## Big Data Analysis

**Deep Learning: CNN (Convolutional Neural Network)**

**Ji-Hyeong Han**

## ( [jhhan@seoultech.ac.kr](mailto:jhhan@seoultech.ac.kr) )

**Dept. of Computer Science and Engineering**

Ref: 모두를 위한 딥러닝 (김성훈 교수)

**Contents**

1. Neural Network
2. Turning point of NN
   1. New activation function (ReLU)
   2. New weight initialization method
   3. Convolutional Neural Network (CNN)

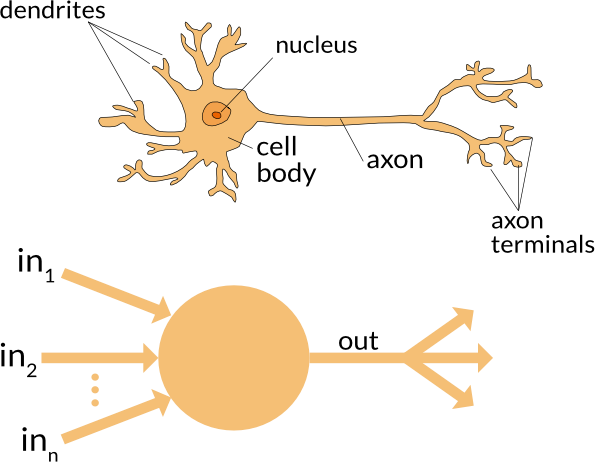
2

# Neural Network

3

**Neural Networks History**

* Origins
  + Algorithms that try to mimic the brain (neuron in the brain)

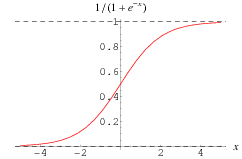


* NN was very widely used in 80s and early 90s
* Popularity diminished in late 90s

4

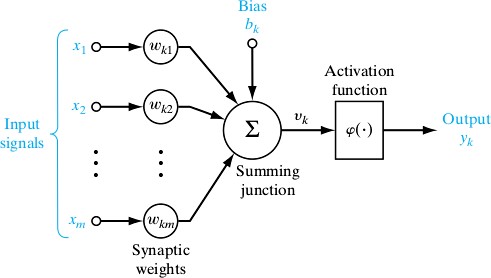
* Logistic unit
  + Using sigmoid activation function, 𝑔

1

= 1+𝑒−𝑧

𝑧

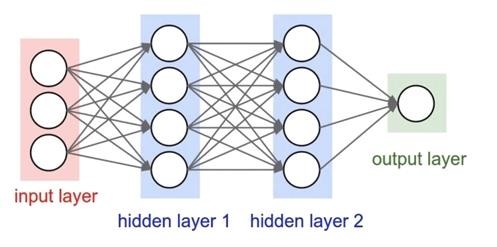
**Neuron Model**



5

**Neural Network Model**

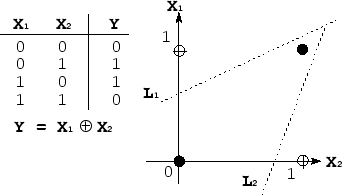
* Multilayer Neural Network



6

**XOR problem**

* The representative non-linear classification example



* Can we solve this problem by using NN?
  + The multilayer NN can represent XOR problem ?

7

* NN for XOR Problem

𝑤11

𝑤12

= 5 , 𝑏1 = −8

5

𝑤21

𝑏1

𝑥1

𝑤11

𝑤12

𝑏

𝑦

ℎ1

𝑤ℎ1𝑦

𝑤21

𝑦

𝑥2

𝑤22

𝑏2

ℎ2

𝑤ℎ2𝑦

𝑤22

𝑤ℎ1𝑦

𝑤ℎ2𝑦

= −7

−7

= −11

−11

, 𝑏2 = 3

, 𝑏𝑦 = 6

input layer

hidden layer

output layer

**XOR Problem**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| x1 | x2 | h1 | h2 | y | XOR |
| 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 |

8

* Yes, we can represent XOR problem by multilayer NN
* This is forward propagation
* Is there another W and b for XOR problem?
  + Yes!

𝑏1

𝑥

1

𝑤11

𝑤12

𝑏

𝑦

ℎ

1

𝑤

ℎ1𝑦

𝑤21

𝑦

𝑥2

𝑤22

𝑏2

ℎ2

𝑤ℎ2𝑦

input layer

hidden layer

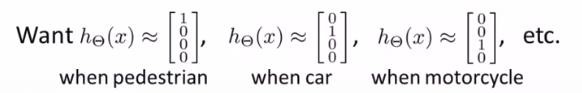
output layer

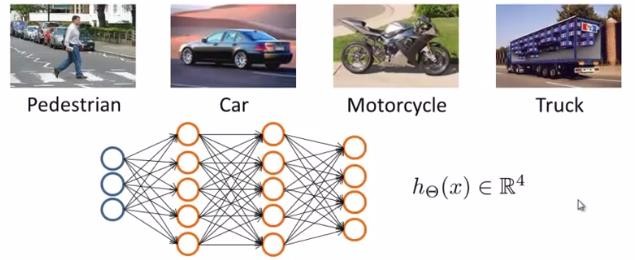
**XOR Problem**

9

**Multiple output units: one-vs-all**

* Multiclass classification

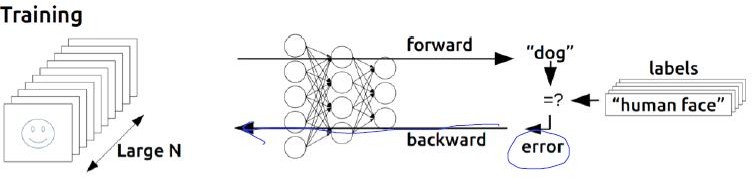




10

**Backpropagation**

* How can we learn W and b from training data?
  + Using gradient descent of error
  + How can we calculate partial derivative of error by W and b?
    - Chain rule
  + How can we update W and b?
    - Error backpropagation
* Error backpropagation !!



11

* We need to derivative the error for multi layers and multi neurons
  + How??
  + Using chain rule

𝑓(𝑔

) 𝜕𝑓=𝜕𝑓

∙ 𝜕𝑔

𝜕𝑥

𝑥

* Simple example

𝜕𝑔

𝜕𝑥

𝑓 = 𝑤𝑥 + 𝑏, 𝑔 = 𝑤𝑥, 𝑓 = 𝑔 + 𝑏

w 1. forward (w=-2, x=5, b=3)

𝜕𝑓

𝜕𝑤

g

\*

𝜕𝑓

𝜕𝑓

𝜕𝑥

𝜕𝑔

+

f

b

𝜕𝑓

→ g = -10, f = -7

2. backward

→ 𝜕𝑓=1, 𝜕𝑓=1

x

**Backpropagation**

𝜕𝑏

𝜕𝑔

→ 𝜕𝑓

𝜕𝑤

→ 𝜕𝑓

𝜕𝑥

𝜕𝑏

= 𝜕𝑓 𝜕𝑔

𝜕𝑔 𝜕𝑤

= 𝜕𝑓 𝜕𝑔

𝜕𝑔 𝜕𝑥

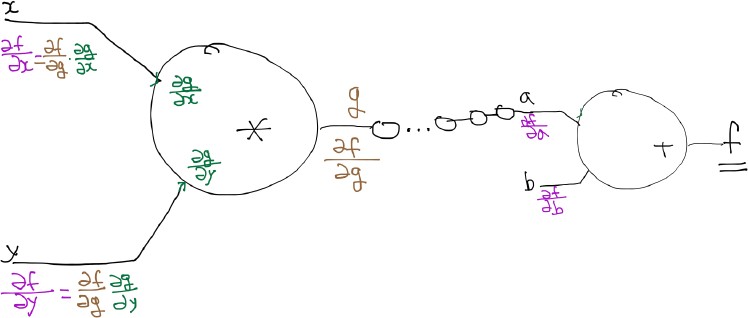
= 1 ∗ 𝑥 = 5

= 1 ∗ 𝑤 = -2

12

**Backpropagation**

* Multilayer back propagation (chain rule)



Back propagation 13

# Turning points of NN

## New activation function: ReLU

* + 1. **New weight initialization method**

14

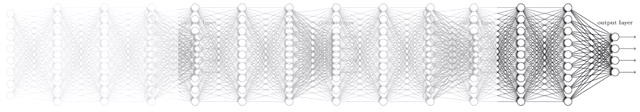
**The problem of NN**

* After late 90s, the NN has dark period
  + Because, it did not work well
* Now the reasons are found
  + Our labeled datasets were thousands of times too small
    - Solved by big data
  + Our computers were millions of times too slow
    - Solved by GPU
  + We used the wrong type of non-linearity (=sigmoid function)
    - Solved by new activation function (=ReLU)
  + We initialized the weights in a stupid way
    - Solved by Xavier initialization method

15

**New Activation Function: ReLU**

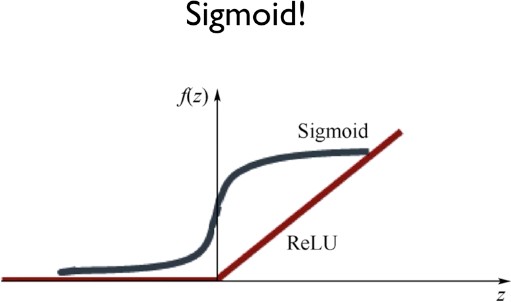
* The problems of sigmoid function
  + Vanishing gradient problem when the network is deep
    - 2~3 layers are OK when we use backpropagation
    - But deep network is not learned when we use backpropagation
    - Because we use sigmoid function as activation function



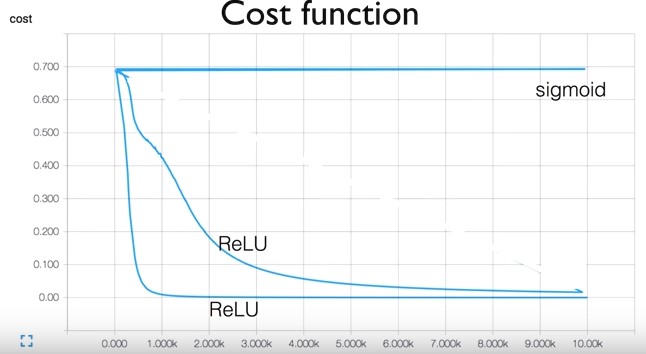
16

**New Activation Function: ReLU**

* New activation function: ReLU!!



* + ReLU: Rectified Linear Unit

17

**Initializing Weights in a Smart Way**

* Need to set the initial weight values wisely
  + Not all 0’s
  + Challenging issue
* Xavier initialization
  + Makes sure the weights are ‘just right’, not too small, not too big
  + Using number of input (fan\_in)



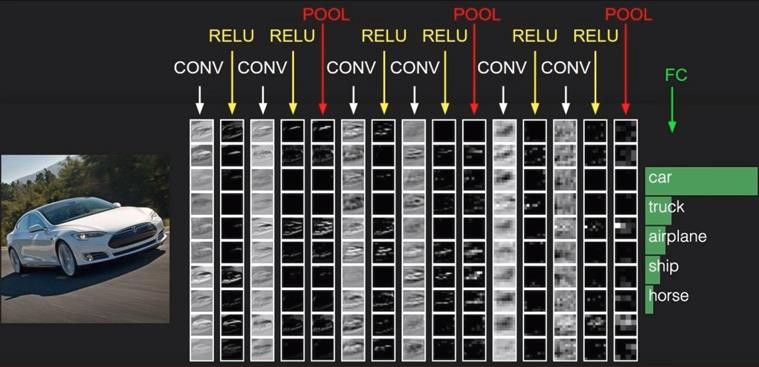
18

# Convolutional Neural Network (CNN)

19

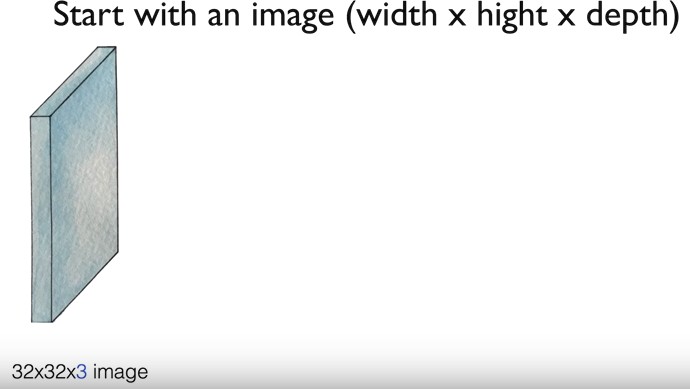
**Convolutional Neural Network (CNN)**

* Basic structure of CNN



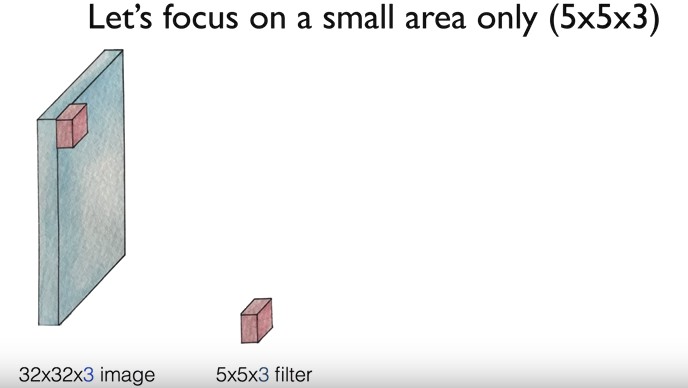
20

**CNN Process**



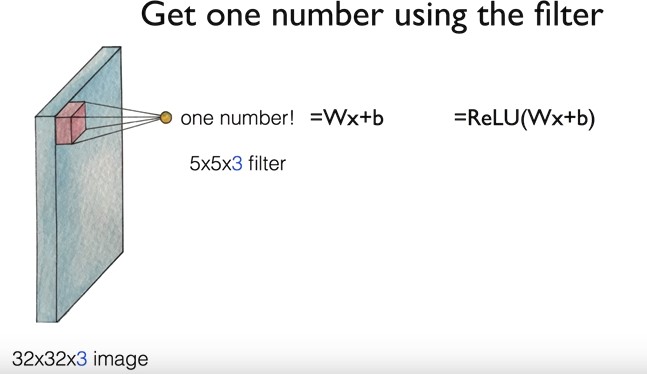
21

**CNN Process**



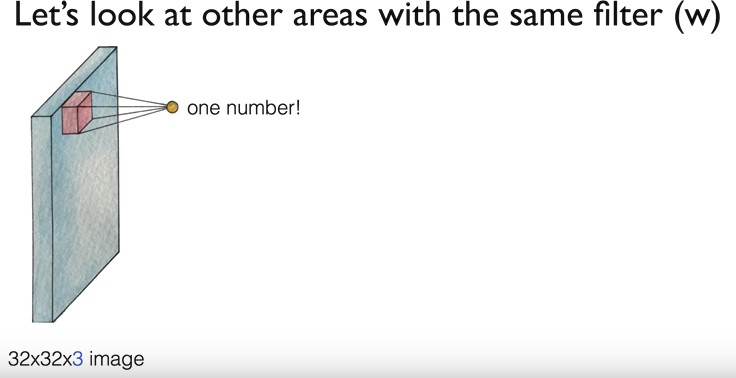
22

**CNN Process**



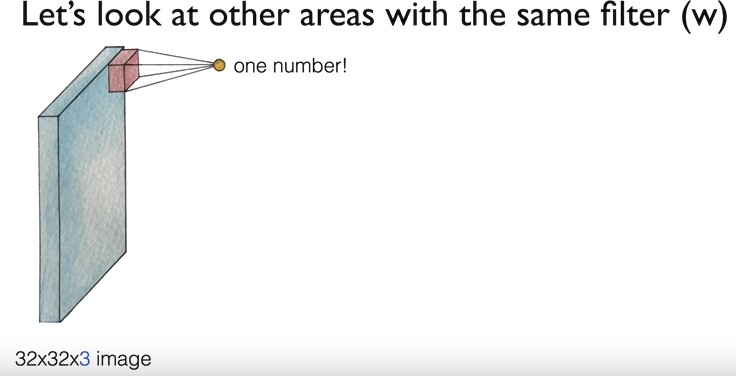
23

**CNN Process**



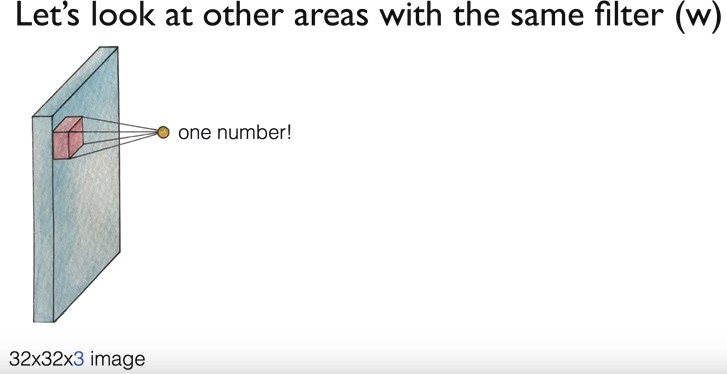
24

**CNN Process**



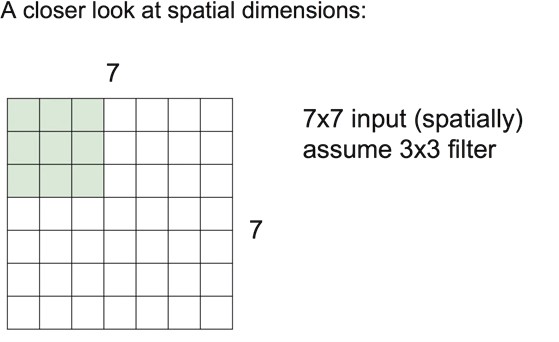
25

**CNN Process**



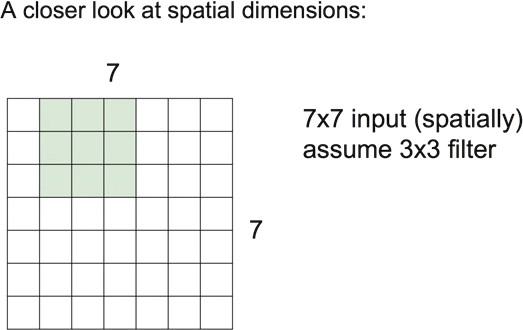
26

**CNN Process**



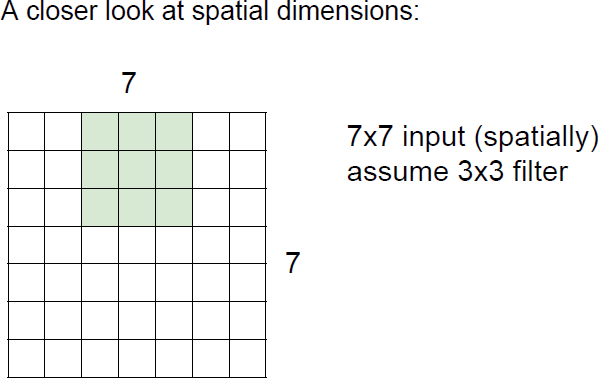
27

**CNN Process**



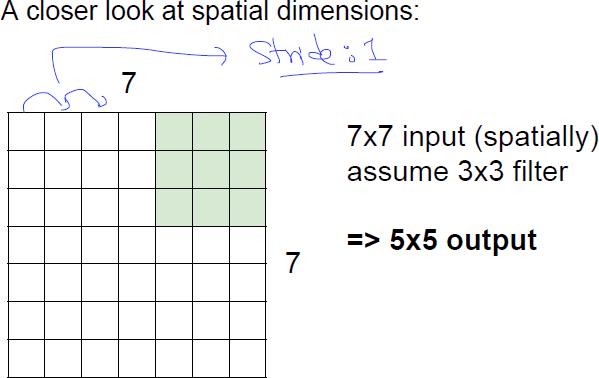
28

**CNN Process**



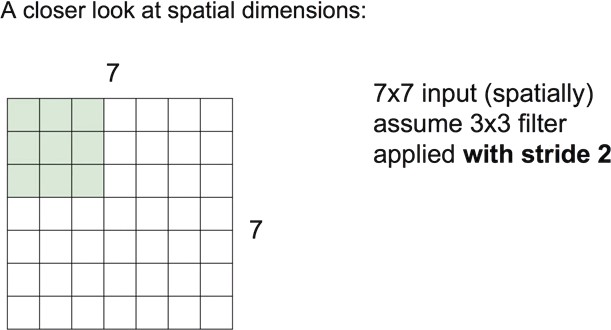
29

**CNN Process**



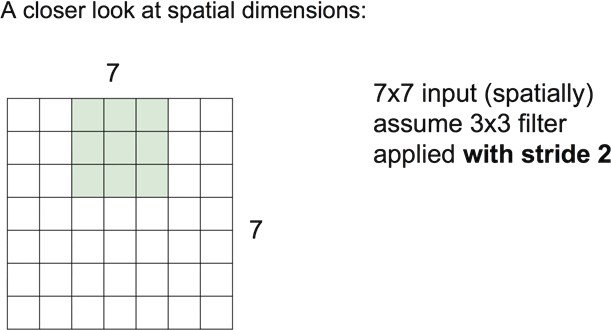
30

**CNN Process**



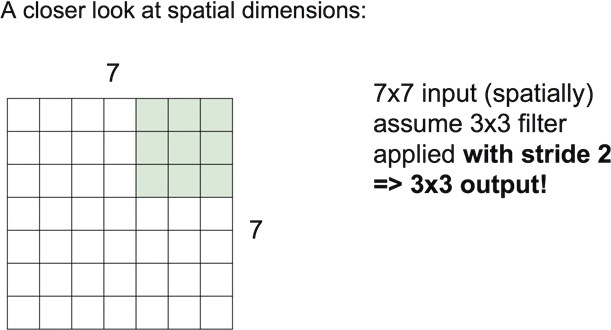
31

**CNN Process**



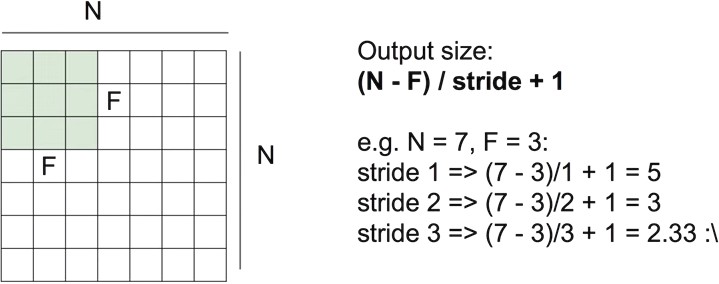
32

**CNN Process**



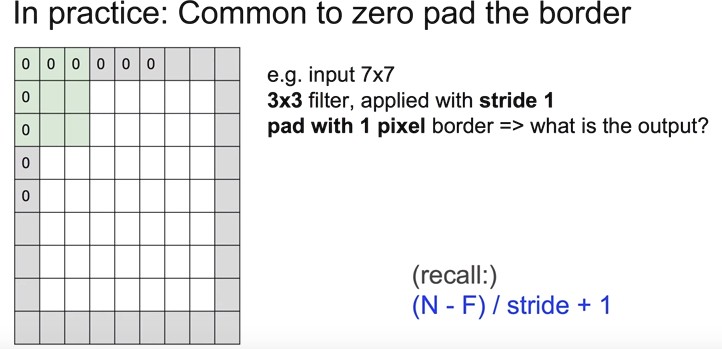
33

**CNN Process**



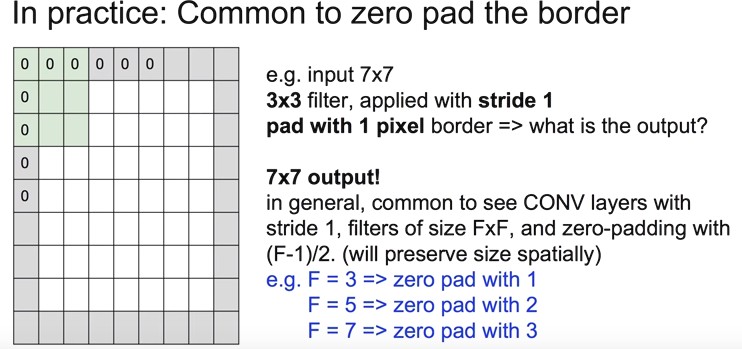
34

**CNN Process**



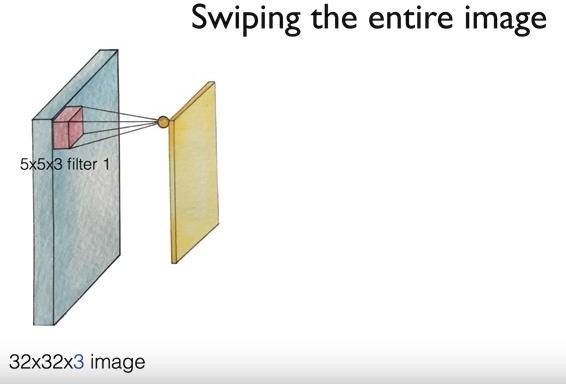
35

**CNN Process**



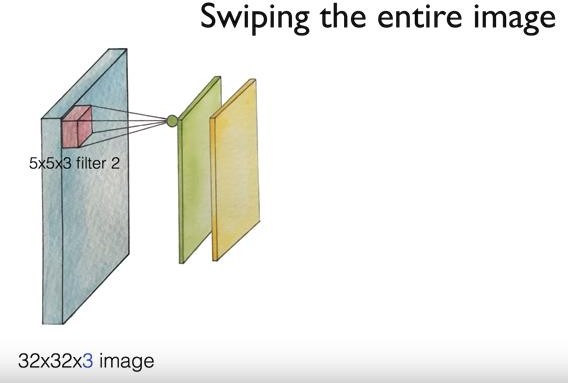
36

**CNN Process**



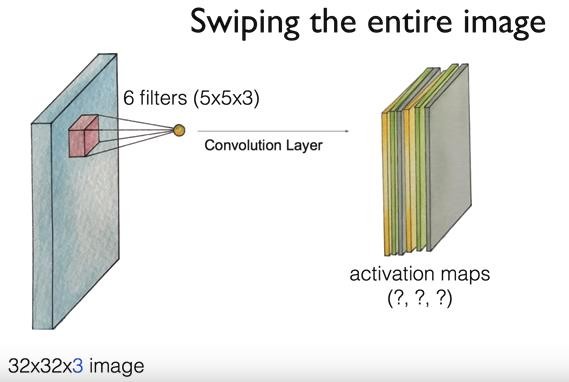
37

**CNN Process**



38

**CNN Process**

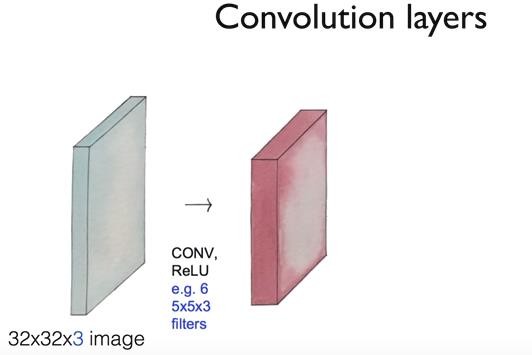


28 28 6

If we do not zero padding

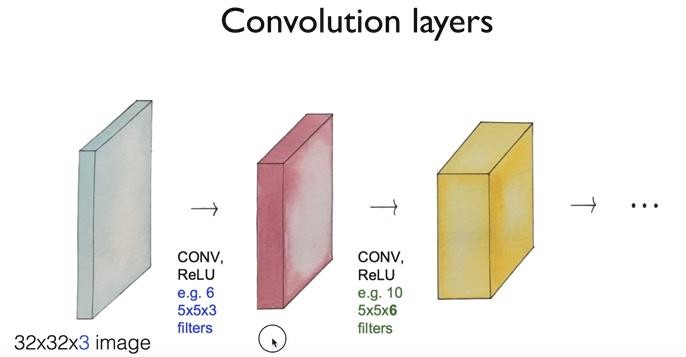
39

**CNN Process**



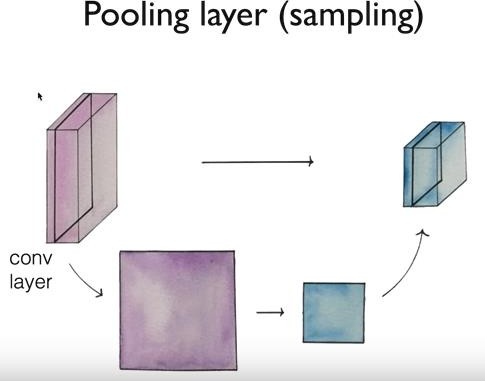
40

**CNN Process**



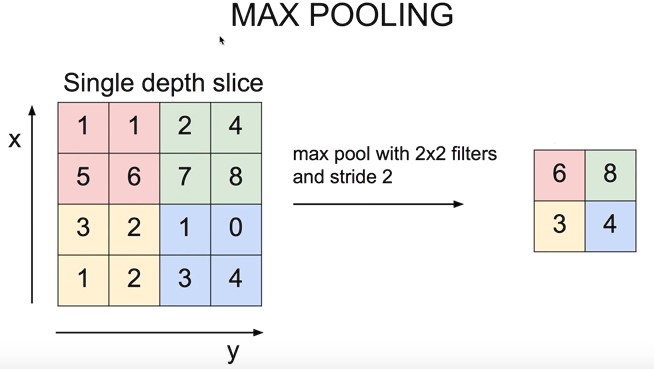
41

**CNN Process**



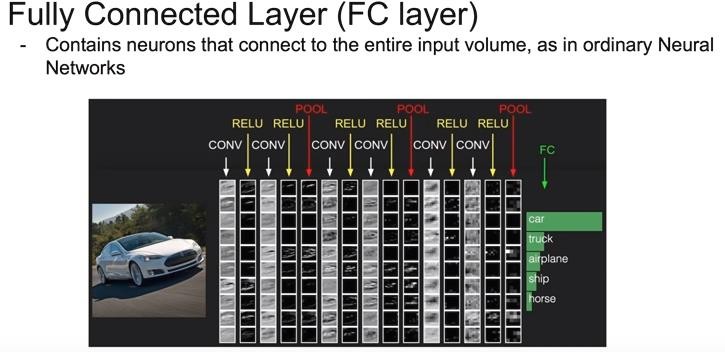
42

**CNN Process**



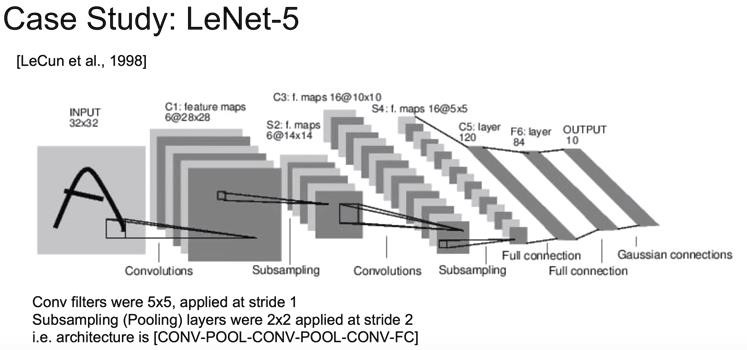
43

**CNN Process**



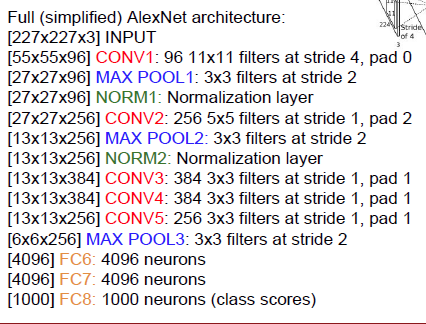
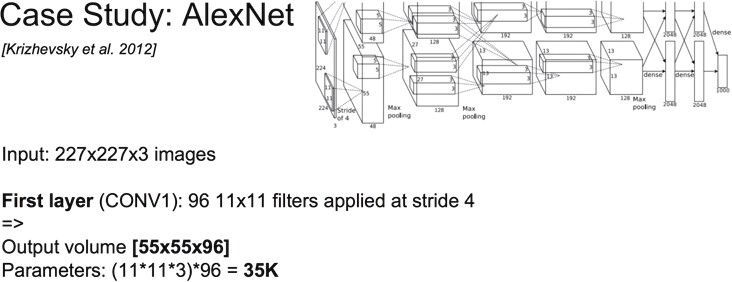
44

**CNN Example**

45

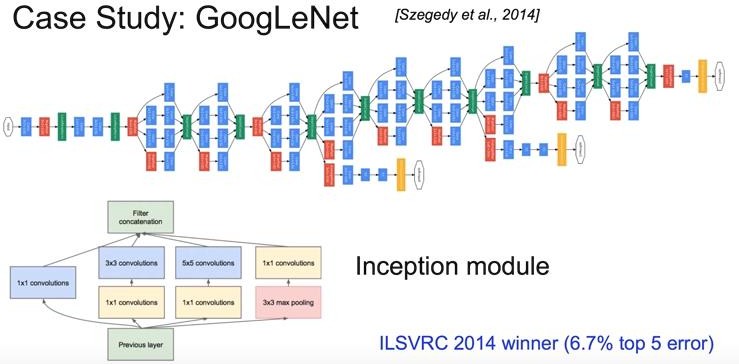


**CNN Example**



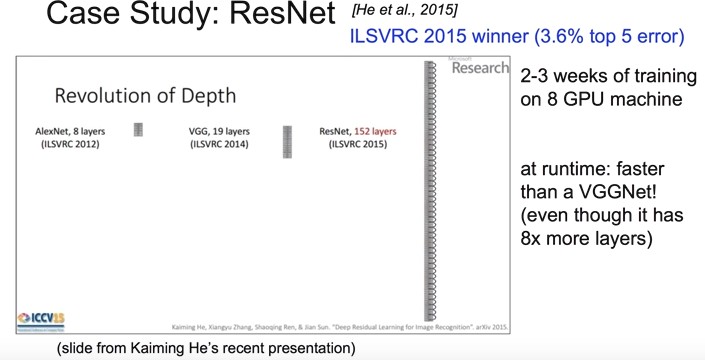
46

**CNN Example**



47

**CNN Example**



48

# Practice

1. **Solve XOR Problem using NN**
2. **Recognize MNIST using NN**
3. **Recognize MNIST using CNN**

49

**Tensorflow**

* We are going to use Tensorflow through Docker
* Open the cmd terminal in windows (administrator)

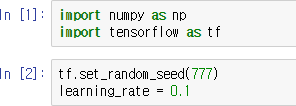
> docker run -it -p 8888:8888 tensorflow/tensorflow:1.4.1-py3

* Then, connect to the web browser
* We are going to use Python to use Tensorflow

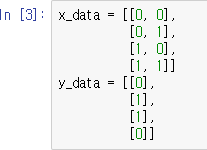
50

**Practice #1 - Solve XOR Problem using NN**

* Import packages and set learning rate



* Input data



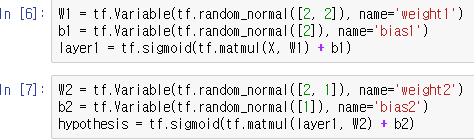
51

**Practice #1 - Solve XOR Problem using NN**

* Make placeholder for Tensorflow, which is some variable space



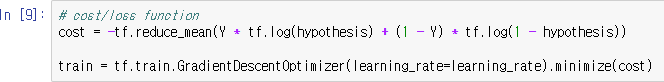
* Define NN



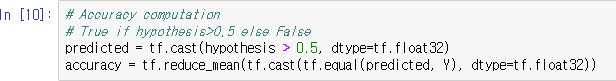
52

**Practice #1 - Solve XOR Problem using NN**

* Define cost function and optimizer



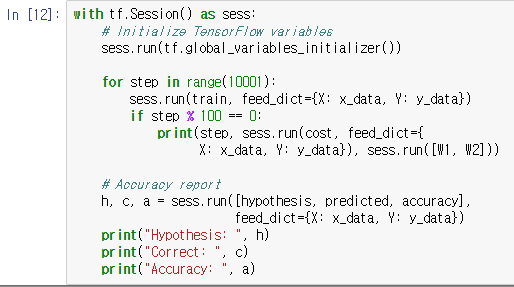
* Define accuracy computation



53

**Practice #1 - Solve XOR Problem using NN**

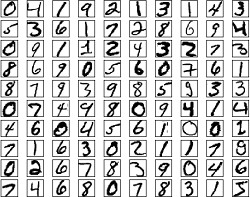
* Start learning and see the result



54

**Practice #2 - Recognize MNIST using NN**

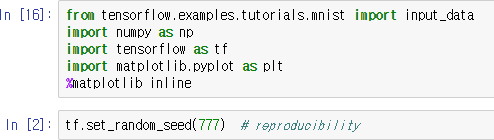
* MNIST
  + Handwritten digit database
  + We want to recognize this handwritten digit



55

**Practice #2 - Recognize MNIST using NN**

* Import packages and set random



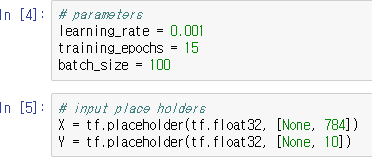
* Load the data



56

**Practice #2 - Recognize MNIST using NN**

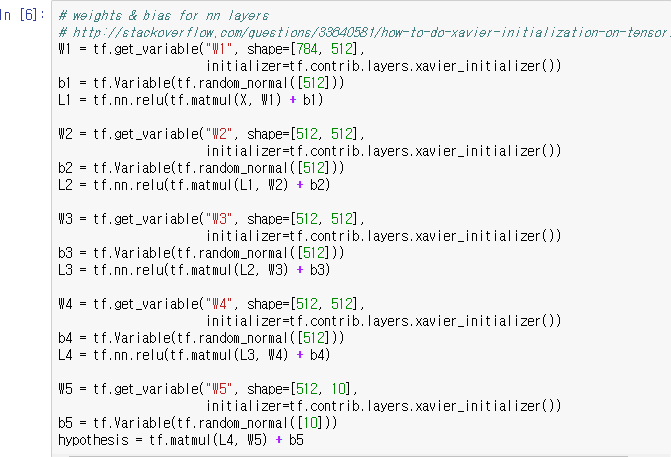
* Set hyperparameters and make placeholder



57

**Practice #2 - Recognize MNIST using NN**

* Define NN (5 layers)



58

**Practice #2 - Recognize MNIST using NN**

* Define cost function and optimizer



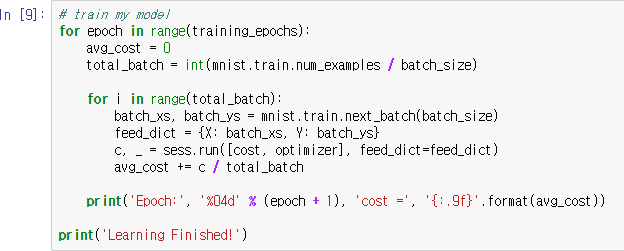
* Start Tensorflow session and initialize variables



59

**Practice #2 - Recognize MNIST using NN**

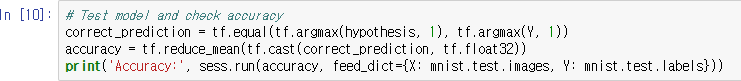
* Start learning
  + You can see the cost of each epoch of learning



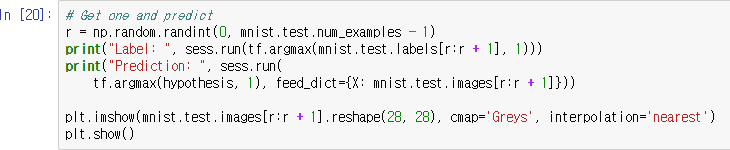
60

**Practice #2 - Recognize MNIST using NN**

* Check the trained model and accuracy



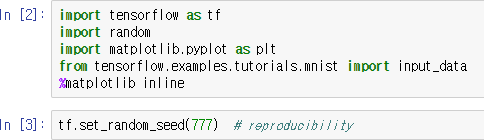
* Get one MNIST data and predict



61

**Practice #3 - Recognize MNIST using CNN**

* Import packages and set random seed



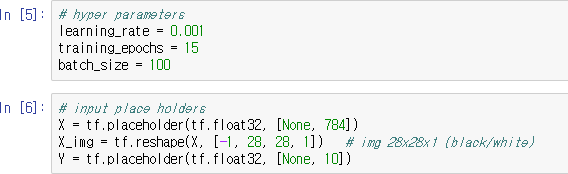
* Load the data



62

**Practice #3 - Recognize MNIST using CNN**

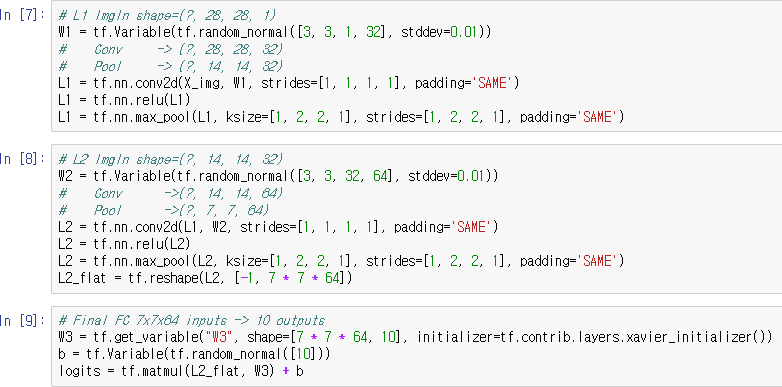
* Define hyperparameters and placeholder for Tensorflow



63

**Practice #3 - Recognize MNIST using CNN**

* Define CNN
  + Conv → ReLU → Pool → Conv → ReLU → Pool → FC



64

**Practice #3 - Recognize MNIST using CNN**

* Define cost function and optimizer



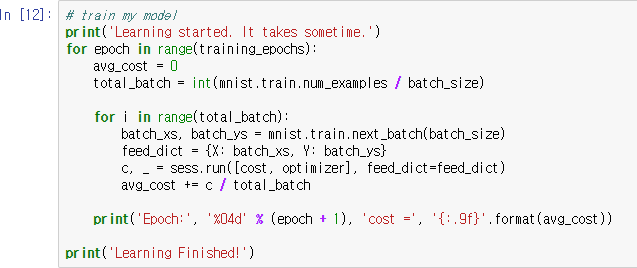
* Start Tensorflow session and initialize variables



65

**Practice #3 - Recognize MNIST using CNN**

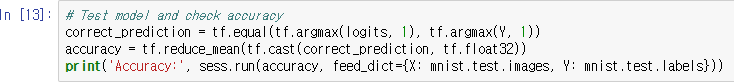
* Start learning
  + You can see cost of each epoch of training



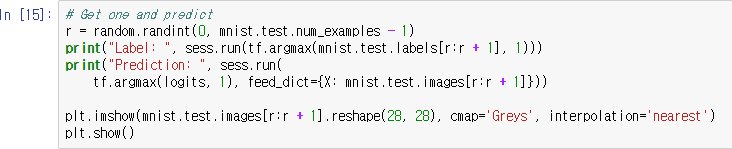
66

**Practice #3 - Recognize MNIST using CNN**

* Test model and check accuracy



* Get one MNIST data and predict



67