EE2211 Introduction to Machine Learning

Lecture 12

Wang Xinchao
xinchao@nus.edu.sg

Course Contents

- Introduction and Preliminaries (Xinchao)
  - Introduction
  - Data Engineering
  - Introduction to Linear Algebra, Probability and Statistics
- Fundamental Machine Learning Algorithms I (Vincent)
  - Systems of linear equations
  - Least squares, Linear regression
  - Ridge regression, Polynomial regression
- Fundamental Machine Learning Algorithms II (Vincent)
  - Over-fitting, bias/variance trade-off
  - Optimization, Gradient descent
  - Decision Trees, Random Forest
- Performance and More Algorithms (Xinchao)
  - Performance Issues
  - K-means Clustering
  - Neural Networks

About this week's lecture…

- Neural Network (NN) is a very big topic
  - In NUS we have multiple full-semester modules to discuss NN
    - EE4305 Fuzzy/Neural Systems for Intelligent Robotics
    - EE5934/EE6934 Deep Learning
    - …
  - In EE2211, we only give a very gentle introduction

Deep Learning

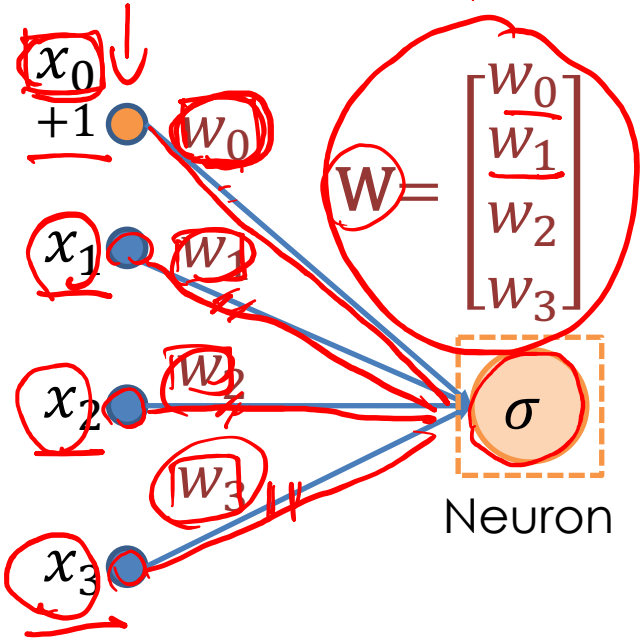
- Understanding at conceptual level is sufficient
  - In final exam, we have only 1 True/False + 1 MCQ about NN
  - No computation is required

- You will do some computation in tutorial, but final exam will be much simpler than the questions in tutorial

Outline

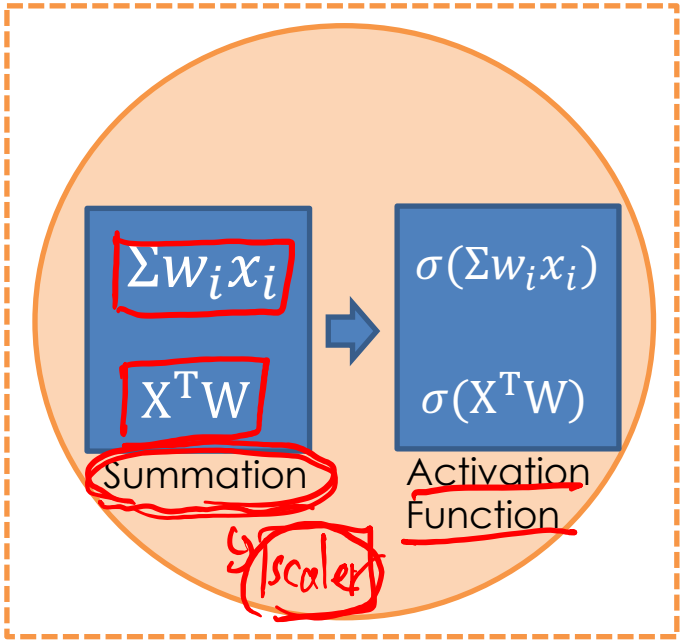
- Introduction to Neural Networks
  - Perceptron
  - Activation Functions
  - Multi-layer Perceptron
- Training and Testing of Neural Networks
  - Training: Forward and Backward
  - Testing: Forward
- Convolutional Neural Networks (CNNs)

Perceptron



$$\mathbf{W} = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ w_3 \end{bmatrix}$$

Neuron



Neuron

$$\mathbf{X} = \begin{bmatrix} 1 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

Output of Neuron: $\sigma(\mathrm{X^T W})$ or $\sigma(\Sigma w_i x_i)$

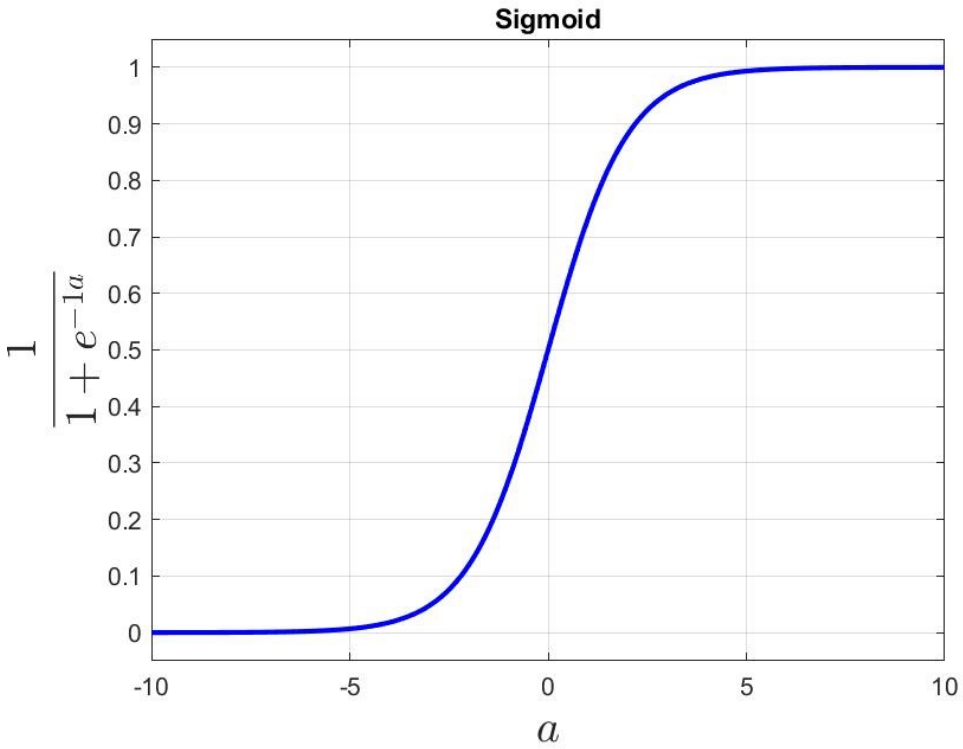
Activation Function: non-linear function to introduce non-linearity into the neural networks!

Goal of training: to learn W!

Activation Functions

Sigmoid Activation Function

$$\sigma(a) = \frac{1}{1 + e^{-\beta a}},$$

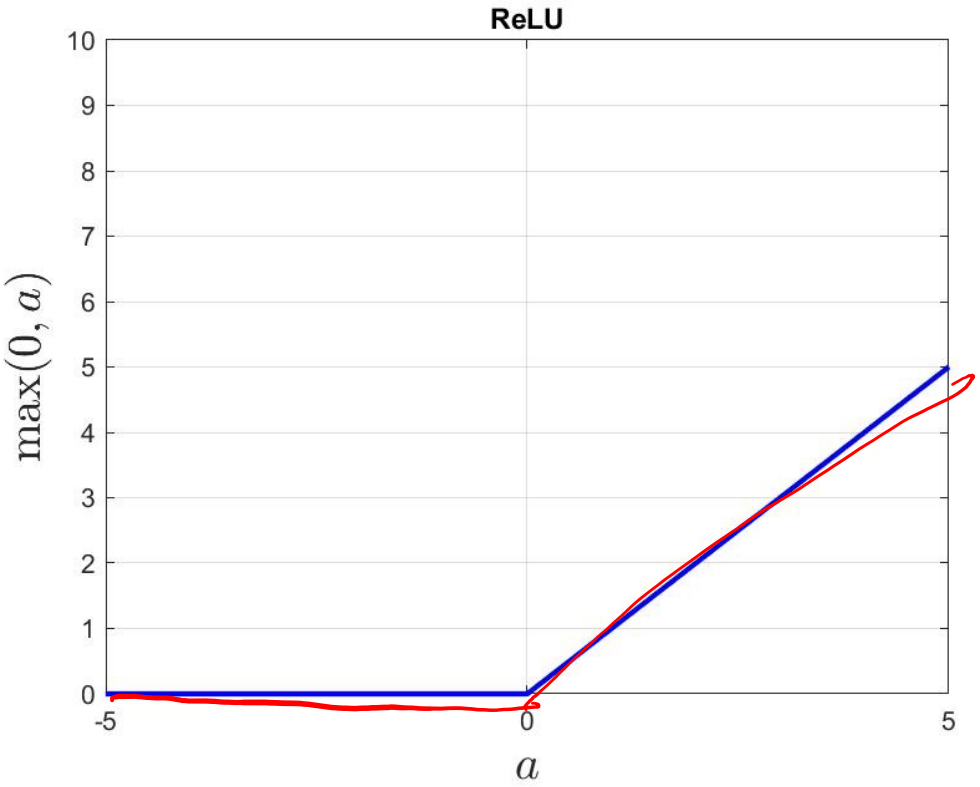


Activation Functions

ReLU Activation Function

$$\sigma(a) = \max(0, a)$$

Rectified Linear Unit (ReLU)



$a = 1.5$

$\sigma(a) = 1.5$

$a = -2$

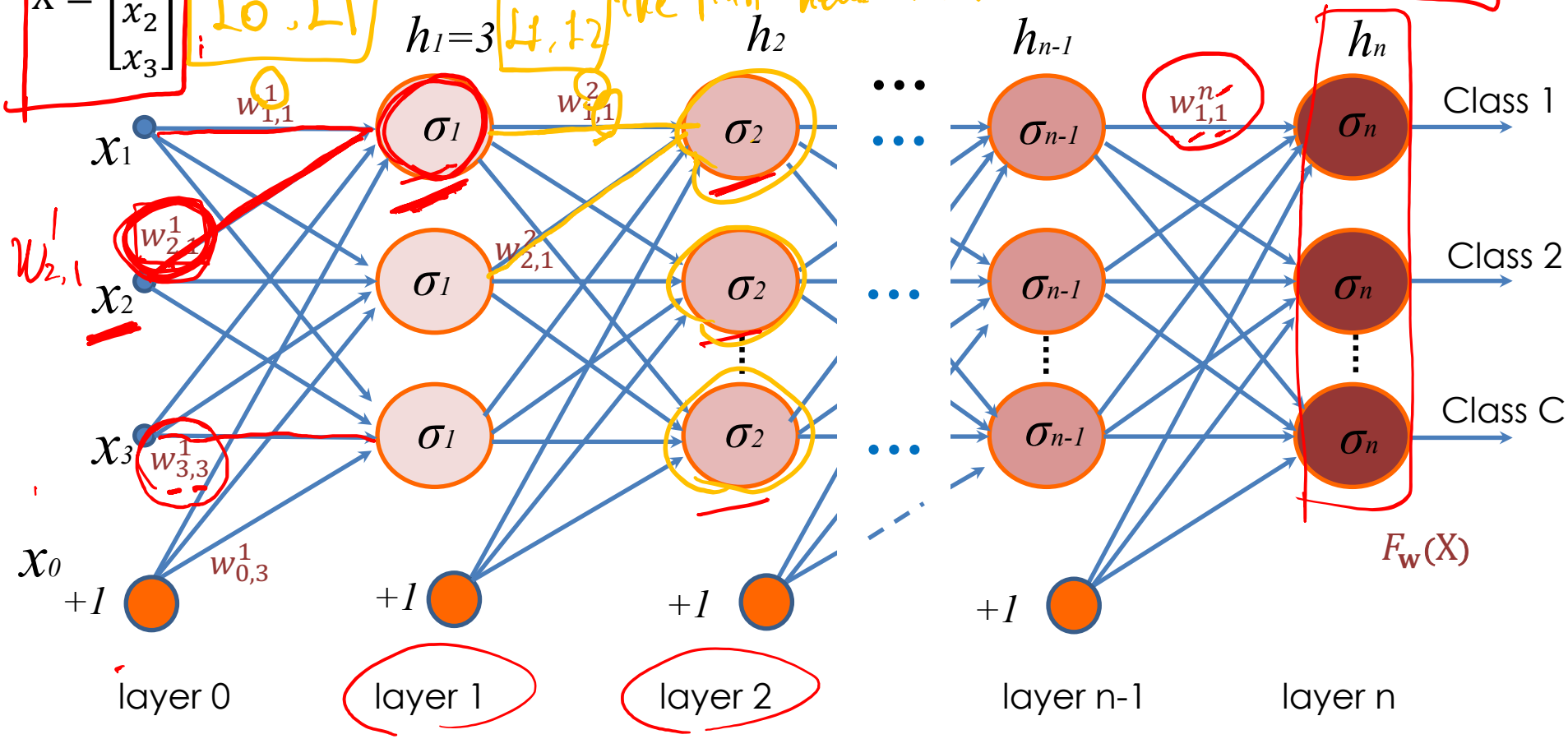
$\sigma(a) = 0$

Multilayer Perceptron (Neural Network)

$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

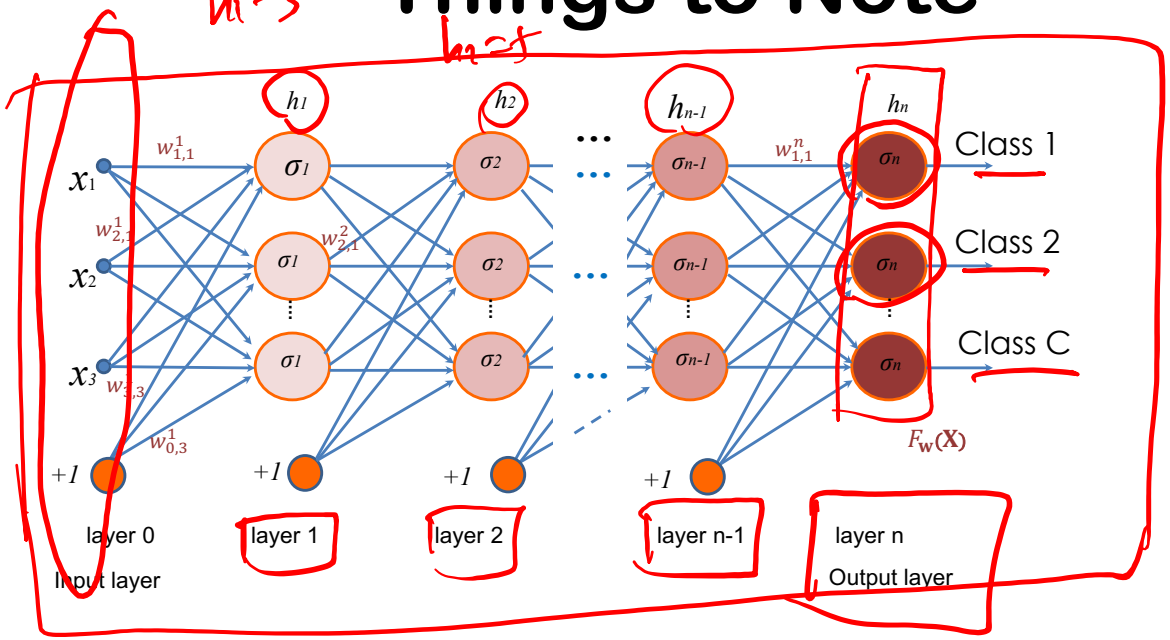
Goal of training: to learn W!

Nested function!



Note: h_n denotes the number of hidden neurons in layer n.

Things to Note



1. The number of hidden neurons in different layers may differ, i.e., h_1 don't have to be equal to h_2.
2. For **classification** task, the number of neurons in the last layer equals to the number of classes.
3. We can treat the whole network as a function $F_{\mathbf{w}}(\mathrm{X})$, where $\mathbf{w}$ is to be learned.

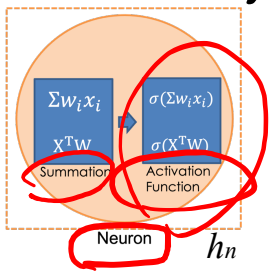
$F_{\mathbf{w}}(\mathrm{X})$

parameters input

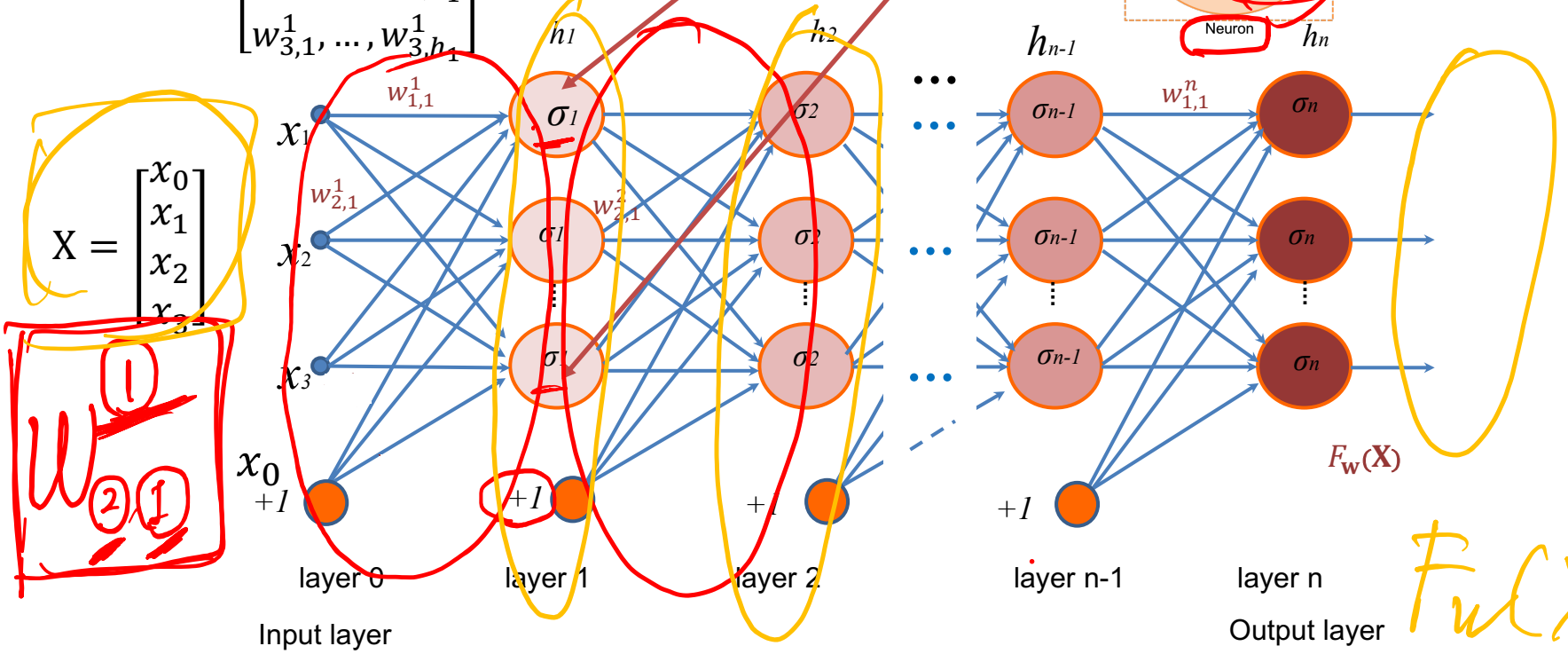
Multilayer Perceptron (Neural Network)

$$W^1 = \begin{bmatrix} w^1_{0,1}, \dots, w^1_{0,h_1} \\ w^1_{1,1}, \dots, w^1_{1,h_1} \\ w^1_{2,1}, \dots, w^1_{2,h_1} \\ w^1_{3,1}, \dots, w^1_{3,h_1} \end{bmatrix}$$

$$X^T W^1 = [\Sigma w^1_{i,1} x_i, \dots, \Sigma w^1_{i,h_1} x_i]$$



$$X = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix}$$



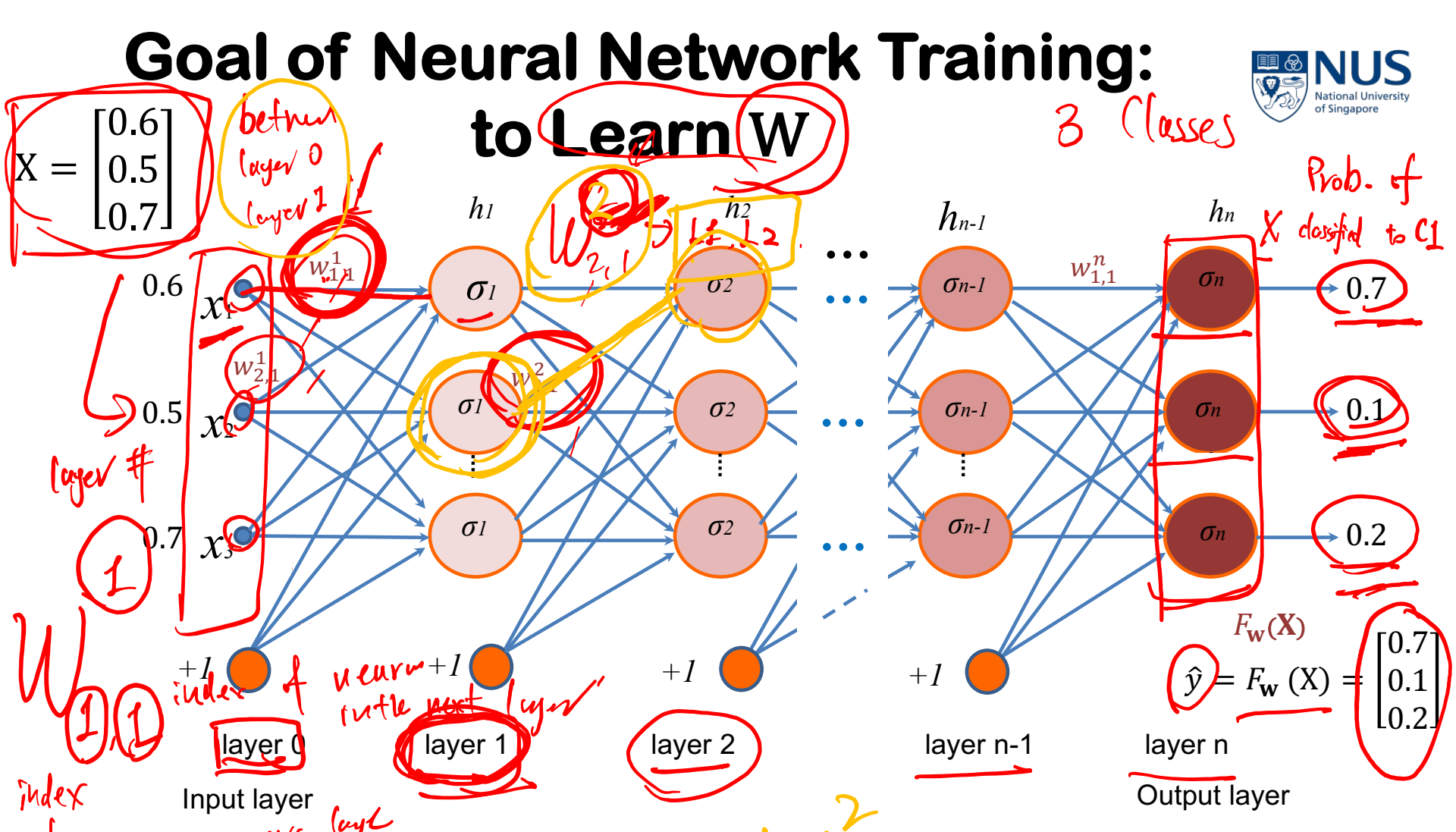
A **neural network** is essentially a **nested function**.

$$F_W(X) = \sigma([1, \dots \sigma([1, \sigma(X^T W^1)]\ W^2) \dots]\ W^n)$$

Outline

- Introduction to Neural Networks
  - Multi-layer perceptron
  - Activation Functions
- Training and Testing of Neural Networks
  - Training: Forward and Backward
  - Testing: Forward
- Convolutional Neural Networks

Goal of Neural Network Training: to Learn W

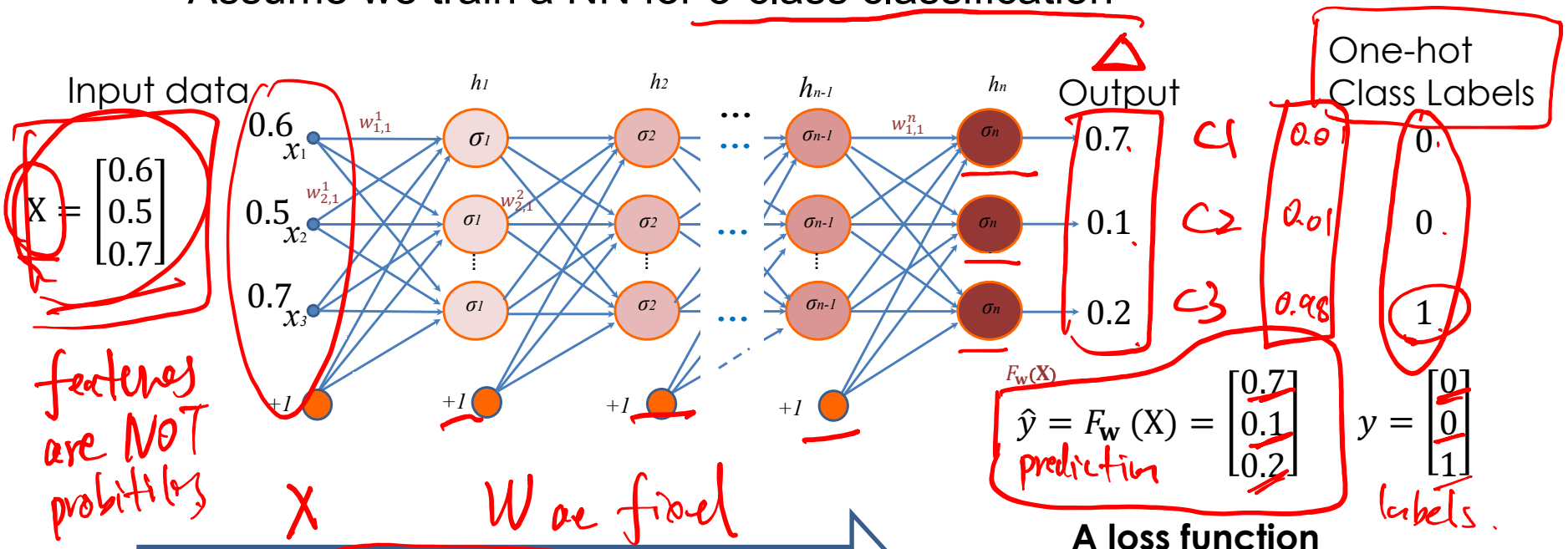


Specifically, W is learned through

1. Random initialization
2. **Backpropagation**

Neural Network Training: Backpropagation

Assume we train a NN for 3-class classification



features are NOT probability

X

W are fixed

1. Forward: (weights are fixed)
To compute network responses
To compute the errors at each output

prediction

loss

2. Backward: (weights are updated)
To pass back the error from the output to the hidden layers
To update all weights to optimize the network

A loss function for a single sample:

$\min_{\mathbf{w}} \sum_{i=1}^{C} (\hat{y}_i - y_i)^2$

or

$\min_{\mathbf{w}} ||\hat{y} - y||^2$

Update W!

Neural Network Training: Backpropagation

- Recall that the parameters W are randomly initialized.
- We use **Backpropagation** to update W.
- In essence, **Backpropagation** is gradient descent!
- Assume we have N samples, each sample denoted by X^j and the output of NN by $\hat{y}^j$, loss function is then

$$J = \sum_{j=1}^{N} \left\| \hat{y}^j - y^j \right\|^2, \quad \min_{\mathbf{w}} J$$

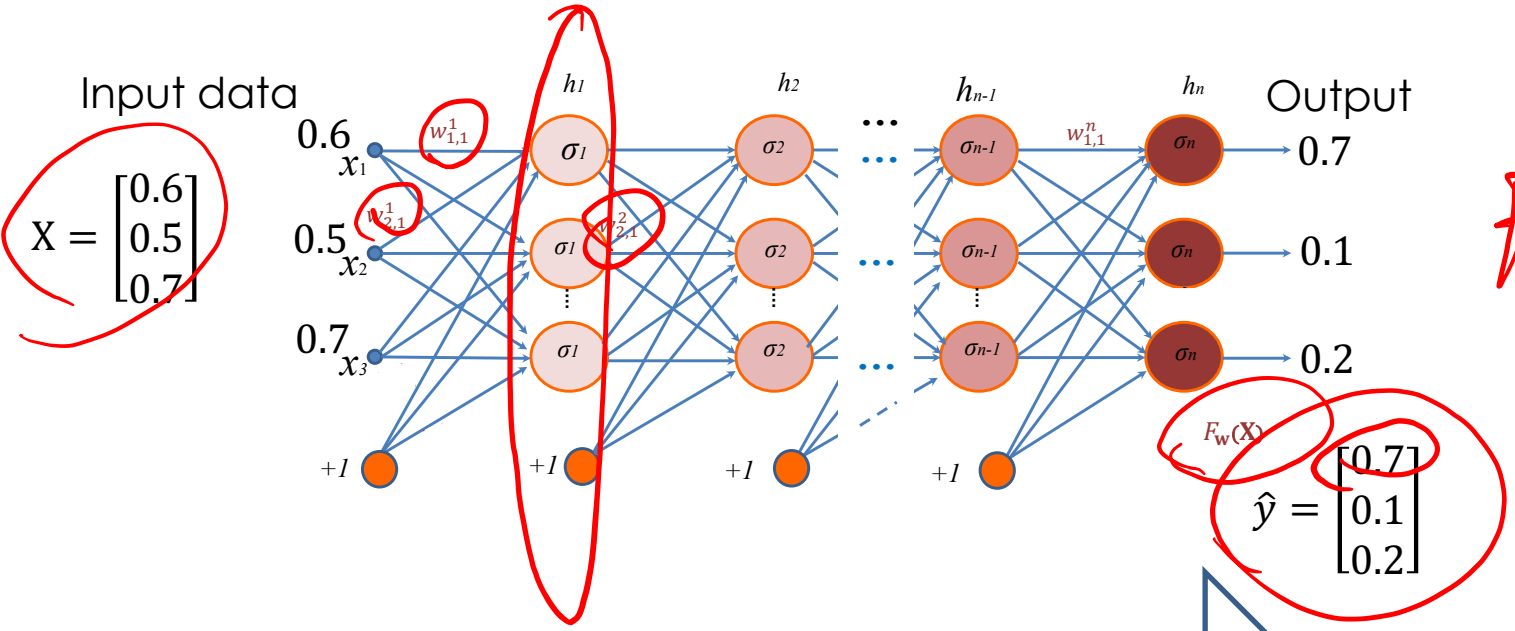
Recall gradient descent in Lec 8: $\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} J$

- We would therefore like to compute $\nabla_{\mathbf{w}} J$!
  - J is a function of $\hat{y}$, and $\hat{y}$ is a function of $\mathbf{w}$, i.e., $\hat{y} = F_{\mathbf{w}}(\mathrm{X})$
  - Use gradient descent and chain rule!

Being aware of the concept is sufficient for exam. No calculation needed.

Neural Network Testing

Once all network is trained and parameters are updated



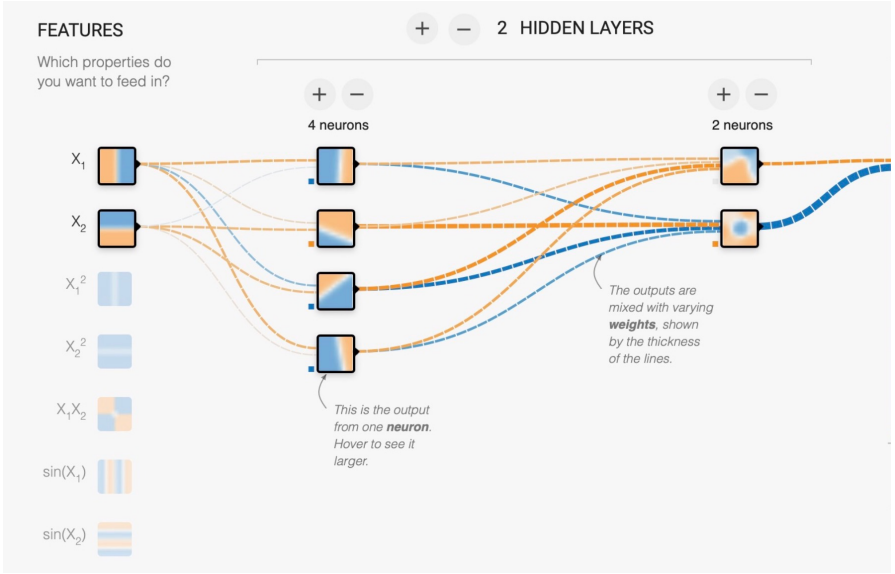
$F_w(X)$

1. **Forward**: (weights are fixed)
To estimate compute network responses
To predict the output labels given novel inputs

Python Demo

```
[ ]  from sklearn.neural_network import MLPClassifier
     # Train the neural network classifier with 3 hidden layers of 200 neurons each
     clf = MLPClassifier(hidden_layer_sizes=(200, 200, 200), activation="relu", learning_rate="invscaling", verbose=True)
     clf.fit(X, y)
     plot_clf(clf, X, y)
```

- Python Code (A simple multi-layer neural network classifier)
  - lec12.ipynb
- Demo
  - https://playground.tensorflow.org/



Supplementary materials (Not required for exam)

1) https://www.youtube.com/watch?v=tIeHLnjs5U8

This video series includes animations that explain backpropagation calculus.

2) https://www.youtube.com/playlist?list=PLQVvvaa0QuDcjD5BAw2DxE6OF2tius3V3

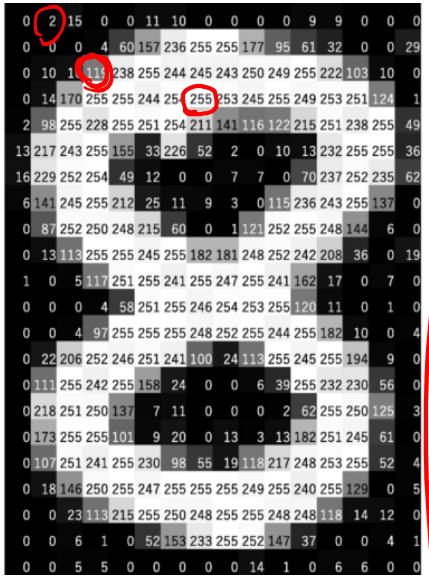
This video series includes hands-on coding examples in Python.

Outline

- Introduction to Neural Networks
  - Multi-layer perceptron
  - Activation Functions
- Training and Testing of Neural Networks
  - Training: Forward and Backward
  - Testing: Forward
- Convolutional Neural Networks

Convolutional Neural Network (CNN)

- A convolutional neural network (CNN) is a special type of neural network that significantly reduces the number of parameters in a deep neural network.
- Very popular in image-related applications
- Each image is stored as a matrix in a computer

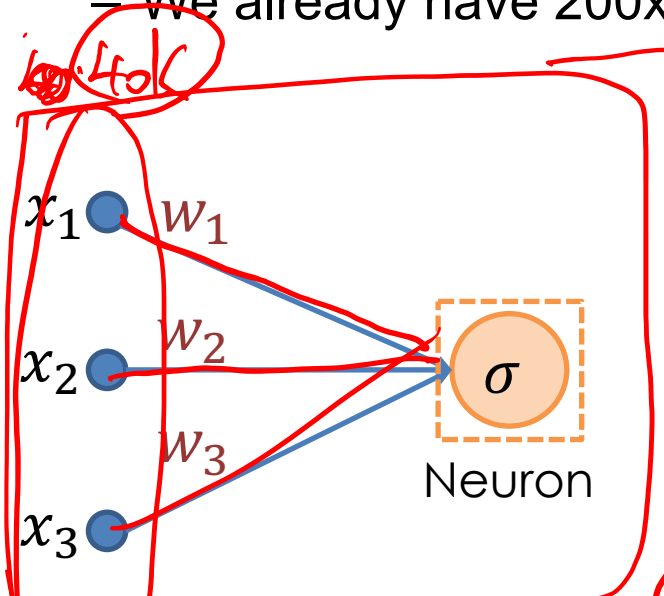


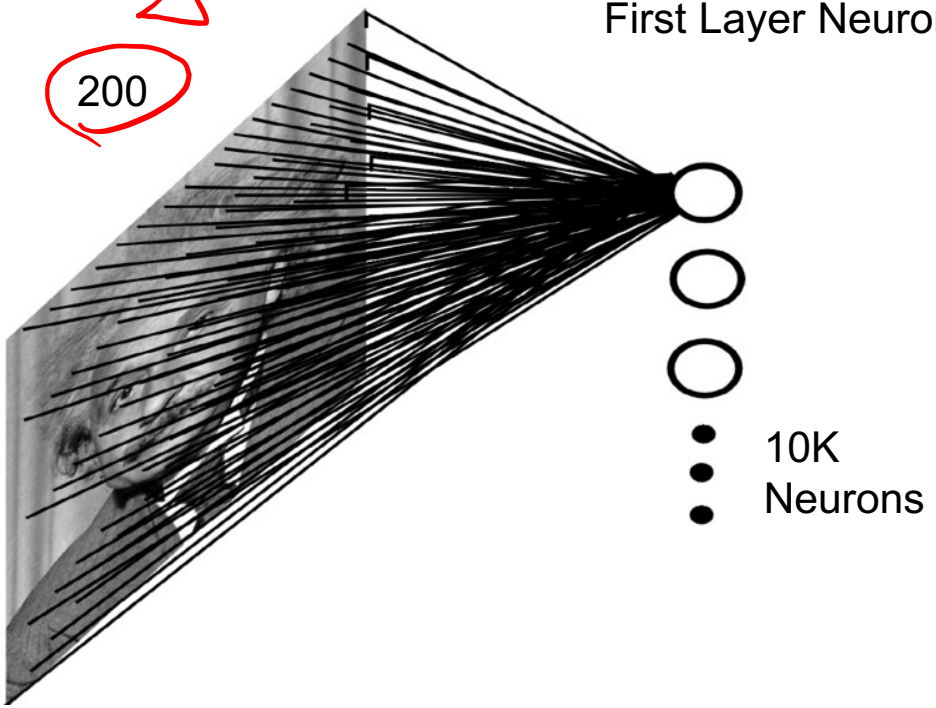
0	2	15	0	0	11	10	0	0	0	0	9	9	0	0	0
0	0	0	4	60	157	236	255	255	177	95	61	32	0	0	29
0	10	16	119	238	255	244	245	243	250	249	255	222	103	10	0
0	14	170	255	255	244	254	255	253	245	255	249	253	251	124	1
2	98	255	228	255	251	254	211	141	116	122	215	251	238	255	49
13	217	243	255	155	33	226	52	2	0	10	13	232	255	255	36
16	229	252	254	49	12	0	0	7	7	0	70	237	252	235	62
6	141	245	255	212	25	11	9	3	0	115	236	243	255	137	0
0	87	252	250	248	215	60	0	1	121	252	255	248	144	6	0
0	13	113	255	255	245	255	182	181	248	252	242	208	36	0	19
1	0	5	117	251	255	241	255	247	255	241	162	17	0	7	0
0	0	0	4	58	251	255	246	254	253	255	120	11	0	1	0
0	0	4	97	255	255	255	248	252	255	244	255	182	10	0	4
0	22	206	252	246	251	241	100	24	113	255	245	255	194	9	0
0	111	255	242	255	158	24	0	0	6	39	255	232	230	56	0
0	218	251	250	137	7	11	0	0	0	2	62	255	250	125	3
0	173	255	255	101	9	20	0	13	3	13	182	251	245	61	0
0	107	251	241	255	230	98	55	19	118	217	248	253	255	52	4
0	18	146	250	255	247	255	255	255	249	255	240	255	129	0	5
0	0	23	113	215	255	250	248	255	255	248	248	118	14	12	0
0	0	6	1	0	52	153	233	255	252	147	37	0	0	4	1
0	0	5	5	0	0	0	0	0	0	14	1	0	6	6	0	0

https://medium.com/lifeandtech/convert-csv-file-to-images-309b6fdb8c49

Convolutional Neural Network (CNN)

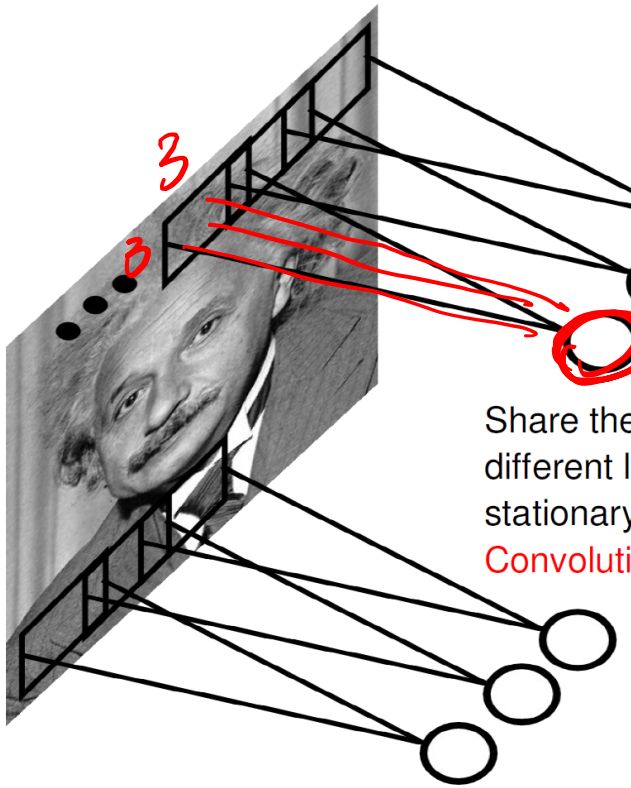
- If we model all matrix entries as inputs all at once
  - Assume we have an image/matrix size of 200x200
  - Assume we have 10K neuros in the first layer
  - We already have 200x200x10K=400 Million parameters to learn!





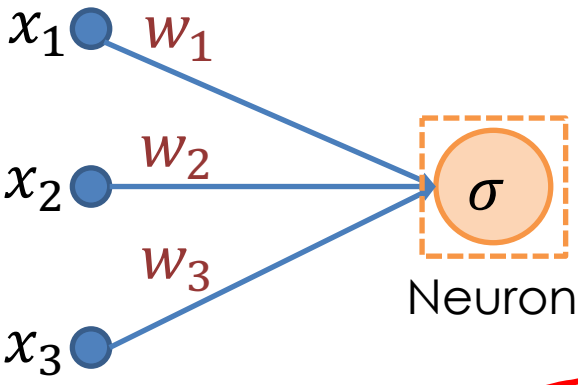
Convolutional Neural Network (CNN)

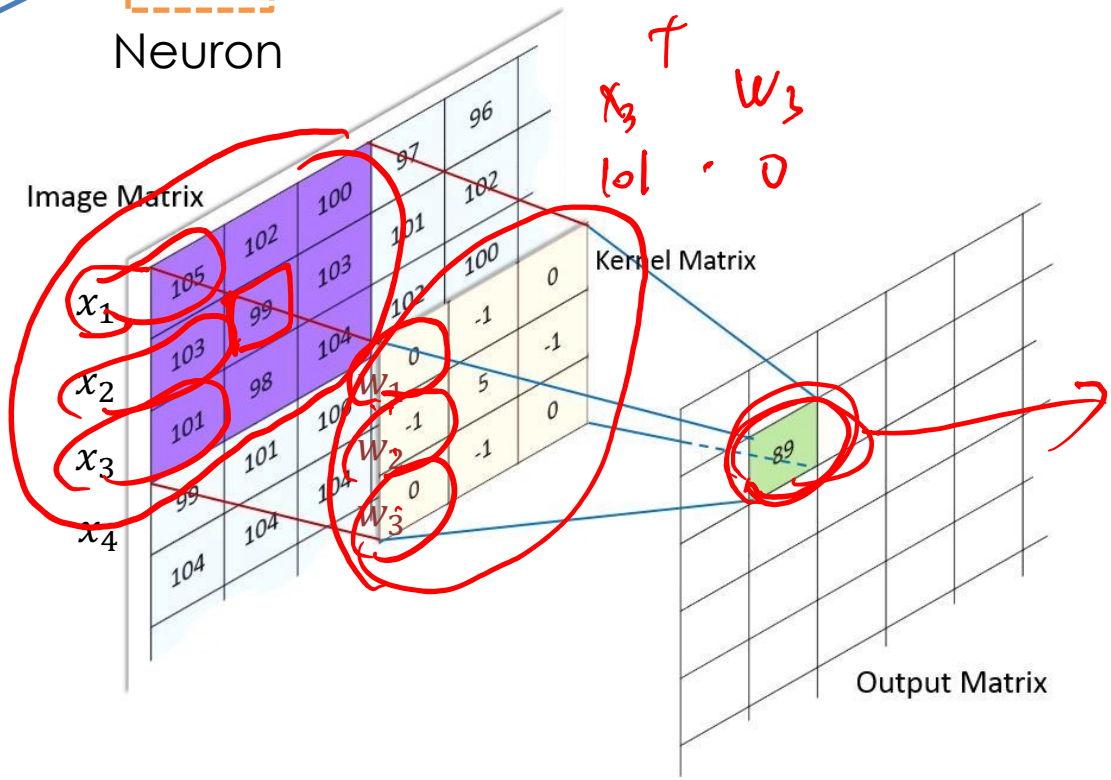
- Hence, we introduce CNN to reduce the number of parameters.
- Works in **a sliding-window manner**!

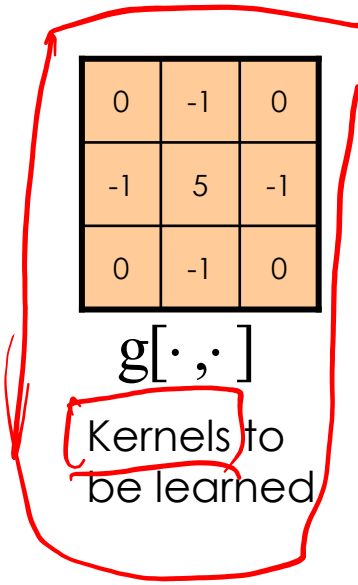


Share the same parameters across different locations (assuming input is stationary):
Convolutions with learned kernels

Convolutional Neural Network (CNN)



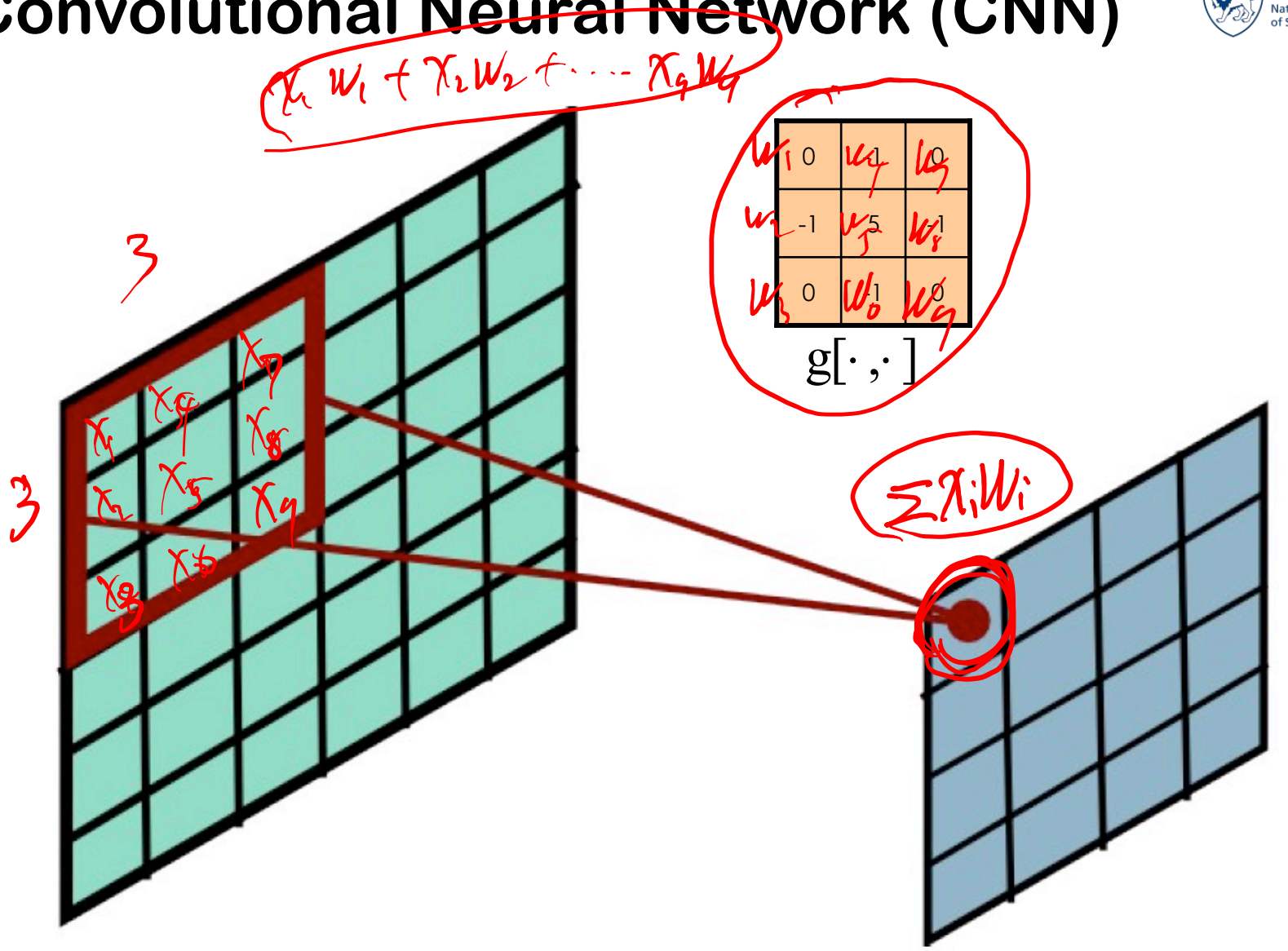




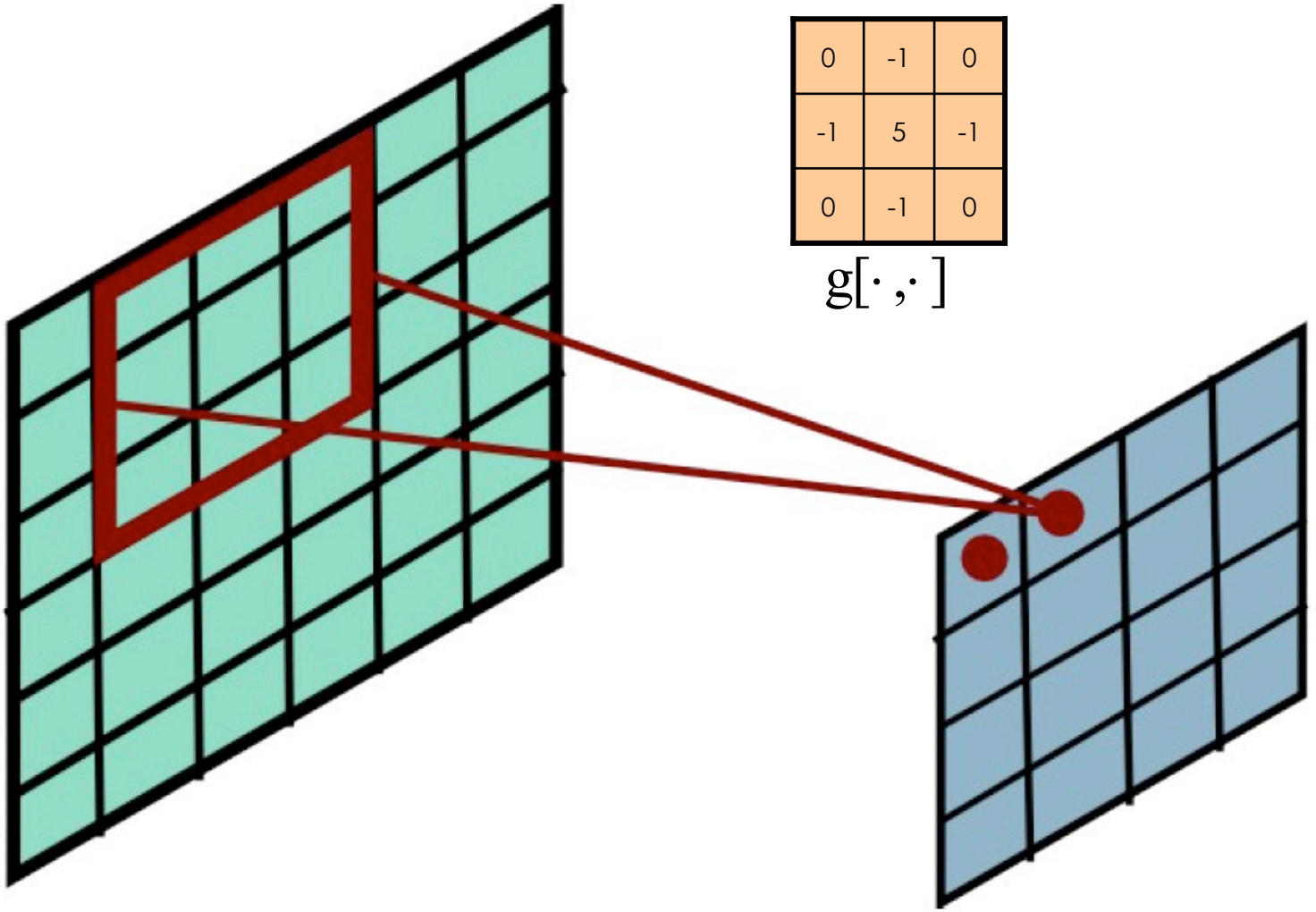
center of the local neighbourhood

Image source: https://brilliant.org/wiki/convolutional-neural-network/

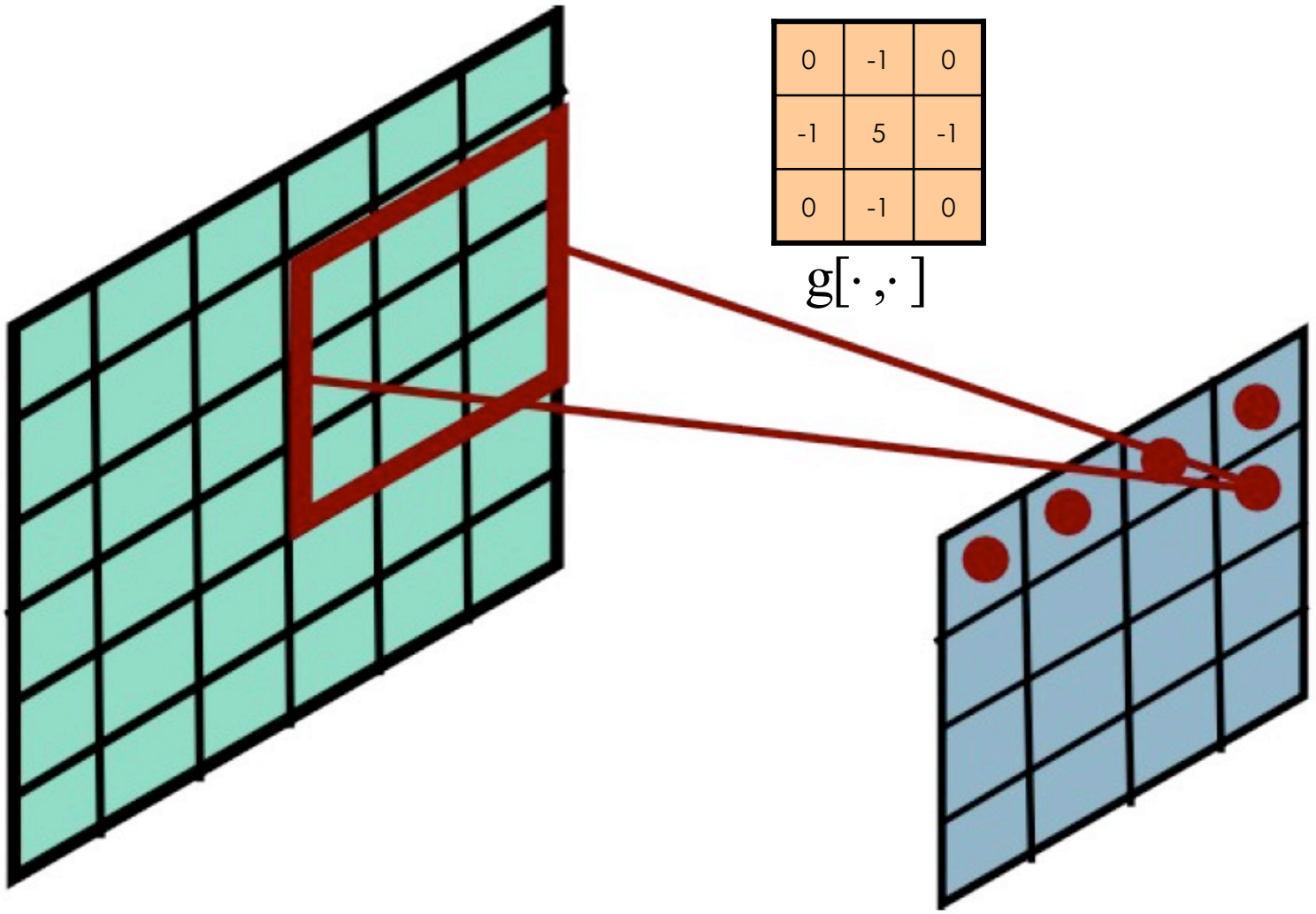
Convolutional Neural Network (CNN)



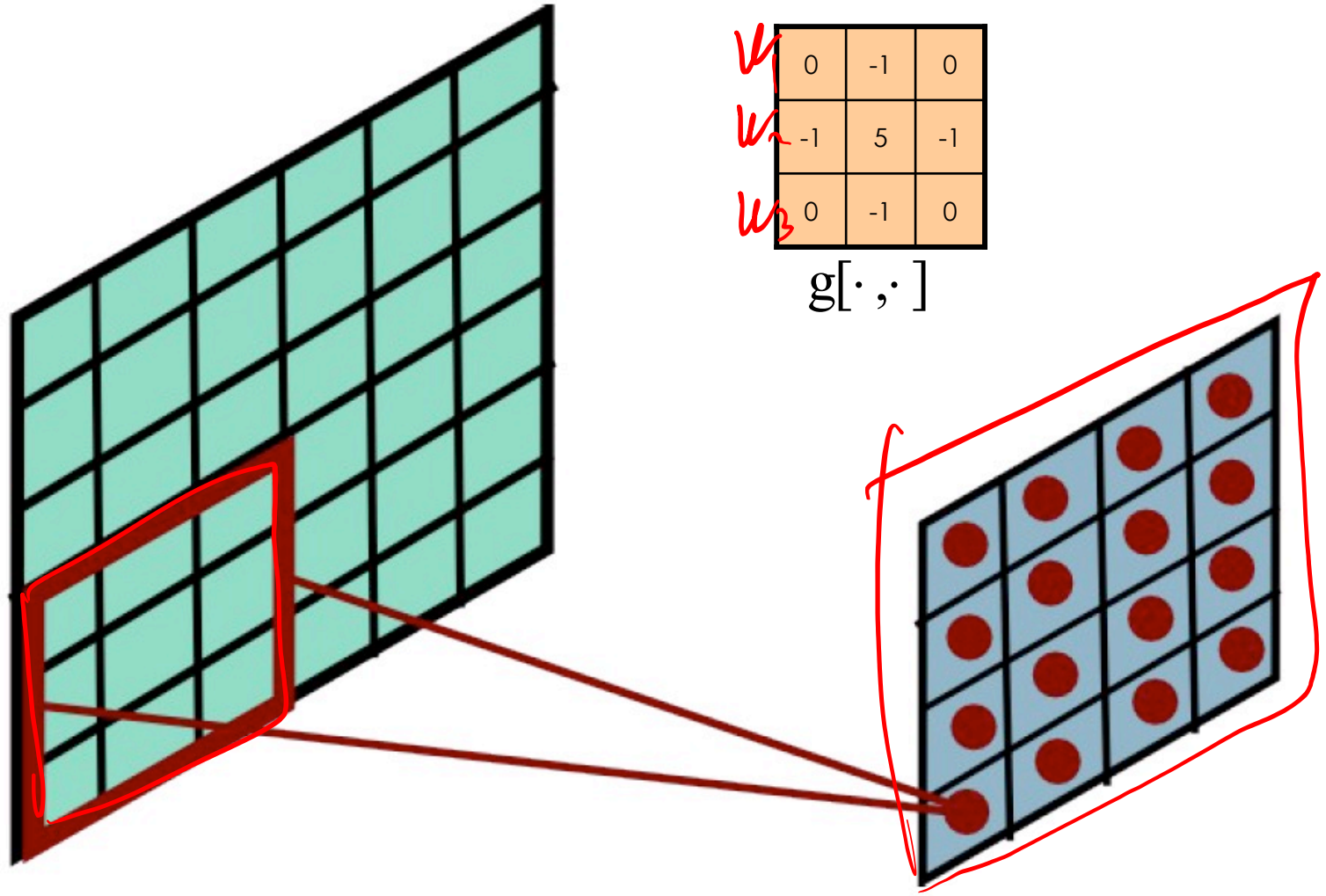
Convolutional Neural Network (CNN)



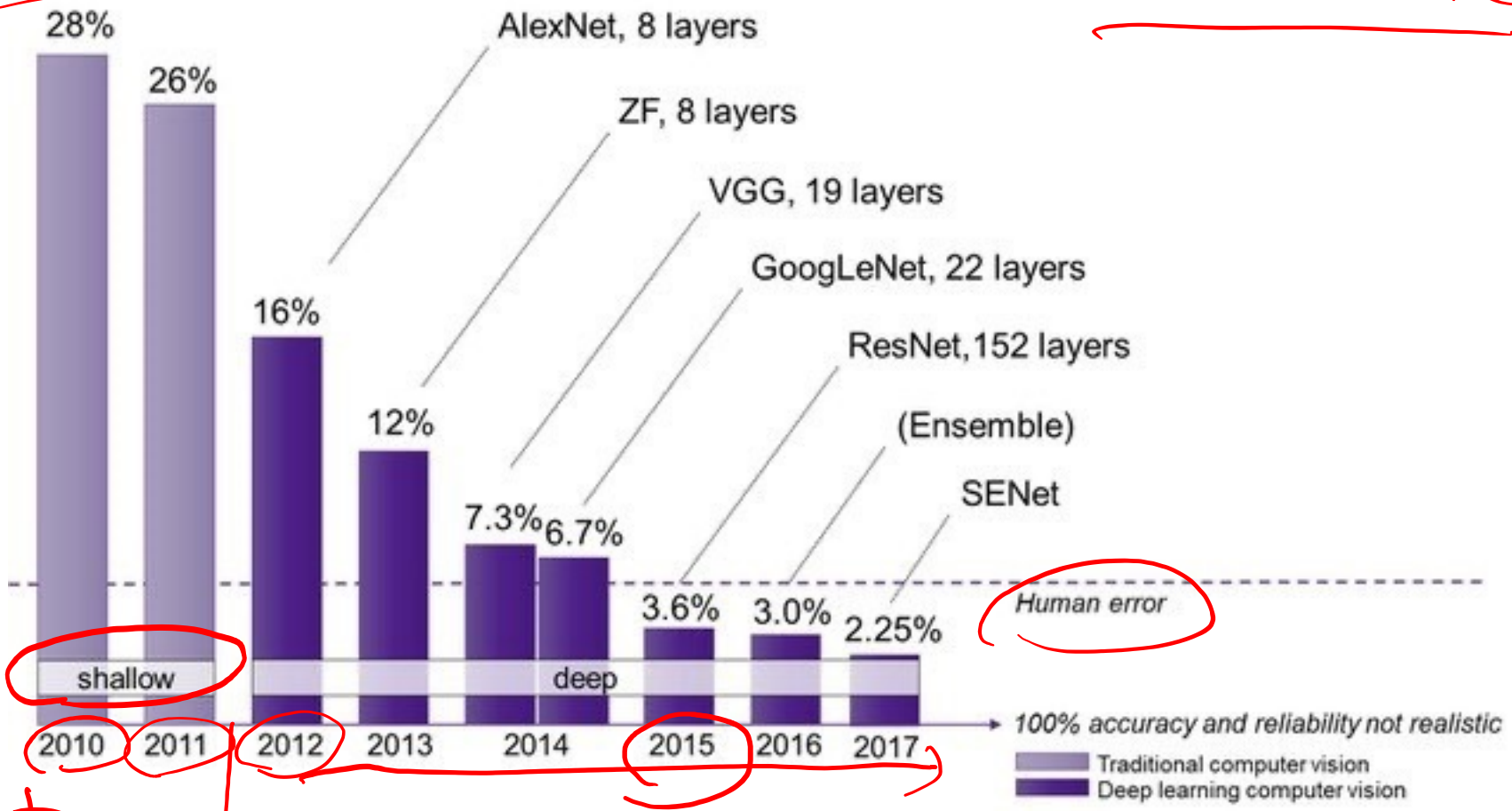
Convolutional Neural Network (CNN)



Convolutional Neural Network (CNN)



Neural Networks are Effective



Summary

- Introduction to Neural Networks
  - Multi-layer perceptron
  - Activation Functions
- Training and Testing of Neural Networks
  - Training: Forward and Backward
  - Testing: Forward
- Convolutional Neural Networks

(CNN)

Thanks everyone, for your time and effort throughout the semester!