SENG
SCHOOL OF ENGINEERING

ENGINEERING
OUR FUTURE

Deep Learning Workshop:

Concepts & Experiments

MSBD 5001 Fall 2019

Outline

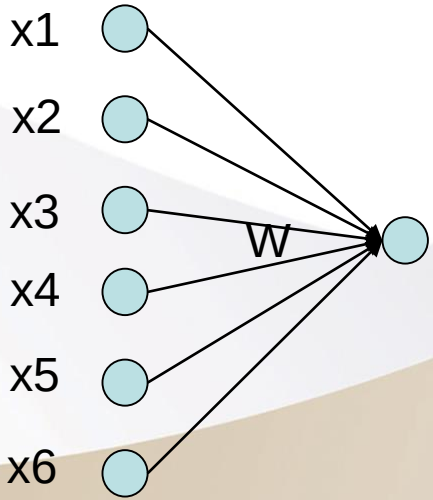
- The basics of fully connected neural network
- The basics of convolutional neural network
- The basics of deep learning
- The tutorial of Tensorflow 2.0.
- Use Tensorflow 2.0 beta to train, validate and test your own deep learning model

Install Tensorflow 2.0

- pip install -U gast==0.2.2
- pip install tensorflow==2.0.0-beta0
- pip install pillow
- Open python and run the following codes:
  - from tensorflow import keras
  - keras.datasets.mnist.load_data()
  - keras.applications.VGG16(input_shape=(224,224,3), weights='imagenet',include_top=False)

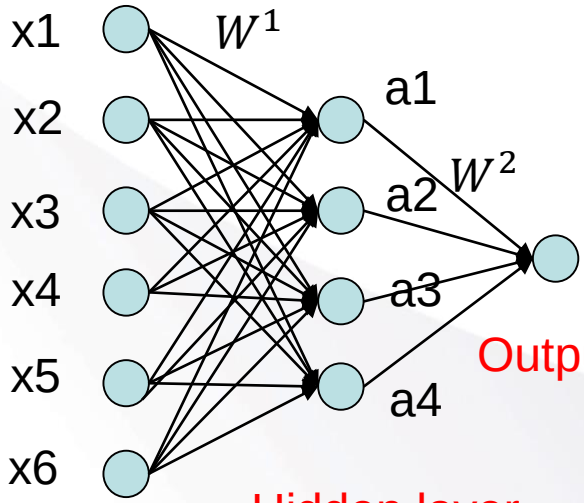
Logistic Regression

- $L = \sum(Error(y,\ \sigma(x^T w))$ #make a swap of x/w.
- Apply a non-linear function (E.g. sigmoid) on $(x^T w)$ to fit the target value.
- We can use graph to represent the process.



$$output = \sigma(x_1 * w_1 + x_2 * w_2 + x_3 * w_3 + x_4 * w_4 + x_5 * w_5 + x_6 * w_6)$$

A Simple Fully Connected Feedforward Neural Network



W^1 : 6*4 matrix.

W^2 : 4*1 matrix. (So transposition is not needed)

W_{ij}^1: The weight connecting x_i and a_j.

$$a = \sigma(xW^1)$$

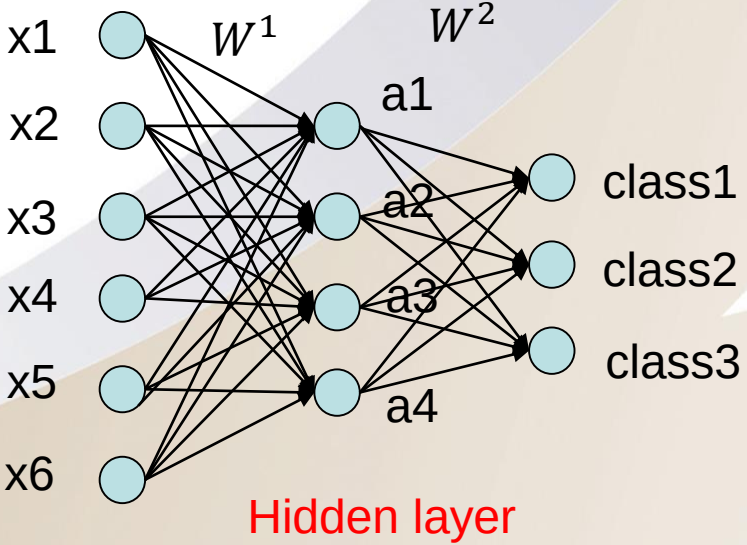
$$output = \mathrm{a}W^2 = \sigma(xW^1)W^2$$

More complex neural networks could be extended based on this with more hidden layers.

Softmax: Classfication

- N output nodes to represent the probability distribution over n different classes.
- W^2 : 4*3 matrix.
- $output = softmax(\mathrm{aW}^2) = softmax(\sigma(xW^1)\mathrm{W}^2)$
- Like logistic regression, use a non-linear function to transform output from $(-\infty, +\infty)$ to [0,1].
- Softmax is similar with sigmoid, supporting multiple outputs.

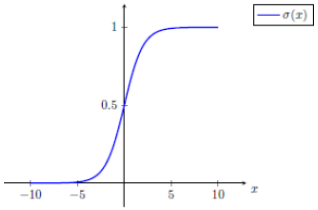
$$\mathrm{softmax}(\boldsymbol{z})_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$



Activation

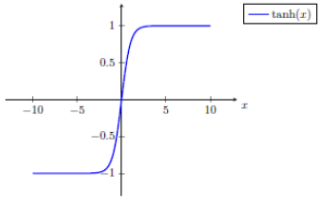
- Activation function is to transform the input signal into an output signal to model complex non-linear patterns.
- Without activation function, stacking linear layers is equivalent to a single linear layer.
- We can replace the sigmoid function in hidden layer with other non-linear functions!

Sigmoid



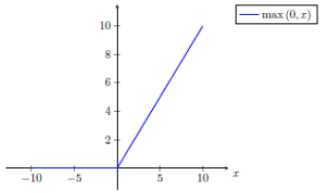
$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

tanh



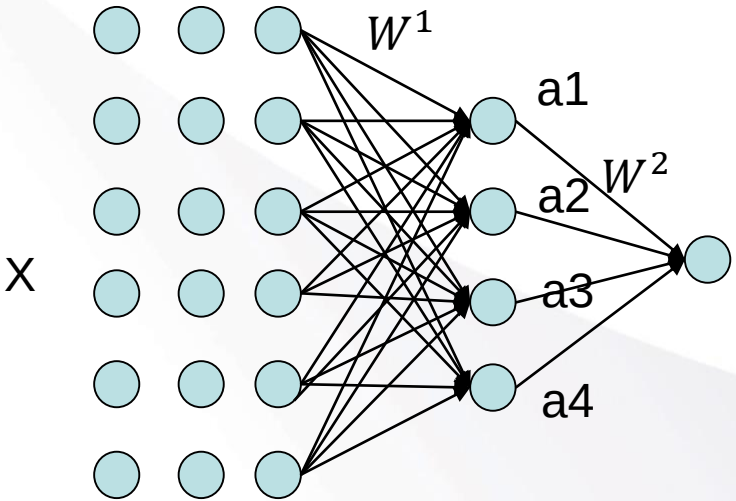
$$\tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$$

ReLu



$$\text{ReLu}(x) = \max(0, x)$$

Batch Forwarding



THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Batch size: 3

W^1 : 6*4 matrix.

W^2 : 4*1 matrix.

W_{ij}^1: The weight connecting x_i and a_j.

X: 3*6 feature matrix (generally a row represents a sample)

X_{ij}: The j-th feature of the i-th sample.

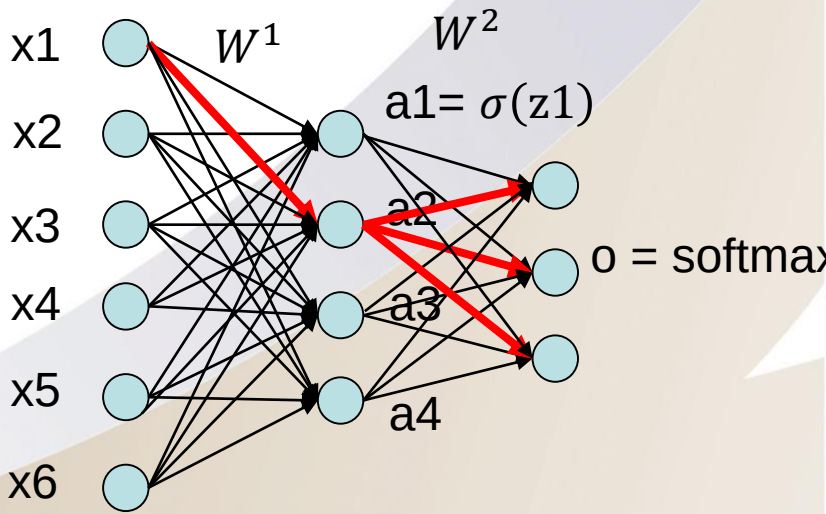
$$A = \sigma(XW^1)$$

$$output = A\mathrm{W}^2 = \sigma(XW^1)\mathrm{W}^2$$

Output: 3*1 matrix.

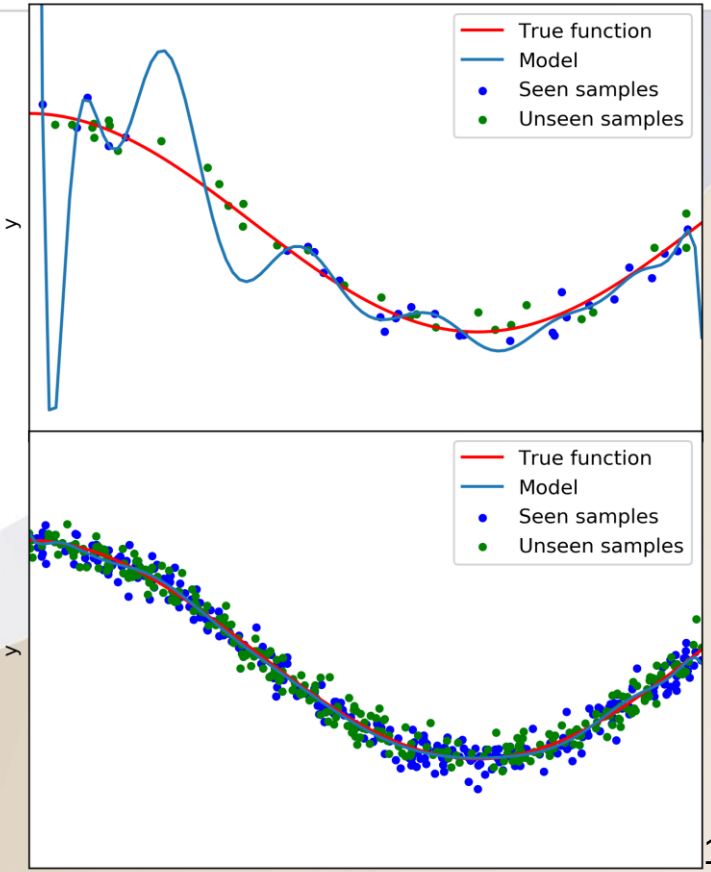
Back-propagation

- Chain Rule of Calculus:
- $z = f(g_1(x) + g_2(x))$
- $\frac{\partial z}{\partial x} = \frac{\partial z}{\partial g_1}\frac{\partial g_1}{\partial x} + \frac{\partial z}{\partial g_2}\frac{\partial g_2}{\partial x}$
- $loss = L(softmax(aW^2)$
- $= L(softmax(\sigma(xW^1)W^2))$
- For W_{12}^1, $\frac{\partial L}{\partial W_{12}^1} = \Sigma_k \frac{\partial L}{\partial o_k}(\frac{\partial o_k}{\partial s_k}\frac{\partial s_k}{\partial a_2}\frac{\partial a_2}{\partial z_2}\frac{\partial z_2}{\partial W_{12}^1})$
- Accelerated by matrix multiplication.



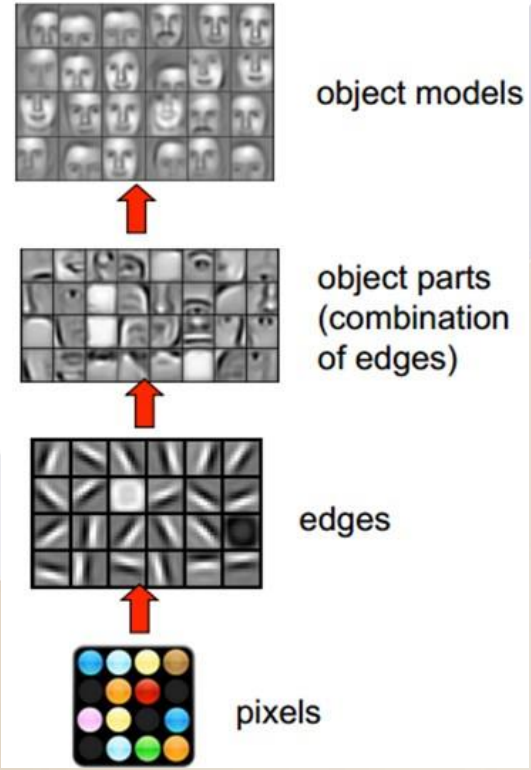
Why deep learning?

- Why neural network?
  - **Universal approximation theorem** (Hornik et al.,1989 Cybenko 1989) states that a feedforward network with a linear output layer and at least one hidden layer can approximate almost any function, given enough hidden units.
  - Huge amount of data lowers the possibility of overfitting.



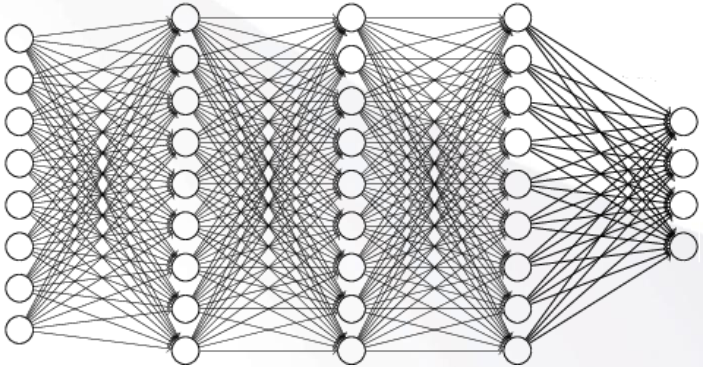
Why deep learning?

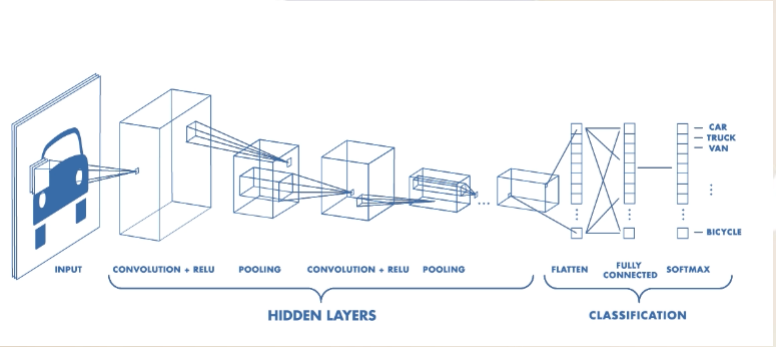
- Why deep?
  - Deeper models reduce the number of units required to represent the function involving composition of simpler functions.



ConVnet for Image Feature Extraction

- Convolutional neural network for images





Source: https://www.mathworks.com/videos/introduction-to-deep -learning-what-are-convolutional-neural-networks--1489512765771.html

ConVnet for Image Feature Extraction

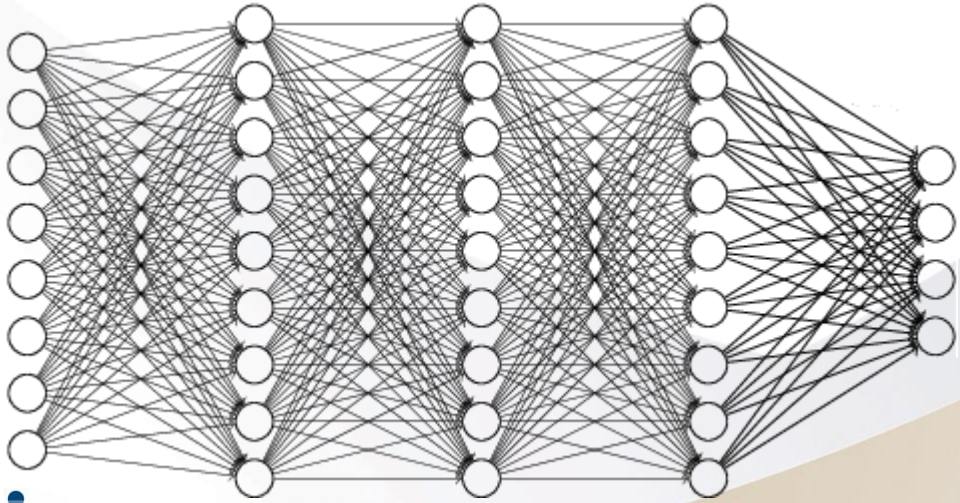
- What makes image different with other machine learning training data?

64*64=4096 pixels
4096*3=12288 dimensions

1000*1000*3=3 million!

ConVnet for Image Feature Extraction

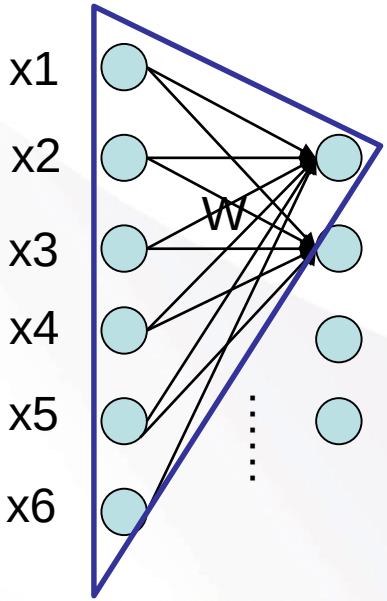
Full connected neural network



Just for the first layer:
3 million * #(first hidden layer nodes) parameters!!

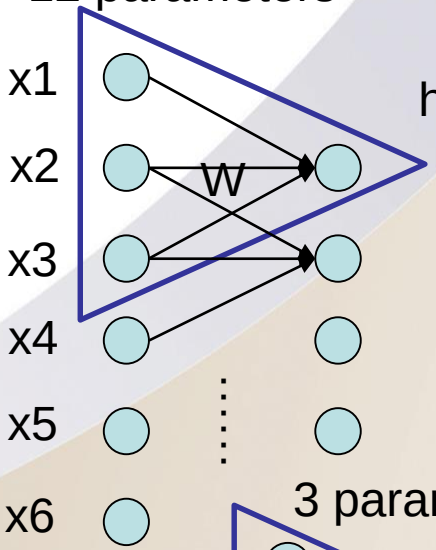
ConVnet for Image Feature Extraction

W:6*4=24 parameters



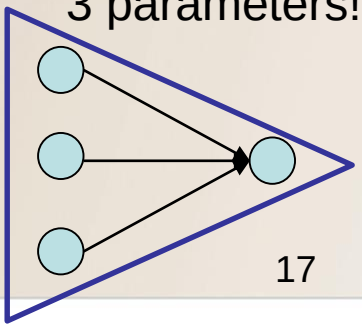
$h1=x1*w_{11} + x2*w_{12} + x3*w_{13} + x4*w_{14} + x5*w_{15} + x6* w_{16}$

3*4=12 parameters



$h1=x1*w_{11} + x2*w_{12} + x3*w_{13}$

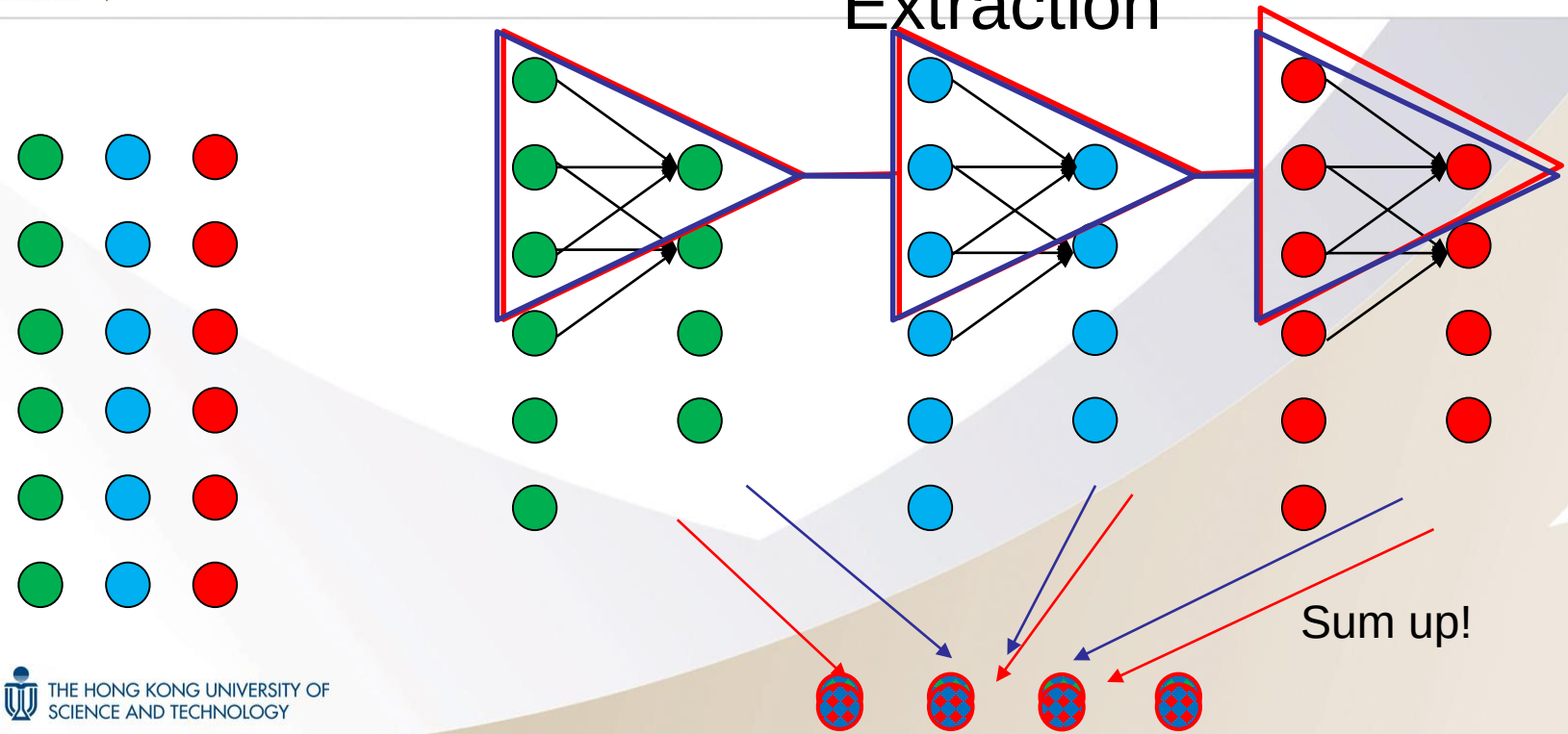
3 parameters!



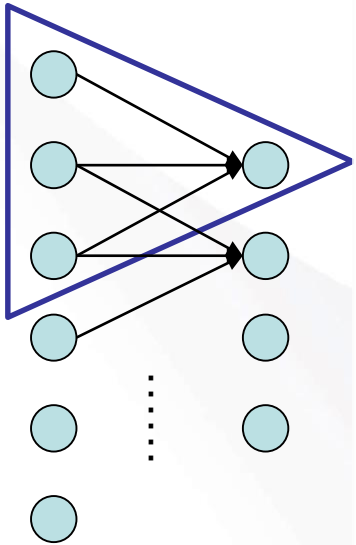
Local receptive field
Shared weights and biases

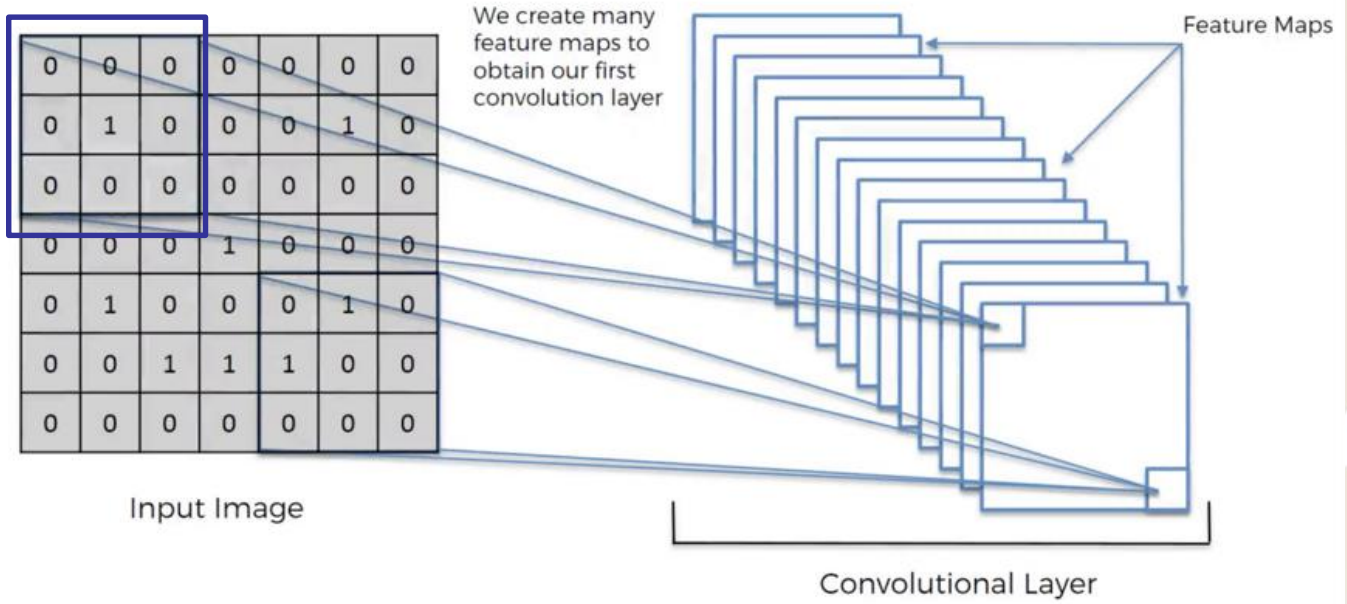
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

ConVnet for Image Feature Extraction

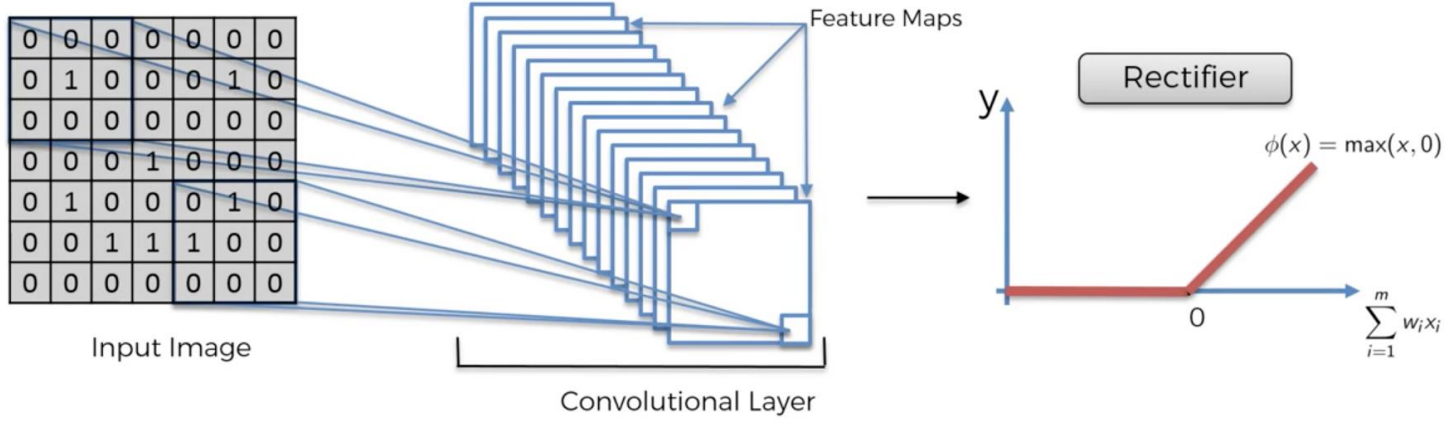


ConVnet for Image Feature Extraction



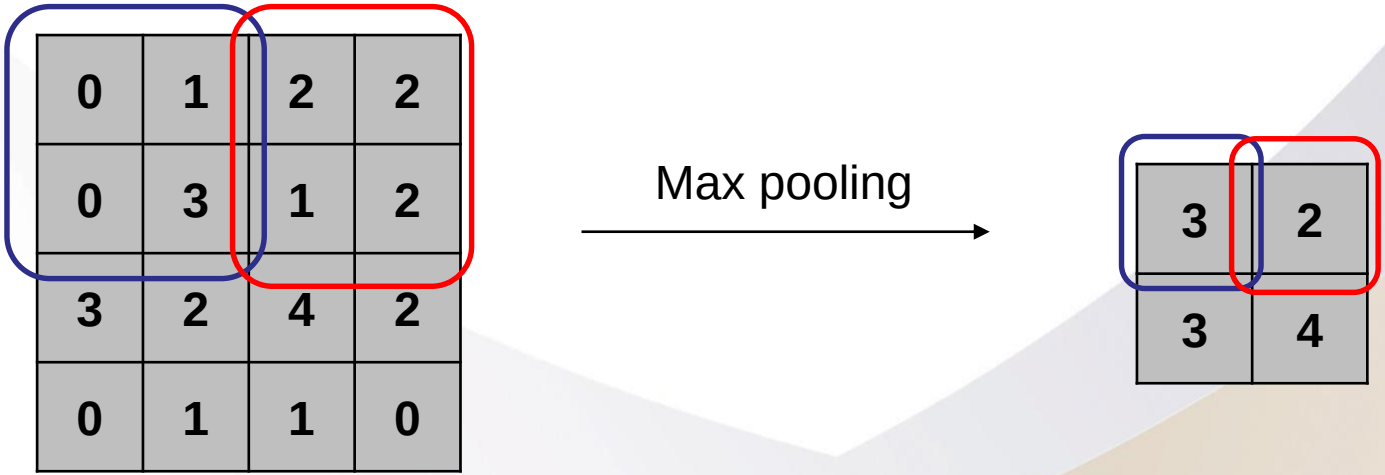


ConVnet for Image Feature Extraction



Activation: make the function non-linear

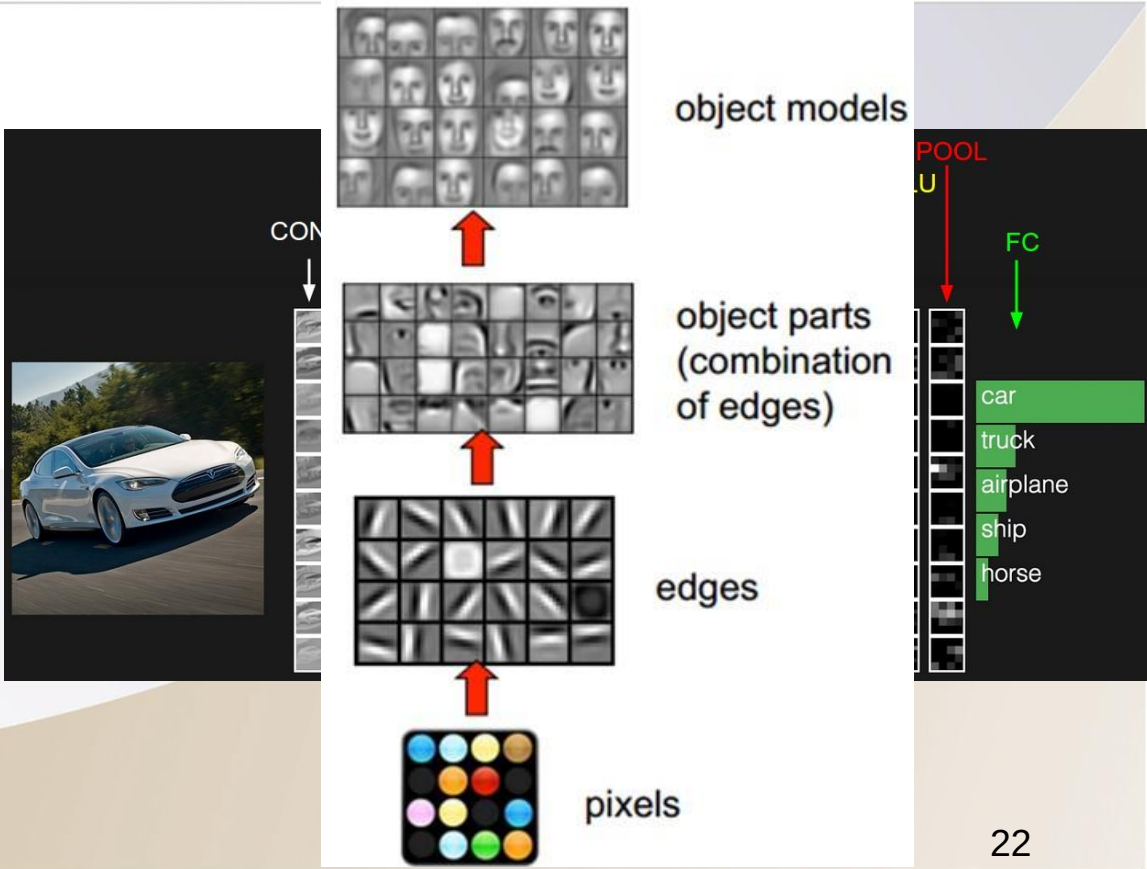
ConVnet for Image Feature Extraction



Pooling: reduce the dimension

CNN for Image Feature Extraction

- Local receptive field
- Shared weights and biases
- Activation
- Pooling



Tensorflow 2.0 features

Eager

Keras

SENG SCHOOL OF ENGINEERING | ENGINEERING OUR FUTURE

Machine Learning Components

Data

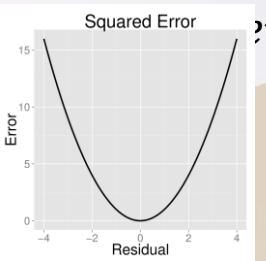
Model

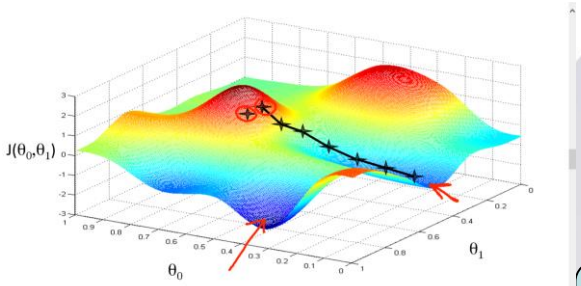
$parameters$

Loss

$$Loss = \sum_{i}^{n} (predicted - truth)^2$$

$$= \quad ers)$$





Optimizer

$$\frac{\partial\ loss}{\partial\ parameters}$$

Metric

THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

Source: https://hackernoon.com/gradient-descent-aynk-7cbe95a778da

Thanks

htianab@connect.ust.hk