School of Computer Science
Carnegie Mellon

10-601B Introduction to Machine Learning

Deep Learning
(Part I)

Matt Gormley
Lecture 16
October 24, 2016

Readings:
Nielsen (online book)
Neural Networks and Deep Learning

Reminders

- Midsemester grades released today

Outline

- **Deep Neural Networks (DNNs)**
  - Three ideas for training a DNN
  - Experiments: MNIST digit classification
  - Autoencoders
  - Pretraining
- **Convolutional Neural Networks (CNNs)**
  - Convolutional layers
  - Pooling layers
  - Image recognition
- **Recurrent Neural Networks (RNNs)**
  - Bidirectional RNNs
  - Deep Bidirectional RNNs
  - Deep Bidirectional LSTMs
  - Connection to forward-backward algorithm

Part I

Part II

PRE-TRAINING FOR DEEP NETS

Goals for Today's Lecture

1. Explore a **new class of decision functions** (Deep Neural Networks)
2. Consider **variants of this recipe** for training

A Recipe for

2. Choose each of these:
   - Decision function

$$\hat{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x}_i)$$

   - Loss function

$$\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}_i) \in \mathbb{R}$$

4. Train with SGD:
(take small steps opposite the gradient)

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} - \eta_t \nabla \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}_i), \boldsymbol{y}_i)$$

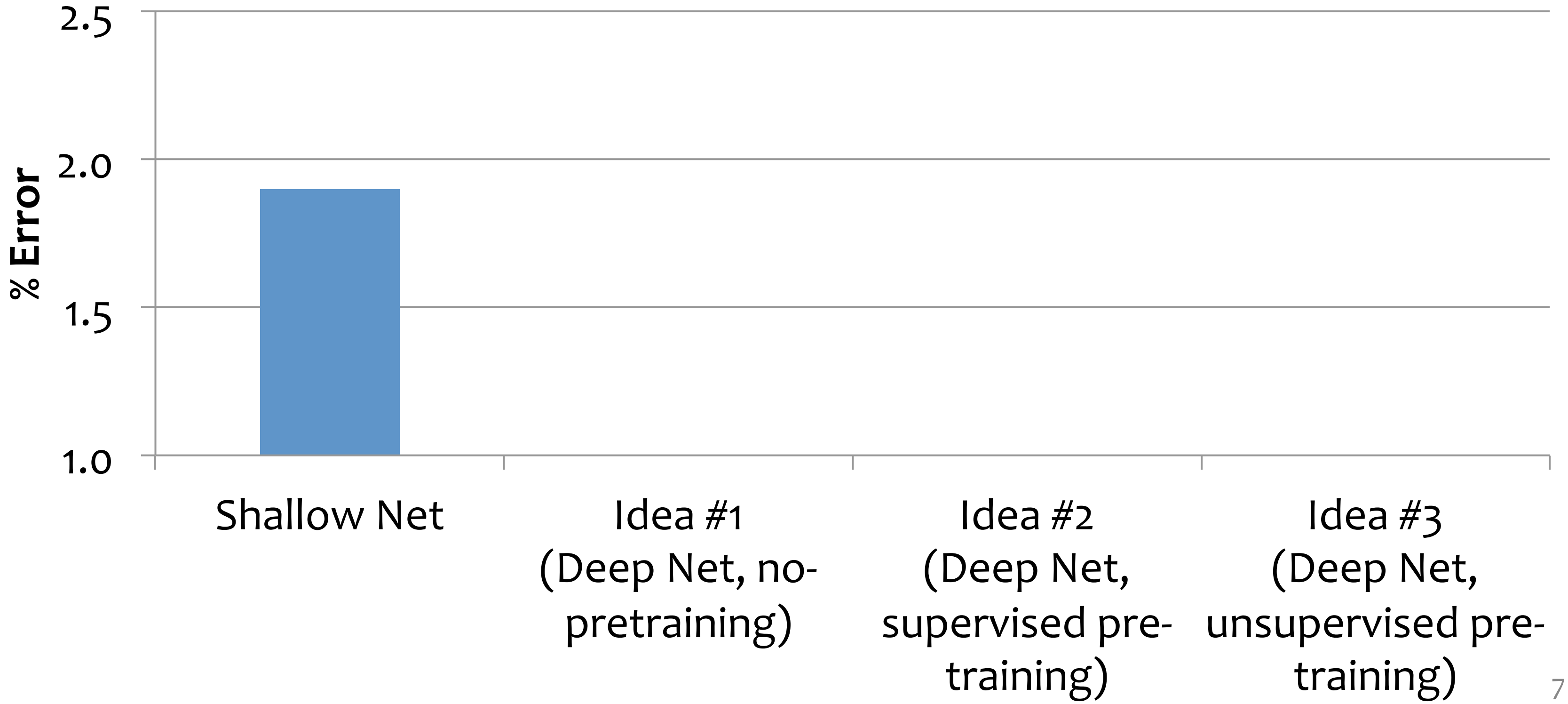
Training

Idea #1: No pre-training

- **Idea #1: (Just like a shallow network)**
  - Compute the supervised gradient by backpropagation.
  - Take small steps in the direction of the gradient (SGD)

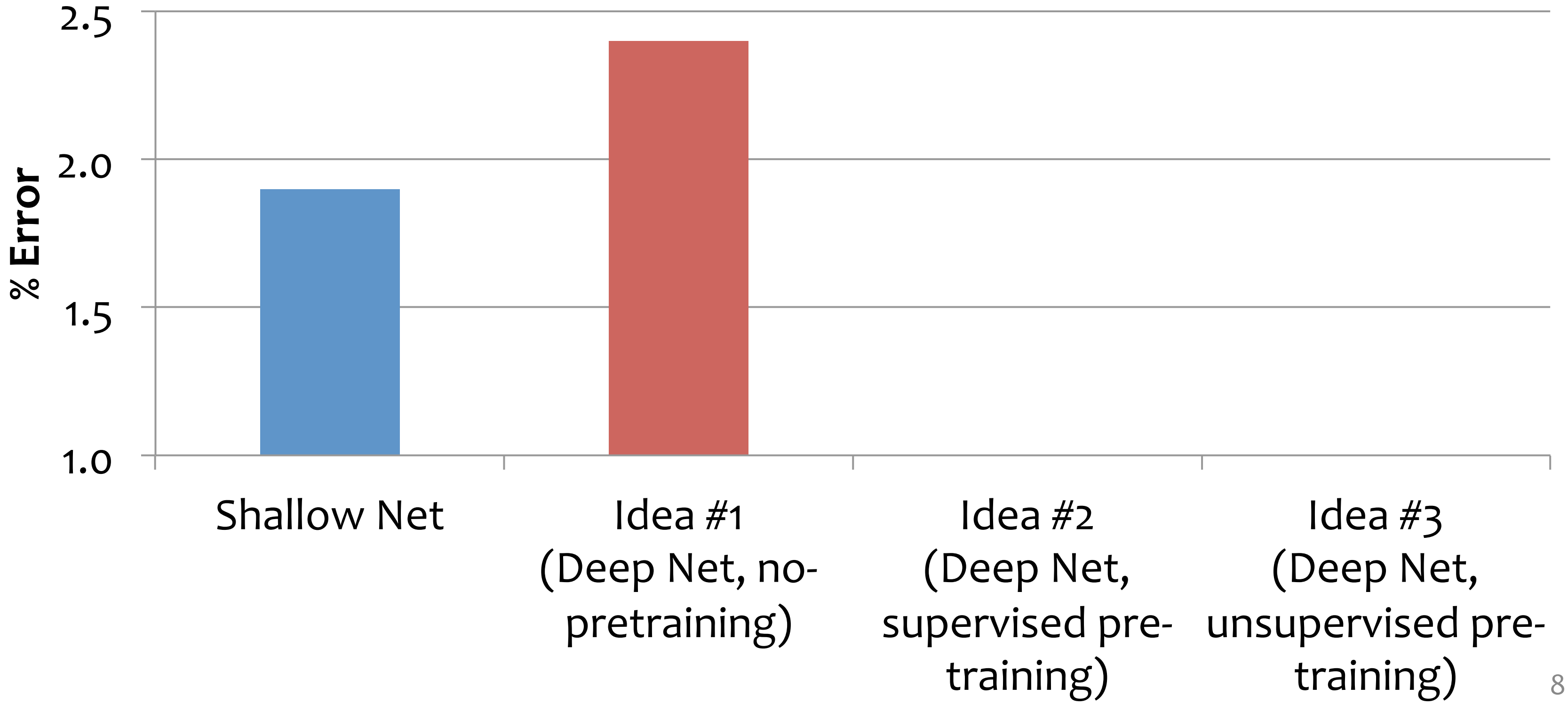
Comparison on MNIST

- Results from Bengio et al. (2006) on MNIST digit classification task
- Percent error (lower is better)



Comparison on MNIST

- Results from Bengio et al. (2006) on MNIST digit classification task
- Percent error (lower is better)



Idea #1: No pre-training

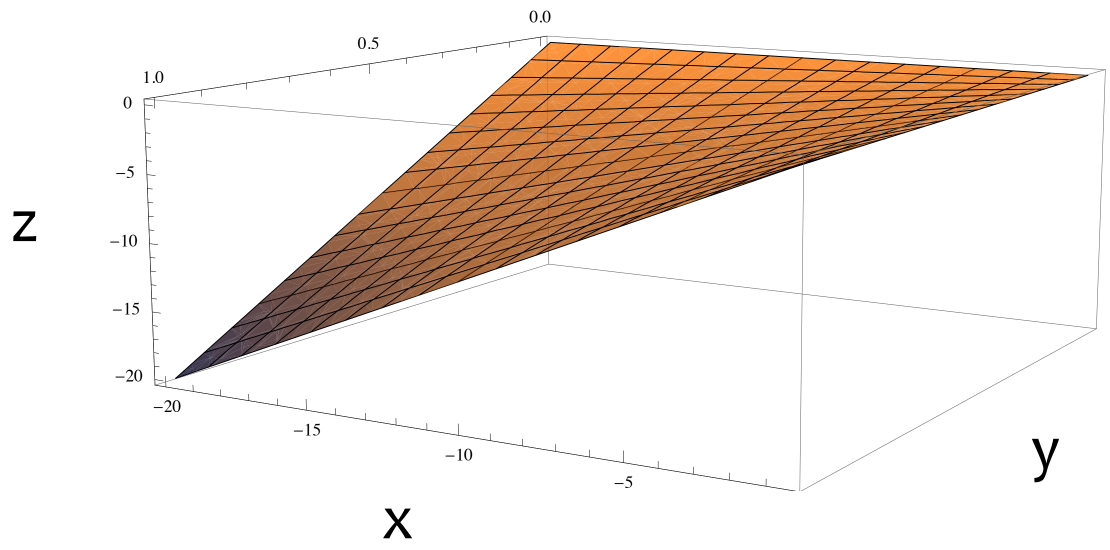
- **Idea #1: (Just like a shallow network)**
  - Compute the supervised gradient by backpropagation.
  - Take small steps in the direction of the gradient (SGD)

- What goes wrong?
  A. Gets stuck in local optima
     - Nonconvex objective
     - Usually start at a random (bad) point in parameter space
  B. Gradient is progressively getting more dilute
     - "Vanishing gradients"

Training

Problem A: *Nonconvexity*

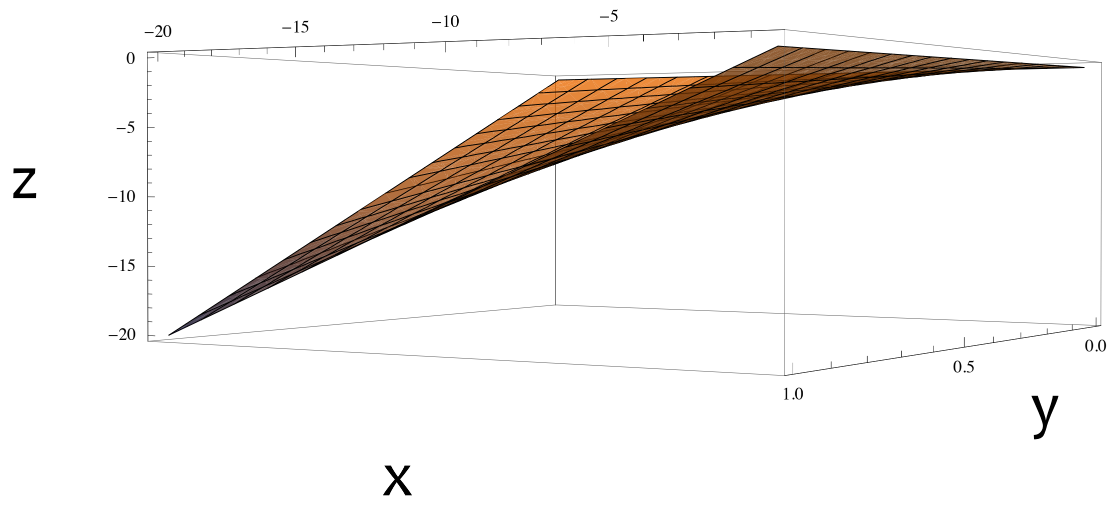
- Where does the nonconvexity come from?
- Even a simple quadratic z = xy objective is nonconvex:



Training

Problem A: *Nonconvexity*

- Where does the nonconvexity come from?
- Even a simple quadratic z = xy objective is nonconvex:

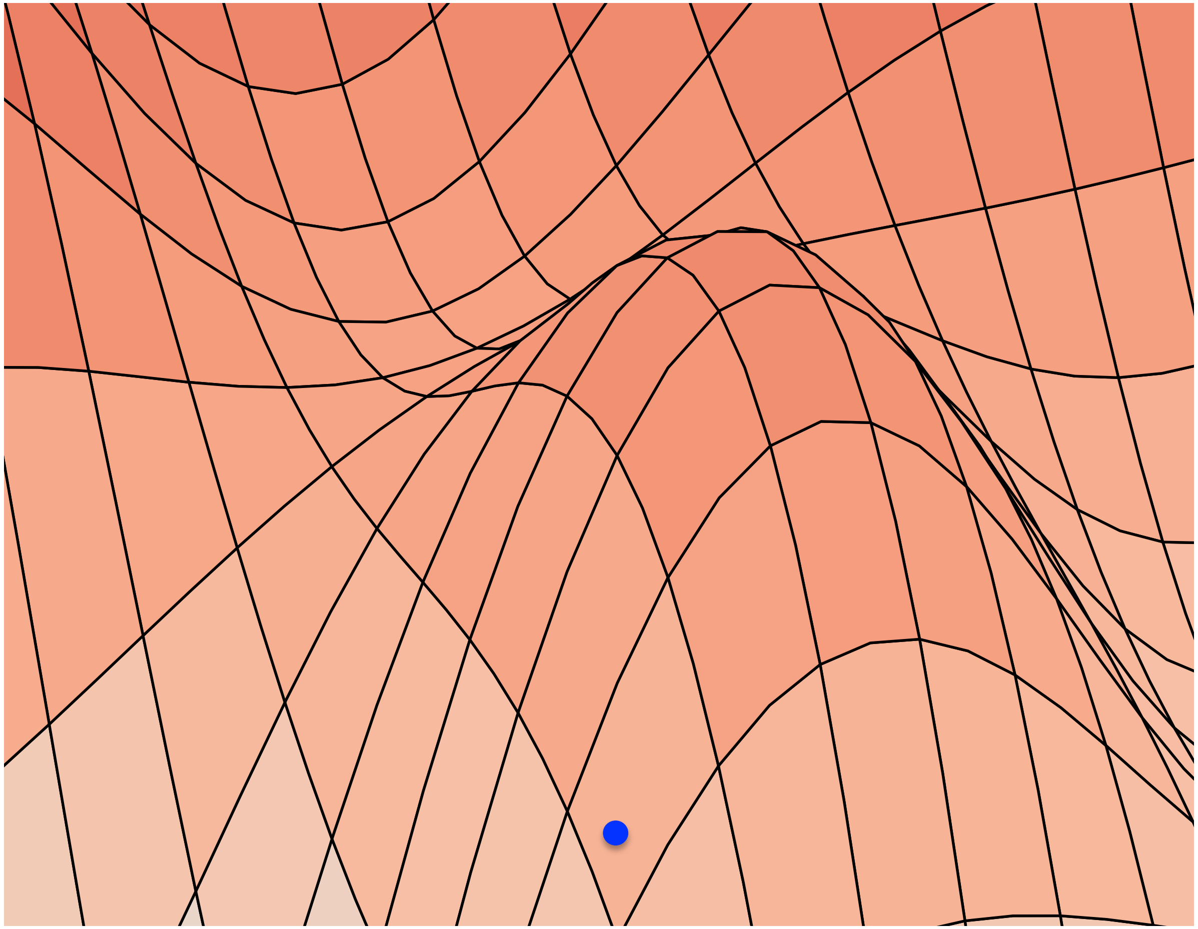


Training

Problem A: *Nonconvexity*

Stochastic Gradient Descent…

…climbs to the top of the nearest hill…

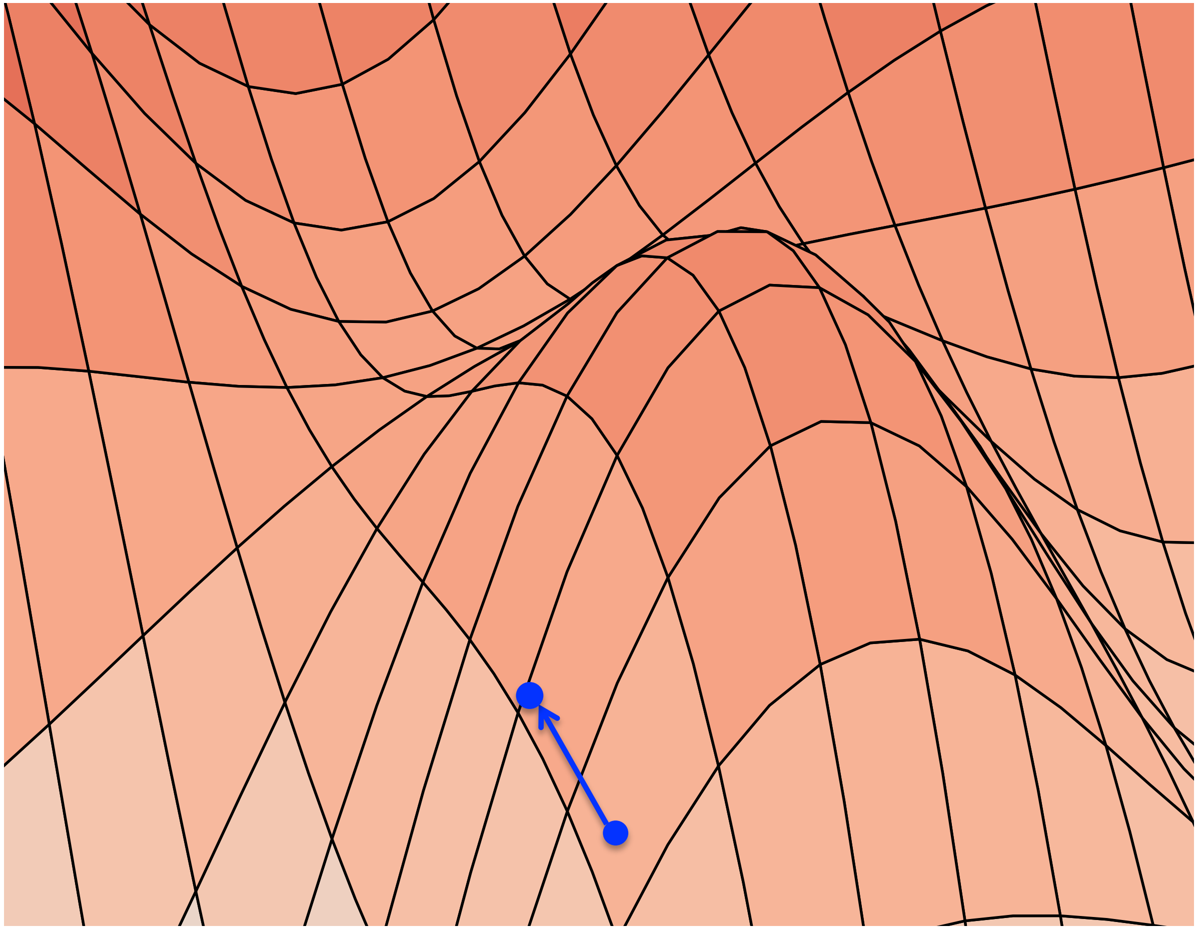


Training

Problem A: *Nonconvexity*

Stochastic Gradient Descent…

…climbs to the top of the nearest hill…

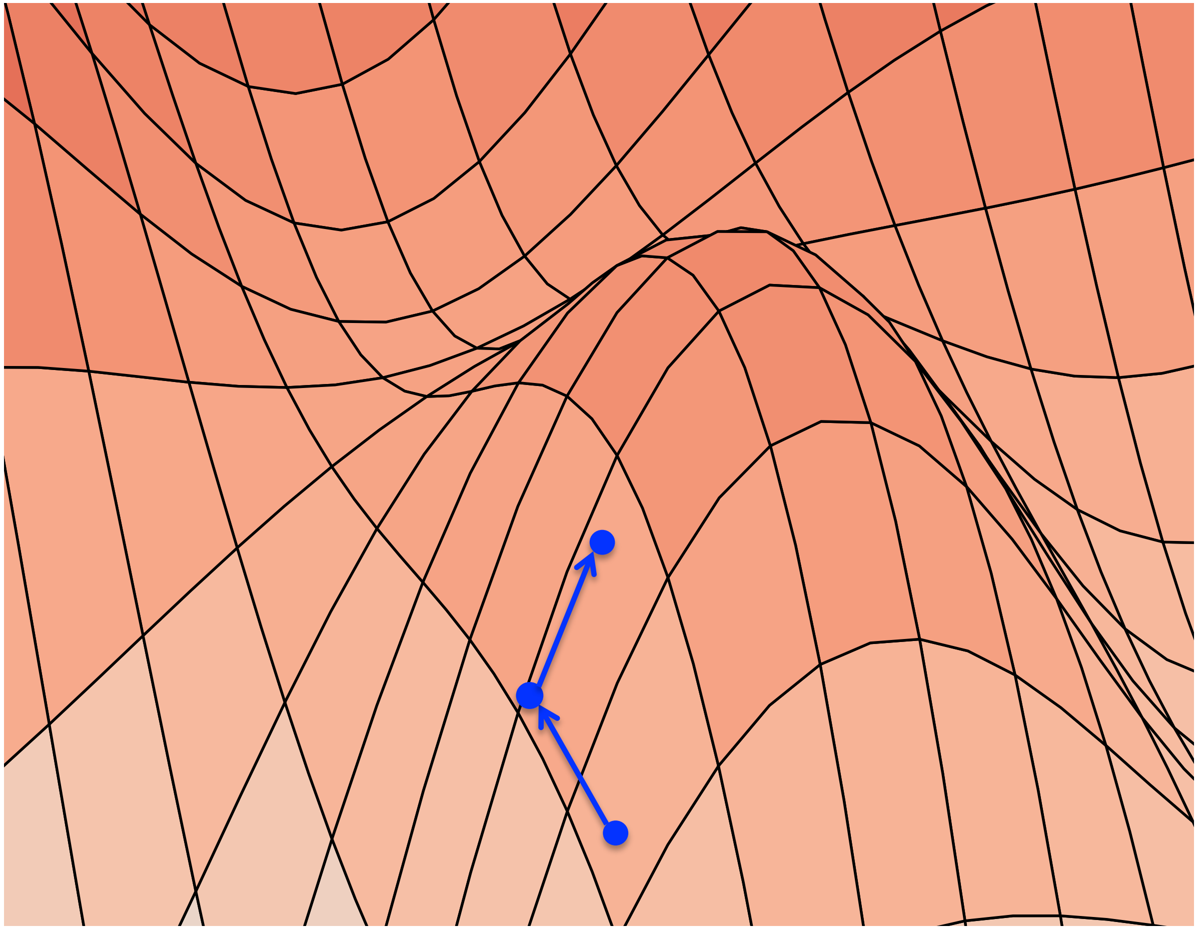


Training

Problem A: *Nonconvexity*

Stochastic Gradient Descent…

…climbs to the top of the nearest hill…

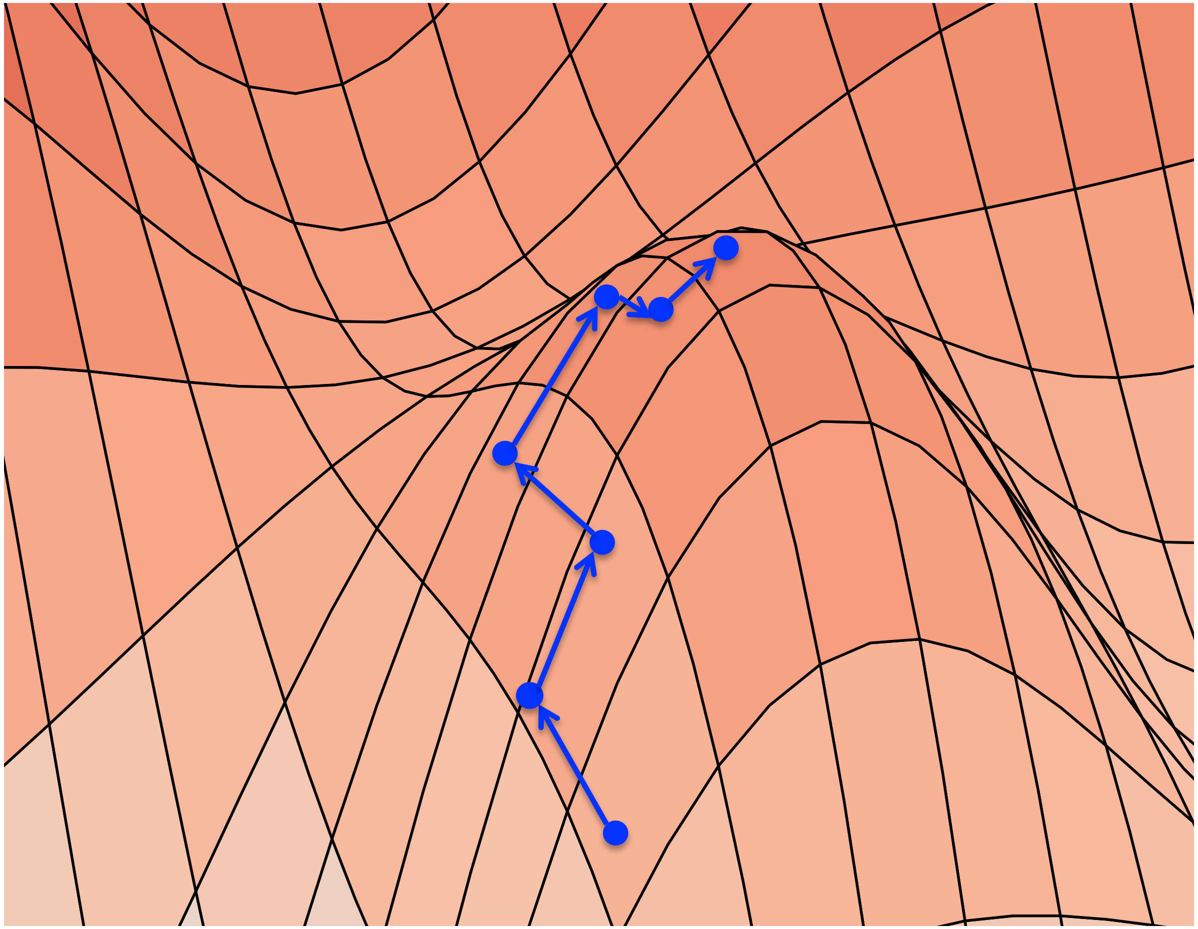


Training

Problem A: *Nonconvexity*

Stochastic Gradient Descent…

…climbs to the top of the nearest hill…



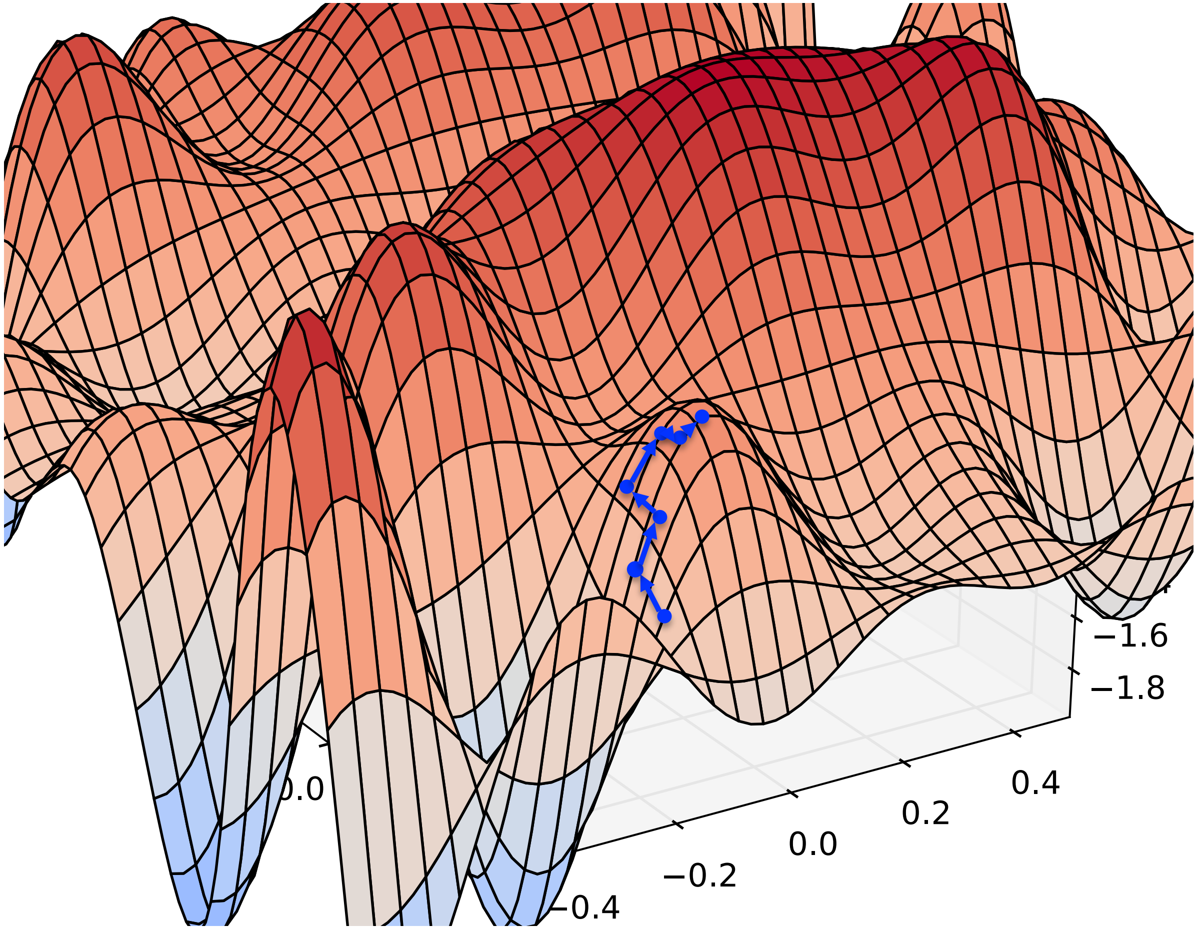
Training

Problem A: *Nonconvexity*

Stochastic Gradient Descent…

… climbs to the top of the nearest hill…

… which might not lead to the top of the mountain

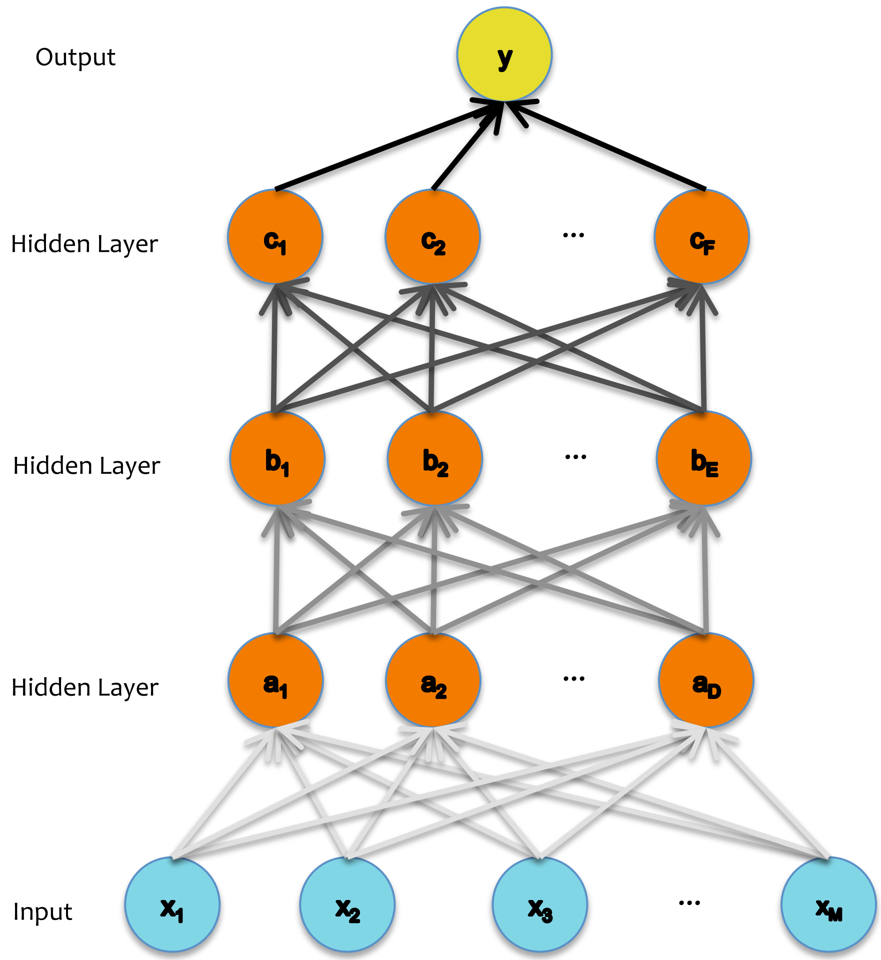


Training

Problem B: *Vanishing Gradients*

The gradient for an edge at the base of the network depends on the gradients of many edges above it

The chain rule multiplies many of these partial derivatives together

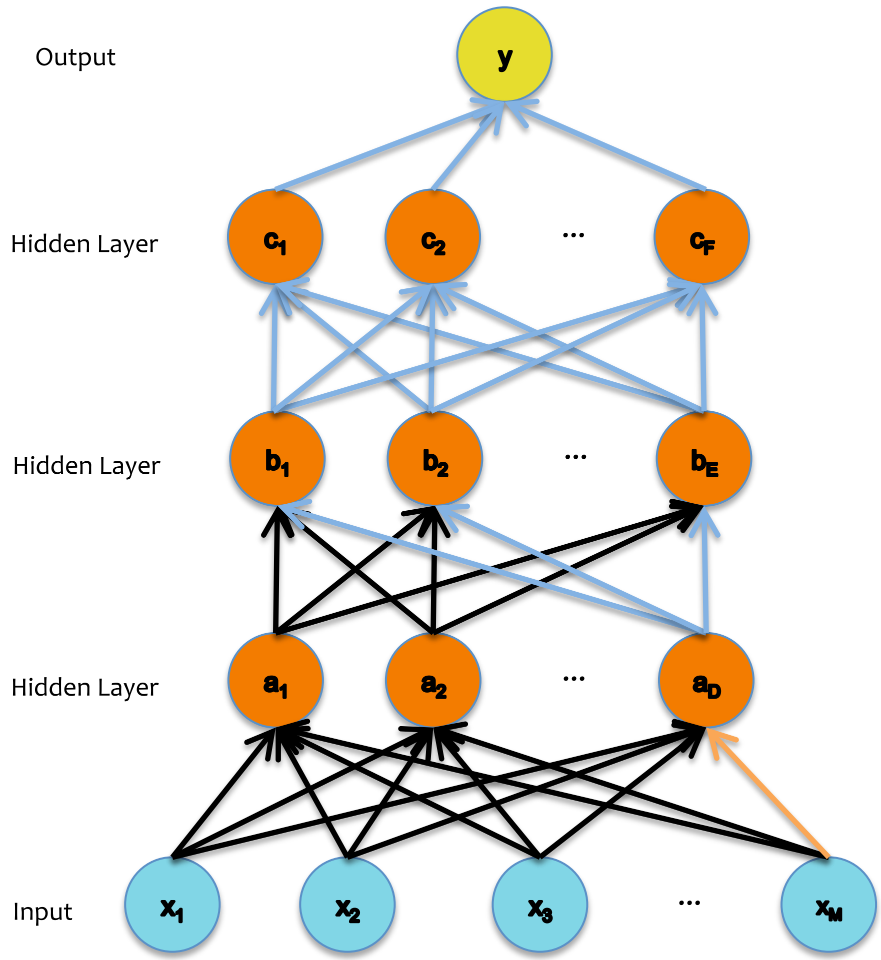


Training

Problem B: *Vanishing Gradients*

The gradient for an edge at the base of the network depends on the gradients of many edges above it

The chain rule multiplies many of these partial derivatives together

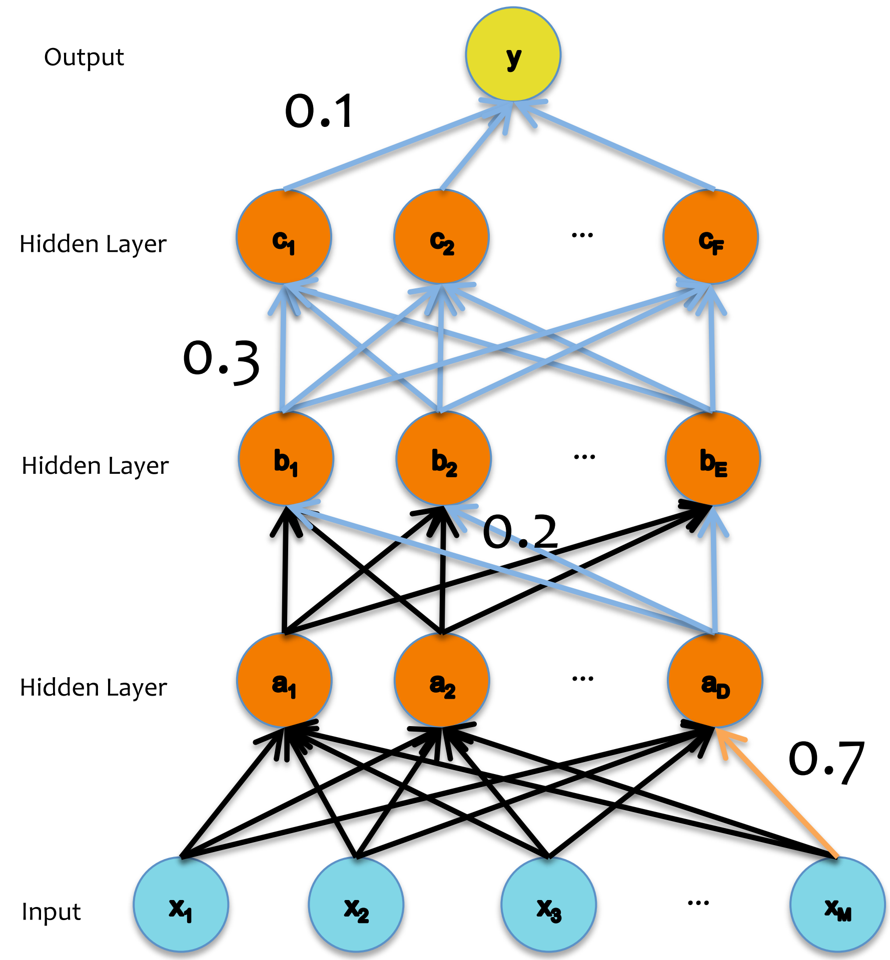


Training

Problem B: *Vanishing Gradients*

The gradient for an edge at the base of the network depends on the gradients of many edges above it

The chain rule multiplies many of these partial derivatives together



Training

Idea #1: No pre-training

- **Idea #1: (Just like a shallow network)**
  - Compute the supervised gradient by backpropagation.
  - Take small steps in the direction of the gradient (SGD)

- What goes wrong?
  A. Gets stuck in local optima
     - Nonconvex objective
     - Usually start at a random (bad) point in parameter space

  B. Gradient is progressively getting more dilute
     - "Vanishing gradients"

Training

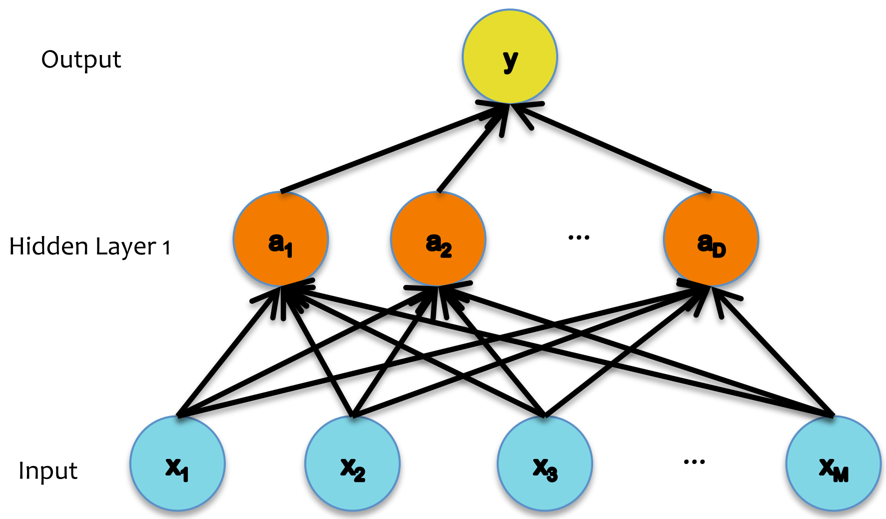
Idea #2: Supervised Pre-training

- **Idea #2: (Two Steps)**
  - **Train each level** of the model in a **greedy** way
  - Then use our **original idea**

1. Supervised Pre-training
   – Use **labeled** data
   – Work bottom-up
     - Train hidden layer 1. Then fix its parameters.
     - Train hidden layer 2. Then fix its parameters.
     - …
     - Train hidden layer n. Then fix its parameters.
2. Supervised Fine-tuning
   – Use **labeled** data to train following "Idea #1"
   – Refine the features by backpropagation so that they become tuned to the end-task

Idea #2: Supervised Pre-training

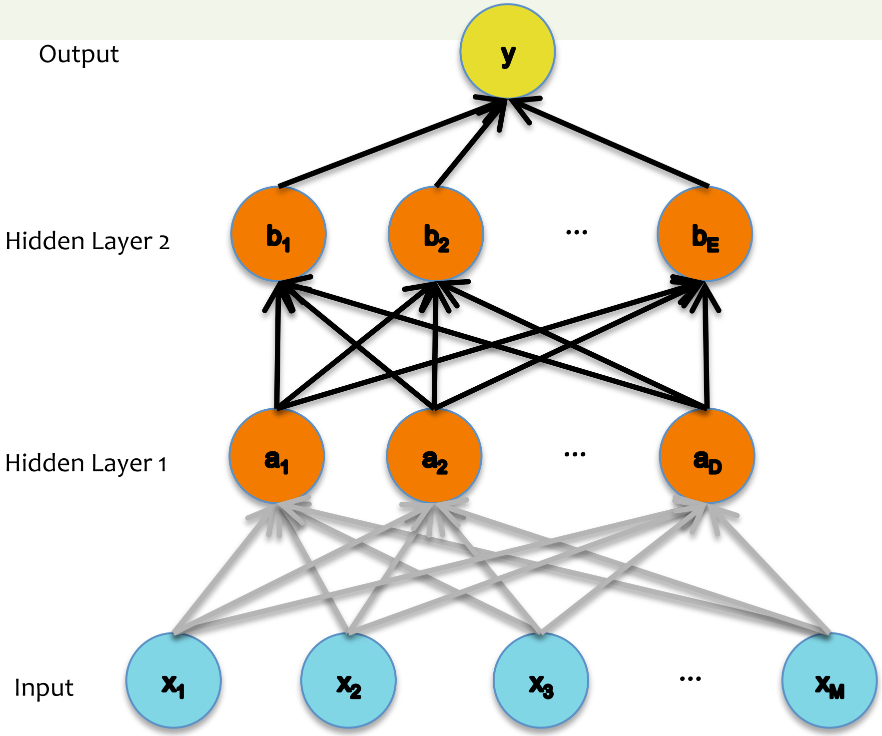
- **Idea #2: (Two Steps)**
  - **Train each level** of the model in a **greedy** way
  - Then use our **original idea**



Training

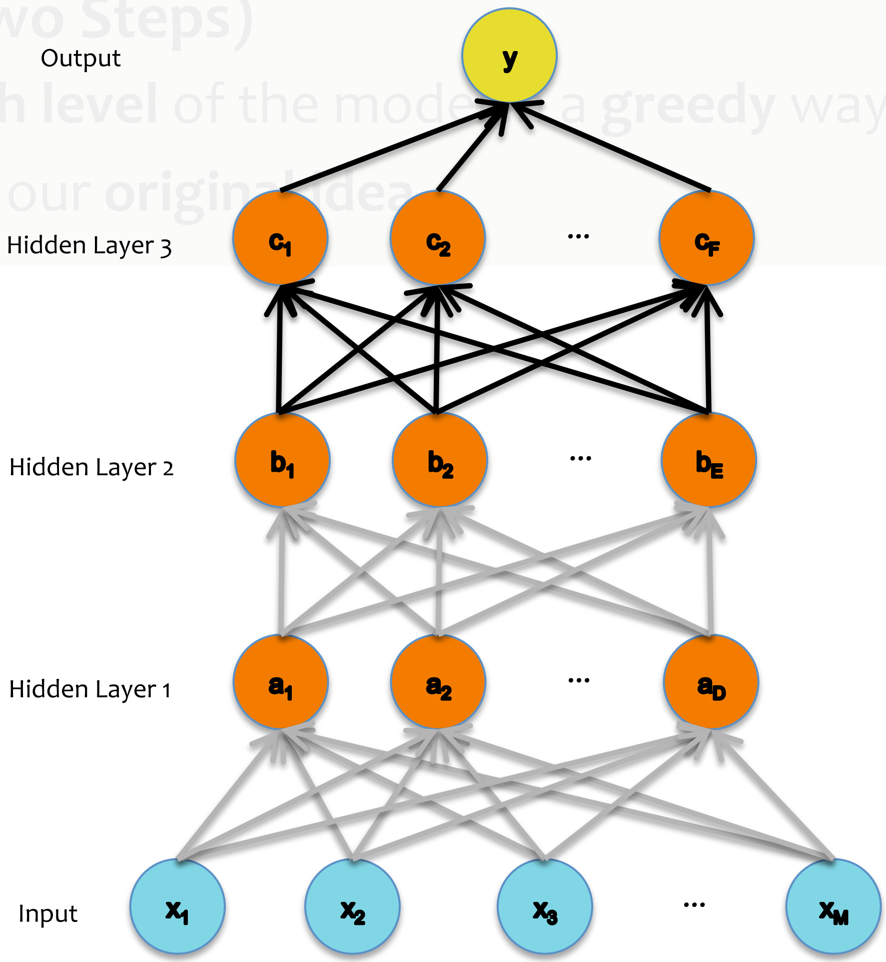
Idea #2: Supervised Pre-training

- Idea #2: (Two Steps)
  - **Train each level** of the model in a **greedy** way
  - Then use our **original idea**



Idea #2: Supervised Pre-training

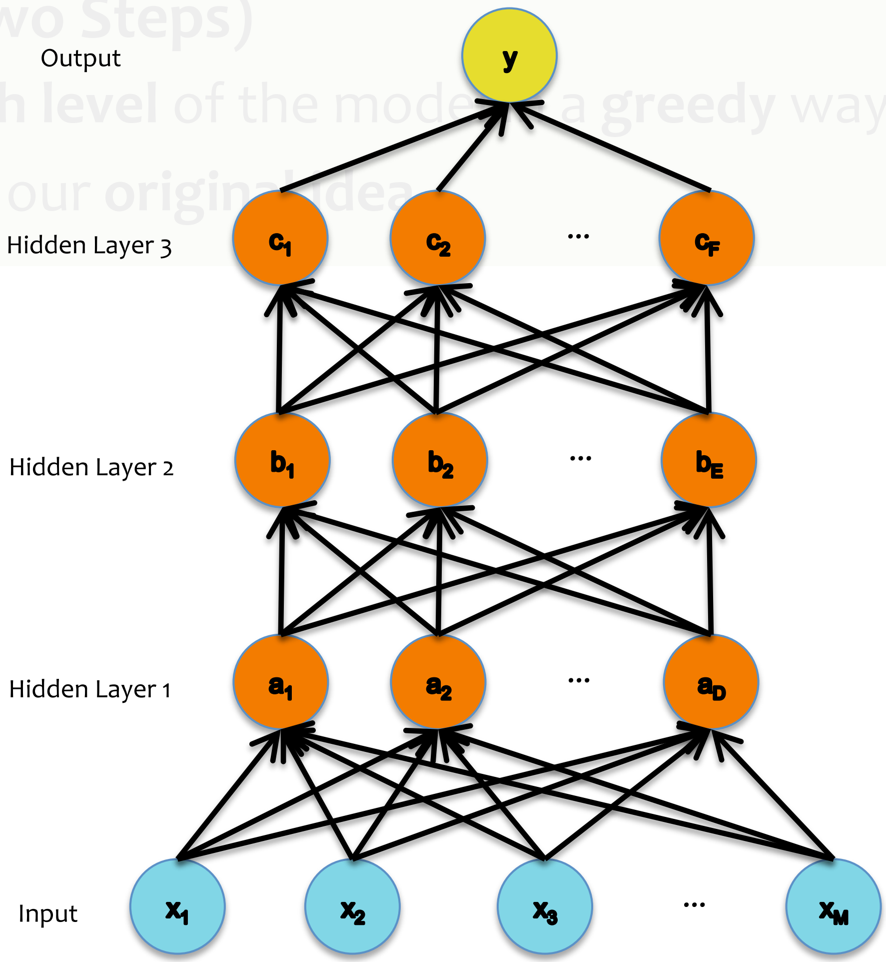
- Idea #2: (Two Steps)
  - Train each level of the model in a greedy way
  - Then use our original idea



Training

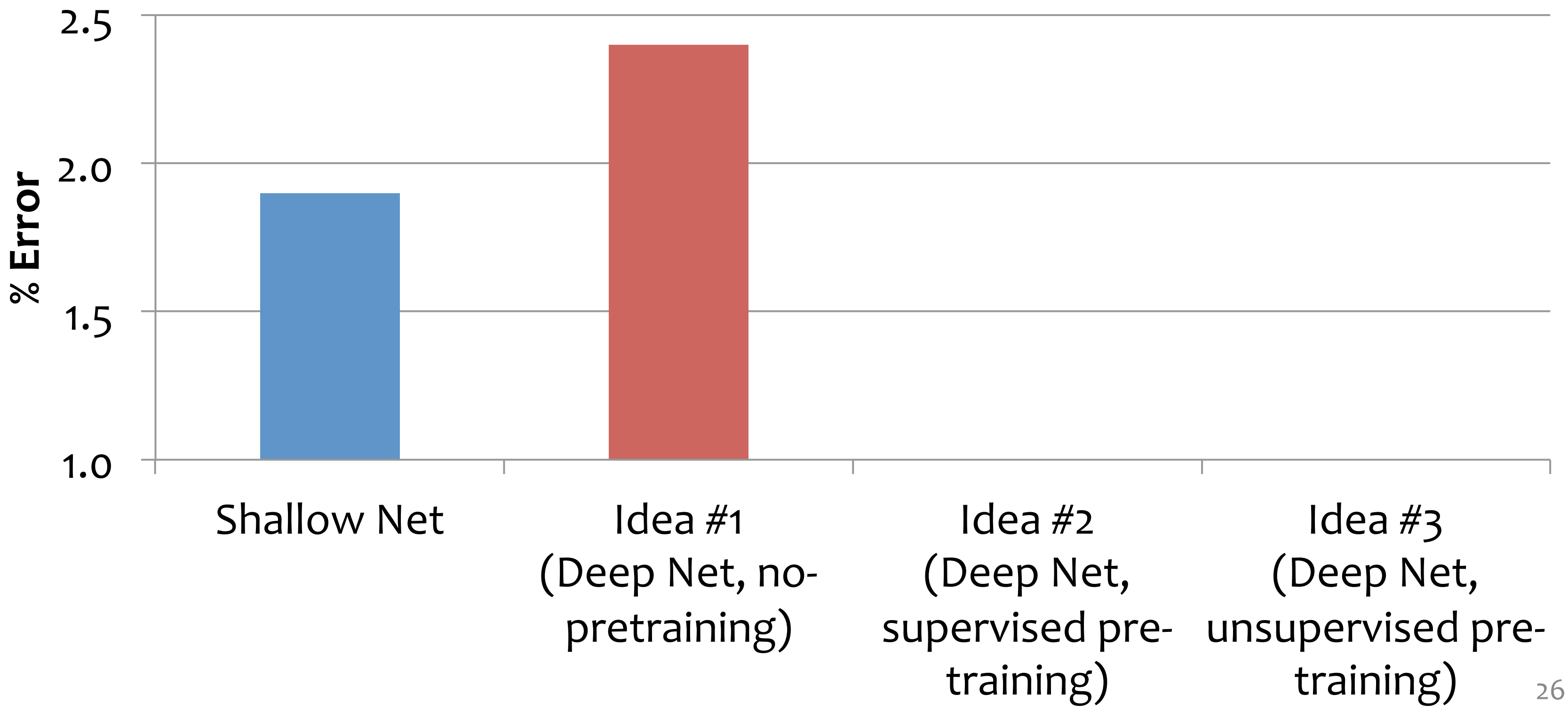
Idea #2: Supervised Pre-training

- Idea #2: (Two Steps)
  - Train each level of the model in a greedy way
  - Then use our original idea



Comparison on MNIST

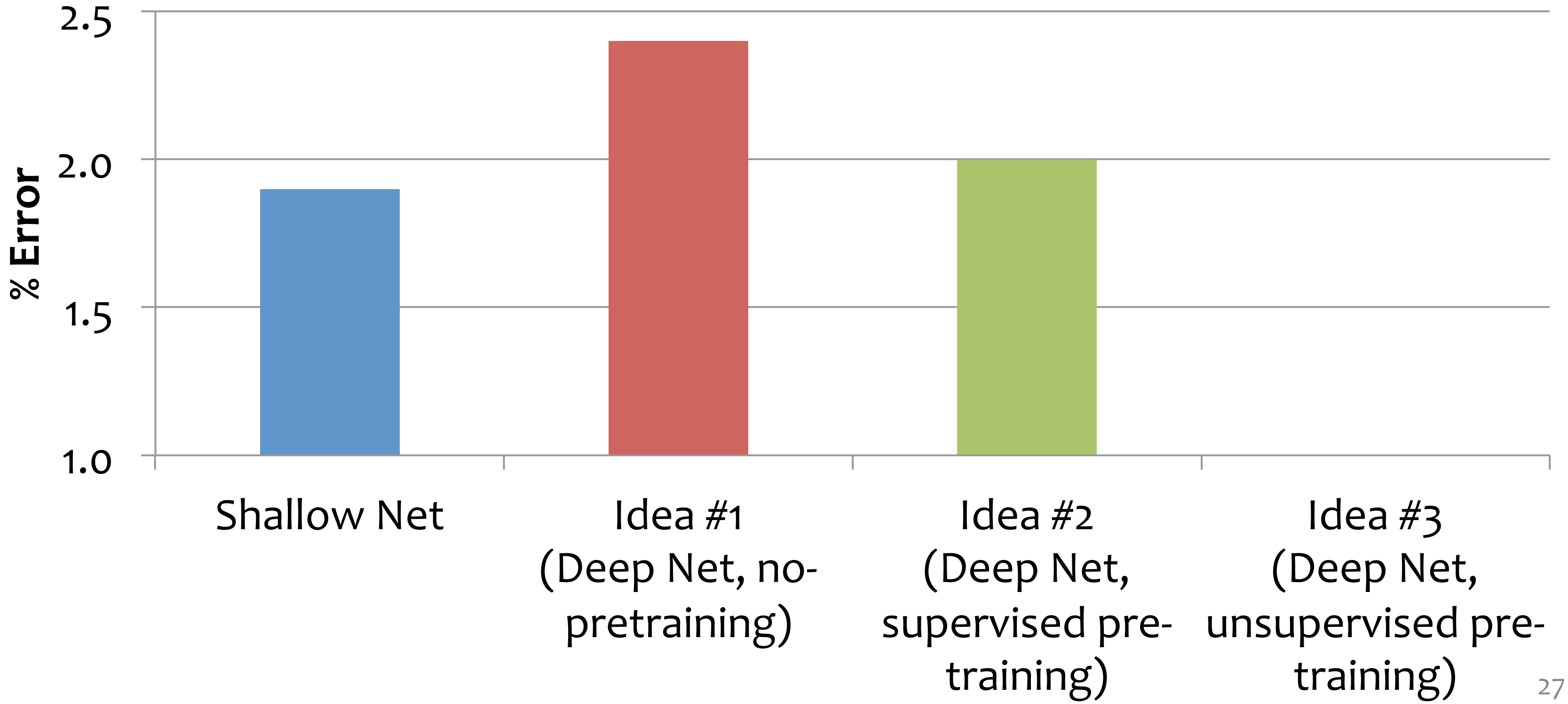
- Results from Bengio et al. (2006) on MNIST digit classification task
- Percent error (lower is better)



Training

Comparison on MNIST

- Results from Bengio et al. (2006) on MNIST digit classification task
- Percent error (lower is better)



Training

Idea #3: Unsupervised Pre-training

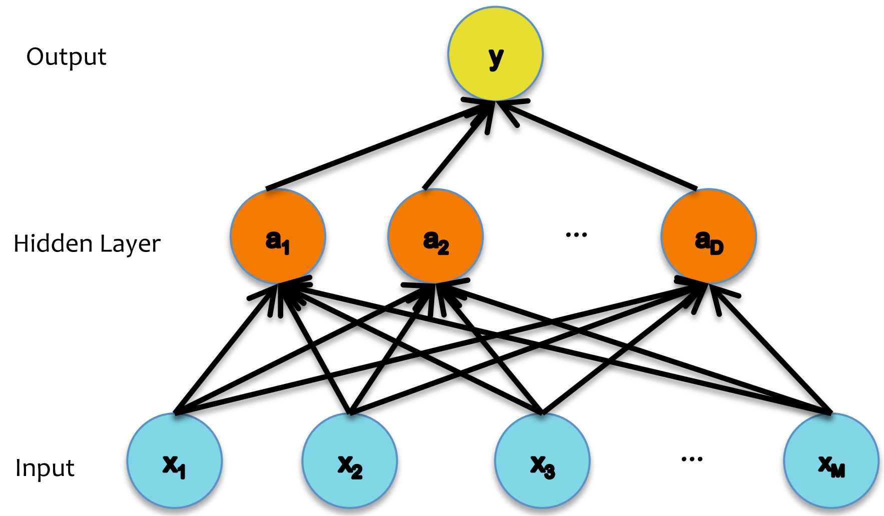
- **Idea #3: (Two Steps)**
  - Use our original idea, but **pick a better starting point**
  - **Train each level** of the model in a **greedy** way

1. Unsupervised Pre-training
   - Use **unlabeled** data
   - Work bottom-up
     - Train hidden layer 1. Then fix its parameters.
     - Train hidden layer 2. Then fix its parameters.
     - …
     - Train hidden layer n. Then fix its parameters.
2. Supervised Fine-tuning
   - Use **labeled** data to train following “Idea #1”
   - Refine the features by backpropagation so that they become tuned to the end-task

The solution: *Unsupervised pre-training*

Unsupervised pre-training of the first layer:

- What should it predict?
- What else do we observe?
- **The input!**

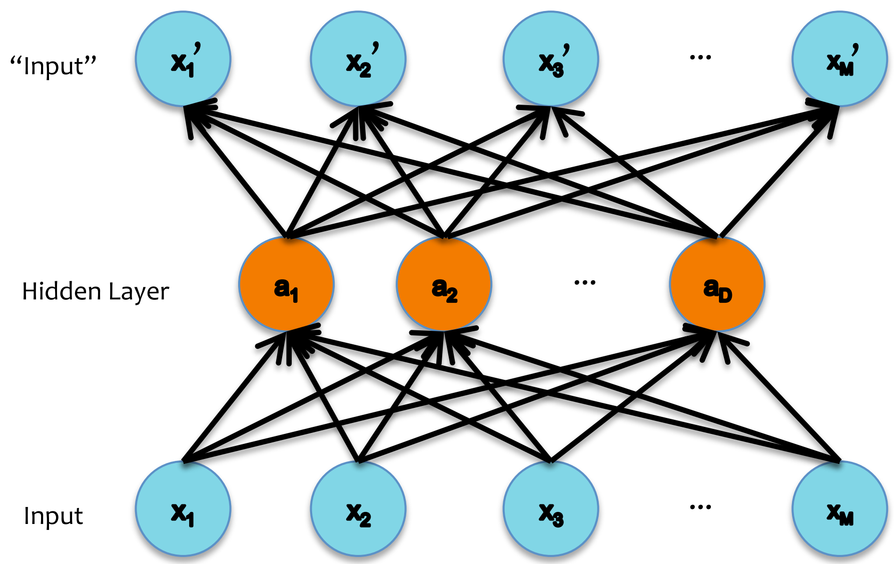


The solution:
Unsupervised pre-training

Unsupervised pre-training of the first layer:

- What should it predict?
- What else do we observe?
- **The input!**

This topology defines an Auto-encoder.



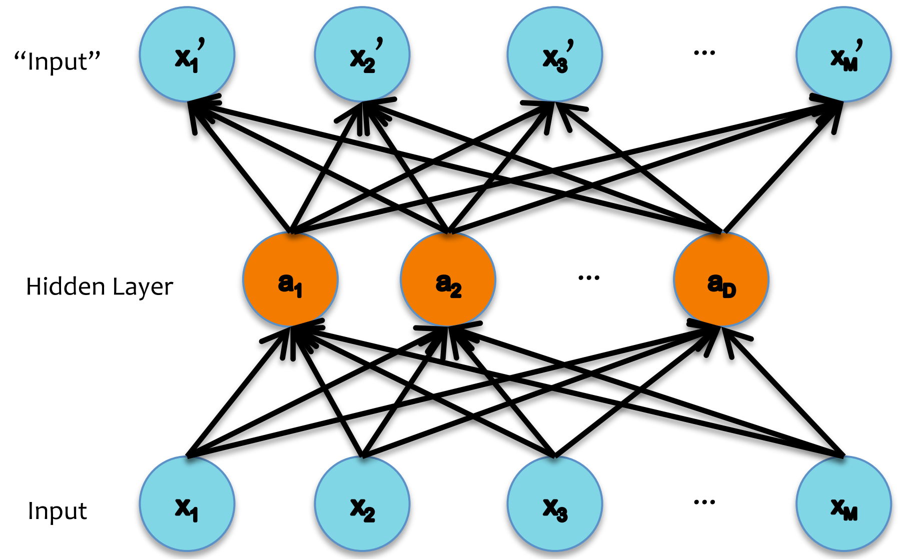
Auto-Encoders

Key idea: Encourage z to give small reconstruction error:

- x’ is the *reconstruction* of x
- Loss = || x – DECODER(ENCODER(x)) ||²
- Train with the same backpropagation algorithm for 2-layer Neural Networks with x_m as both input and output.

DECODER: x’ = h(W’z)

ENCODER: z = h(Wx)

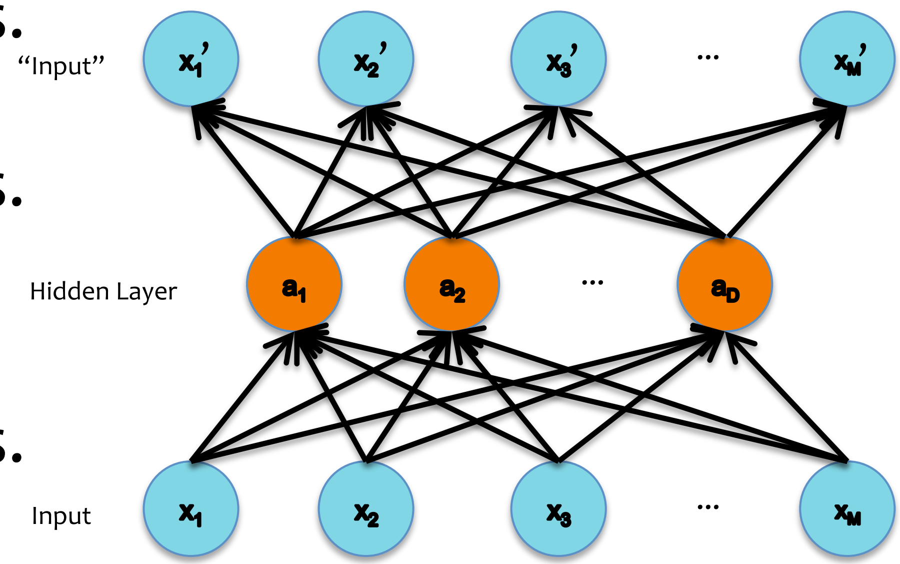


Slide adapted from Raman Arora

The solution: *Unsupervised pre-training*

Unsupervised pre-training

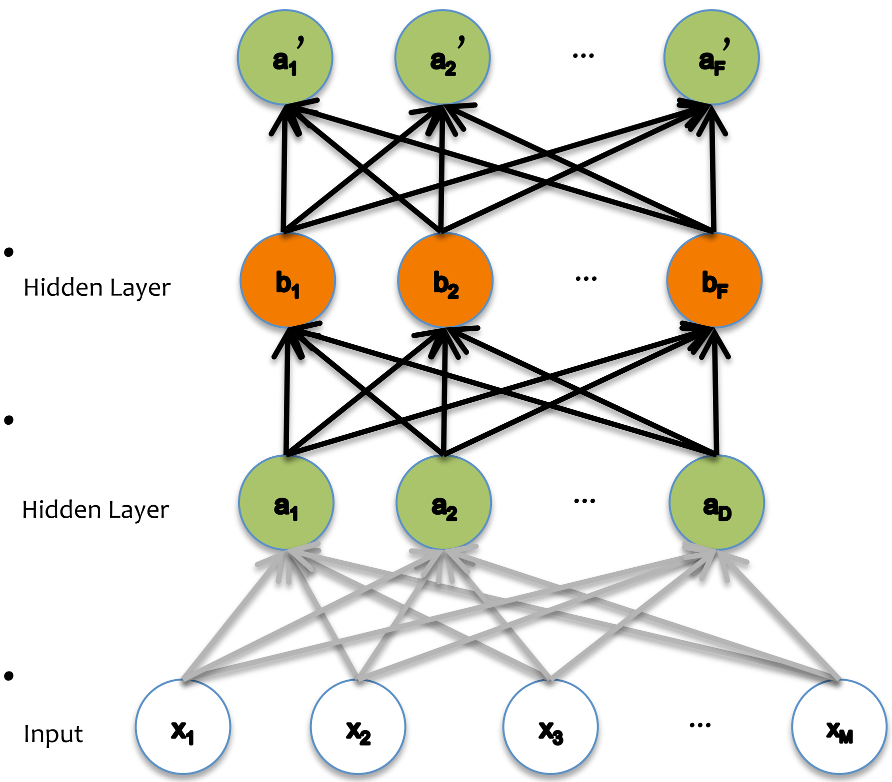
- Work bottom-up
  - Train hidden layer 1. Then fix its parameters.
  - Train hidden layer 2. Then fix its parameters.
  - …
  - Train hidden layer n. Then fix its parameters.



The solution: *Unsupervised pre-training*

Unsupervised pre-training

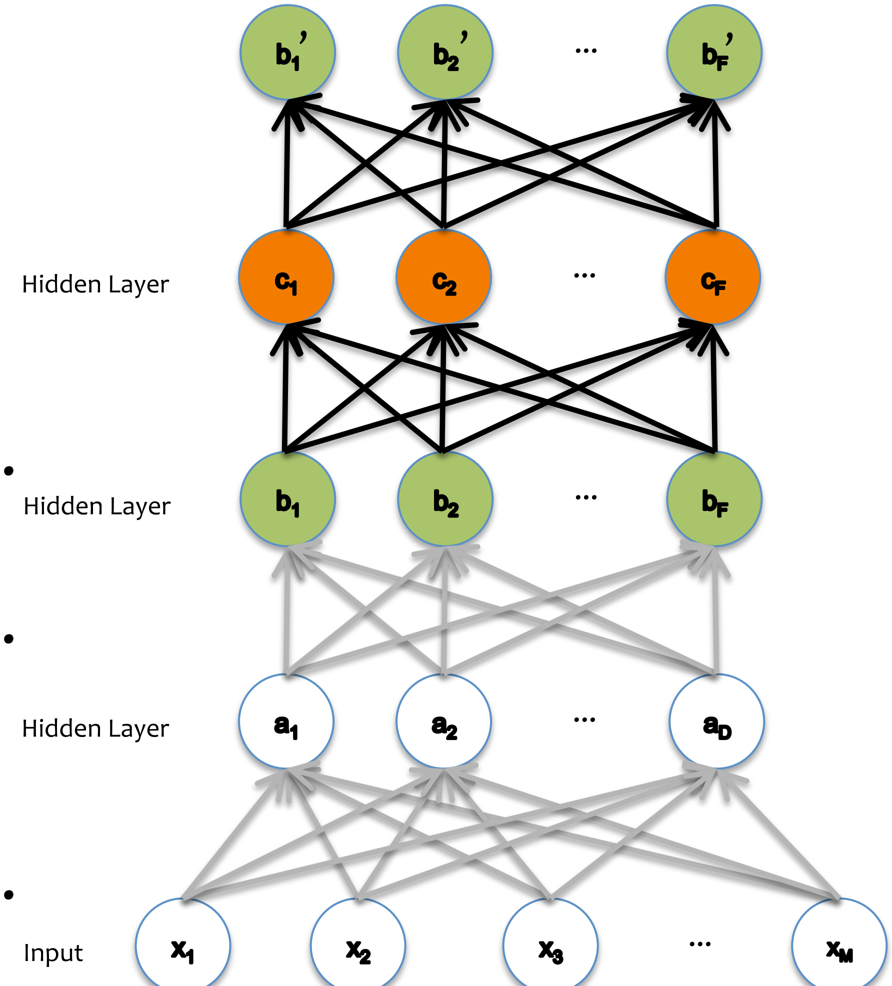
- Work bottom-up
  - Train hidden layer 1. Then fix its parameters.
  - Train hidden layer 2. Then fix its parameters.
  - …
  - Train hidden layer n. Then fix its parameters.



The solution: *Unsupervised pre-training*

Unsupervised pre-training

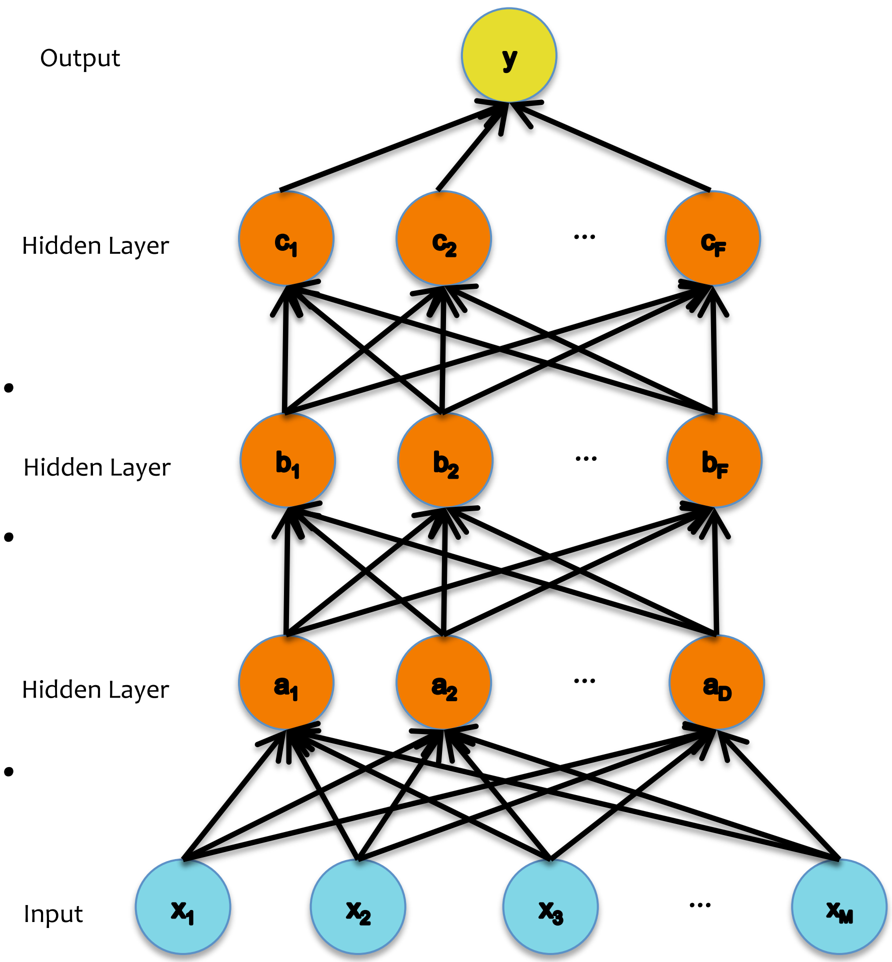
- Work bottom-up
  - Train hidden layer 1. Then fix its parameters.
  - Train hidden layer 2. Then fix its parameters.
  - …
  - Train hidden layer n. Then fix its parameters.



The solution: *Unsupervised pre-training*

Unsupervised pre-training

- Work bottom-up
  - Train hidden layer 1. Then fix its parameters.
  - Train hidden layer 2. Then fix its parameters.
  - …
  - Train hidden layer n. Then fix its parameters.



Supervised fine-tuning

Backprop and update all parameters

Deep Network Training

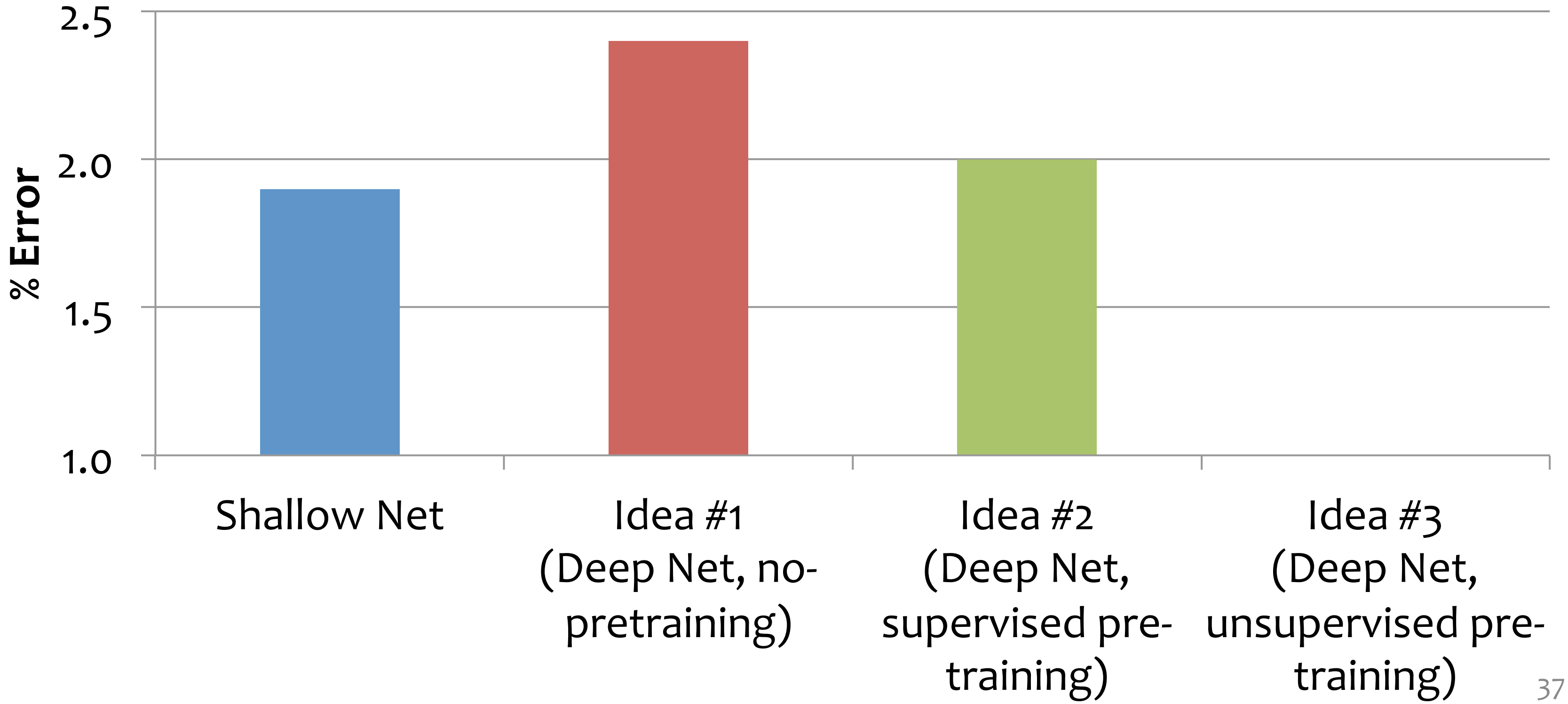
- **Idea #1:**
  1. Supervised fine-tuning only

- **Idea #2:**
  1. Supervised layer-wise pre-training
  2. Supervised fine-tuning

- **Idea #3:**
  1. Unsupervised layer-wise pre-training
  2. Supervised fine-tuning

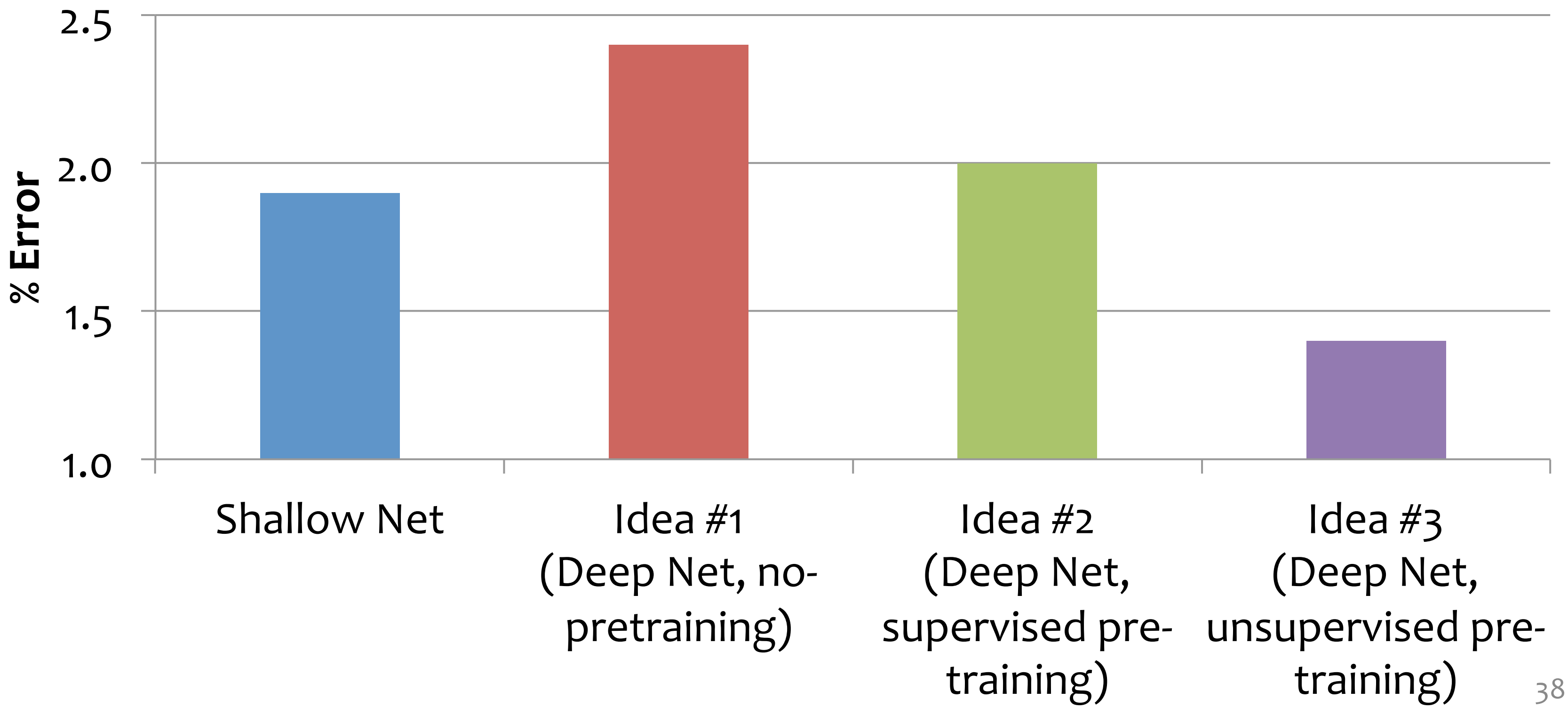
Comparison on MNIST

- Results from Bengio et al. (2006) on MNIST digit classification task
- Percent error (lower is better)



Comparison on MNIST

- Results from Bengio et al. (2006) on MNIST digit classification task
- Percent error (lower is better)



Training

Is layer-wise pre-training always necessary?

In 2010, **a record on a hand-writing recognition task was set by standard supervised backpropagation (our Idea #1).**

How? **A very fast implementation on GPUs.**

See Ciresen et al. (2010)

Deep Learning

- Goal: learn features at different levels of abstraction
- Training can be tricky due to…
  - Nonconvexity
  - Vanishing gradients
- Unsupervised layer-wise pre-training can help with both!