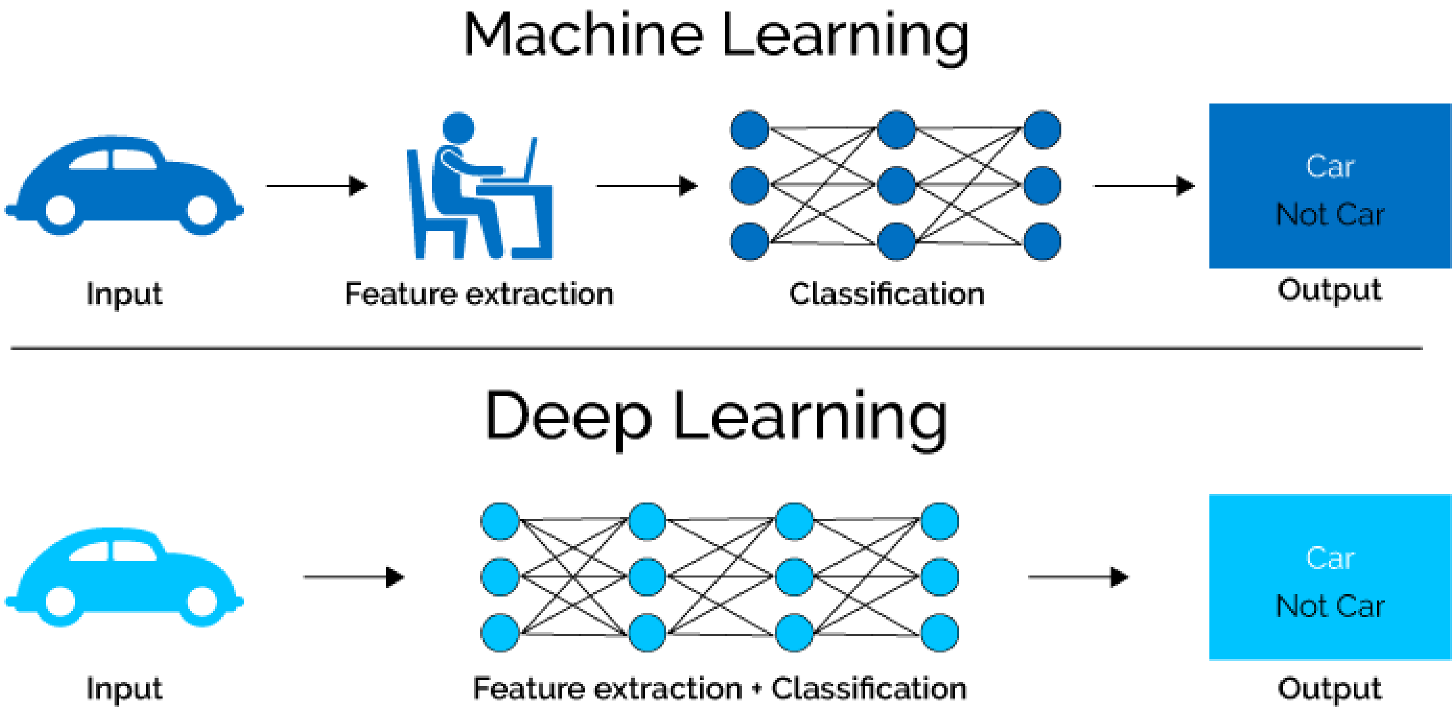
CSci 8980

Basics of Machine Learning and Deep Learning (Part II)

DL vs. ML

- Learning representations and patterns of data
- Generalization (failure of classic AI/ML)
- Learn (multiple levels of) representation by using a hierarchy of multiple layers



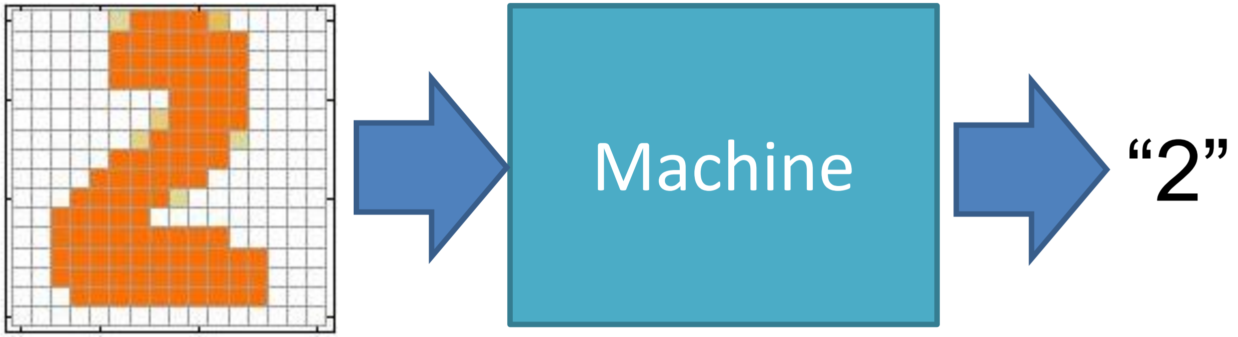
https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png

Why Now?

- Increasing data sets
- Increasing model sizes (machine power)
- The basis for deep learning is Neural Networks
- Let’s take a look at Neural Networks

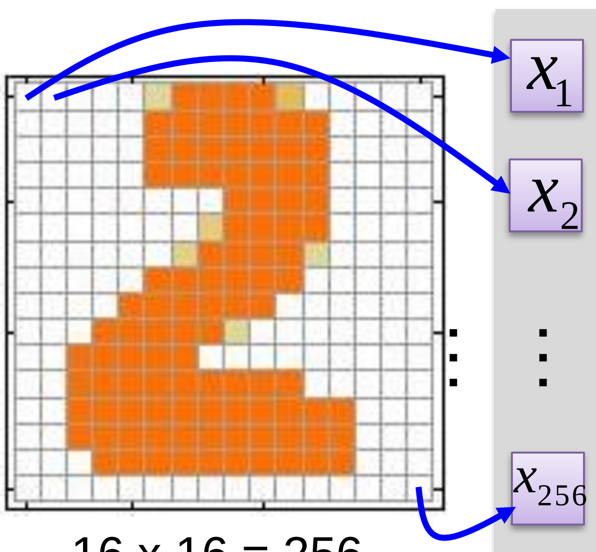
Example Application

- Handwriting Digit Recognition



Handwriting Digit Recognition

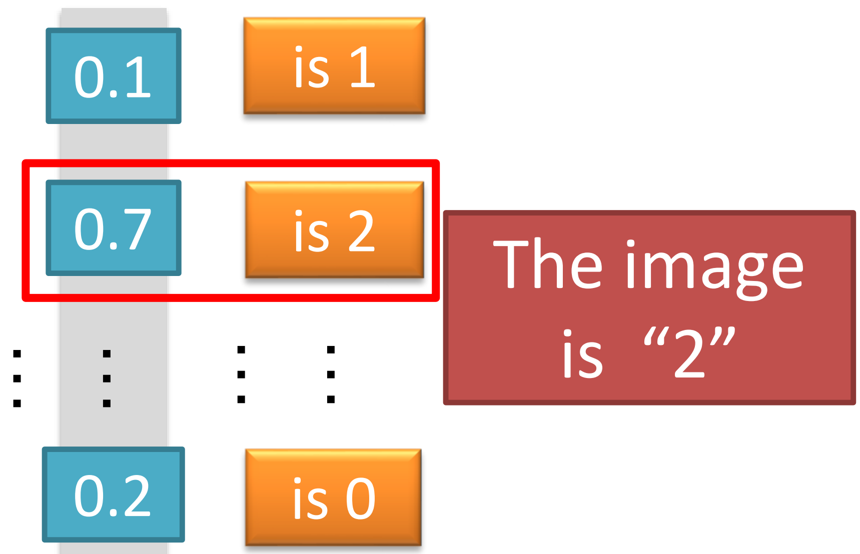
Input



16 x 16 = 256

Ink → 1
No ink → 0

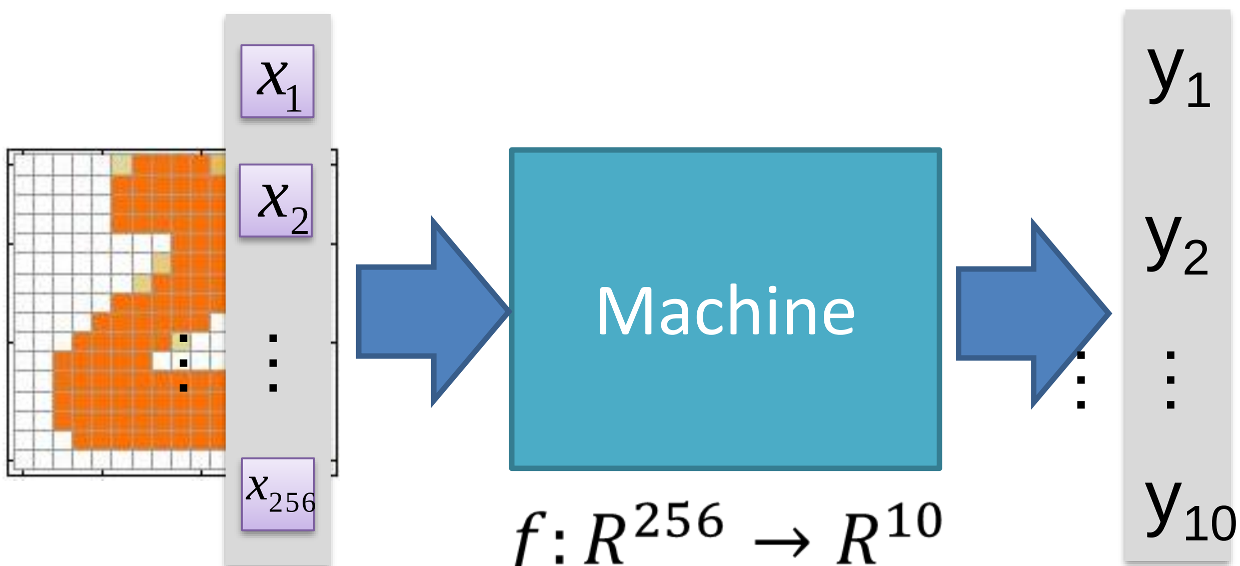
Output



Each dimension represents the confidence of a digit.

Example Application

- Handwriting Digit Recognition



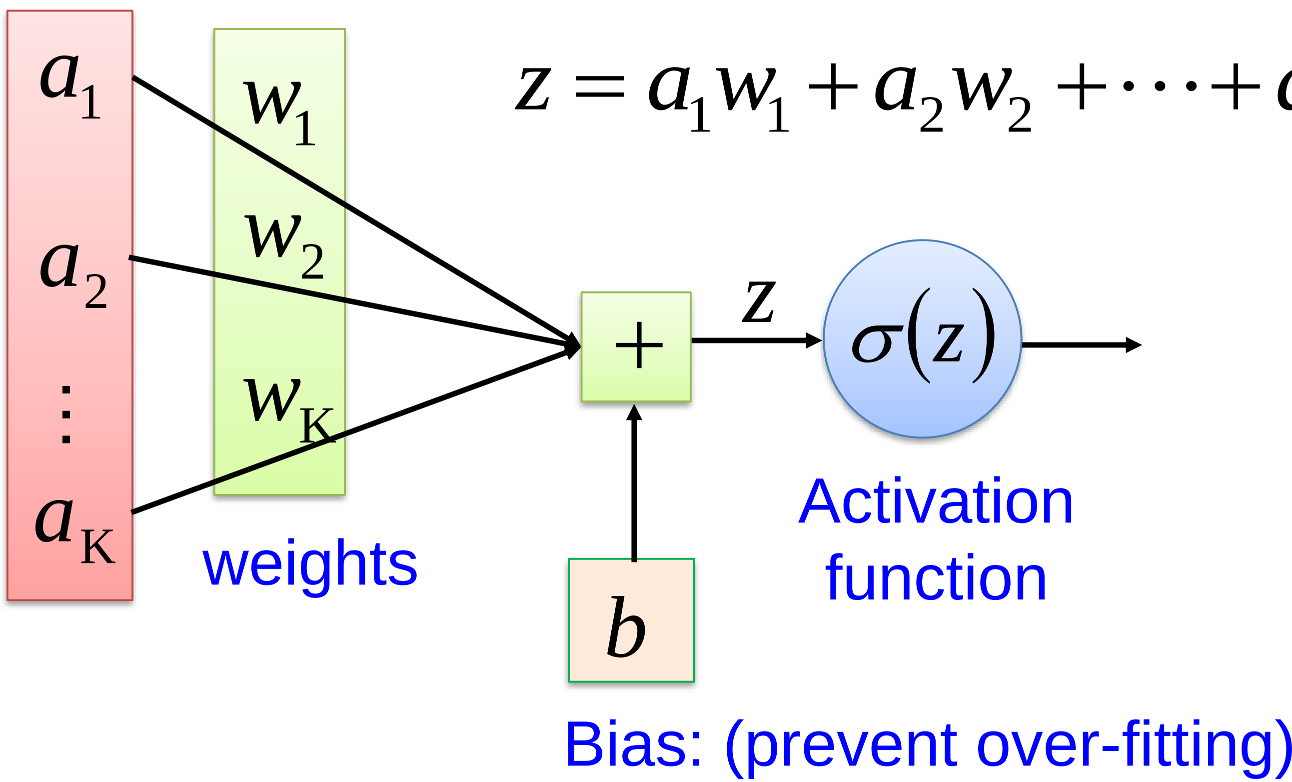
$$f: R^{256} \rightarrow R^{10}$$

In deep learning, the function f is represented by neural network

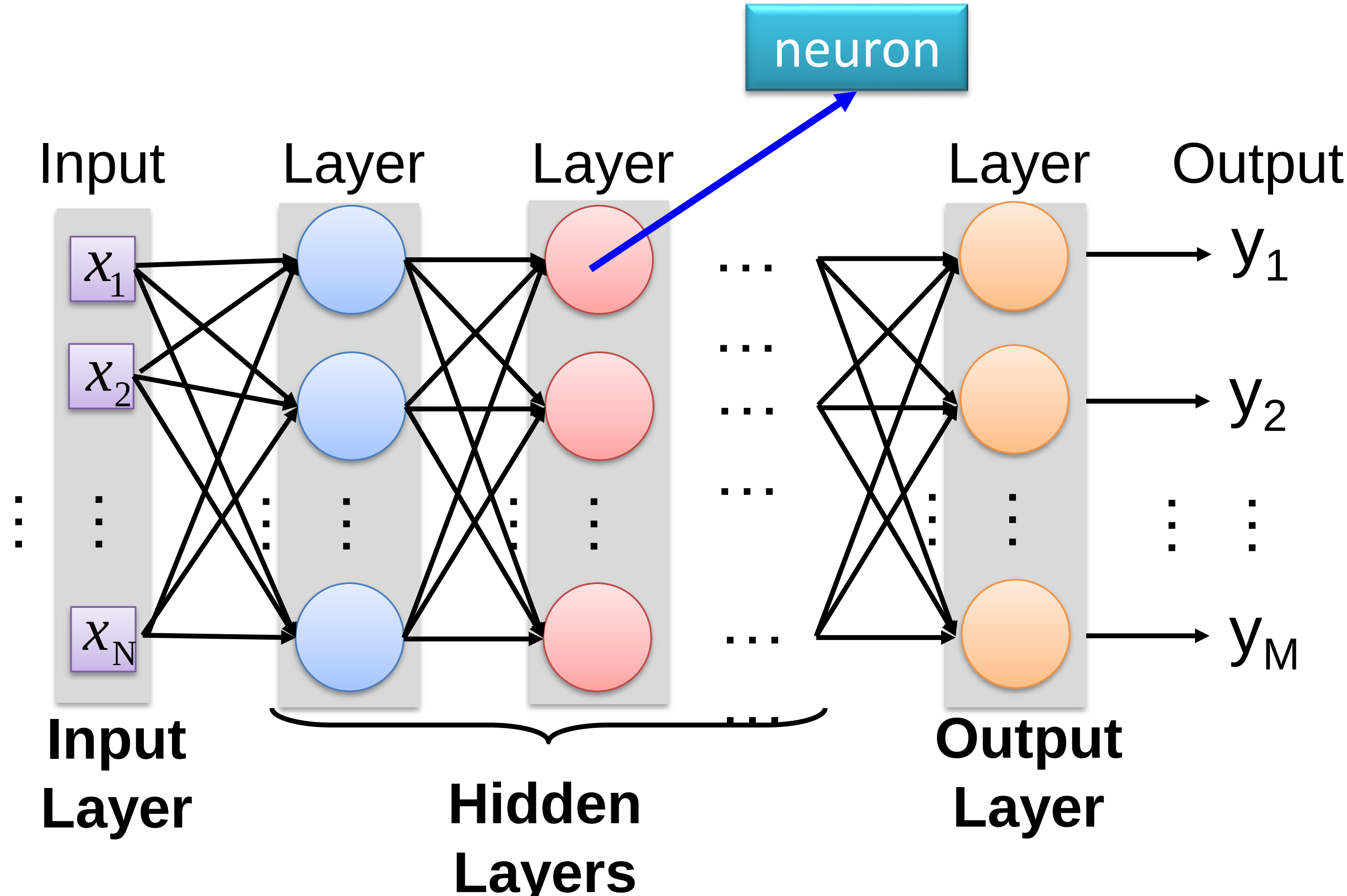
Element of Neural Network

Neuron (perceptron) $f: R^K \rightarrow R$

$$z = a_1 w_1 + a_2 w_2 + \cdots + a_K w_K + b$$

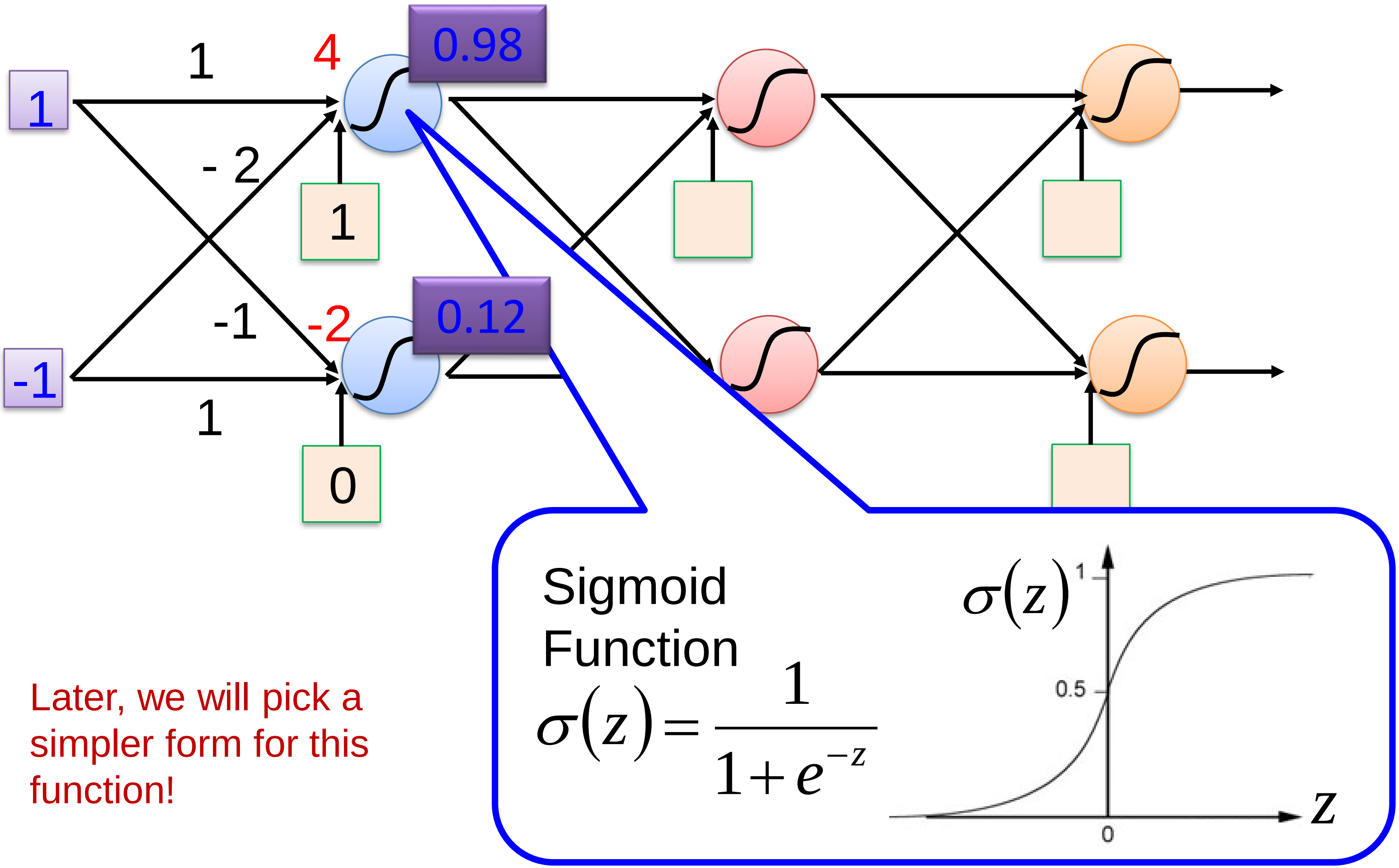


Neural Network



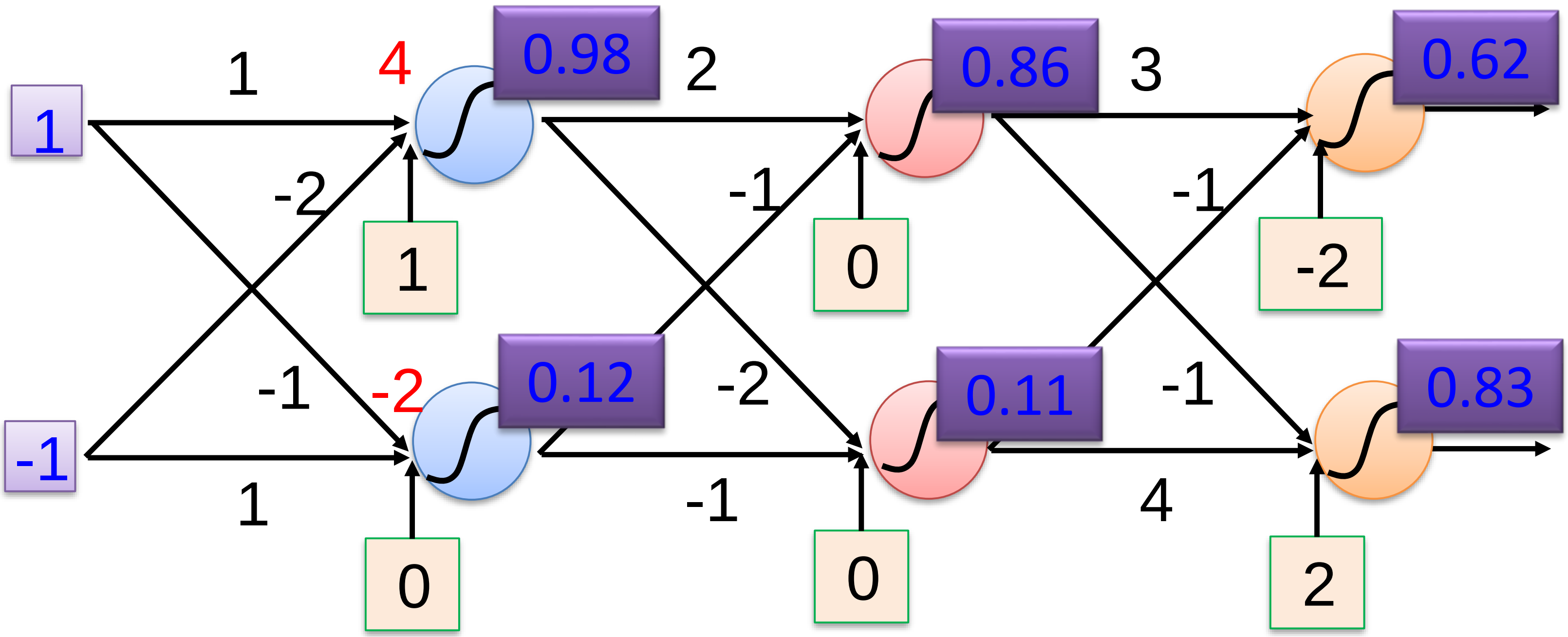
Deep means many hidden layers

Example of Neural Network



Later, we will pick a simpler form for this function!

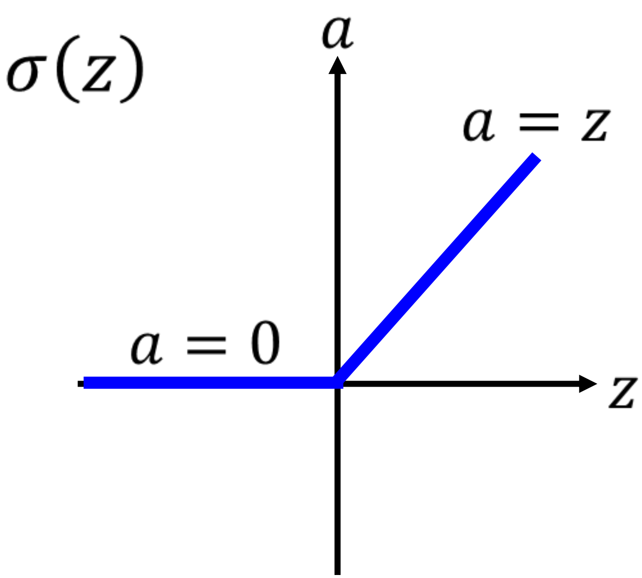
Example of Neural Network



This is called a feed-forward network

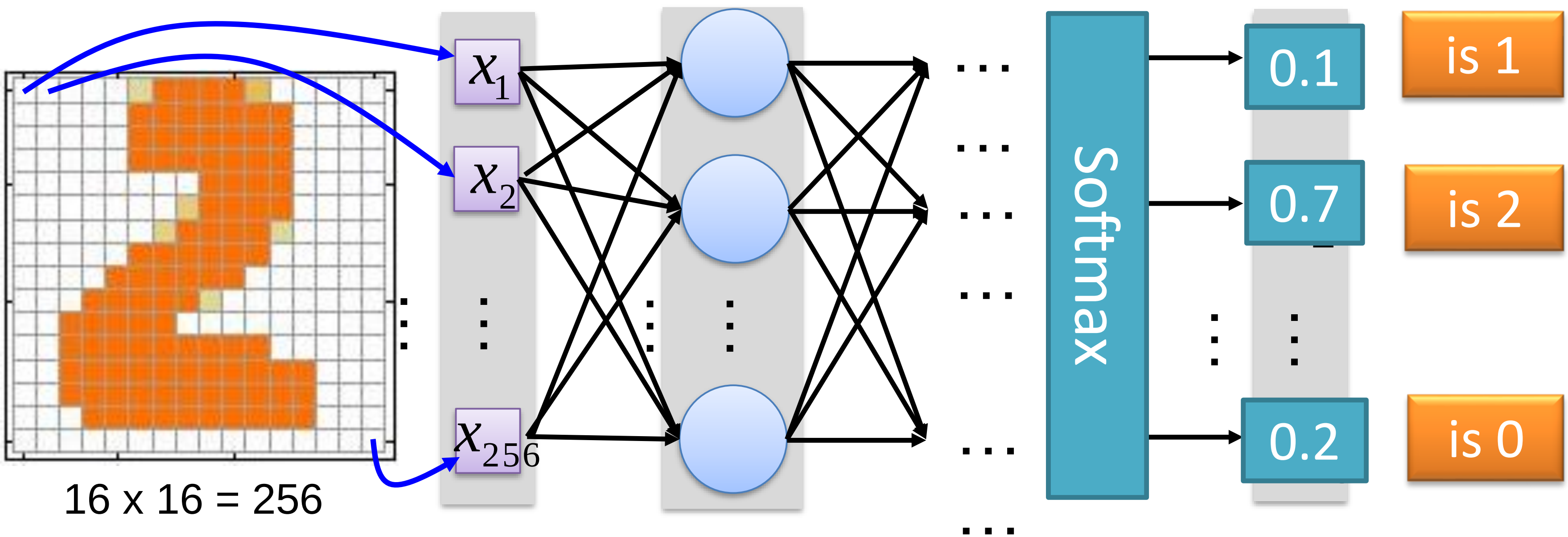
Better Activation Function: ReLU

- Rectified Linear Unit (ReLU): faster convergence than sigmoid



How to set network parameters? Learning!

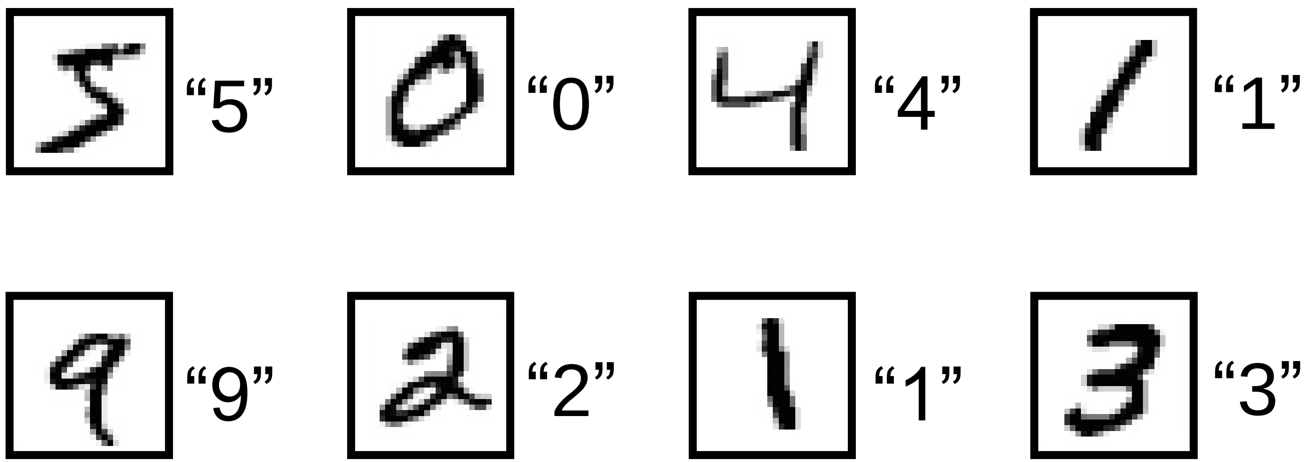
$$\theta = \{W^1, b^1, W^2, b^2, \cdots W^L, b^L\}$$



Ink → 1
No ink → 0

Training Data

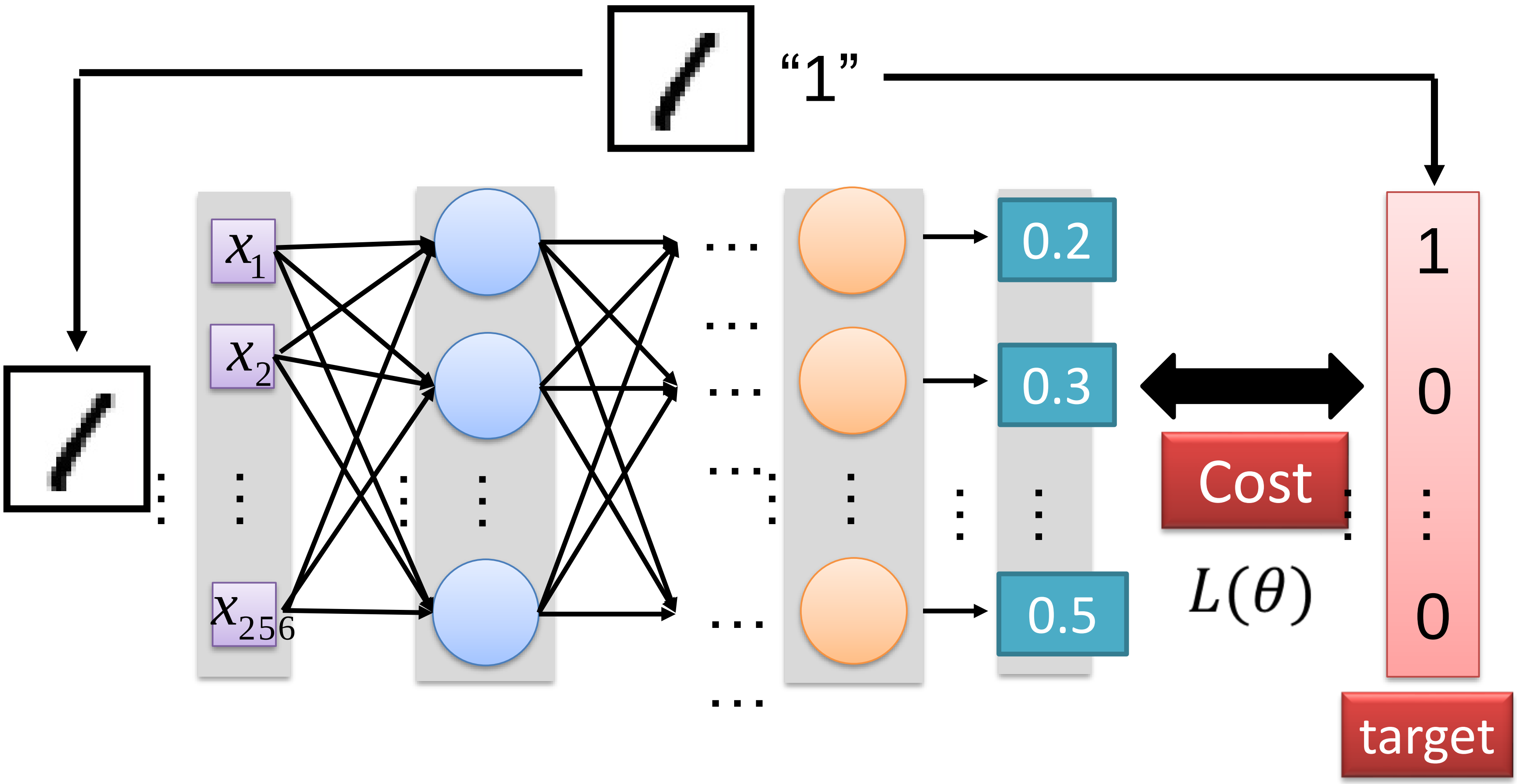
- Preparing training data: images and their labels



Using the training data to find the network parameters.

Cost

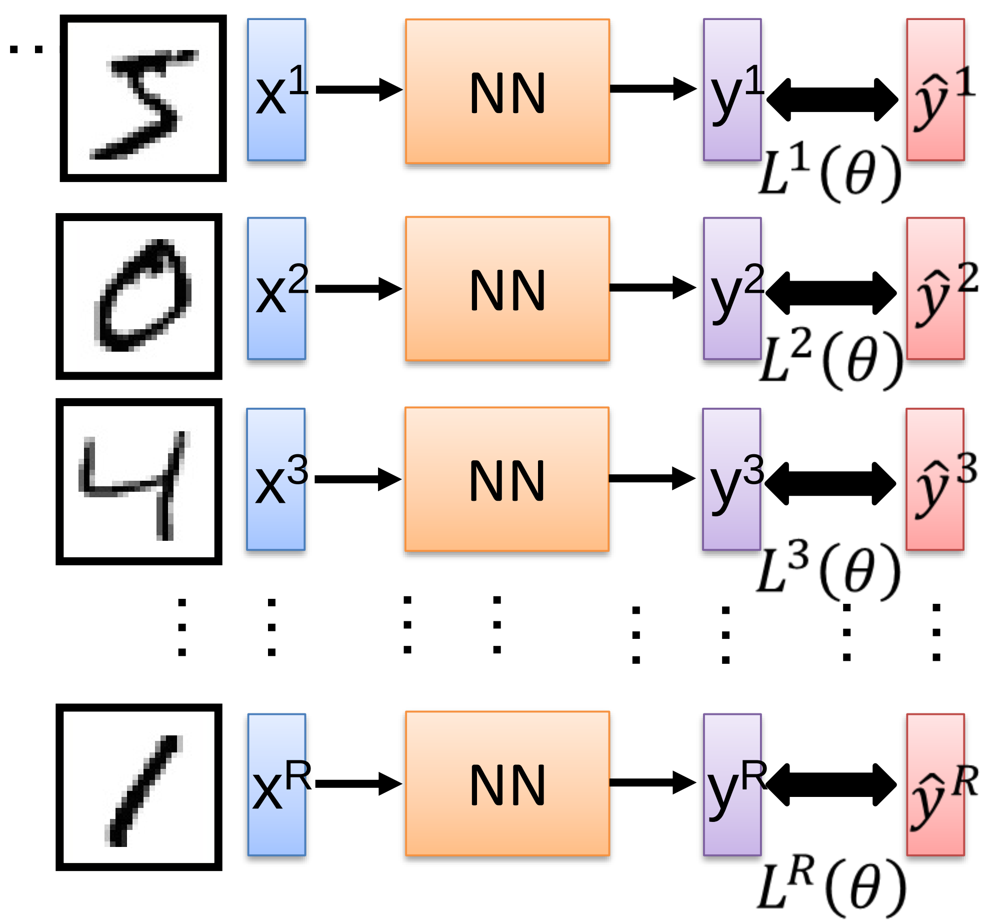
Given a set of network parameters θ, each example has a cost value.



Later we will see a good cost metric is ERROR

Total Cost

For all training data



Total Cost:

$$C(\theta) = \sum_{r=1}^{R} L^r(\theta)$$

How poorly the network parameters θ are for this task

Find the network parameters θ^* that minimize this value

Cost typically measured as error

- Total-Sum-Squared-Error (TSSE)

$$TSSE = \frac{1}{2} \sum_{patterns} \sum_{outputs} (desired - actual)^2$$

- Root-Mean-Squared-Error (RMSE)

$$RMSE = \sqrt{\frac{2 * TSSE}{\# patterns * \# outputs}}$$

Intuition

- Search for parameters along a gradient that minimize cost (i.e. error)
- The idea:
  - Tweak parameters, see how cost/error changes
  - Do it again, and again, …
- Gradient descent gives you a mathematical recipe to tweak parameters
  - Far away: take big jumps
  - Get close: take small jumps

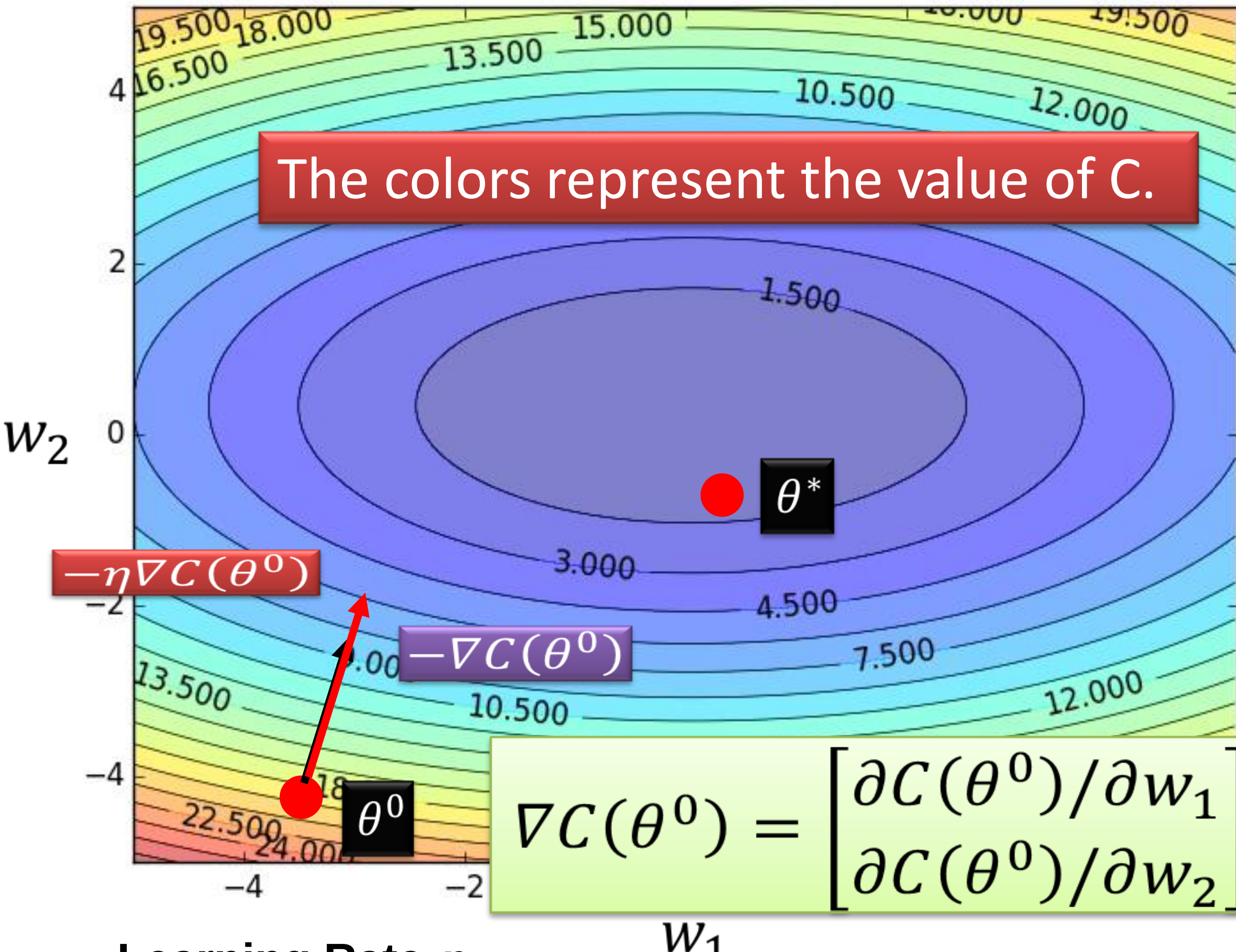
Gradient Descent

to find a minima

Assume there are only two parameters w_1 and w_2 in a network.

$$\theta = \{w_1, w_2\}$$

Error Surface



$$\nabla C(\theta^0) = \begin{bmatrix} \partial C(\theta^0)/\partial w_1 \\ \partial C(\theta^0)/\partial w_2 \end{bmatrix}$$

Randomly pick a starting point θ^0

Compute the negative gradient at θ^0

$\Rightarrow -\nabla C(\theta^0)$

Times the learning rate η

$\Rightarrow -\eta\nabla C(\theta^0)$

Learning Rate-η
A scalar parameter, analogous to step size in numerical integration, used to set the rate of adjustments

Backpropagation

- Backpropagation is a popular way to compute the gradients and the weights efficiently
  - Many toolkits can compute the gradients automatically

Backpropagation Algorithm

- Randomly choose the initial weights
- While error is too large
  - For each training pattern (presented in random order)
    - Apply the inputs to the network
    - Calculate the output for every neuron from the input layer, through the hidden layer(s), to the output layer
    - Calculate the error at the outputs
    - Use the output error to compute error signals for pre-output layers
    - Use the error signals to compute weight adjustments
    - Apply the weight adjustments
  - Periodically evaluate the network performance

Error Backpropagation

TSSE

$$L_n = \frac{1}{2}\left(y_n - f(x_n)\right)^2$$

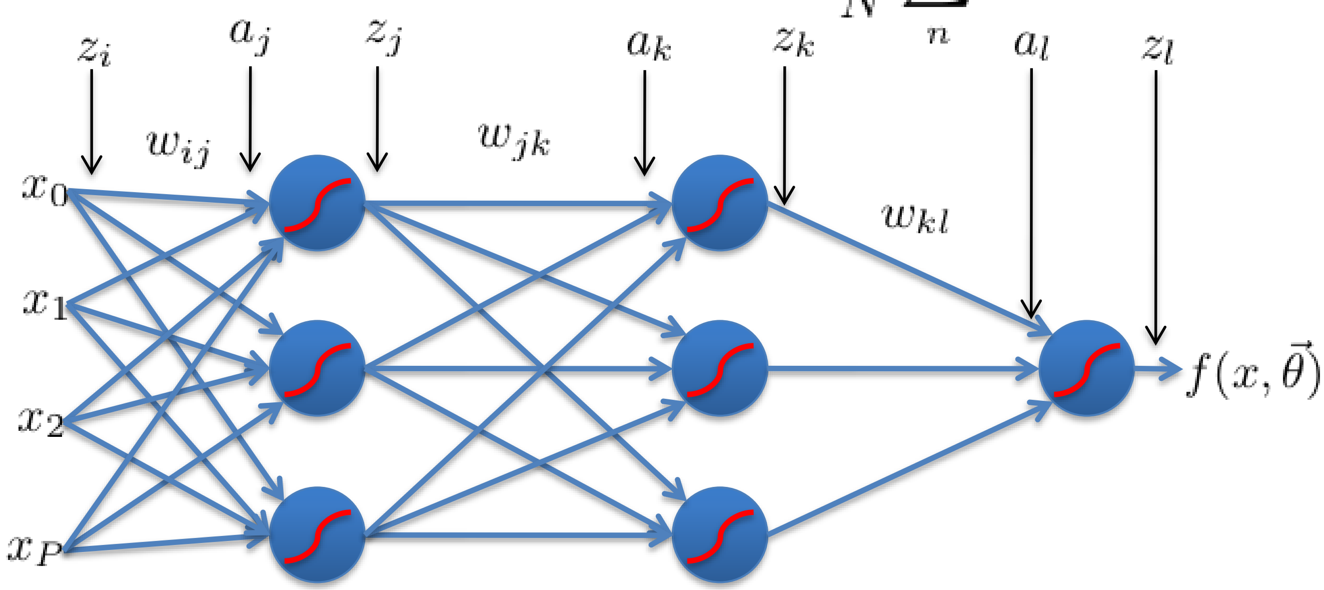
Optimize last layer weights w_{kl}

$$\frac{\partial R}{\partial w_{kl}} = \frac{1}{N}\sum_n \left[\frac{\partial L_n}{\partial a_{l,n}}\right]\left[\frac{\partial a_{l,n}}{\partial w_{kl}}\right]$$

Calculus chain rule

$$\frac{\partial R}{\partial w_{kl}} = \frac{1}{N}\sum_n \left[\frac{\partial \frac{1}{2}(y_n - g(a_{l,n}))^2}{\partial a_{l,n}}\right]\left[\frac{\partial z_{k,n} w_{kl}}{\partial w_{kl}}\right] = \frac{1}{N}\sum_n \boxed{\left[-(y_n - z_{l,n})g'(a_{l,n})\right]} z_{k,n}$$

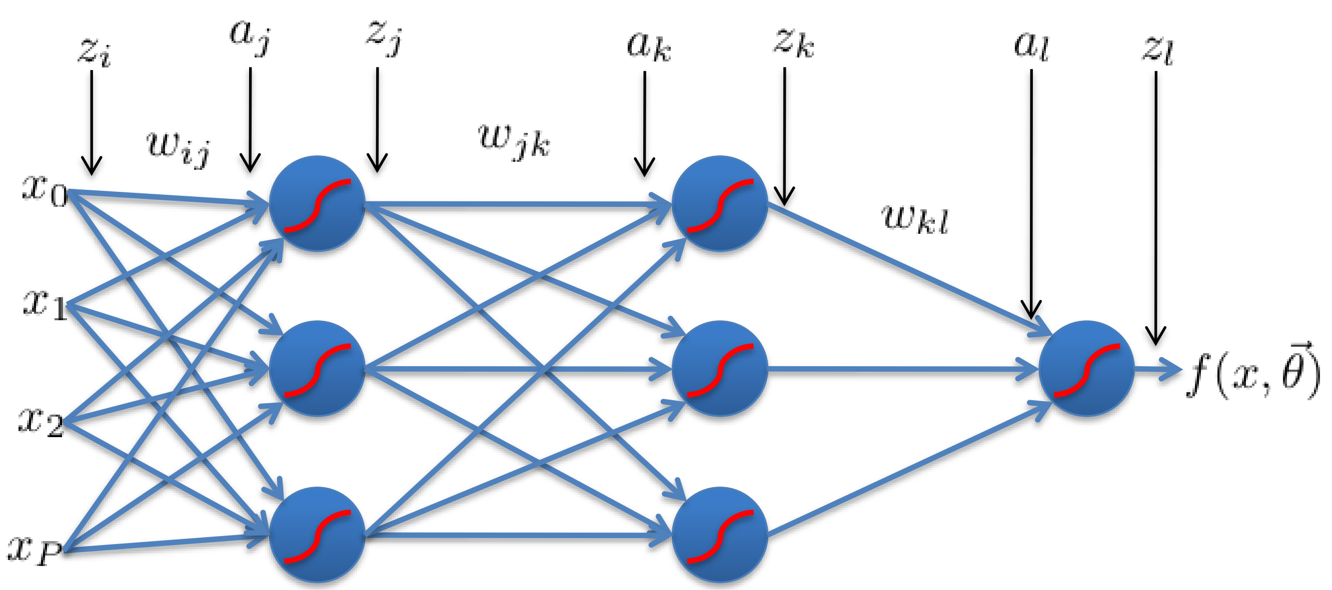
$$= \frac{1}{N}\sum_n \delta_{l,n} n z_{k,n}$$



Error Backpropagation

Now that we have well defined gradients for each parameter, update using Gradient Descent

$$
\begin{aligned}
w_{ij}^{t+1} &= w_{ij}^{t} - \eta \frac{\partial R}{w_{ij}} \\
w_{jk}^{t+1} &= w_{jk}^{t} - \eta \frac{\partial R}{w_{kl}} \\
w_{kl}^{t+1} &= w_{kl}^{t} - \eta \frac{\partial R}{w_{kl}}
\end{aligned}
$$



How many layers: Deeper is Better?

Layer X Size	Word Error Rate (%)
1 X 2k	24.2
2 X 2k	20.4
3 X 2k	18.4
4 X 2k	17.8
5 X 2k	17.2
7 X 2k	17.1

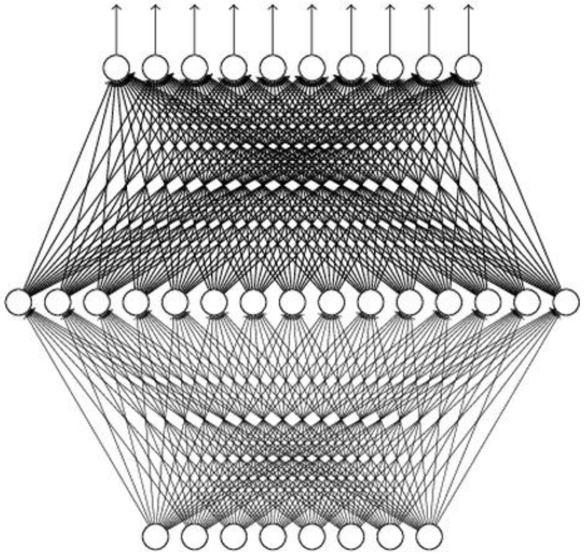
Not surprised, more parameters, better performance

(word recognition task)

Universality Theorem

Any continuous function f

$$f : R^N \rightarrow R^M$$

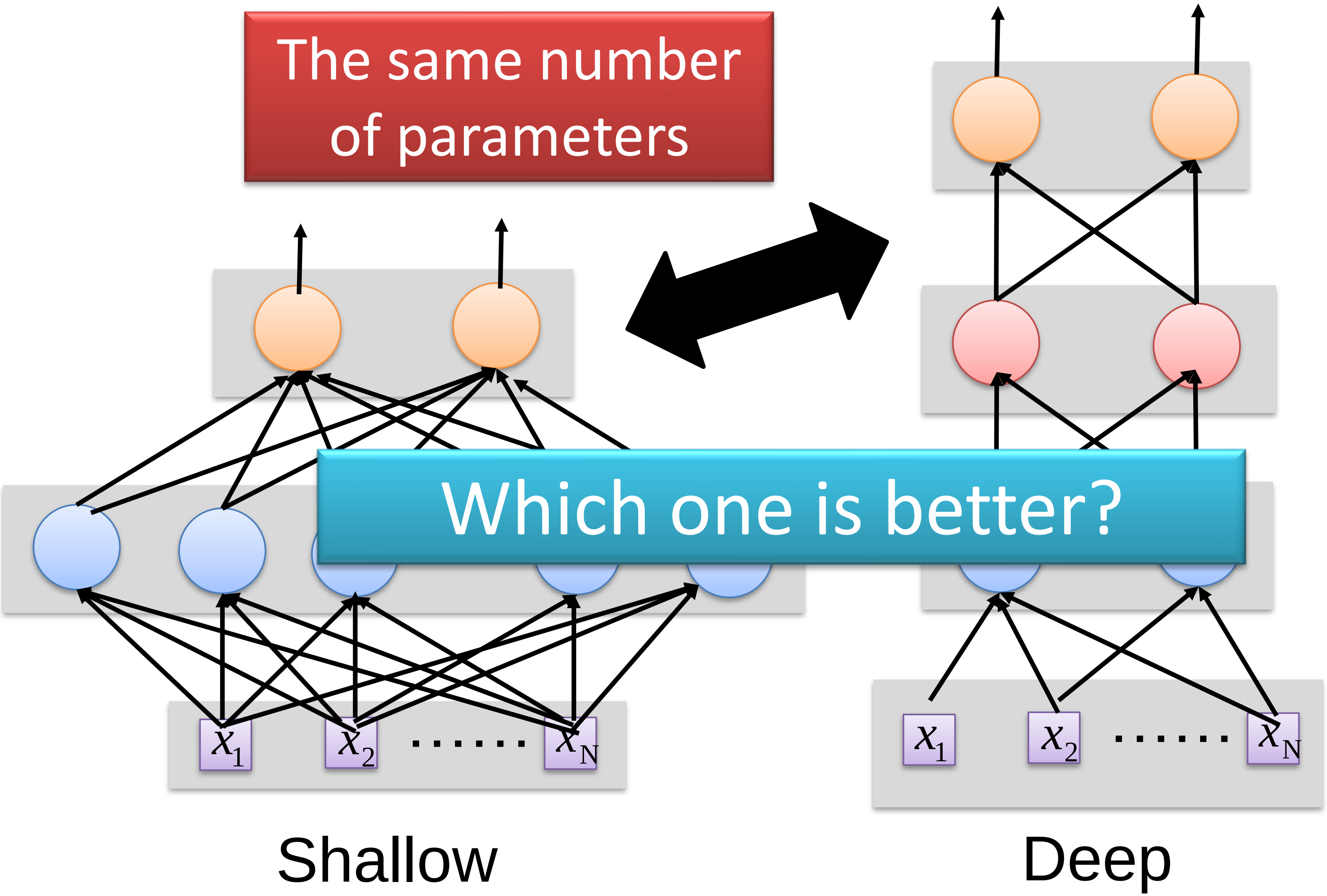


Can be realized by a network with one hidden layer

(given **enough** hidden neurons)

Why "Deep" neural network not "Fat" neural network?

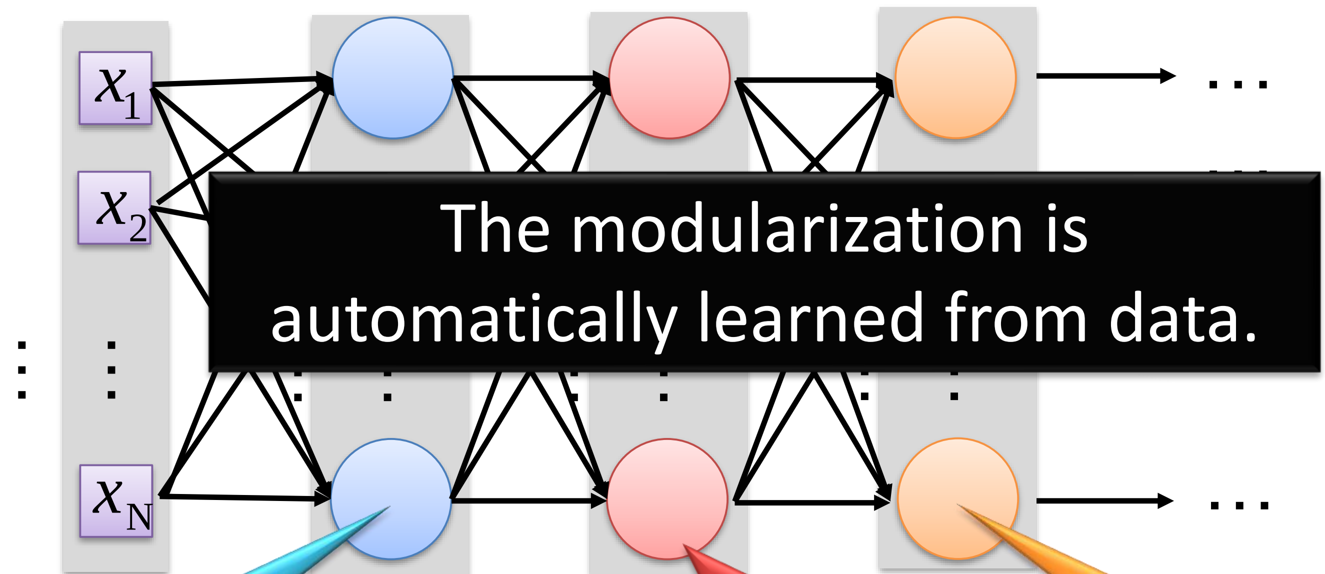
Fat + Short v.s. Thin + Tall



Deep Wins

Deep Learning also works on small data sets

- Deep → Modularization



The modularization is automatically learned from data.

The most basic classifiers

Use 1st layer as module to build classifiers

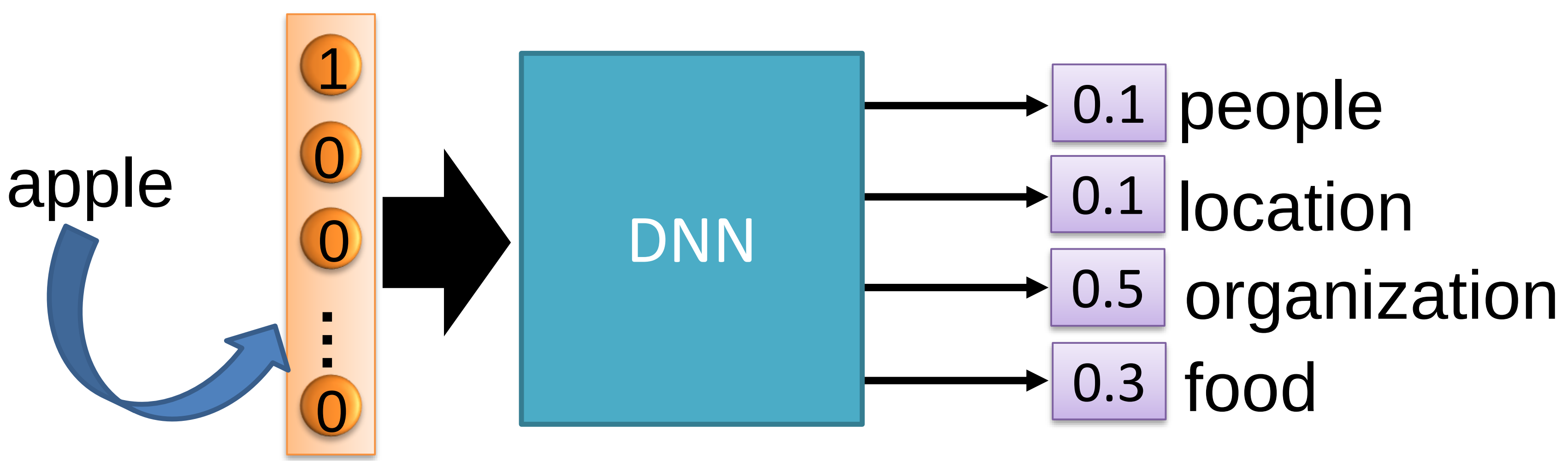
Use 2nd layer as module ……

module = subtasks

Neural Network with Memory

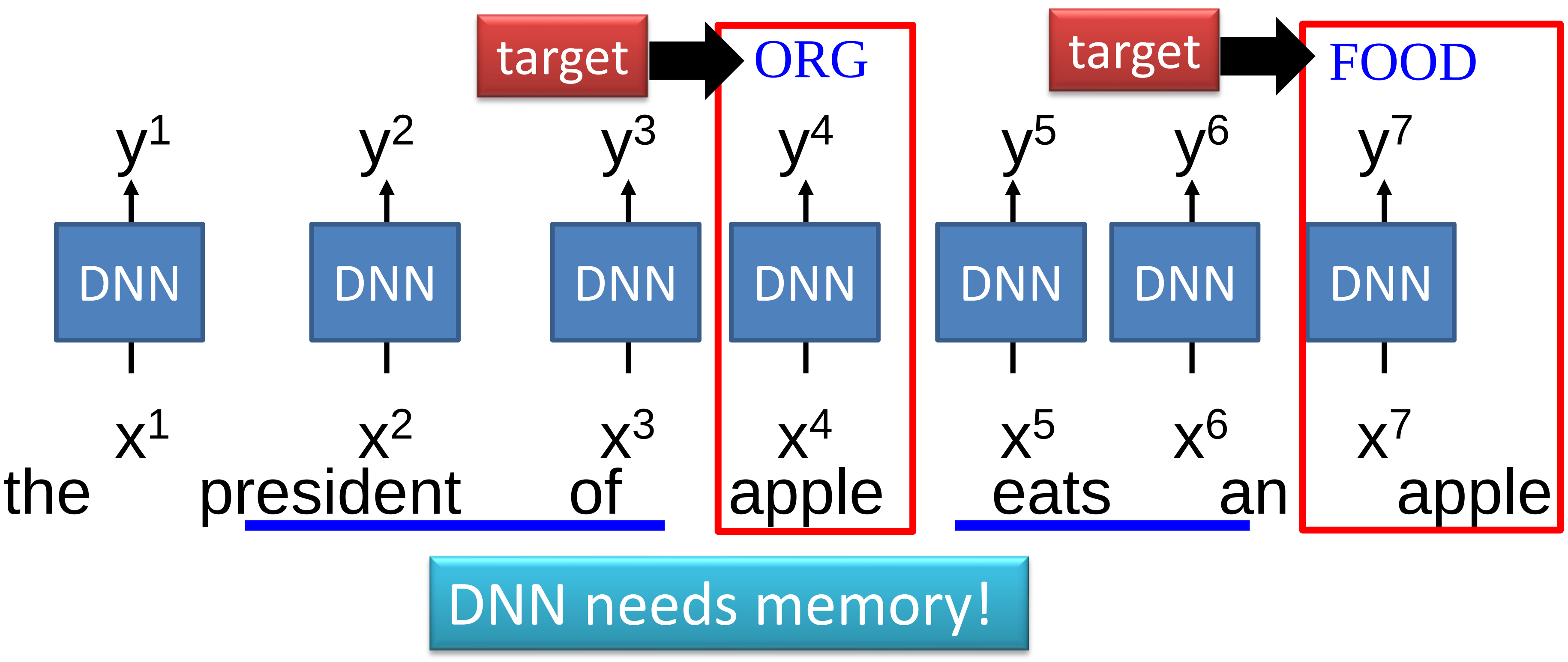
Neural Network needs Memory

- Name Entity Recognition
  - Detecting named entities like name of people, locations, organization, etc. in a sentence.



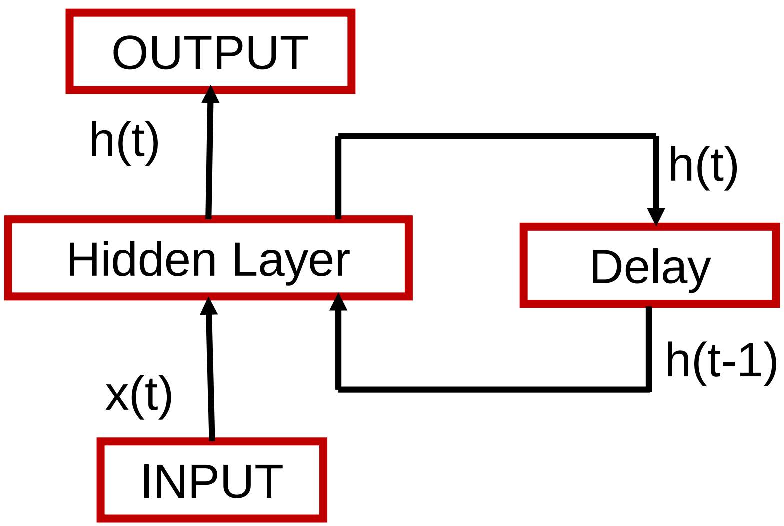
Neural Network needs Memory

- Name Entity Recognition
  - Detecting named entities like name of people, locations, organization, etc. in a sentence.



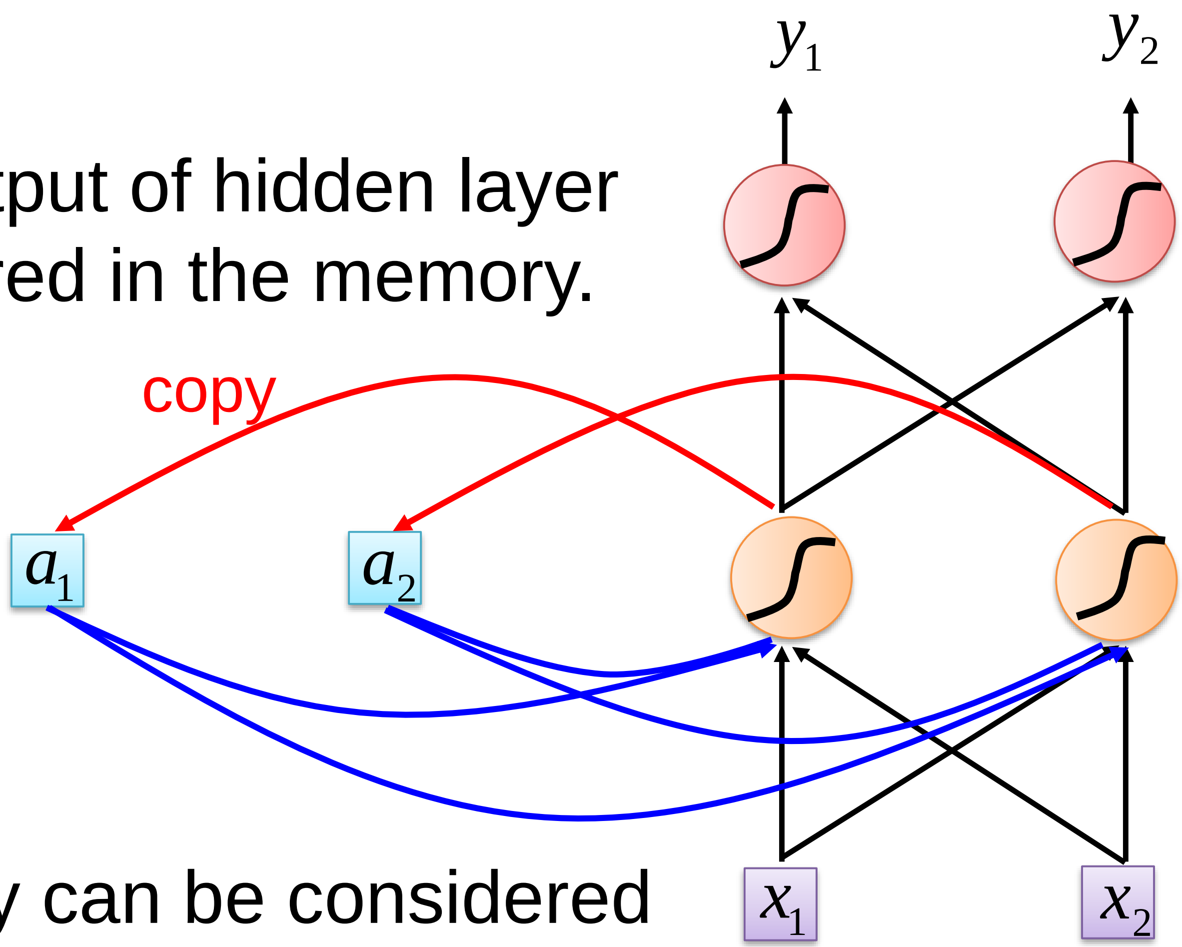
Solution: Recurrent Neural Network (RNN)

- Recurrent neural networks selectively pass information across sequence steps, while processing seq. data one element at a time.
- Allows a memory of the previous inputs to persist in the model's internal state and influence the outcome.



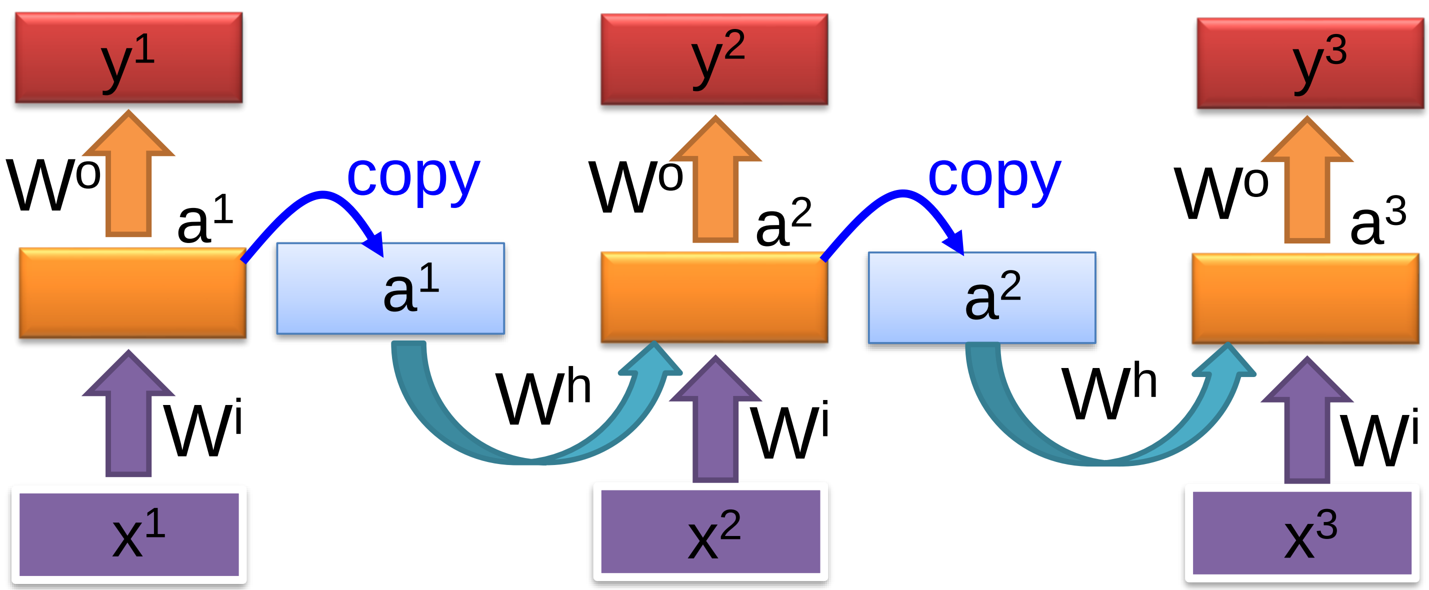
Recurrent Neural Network (RNN)

The output of hidden layer are stored in the memory.



Memory can be considered as another input.

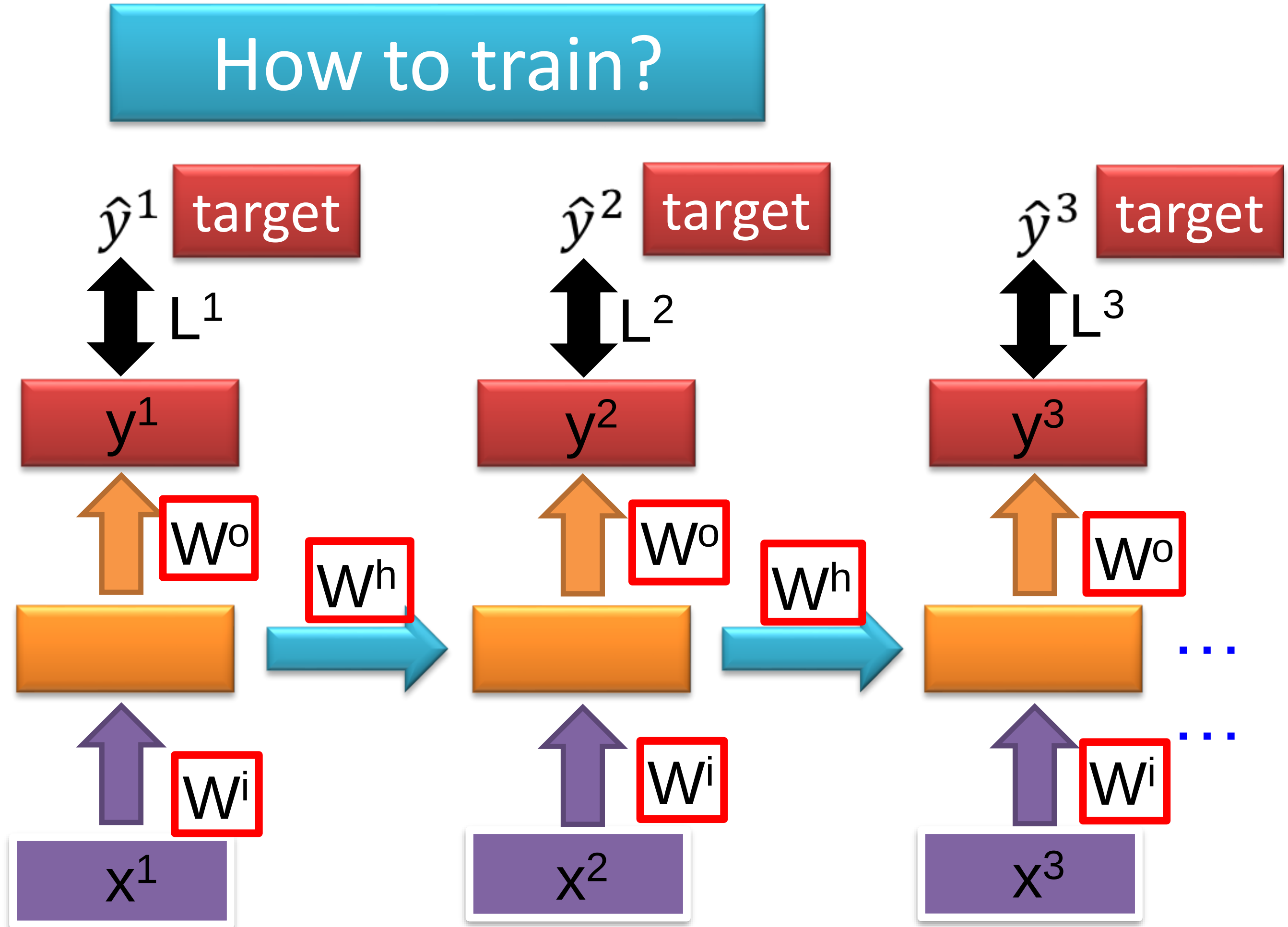
RNN



The same network is used again and again.

Output y^i depends on x^1, x^2, …… x^i

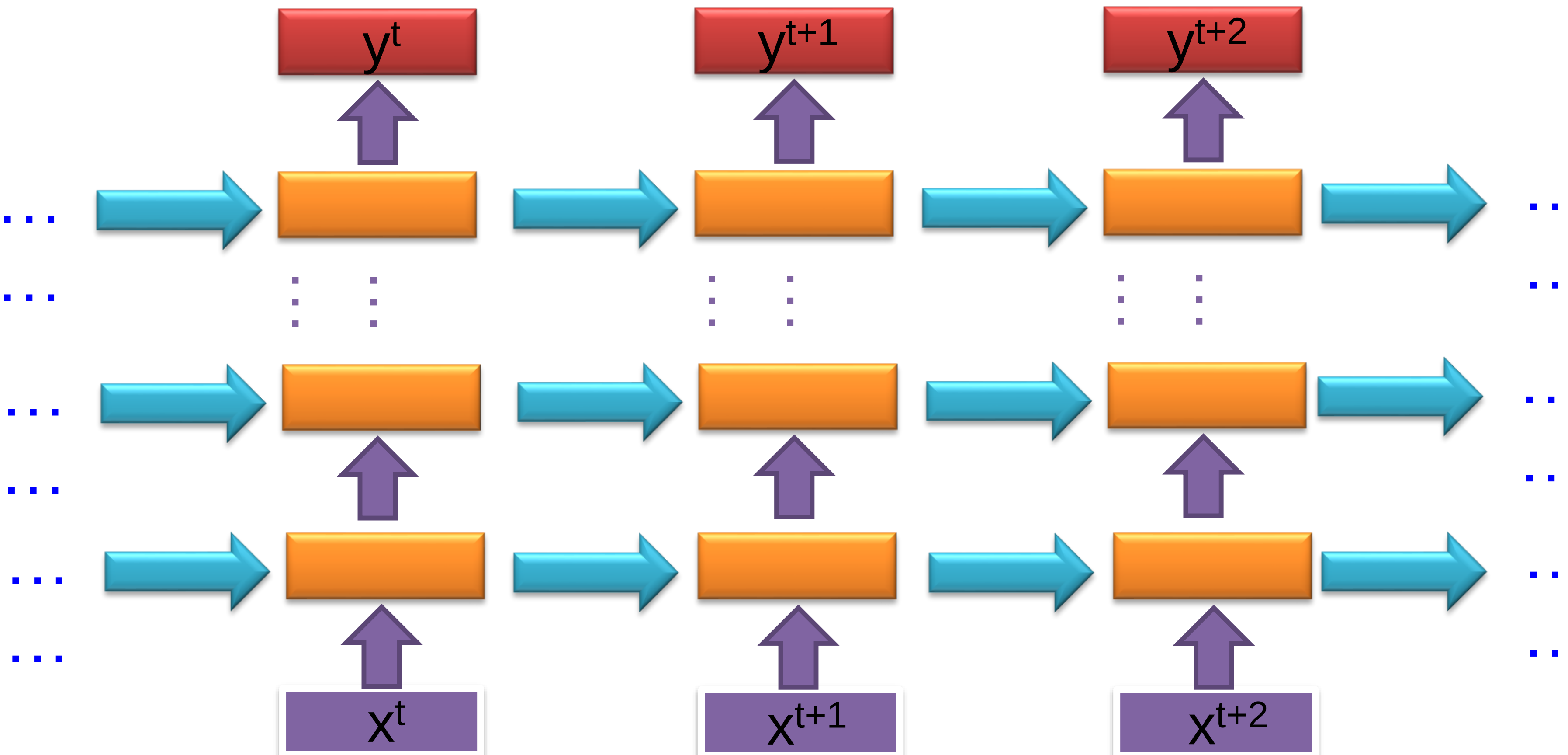
RNN



Find the network parameters to minimize the total cost:

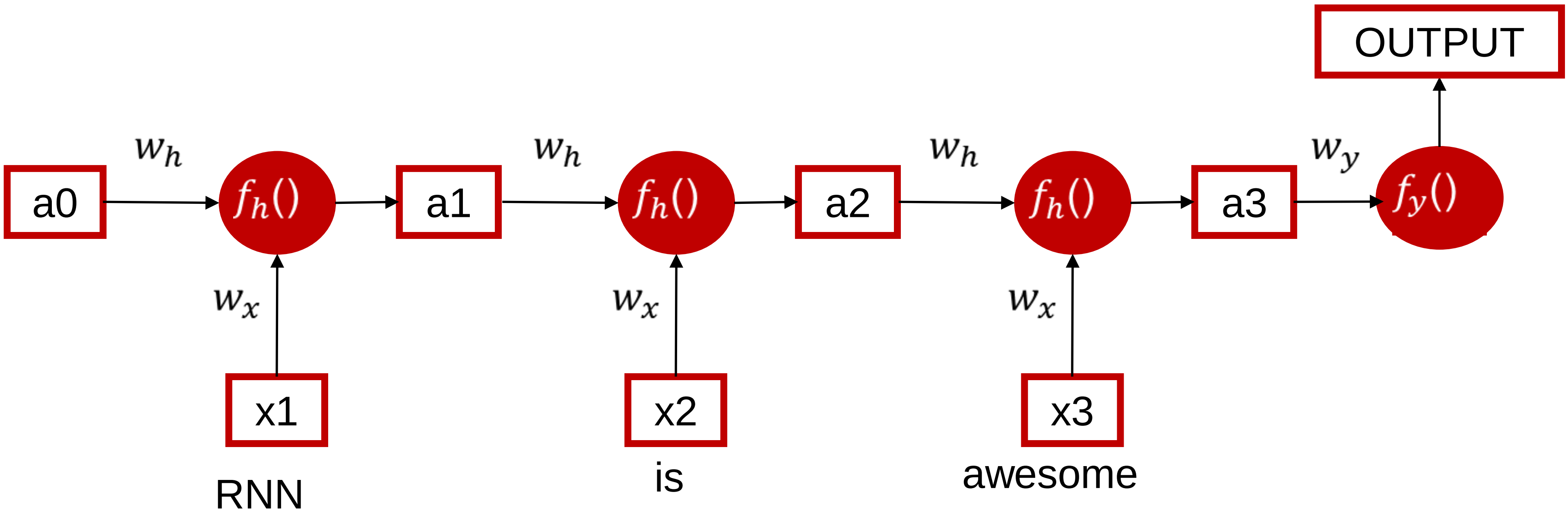
Backpropagation through time (BPTT)

Of course it can be deep …



RNN (rolled over time)

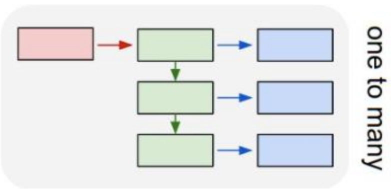
RNN is awesome



$$a(t) = f_h\left(w_h * a(t-1) + w_x * x(t)\right)$$

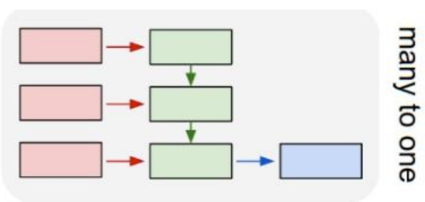
f_h are f_y are the activation function(s)

Modeling Sequences



A person riding a motorbike on dirt road

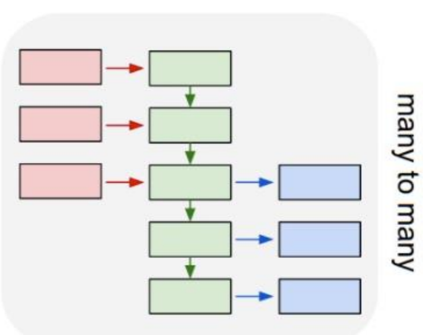
Image Captioning



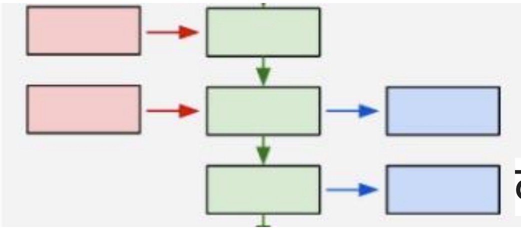
Awesome tutorial.

Positive

Sentiment Analysis



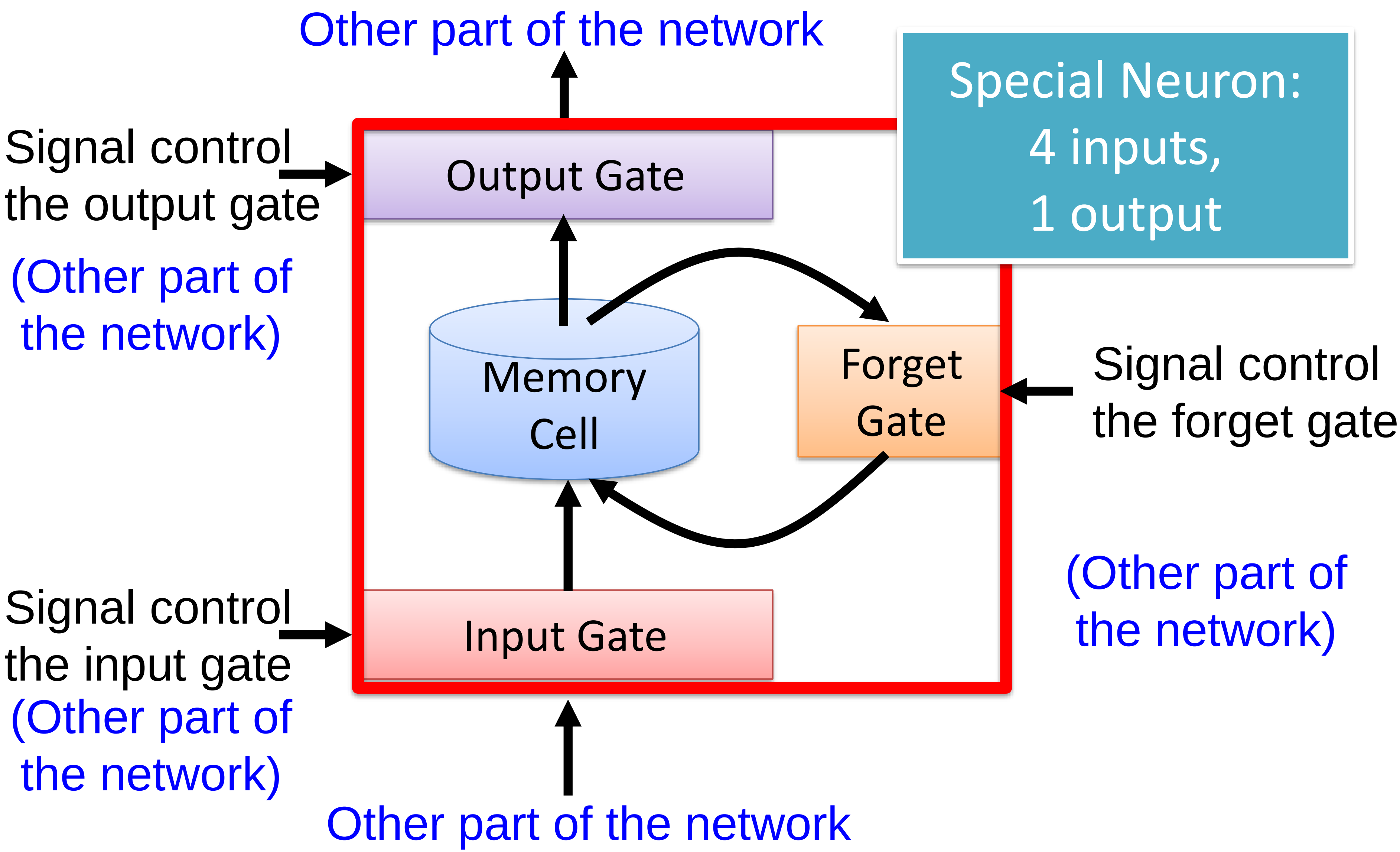
Happy Diwali



शुभ दीपावली

Machine Translation

Long Short-term Memory (LSTM)



Tuesday

- Starting with ML/DL -> Databases

•The Case for Learned Index Structures
Tim Kraska et al
SIGMOD 2018

•Lifting the Curse of Multidimensional Data with Learned Existence Indexes
Stephen Macke et al
NIPS 2018, MLSys: Workshop on Systems for ML and Open Source Software