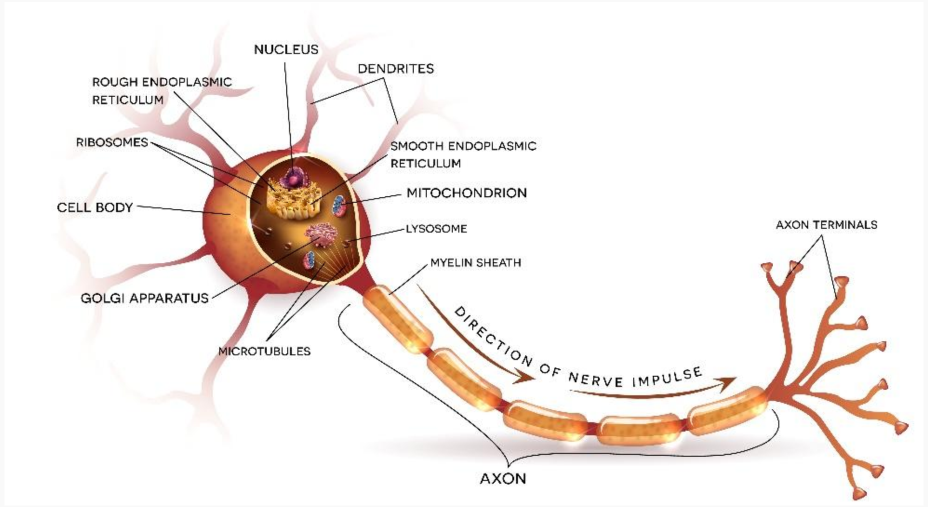
SIC Batch 5

Week 7 - Deep Learning for Computer Vision

Introduction to Deep Learning

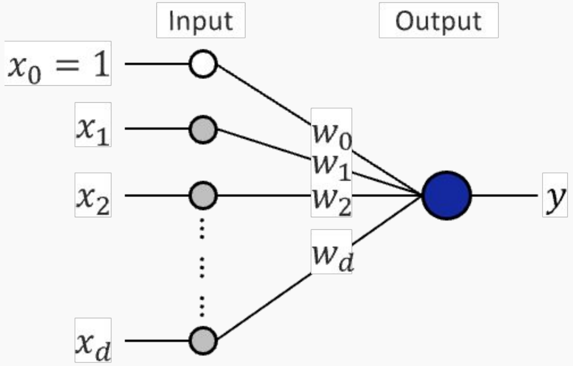
Biological Origin

- They receive and store various information by exchanging chemical signals with adjacent neurons through a structure named synapse.
- The number of neurons in the brain of a human being is about 1011 neurons. And the average neuron has around 1,000 synapses. Thus, 100 trillion (1014) synapses are interconnected in the human brain.
- In the 1940s, several researchers who studied biological neural networks began to investigate how to emulate the mechanisms of the neural network. Perceptron is one of the early artificial neural network models.
- From here on, the term ‘neural network’ refers to an ‘artificial’ neural network.

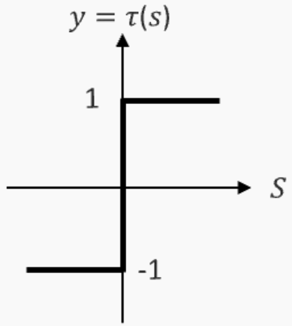


Perceptron

- It has an input layer and an output layer.
- The input layer does not operate, so the perceptron is considered a single-layer structure.
- The i th node of the input layer takes x_i from the feature vector x=(x_1,x_2,⋯,x_d)^T .
- The bias node always takes 1 as input.
- The output layer has a single node.
- The connection of the i th node of the input layer and the output layer has weight w_i.



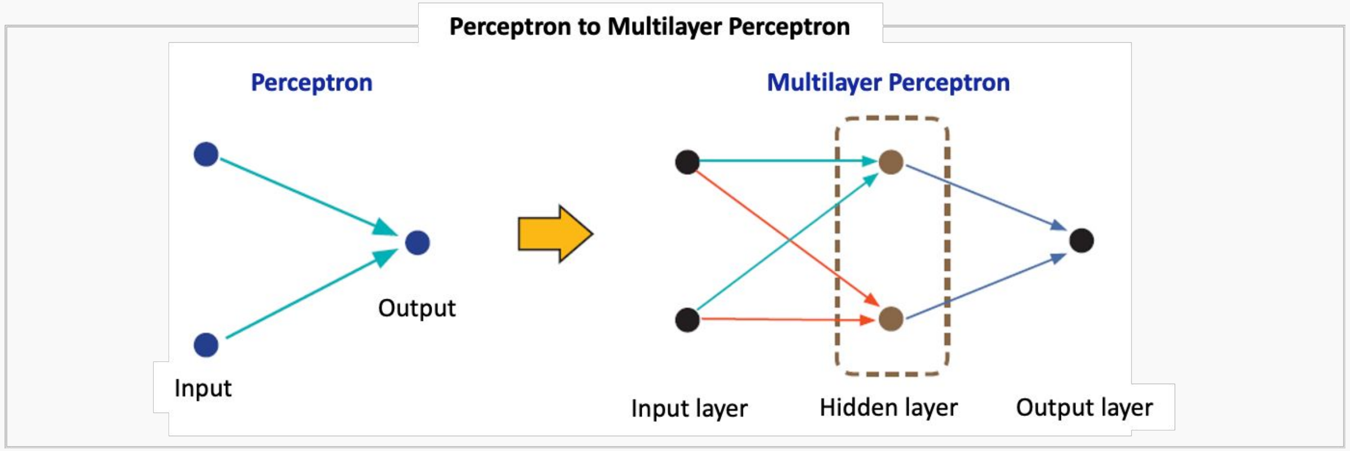
(a) Structure of a perceptron



(b) Use threshold function as an activation function $\tau(s)$

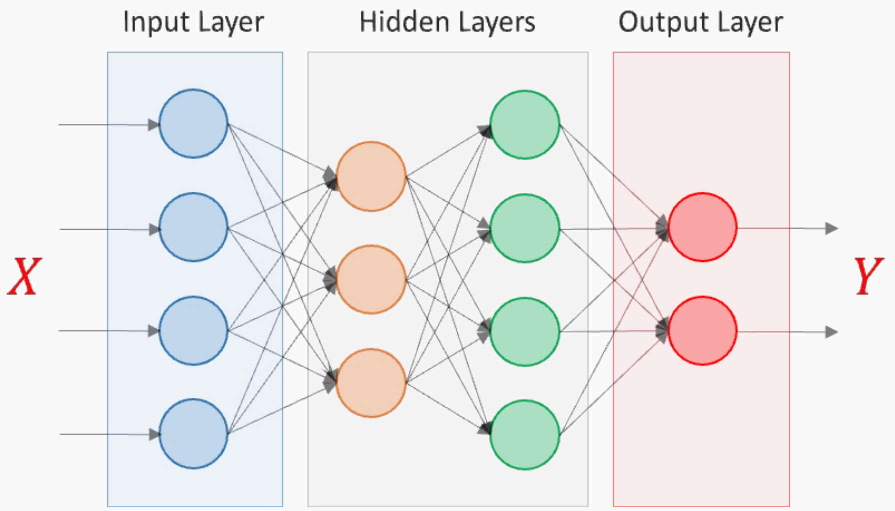
Mechanism of a perceptron

Perceptron -> Multilayer Perceptron



- Multilayer perceptron easily work with non-linear problems.
- It can handle complex problems while dealing with large datasets.
- Makes quick predictions after training.

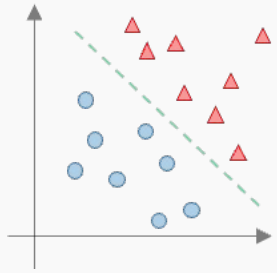
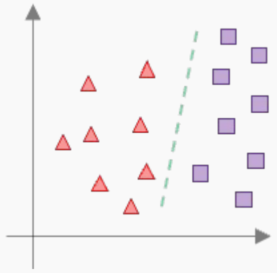
Artificial Neural Network (ANN)



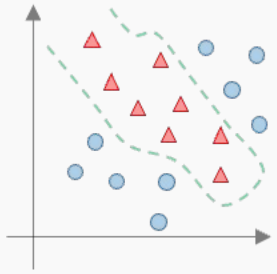
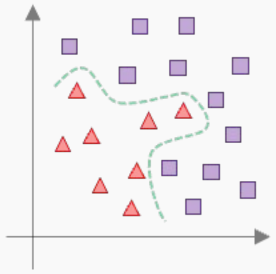
- Mimics the neural connections of a biological brain.
- There can be several hidden layers.

Why Artificial Neural Network (ANN)?

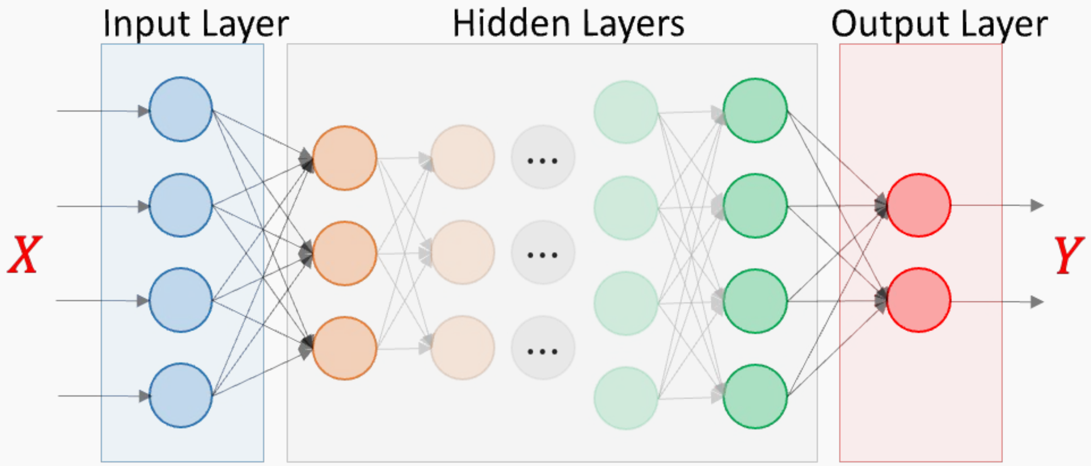
For a logistic regression (even for the multi-class variant), the decision boundaries are linear.

How about the cases that require non-linear decision boundaries? ANN with hidden layers!

What is Deep Neural Network

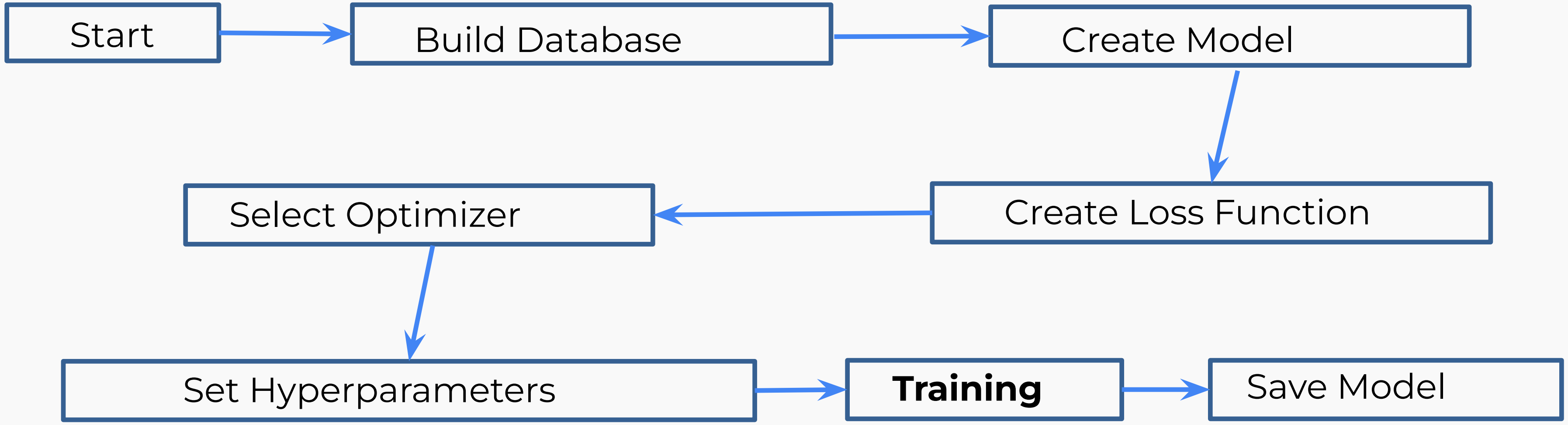


- Deep Neural Network often have a complex hidden layer structure with a wide variety of different layers, such as a convolutional layer, max-pooling layer, dense layer, and other unique layers.
- A deep neural network has more layers (more depth) than ANN and each layer adds complexity to the model while enabling the model to process the inputs concisely for outputting the ideal solution.

Model Training

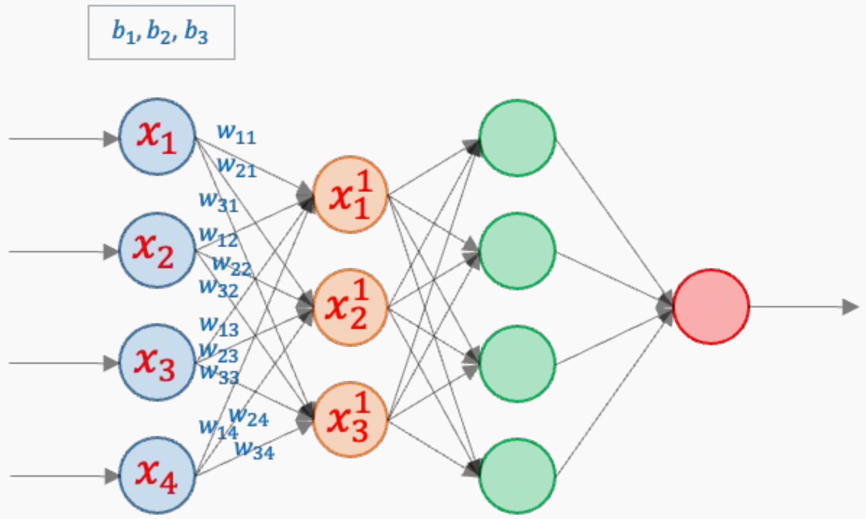
Training Models

Main Processing in Modeling :



core purpose : find the best W matrix

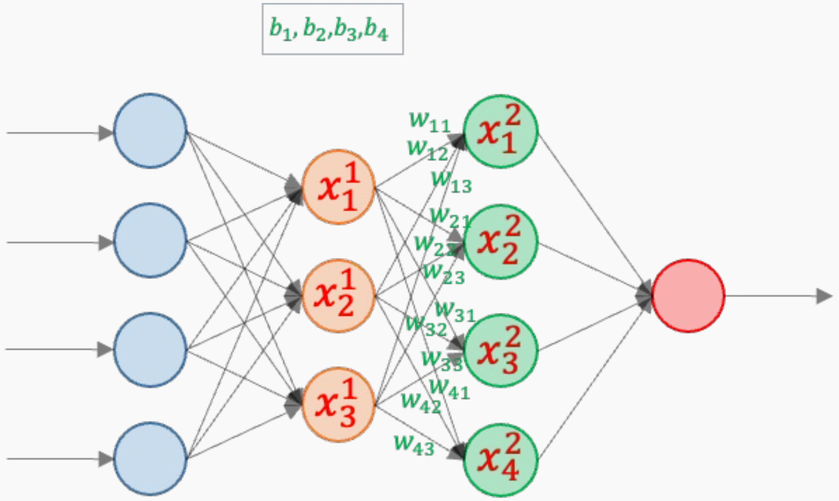
ANN Training : Forward Propagation



$$\begin{bmatrix} x_1^1 \\ x_2^1 \\ x_3^1 \end{bmatrix} = Activation\left(\begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} \\ w_{21} & w_{22} & w_{23} & w_{24} \\ w_{31} & w_{32} & w_{33} & w_{34} \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}\right)$$

When the values of X_i are given, apply the weights and propagate the signal forward to the next layer.

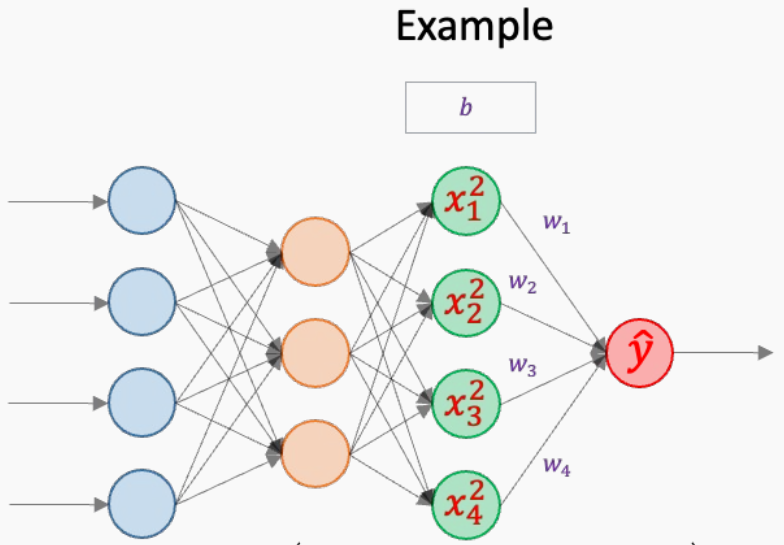
ANN Training : Forward Propagation



$$\begin{bmatrix} x_1^2 \\ x_2^2 \\ x_3^2 \\ x_4^2 \end{bmatrix} = Activation\left(\begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \\ w_{42} & w_{42} & w_{43} \end{bmatrix} \cdot \begin{bmatrix} x_1^1 \\ x_2^1 \\ x_3^1 \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \\ b_3 \\ b_4 \end{bmatrix} \right)$$

Propagate forward to the next layer.

ANN Training : Forward Propagation



$$\cong y?$$

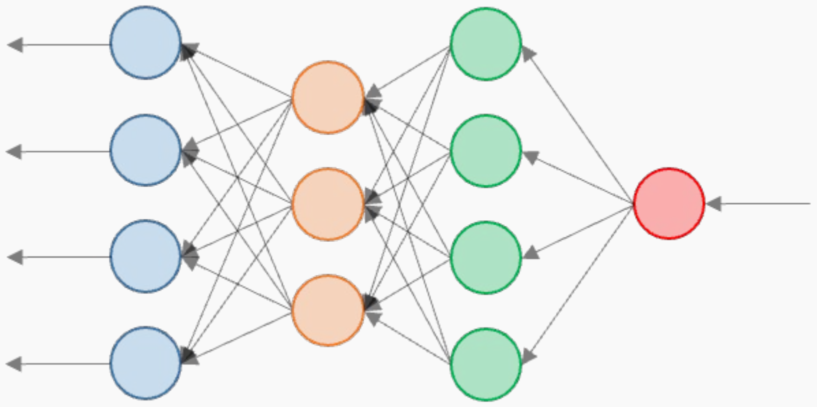
$$Error = y - \hat{y}$$

$$\hat{y} = Activation\left(\begin{bmatrix} w_1 & w_2 & w_3 & w_4 \end{bmatrix} \cdot \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_3^2 \\ x_4^2 \end{bmatrix} + b\right)$$

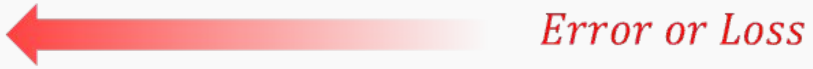
Propagate all the way; the difference between the estimated value $\hat{y}$ and the true value y is the error.

ANN Training : Backward Propagation

Example



$$W \leftarrow W - \eta \nabla Loss$$
$$b \leftarrow b - \eta \nabla Loss$$

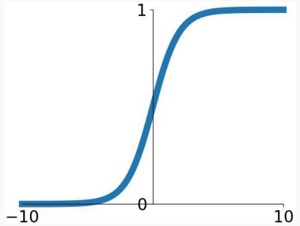


Propagate the error backward and update the parameters by gradient descent algorithm. Repeat from 1)

Activation Functions

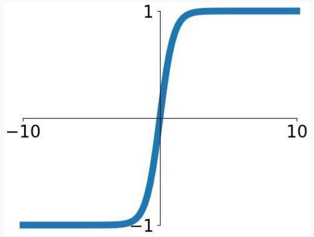
Sigmoid

$\sigma(x) = \frac{1}{1+e^{-x}}$



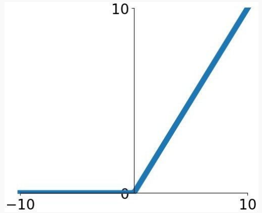
tanh

$\tanh(x)$



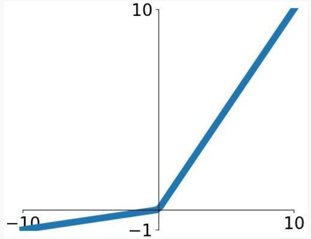
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

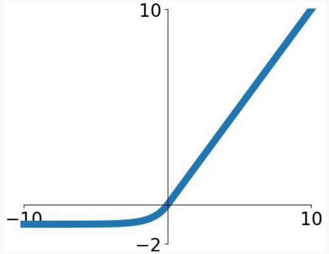


Maxout

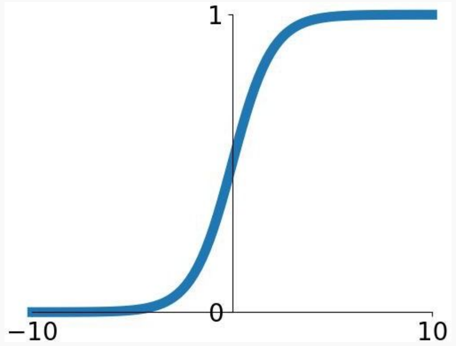
$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Activation Functions



Sigmoid

Fig4.3 Sigmoid function

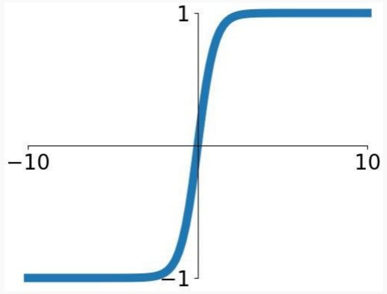
$$\sigma(x) = \frac{1}{1+e^{-x}}$$

Compress numbers to range [0,1]

Problems

1. Saturated neurons “kill” the gradients
2. exp( ) is a bit compute expensive
3. Sigmoid outputs are not zero-centered

Activation Functions



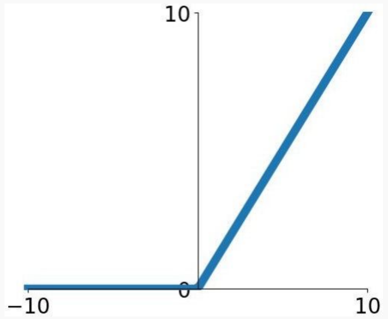
tanh(x)

Fig4.4 tanh function

Squashes numbers to range [-1,1]

- zero centered (nice)
- still kills gradients when saturated

Activation Functions



ReLU

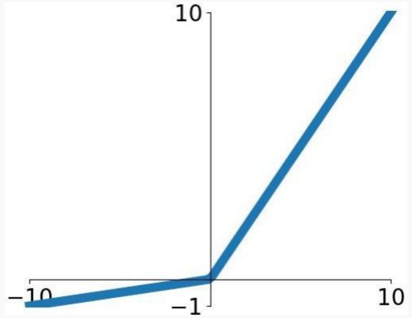
Fig4.5 ReLU function

Computes **f(x) = max(0,x)**

- Does not saturate (**in +region**)
- Very computationally efficient
- Converges much faster than sigmoid/tanh in practice

- what is the gradient when x < 0
- Not zero-centered output

Activation Functions



Leaky ReLU

Fig4.6 ReLU function

$$f(x) = \max(0.01x, x)$$

- Does not saturate
- Computationally efficient
- Converges much faster than sigmoid/tanh in practice!
- **will not "die"**.

Summary Activation Functions

- This summarizes some of the most commonly used activation functions.
- Softmax is often used at the output layer of neural networks that do classification.

Name	Formula	TensorFlow
Step	$\mathrm{Step}(x)=\theta(x-x_0)$	(tf.math.sign(x-x_0)+1)/2
Sigmoid	$\sigma(x)=\frac{1}{1+e^{-x}}$	tf.math.sigmoid(x)
Tanh	$\tanh(x)=\frac{e^x-e^{-x}}{e^x+e^{-x}}$	tf.math.tanh(x)
ReLu	$\mathrm{ReLu}(x)=\max(0,x)$	tf.nn.relu(x)
Softmax	$\sigma(x)_j=\frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$	tf.nn.softmax(x)

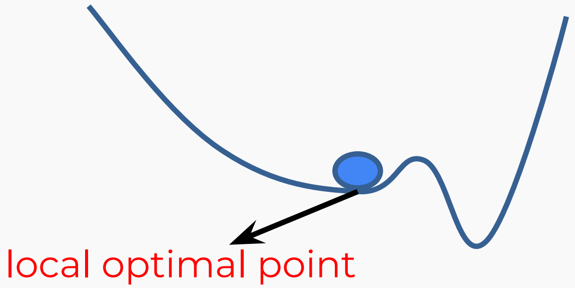
Overfitting

How to solve **Overfitting ?**

1. Early Stopping - local optimal

2. L1 and L2 regularization weights

Early stopping



L1 regularization $$J(\theta) = [y_\theta(x) - y]^2 + \lambda[|\theta_1| + |\theta_2| + \ldots + |\theta_n|]$$

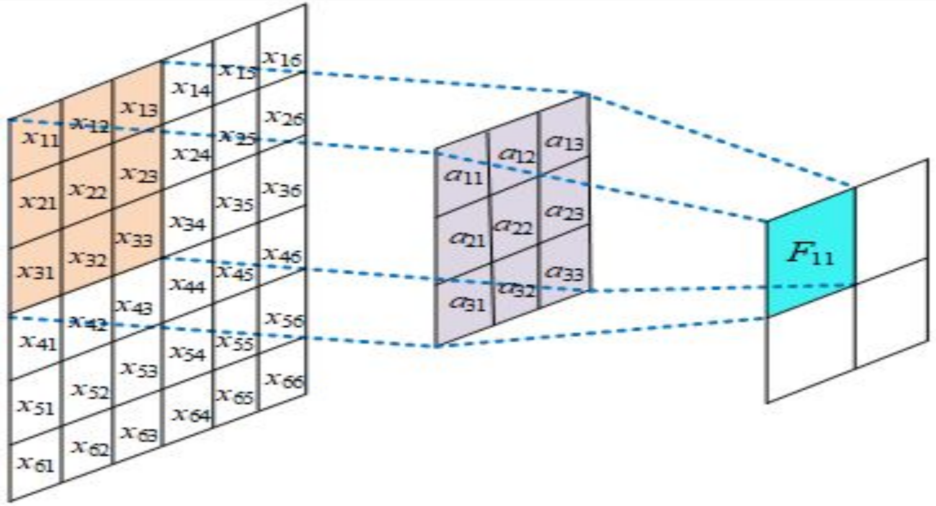
L2 regularization $$J(\theta) = [y_\theta(x) - y]^2 + \lambda[\theta_1^2 + \theta_2^2 + \ldots + \theta_n^2]$$

Down the influence of **high power function**

Overfitting

How to solve **Overfitting ?**

3. Weight Sharing



x_{11} x_{12} x_{13} x_{14} x_{15} x_{16}
x_{21} x_{22} x_{23} x_{24} x_{25} x_{26}
x_{31} x_{32} x_{33} x_{34} x_{35} x_{36}
x_{41} x_{42} x_{43} x_{44} x_{45} x_{46}
x_{51} x_{52} x_{53} x_{54} x_{55} x_{56}
x_{61} x_{62} x_{63} x_{64} x_{65} x_{66}
a_{11} a_{12} a_{13}
a_{21} a_{22} a_{23}
a_{31} a_{32} a_{33}
F_{11}

Fig4.7 expression for weight sharing in convolutional process

Parameters : 9

Overfitting

How to solve **Overfitting ?**

4. Dropout

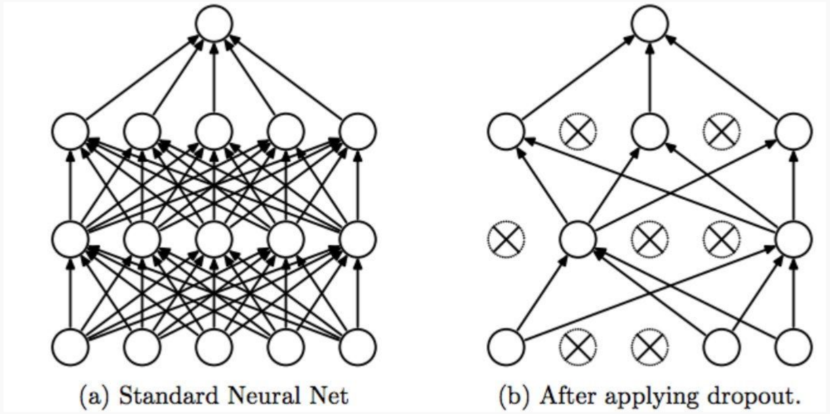


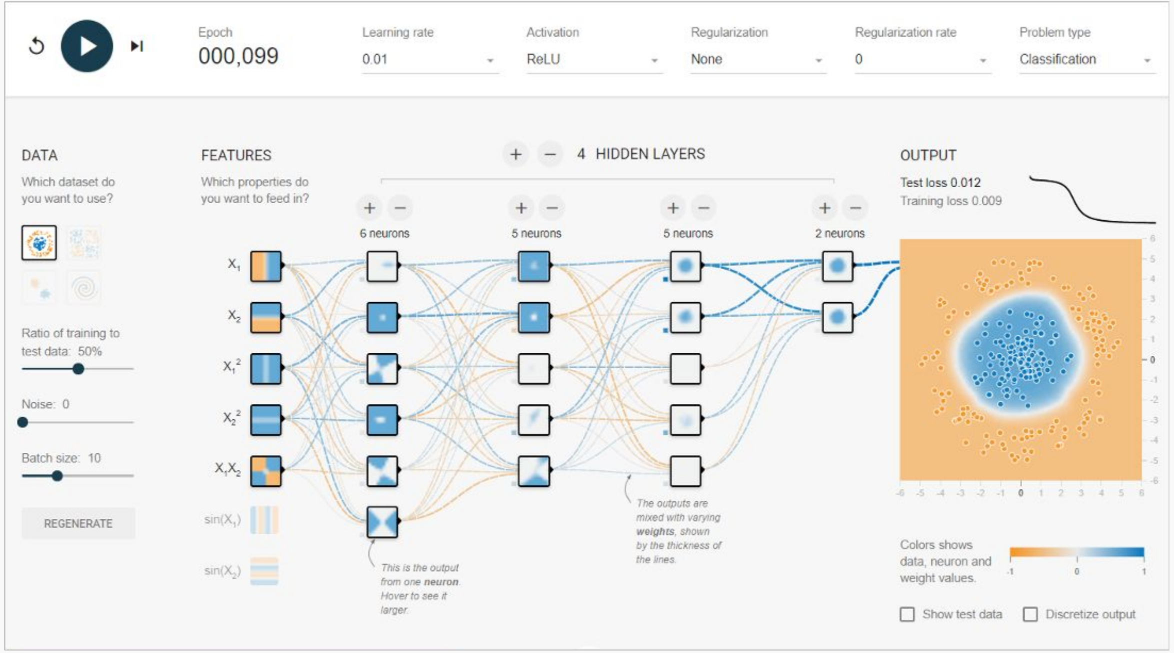
Fig4.8 expression for dropout in neural network

Every node is reduced by **probability**

Probability is usually set to **0.5**

Every epoch only the **weights** of non-reducing nodes can be updated

Deep Learning Playground


Epoch 000,099

Learning rate 0.01

Activation ReLU

Regularization None

Regularization rate 0

Problem type Classification

DATA

Which dataset do you want to use?

Ratio of training to test data: 50%

Noise: 0

Batch size: 10

FEATURES

Which properties do you want to feed in?

4 HIDDEN LAYERS

6 neurons

5 neurons

5 neurons

2 neurons

This is the output from one ***neuron***. Hover to see it larger.

The outputs are mixed with varying ***weights***, shown by the thickness of the lines.

OUTPUT

Test loss 0.012

Training loss 0.009

Colors shows data, neuron and weight values.

Website: https://playground.tensorflow.org

- The GUI of the website (https://playground.tensorflow.org) is intuitive.
- Select data from the left side panel of the website.
- Adjust the features, layers, and nodes.
- From the top dropdown menus, customize the learning rate, activation function, regularization, etc.
- Press the Play button to start training and the training loss will decreases as the training progresses.
- The thickness of an edge line represents the weight that can also be changed manually.

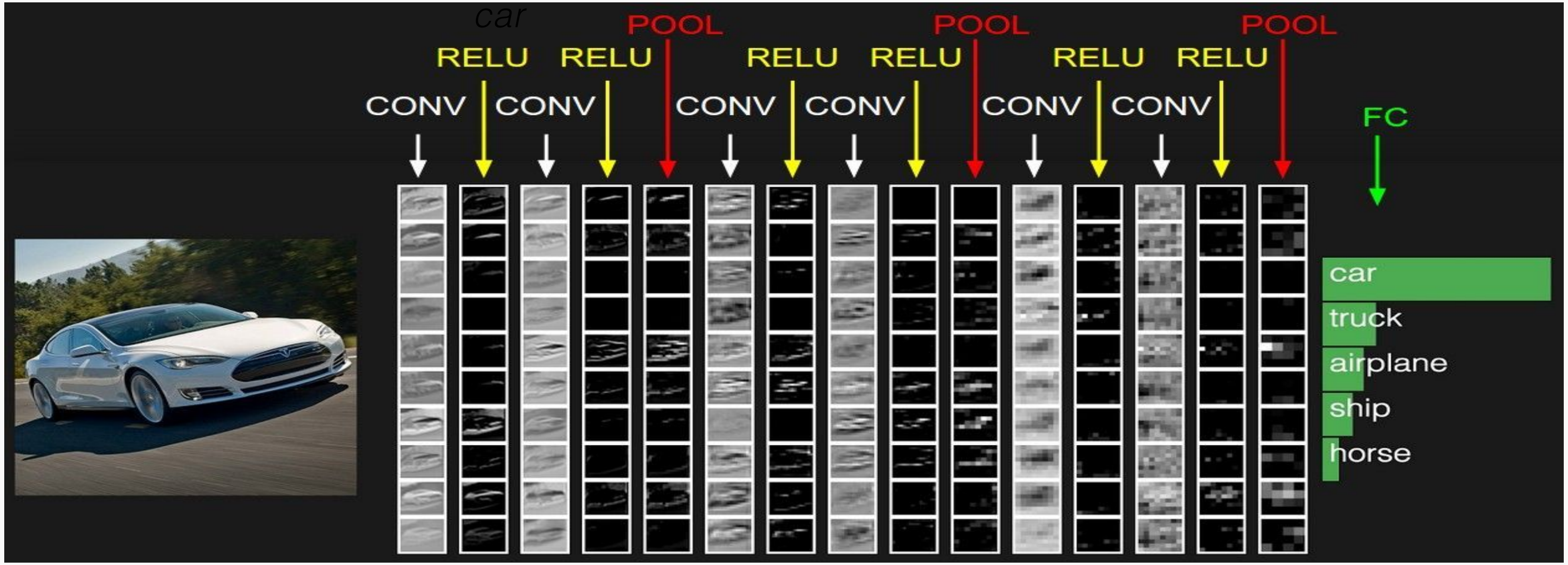
Q&A

Convolution Neural Network

Convolution Neural Network

CNN adalah DL yang terdiri dari :
Convolutional Layer, Activation Layer, Pooling Layer and so on

Fig3.1 process of CNN to classify a car



Convolution Neural Network

- Convolutional Layer: Menggunakan filter (kernel) untuk mengekstrak feature gambar pada gambar. Dengan mengalikan image dengan filter maka Convolution Layer bisa mengekstrak image feature.

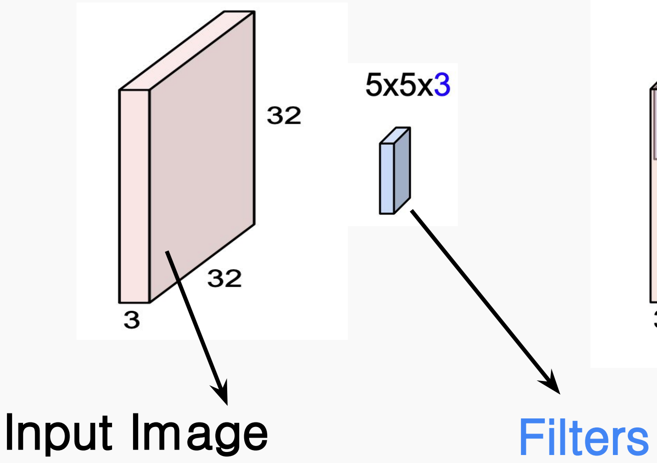
- Activation Layer: Menggunakan layer non-linear untuk mengekstrak feature pada image. Contoh activation layer : ReLU, sigmoid, and tanh.

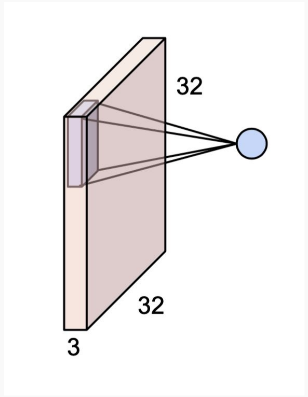
- Pooling Layer: Mengurangi ukuran feature pada image untuk mempercepat proses training dan mengekstrak feature paling penting pada gambar.

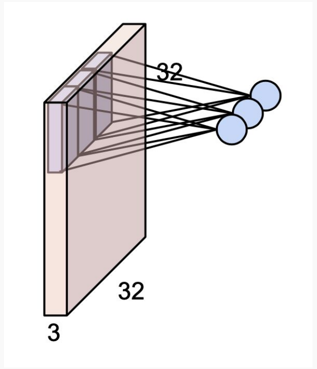
Convolution Neural Network

Convolutional Layer : Ekstrak feature pada gambar menggunakan kernel.

Slide over the image spatially, computing dot products



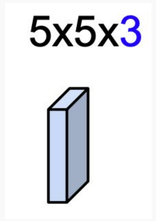


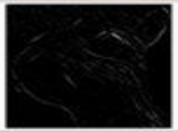
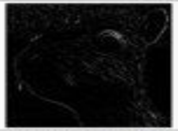
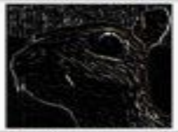
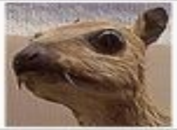
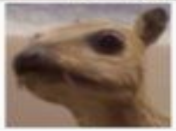


Convolution Neural Network

Convolution filters

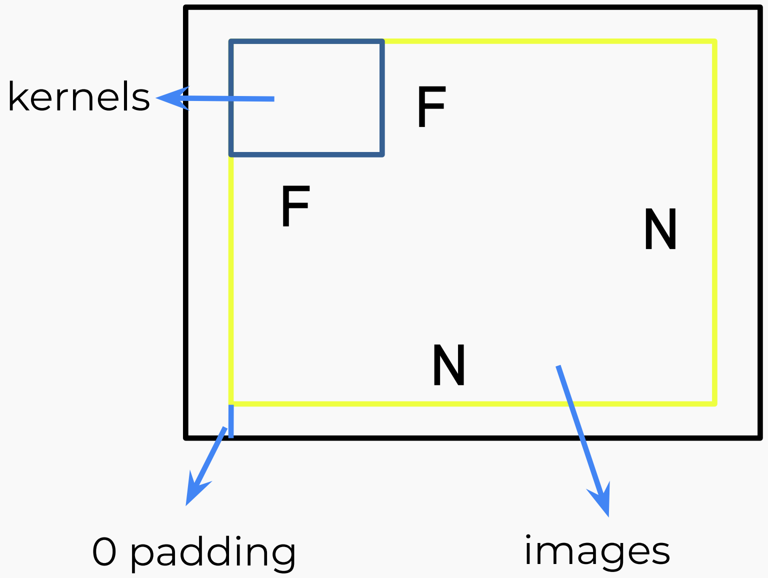
Different filters have different convolution performance



Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	

Convolution Neural Network

Basic Parameters



Output size:
(N+pad×2 - F) / stride + 1

e.g. N = 7, F = 3, pad=1,stride=1:
output size = (7+2×1-3)/1+1=7×7

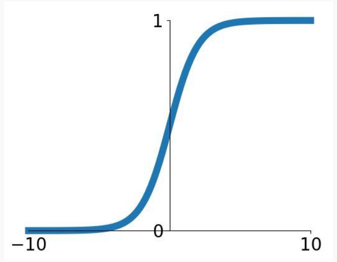
parameters:
F×F×channels×filters numbers

Convolution Neural Network

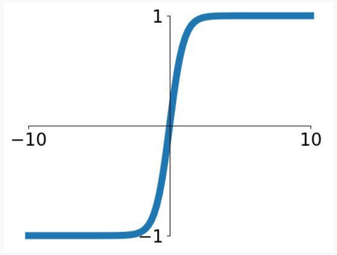
Activation Layer: add nonlinear learning ability to CNN

preview the **Linear Classifier**

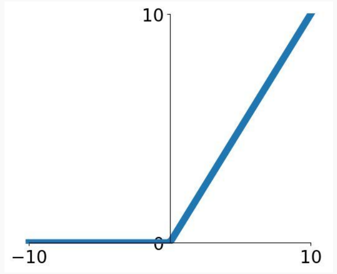
$$\textbf{W} \times \textbf{input} + \textbf{b} \xrightarrow{\text{activation function}} \textbf{nonlinear output}$$



Sigmoid



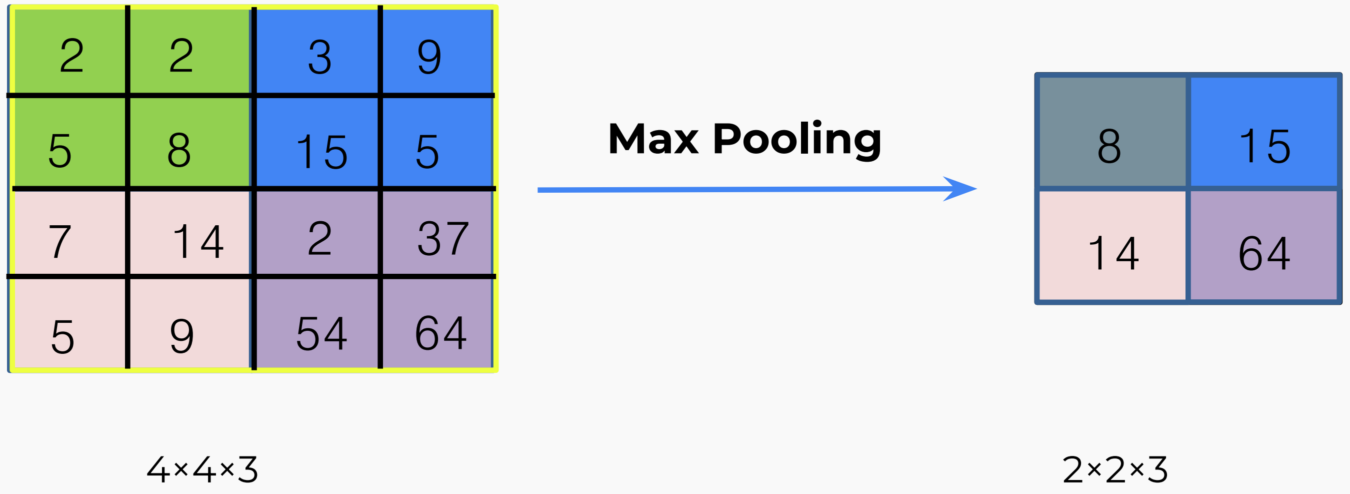
tanh



ReLU

Convolution Neural Network

Pooling Layer: data dimension reduction



without changing the **depth** but just **downsampling**

Convolution Neural Network

Summary:

- **CNN** mainly includes three parts : convolutional layer,activation layer and pooling layer
- **Convolutional Layer** can extract the features by different kernels
- **Activation Layer** adds the nonlinear learning ability to model which is better than the linear classifier
- **Pooling** is a downsampling method can stand out the features and reduce meaningless pixel

Q&A

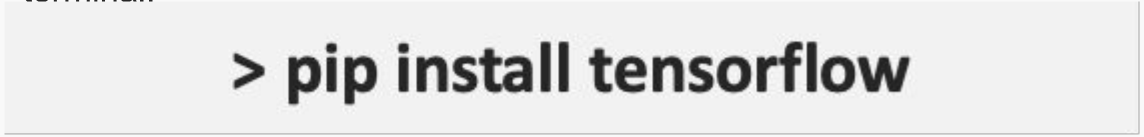
Introduction to TensorFlow

- An open source machine learning/deep learning library developed by the Google Brain team.
- Computations are expressed as graphs.
- TensorFlow can utilize both CPU and GPU to improve the training performance of a machine learning model.
- Supports Windows, mac OS, Linux, etc.
- Supports Python, Java, R, etc.
- Runs on multiple CPUs and GPUs.

https://www.tensorflow.org/

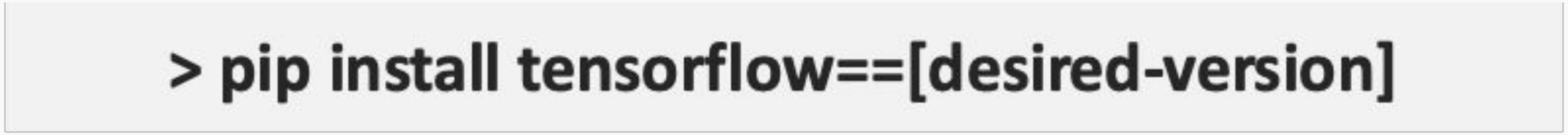
Installing Tensorflow

- Use Python’s pip installer to install TensorFlow from PyPI.
- Run the following command to install most recent stable version of tensorflow on the terminal.



```
> pip install tensorflow
```

- To install specific version of tensorflow



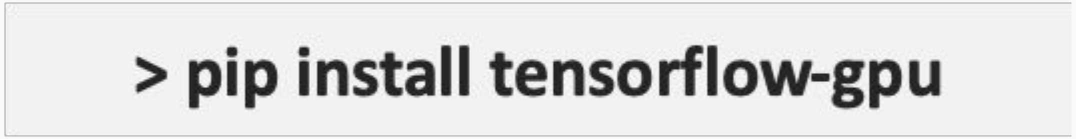
```
> pip install tensorflow==[desired-version]
```

Installing Tensorflow

- For GPU, you need an NVIDIA-compatible graphic card and an installation of CUDA Toolkit and cuDNN library

(GPU is recommended for neural network training).

- If the above conditions are fulfilled, you can install the TensorFlow GPU version.



```
> pip install tensorflow-gpu
```

Hands On Lab

https://drive.google.com/file/d/1IoeXfccZnbFMY2hyK4fJdSqqpSZKoNi7/view?usp=sharing