MACHINE LEARNING FOR SIGNAL PROCESSING

26-2-2025

Sriram Ganapathy
LEAP lab, Electrical Engineering, Indian Institute of Science,
sriramg@iisc.ac.in

Viveka Salinamakki, Varada R.
LEAP lab, Electrical Engineering, Indian Institute of Science

http://leap.ee.iisc.ac.in/sriram/teaching/MLSP25/

STORY SO FAR

EM algorithm

Generative Modeling

Gaussian Modeling

Gaussian Mixture Modeling

Decision Theory

Classification Problem

Function Modeling

Linear Models for Regression and Classification

Kernel Machines & Max-margin classifiers

Support Vector Machines

Data Representations PCA, LDA

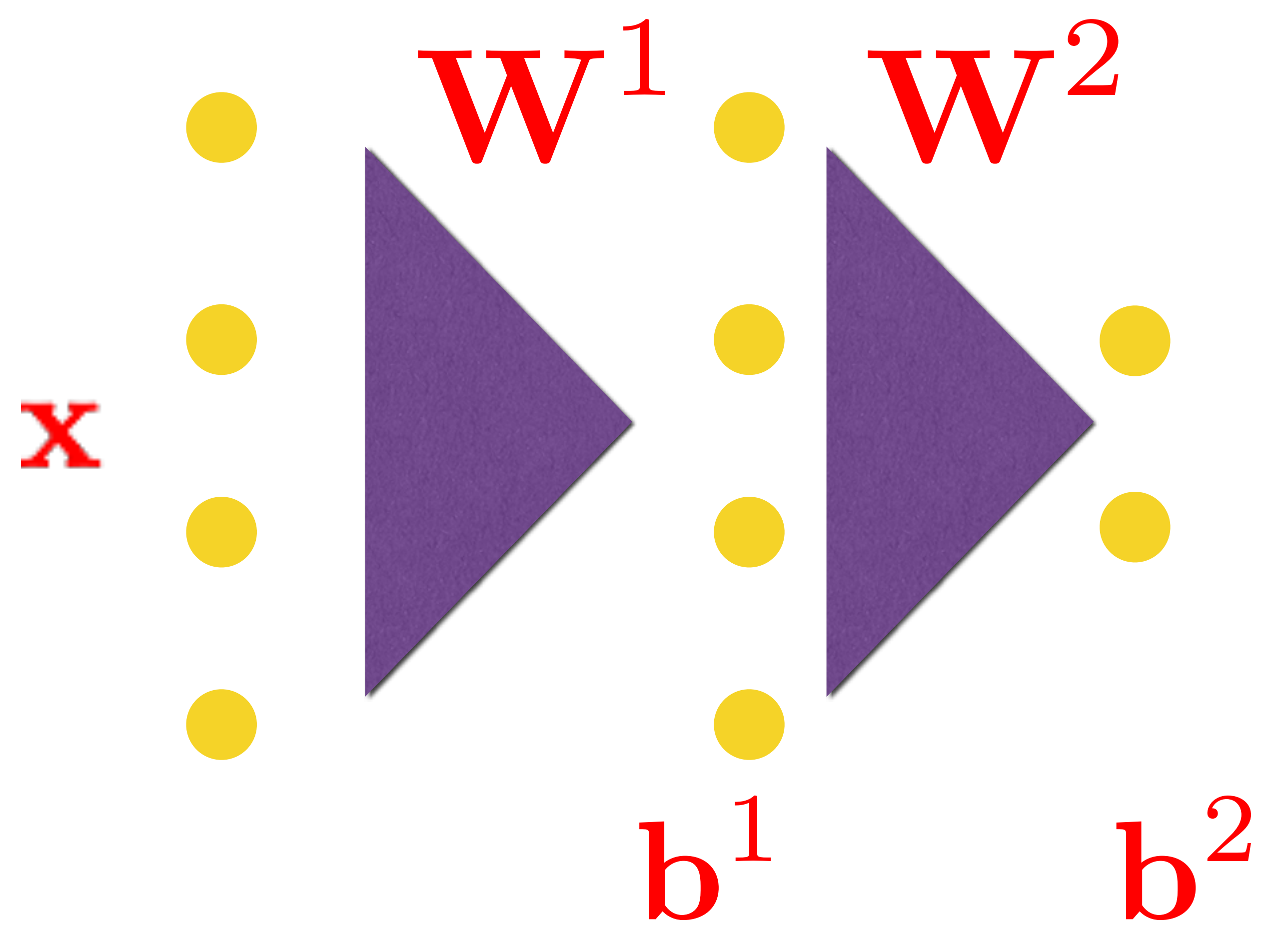
Discriminative Modeling

Neural Networks

Gradient Descent

Neural Network - 1- Hidden Layer

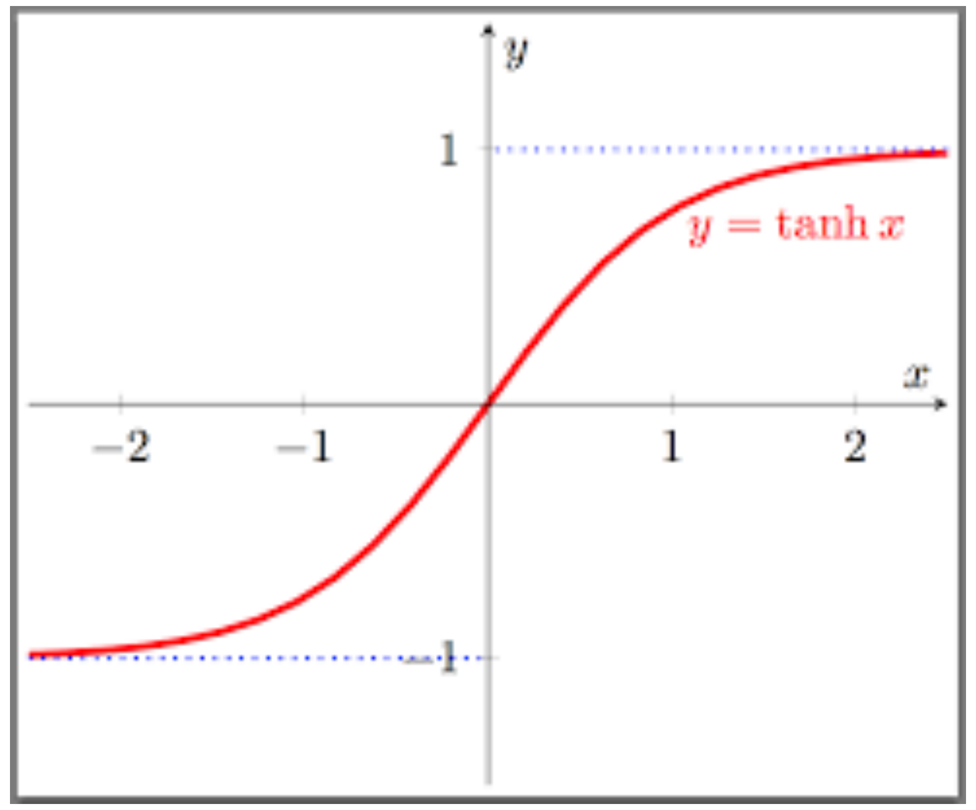
- Has more capacity than logistic regression
  - can learn non-linear data separation functions
  - both 2-class and K-class classification possible
  - can be learnt using gradient descent



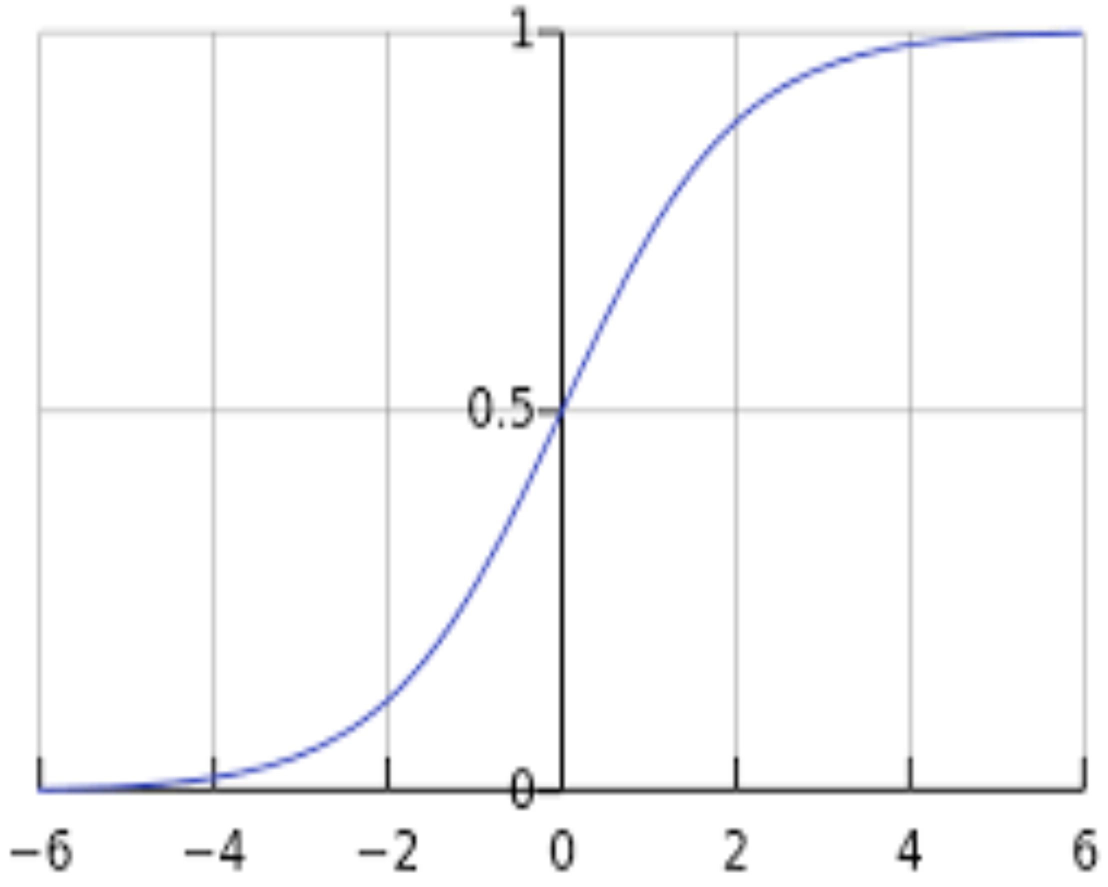
TYPES OF NON-LINEARITIES

Non-linearity in hidden layer

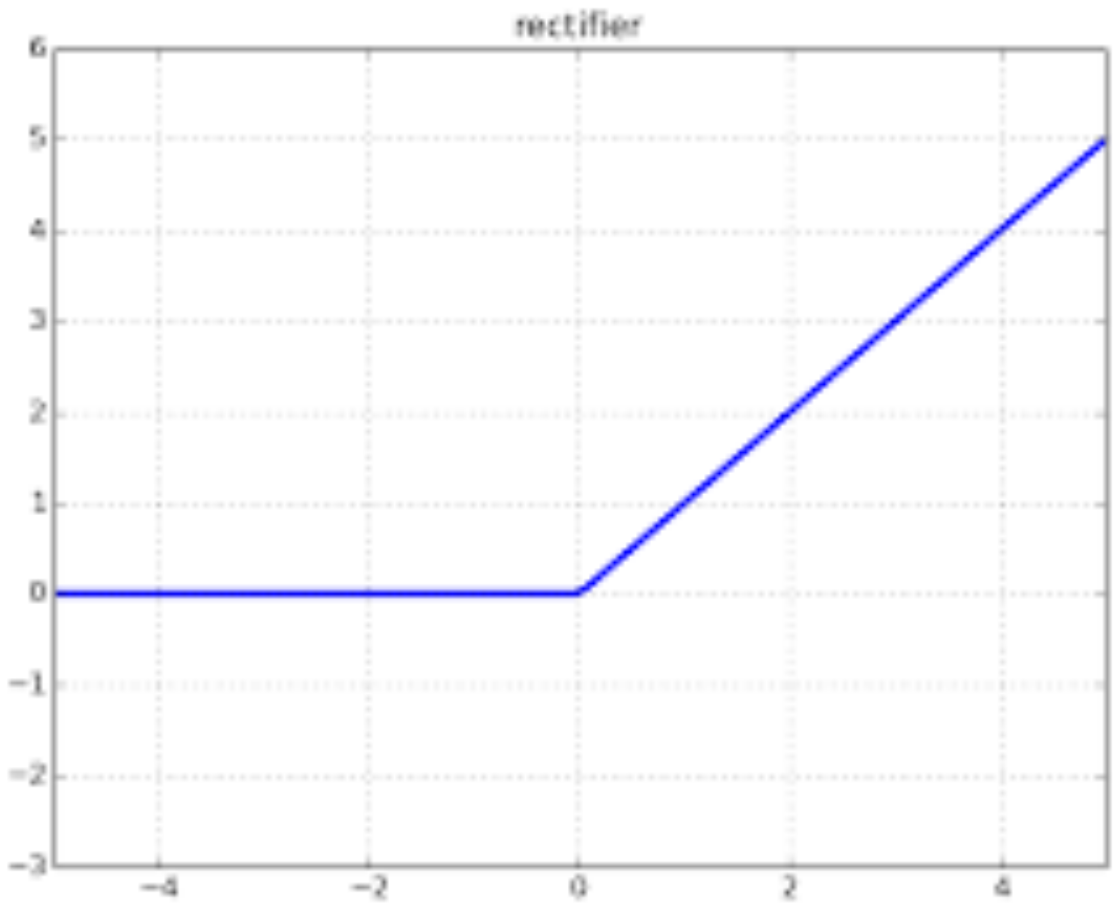
tanh



sigmoid



ReLu



OUTPUT LAYER NON-LINEARITY AND COST FUNCTIONS

- Using a softmax non-linearity
  - error function is cross entropy

$$E_{CE} = -\sum_n \sum_k t_{nk} \, log(v_{nk})$$

- For regression style tasks - output is linear
  - error function is mean square error

$$E_{MSE} = -\sum_n \sum_k (t_{nk} - v_{nk})^2$$

FORWARD THROUGH THE MODEL PROPAGATION LEARNING

- Computations in the forward direction

$$\mathbf{a}^1 = \mathbf{W}^1\mathbf{x} + \mathbf{b}^1$$
$$\mathbf{z}^1 = \sigma(\mathbf{a}^1)$$
$$\mathbf{a}^2 = \mathbf{W}^2\mathbf{z}^1 + \mathbf{b}^2$$
$$\mathbf{y} = softmax(\mathbf{a}^2)$$

- Loss function

$$E_{CE} = -\sum_n \sum_k t_{nk}\ log(v_{nk})$$

$$\mathbf{\Theta} = \{\mathbf{W}^1\ ,\ \mathbf{b}^1\ ,\ \mathbf{W}^2\ ,\ \mathbf{b}^2\}$$

- Parameters in the model

Need to be updated based on the gradients w.r.t. the error

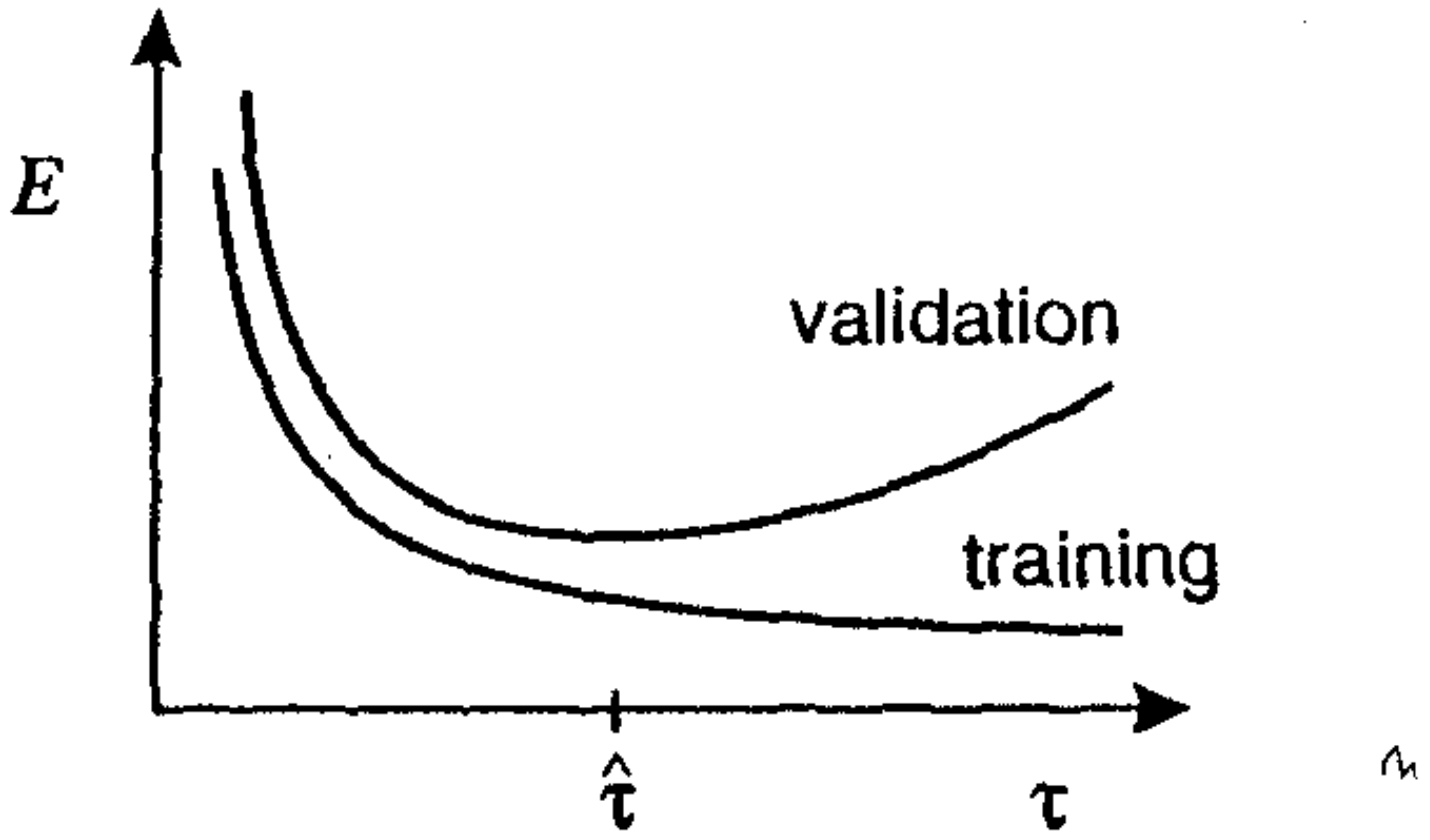
Gradient Computation in the Model

$$\mathbf{a}^1 = \mathbf{W}^1\mathbf{x} + \mathbf{b}^1$$
$$\mathbf{z}^1 = \sigma(\mathbf{a}^1)$$
$$\mathbf{a}^2 = \mathbf{W}^2\mathbf{z}^1 + \mathbf{b}^2$$
$$\mathbf{y} = softmax(\mathbf{a}^2)$$

$$E_{CE} = -\sum_n \sum_k t_{nk}\ log(v_{nk})$$

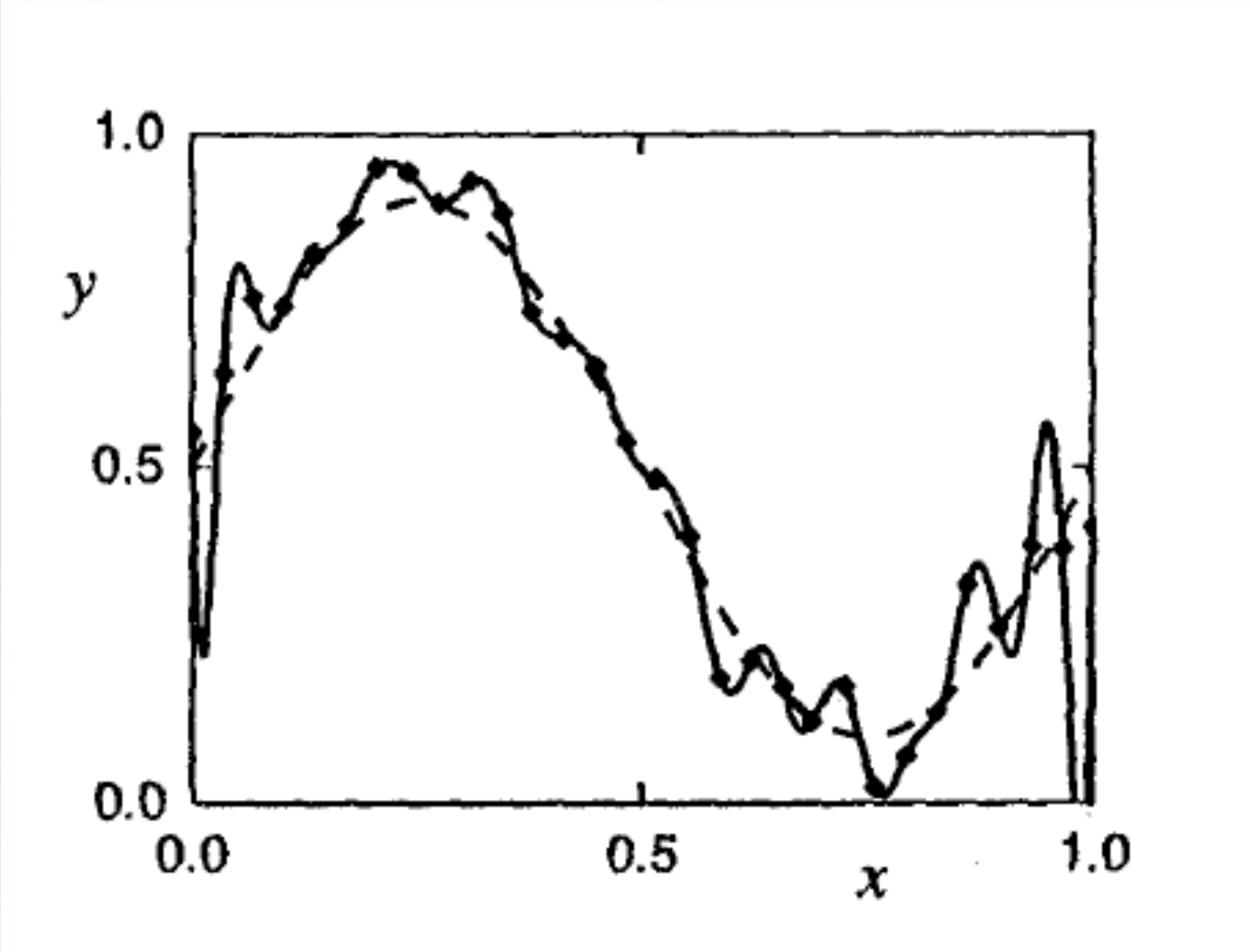
- When computing the gradients
  - Order of computations
    - The derivative of the loss function w.r.t output layer
    - The derivative of the loss function w.r.t output activation
    - The derivative of the loss function w.r.t hidden layer outputs
    - The derivative of the loss function w.r.t. hidden layer activations

REGULARIZATION IN NEURAL NETWORKS

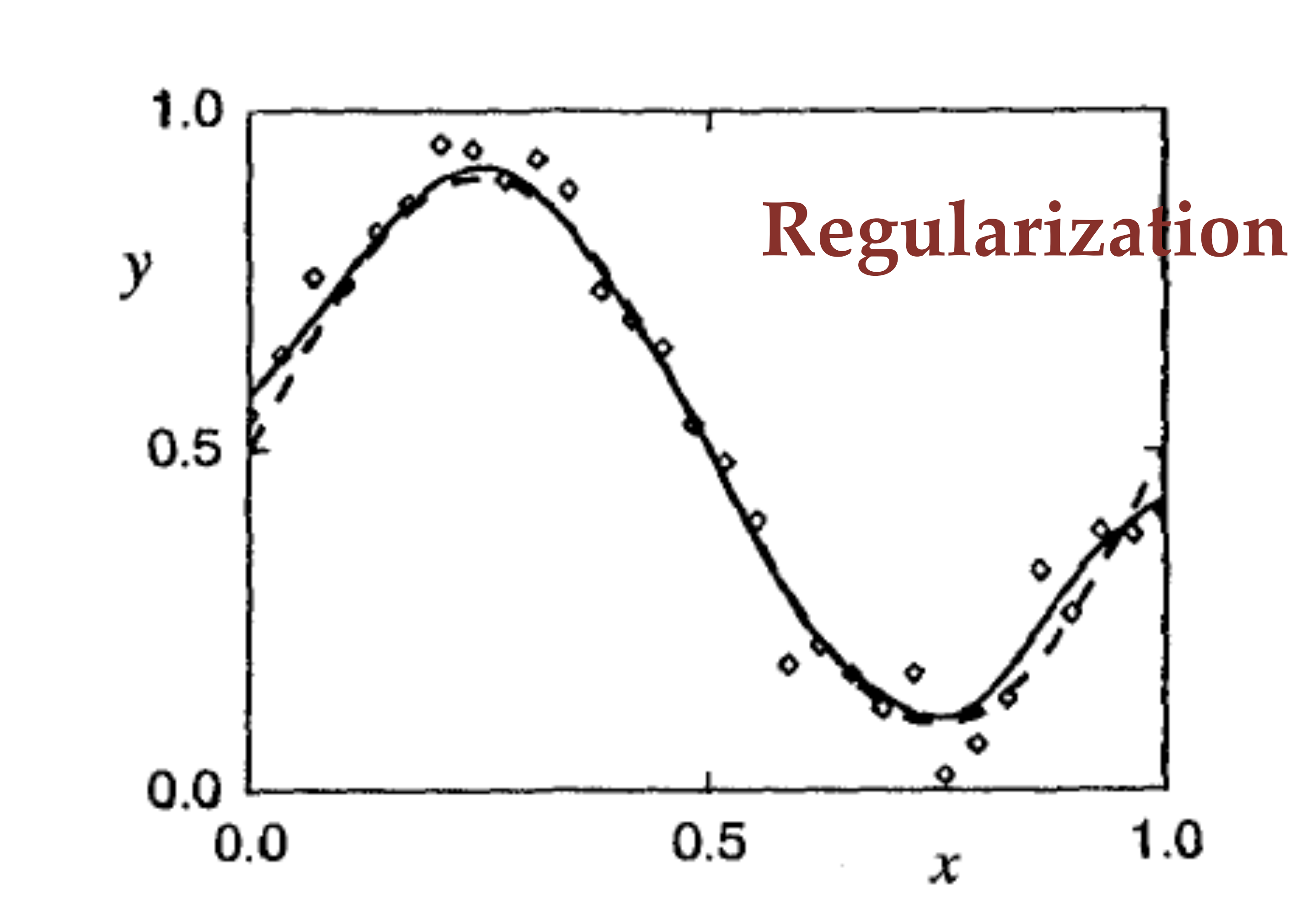


Weight Decay Regularization

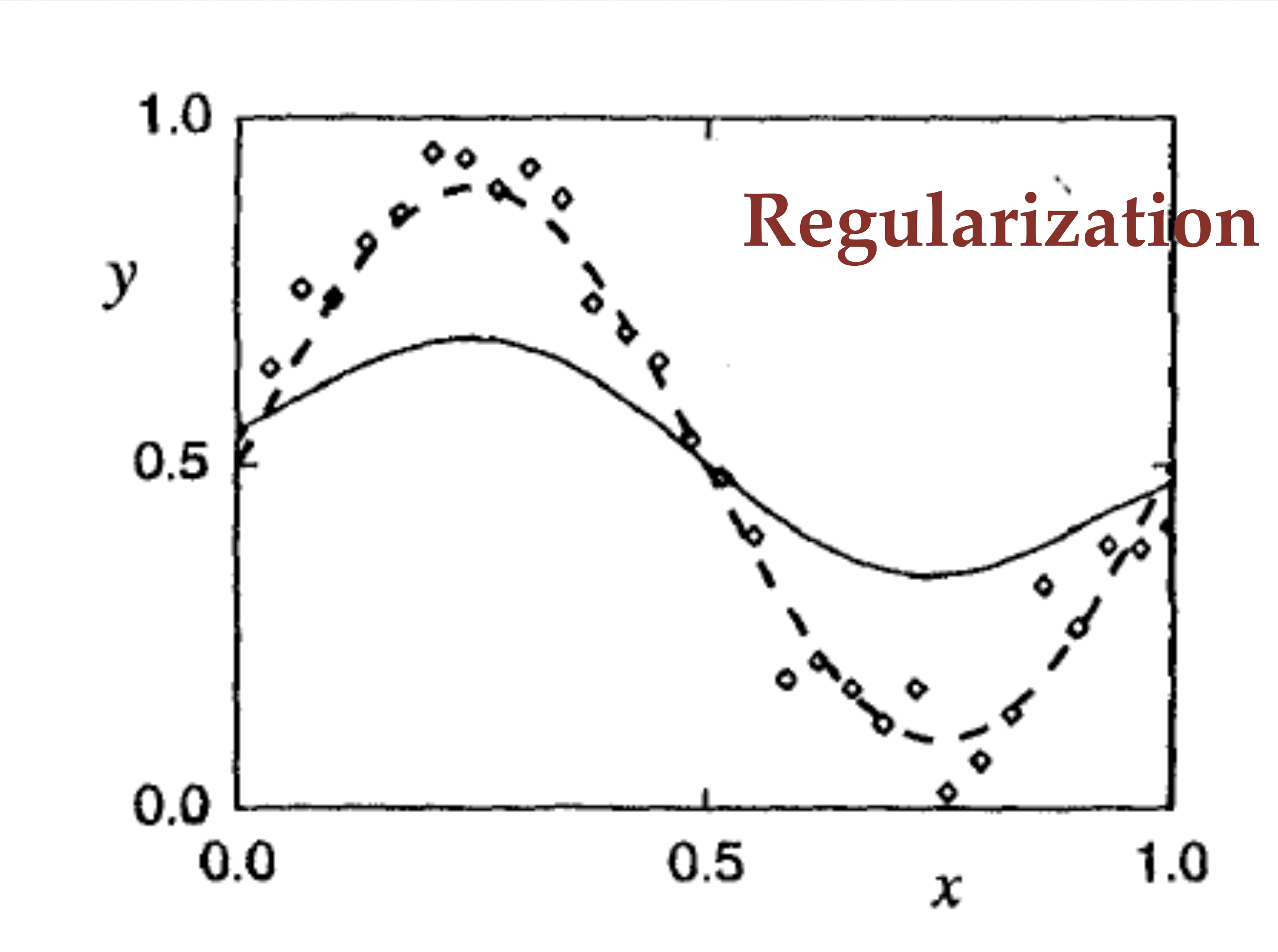
Regularization = 0



Regularization = 40



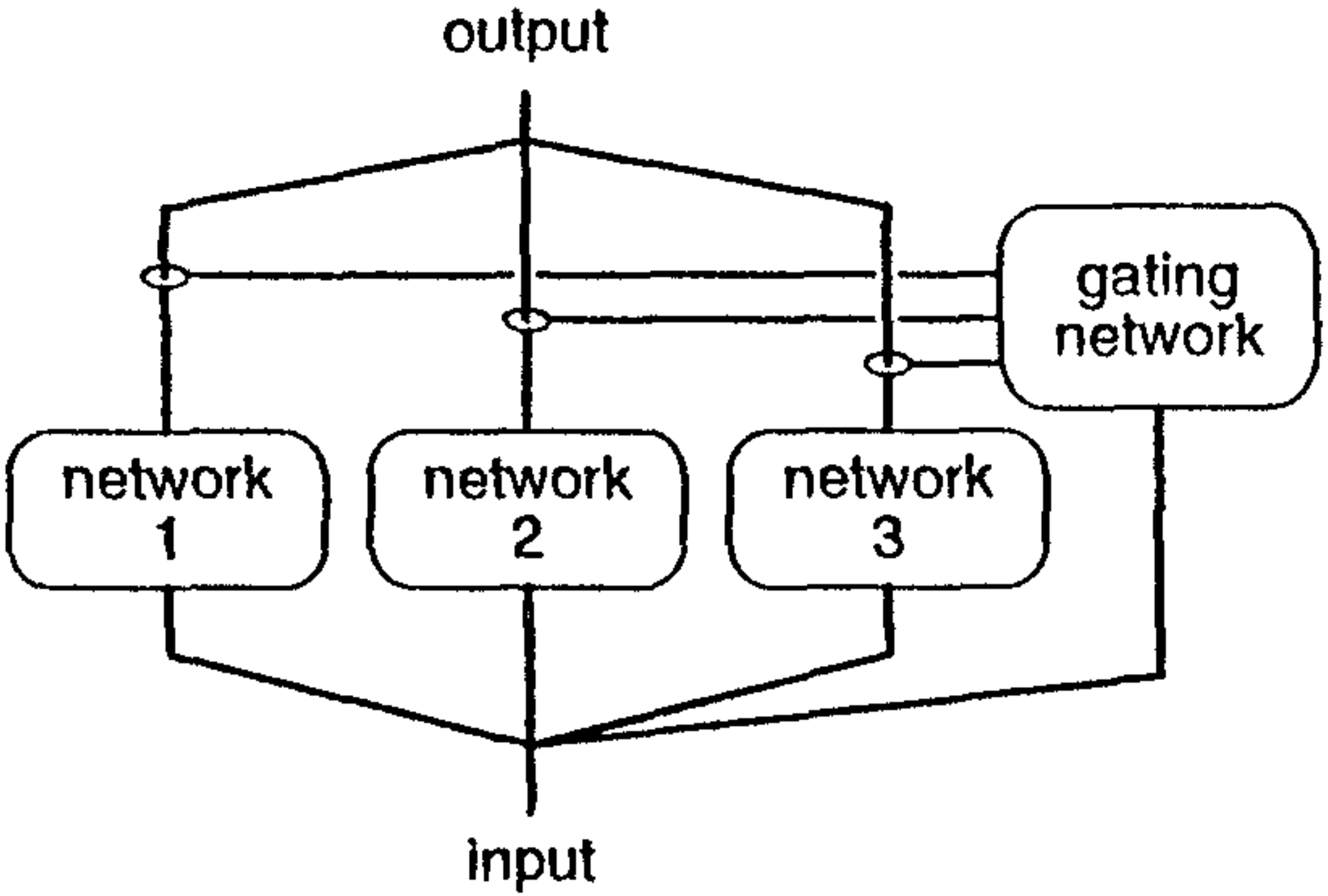
Regularization = 4000



Other Approaches

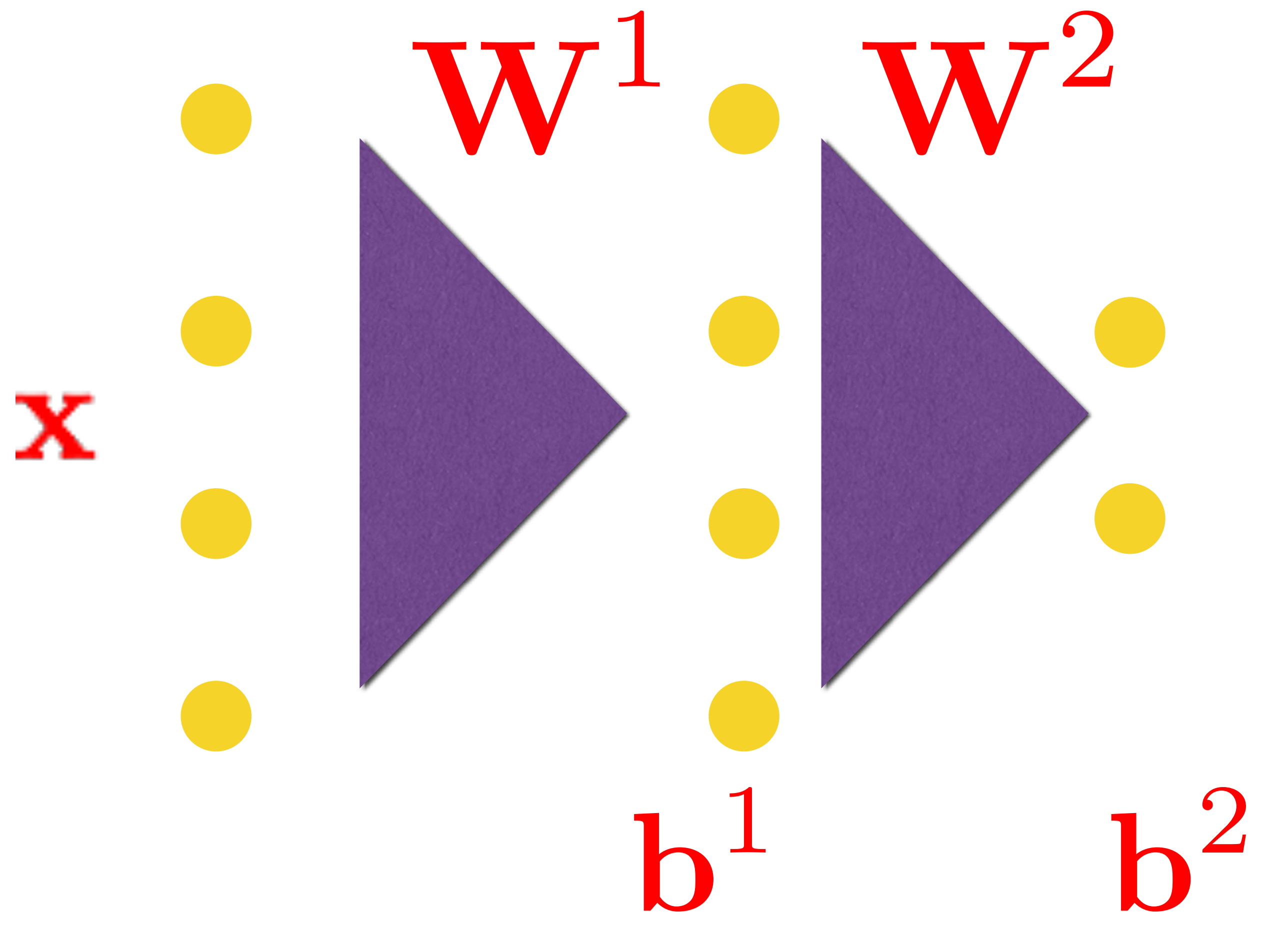
- Training with noise
- Mixture of models
- Mixture of experts approach

MOE

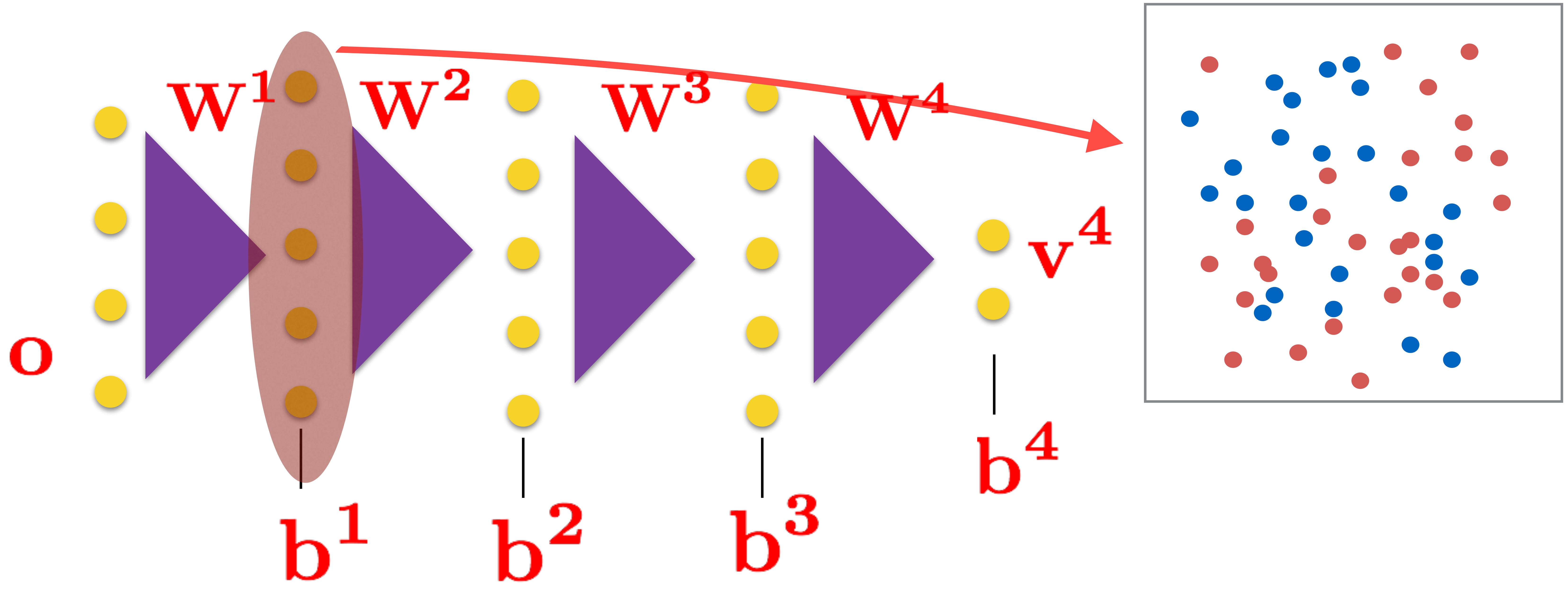


Neural Networks - 1 Hidden Layer

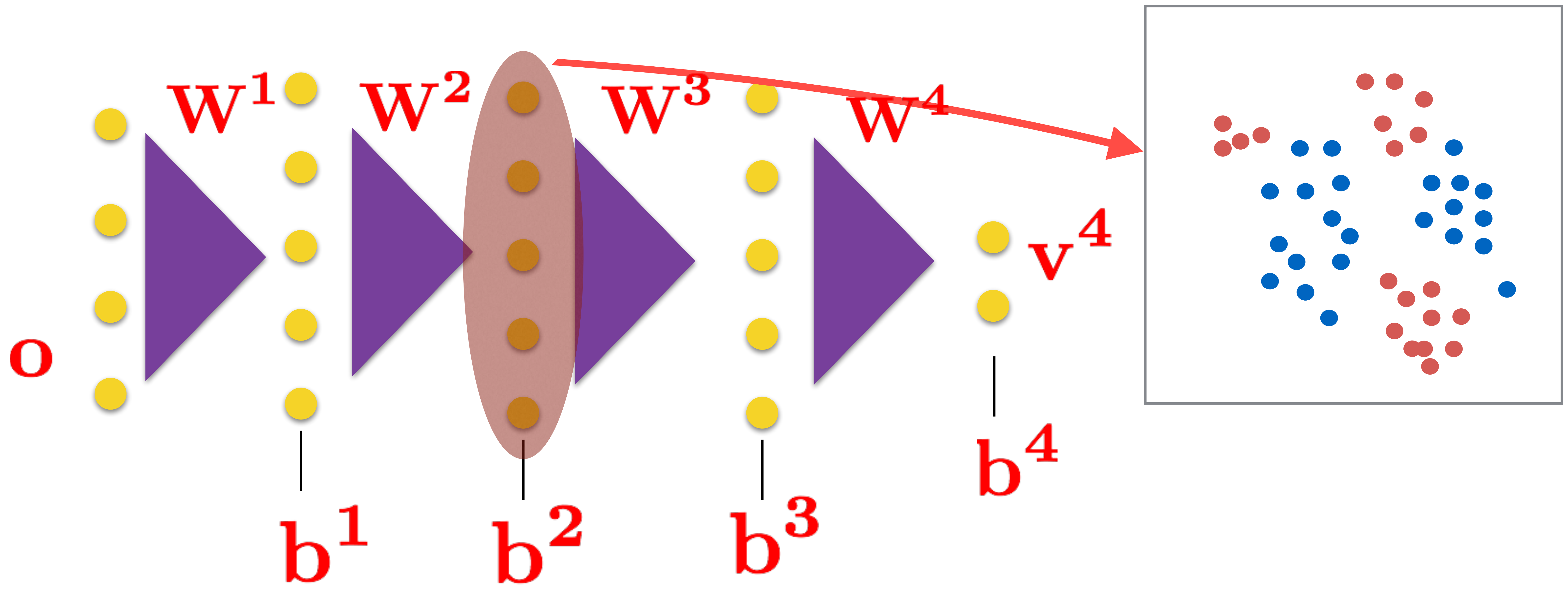
- For complex problems in audio/image/text
  - Single hidden layer may be too restrictive in learning the model parameters
  - May not scale up with availability of big data.



Neural Networks — Deep

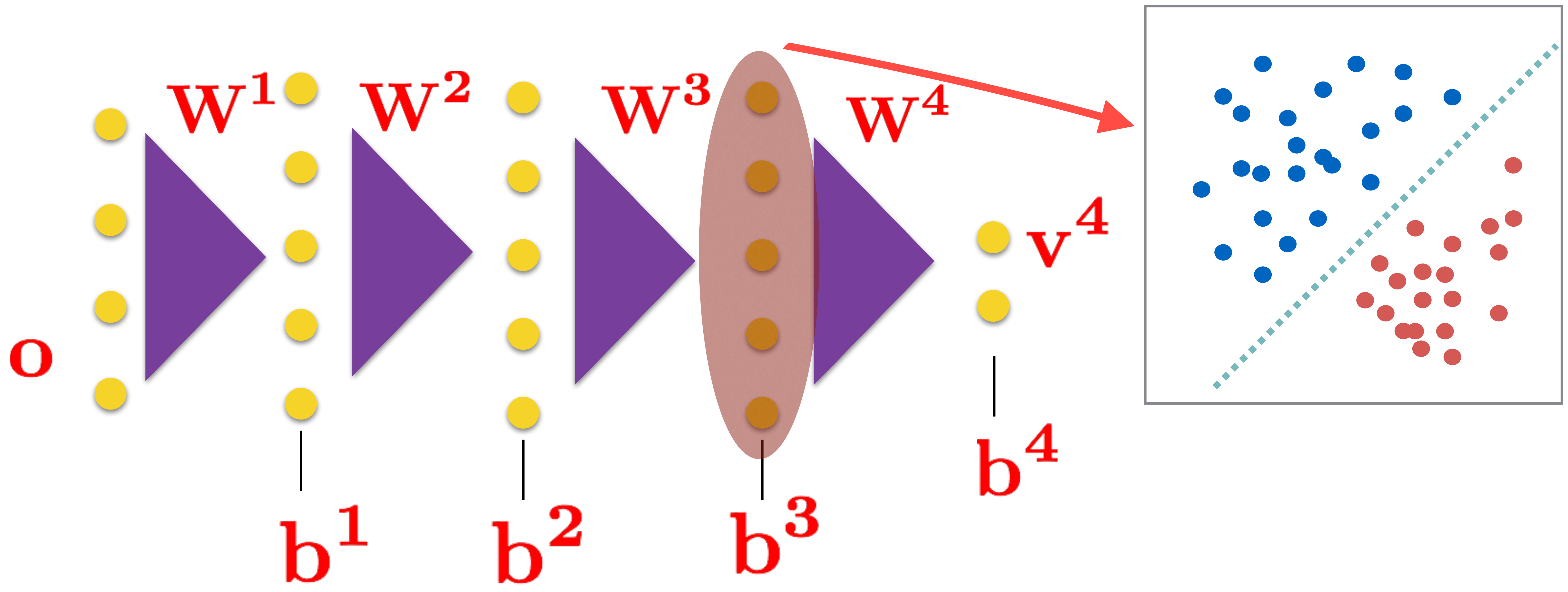


Neural networks with multiple hidden layers - Deep networks



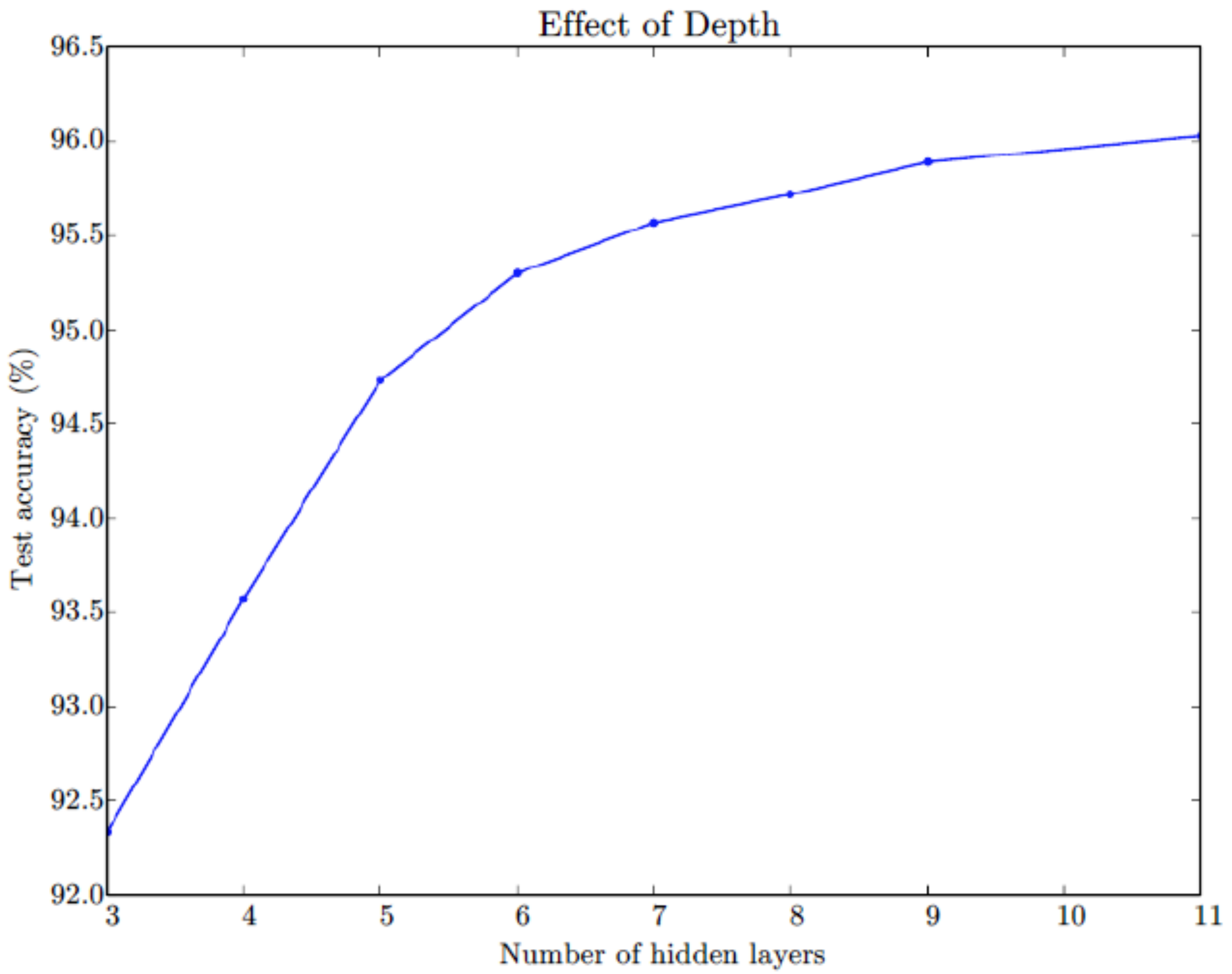
DEEP NEURAL NETWORKS

Neural networks with multiple hidden layers - Deep networks



Deep networks perform hierarchical data abstractions which enable the non-linear separation of complex data samples.

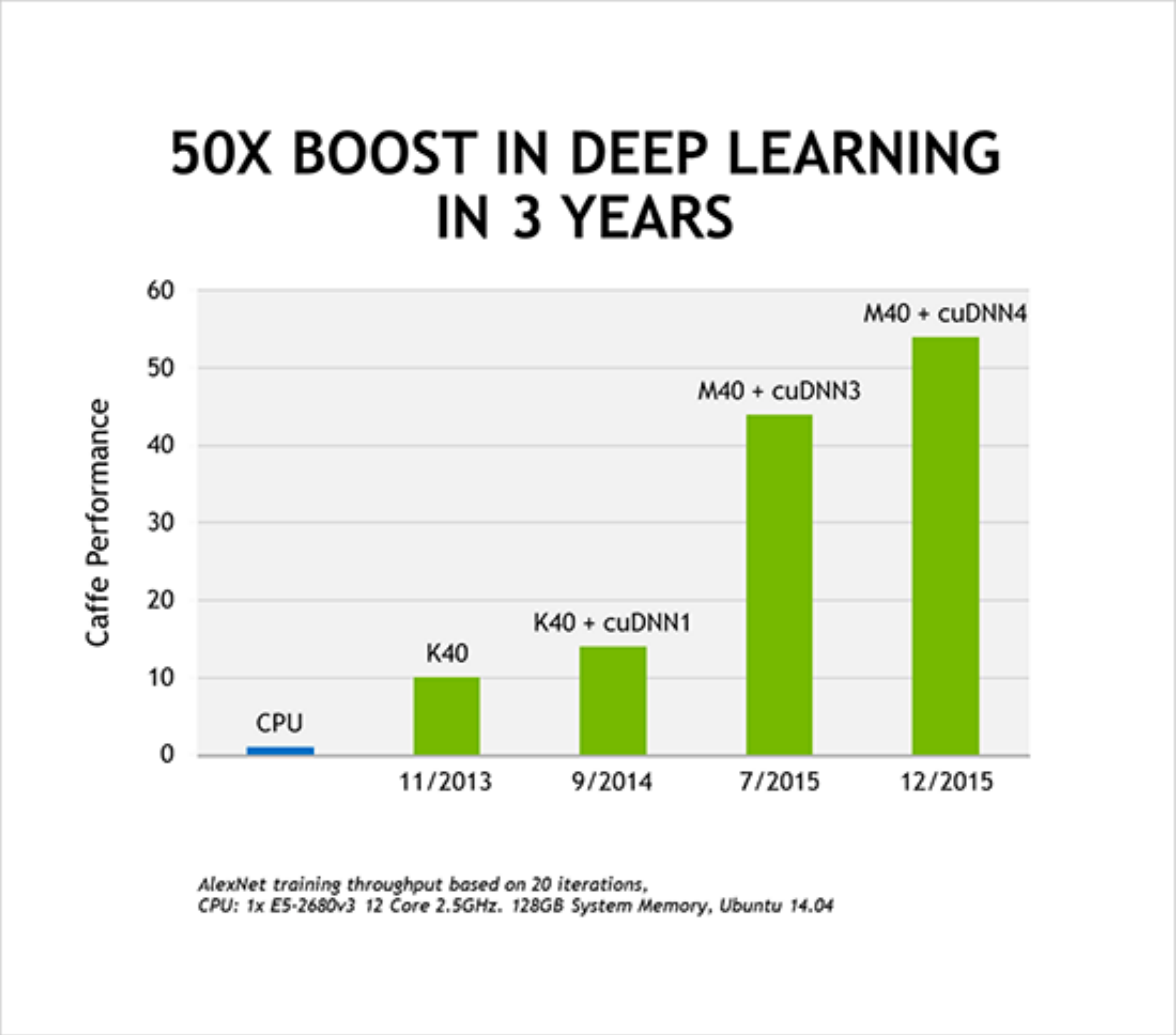
Need for Depth



$$\boldsymbol{h}^{(1)} = g^{(1)}\left(\boldsymbol{W}^{(1)\top}\boldsymbol{x} + \boldsymbol{b}^{(1)}\right)$$

$$\boldsymbol{h}^{(2)} = g^{(2)}\left(\boldsymbol{W}^{(2)\top}\boldsymbol{h}^{(1)} + \boldsymbol{b}^{(2)}\right)$$

DEEP NEURAL NETWORKS



- Are these networks trainable ?
  - Advances in computation and processing
  - Graphical processing units (GPUs) performing multiple parallel multiply accumulate operations.
  - Large amounts of supervised data sets

DEEP NEURAL NETWORKS

- Will the networks generalize with deep networks
  - DNNs are quite data hungry and performance improves by increasing the data.
  - Generalization problem is tackled by providing training data from all possible conditions.
    - Many artificial data augmentation methods have been successfully deployed
  - Providing the state-of-art performance in several real world applications.

Representation Learning in Deep Networks

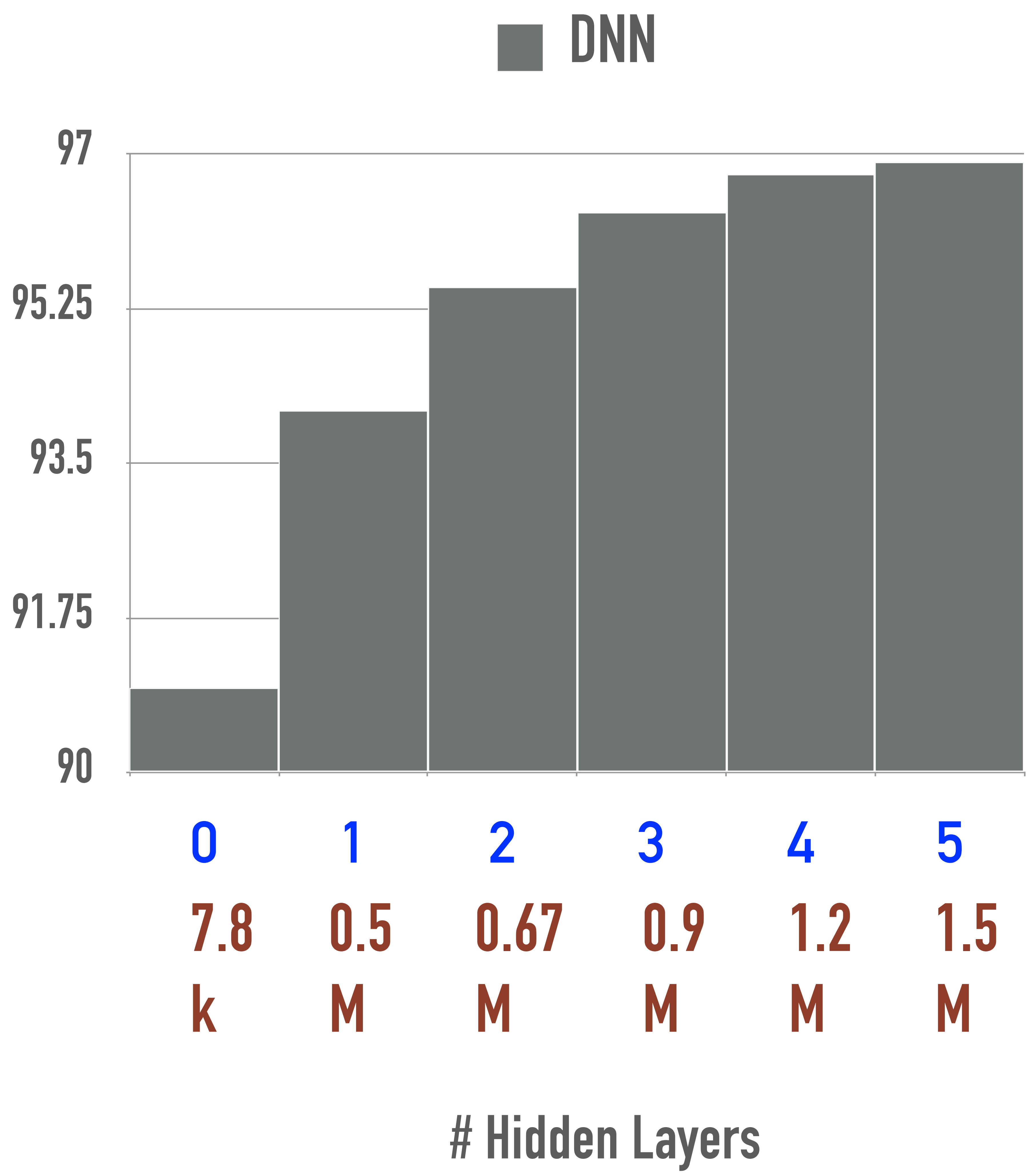
- The input data representation is one of most important components of any machine learning system.
  - Extract factors that enable classification while suppressing factors which are susceptible to noise.

- Finding the right representation for real world applications - substantially challenging.
  - Deep learning solution - build complex representations from simpler representations.
  - The dependencies between these hierarchical representations are refined by the target.

DEEP NEURAL NETWORKS

DNNS FOR MNIST

- Generally
  - Depth improves the performance
  - Saturating effects of increasing the depth.



An overview of gradient descent optimization algorithms*

Sebastian Ruder
Insight Centre for Data Analytics, NUI Galway
Aylien Ltd., Dublin
ruder.sebastian@gmail.com

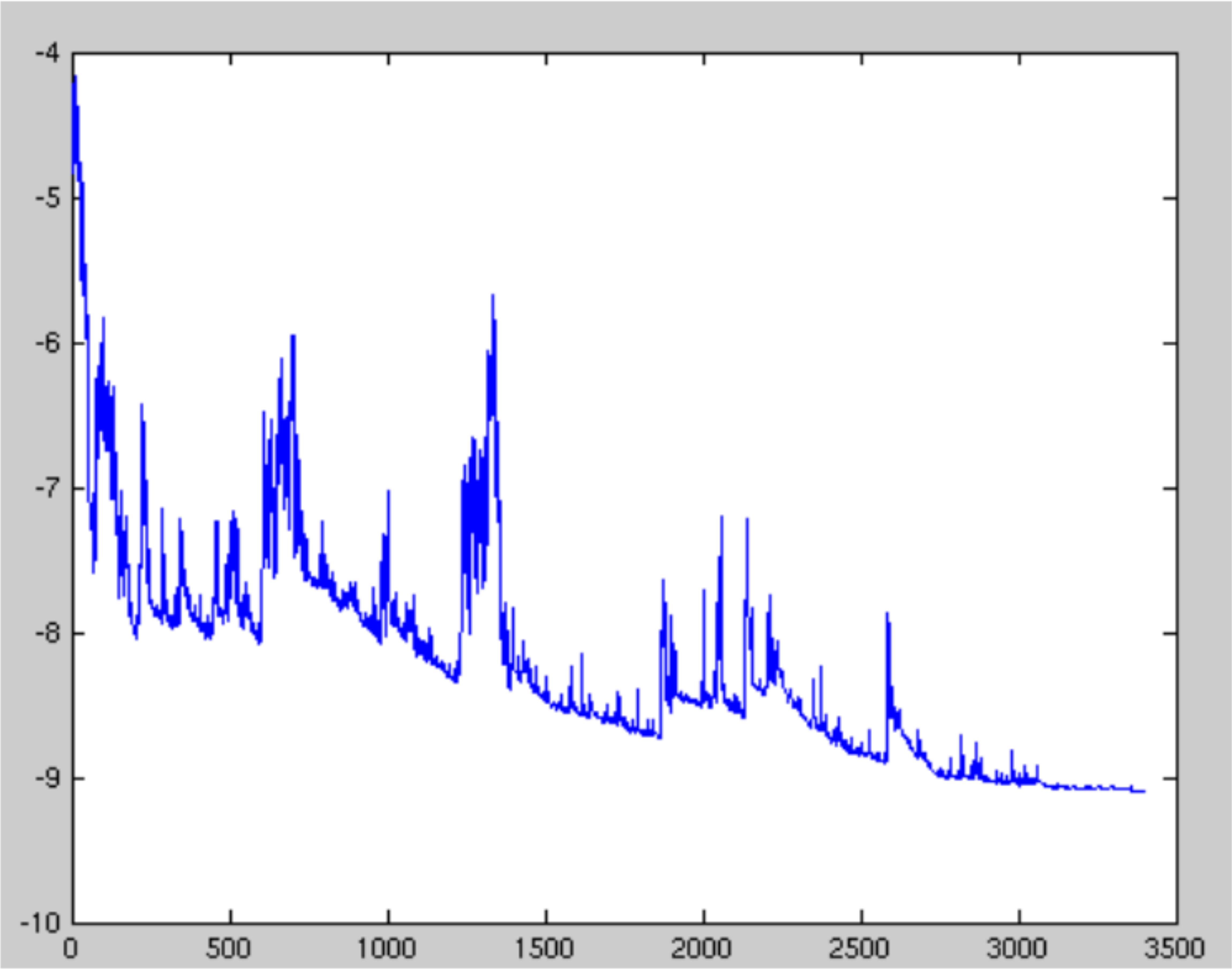
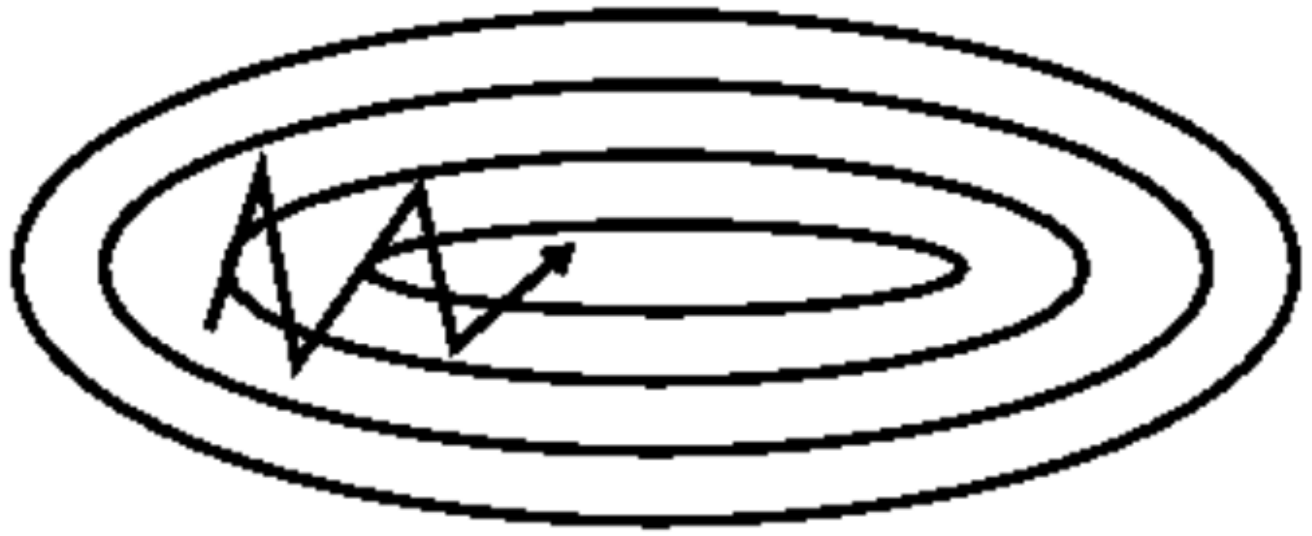


Figure 1: SGD fluctuation (Source: Wikipedia)

Momentum

(a) SGD without momentum



(b) SGD with momentum

Batch Normalization

Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift

Sergey Ioffe
Google Inc., *sioffe@google.com*

Christian Szegedy
Google Inc., *szegedy@google.com*

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_\mathcal{B} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

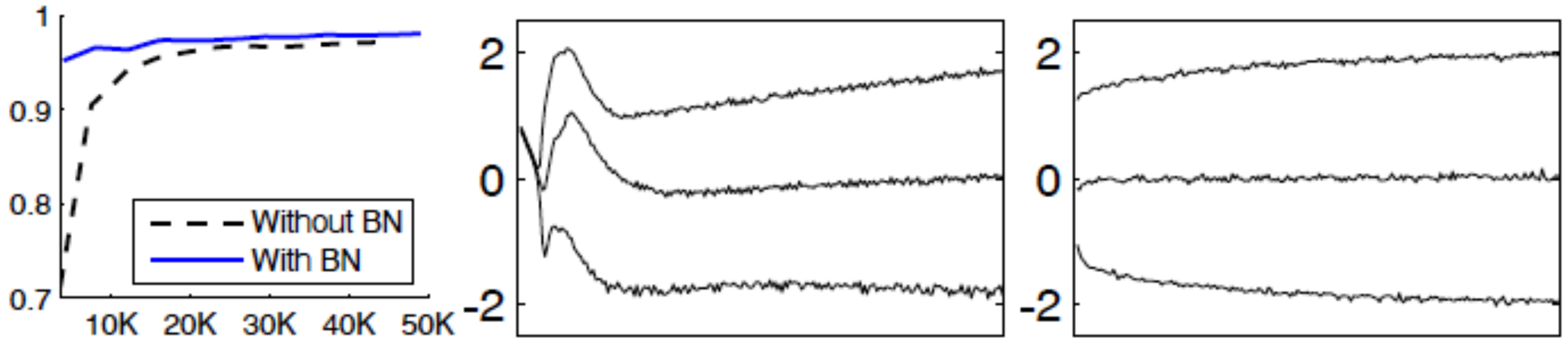
$$\sigma_\mathcal{B}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_\mathcal{B})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_\mathcal{B}}{\sqrt{\sigma_\mathcal{B}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini-batch.

Effect of Batch Normalization



(a) (b) Without BN (c) With BN

Figure 1: (a) *The test accuracy of the MNIST network trained with and without Batch Normalization, vs. the number of training steps. Batch Normalization helps the network train faster and achieve higher accuracy.* (b, c) *The evolution of input distributions to a typical sigmoid, over the course of training, shown as* $\{15, 50, 85\}$*th percentiles. Batch Normalization makes the distribution more stable and reduces the internal covariate shift.*

THANK YOU

Sriram Ganapathy and TA team
LEAP lab, C328, EE, IISc
sriramg@iisc.ac.in