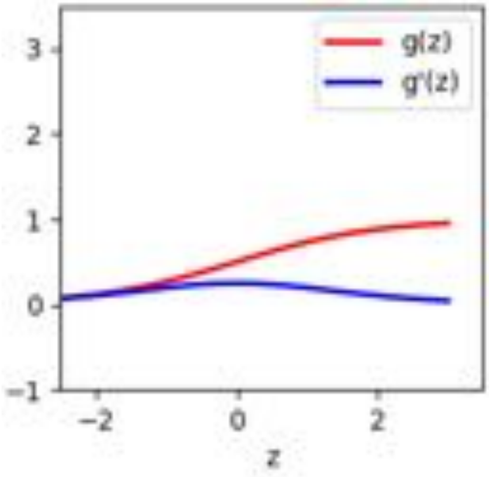



$$g(\boldsymbol{z}_i) = \frac{\exp \boldsymbol{z}_i}{\sum_j \exp z_j}$$

$$p(y = i \mid \boldsymbol{z}) = g(\boldsymbol{z}_i)$$

# Hidden Neuron Types

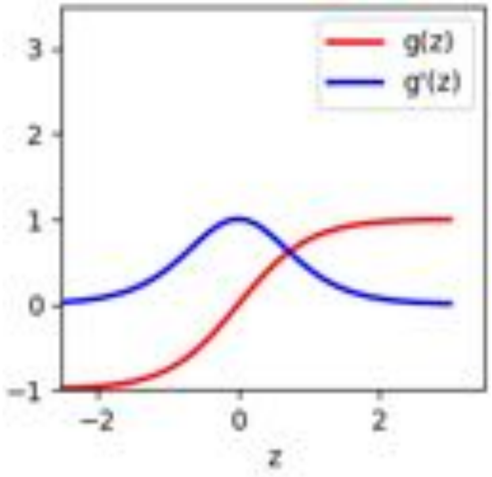
**Sigmoid Neuron**




$$g(\boldsymbol{z}_i) = \sigma(\boldsymbol{z}_i) = \frac{1}{1+e^{-z_i}}$$

$$g'(\boldsymbol{z}_i) = \sigma(\boldsymbol{z}_i)\left(1 - \sigma(\boldsymbol{z}_i)\right)$$

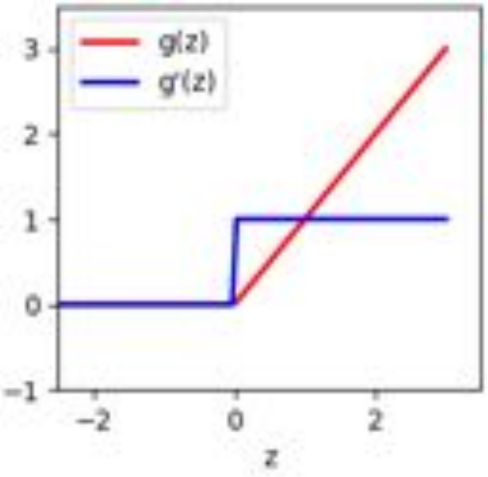
**Tanh Neuron**




$$g(\boldsymbol{z}_i) = \tanh(\boldsymbol{z}_i)$$

$$g'(\boldsymbol{z}_i) = 1 - \tanh(\boldsymbol{z}_i)^2$$

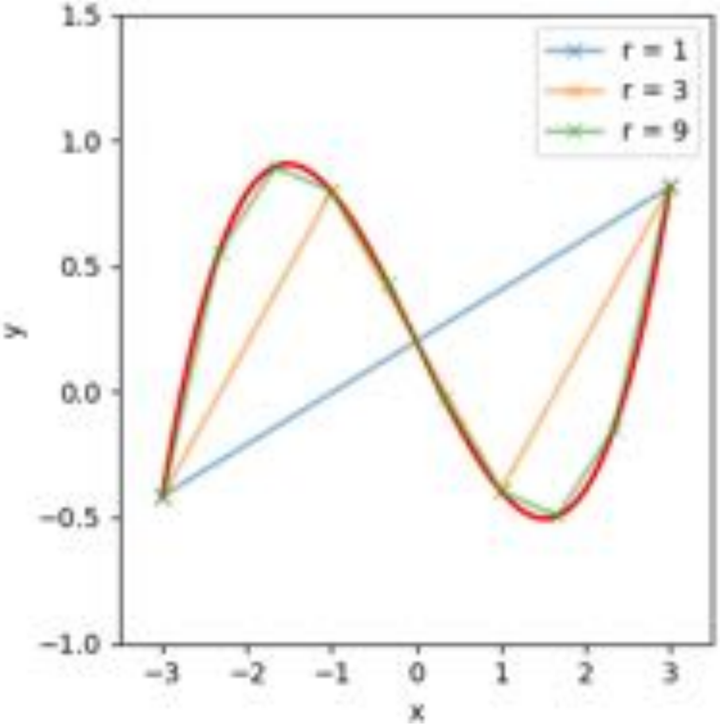
**ReLu Neuron**




$$g(\boldsymbol{z}_i) = \max(\boldsymbol{0}, \boldsymbol{z}_i)$$

$$g'(\boldsymbol{z}_i) = \begin{cases} 1, & \boldsymbol{z}_i \geq 0 \\ 0, & \boldsymbol{z}_i < 0 \end{cases}$$

# Universal Approximation Theorem




$$O\left(\binom{n}{d}^{d(l-1)} n^d\right)$$

$n$ = Number of Neurons per Layer
$l$ = Number of Hidden Layers
$d$ = Number of Inputs

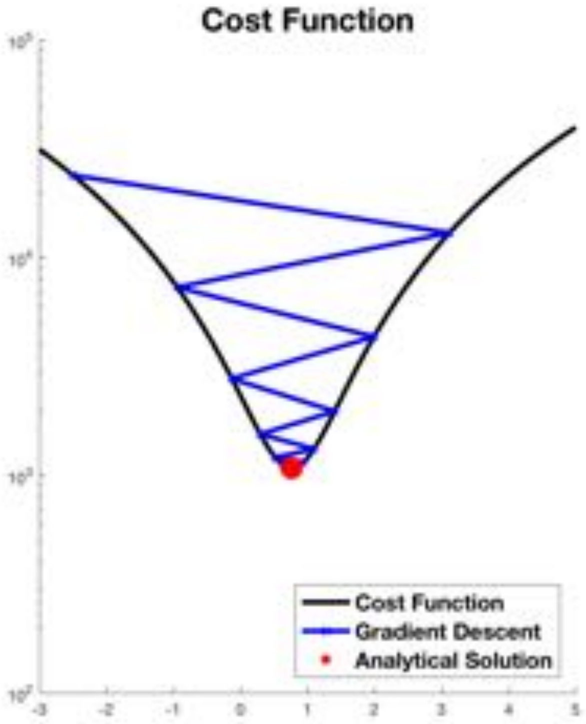
$$O\left(\binom{n}{1}^{1(1-1)} n^1\right) = O(n) \qquad l = 1,\ d = 1$$

$$O\left(\binom{n}{1}^{1(2-1)} n^1\right) = O(n^2) \qquad l = 2,\ d = 1$$

$$O\left(\binom{n}{1}^{1(k-1)} n^1\right) = O(n^k) \qquad l = k,\ d = 1$$

Kurt Hornik et. al., “Multilayer feedforward networks are universal approximators”, 1989
Guido Montufar et.al., “On the Number of Linear Regions of Deep Neural Networks”, 2014

# Gradient Descent




**Optimization Objective:**

$$\theta^* = \underset{\theta}{\operatorname{argmin}} J(\theta)$$

$$\theta_{i+1} = \theta_i + \Delta\theta_i = \theta_i - \alpha \, \nabla_{\theta_i} J(\theta)$$

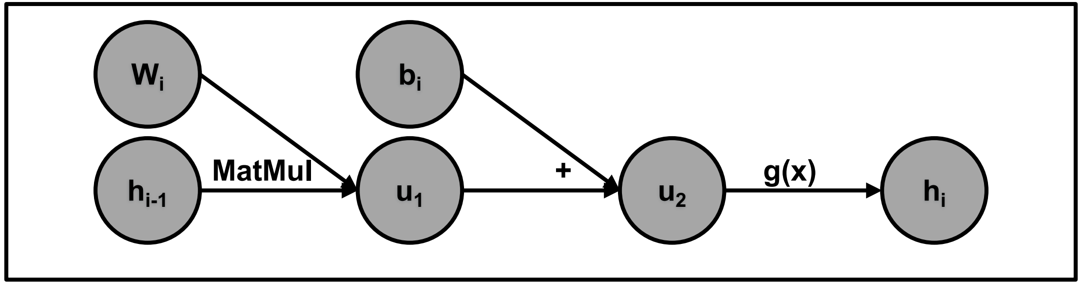
**Cost Functions:**

$$J(\theta) = \underset{p_d}{E}\{|y - f(x, \theta)|_1\}$$ → Median of $p(y \mid z)$

$$J(\theta) = \underset{p_d}{E}\{|y - f(x, \theta)|_2\}$$ → Mean of $p(y \mid z)$

$$J(\theta) = \underset{p_d}{E}\{-log(p_m(y \mid x, \theta))\}$$

# Backpropagation




$$\boldsymbol{u}_0 = \boldsymbol{h}_{i-1}$$

$$\boldsymbol{u}_1 = \boldsymbol{W}_i^T \boldsymbol{u}_0$$

$$\boldsymbol{u}_2 = \boldsymbol{u}_1 + \ \boldsymbol{b}_i$$

$$\boldsymbol{u}_3 = g(\boldsymbol{u}_2) = \boldsymbol{h}_i$$

$$\frac{d}{d\boldsymbol{u}_0}\boldsymbol{u}_1 = \boldsymbol{W}_i^T$$

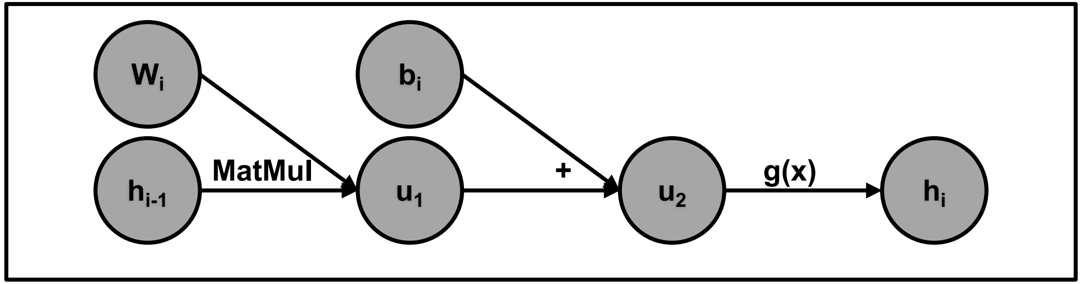
$$\frac{d}{d\boldsymbol{u}_1}\boldsymbol{u}_2 = \boldsymbol{I}$$

$$\frac{d}{d\boldsymbol{u}_2}\boldsymbol{u}_3 = \boldsymbol{g}'(\boldsymbol{u}_2)$$

$$\frac{d}{dW_i}\boldsymbol{u}_1 = [\boldsymbol{u}_0 \quad \dots \quad \boldsymbol{u}_0]$$

$$\frac{d}{d\boldsymbol{b}_i}\boldsymbol{u}_2 = \boldsymbol{I}$$

# Backpropagation




$$\nabla_{\boldsymbol{b}_i} J(\theta) = \frac{d\boldsymbol{u}_2}{d\boldsymbol{b}_i}\frac{d\boldsymbol{u}_3}{d\boldsymbol{u}_2} \odot \nabla J_{\boldsymbol{u}_3} \qquad = \boldsymbol{I}\ \boldsymbol{g}'(\boldsymbol{u}_2) \odot \nabla J_{\boldsymbol{u}_3}$$

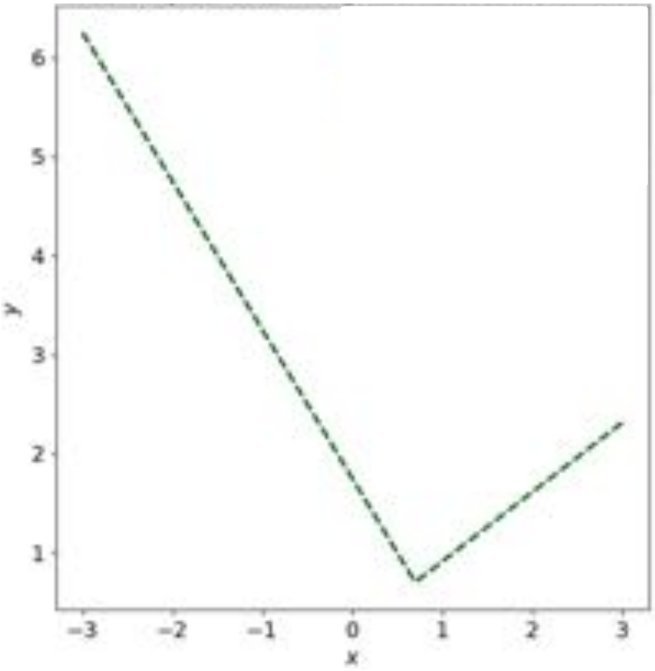
$$\nabla_{\boldsymbol{W}_i} J(\theta) = \frac{d\boldsymbol{u}_1}{d\boldsymbol{W}_1}\frac{d\boldsymbol{u}_2}{d\boldsymbol{u}_1}\frac{d\boldsymbol{u}_3}{d\boldsymbol{W}_i} \odot \nabla_{\boldsymbol{u}_3} J \ = (\boldsymbol{g}'(\boldsymbol{u}_2) \odot \nabla J_{\boldsymbol{u}_3})\ \boldsymbol{u}_{\boldsymbol{0}}^T$$

$$\nabla_{\boldsymbol{u}_0} J(\theta) = \frac{d\boldsymbol{u}_1}{d\boldsymbol{u}_0}\frac{d\boldsymbol{u}_2}{d\boldsymbol{u}_1}\frac{d\boldsymbol{u}_3}{d\boldsymbol{u}_2} \odot \nabla J_{\boldsymbol{u}_3} \ = \boldsymbol{W}_i^T\ \boldsymbol{g}'(\boldsymbol{u}_2) \odot \nabla J_{\boldsymbol{u}_3}$$

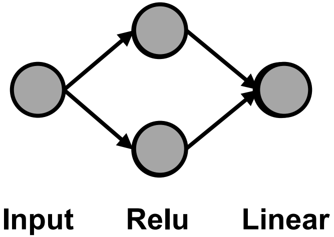
# Agenda

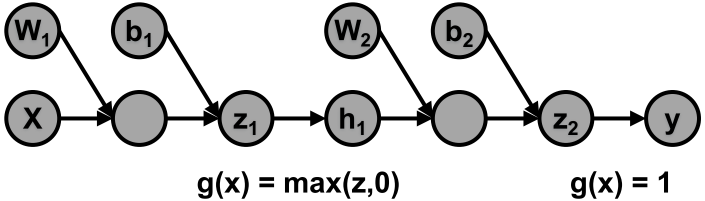
- Neural Network History
- Neural Network Basics
  - Computational Graph
  - Neurons
  - Parameter Learning
- **Optimization for Deep Neural Networks**
  - Non-Convex Cost Functions
  - Weight Initialization
  - Stochastic Gradient Descent (SGD)
  - Plugins for SGD

# Problem Statement




**Model:**



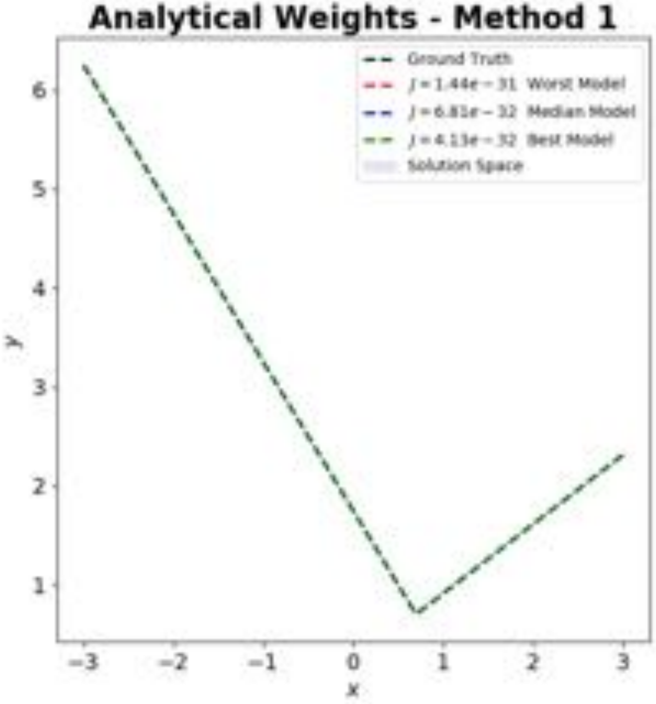



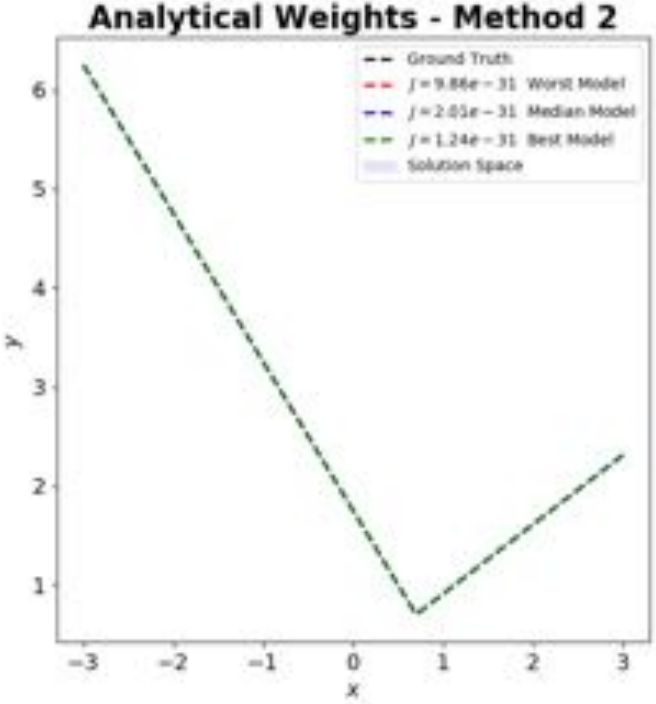

**Data:**

- 500 data points without noise

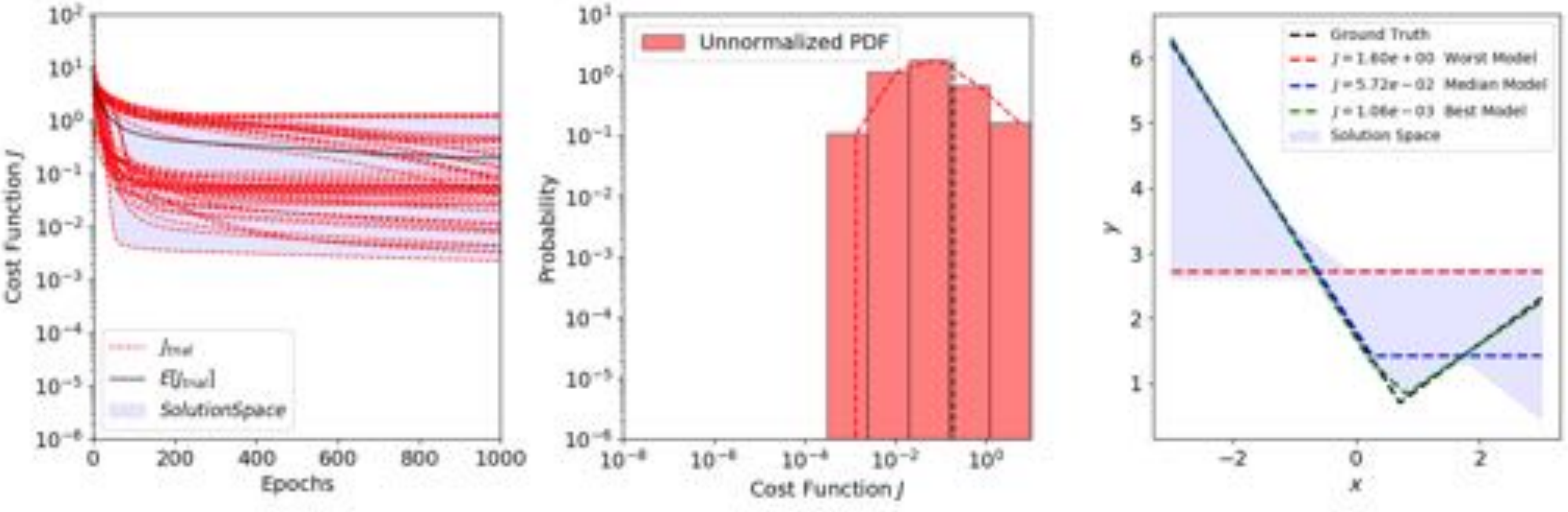
$$f(x) = m_1(x - x_0) + m_2(x - x_0) + b$$

# Problem Solution

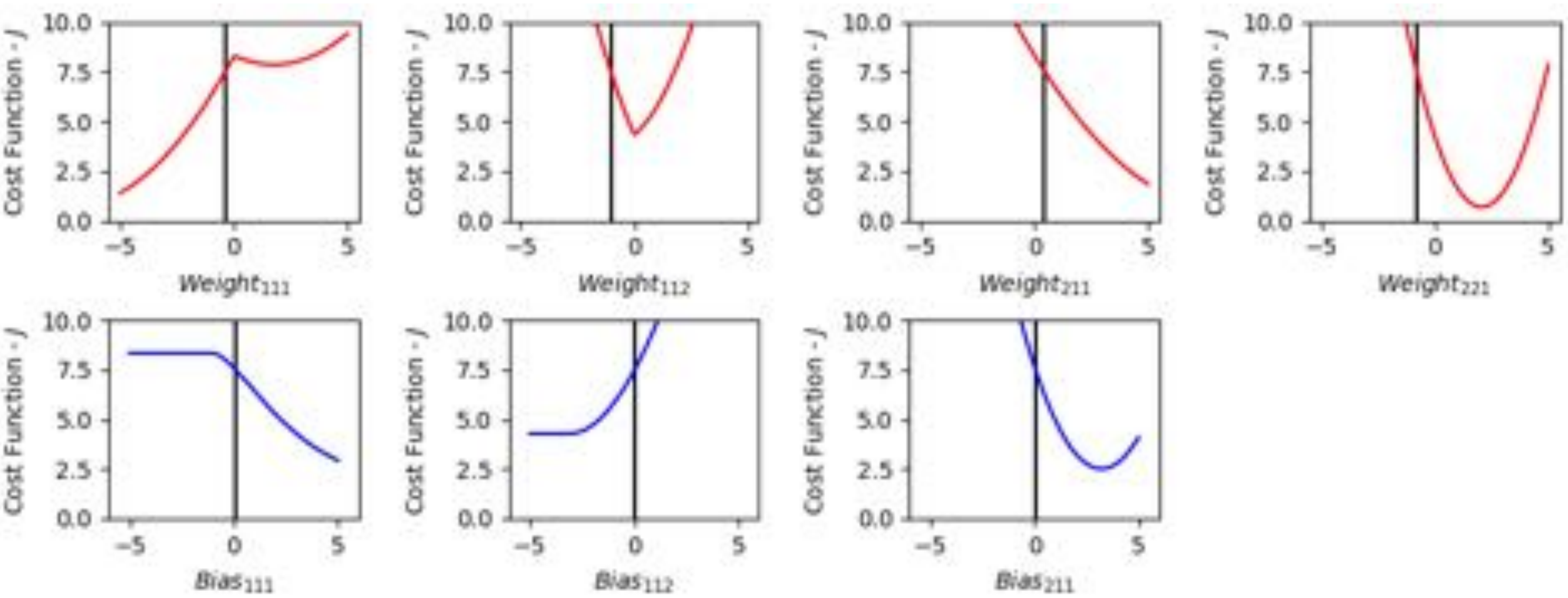


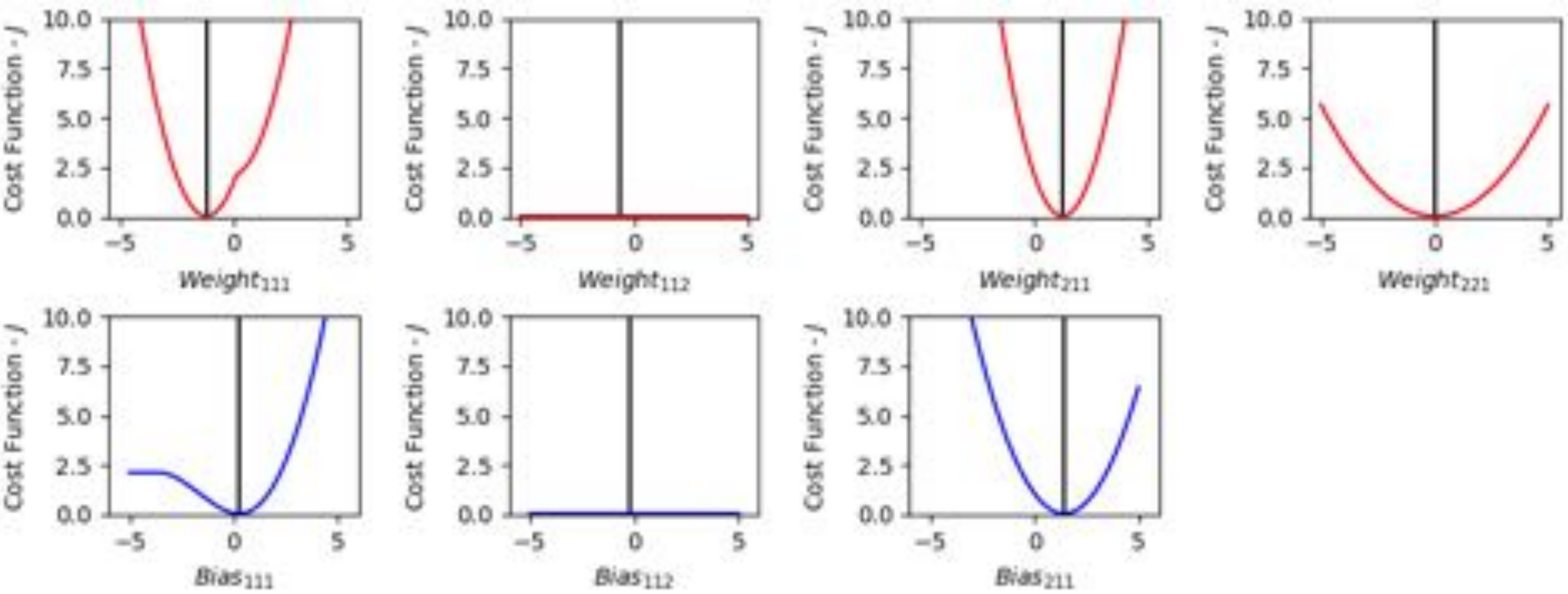
# Performance: Stochastic Gradient Descent



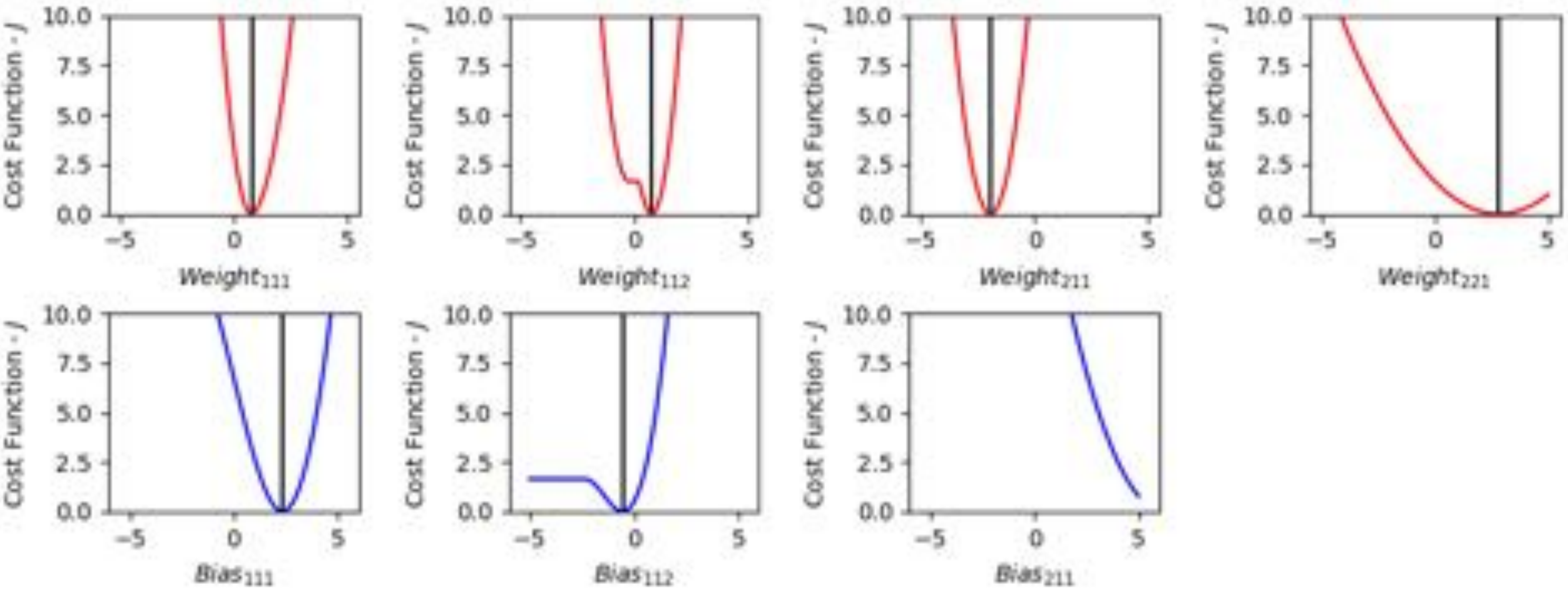
# Cost Function– Random Initialization



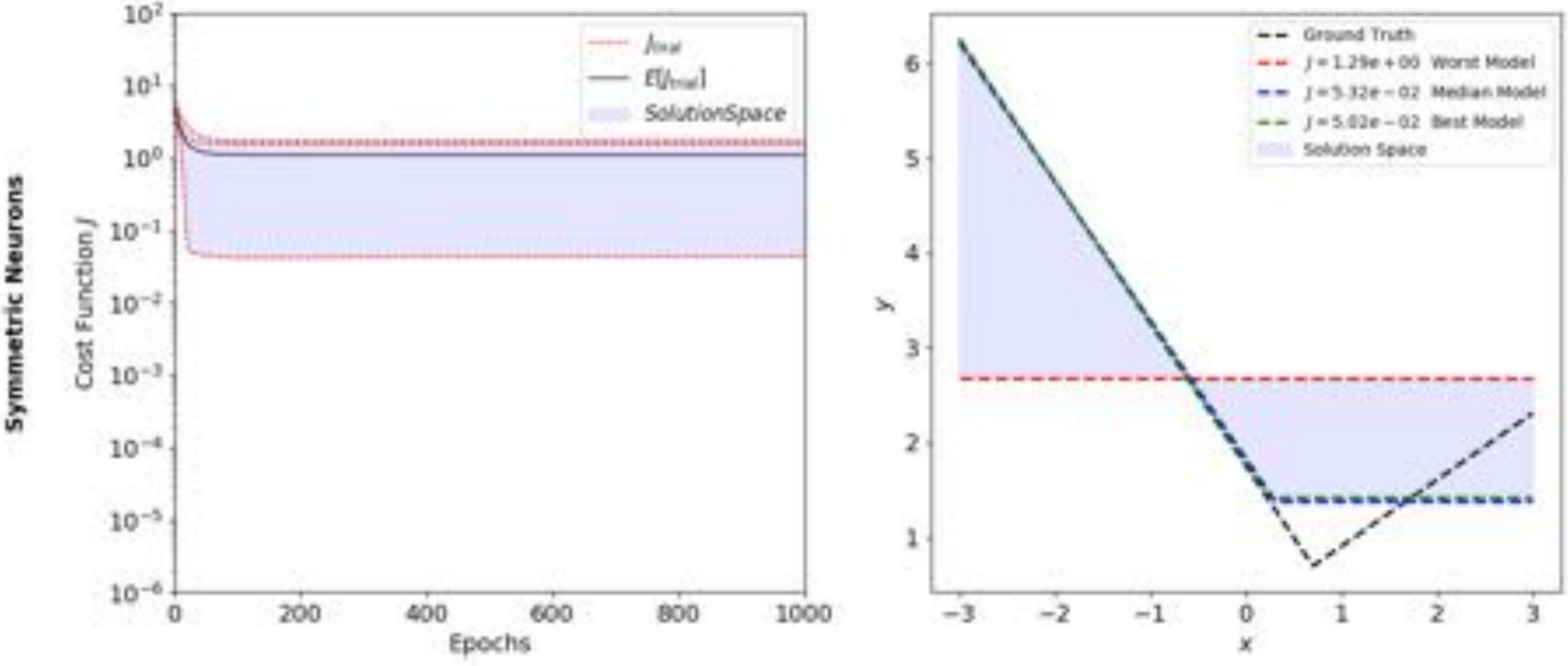
# Cost Function – After Training



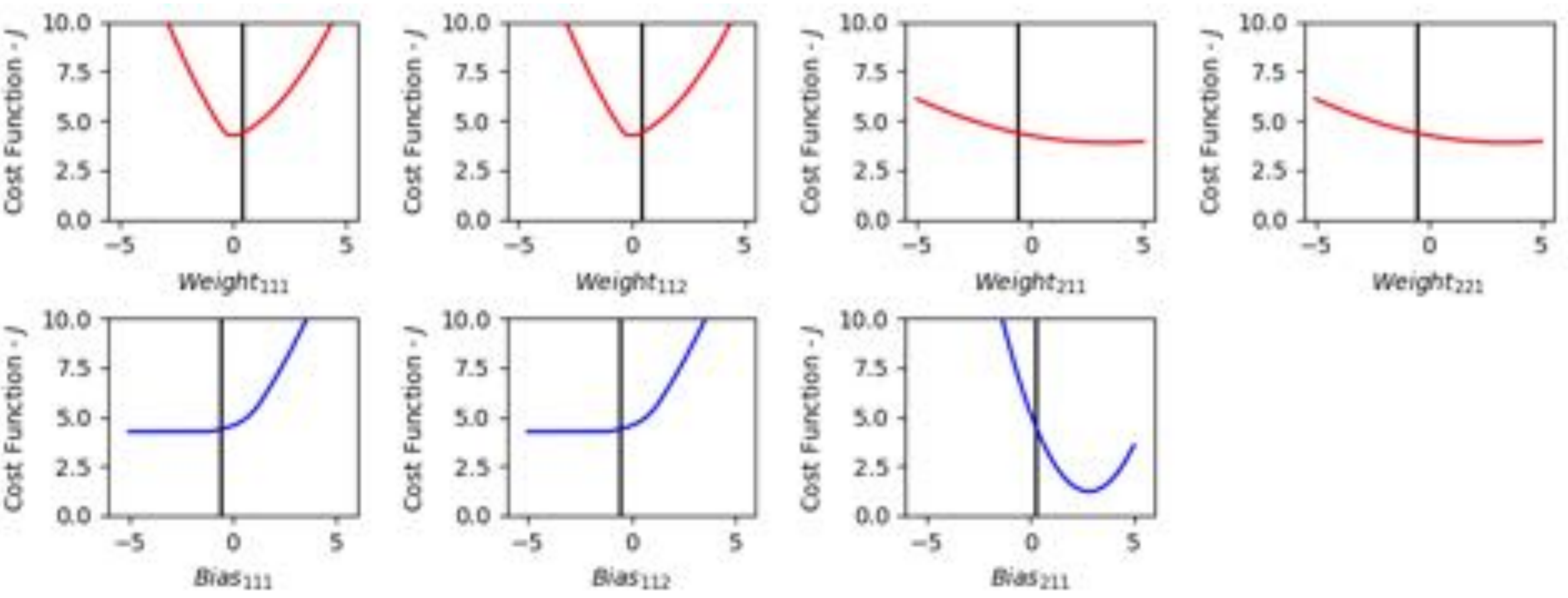
# Cost Function – Global Optimum



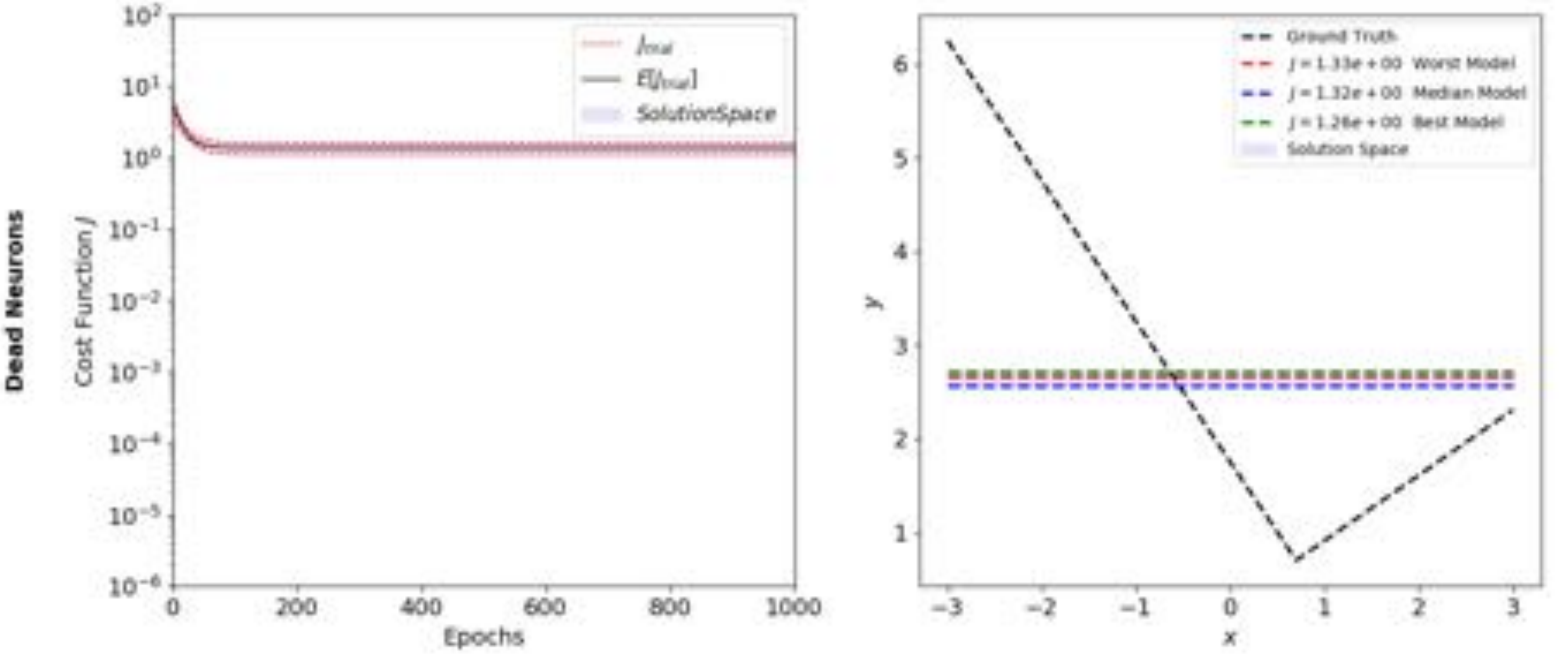
# Weight Initialization – Symmetric Neurons



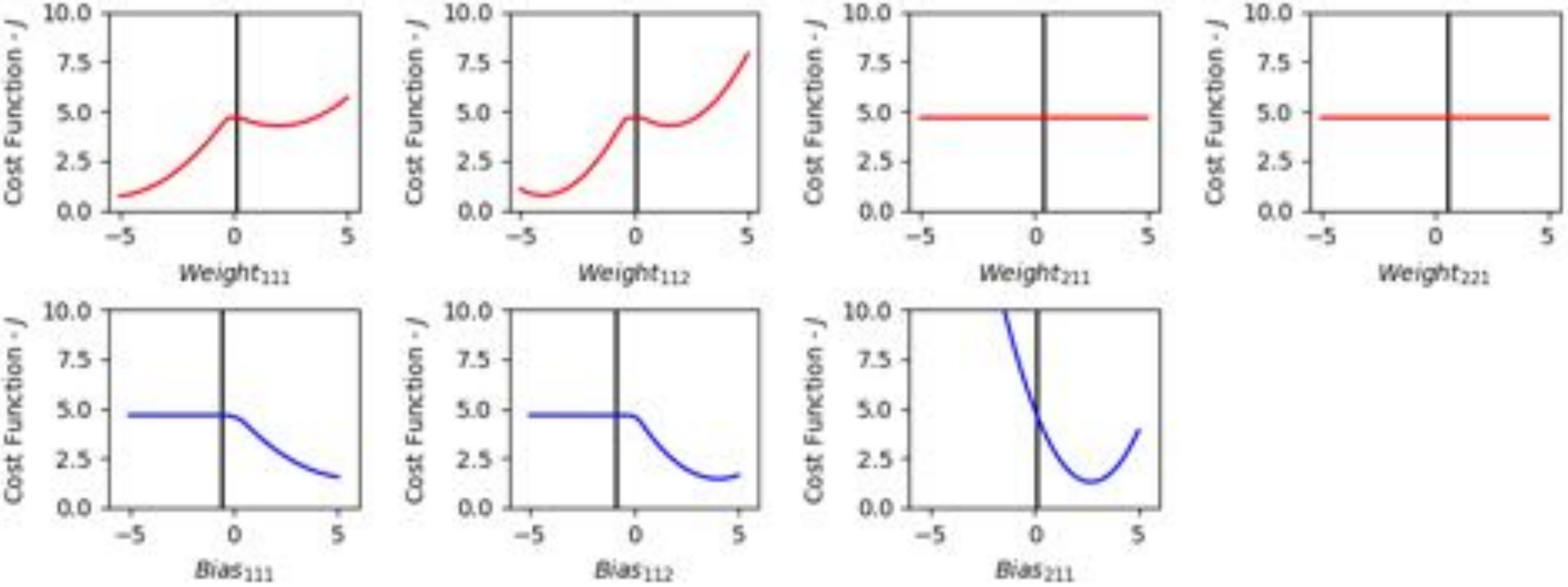
# Weight Initialization – Symmetric Neurons



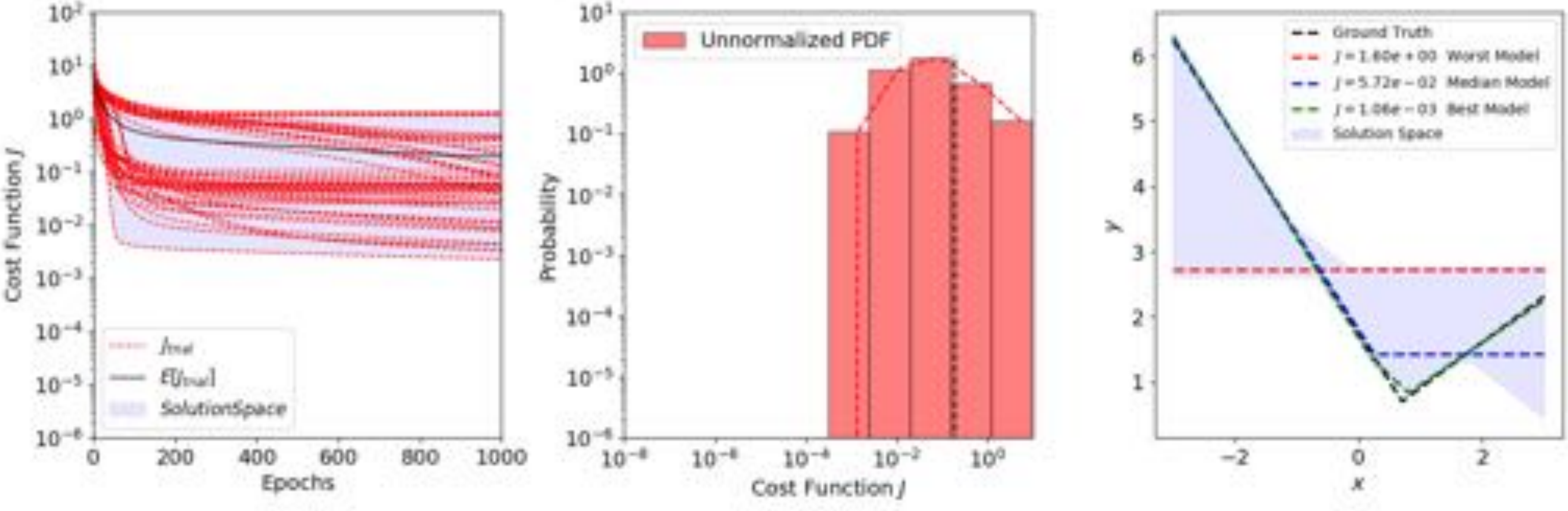
# Weight Initialization – Dead Neurons



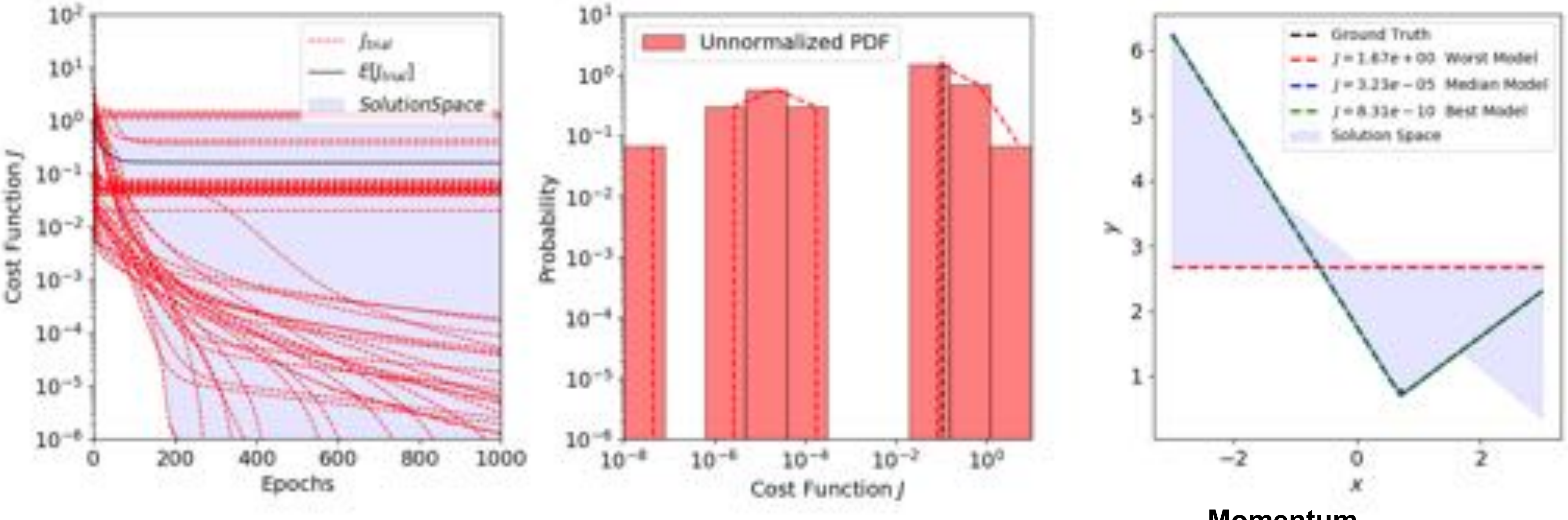
# Weight Initialization – Dead Neurons



# Performance: Stochastic Gradient Descent



# Performance: Momentum

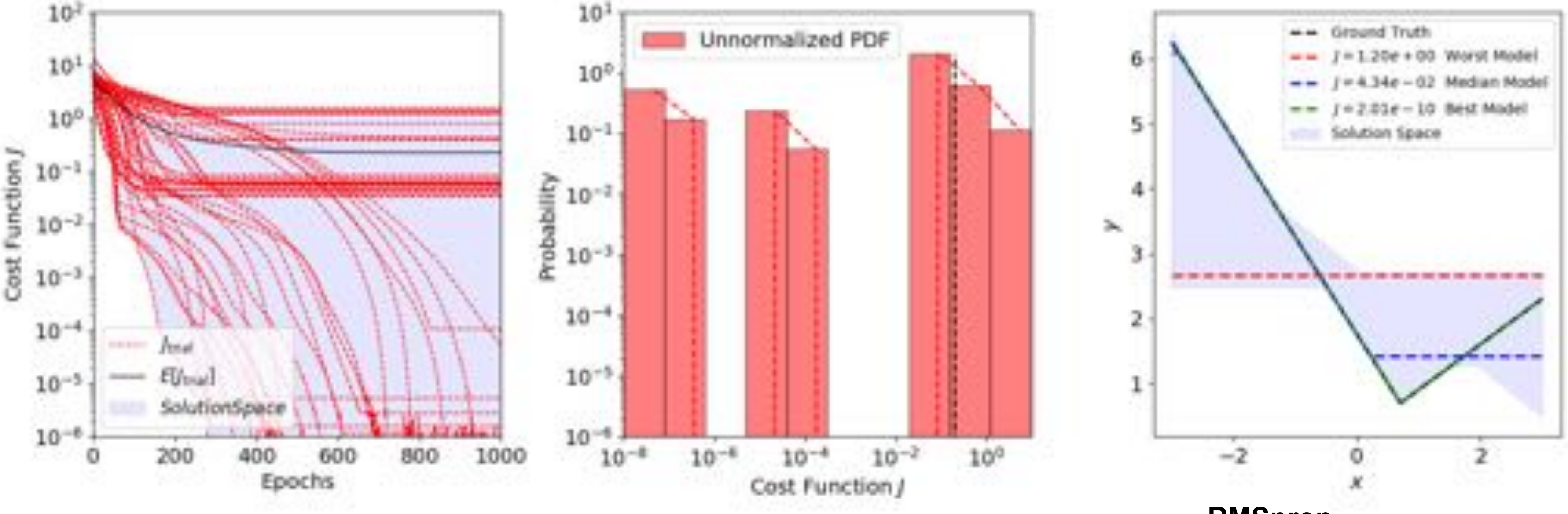



**Momentum**

Batch Size = 3

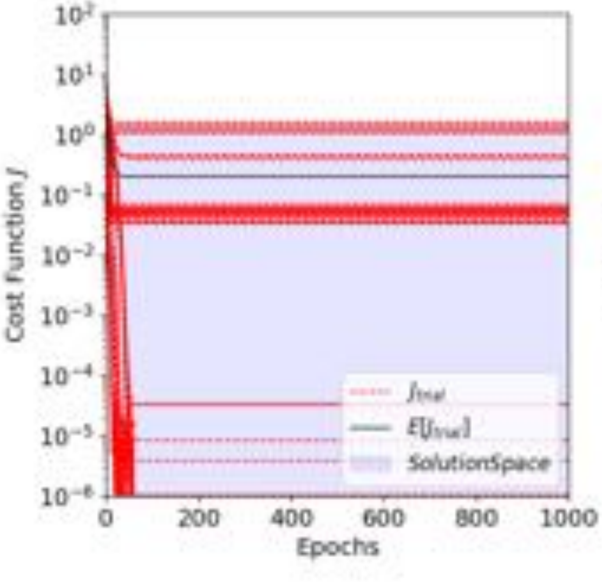
$\text{rho}_{\text{Momentum}}$ = 0.95

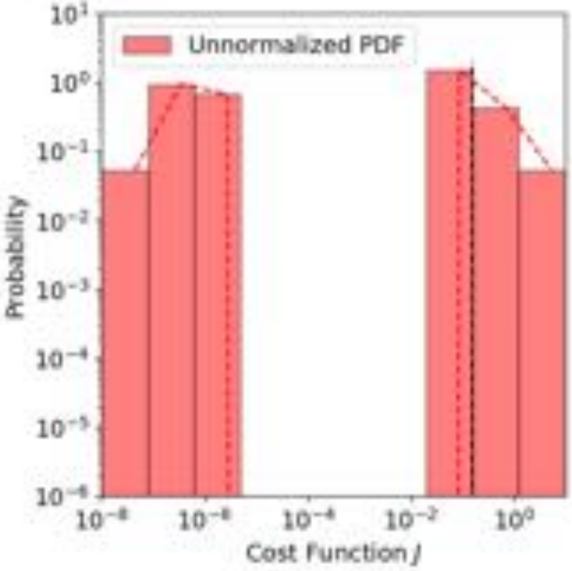
# Performance: RMSprop

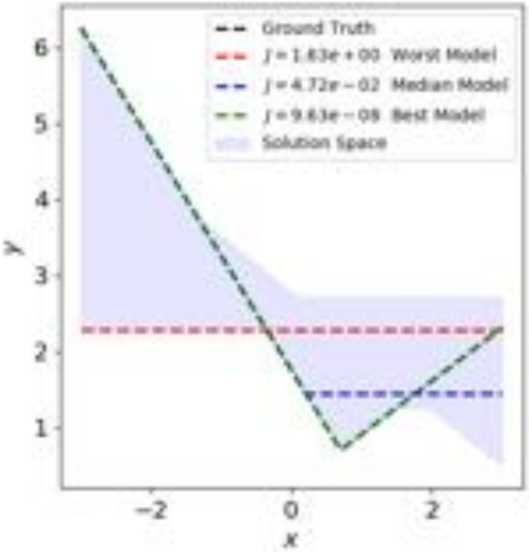



**RMSprop**
Batch Size = 3
$\text{rho}_{\text{learningRates}}$ = 0.9

# Performance: Momentum & RMSprop



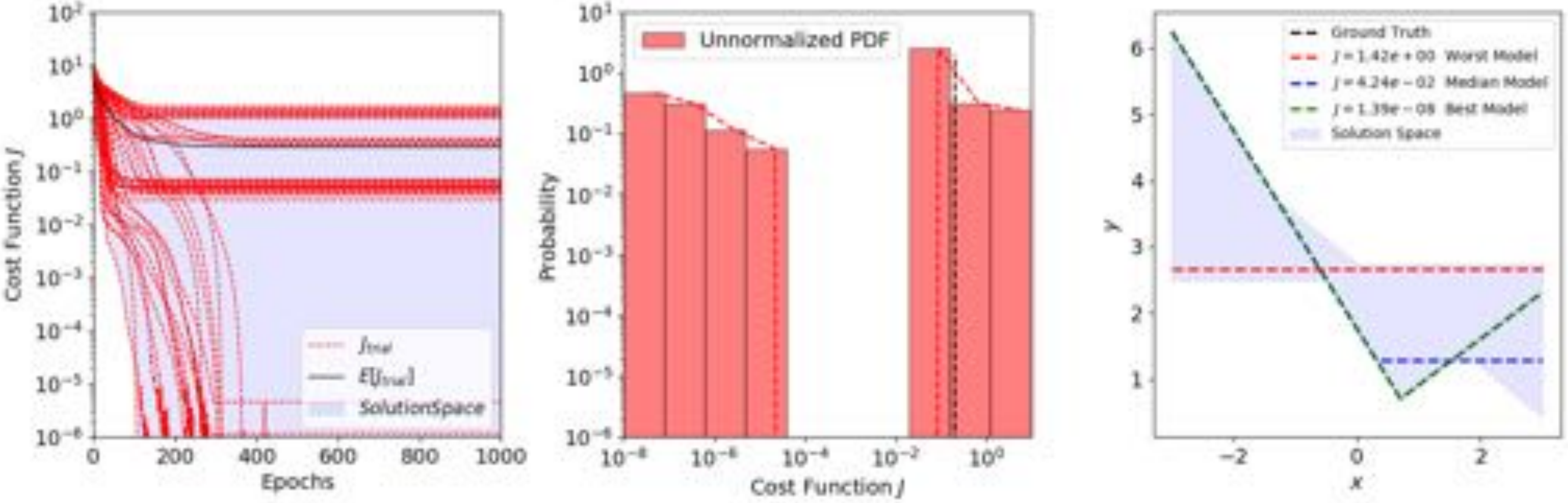

**Momentum & RMSprop**

Batch Size = 3

$\text{rho}_{\text{Momentum}}$ = 0.95

$\text{rho}_{\text{learningRates}}$ = 0.9

# Performance: ADAM



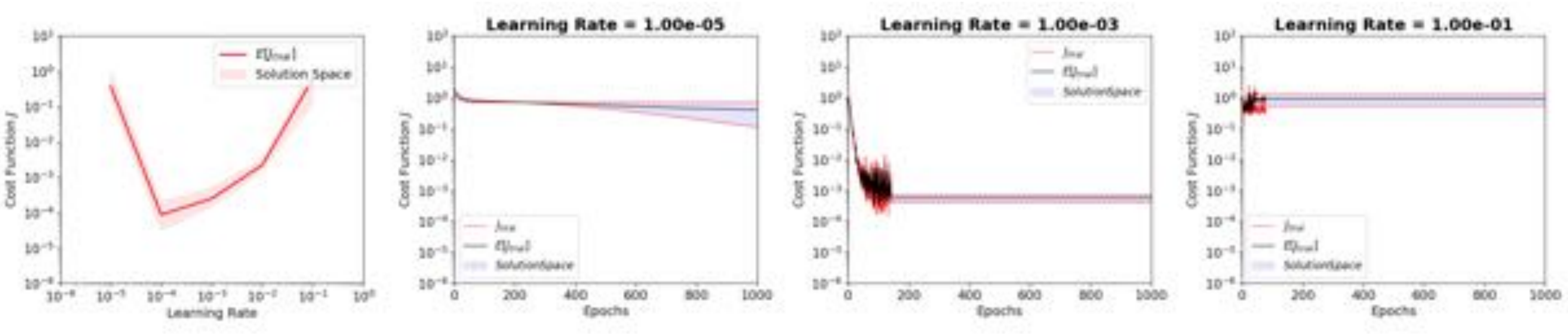

**ADAM**

Batch Size = 3

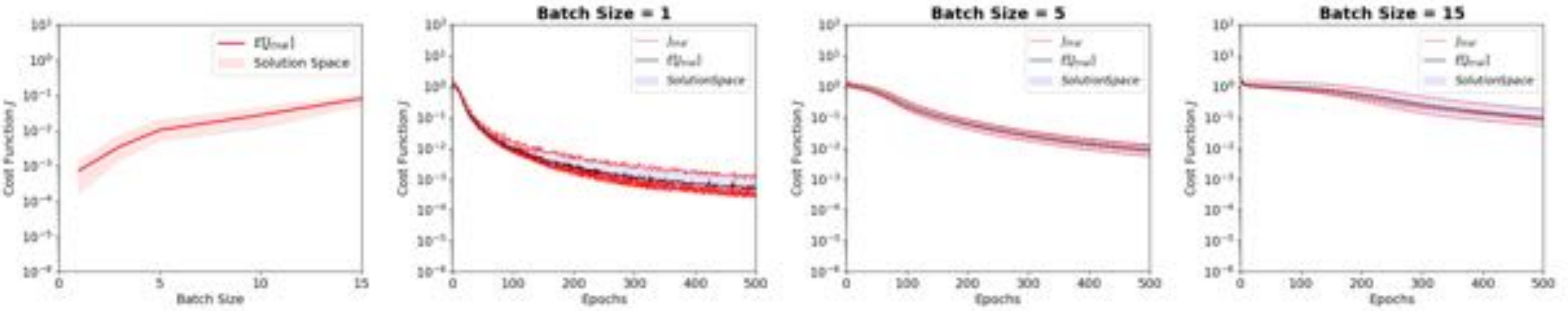
$\text{rho}_{\text{Momentum}}$ = 0.95

$\text{rho}_{\text{learningRates}}$ = 0.9

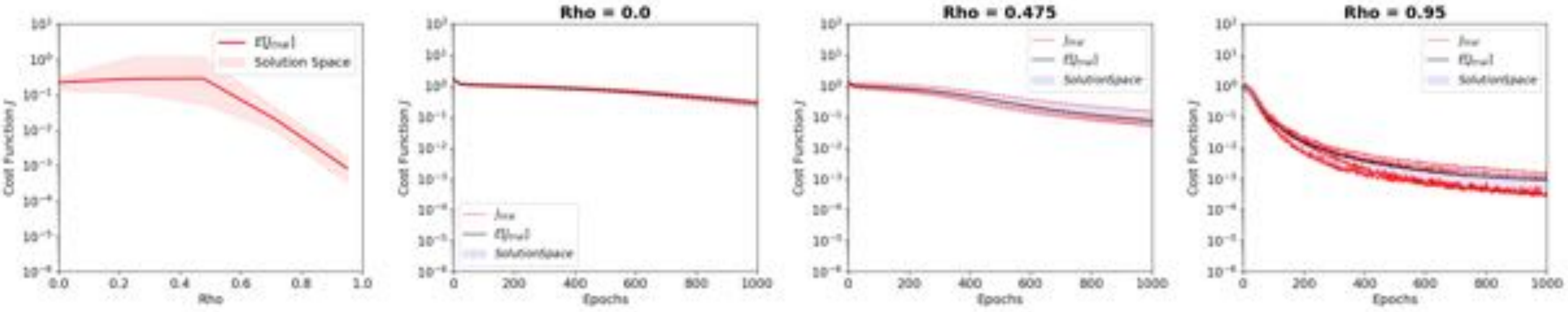
# Hyperparameter – Learning Rate



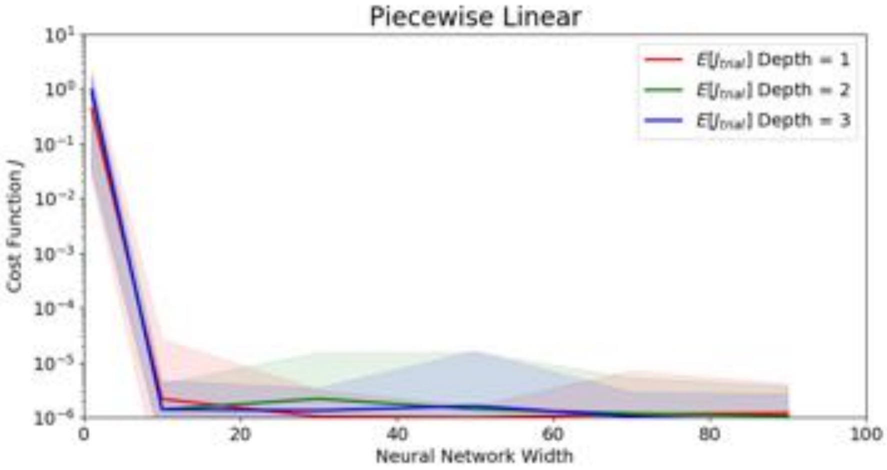
# Hyperparameter – Batch Size

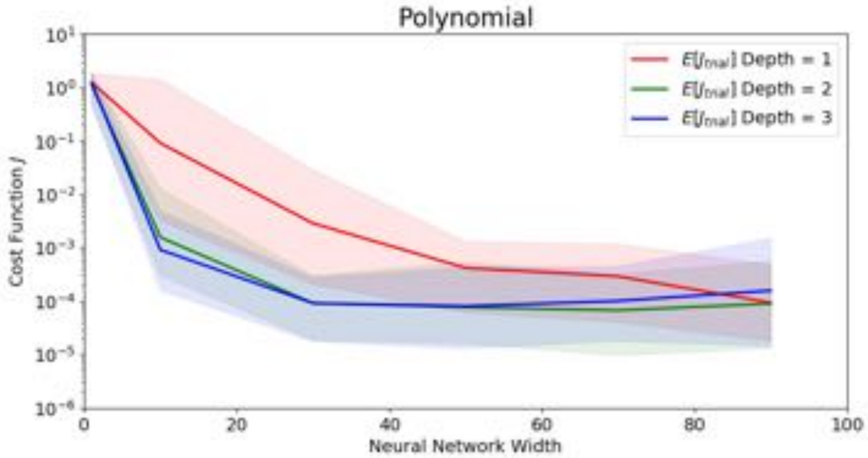


# Hyperparameter – Momentum

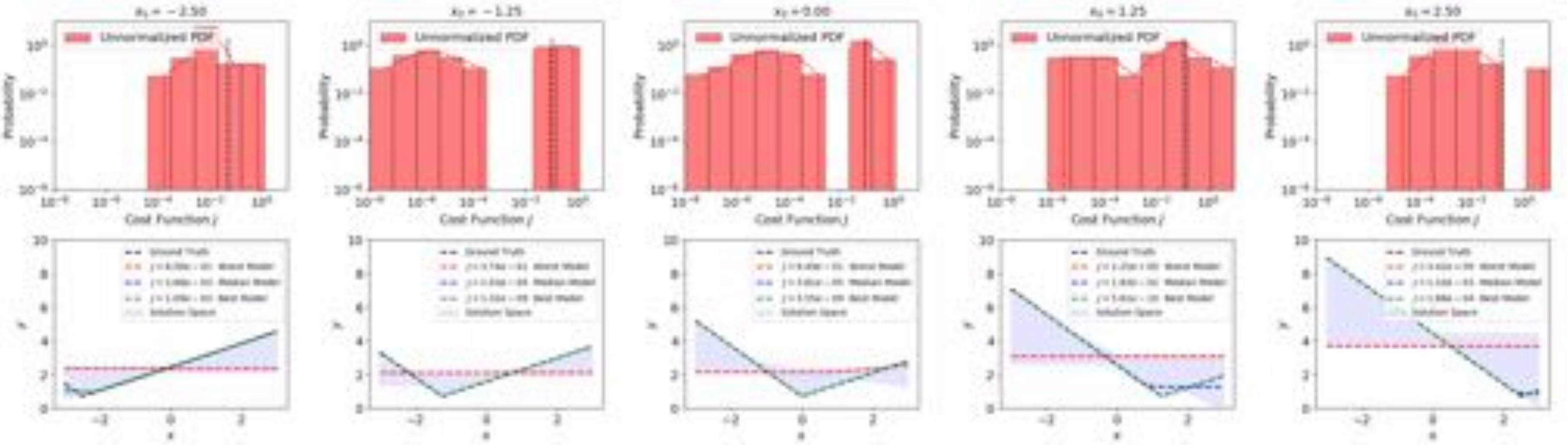


# Hyperparameter – Network Dimensions

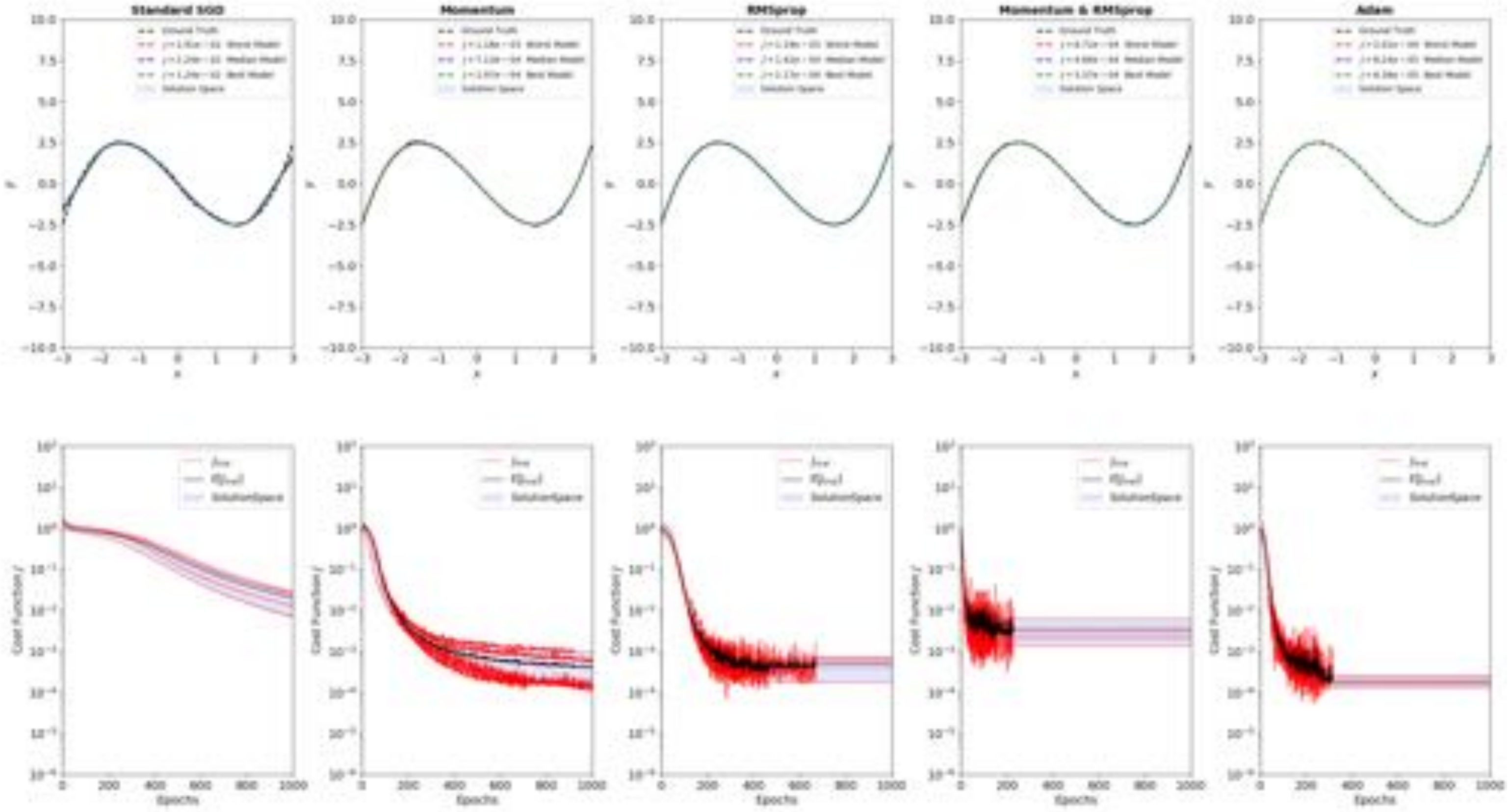


# Performance w.r.t. $x_0$



# Comparison SGD, Momentum, RMSprop, ADAM



# Neural Network Dimension