CHULA ENGINEERING
Foundation toward Innovation
COMPUTER

Data Mining Group
Machine Intelligence and Knowledge Discovery Lab
Chulalongkorn University

Introduction to Deep Learning

Asst. Prof. Peerapon Vateekul, Ph.D.
Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University
Peerapon.v@chula.ac.th
www.cp.eng.chula.ac.th/~peerapon/

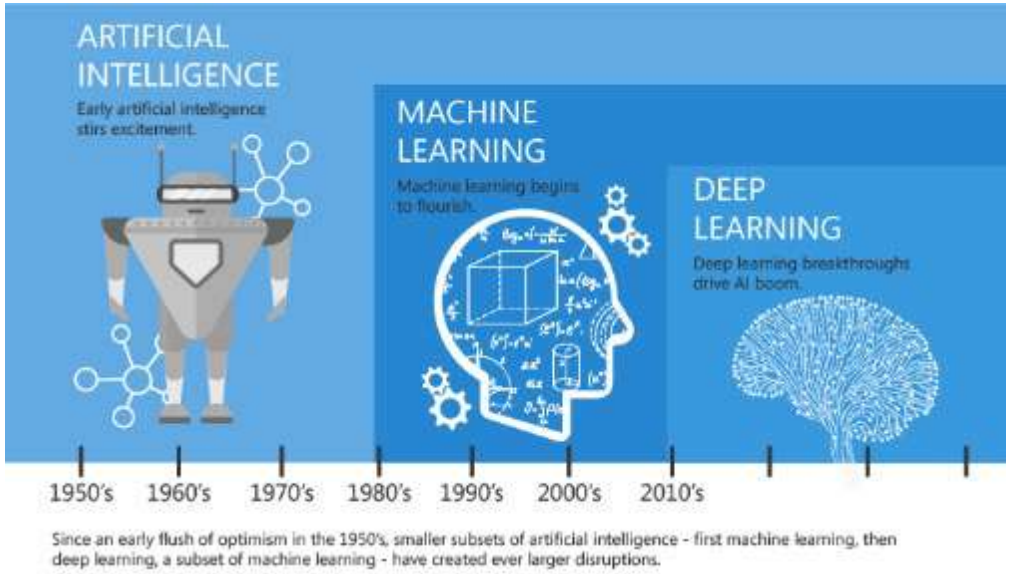
Outline

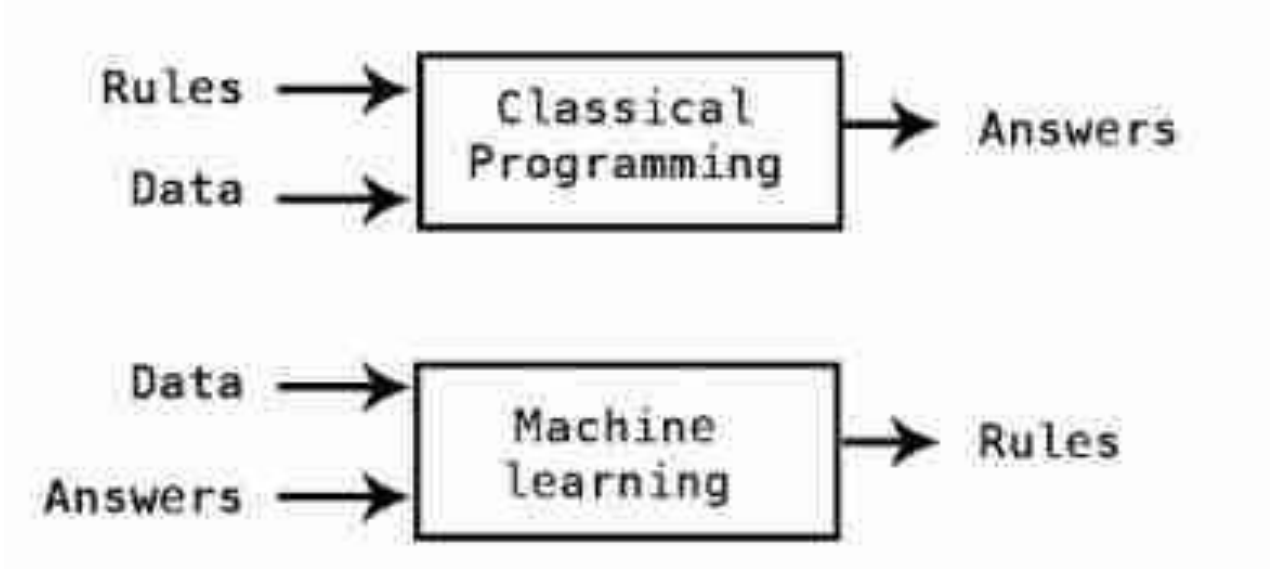
- Introduction to Deep Learning
- Convolutional Neural Networks (CNN) [Imaging Task]
- Recurrent Neural Networks (RNN) [Time Series Forecasting]
- Language Technologies

Introduction to Deep Learning

AI = Automation

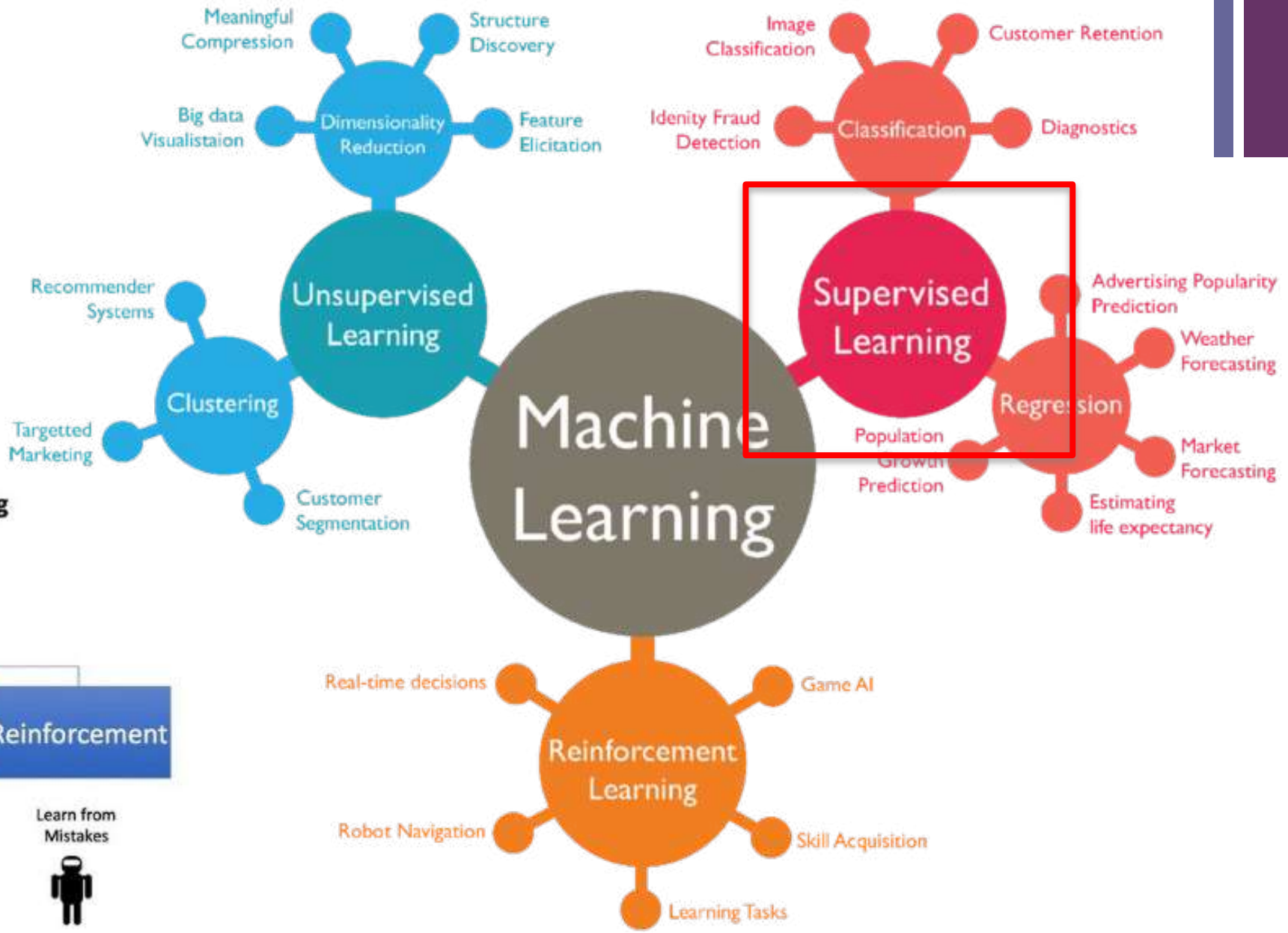
- 1) Rule-based AI (Symbolic AI)
- 2) Machine Learning



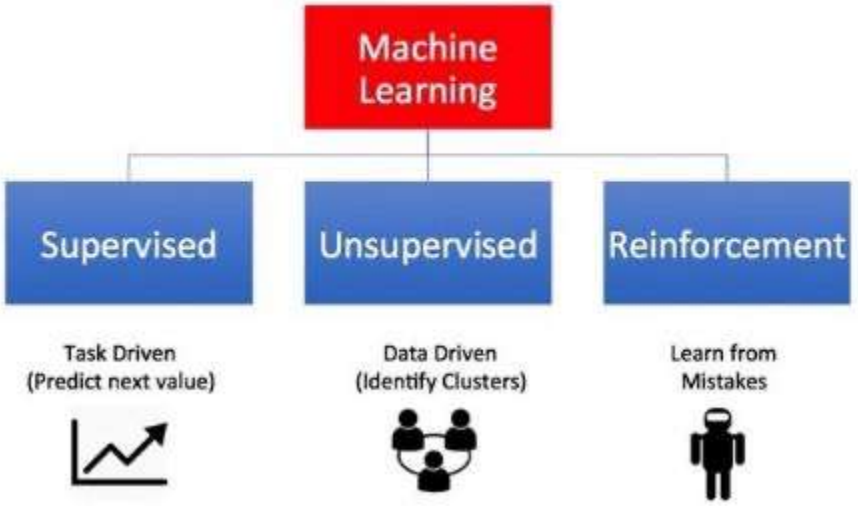


https://mc.ai/machine-learning-basics-artificial-intelligence-machine-learning-and-deep-learning/

Machine Learning



Types of Machine Learning



Task1: Supervised learning

Handcrafted features

Training Data

inputs | target

Age	Gender	BodyTemp	Cough	Corona
12	Female	37	Yes	Yes
35	Female	39	No	Yes
32	Male	38	Yes	No

Testing Data

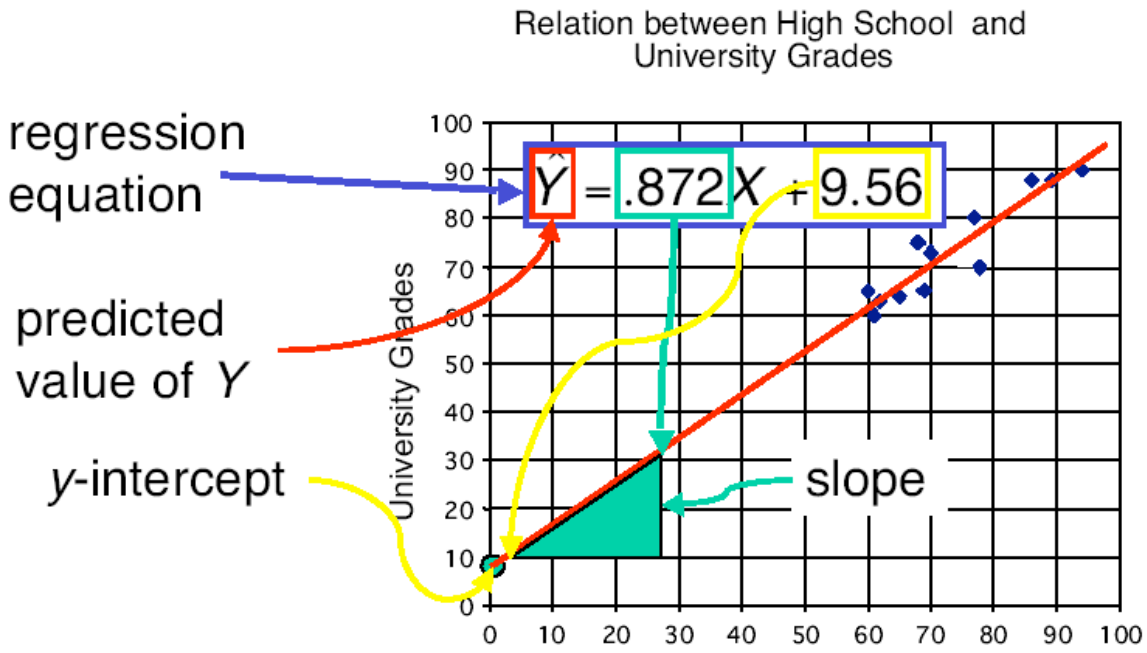
Age	Gender	BodyTemp	Cough	Corona
25	Male	40	No	?

Application: Corona Prediction

Prediction algorithms

- Decision Tree
- (Logistic) Regression
- kNN
- Support Vector Machine
- Neural Networks (NN)
- Deep Learning

Regression – Linear Relationship



weight, coefficient

$$\underset{\text{target}}{\hat{y}} = \underset{\text{intercept}}{\hat{w}_0} + \hat{w}_1 x_1 + \hat{w}_2 \underset{\text{input}}{x_2}$$

- The least square method aims to minimize the following term

$$\sum_{\substack{training \\ data}} (y_i - \hat{y}_i)^2$$

Logistic Regression (cont.)

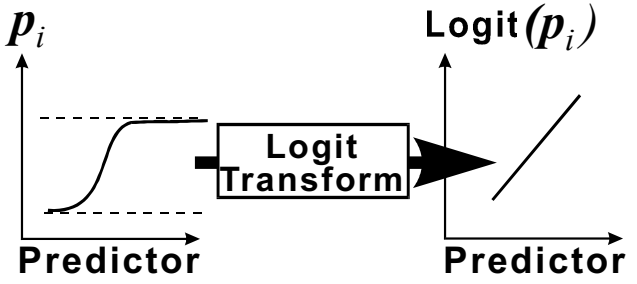
Linear Relationship

Training Data

	inputs			target
Age	**Gender**	**BodyTemp**	**Cough**	**Corona**
12	Female (0)	37	Yes (1)	Yes
35	Female (0)	39	No (0)	Yes
32	Male (1)	38	Yes (1)	No

Logit_score = w0 + w1*Age + w2*Gender + w3*Temp + w4*Cough
Logit_score = 0.01 – 0.3*Age +0.2*Gender + 0.2*Temp + 0.9*Cough

Example
Logit_score = 0.01 – 0.3*12 +0.2*0 + 0.2*37 + 0.9*1 = 4.71
prob = 0.9911 → "Yes"

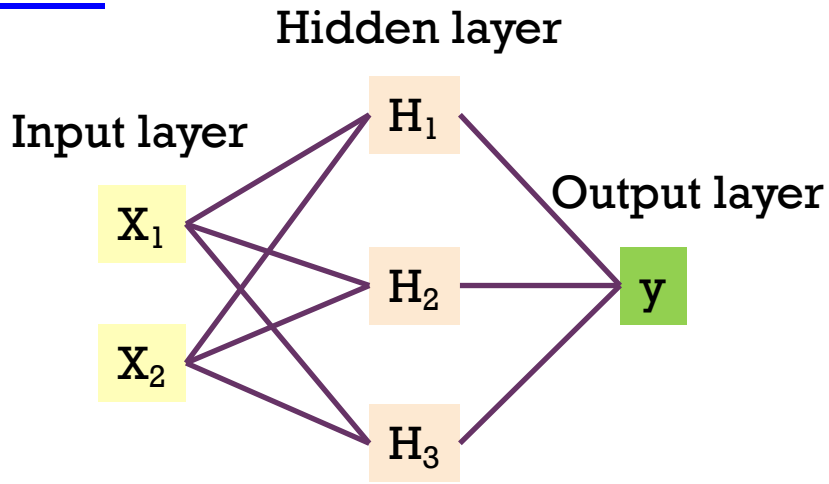


$$\hat{p} = \frac{1}{1 + e^{-\text{logit}(\hat{p})}}$$

Application: Corona Prediction

Neural Networks (universal approximator)

Non-linear relationship

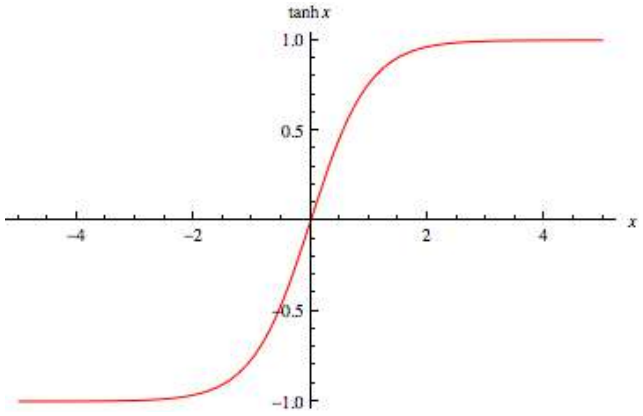


$$log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{w}_0 + \hat{w}_1 H_1 + \hat{w}_2 H_2 + \hat{w}_3 H_3$$

$$H_1 = tanh(\hat{w}_{10} + \hat{w}_{11} x_1 + \hat{w}_{12} x_2)$$

$$H_2 = tanh(\hat{w}_{20} + \hat{w}_{21} x_1 + \hat{w}_{22} x_2)$$

$$H_3 = tanh(\hat{w}_{30} + \hat{w}_{31} x_1 + \hat{w}_{32} x_2)$$

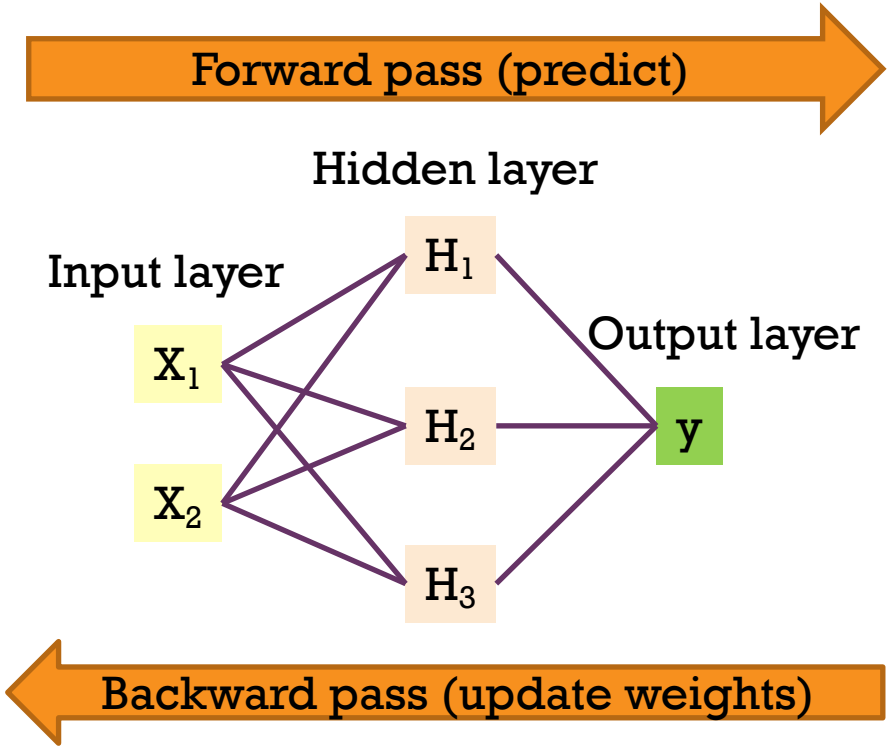


Stop when?
- Converge (no change in loss)
- Max epochs

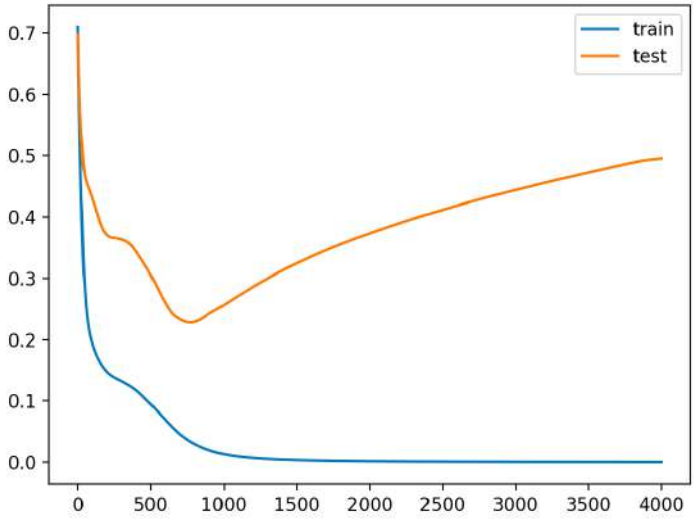
Important Params:
- #hidden units, #hidden layers
- Learning rate, Momentum, decay
- Seed number, etc.

Neural Networks (cont.): Training

Non-linear relationship



Age	Income	Gender	Province	Corona
25	25,000	Female	Bangkok	Yes
35	50,000	Female	Nontaburi	Yes
32	35,000	Male	Bangkok	No



#iteration

$$log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{w}_0 + \hat{w}_1 H_1 + \hat{w}_2 H_2 + \hat{w}_3 H_3$$

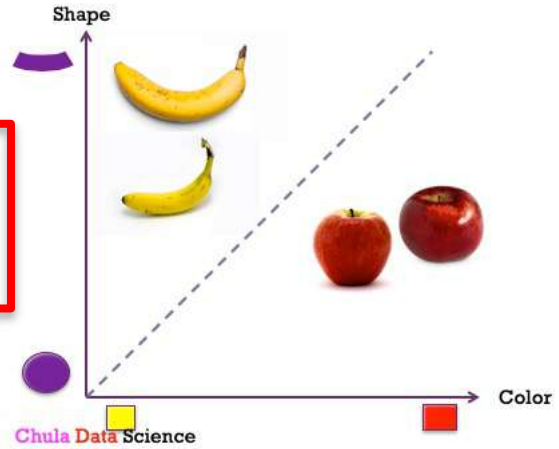
$$H_1 = tanh(\hat{w}_{10} + \hat{w}_{11} x_1 + \hat{w}_{12} x_2)$$

$$H_2 = tanh(\hat{w}_{20} + \hat{w}_{21} x_1 + \hat{w}_{22} x_2)$$

$$H_3 = tanh(\hat{w}_{30} + \hat{w}_{31} x_1 + \hat{w}_{32} x_2)$$

Handcrafted features

Age	Income	Gender	Province	Corona
25	25,000	Female	Bangkok	Yes
35	50,000	Female	Nontaburi	Yes
32	35,000	Male	Bangkok	No

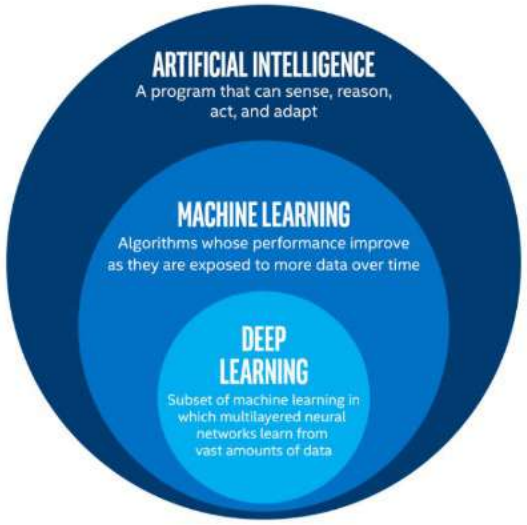


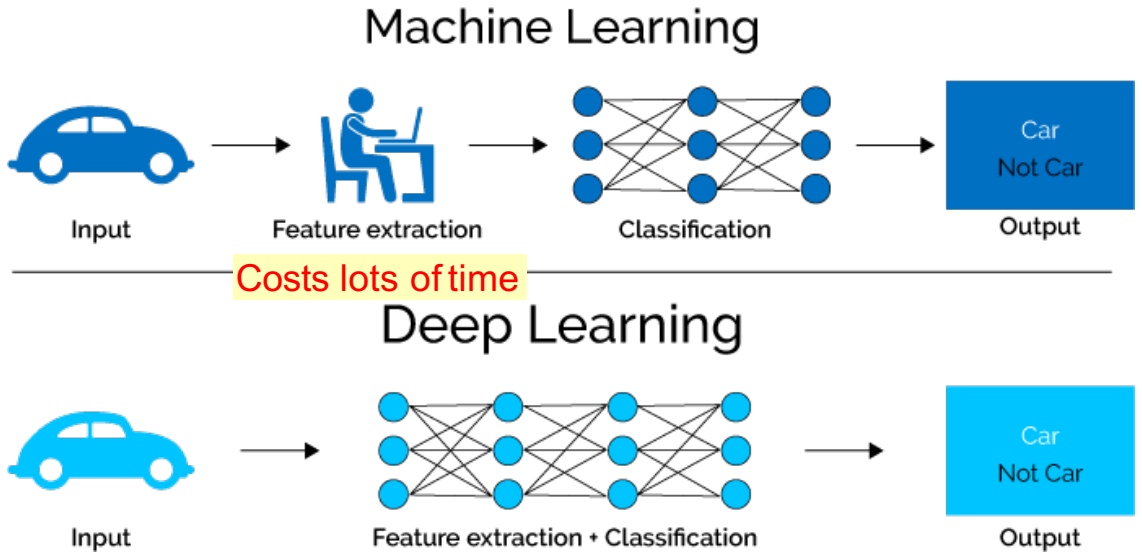
Can we still tell the features (columns)?

What is Deep Learning (DL)?

Part of the machine learning field of learning representations of data. Exceptional effective at learning patterns.

Utilizes learning algorithms that derive meaning out of data by using a hierarchy of multiple layers that mimic the neural networks of our brain.



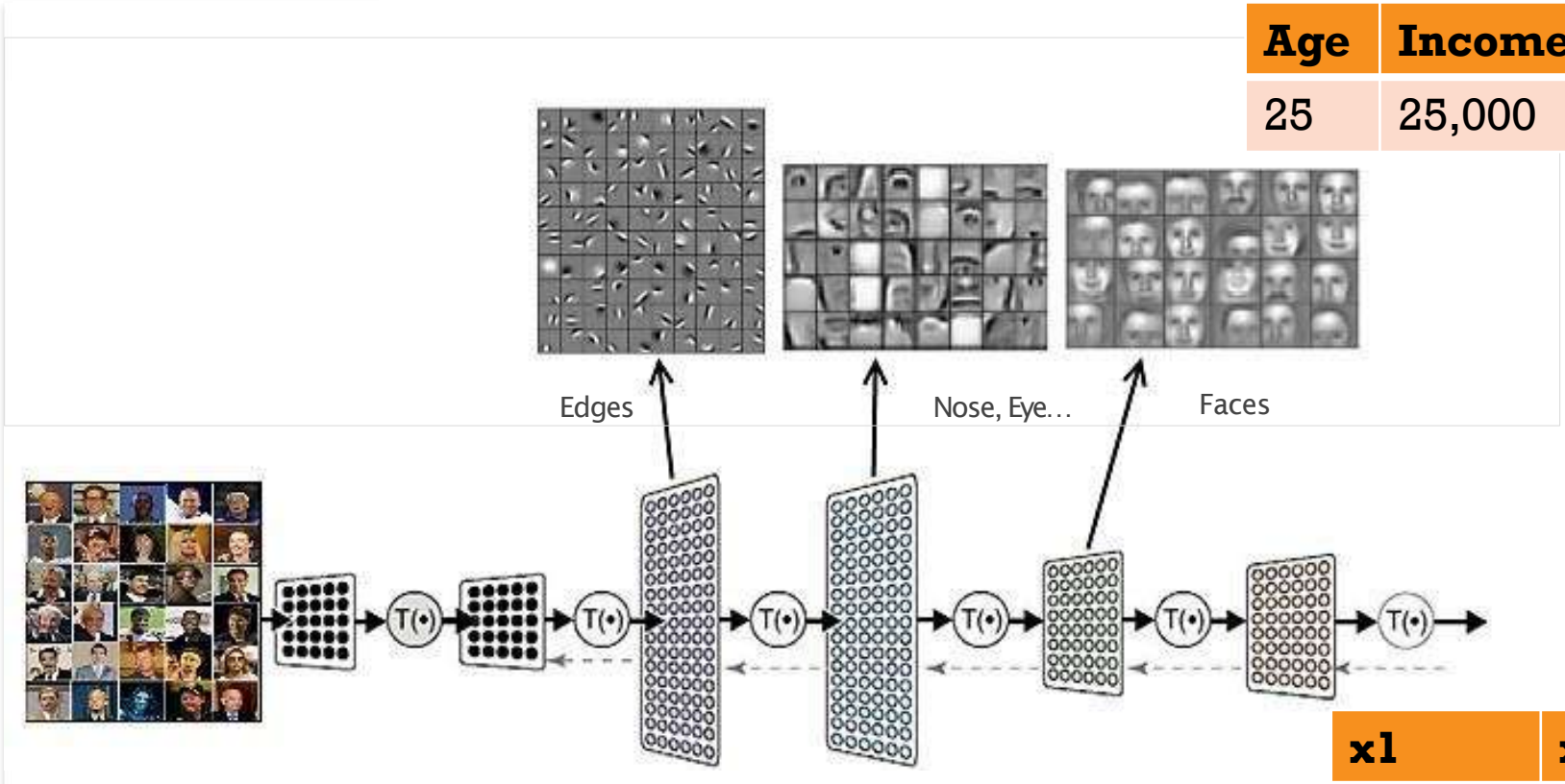


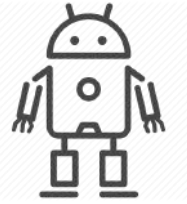
Deep Learning – Basics (cont.)

What did it learn?

A deep neural network consists of a hierarchy of layers, whereby each layer transforms the input data into more abstract representations (e.g., edge -> nose -> face). The output layer combines those features to make predictions.

Age	Income	Gender	Province	Corona
25	25,000	Female	Bangkok	Yes





x1	x2	x3	x4	Corona
0.7	0.2	-0.5	-0.1	Yes

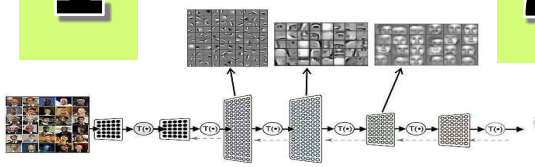
Deep Learning Application

Speech Recognition

Computer Vision

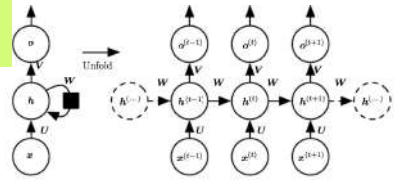
Natural Language Processing

1



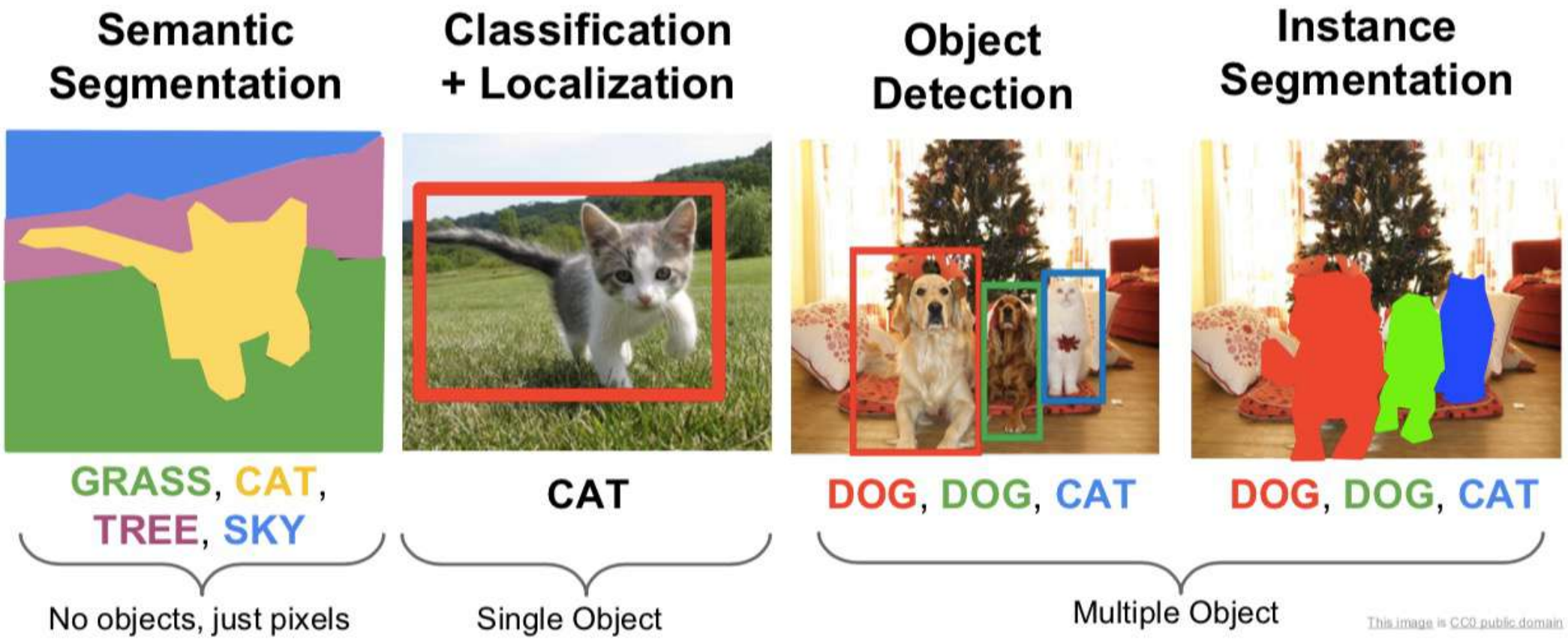
CNN

2



RNN (LSTM)

Type of image tasks

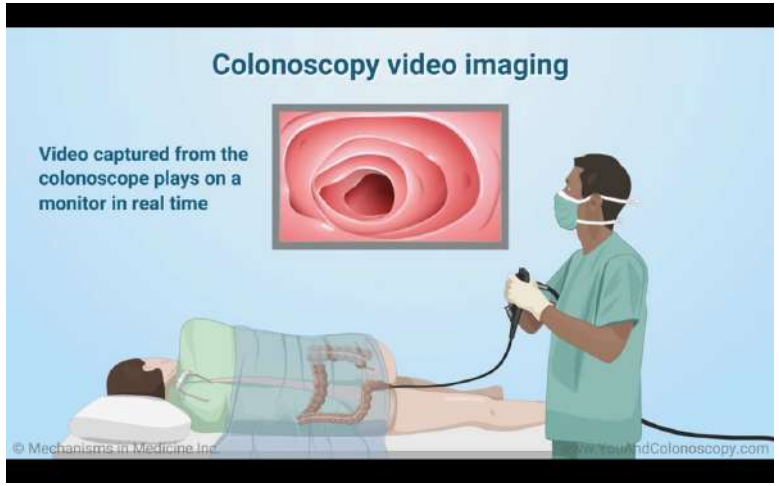


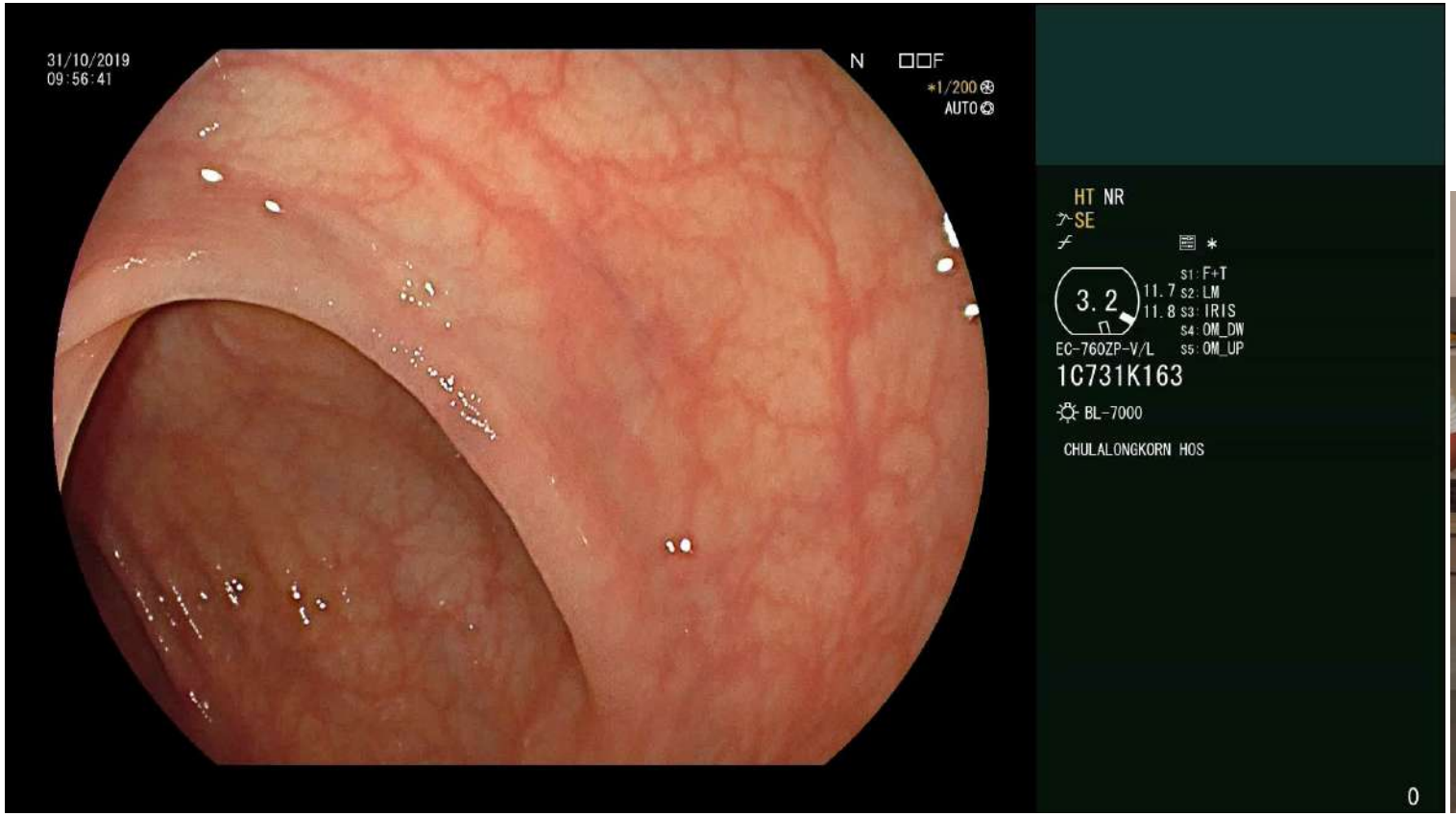
Face Recognition

Smart Medical Diagnosis

CHULA ENGINEERING
Foundation toward Innovation

King Chulalongkorn Memorial Hospital
The Thai Red Cross Society





+

CNN

Outline

- Introduction
- Basic Image Processing
- Overview of CNN
- Image Processing Tasks with SOTA
- Case Study in Polyp Detection

IMAGENET www.image-net.org

The **ImageNet** project is a large visual database designed for use in visual object recognition software research. Over **14M** URLs of images have been hand-annotated by ImageNet to indicate what objects are pictured on **22K** categories.

IMAGENET Large Scale Visual Recognition Challenge

ILSVRC

The Image Classification Challenge:

1,000 object classes

1,431,167 images

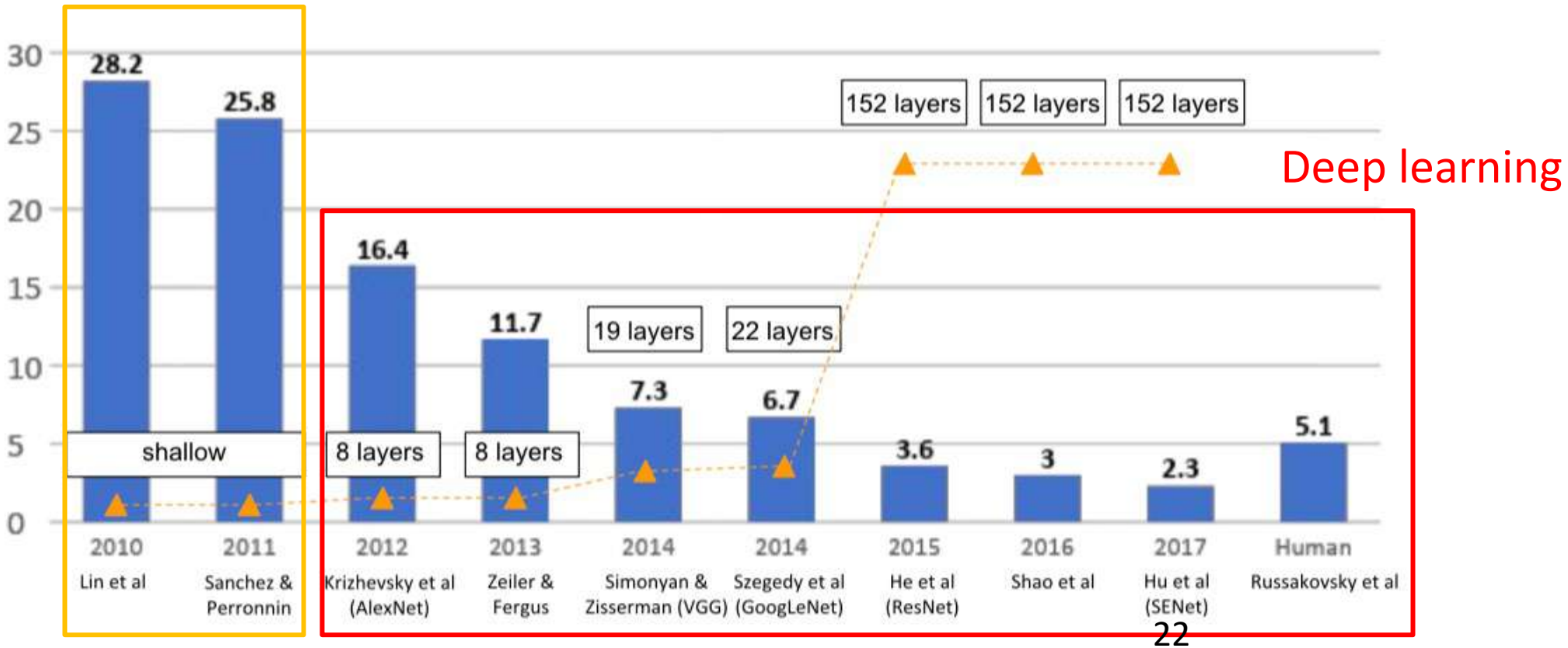
ImageNet 2017 is the last challenge.

21

ImageNet Large Scale Visual Recognition Challenge (ILSVRC winners): From SVM to Deep Learning

CNN

SVM

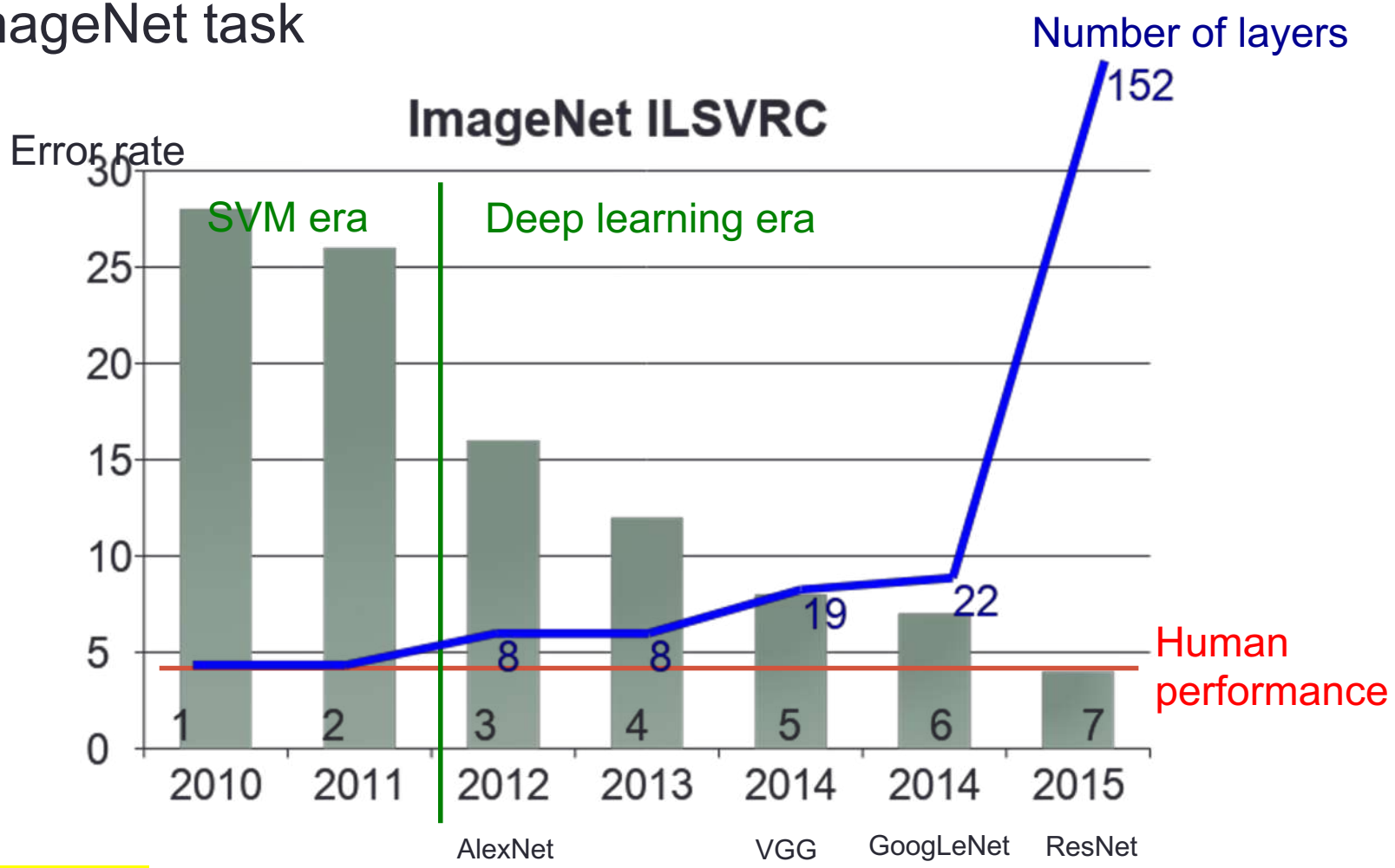


Deep learning

22

Wider and deeper networks (Beyond Human)

- ImageNet task

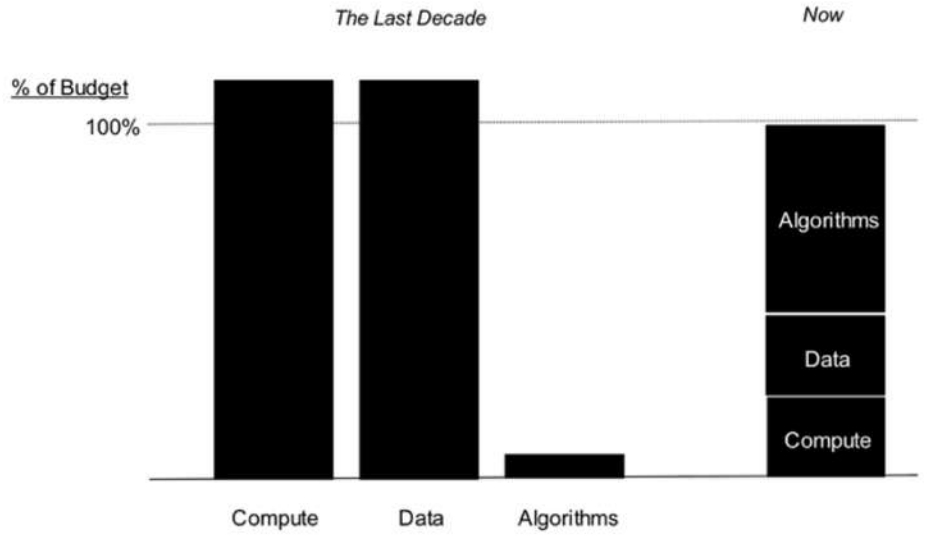


Based on the slide of Aj.Ekapol

eNet Large Scale Visual Recognition Challenge, 2014 https://arxiv.org/abs/1409.0575

Why now

- Neural Networks has been around since 1990s
- Big data – DNN can take advantage of large amounts of data better than other models
- GPU – Enable training bigger models possible
- Deep – Easier to avoid bad local minima when the model is large



Based on the slide of Aj.Ekapol /www.kdnuggets.com/2017/06/practical-guide-machine-learning-understand-differentiate-apply.html

Application 1: Basic Image Processing

https://softwaredevelopmentperestroika.wordpress.com/2014/02/11/image-processing-with-python-numpy-scipy-image-convolution/

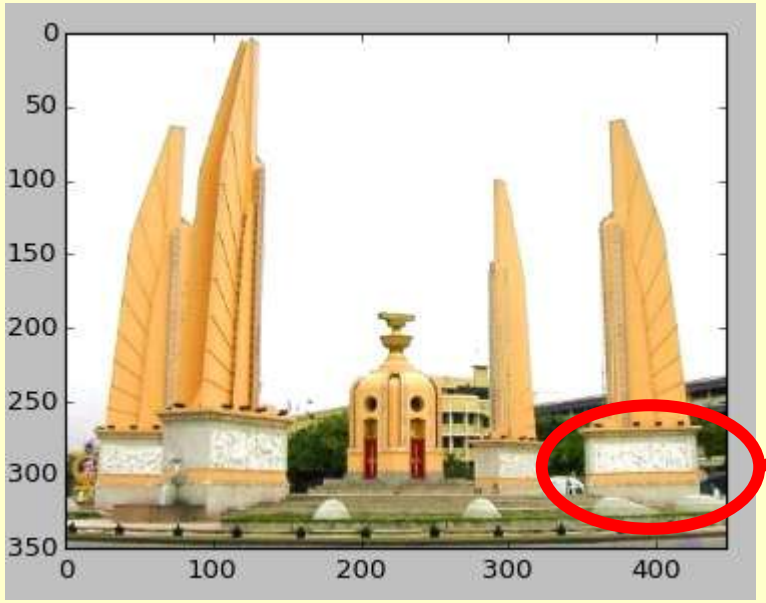
ภาควิชาวิศวกรรมคอมพิวเตอร์
จุฬาลงกรณ์มหาวิทยาลัย

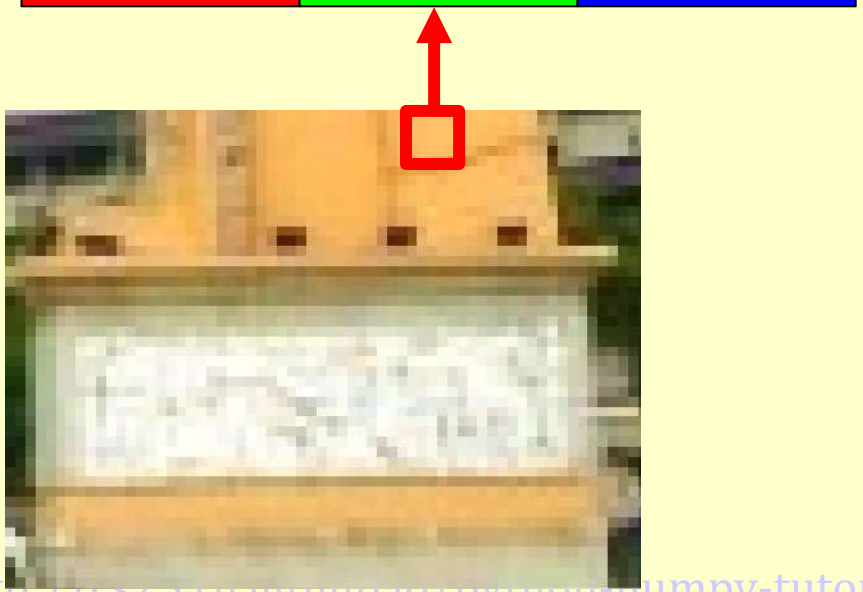
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

image = mpimg.imread('monument.png')
plt.imshow(image)

plt.show()
```

เพื่อความง่ายโปรแกรมที่จะเขียนจากนี้ไปใช้กับแฟ้ม png เท่านั้น

image เป็นอาเรย์ที่แต่ละช่องมีค่า 0 ถึง 1 แทนความเข้มของสี

1 pixel = 1 จุดสี มี 3 สีย่อย

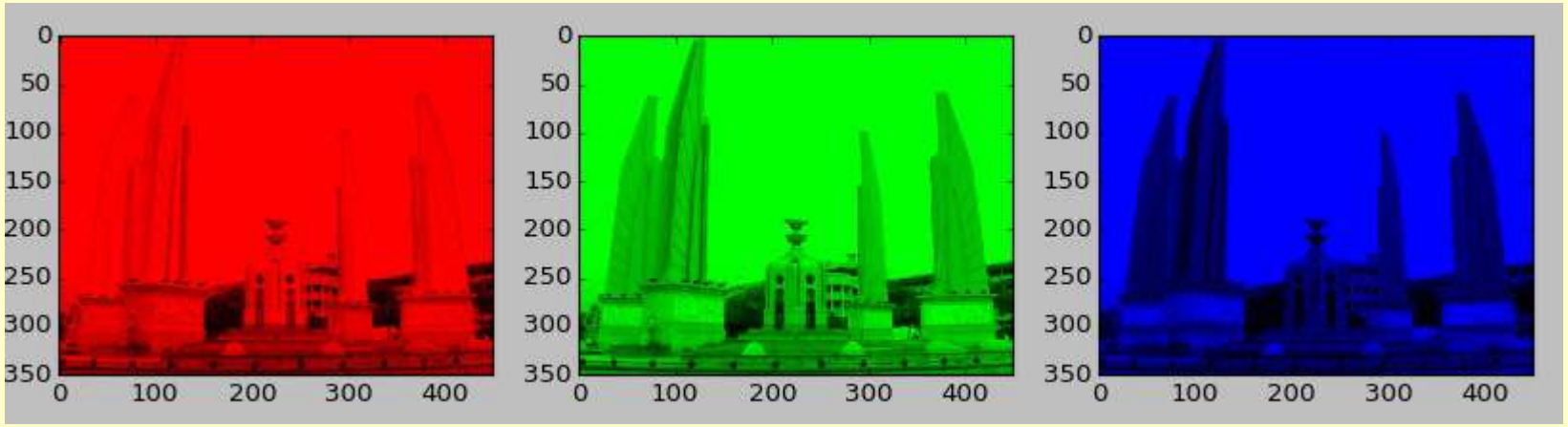




http://cs231n.github.io/python-numpy-tutorial/#numpy

1 ภาพ เป็นอาเรย์ 3 สามมิติเก็บค่าความเข้มของ R G B

```
>>> image.shape
(350, 449, 3)
```



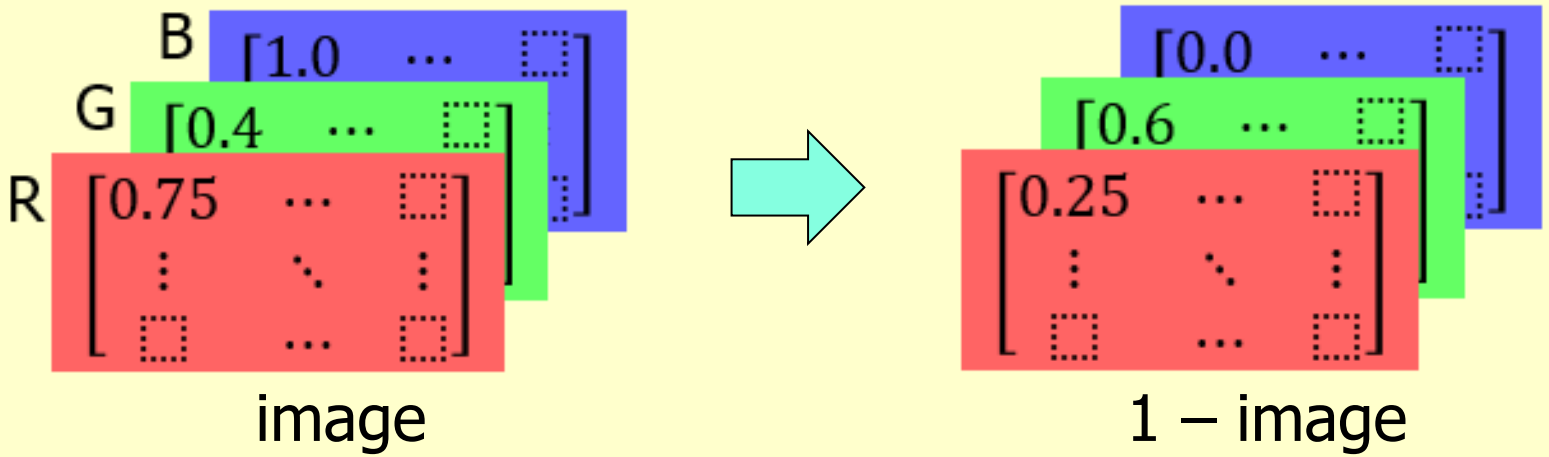
อาร์เรย์ 2 มิติสีแดง
image[:,:,0]

อาร์เรย์ 2 มิติสีเขียว
image[:,:,1]

อาร์เรย์ 2 มิติสีน้ำเงิน
image[:,:,2]

การประมวลผลภาพเบื้องต้น

- image = mpimg.imread('monument.png')
- image.shape → (350, 449, 3)
- image = 1 – image (broadcast & element-wise op.)

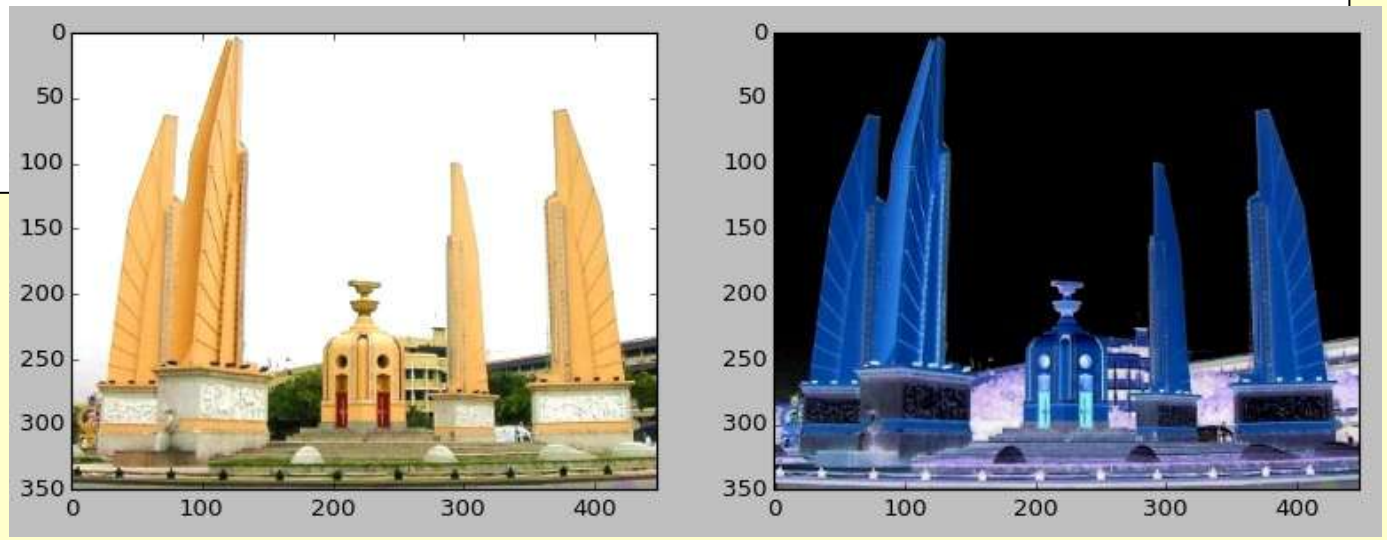


ทำแบบนี้แล้ว ภาพเปลี่ยนแปลงไปอย่างไร ?

การประมวลผลภาพเบื้องต้น : ภาพ Negative

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

image = mpimg.imread('monument.png')
plt.subplot(1, 2, 1)
plt.imshow(image)

negative = 1 - image
plt.subplot(1, 2, 2)
plt.imshow(negative)

plt.show()
```

1 – image คือนำค่าทุกช่องไปลบออกจาก 1 ความเข้มสีเปลี่ยนเป็นตรงข้าม
(ขาว → ดำ, เหลือง → น้ำเงิน, ...)
broadcast & element-wise subtract

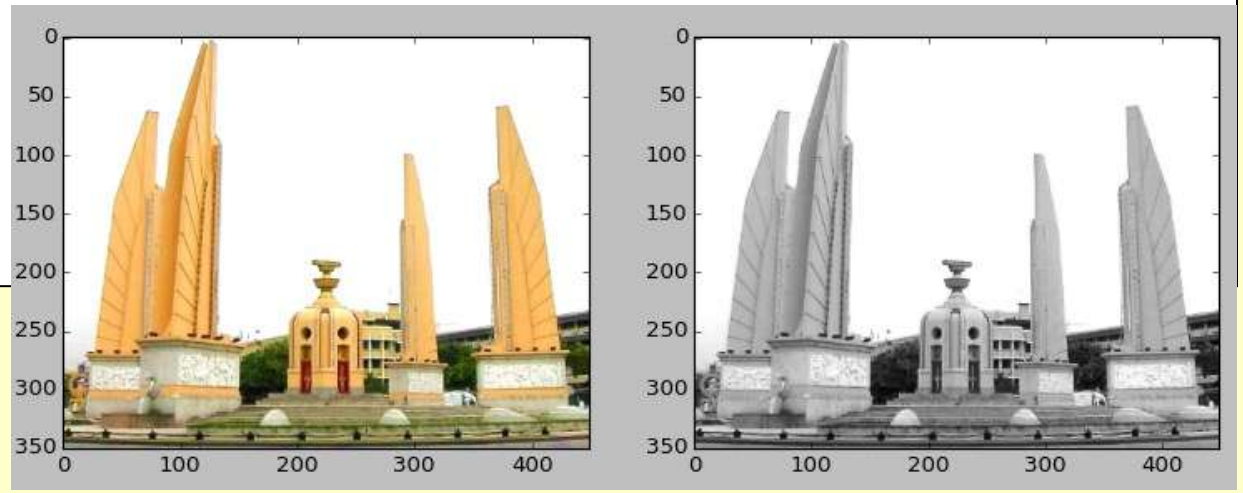


การประมวลผลภาพเบื้องต้น : ภาพสีเทา

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

image = mpimg.imread('monument.png')
plt.subplot(1, 2, 1)
plt.imshow(image)
gray = np.ndarray(image.shape)
gray[:,:,0] = \
gray[:,:,1] = \
gray[:,:,2] = (image[:,:,0]+image[:,:,1]+image[:,:,2]) / 3
plt.subplot(1, 2, 2)
plt.imshow(gray)

plt.show()
```

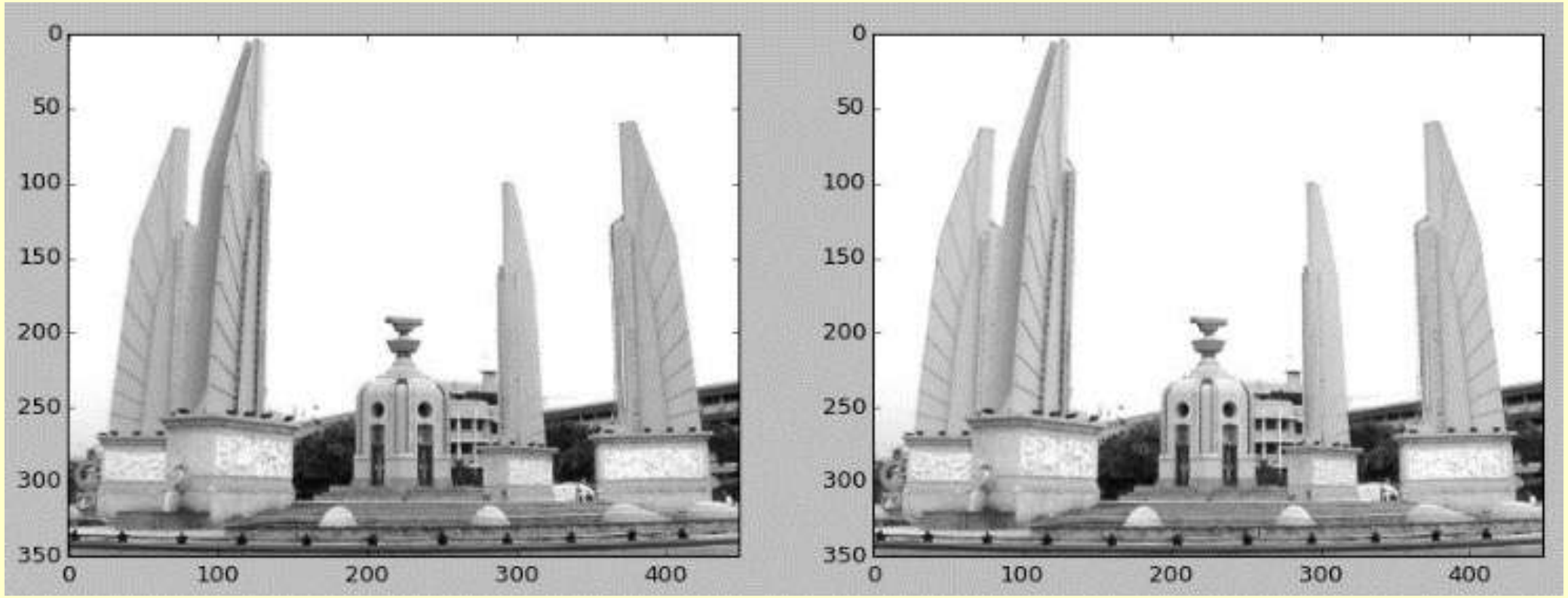
จุดสีที่ R G B มีค่าความเข้มเท่ากัน ได้จุดสีเทา

นำค่าความเข้มของ R G B มาหาค่าเฉลี่ย แล้วเปลี่ยน R G B ให้เป็นความเข้มระดับเดียวกัน

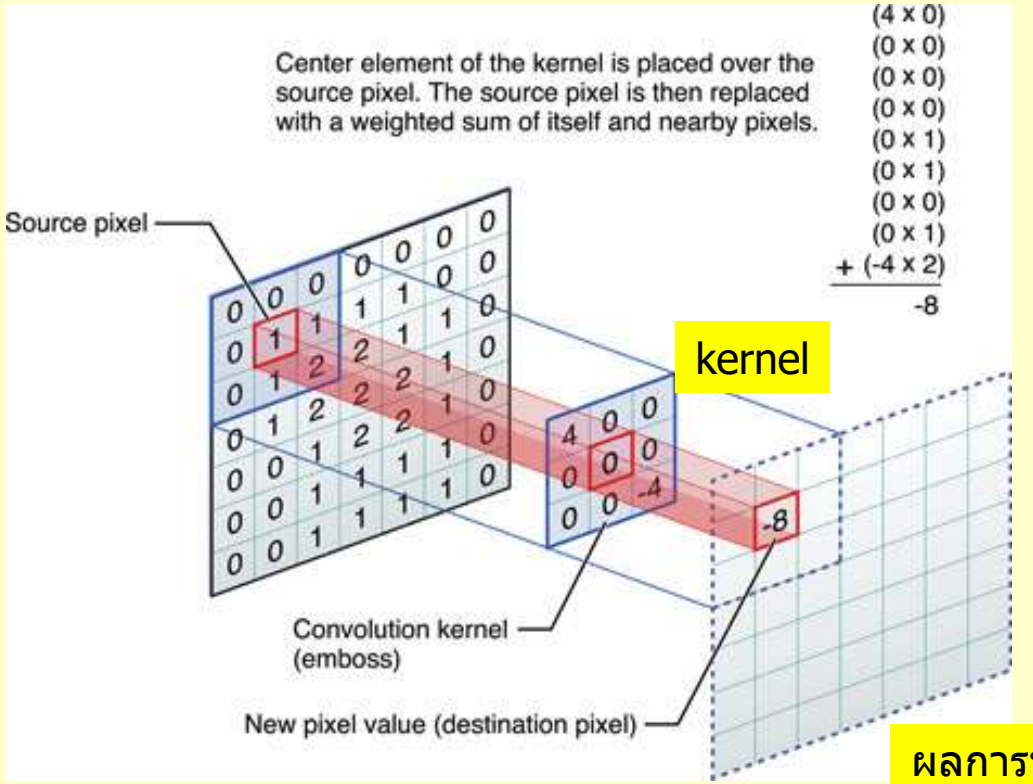


สีเทา เขาว่าสูตรนี้ดีกว่า 0.299*R + 0.587*G + 0.114*B

- CH08_6 จงเขียนโปรแกรมเพื่อแปลงรูปภาพจากรูปสีให้เป็นรูปสีเทา (gray scale) ด้วยสูตรข้างบนนี้ เทียบกับสูตร (R + G + B)/3



Convolution

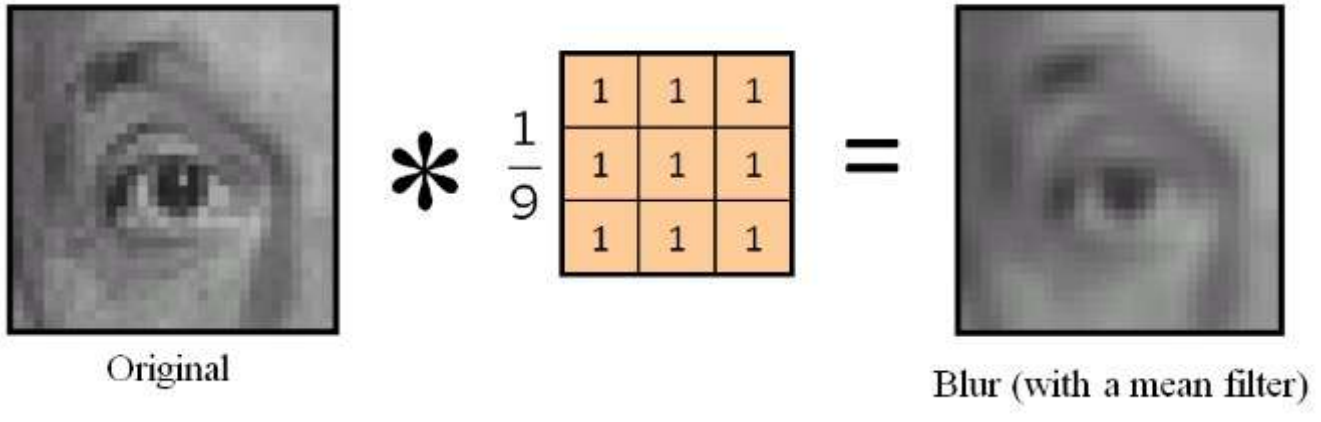


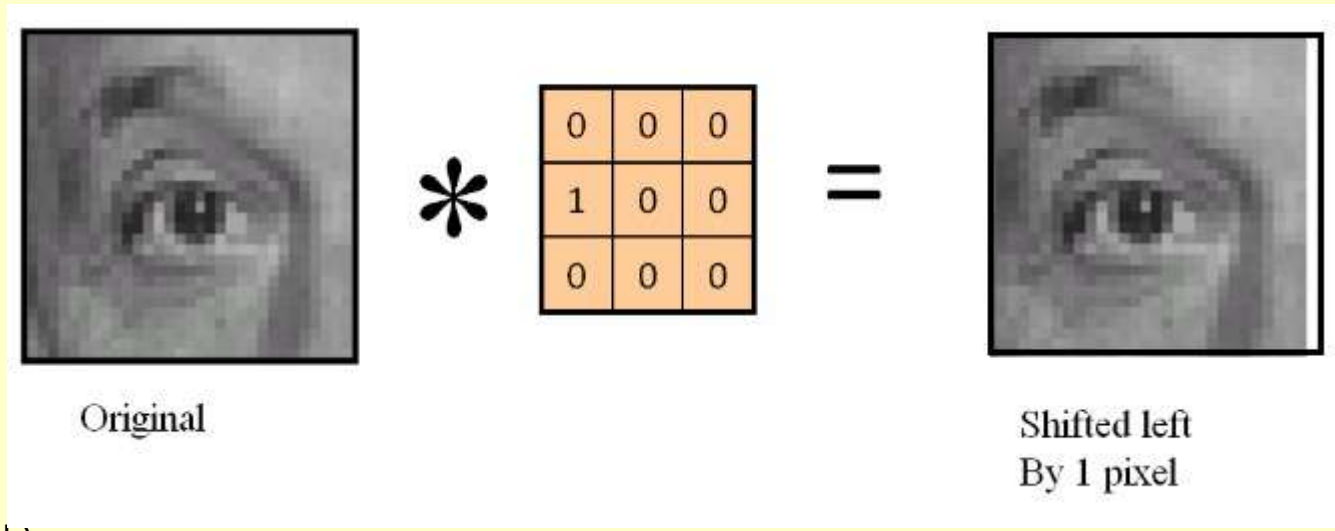
ผลการทำ convolute

เปลี่ยนค่าของเมทริกซ์ kernel จะได้การประมวลผลภาพแบบอื่น ๆ

Image Convolution

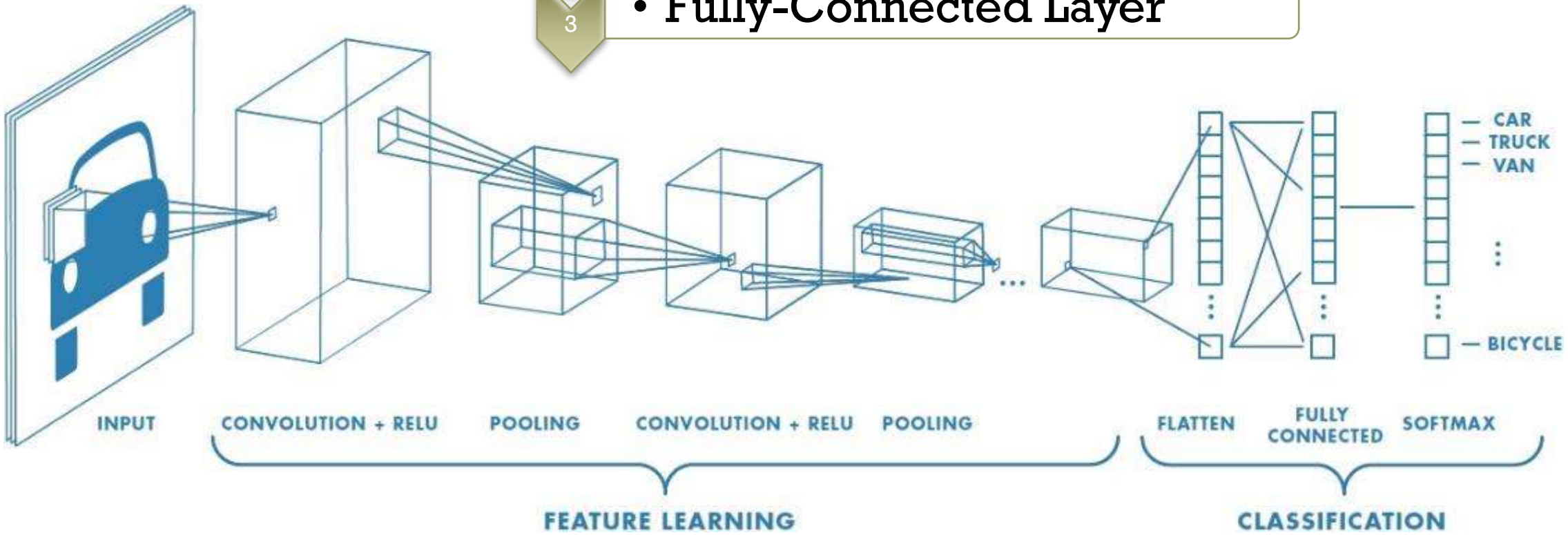
- Image Convolution คือการนำรูปภาพมาผ่านตัวกรอง (kernel) เพื่อให้ได้ผลลัพธ์ตามที่ต้องการ เช่น blur, ขยับรูป, จับขอบรูป เป็นต้น



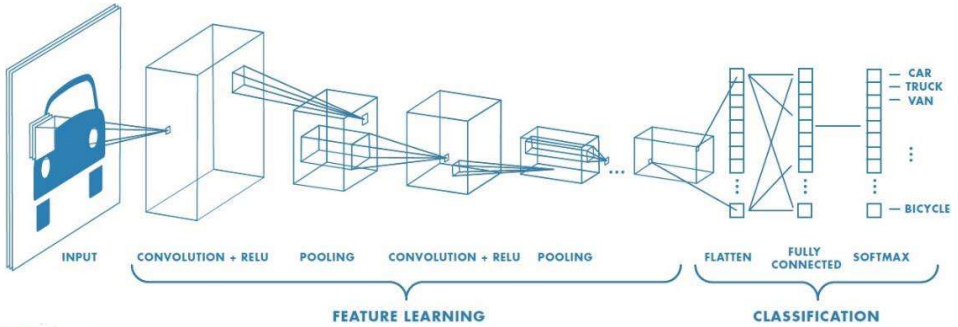


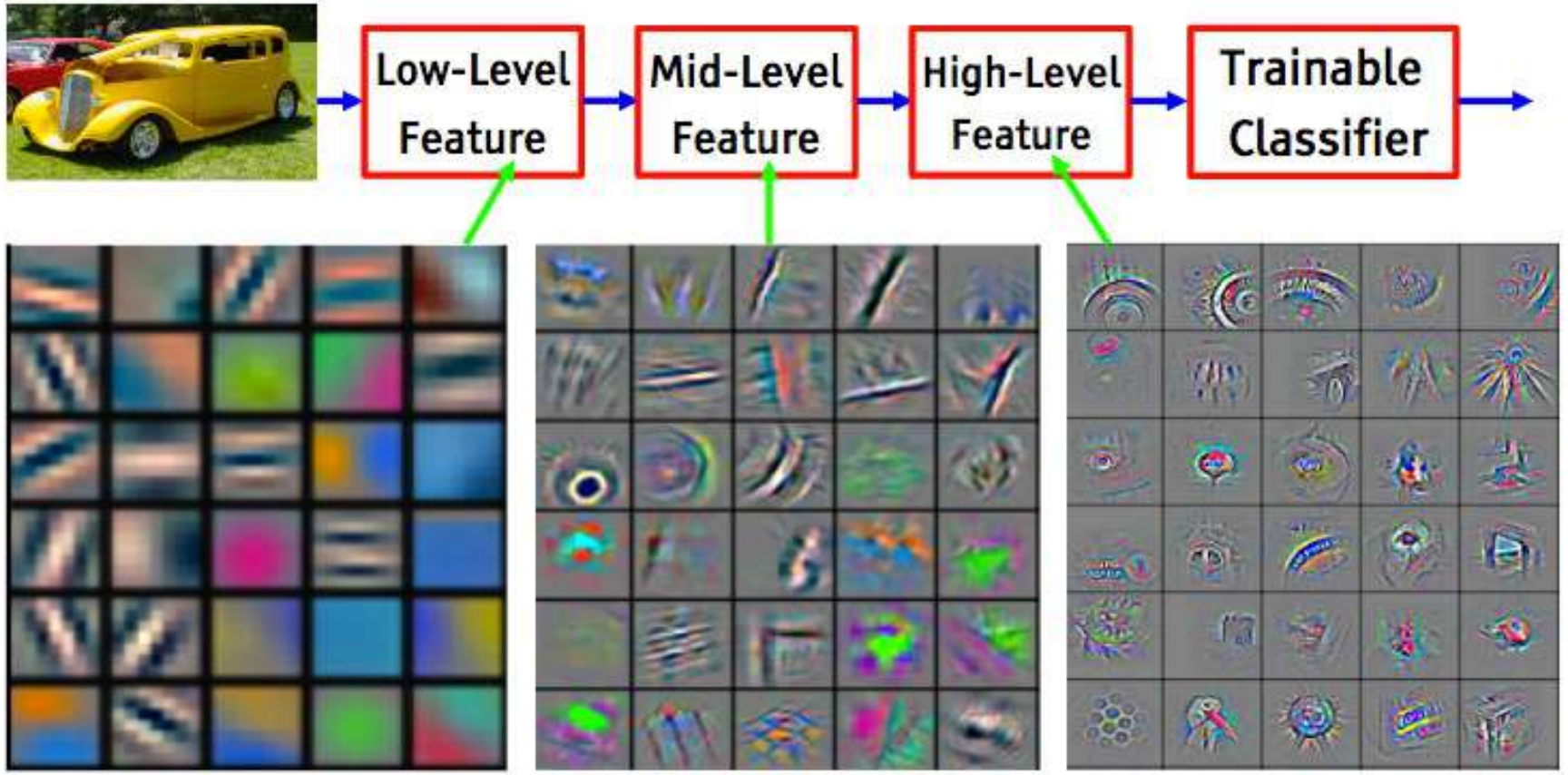
Convolutional Neural Networks (CNN)

1. Convolutional Layer
2. Pooling Layer
3. Fully-Connected Layer



CNN (cont.)

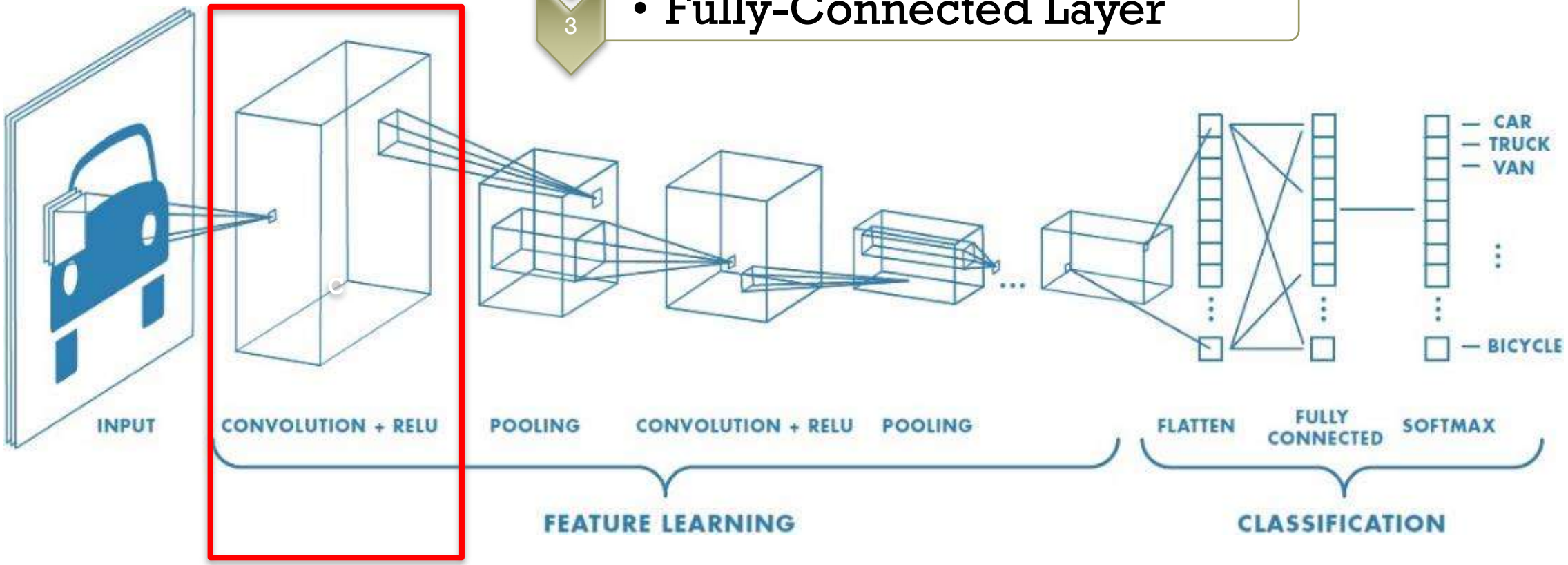

CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION


Low-Level Feature
Mid-Level Feature
High-Level Feature
Trainable Classifier

Convolutional Neural Networks (CNN)

1) Convolutional Layer

1. Convolutional Layer
2. Pooling Layer
3. Fully-Connected Layer

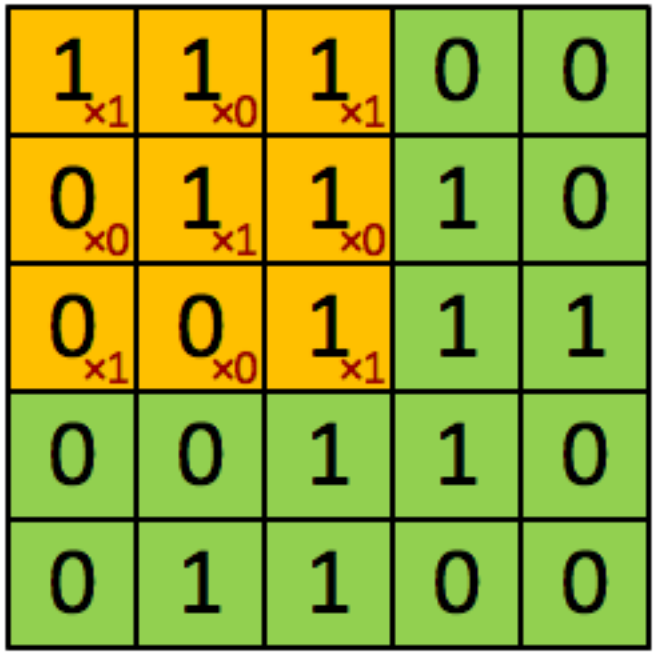


Layer 1: Convolutional Layer

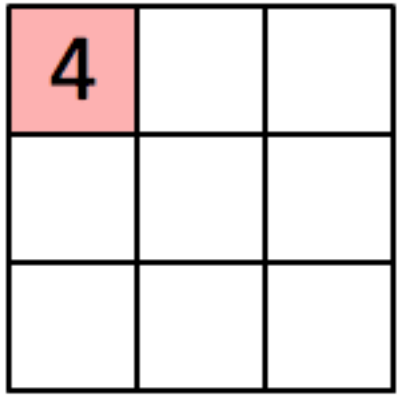
(3*3 filter)

1	0	1
0	1	0
1	0	1

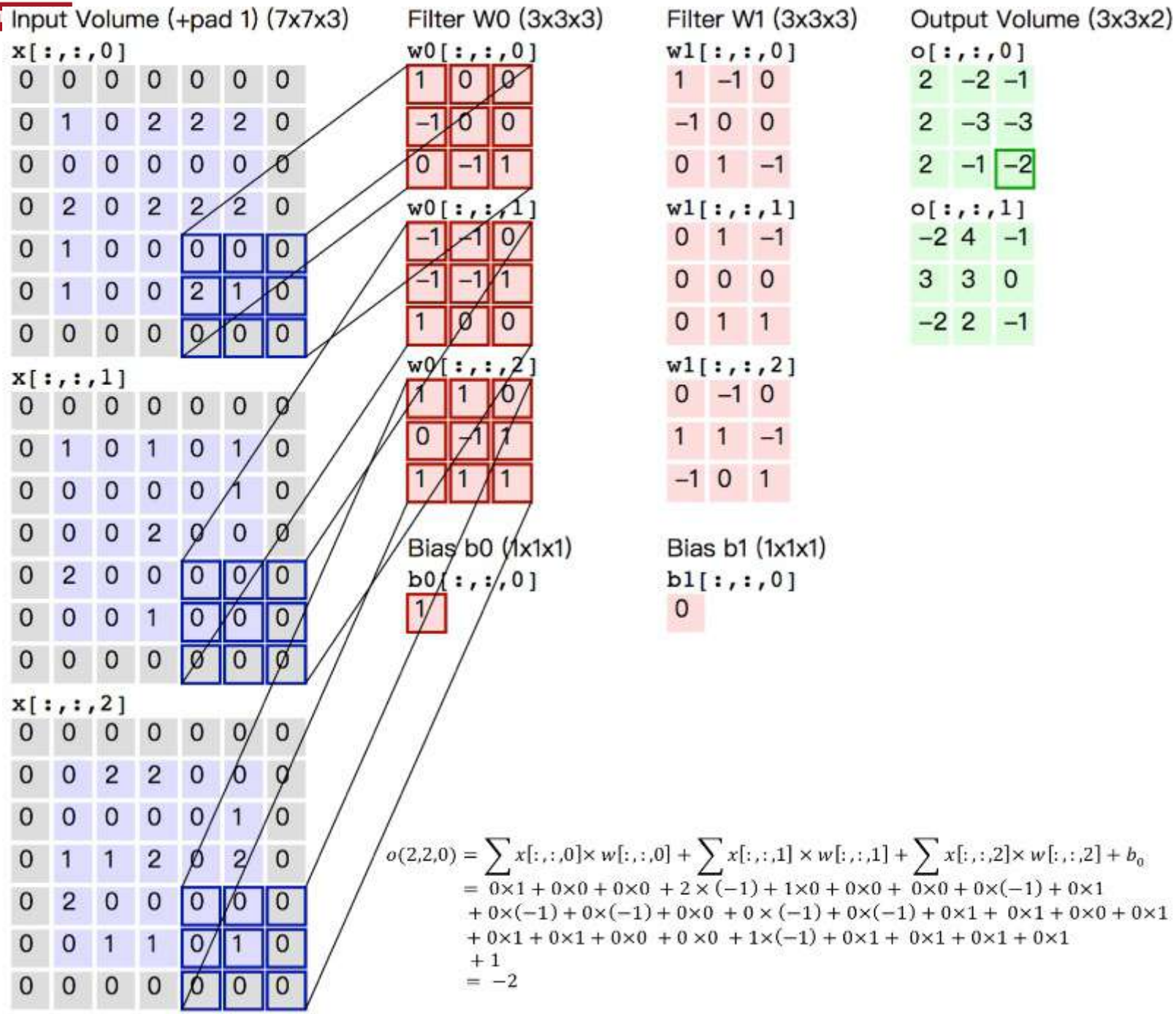
1. Convolutional Layer
2. Pooling Layer
3. Fully-Connected Layer



Image



Convolved Feature



$$
\begin{aligned}
o(2,2,0) &= \sum x[:,:,0]\times w[:,:,0] + \sum x[:,:,1]\times w[:,:,1] + \sum x[:,:,2]\times w[:,:,2] + b_0 \\
&= 0\times1 + 0\times0 + 0\times0 + 2\times(-1) + 1\times0 + 0\times0 + 0\times0 + 0\times(-1) + 0\times1 \\
&\quad + 0\times(-1) + 0\times(-1) + 0\times0 + 0\times(-1) + 0\times(-1) + 0\times1 + 0\times1 + 0\times0 + 0\times1 \\
&\quad + 0\times1 + 0\times1 + 0\times0 + 0\times0 + 1\times(-1) + 0\times1 + 0\times1 + 0\times1 + 0\times1 \\
&\quad + 1 \\
&= -2
\end{aligned}
$$

CNNs: 1) Convolutional Layer

Feature Extraction

http://cs231n.github.io/convolutional-networks/

$$w^T x + b$$

(3*3 filter)

1	0	1
0	1	0
1	0	1

$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$	0	0
$0_{\times 0}$	$1_{\times 1}$	$1_{\times 0}$	1	0
$0_{\times 1}$	$0_{\times 0}$	$1_{\times 1}$	1	1
0	0	1	1	0
0	1	1	0	0

Image

4		

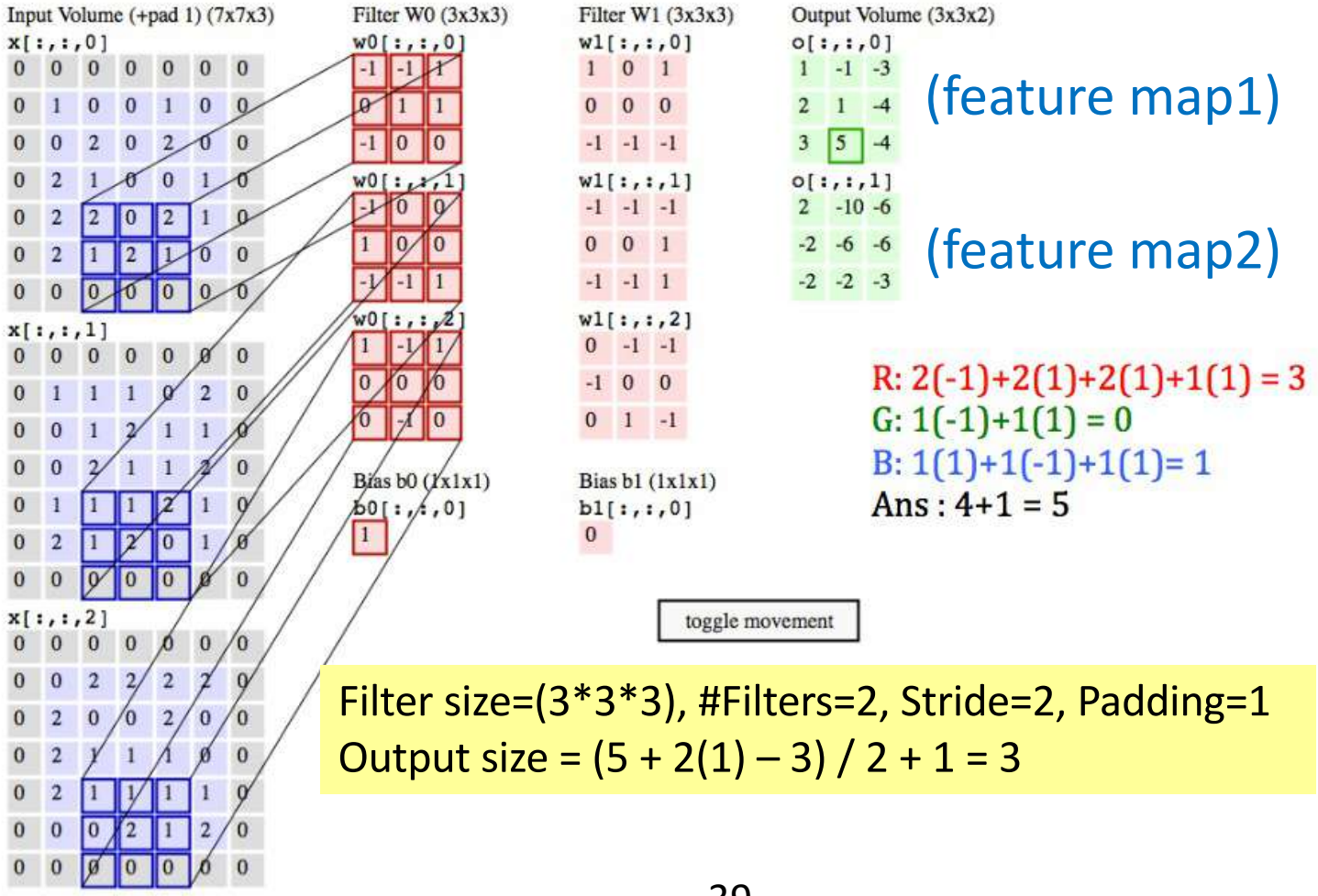
Convolved Feature (feature map)

Filter size=(3*3), #Filters=1, Stride=1, Padding=0
Output size = (5 +2(0) – 3) / 1 + 1 = 3

Summary: **4 parameters** for convolutional layer

- (1) Filter size, (2) #Filters, (3) Stride, (4) Padding

Output size = (N + 2*P – F) / S + 1



(feature map1)

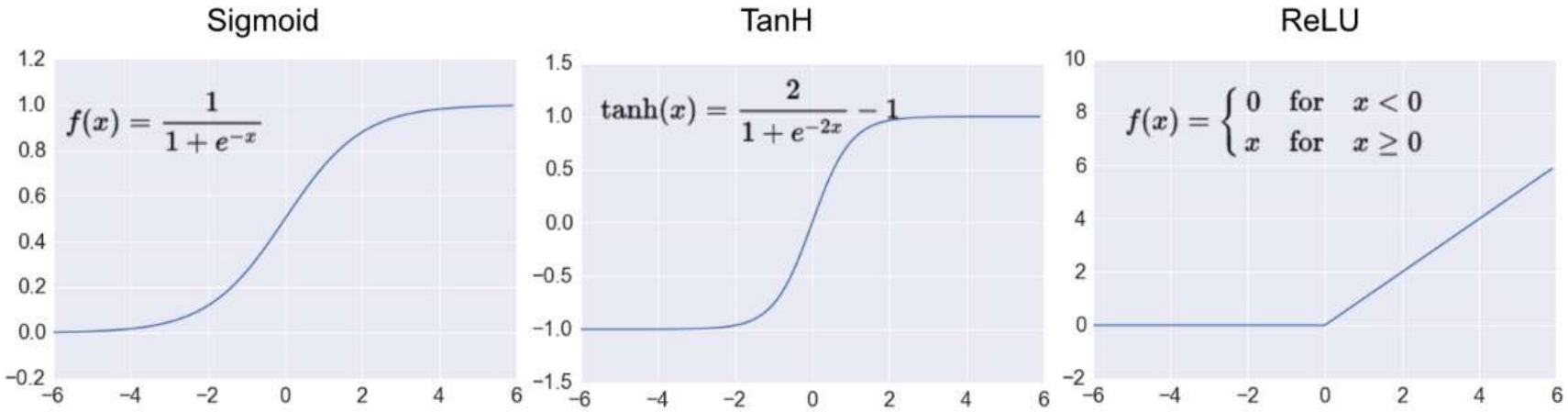
(feature map2)

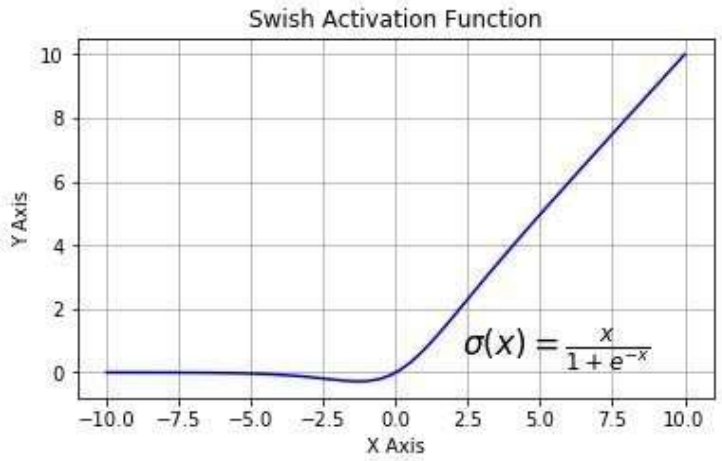
R: 2(-1)+2(1)+2(1)+1(1) = 3
G: 1(-1)+1(1) = 0
B: 1(1)+1(-1)+1(1)= 1
Ans : 4+1 = 5

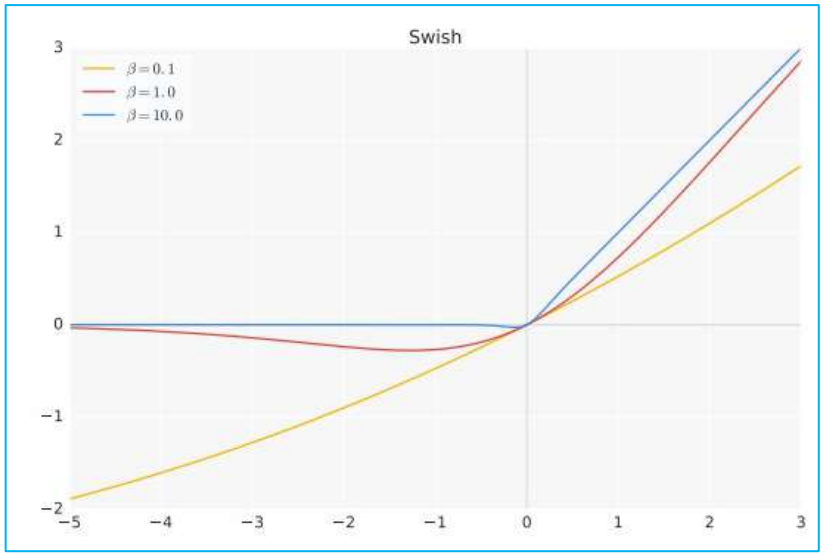
Filter size=(3*3*3), #Filters=2, Stride=2, Padding=1
Output size = (5 + 2(1) – 3) / 2 + 1 = 3

CNNs: 1) Convolutional Layer (cont.)

Non-Linear Activation Function



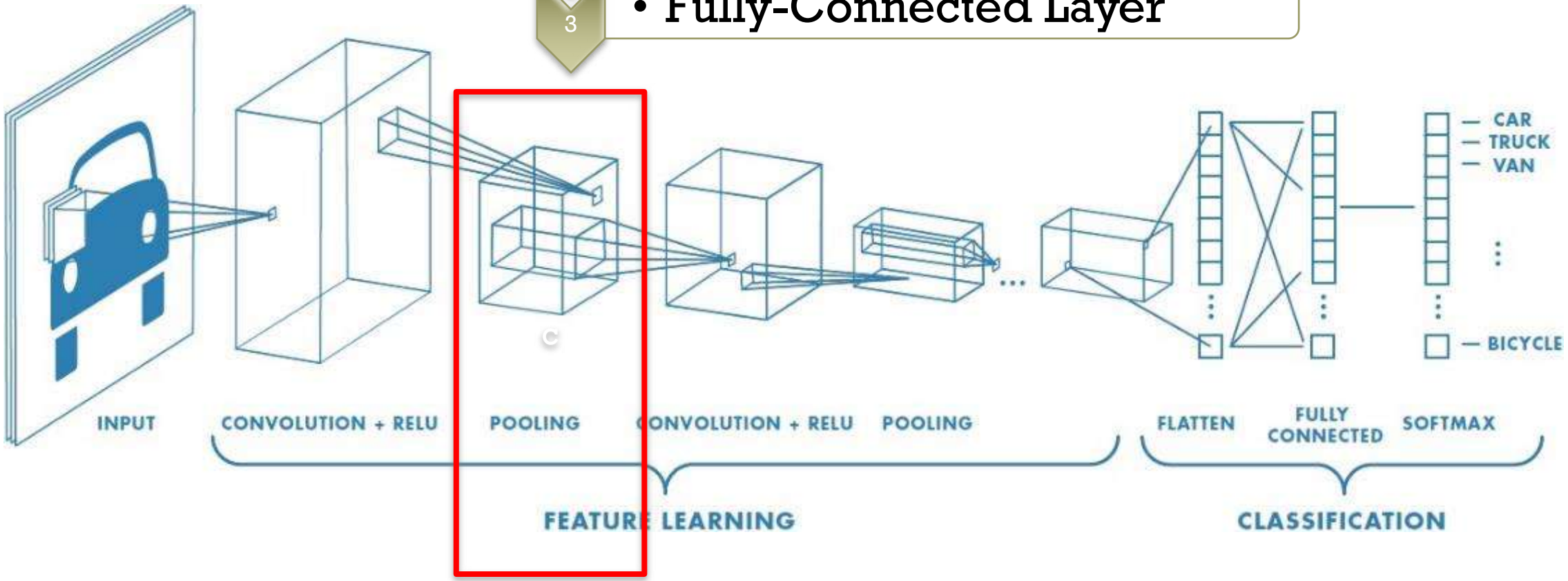




Convolutional Neural Networks (CNN)

2) Pooling Layer

1. Convolutional Layer
2. Pooling Layer
3. Fully-Connected Layer

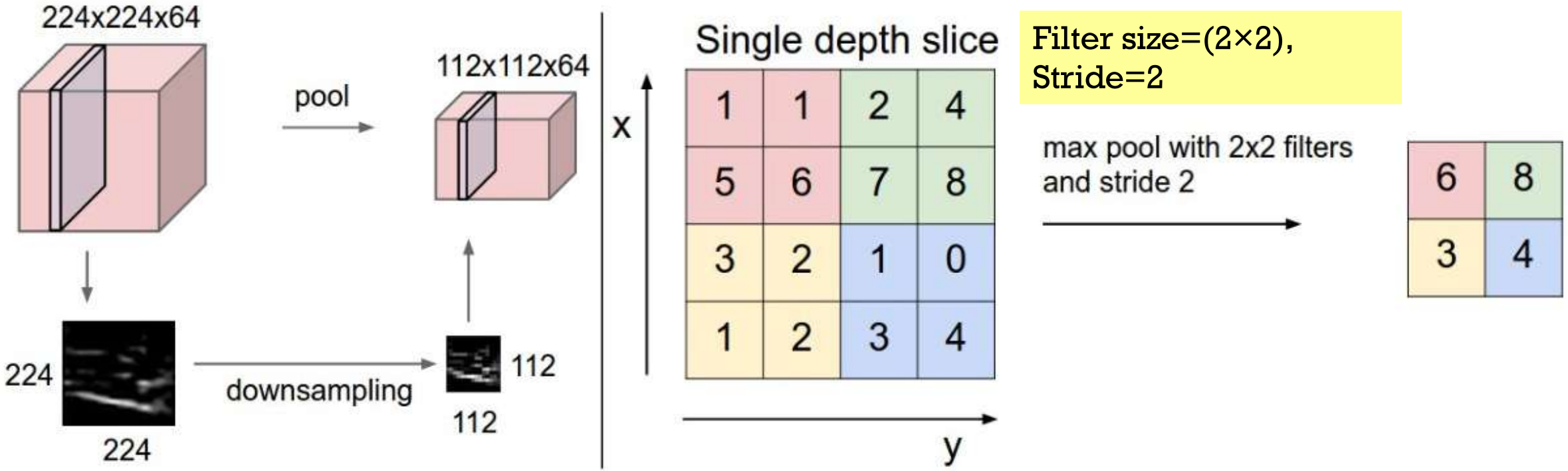


CNNs: 2) Pooling Layer

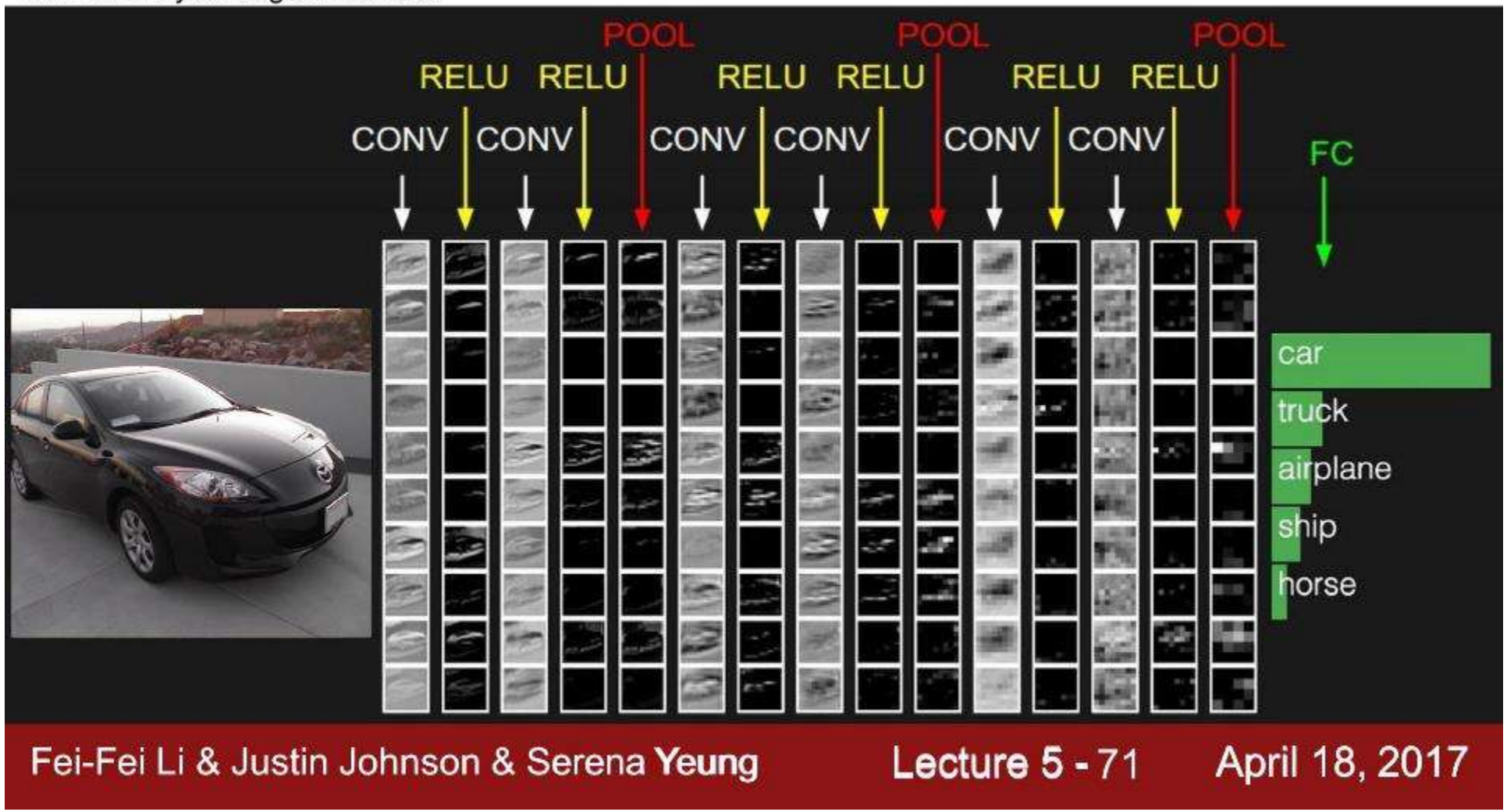
http://cs231n.github.io/convolutional-networks/

Summary: **2 parameters** for pooling layer
- Filter size, Stride

- Feature selection (reduction); downsampling

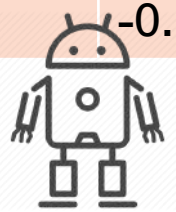


two more layers to go: POOL/FC



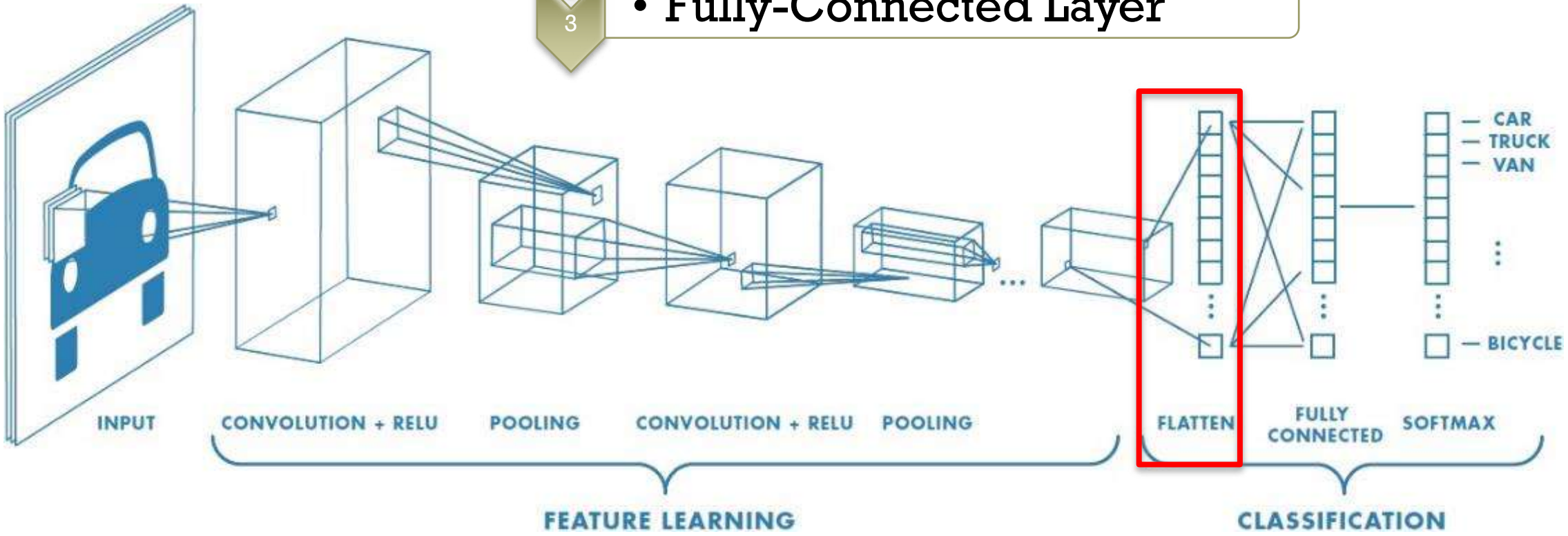
Convolutional Neural Networks

x1	x2	x3	x4	Corona
0.7	0.2	-0.5	-0.1	Yes



Embedding Vector

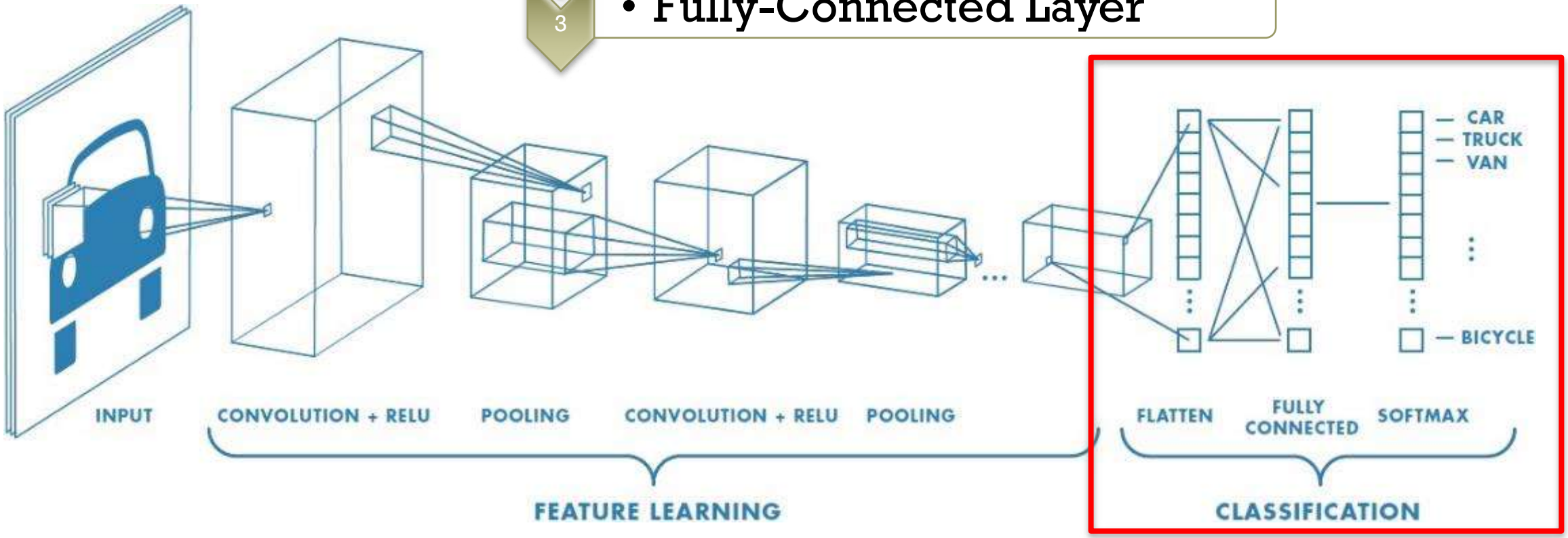
1. Convolutional Layer
2. Pooling Layer
3. Fully-Connected Layer



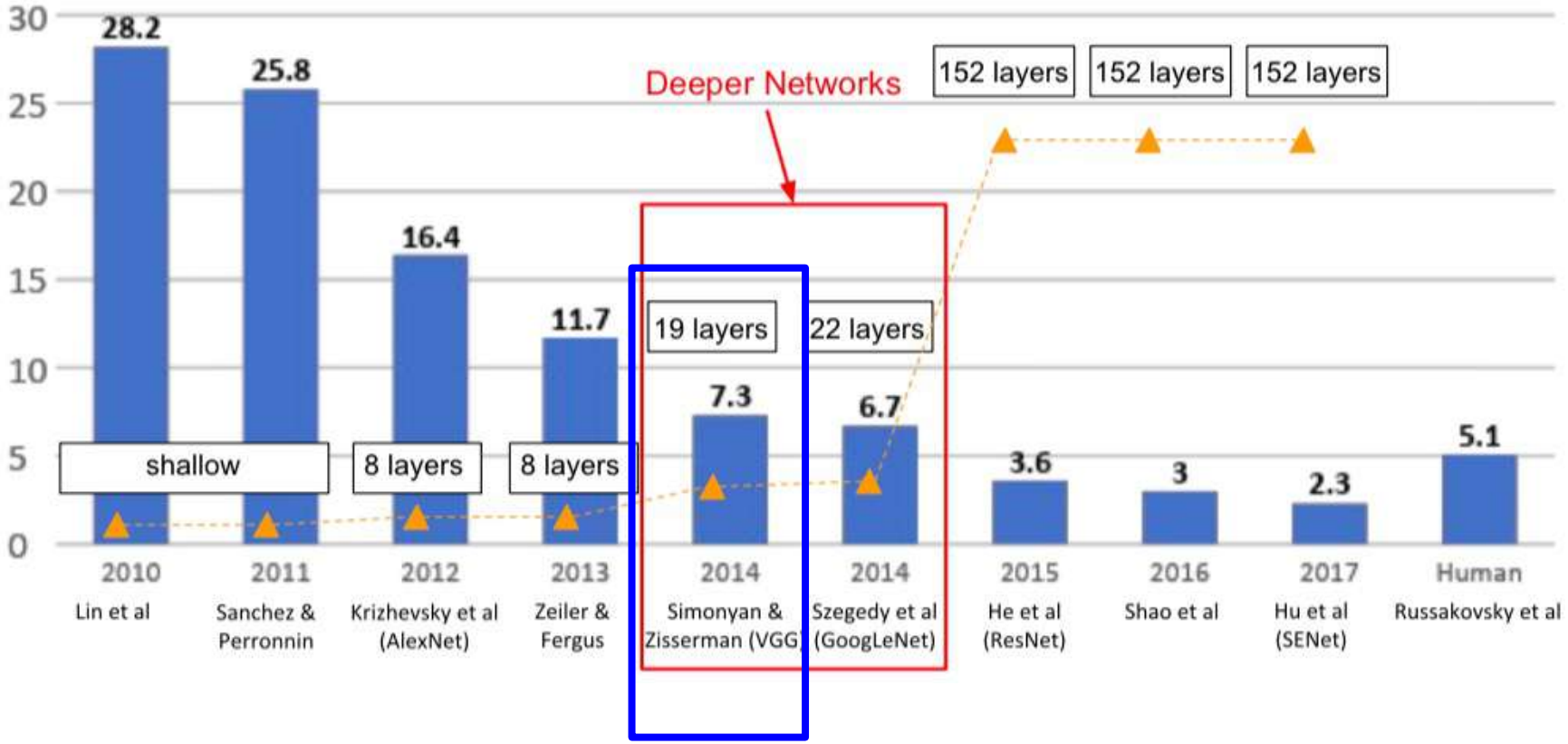
Convolutional Neural Networks (CNN)

3) Fully-Connected Layer (FC) = Neural Networks

1. Convolutional Layer
2. Pooling Layer
3. Fully-Connected Layer



ImageNet Large Scale Visual Recognition Challenge (ILSVRC 2nd runner-up in 2014): VGG19



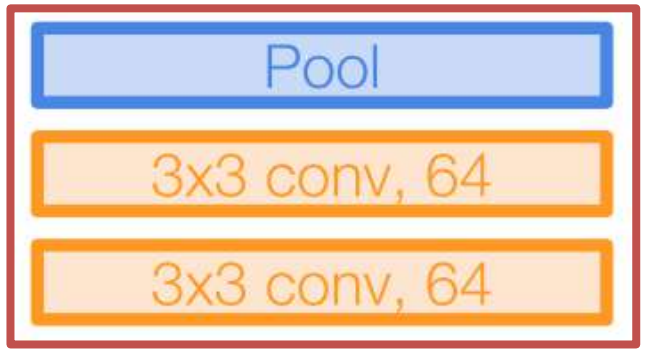
Case Study: VGGNet (2014)

[Simonyan and Zisserman, 2014]

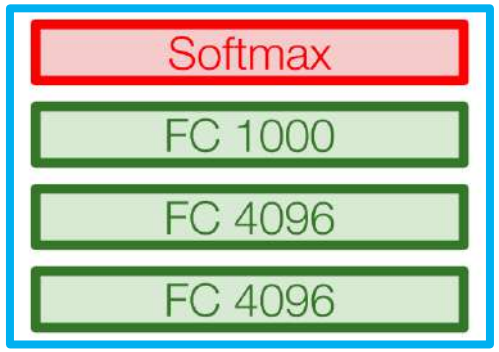
Small filters, Deeper networks

8 layers (AlexNet) → 16-19 layers (VGG16Net)

Only 3x3 CONV Stride 1, pad 1
And 2x2 MAX POOL stride 2

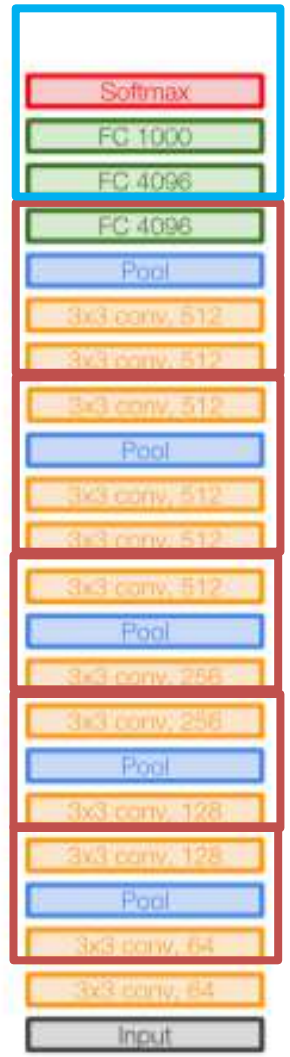


1) Feature extraction block

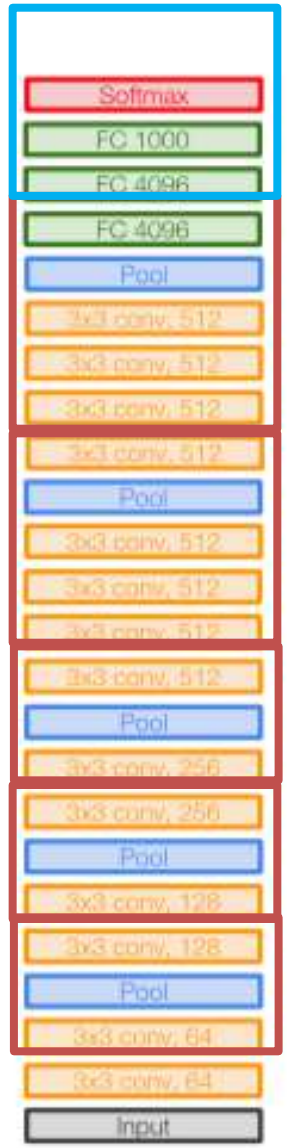


2) Classification block

AlexNet



VGG16



VGG19

Case Study: VGGNet (2014)

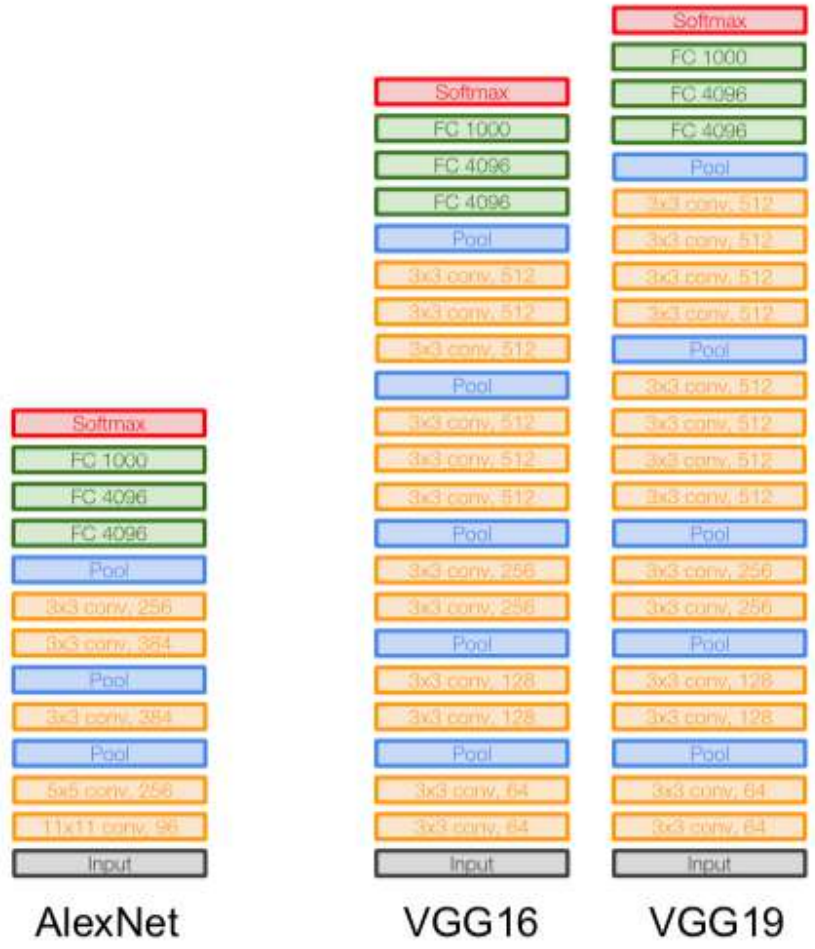
[Simonyan and Zisserman, 2014]

Q: Why use smaller filters? (3x3 conv)

Stack of three 3×3 conv (stride 1) layers has same **effective receptive field** as one 7×7 conv layer in ZFNet

But deeper, more non-linearities

And fewer parameters: $3 \times (3^2C^2)$ vs. 7^2C^2 for C channels per layer



Keras

```
#VGG Block
conv1 = Conv2D(64, (3,3), strides=(1,1), padding='same', activation='ReLU')(x)
conv2 = Conv2D(64, (3,3), strides=(1,1), padding='same', activation='ReLU')(conv1)
maxpool1 = MaxPooling2D((2,2))(conv2)
```

Padding='valid'
No padding

48

Variants of VGG

- **ConvNet configurations**
  - The depth of the configurations increases from the left (A) to the right (E), as more layers are added (the added layers are shown in bold)
- The convolutional layer parameters are denoted as "conv⟨receptive field size⟩-⟨number of channels⟩"

- ILSVRC'14 2nd in classification, 1st in localization
- Use VGG16 or VGG 19 (VGG19 only slightly better, more memory)

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

VGG16 VGG19

49

Calculate the number of parameters on VGG Net

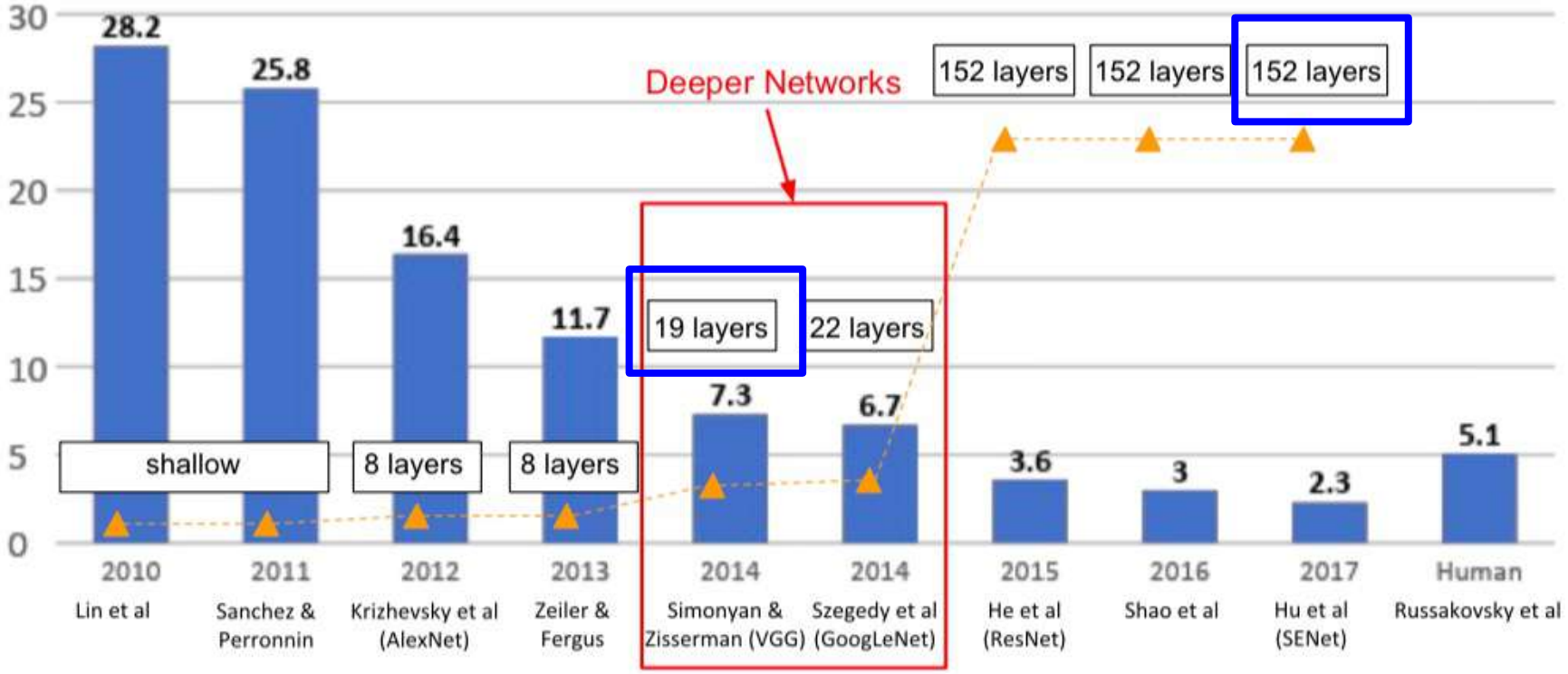
ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Network	A	B	C	D	E
Number of parameters (Millions)	133	133	134	138	144

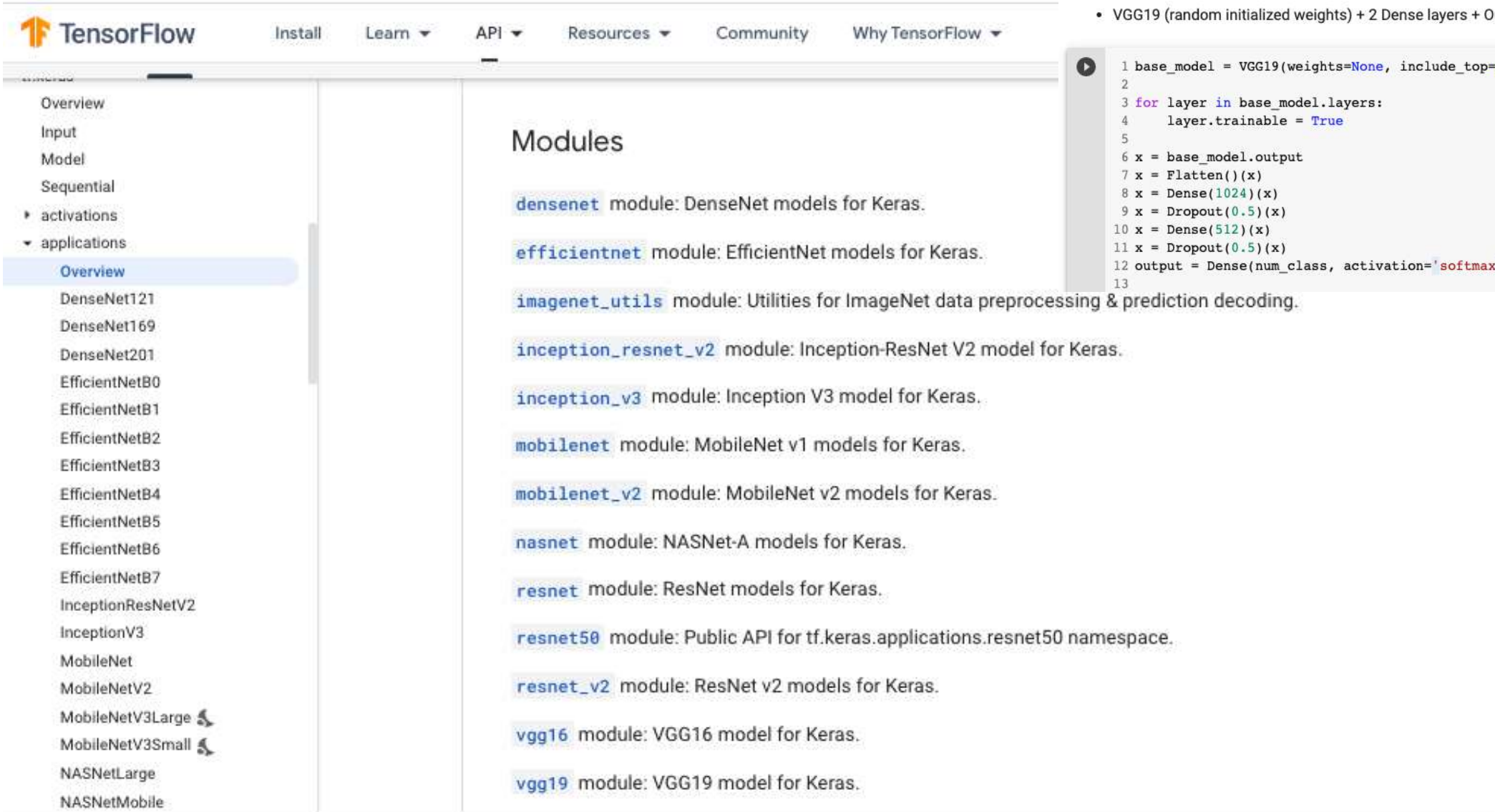
https://hoanglehaithanh.com/calculate-the-number-of-parameters-on-vgg-net/

50

ImageNet Large Scale Visual Recognition Challenge (ILSVRC 2nd runner-up in 2014): Deeper than VGG19



51



Model 1

- VGG19 (random initialized weights) + 2 Dense layers + Output layer

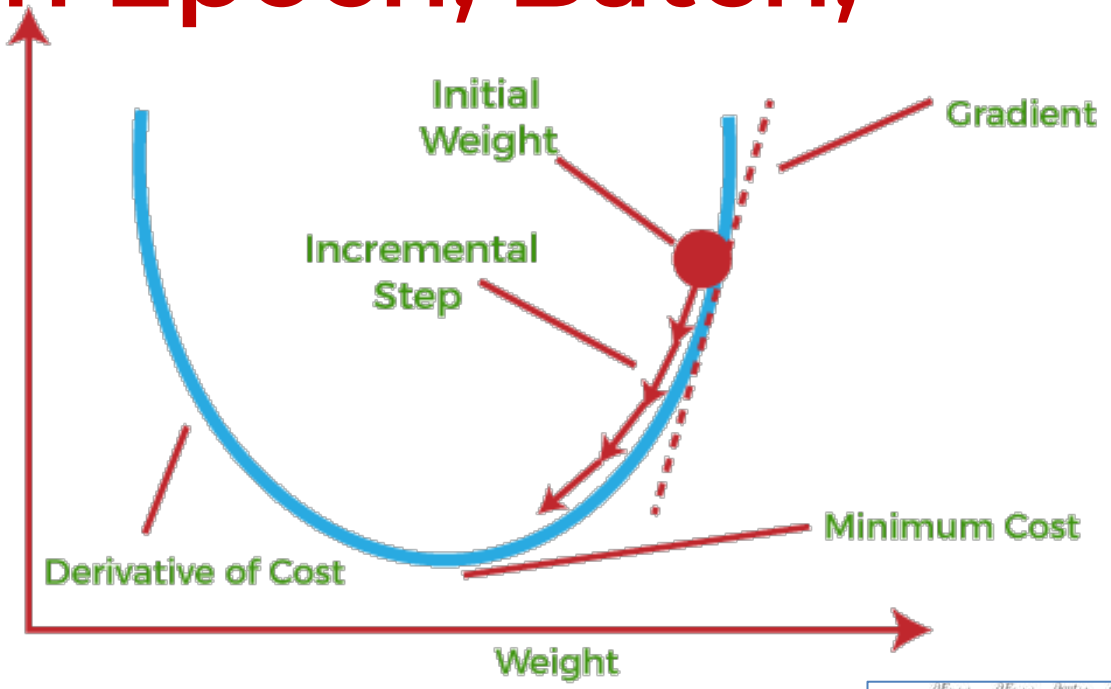
```
base_model = VGG19(weights=None, include_top=False, input_shape=(224, 224, 3))

for layer in base_model.layers:
    layer.trainable = True

x = base_model.output
x = Flatten()(x)
x = Dense(1024)(x)
x = Dropout(0.5)(x)
x = Dense(512)(x)
x = Dropout(0.5)(x)
output = Dense(num_class, activation='softmax')(x)
```

https://www.tensorflow.org/api_docs/python/tf/keras/applications

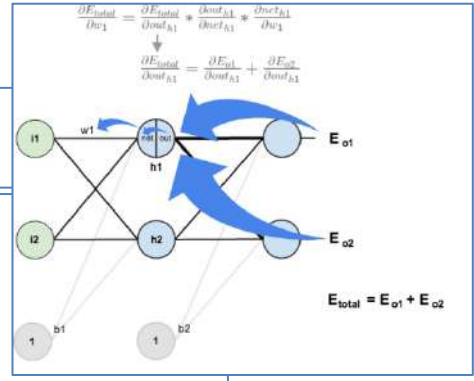
52

Differences Between Epoch, Batch, and Mini-batch

- The gradient descent algorithm works in two steps that are performed continuously for a specific number of iterations:
  - First, we compute the gradient, which is the first-order derivative of the objective function with respect to the variables.
  - Then, we update the variables in the opposite direction of the gradient
- Terms:
  - Epoch = use all training examples
  - Iteration = each round of updating weight



$$w_5^+ = w_5 - \eta * \frac{\partial E_{total}}{\partial w_5} = 0.4 - 0.5 * 0.082167041 = 0.35891648$$



Putting it all together:

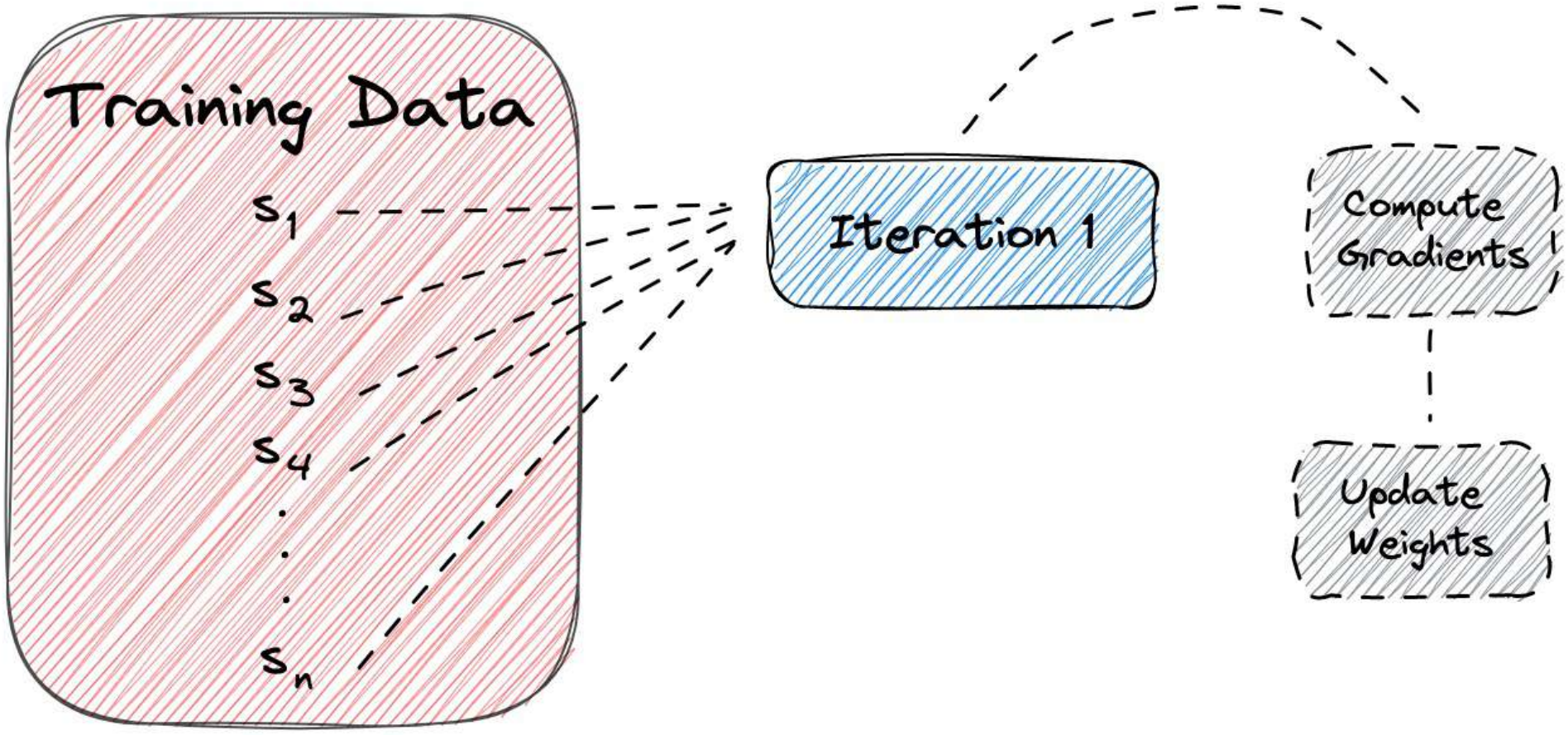
$$\frac{\partial E_{total}}{\partial w_5} = \frac{\partial E_{total}}{\partial out_{o1}} * \frac{\partial out_{o1}}{\partial net_{o1}} * \frac{\partial net_{o1}}{\partial w_5}$$

$$\frac{\partial E_{total}}{\partial w_5} = 0.74136507 * 0.186815602 * 0.593269992 = 0.082167041$$

https://www.baeldung.com/cs/epoch-vs-batch-vs-mini-batch#:~:text=So%2C%20a%20batch%20is%20equal,in%20mini%2Dbatch%20gradient%20descent.

1) Batch Gradient Descent

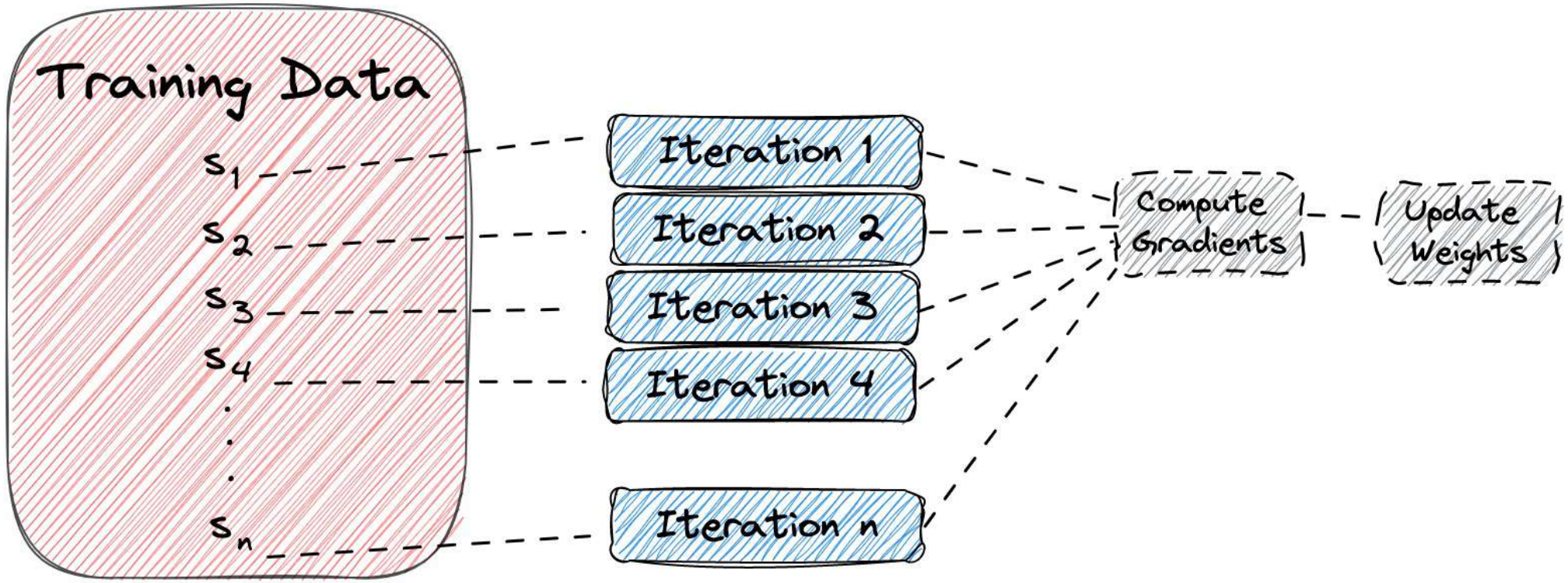
- In batch gradient descent, we use all our training data in a single iteration of the algorithm.



54

2) Stochastic Gradient Descent

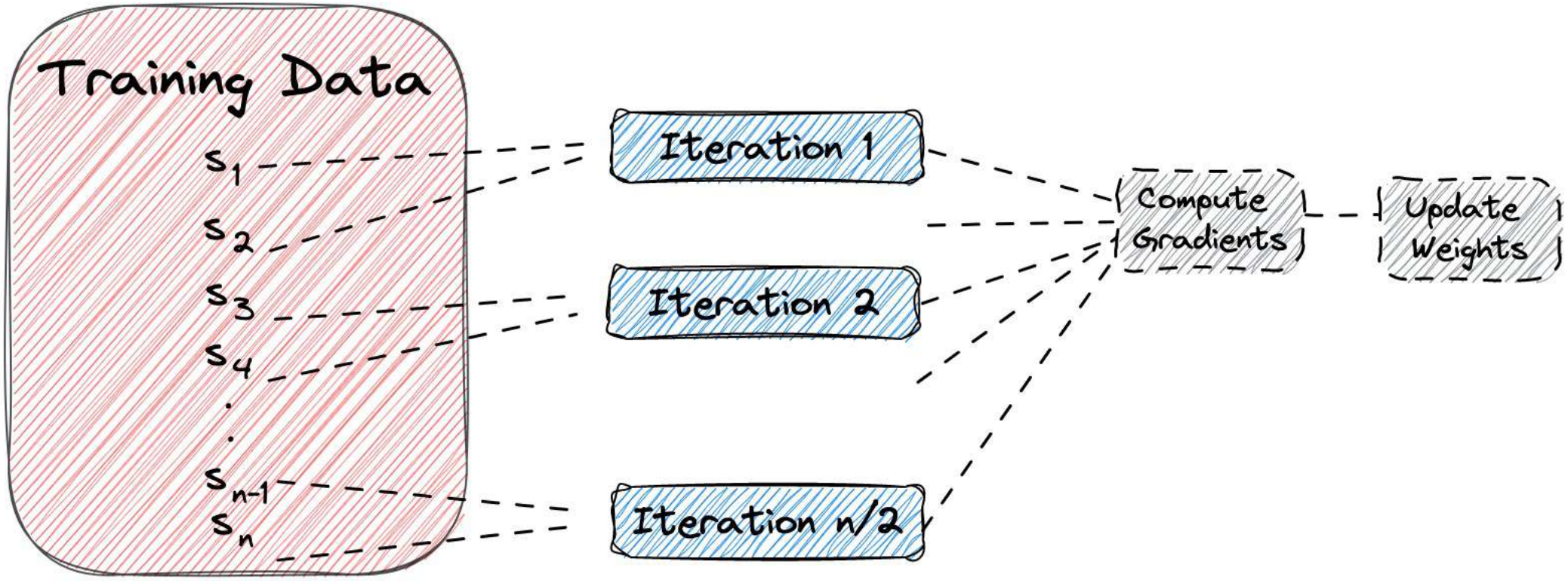
- The previous method can be very time-consuming and inefficient in case the size of the training dataset is large. To deal with this, we use stochastic gradient descent, where we use just one sample in a single iteration of the algorithm.



55

3) Mini-Batch Gradient Descent

- Mini-batch gradient descent is a combination of the previous methods where we use a group of samples called mini-batch in a single iteration of the training algorithm.
- The mini-batch is a fixed number of training examples that is less than the actual dataset. So, in each iteration, we train the network on a different group of samples until all samples of the dataset are used.



Conclusion about batch

An epoch means that we have passed each sample of the training set one time through the network to update the parameters. Generally, the number of epochs is a hyperparameter that defines the number of times that gradient descent will pass the entire dataset.

If we look at the previous methods, we can see that:

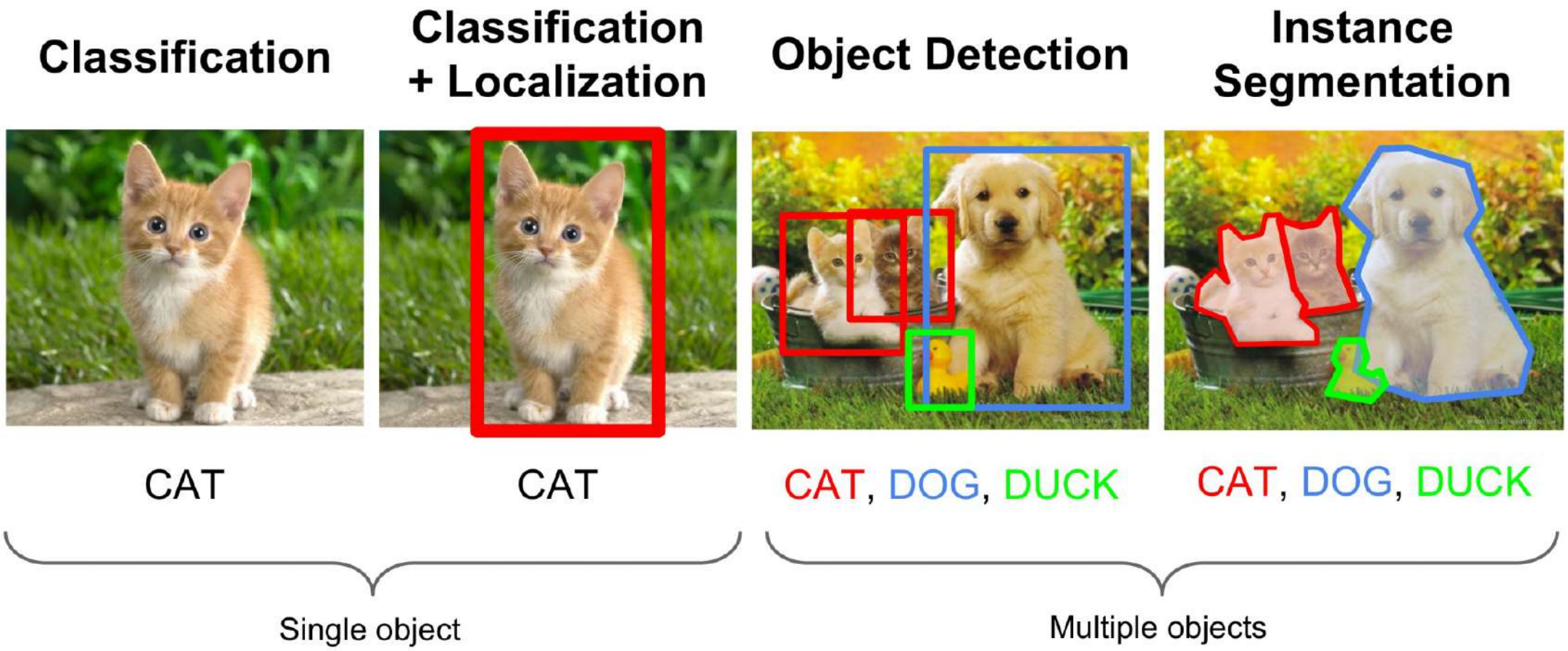
- In batch gradient descent, one epoch corresponds to a single iteration.
- In stochastic gradient descent, one epoch corresponds to n iterations where n is the number of training samples.
- In mini-batch gradient descent, one epoch corresponds to $\frac{n}{b}$ iterations where b is the size of the mini-batch.

Finally, let's present a simple example to better understand the three terms.

Let's assume that we have a dataset with $\mathbf{n = 2000}$ samples, and we want to train a deep learning model using gradient descent for $\mathbf{10}$ epochs and mini-batch size $\mathbf{b = 4}$:

- In batch gradient descent, we'll update the network's parameters (using all the data) 10 times which corresponds to 1 time for each epoch.
- In stochastic gradient descent, we'll update the network's parameters (using one sample each time) $2000 * 10 = 20000$ times which corresponds to 2000 times for each epoch.
- In mini-batch gradient descent, we'll update the network's parameters (using $b = 4$ samples each time) $\frac{2000}{4} * 10 = 5000$ times that corresponds to $\frac{2000}{4} = 500$ times for each epoch.

Image Classification Tasks

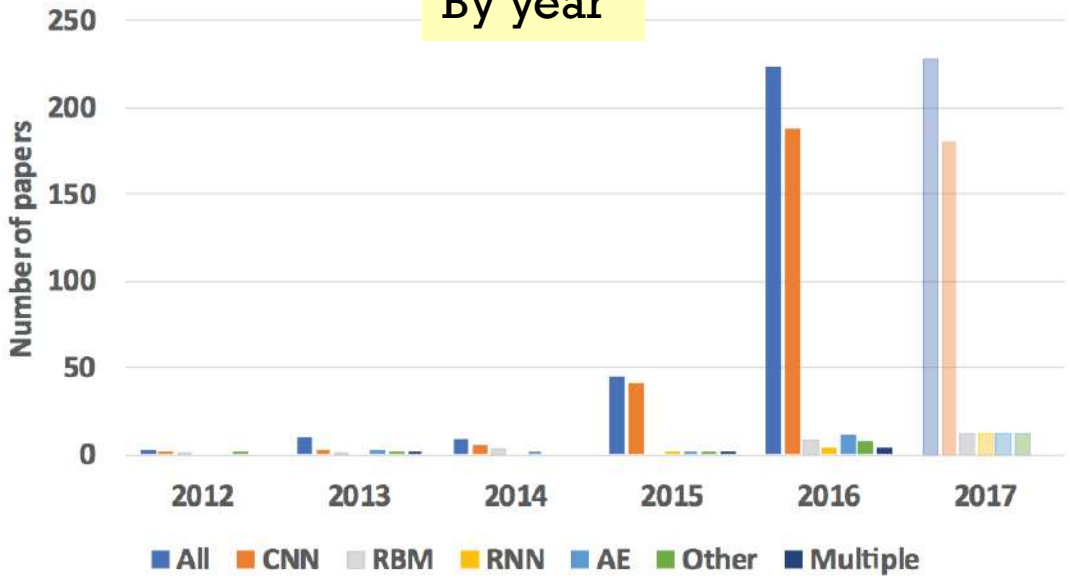


https://analyticsindiamag.com/top-5-image-classification-research-papers-every-data-scientist-should-know/

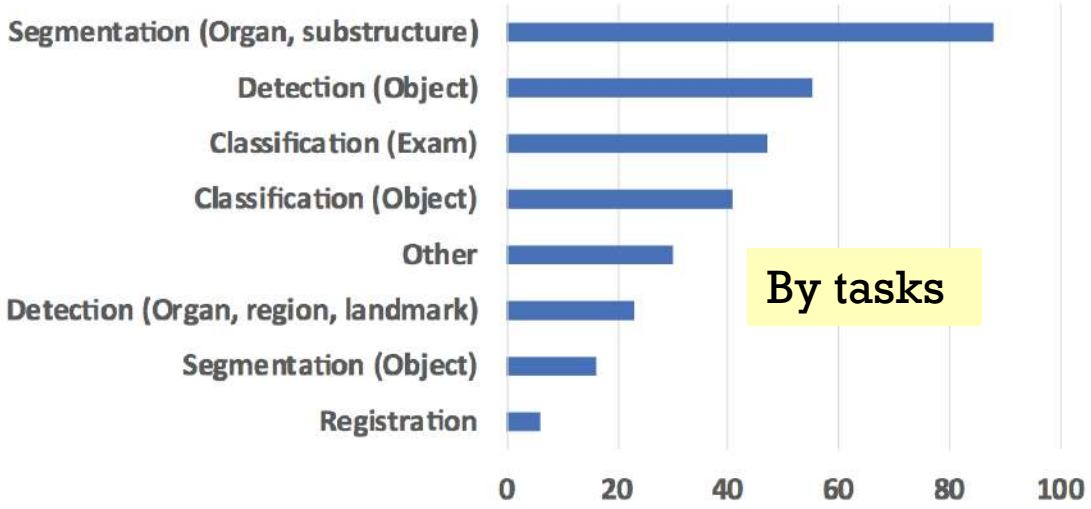
https://arxiv.org/pdf/1702.05747.pdf

Deep Learning in Medical Image Analysis [arXiv 2017]

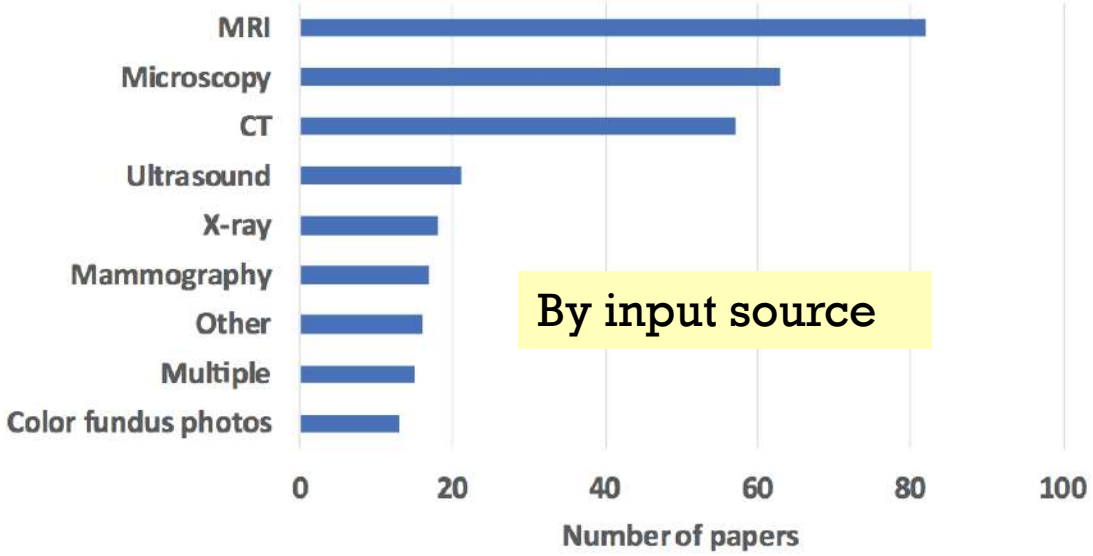
By year



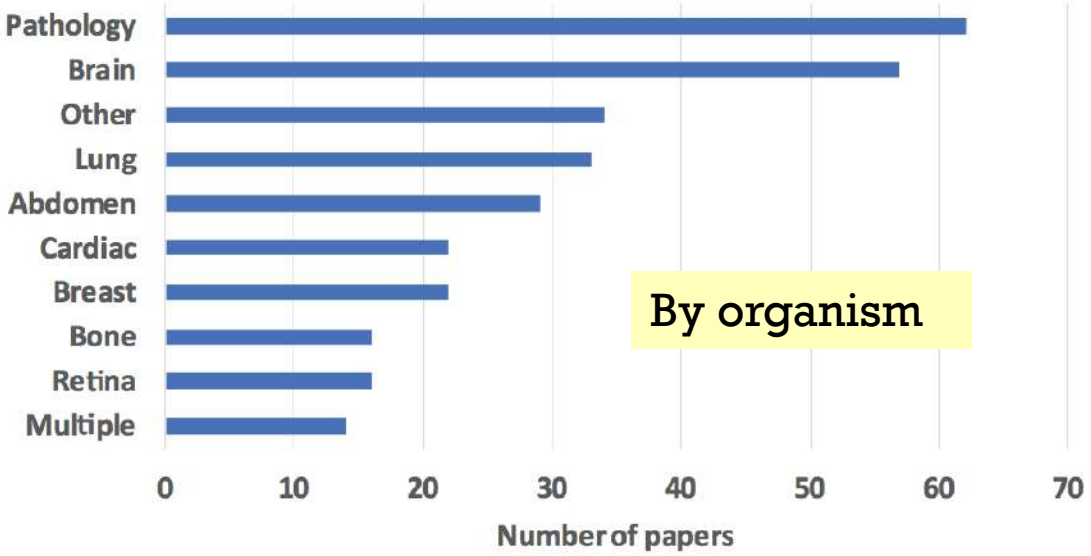
By tasks



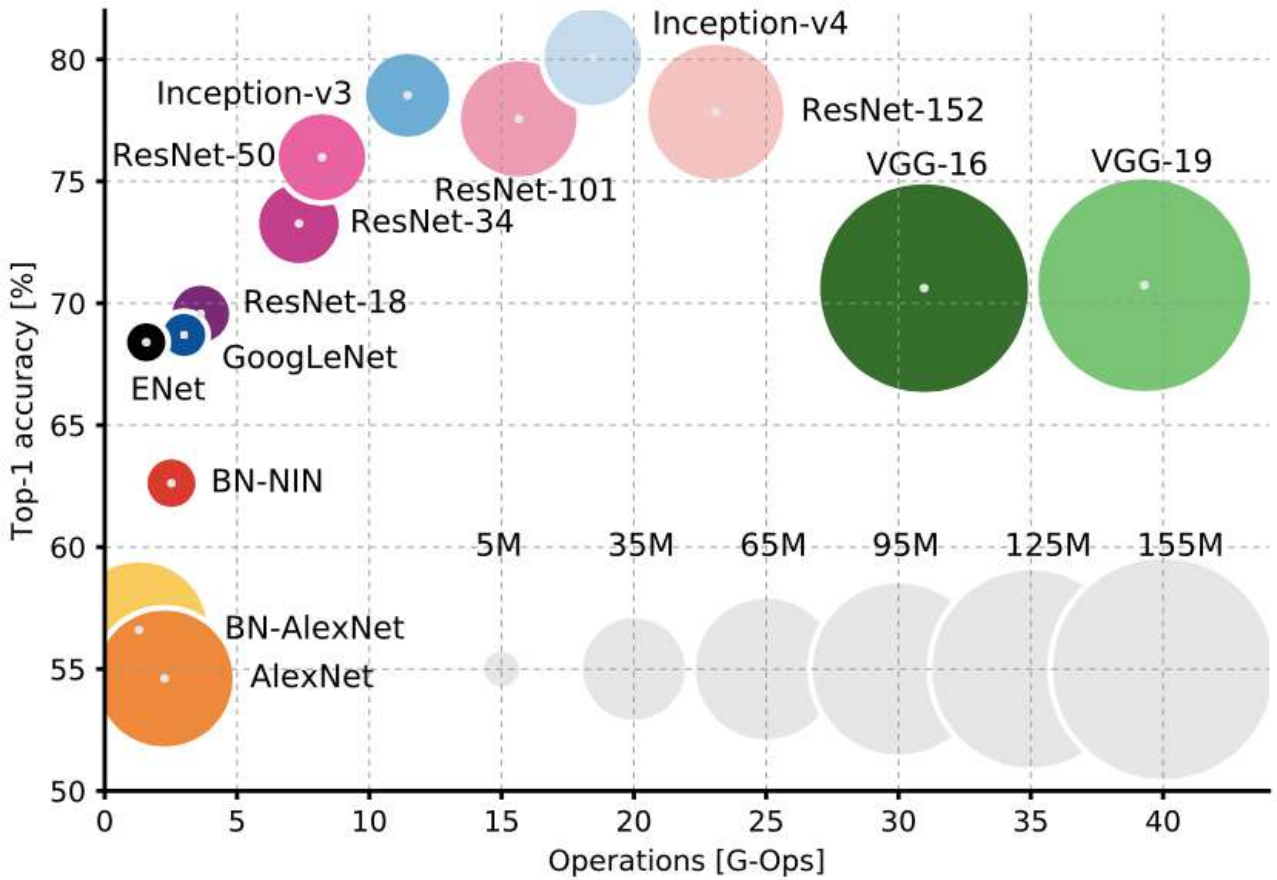
By input source



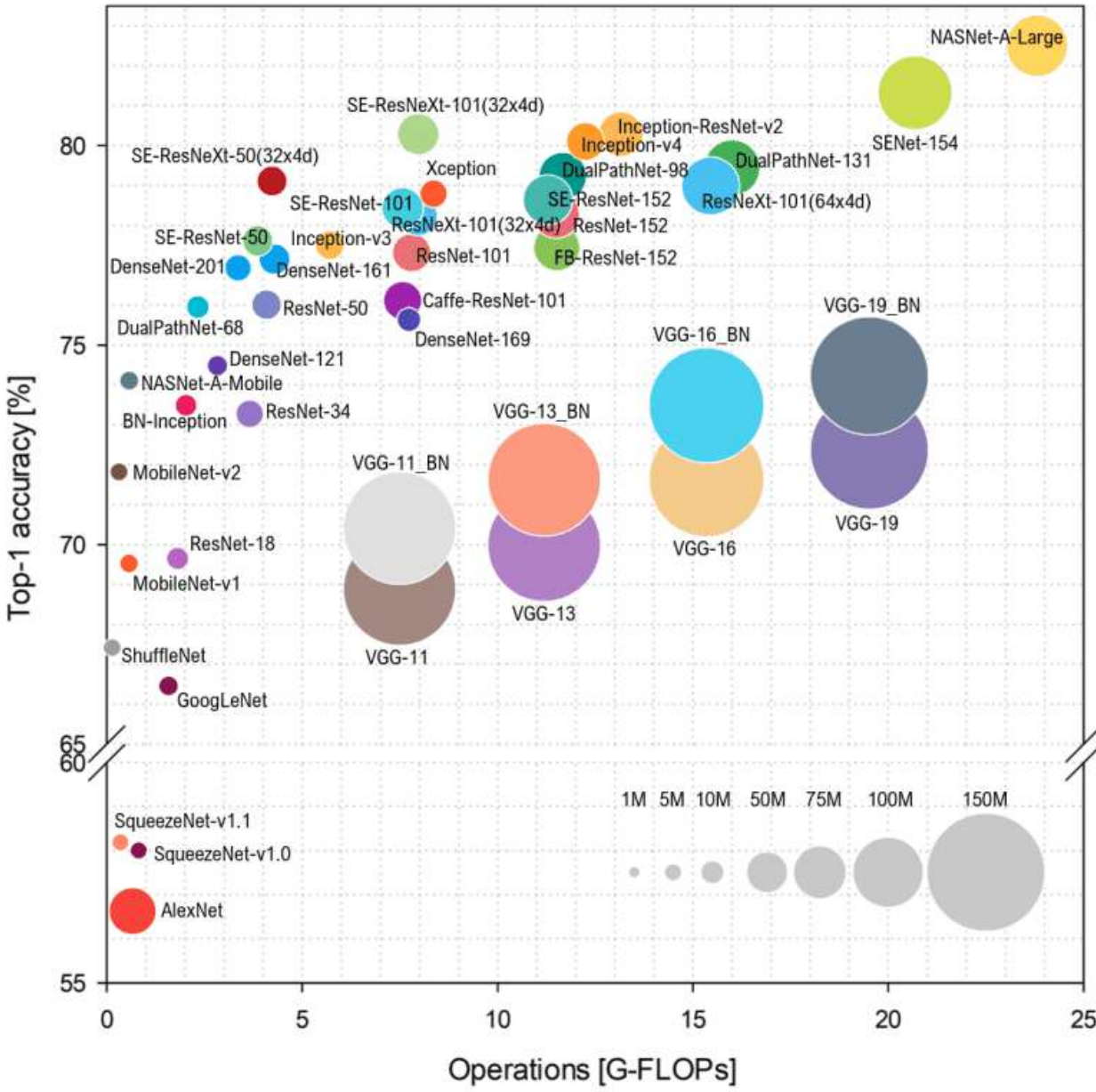
By organism



SOTA of Image Classification



https://blog.csdn.net/qq_34216467/article/details/83061692



https://theaisummer.com/cnn-architectures/

EfficientNet (May, 2019) → EffNetV2 (2021)

EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks

Mingxing Tan [1] Quoc V. Le [1]

Abstract

Convolutional Neural Networks (ConvNets) are commonly developed at a fixed resource budget, and then scaled up for better accuracy if more resources are available. In this paper, we systematically study model scaling and identify that carefully balancing network depth, width, and resolution can lead to better performance. Based on this observation, we propose a new scaling method that uniformly scales all dimensions of depth/width/resolution using a simple yet highly effective *compound coefficient*. We demonstrate the effectiveness of this method on scaling up MobileNets and ResNet.

To go even further, we use neural architecture

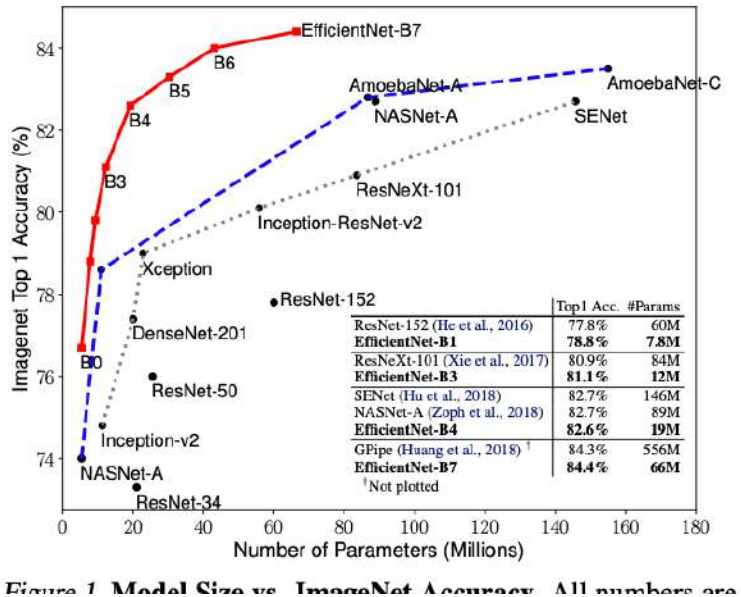
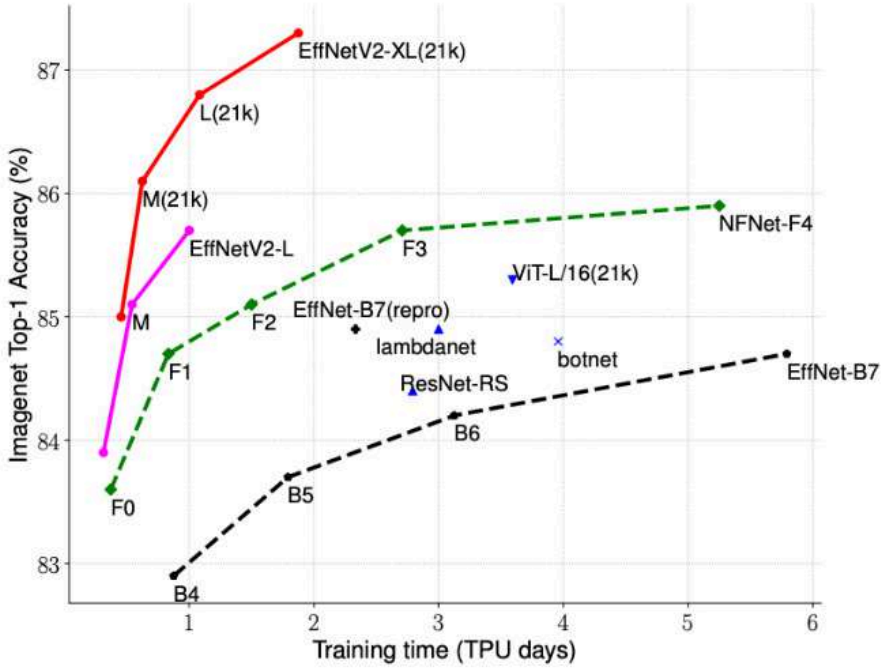


Figure 1. Model Size vs. ImageNet Accuracy. All numbers are



(a) Training efficiency.

	EfficientNet (2019)	ResNet-RS (2021)	DeiT/ViT (2021)	EfficientNetV2 (ours)
Top-1 Acc.	84.3%	84.0%	83.1%	83.9%
Parameters	43M	164M	86M	24M

(b) Parameter efficiency.

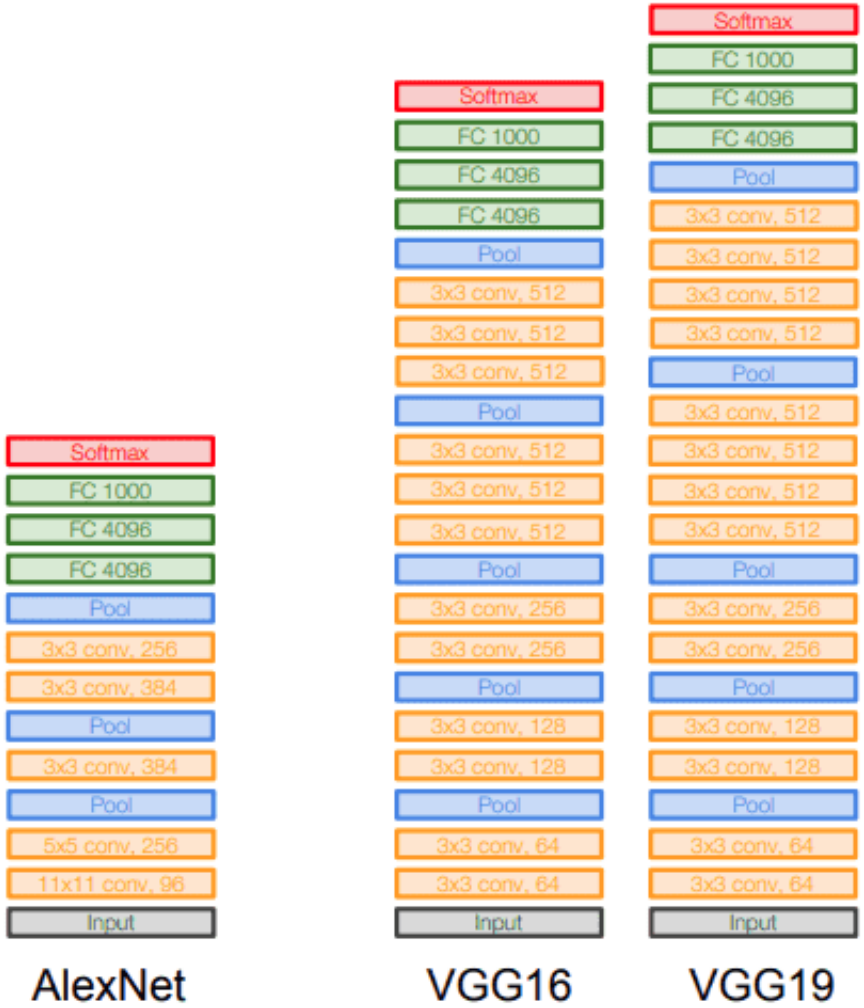
http://proceedings.mlr.press/v97/tan19a/tan19a.pdf

https://twitter.com/TensorFlow/status/1423359562724175873/photo/1

Network series

1) VGG (2014)

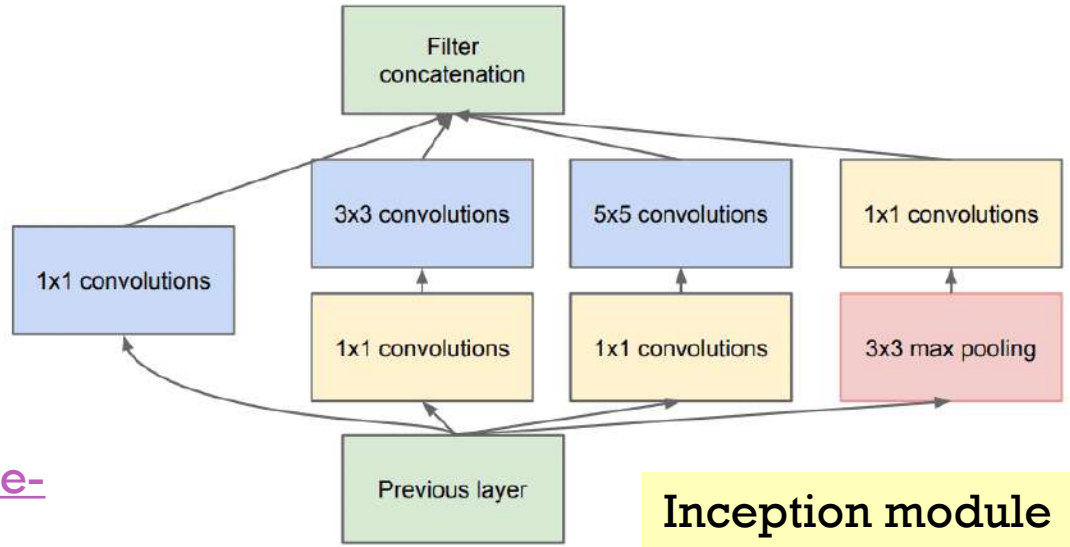
1. VGG (2014)
2. Inception/GoogleNet (2014)
3. Inception V2, V3
4. ResNet
5. DenseNet
6. Inception-V4 & Inception-ResNet
7. EfficientNet V1, V2



+ 2) Inception V1/GoogLeNet (2014)

- After VGG, the paper "Going Deeper with Convolutions" [3] by Christian Szegedy et al. was a huge breakthrough. What about increasing both the depth (more layers) and width (more feature maps) of the network while keeping computations to a constant level?
- The answer is with 1×1 convolutions! The main purpose is dimension reduction, by reducing the output channels of each convolution block.
  - 1×1 convolutions are used to compute reductions before the computationally expensive convolutions (3×3 and 5×5).
- Moreover, it uses convolutions of different kernel sizes (5×5, 3×3, 1×1) to capture details at multiple scales.
- The InceptionNet/GoogLeNet architecture consists of 9 inception modules stacked together, with max-pooling layers between (to halve the spatial dimensions). It consists of 22 layers (27 with the pooling layers). It uses global average pooling after the last inception module.

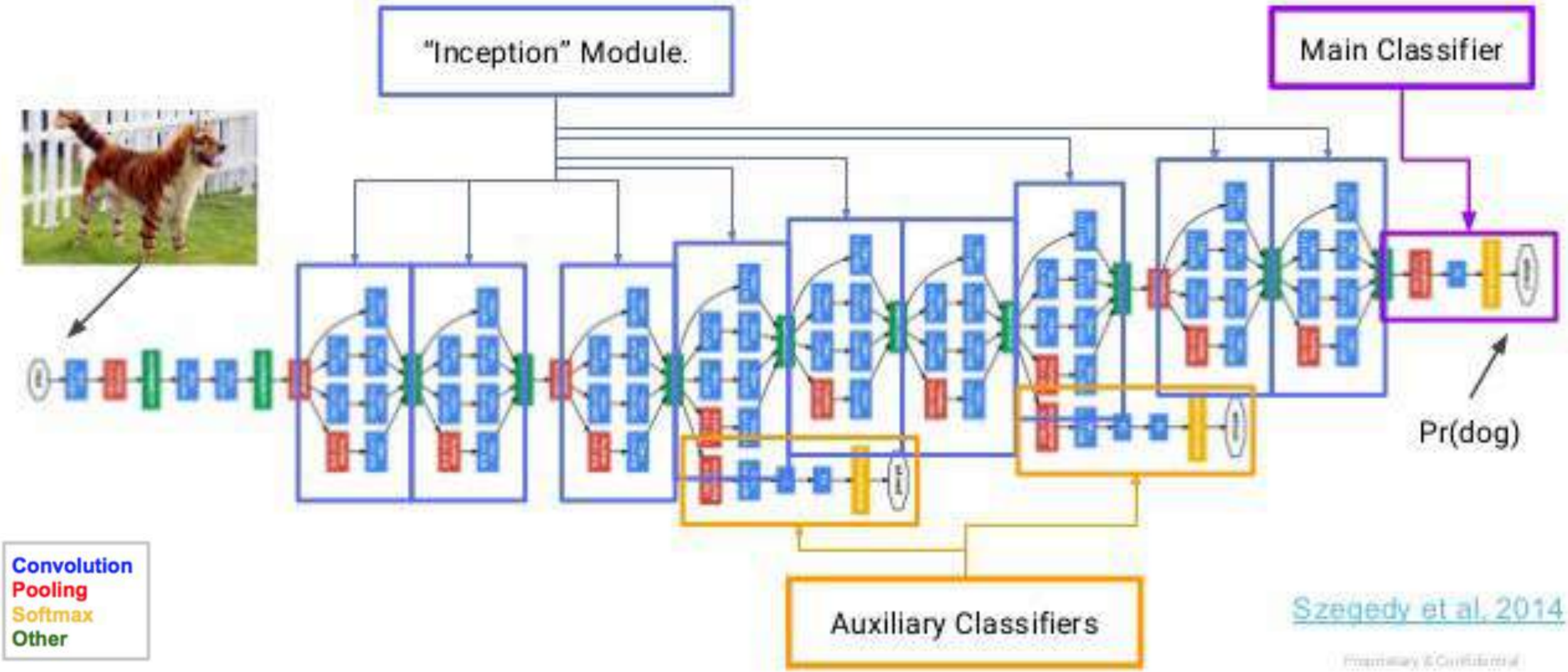
https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202



Inception module

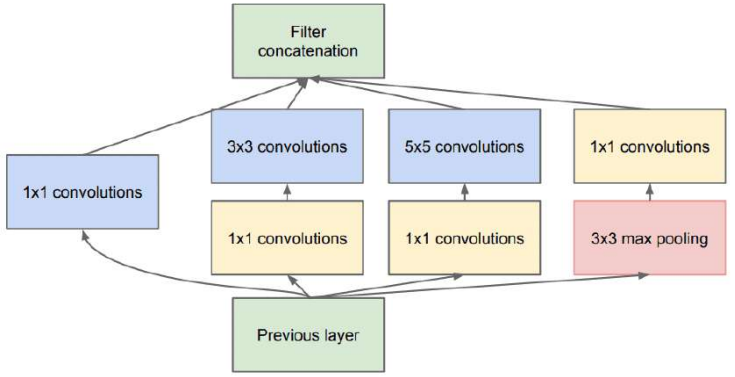
GoogLeNet (aka "Inception") Architecture

Inception is a deep network, to prevent the **middle part** of the network from "**dying out**"(vanishing gradient problem), the authors introduced **two auxiliary classifiers.** Softmax is applied in each of them and then **Auxilary loss** is calculated on the same labels of the output classifier.

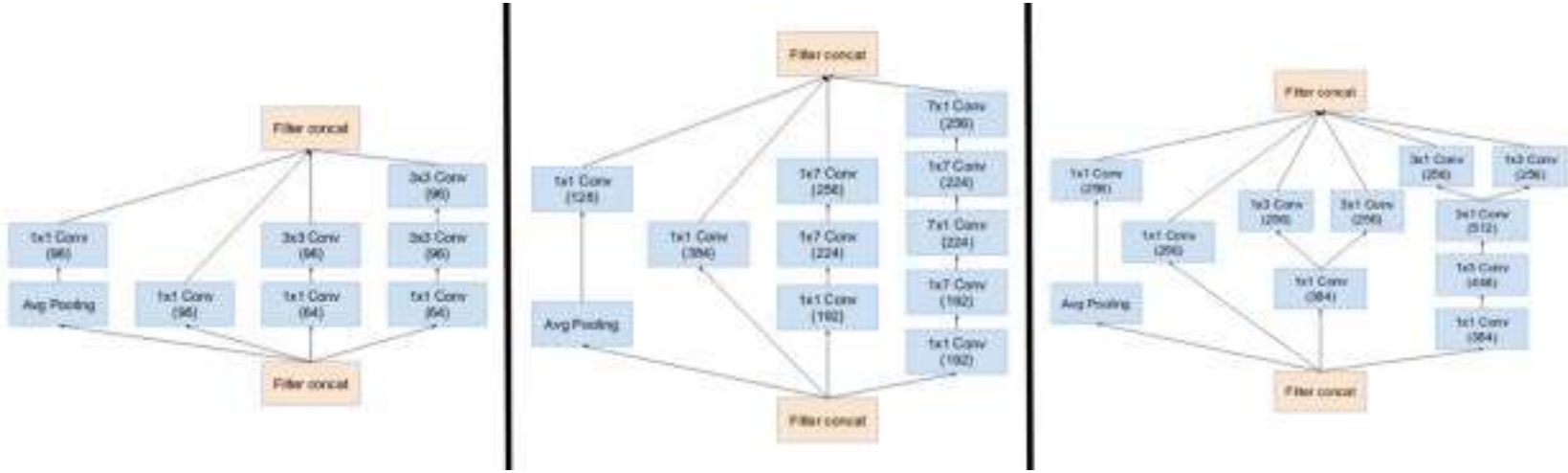


3) Inception V2, V3 (2015) – improve speed

- Factorize 5x5 and 7x7 (in InceptionV3) convolutions to two and three 3x3 sequential convolutions respectively. This improves computational speed. This is the same principle as VGG. (analogy: 5*5 = 25 parameters vs. 2*(3*3) = 18 parameters)
- Simply, a 3x3 kernel is decomposed into two smaller ones: a 1x3 and a 3x1 kernel, which are applied sequentially. (analogy: 3*3 = 9 parameters vs. (1*3) + (3*1) = 6 parameters)
- The inception modules became wider (more feature maps).
- They added batch normalization.



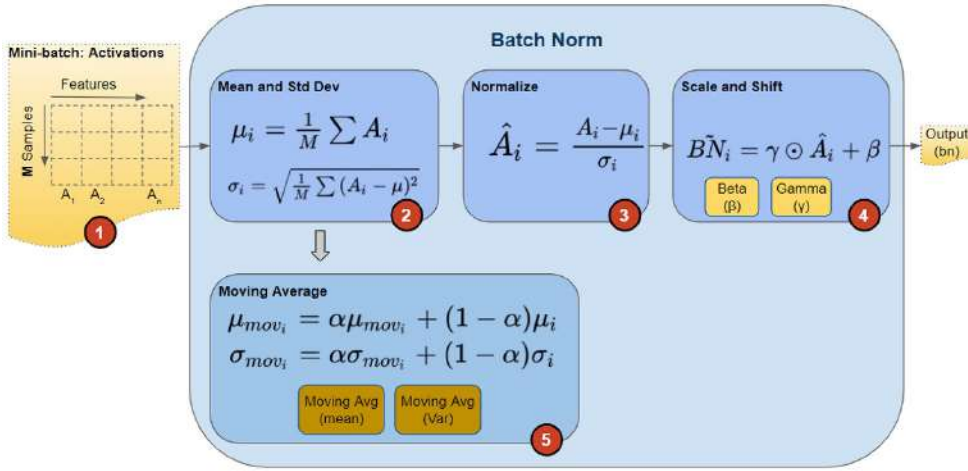
Inception-V1 module



Inception Module (A, B, C) in Inception V2, V3

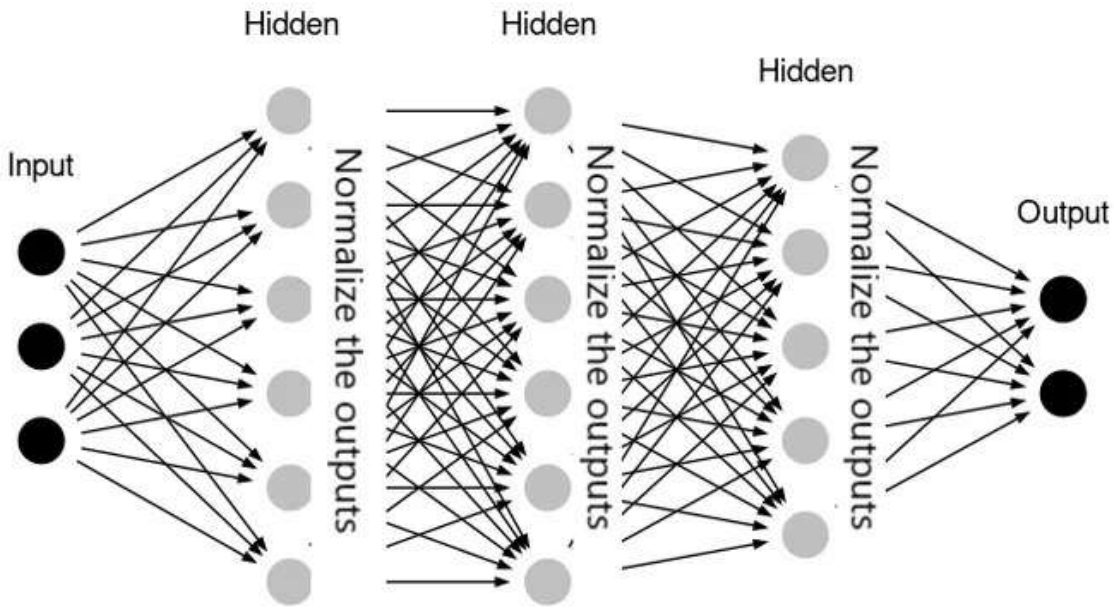
Batch normalization

$$\hat{x} = \frac{x - mean(\bar{x})}{std(\bar{x}))}$$



https://towardsdatascience.com/batch-norm-explained-visually-how-it-works-and-why-neural-networks-need-it-b18919692739

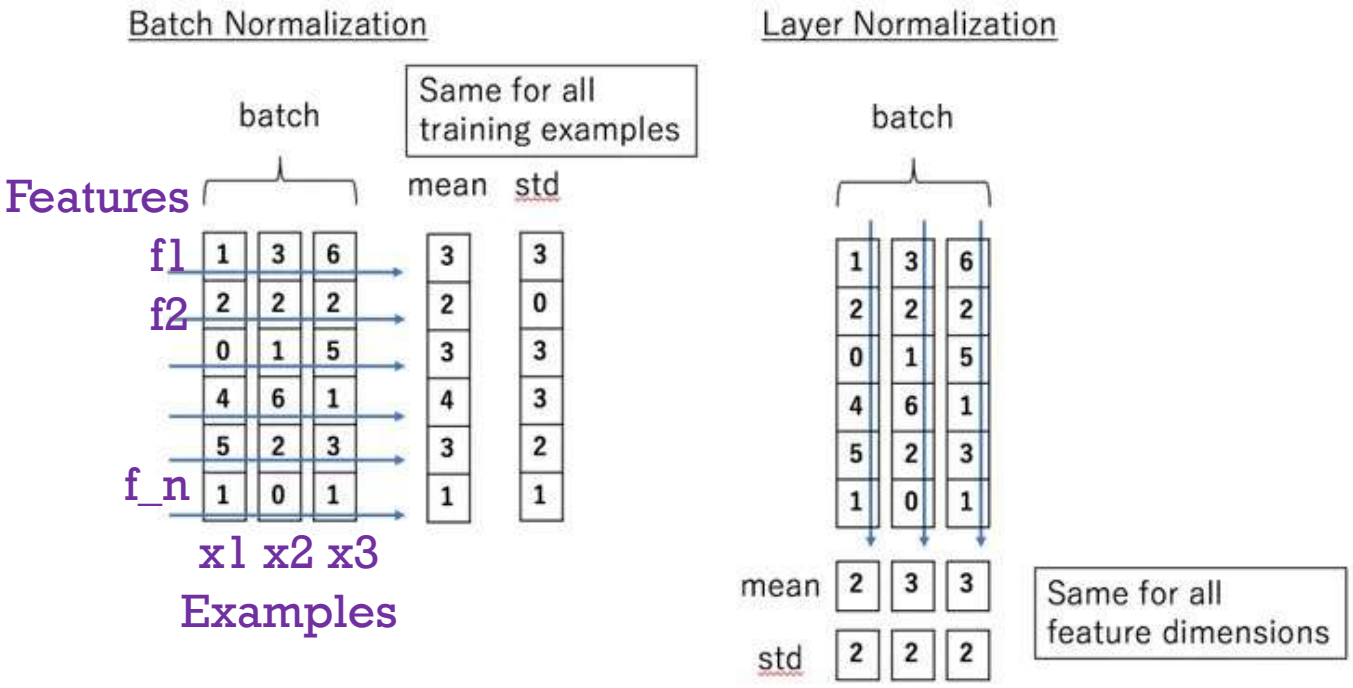
- Batch Normalization focuses on standardizing the inputs to **any particular layer** (i.e., activations from previous layers).
- **Advantages of Batch Normalization Layer**
- Batch normalization improves the training time and accuracy of the neural network.
- It decreases the effect of weight initialization.
- It also adds a regularization effect on the network.
- It works better with the fully Connected Neural Network (FCN) and Convolutional Neural Network.
- **Disadvantages of Batch Normalization Layer**
- Batch normalization is dependent on mini-batch size which means if the mini-batch size is small, it will have little to no effect
- Batch normalization does not work well with Recurrent Neural Networks (RNN)

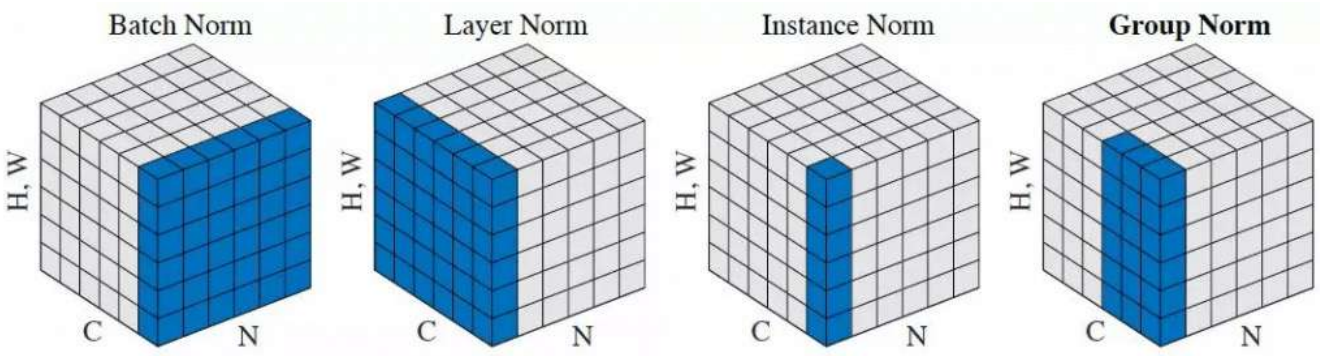


https://machinelearningknowledge.ai/keras-normalization-layers-explained-for-beginners-batch-normalization-vs-layer-normalization/#Layer_Normalization_Layer

Layer normalization

- Layer Normalization which addresses the drawbacks of batch normalization. This technique is not dependent on batches and the normalization is applied on the neuron for a single instance across all features.
- **Advantages of Layer Normalization**
- It is not dependent on any batch sizes during training.
- It works better with Recurrent Neural Network.
- **Disadvantages of Layer Normalization**
- It may not produce good results with Convolutional Neural Networks (CNN)



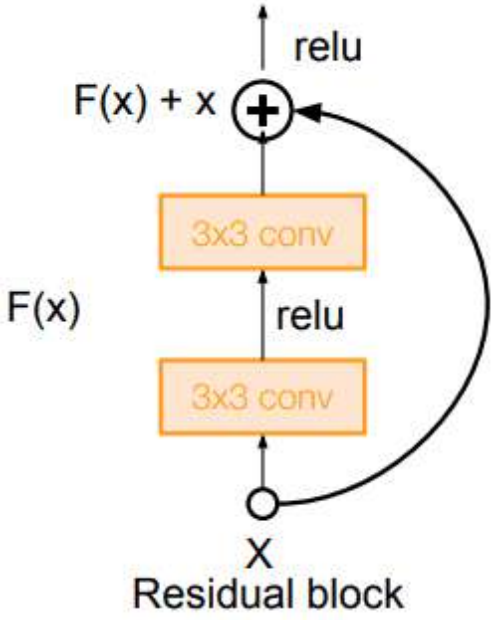


C = features (channels)
N = each mini-batch

https://machinelearningknowledge.ai/keras-normalization-layers-explained-for-beginners-batch-normalization-vs-layer-normalization/#Layer_Normalization_Layer

4) ResNet: Deep Residual Learning for Image Recognition (2015)

- All the predescribed issues such as vanishing gradients were addressed with two tricks:
  - batch normalization and
  - short skip connections - Instead of H(x) = F(x), we ask them model to learn the difference (residual) H'(x) = F(x) + x, which means **H(x) - x = F(x)** will be the residual part [4].
- Designed deeper architectures ranging from 18 (Resnet-18) to 150 (Resnet-150) layers.
- For the deepest models they adopted **1x1 convs**, as illustrated on the right:



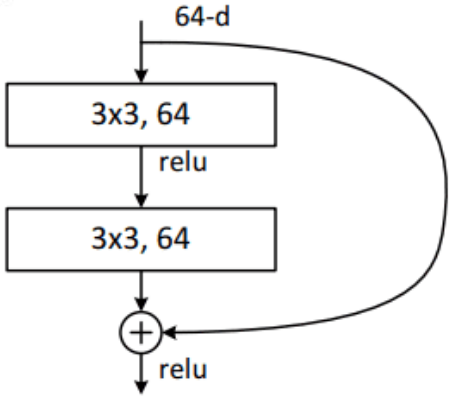
the layer followed by adding a bias term.

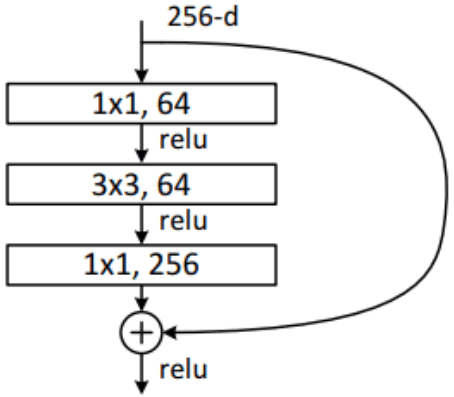
Then comes the activation function, f() and we get the output as H(x).

H(x)=f(wx + b) or H(x)=f(x)

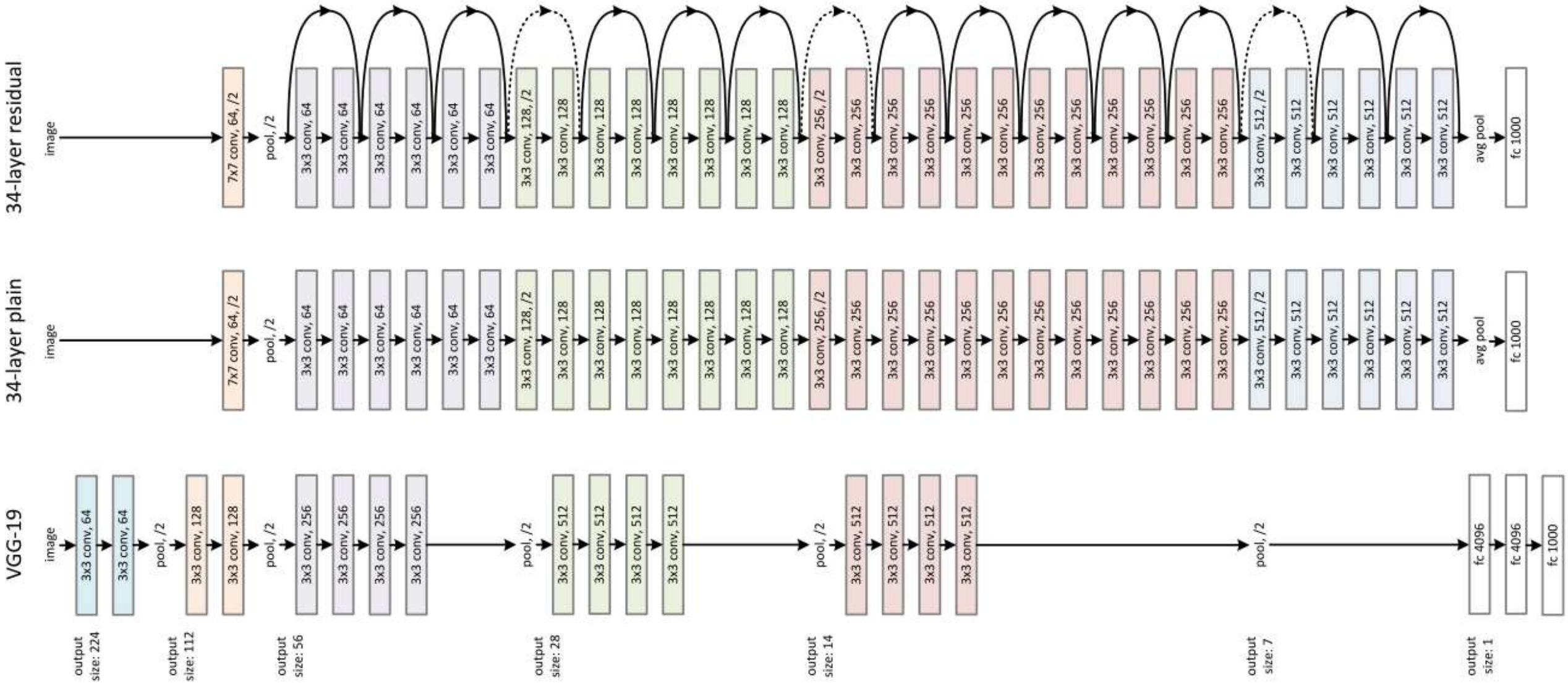
Now with the introduction of a new skip connection technique, the output is H(x) is changed to

H(x)=f(x)+x

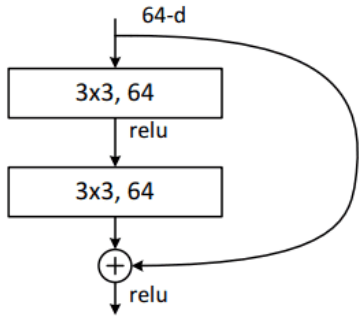


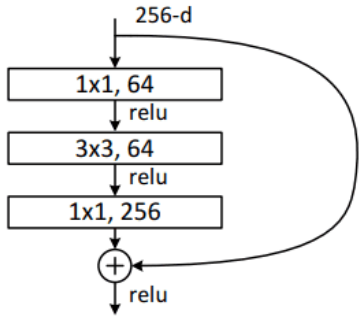


ResNet-34 vs. Plain Network (no skip connection)



ResNet (cont.)
ResNet-18, 34, 50, 101, 152



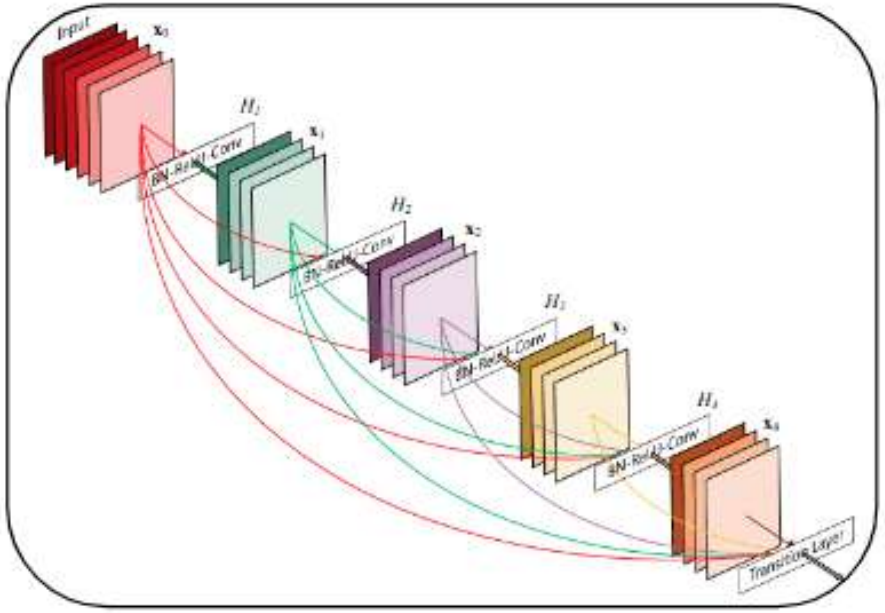


layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

5) DenseNet: Densely Connected Convolutional Networks (2017)

- Skip connections are a pretty cool idea. Why don't we just skip-connect everything?

Dense Block 1

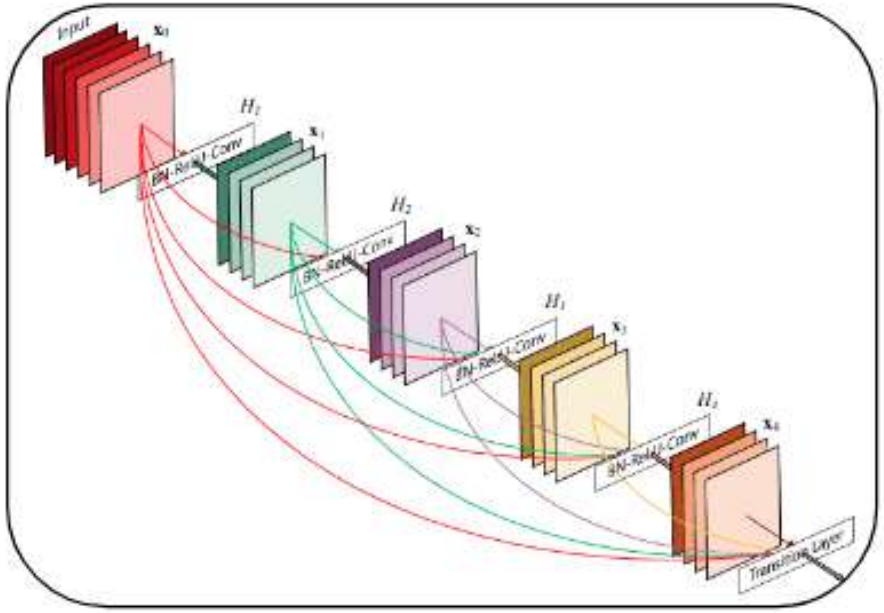


Spatial dims: 256x256
512 feature maps

Transition Layer

1×1×K convs
+
2×2 average pooling

Dense Block 2



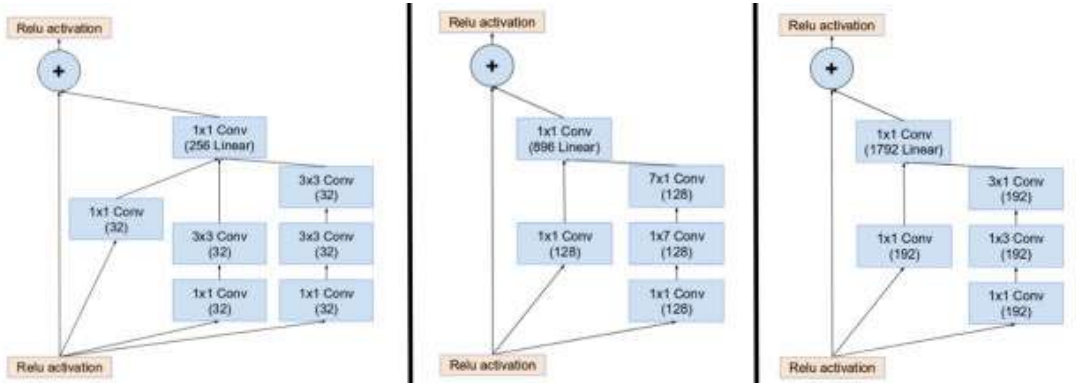
Spatial dims: 128x128
K feature maps < 512

6) Inception-V4 & Inception-ResNet

- Inception v4 and Inception-ResNet were introduced in the same paper.

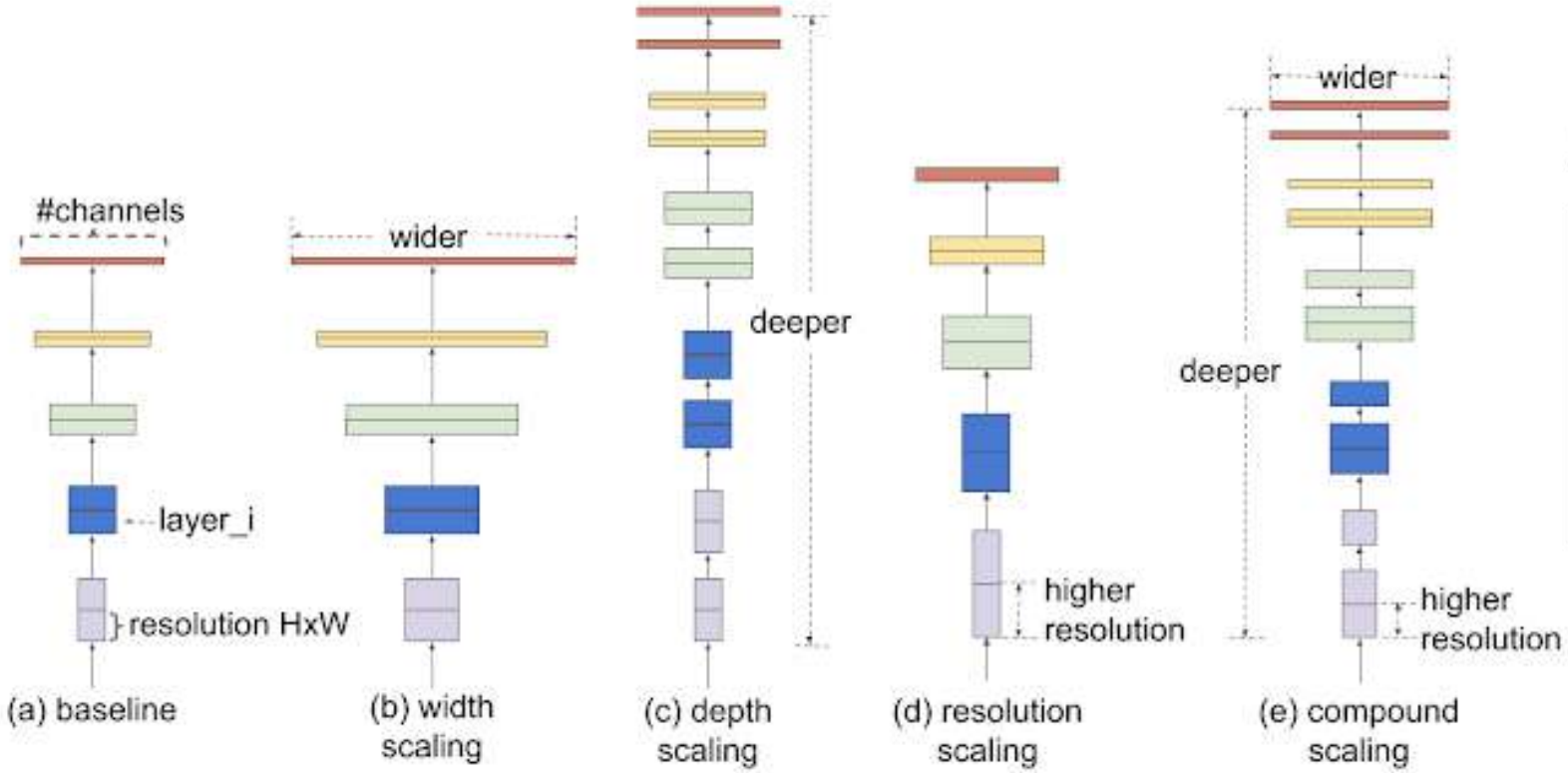
- Inception-V4 (simplified previous version)
  - Make the modules more uniform. The authors also noticed that some of the modules were more complicated than necessary. Thus; the "stem" of Inception v4 was modified (simplified).
  - Inception v4 introduced specialized "Reduction Blocks" (simplified) which are used to change the width and height of the grid.

- Inception-ResNet V1, V2
  - For residual addition to work, the input and output after convolution must have the same dimensions. Hence, we use `1x1` convolutions after the original convolutions, to match the depth sizes (Depth is increased after convolution).
  - Inception-ResNet v1 has a computational cost that is similar to that of Inception v3.
  - Inception-ResNet v2 has a computational cost that is similar to that of Inception v4.



7) EfficientNet (May, 2019)

- So, let's instead scale up network depth (more layers), width (more feature maps), resolution (various sizes of filters) simultaneously. This is known as compound scaling.



For B0 to B7 base models, the input shapes are different. Here is a list of input shape expected for each model:

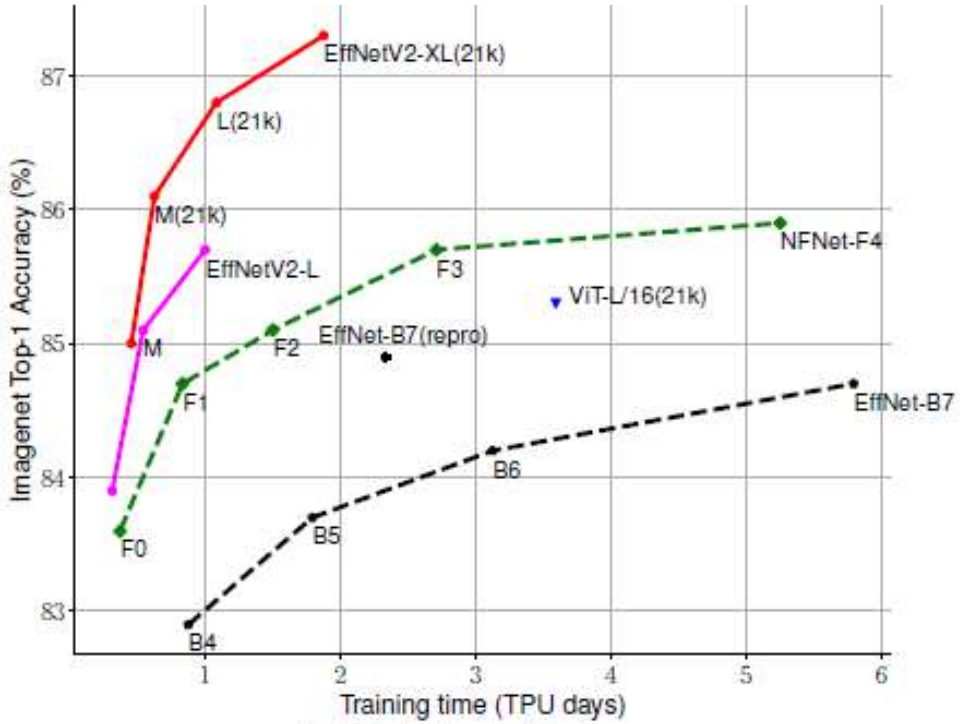
Base model	resolution
EfficientNetB0	224
EfficientNetB1	240
EfficientNetB2	260
EfficientNetB3	300
EfficientNetB4	380
EfficientNetB5	456
EfficientNetB6	528
EfficientNetB7	600

https://blog.roboflow.com/how-to-train-efficientnet/

7) EfficientNetV2 (2021)

Smaller models & faster training

- Our experiments show that EfficientNetV2 models train much faster than state-of-the-art models while being up to 6.8x smaller.
- There are many versions: small, medium, large, and extra-large (XL).

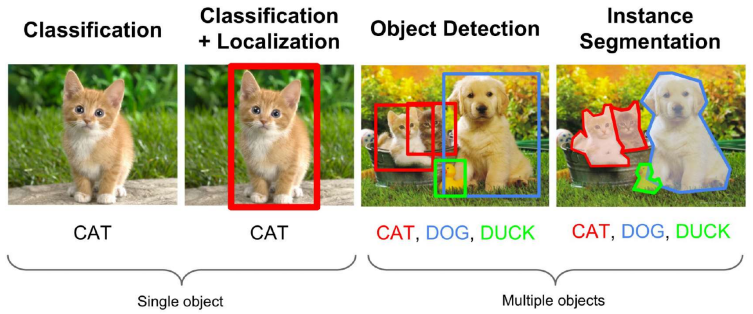


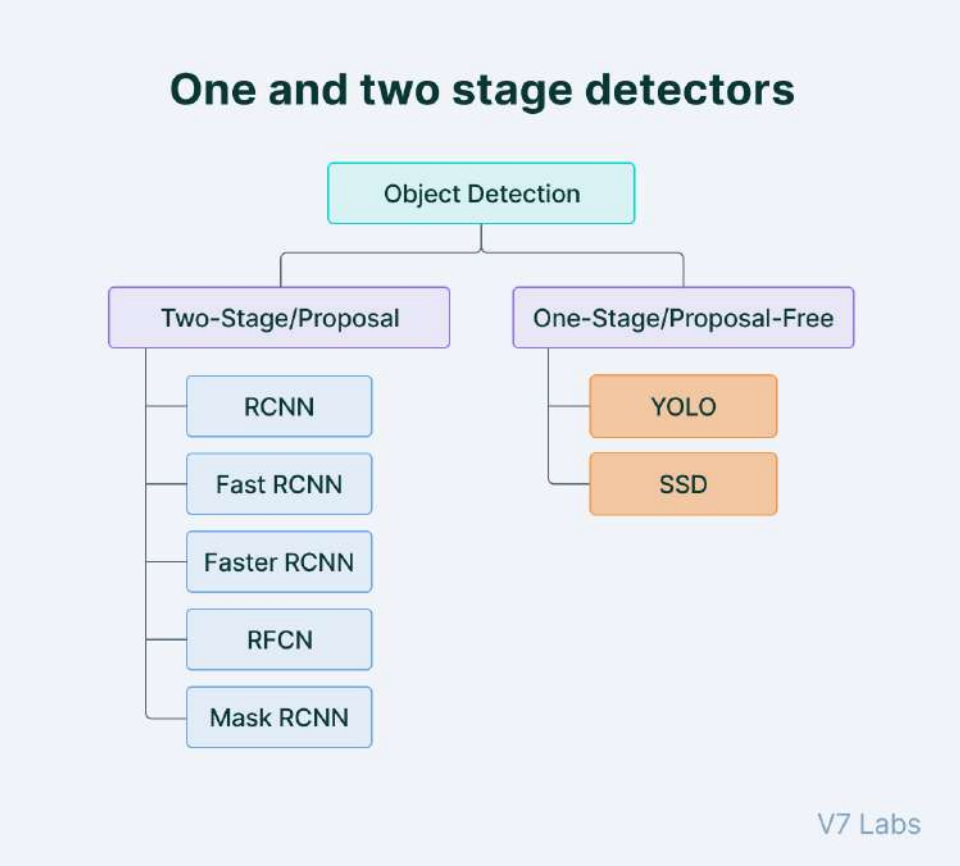
(a) Training efficiency.

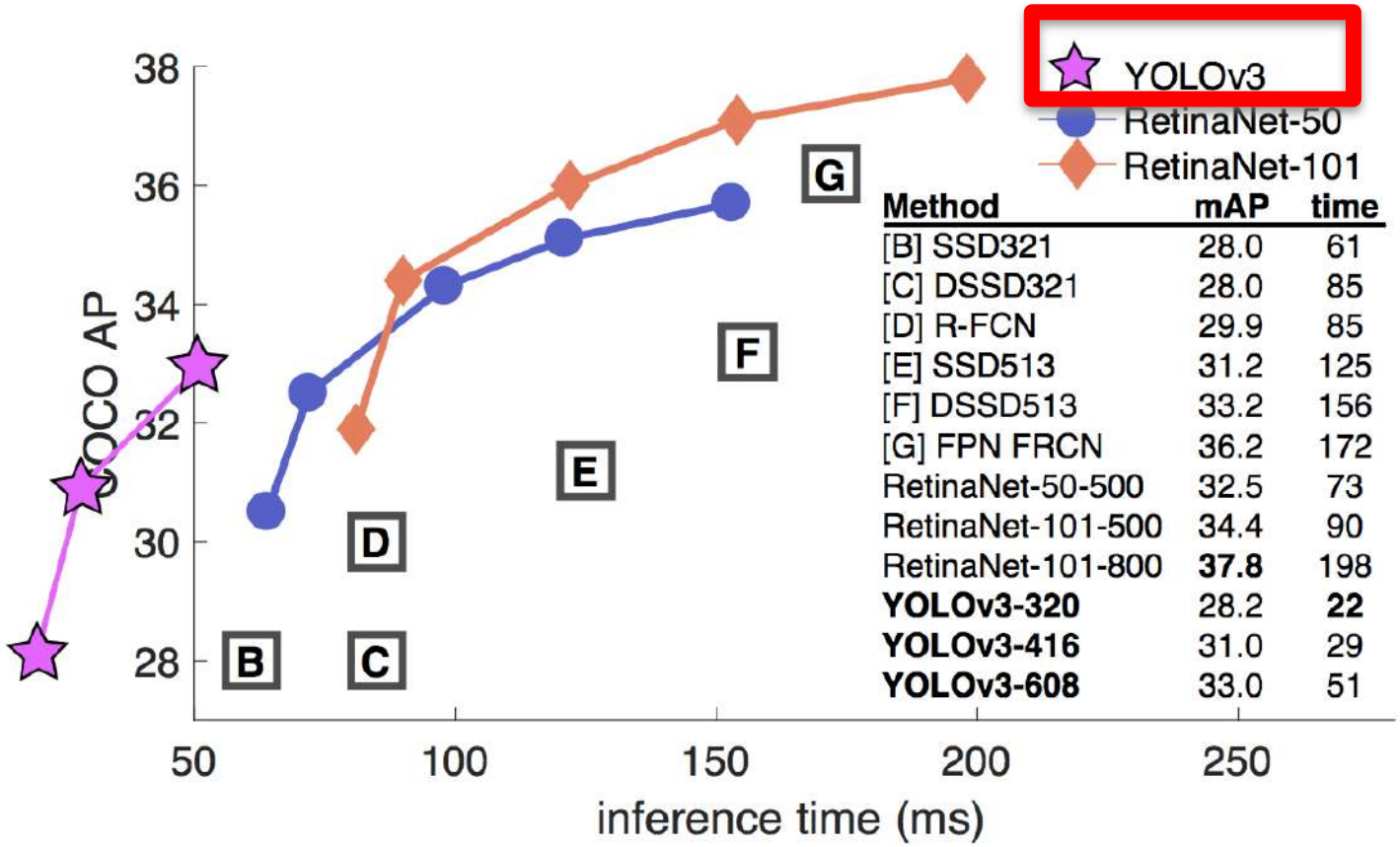
	EfficientNet (2019)	ResNet-RS (2021)	DeiT/ViT (2021)	EfficientNetV2 (ours)
Top-1 Acc.	84.3%	84.0%	83.1%	83.9%
Parameters	43M	164M	86M	24M

(b) Parameter efficiency.

SOTA of Object Detection







https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

YOLO Open Images in New York

YOLO's output

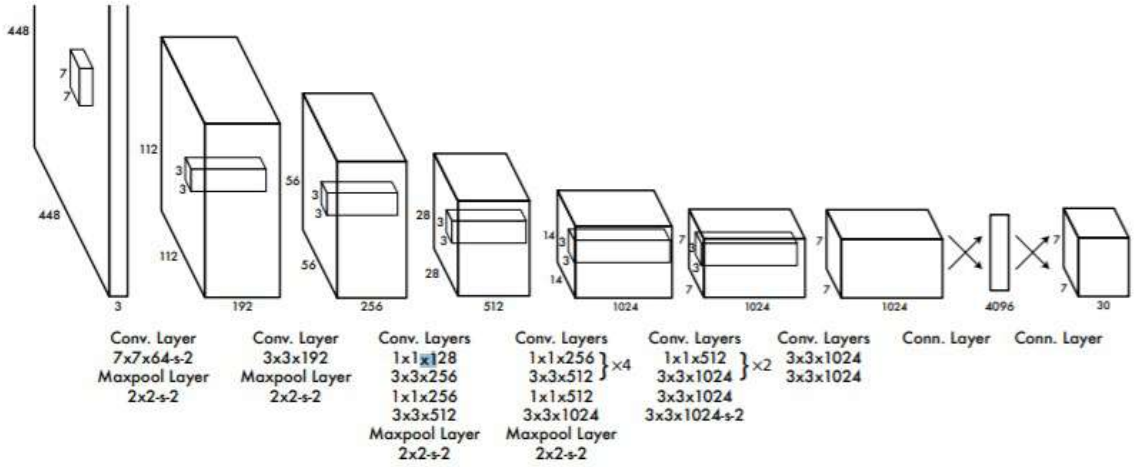
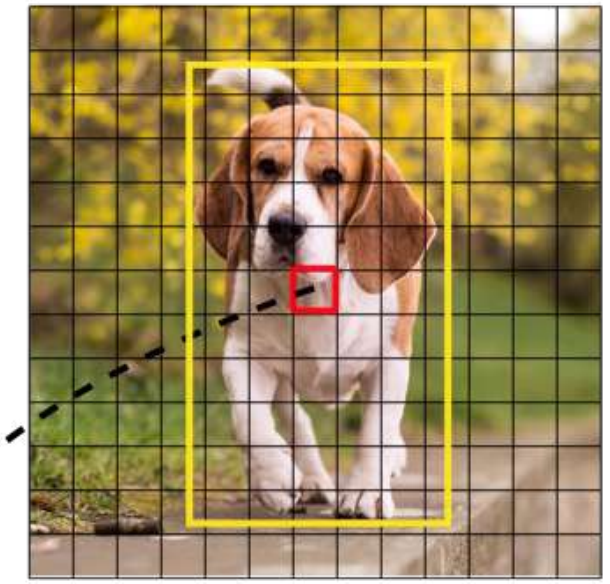
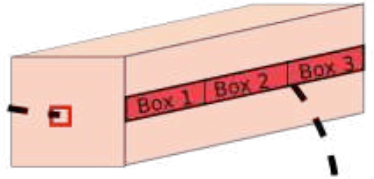


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

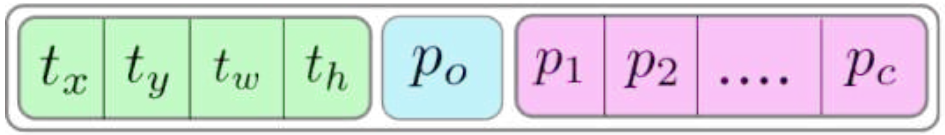
Image Grid. The Red Grid is responsible for detecting the dog



Prediction Feature Map



Attributes of a bounding box



https://blog.paperspace.com/how-to-implement-a-yolo-object-detector-in-pytorch/

SOTA of Object Detection (cont.)

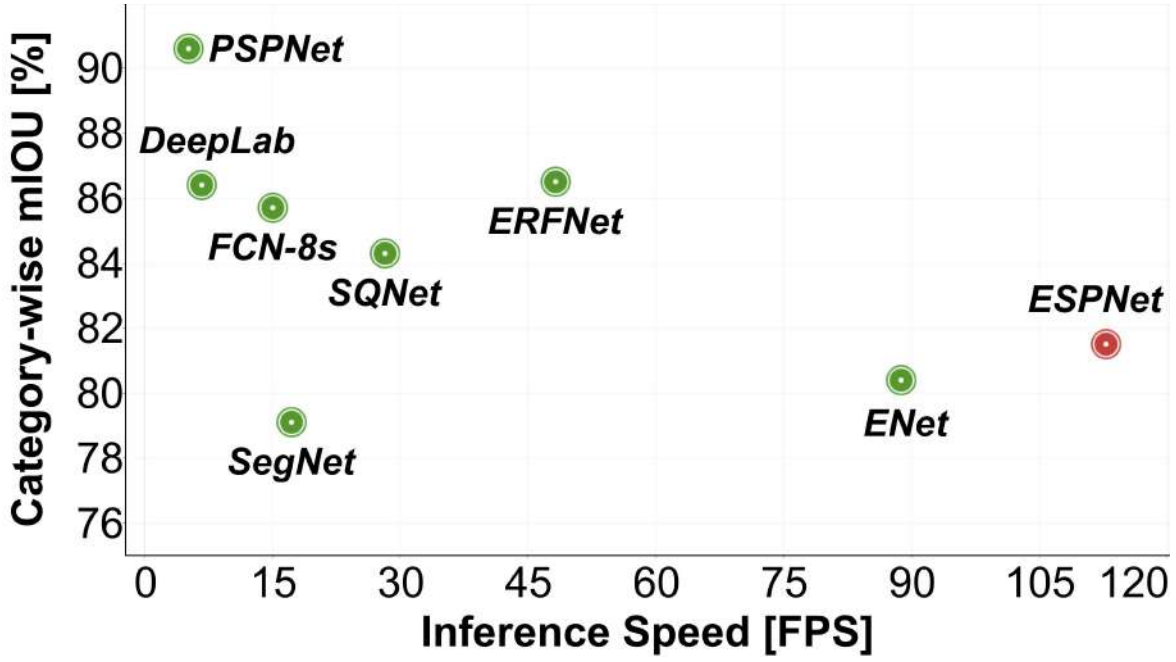
Top YOLO Variants Of 2021

You can read more about YOLOP here.

Rank	Model	box AP	AP50	AP75	APS	APM	APL	Extra Training Data	Paper	Code	Result	Year	Tags
1	YOLOR-D6 (1280, single-scale, 30 fps)	57.3	75.0	62.7	40.4	61.2	69.2	×	You Only Learn One Representation: Unified Network for Multiple Tasks			2021	
2	YOLOR-E6 (1280, single-scale, 37 fps)	56.4	74.1	61.6	39.1	60.1	68.2	×	You Only Learn One Representation: Unified Network for Multiple Tasks			2021	
3	YOLOv4-P7 with TTA	55.8	73.2	61.2				×	Scaled-YOLOv4: Scaling Cross Stage Partial Network			2020	
4	YOLOR-W6 (1280, single-scale, 47 fps)	55.5	73.2	60.6	37.6	59.5	67.7	×	You Only Learn One Representation: Unified Network for Multiple Tasks			2021	
5	YOLOv4-P7 CSP-P7 (single-scale, 16 fps)	55.4	73.3	60.7	38.1	59.5	67.4	×	Scaled-YOLOv4: Scaling Cross Stage Partial Network			2020	

https://medium.com/augmented-startups/top-yolo-variants-of-2021-19dddc23043c

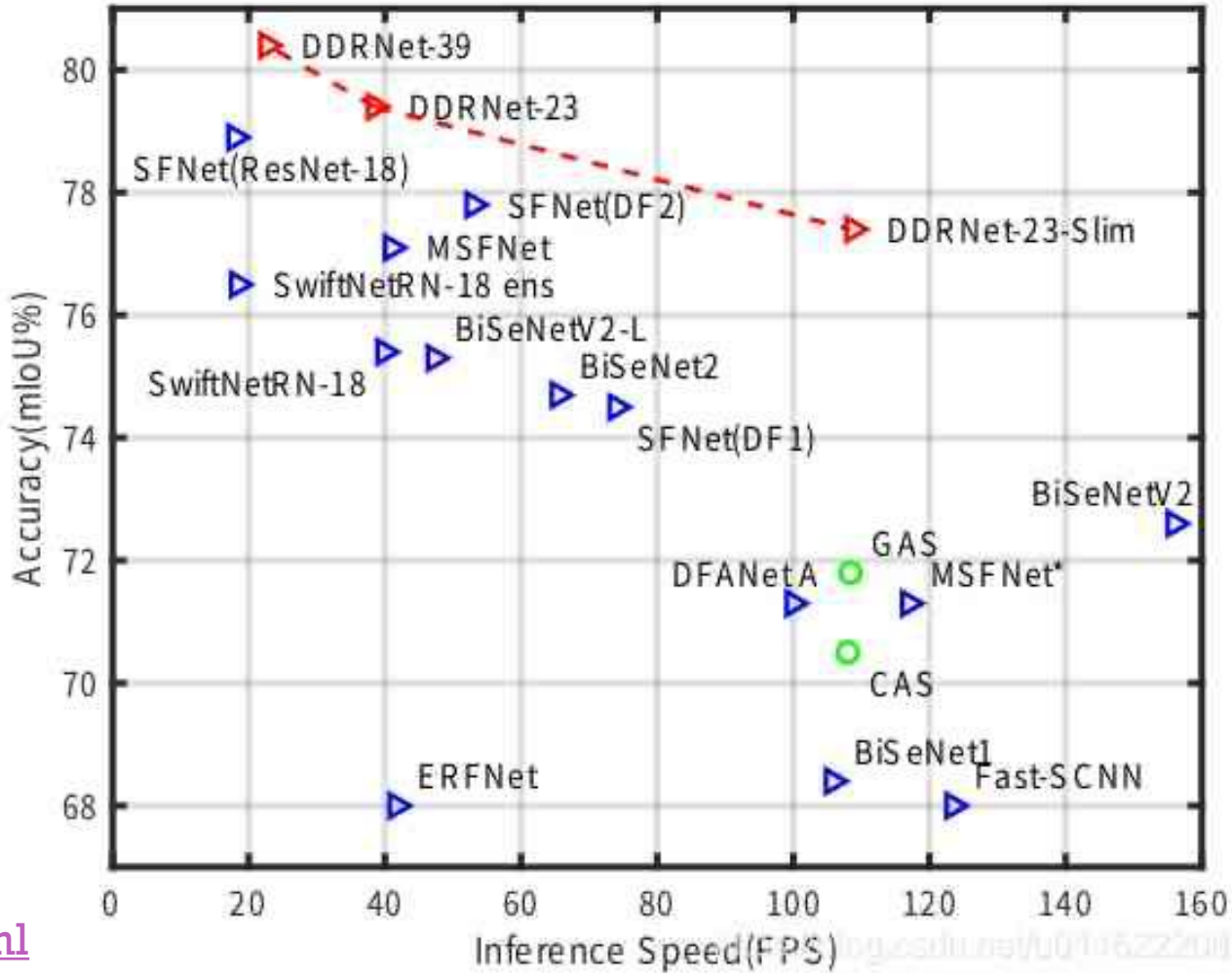
SOTA of Semantic Segmentation



- **Real-time semantic segmentation**
- BiSeNet V1,V2
- DDRNet

[Personal open source] - -
Real - time Semantic Segmentation ddrnet

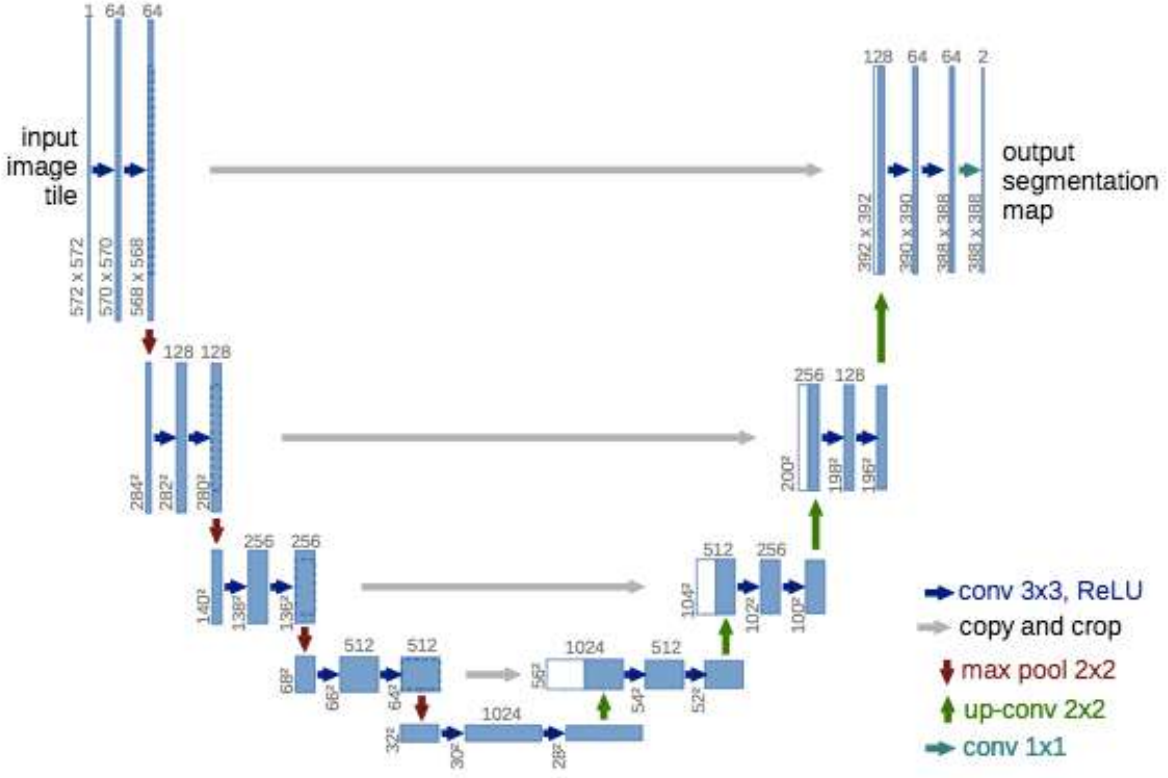
2021-11-09 04:24:01 【Nongfu Mountain Spring 2】

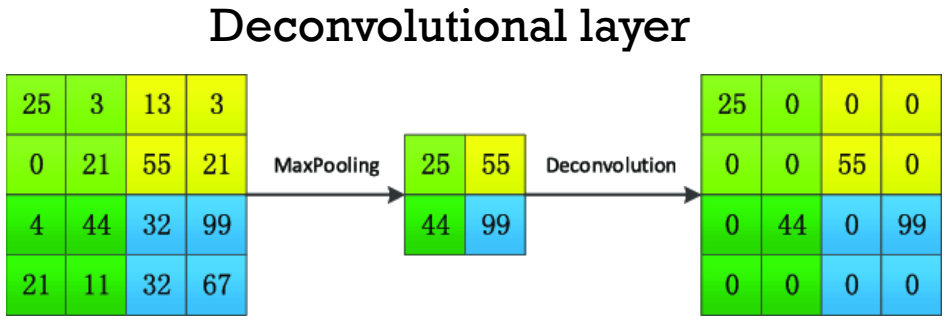


https://chowdera.com/2021/11/20211109042359120o.html

Review of Deep Learning Algorithms for Semantic Segmentation (cont.)

- **U-Net**

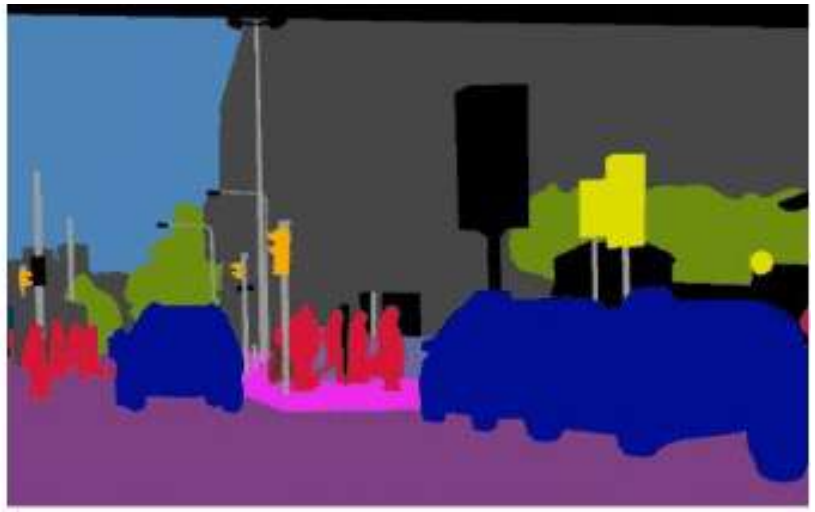




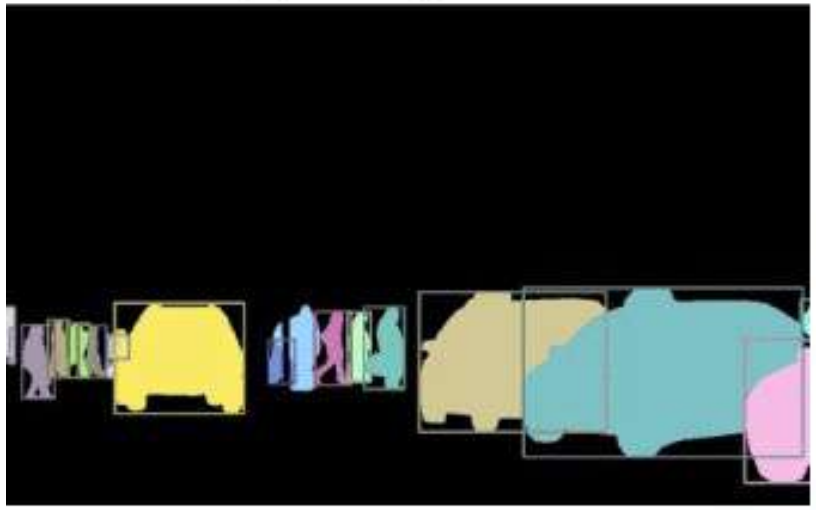
Architecture of the U-net for a given input image. The blue boxes correspond to feature maps blocks with their denoted shapes. The white boxes correspond to the copied and cropped feature maps.
Source: O. Ronneberger et al. (2015)

Panoptic Segmentation

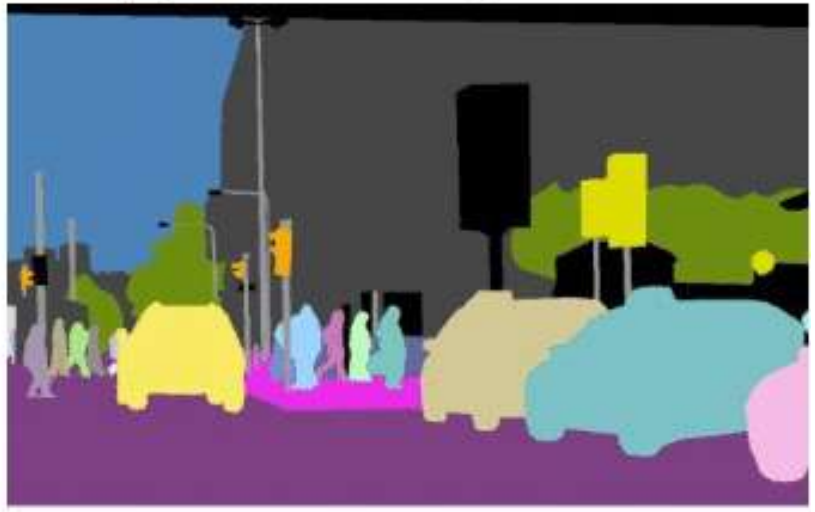
(a) Image



(b) Semantic Segmentation



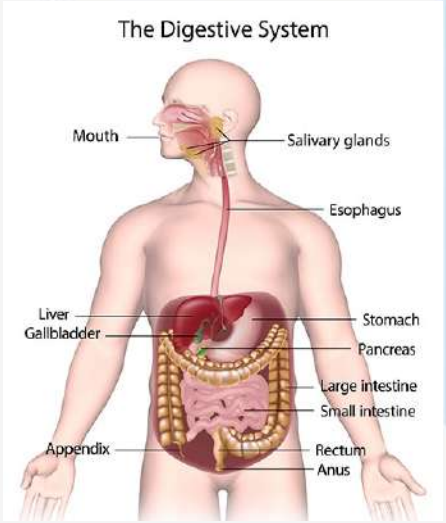
(c) Instance Segmentation

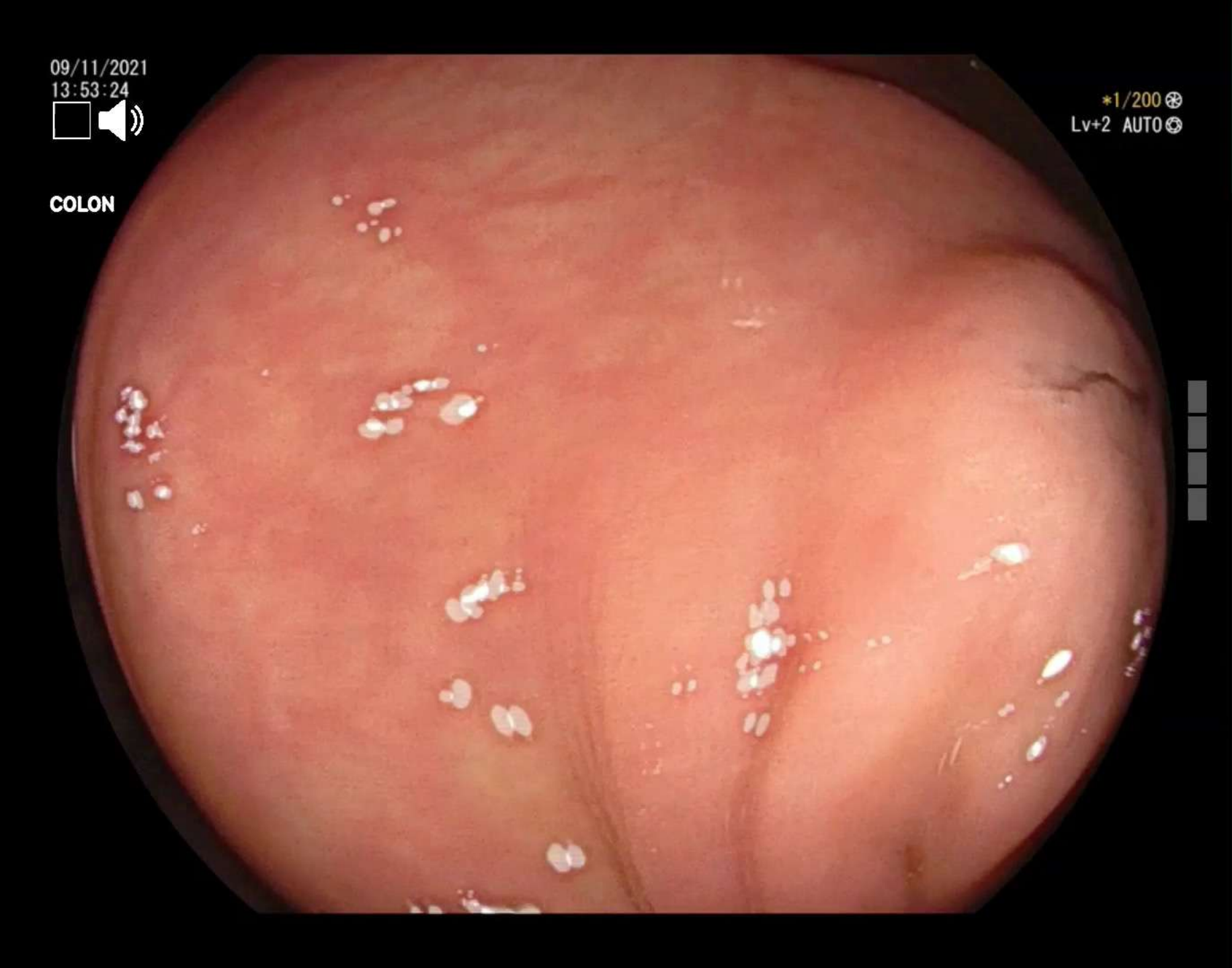


(d) Panoptic Segmentation

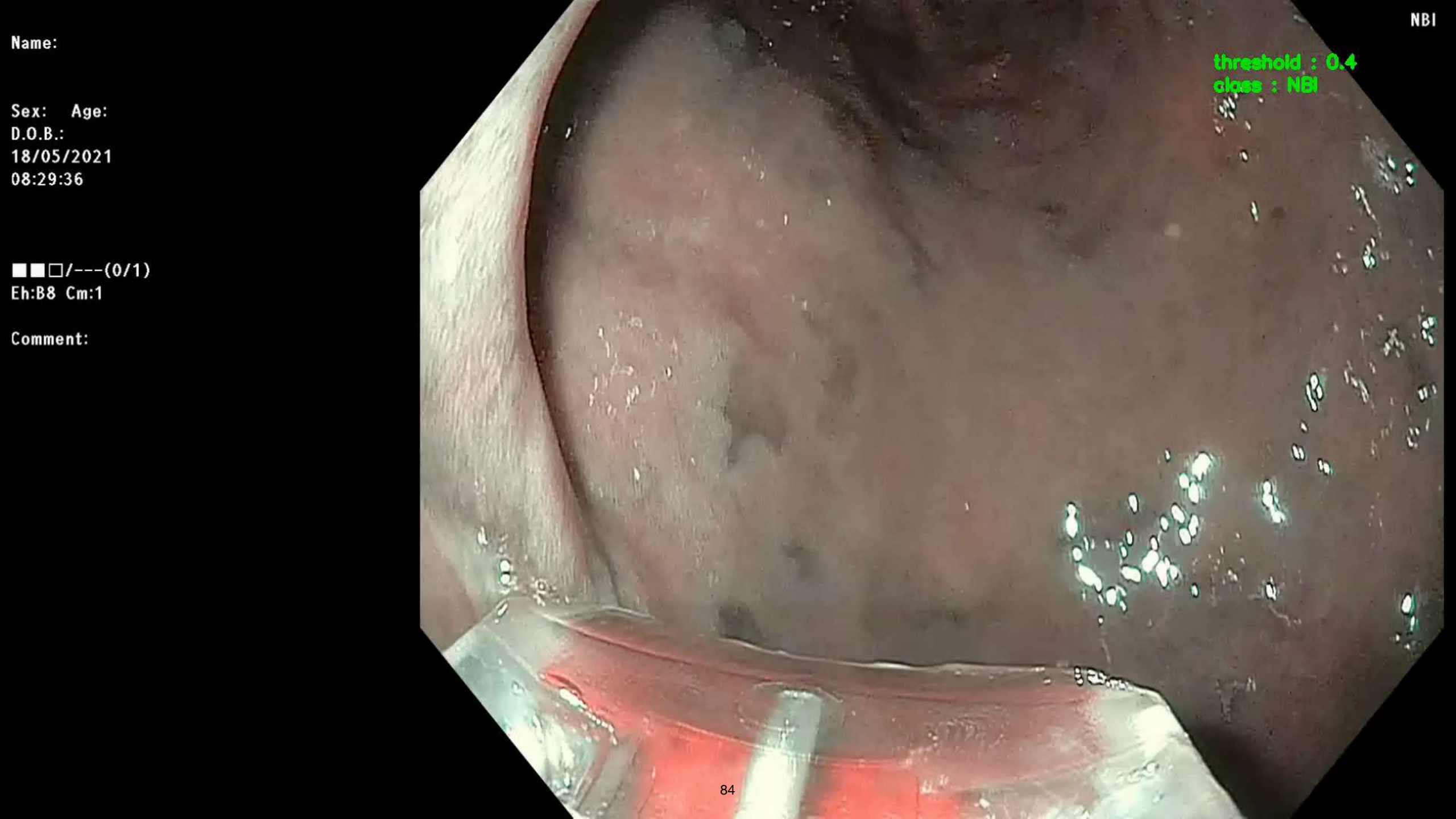
Real-Time Colonic Polyp Detection

An AI-assisted solution that aims to improve colonic polyp detection in real-time. It is compatible with all standard colonoscope systems.

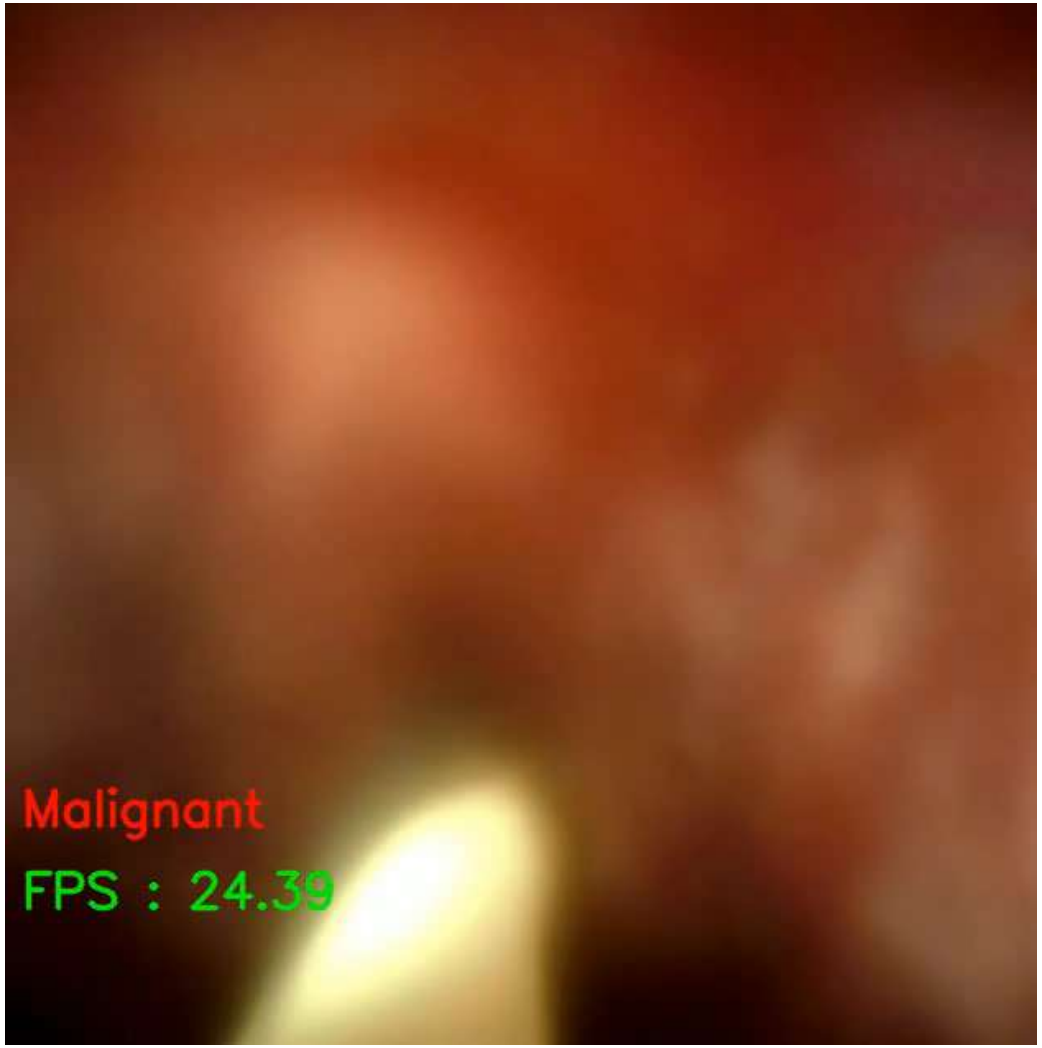


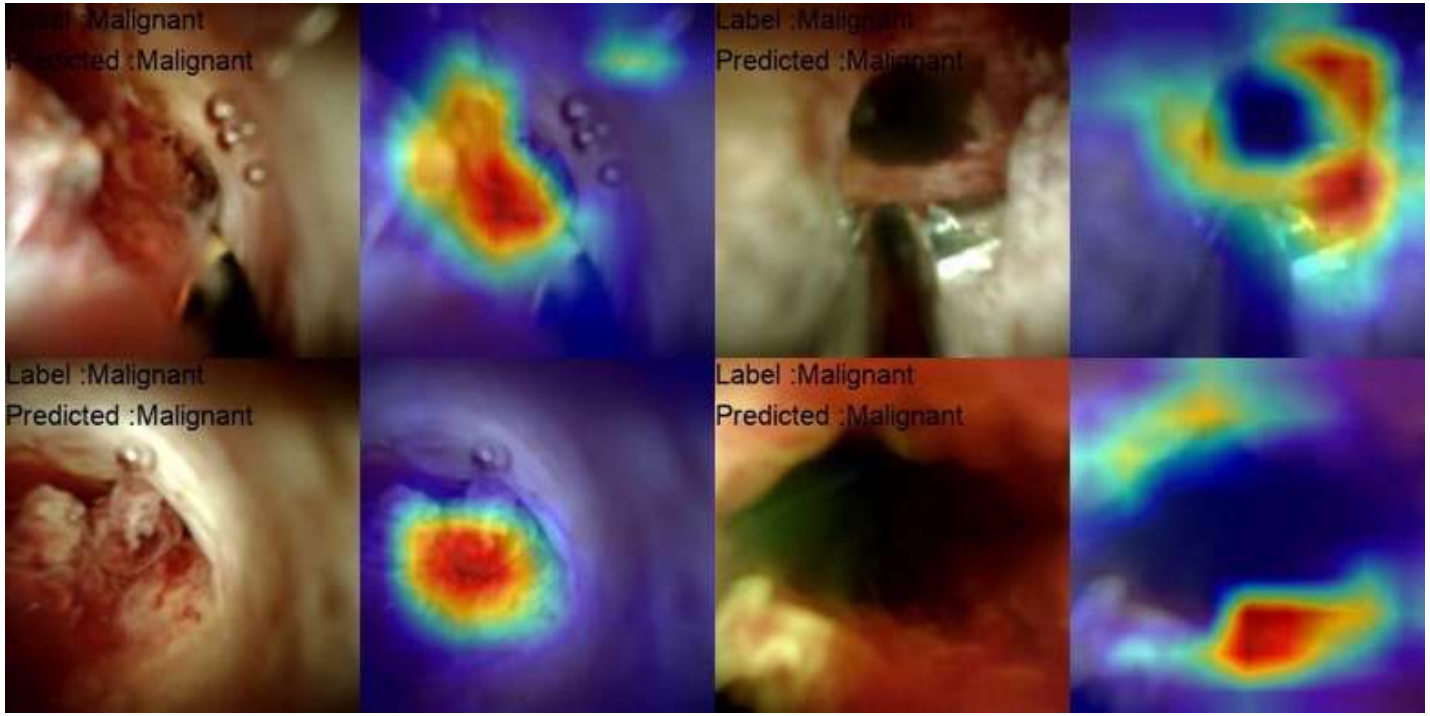

09/11/2021
13:53:24
COLON
*1/200
Lv+2 AUTO


HT NR
SE
3.8
12.0
12.0
S1: F+T
S2: LM
S3: CAD
S4:
EC-760R-V/L
3C729K035
BL-7000
CHULALONGKORN HOS


Name:
Sex: Age:
D.O.B.:
18/05/2021
08:29:36
■■□/---(0/1)
Eh:B8 Cm:1
Comment:
NBI
threshold : 0.4
class : NBI

Cholangioscopy





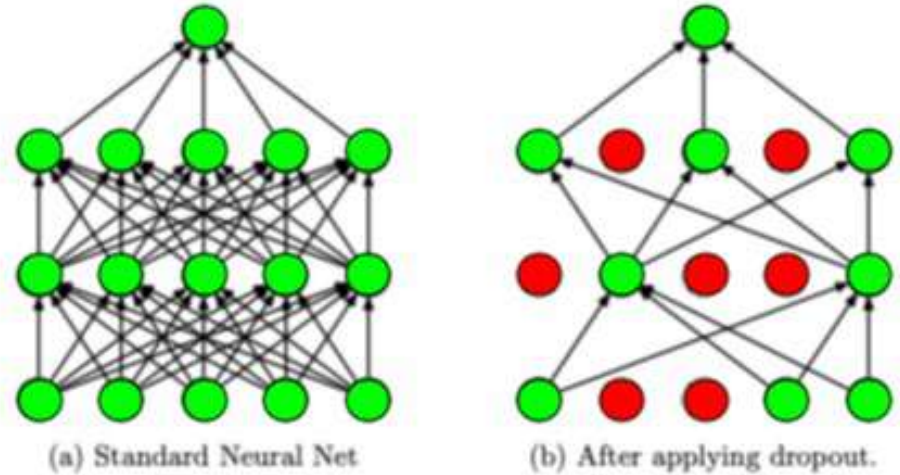
More techniques to prevent overfitting

- Dropout
- Augmentation
- Best validation loss

Droupout

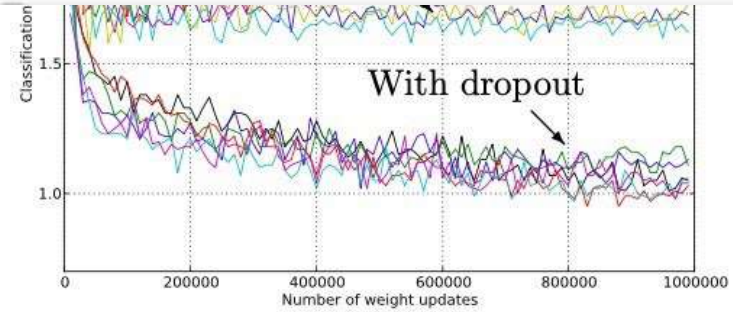
- The Dropout layer prevents overfitting the model during training. Notably, Dropout randomly deactivates some neurons of a layer, thus nullifying their contribution to the output.

- Dropout is only used in training, so we don't want these weights to be fixed at this high a number during testing.



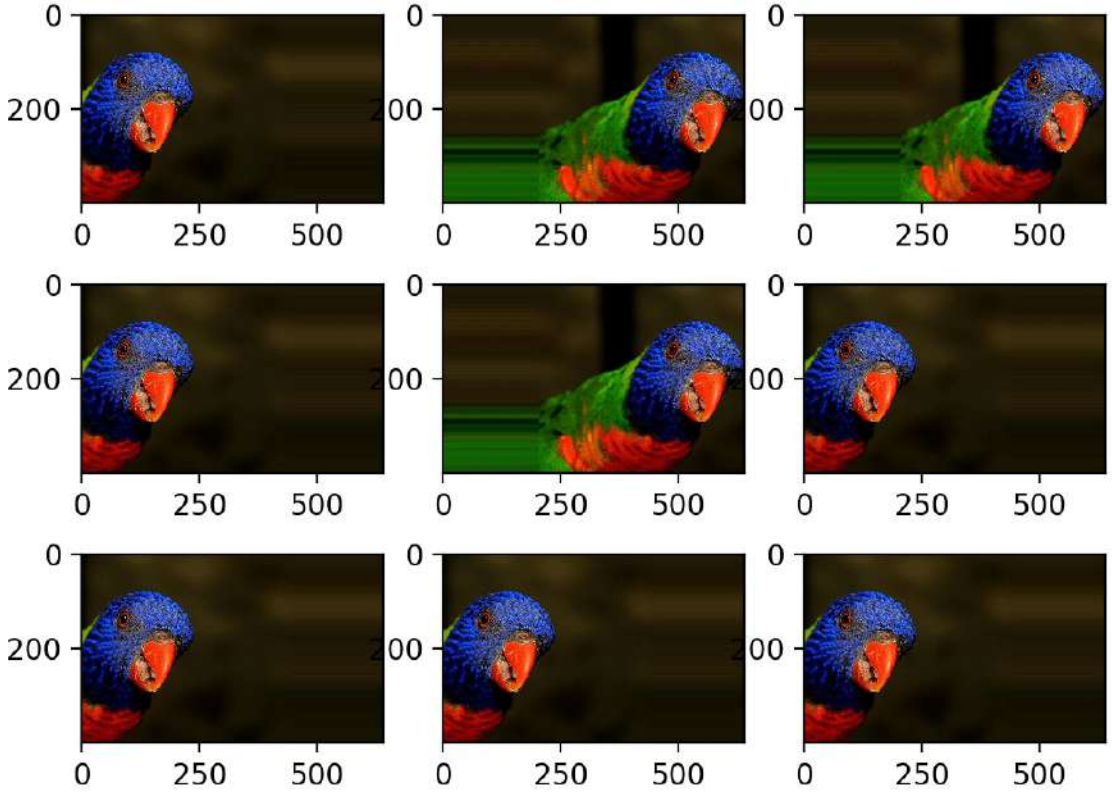
(a) Standard Neural Net (b) After applying dropout.

```
#create model
model = Sequential()
#add model layers
model.add(Conv2D(64, kernel_size=3, activation='relu', input_shape=(28,28,1)))
model.add(Conv2D(32, kernel_size=3, activation='relu'))
model.add(Dropout(.5, noise_shape=None, seed=None)) #dropout layer
model.add(Conv2D(32, kernel_size=3, activation='relu'))

model.add(Flatten())
model.add(Dense(10, activation='softmax'))
```

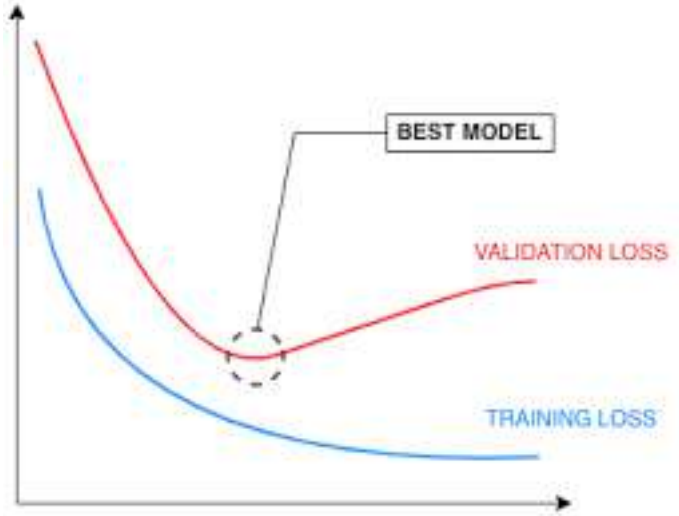


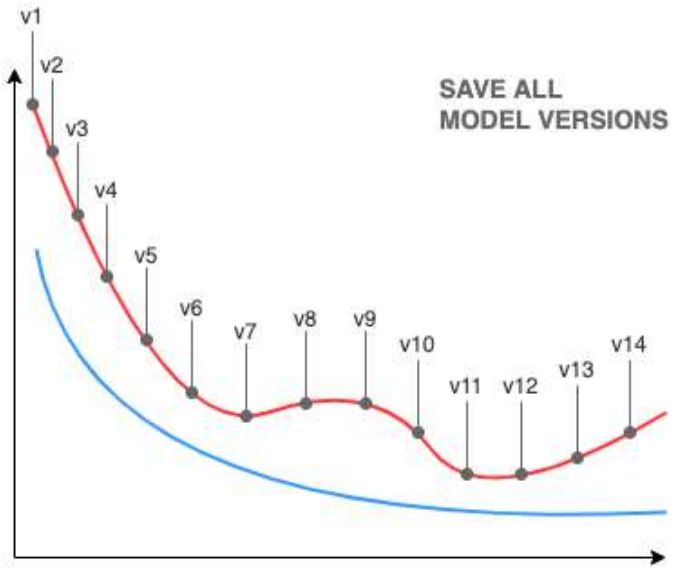
https://deepakbattini.medium.com/implementing-drop-out-regularization-in-neural-networks-ab2fc8d985e8

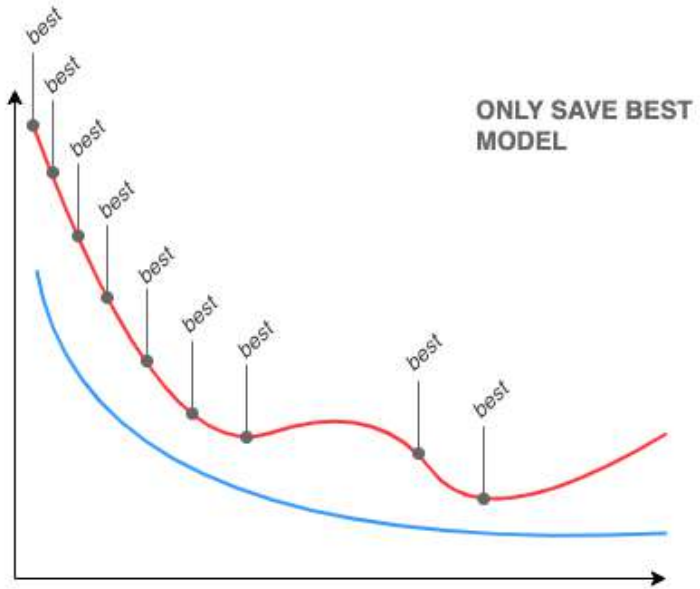
Augmentation



Best validation loss







https://nicjac.dev/posts/identify-best-model/

+

RNN

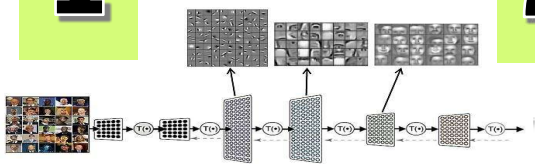
Deep Learning Application

Speech Recognition

Computer Vision

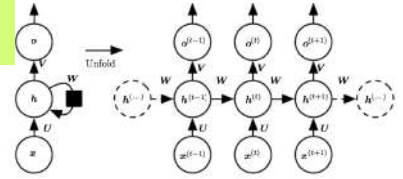
Natural Language Processing

1



CNN

2



RNN (LSTM)

Top 10 Strategic Technology Trends for 2020

People-centric

- Hyperautomation
- Multiexperience
- Democratization
- Human Augmentation
- Transparency and Traceability

Smart spaces

- Empowered Edge
- Distributed Cloud
- Autonomous Things
- Practical Blockchain
- AI Security

gartner.com/SmarterWithGartner

Source: Gartner

Gartner

Gartner Top Strategic Technology Trends for 2021

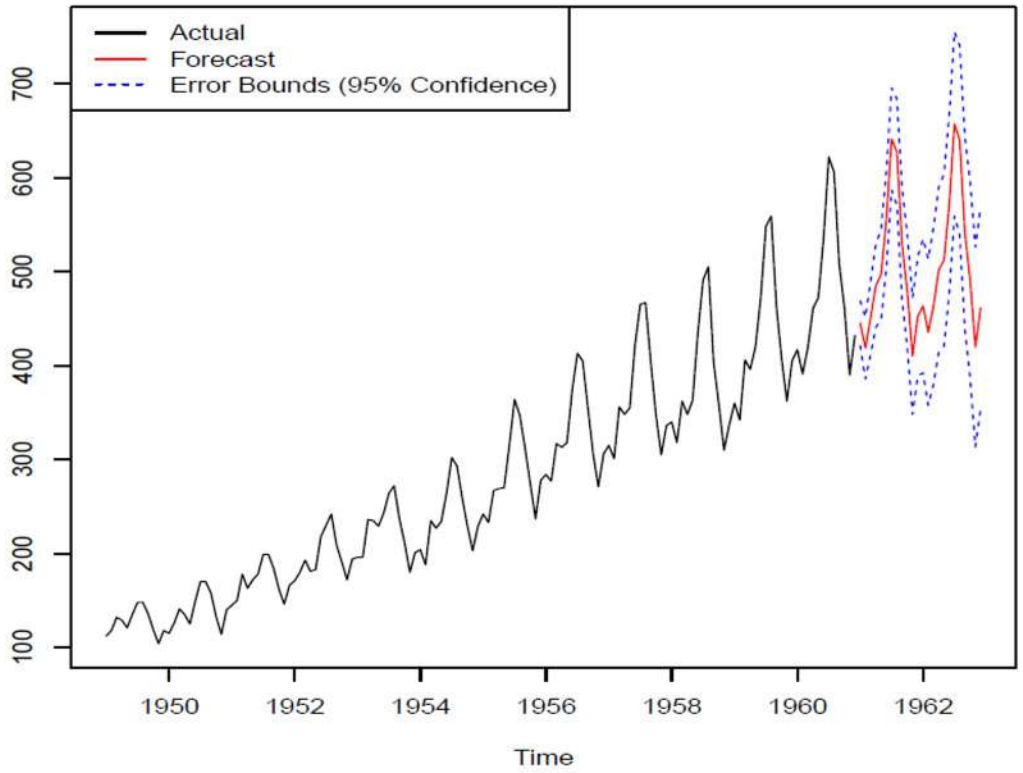
People centricity	Location independence	Resilient delivery
Internet of Behaviors	Distributed cloud	Intelligent composable business
Total experience strategy	Anywhere operations	AI engineering
Privacy-enhancing computing	Cybersecurity mesh	Hyperautomation

Combinatorial innovation

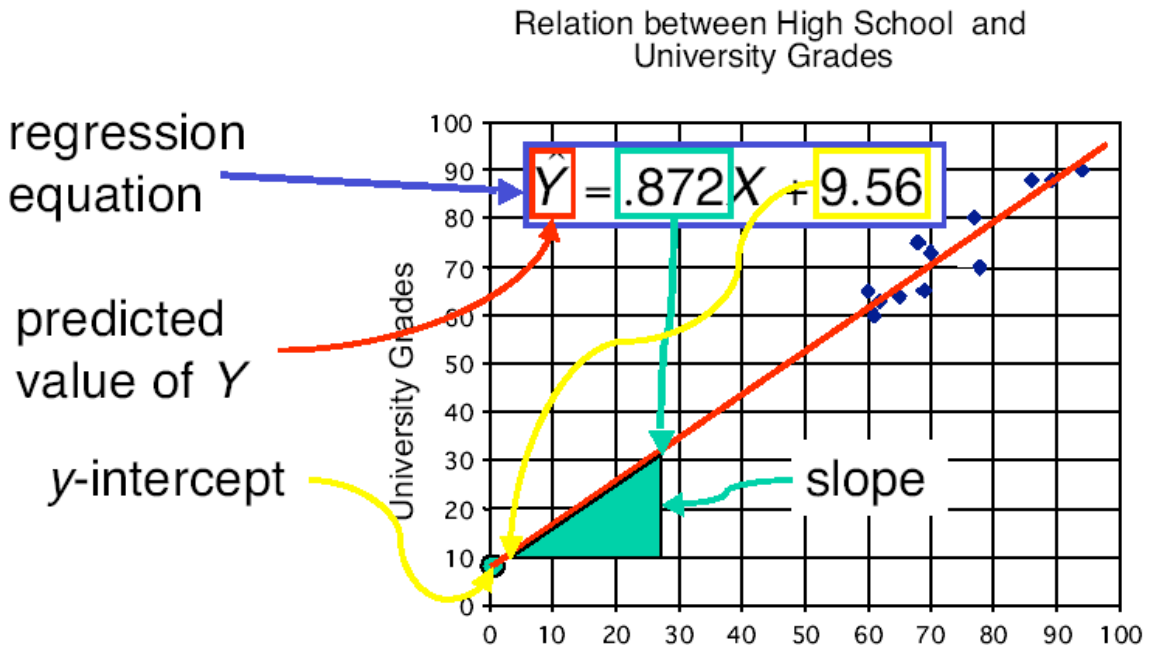
gartner.com/SmarterWithGartner

Source: Gartner

Gartner



Regression – Linear Relationship



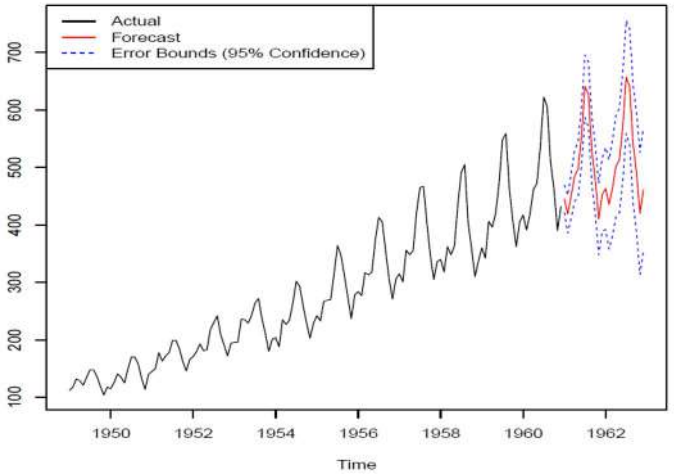
weight, coefficient

$$\underset{\text{target}}{\hat{y}} = \underset{\text{intercept}}{\hat{w}_0} + \hat{w}_1 x_1 + \hat{w}_2 x_2 \quad \text{input}$$

- The least square method aims to minimize the following term

$$\sum_{\substack{training \\ data}} (y_i - \hat{y}_i)^2$$

Autoregressive Model – Linear Relationship



- Based on linear regression, but using previous timestep data

$$X_t = c + \Sigma_{i=1}^{p} \varphi_i X_{t-i}$$

- Where X_t is data at timestep t, c is constant, and $\varphi_1, \varphi_2, \varphi_3, \cdots, \varphi_p$ are parameters for each timestep data

Example: When p = 2 (looking back two steps), the equation will be

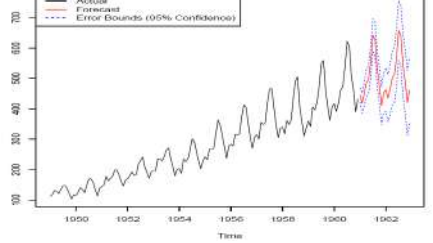
$$X_t = c + \varphi_1 X_{t-1} + \varphi_2 X_{t-2}$$

- Parameters are able to calculate using various method, such as ordinary least square procedure or Yule–Walker equations

$$x(t+1) = \hat{w}_0 + \hat{w}_1 x(t) + \hat{w}_2 x(t-1)$$

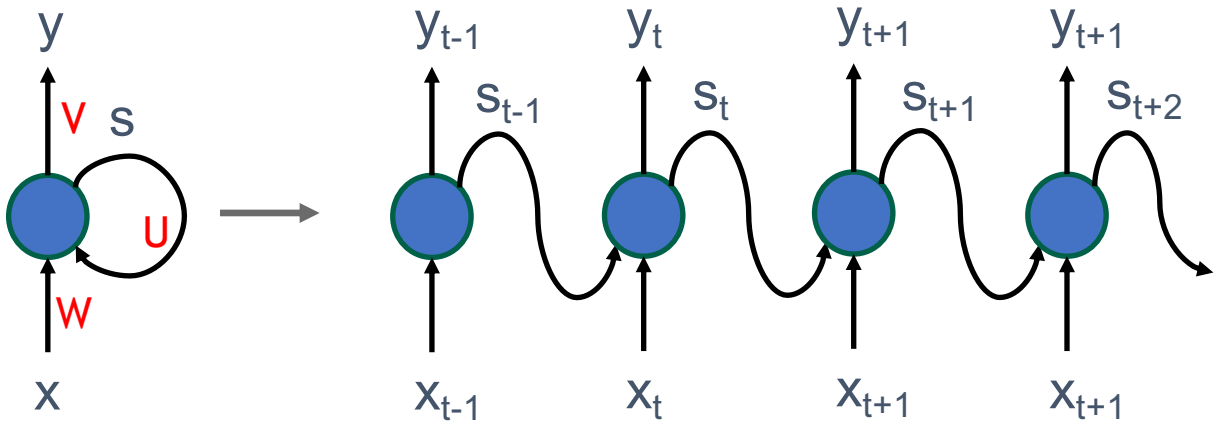
RECURRENT NEURONS (RNN)

Non-Linear Relationship



Forward Pass

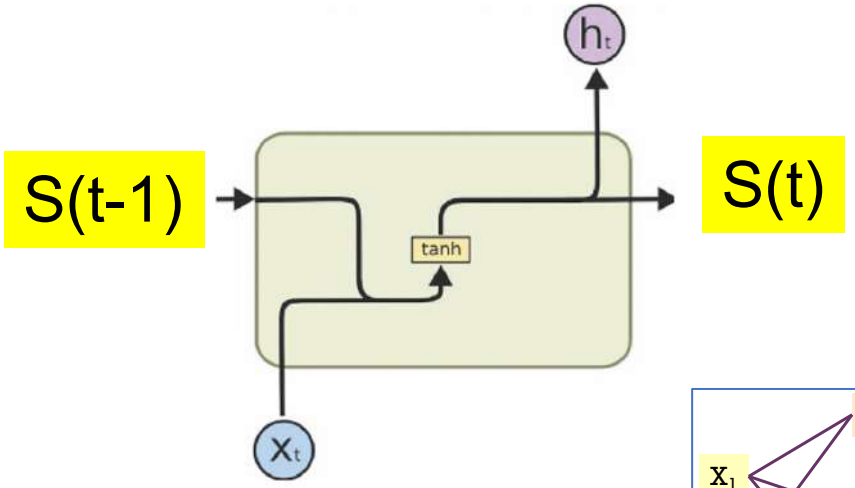
x(t) x(t+1) x(t+2) x(t+3)

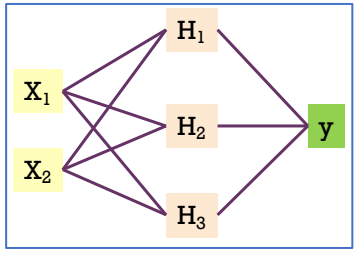


Unrolling in Time

x(t+1)

y(t)





$$s_t = \tanh\left(Ux_t + Ws_{t-1}\right)$$

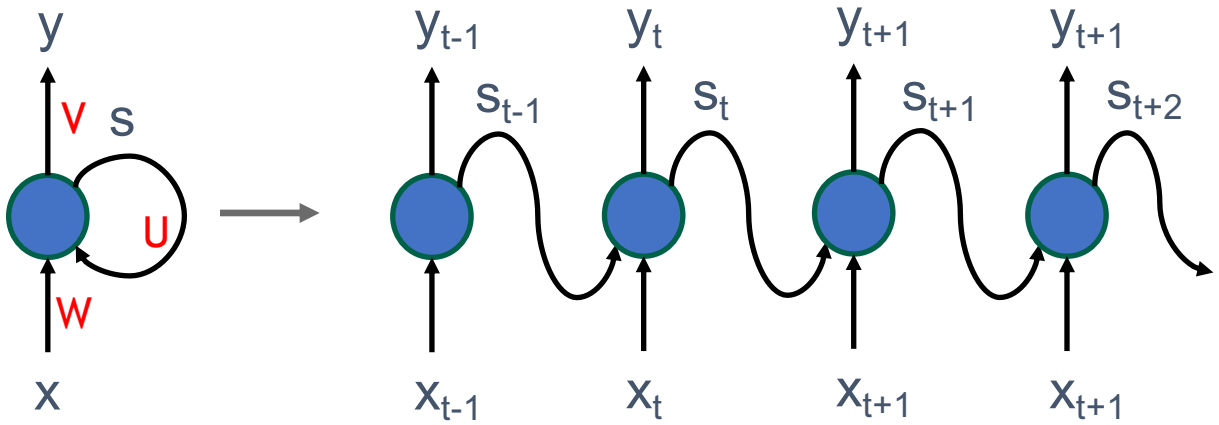
x(t+1) $$\hat{y}_t = softmax\left(Vs_t\right)$$

Reference: https://www.cse.iitk.ac.in/users/sigml/lec/Slides/LSTM.pdf

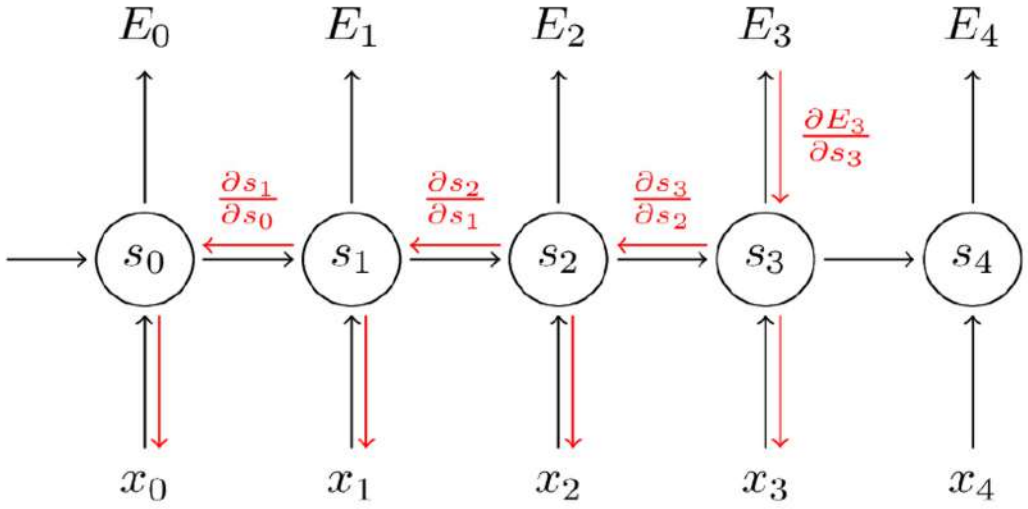
RECURRENT NEURONS (RNN)

Ggradient vanishing
= multiplication

Forward Pass



Backward Pass

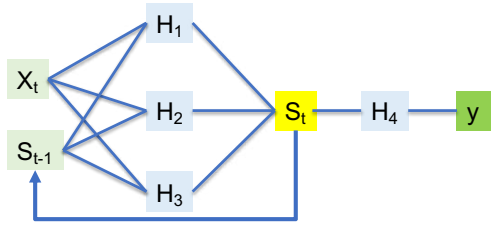


Unrolling in Time

Backpropagation-Through-Time (BPTT)

$$s_t = \tanh\left(Ux_t + Ws_{t-1}\right)$$

$$\hat{y}_t = softmax\left(Vs_t\right)$$

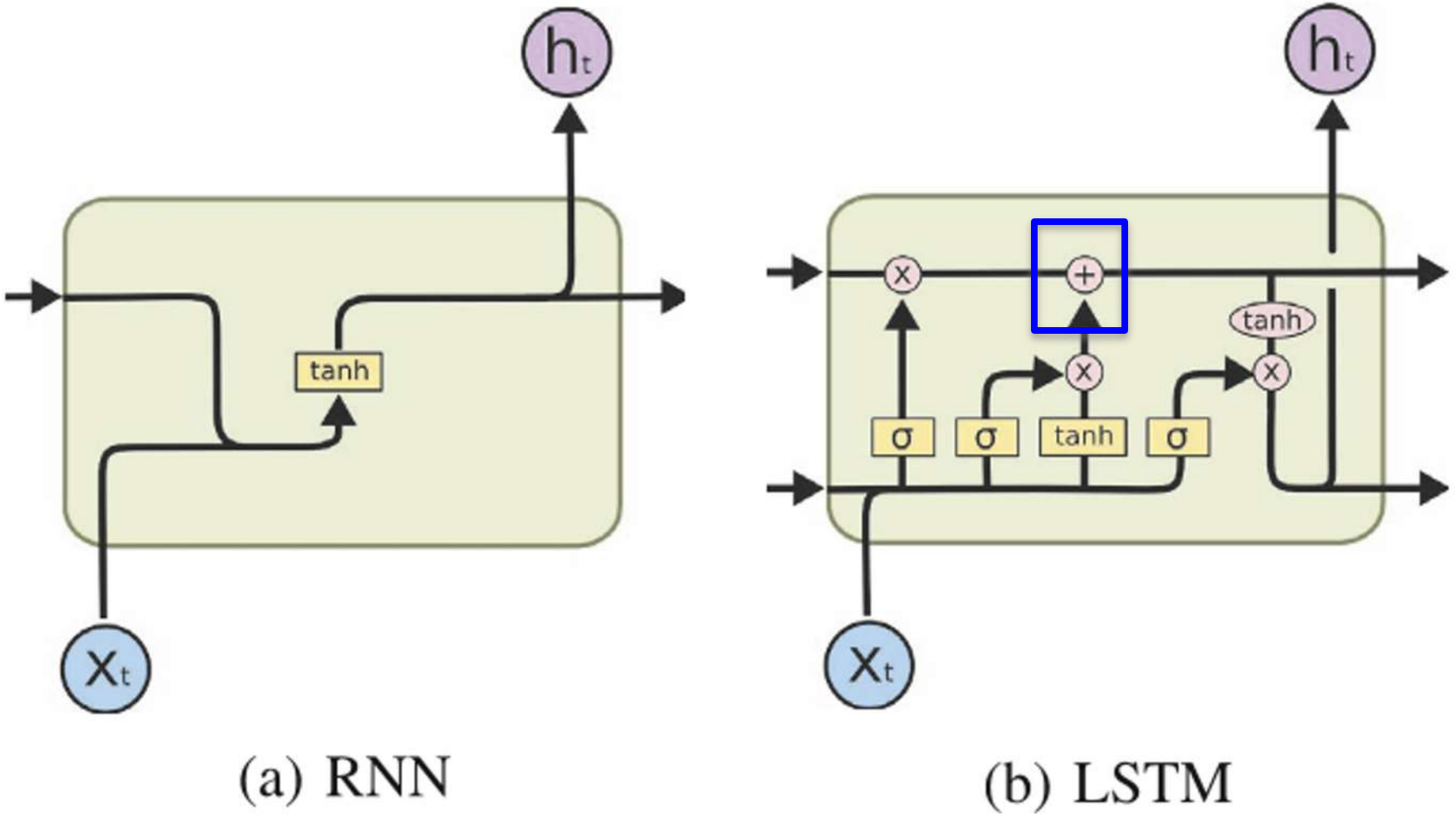


$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^{3} \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}$$

Reference: https://www.cse.iitk.ac.in/users/sigml/lec/Slides/LSTM.pdf

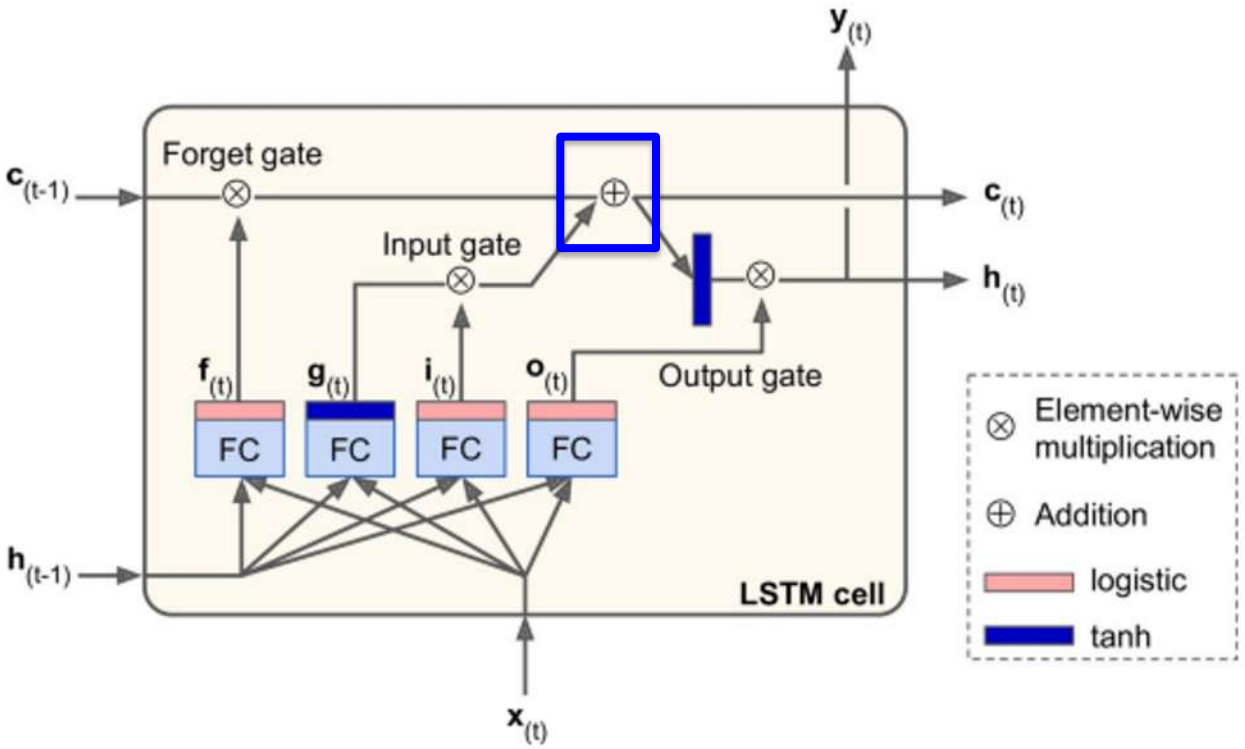
RNN VS LSTM



(a) RNN (b) LSTM

Reference: https://cs.uwaterloo.ca/~mli/Deep-Learning-2017-Lecture6RNN.ppt

LONG SHORT TERM (LSTM) CELL



There are 3 gates with 2 outputs.

$$\mathbf{i}_{(t)} = \sigma(\mathbf{W}_{xi}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hi}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_i)$$

$$\mathbf{f}_{(t)} = \sigma(\mathbf{W}_{xf}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hf}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_f)$$

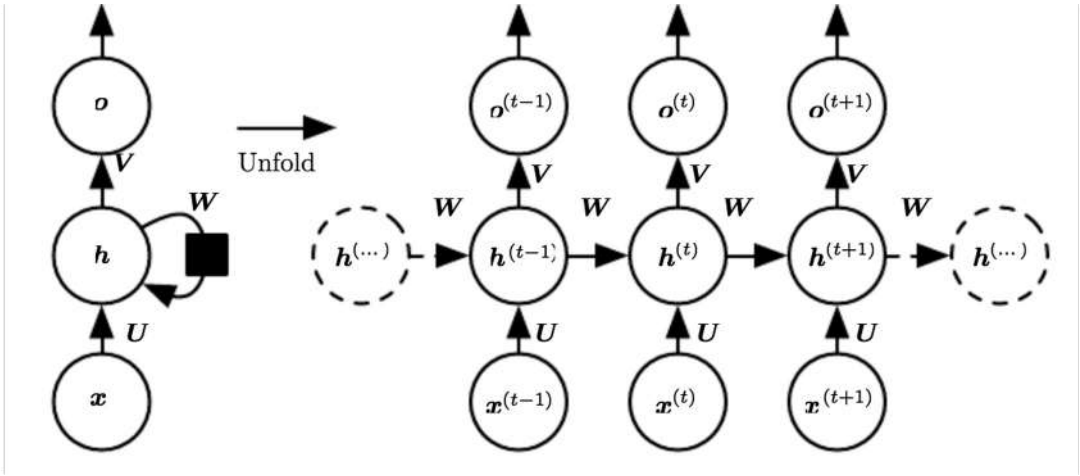
$$\mathbf{o}_{(t)} = \sigma(\mathbf{W}_{xo}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{ho}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_o)$$

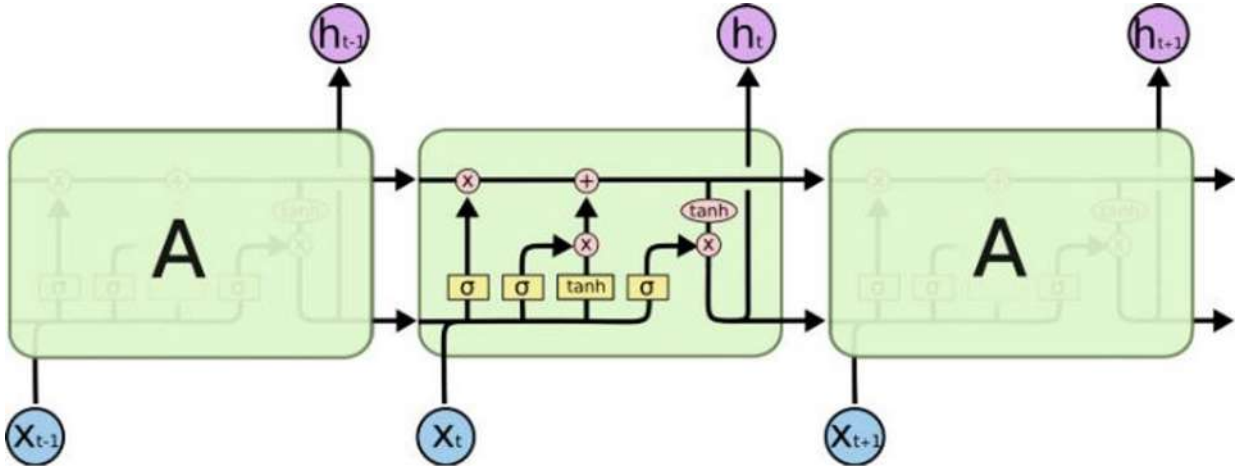
$$\mathbf{g}_{(t)} = \tanh(\mathbf{W}_{xg}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_g)$$

$$\mathbf{c}_{(t)} = \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)}$$

$$\mathbf{y}_{(t)} = \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh(\mathbf{c}_{(t)})$$

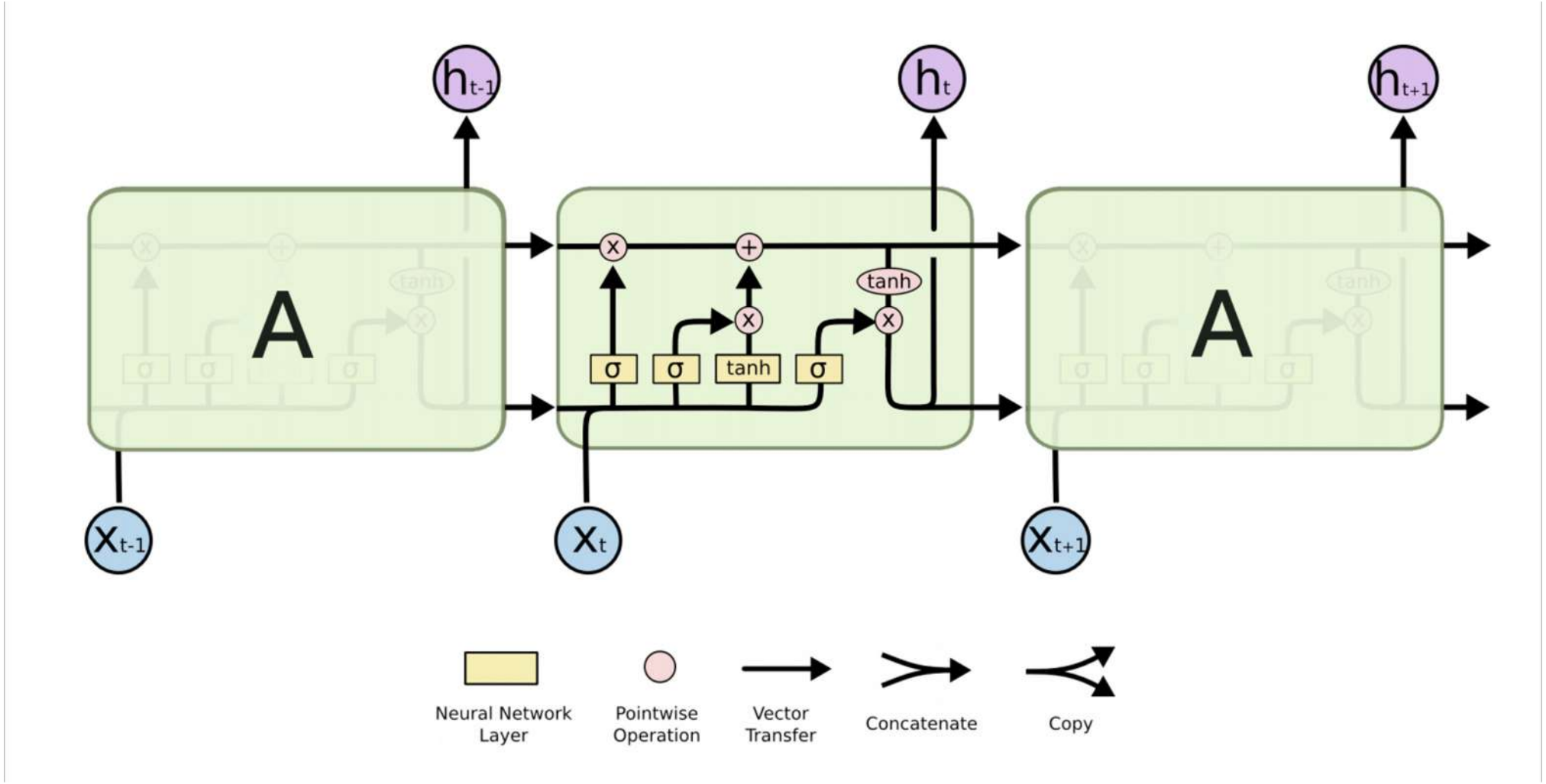
Deep Learning Approach: LSTM



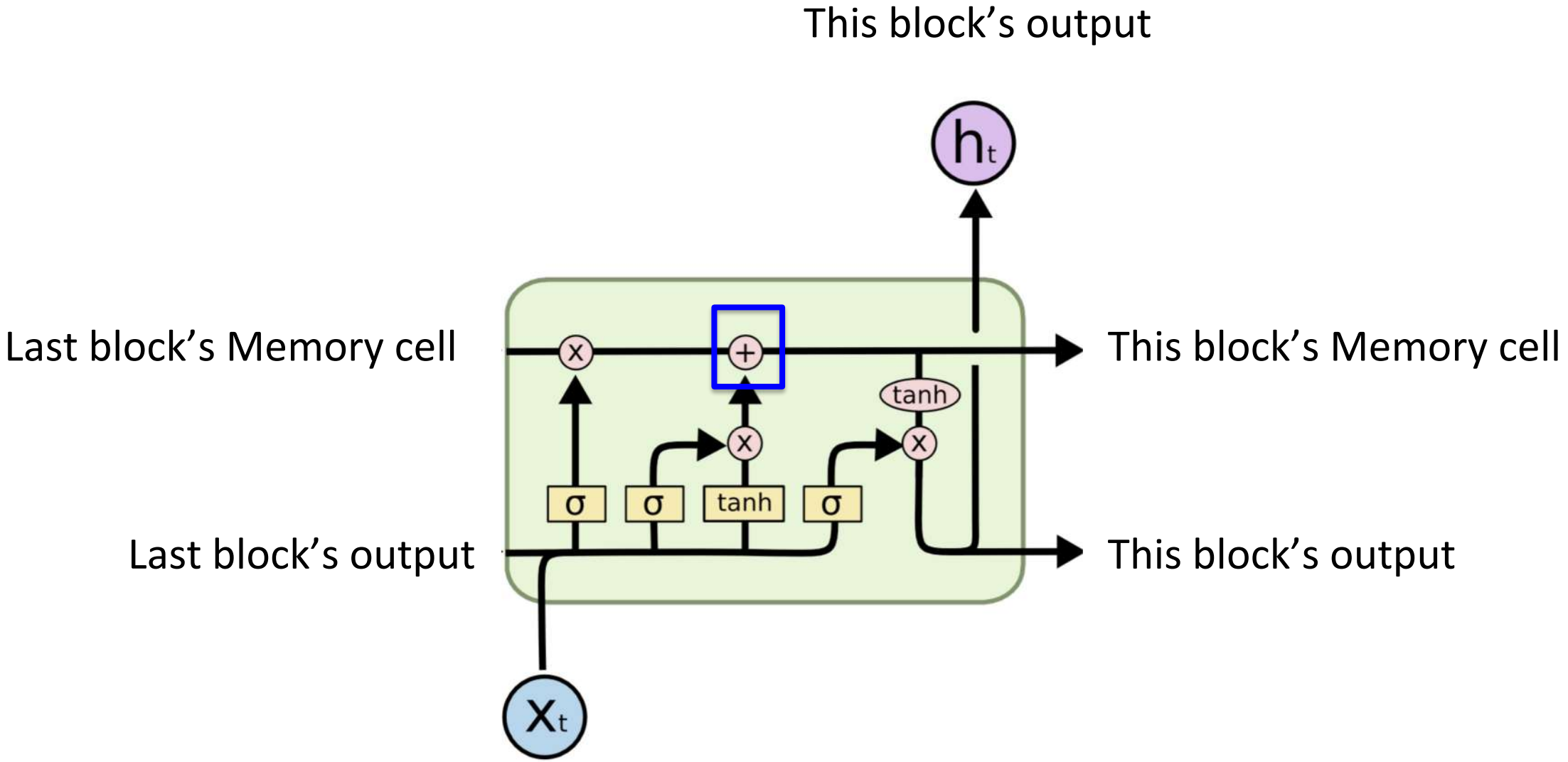


Reference: Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. Deep Learning. MIT Press. http://www.deeplearningbook.org

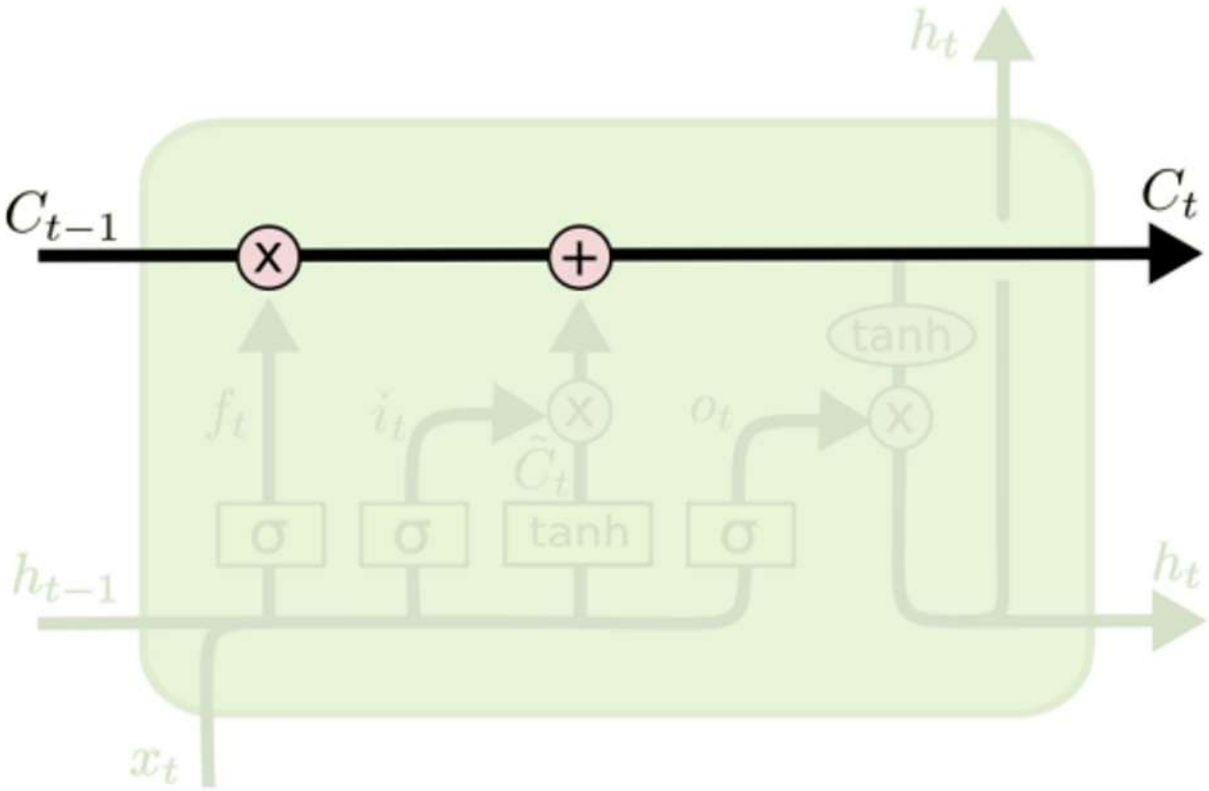
Inside LSTM

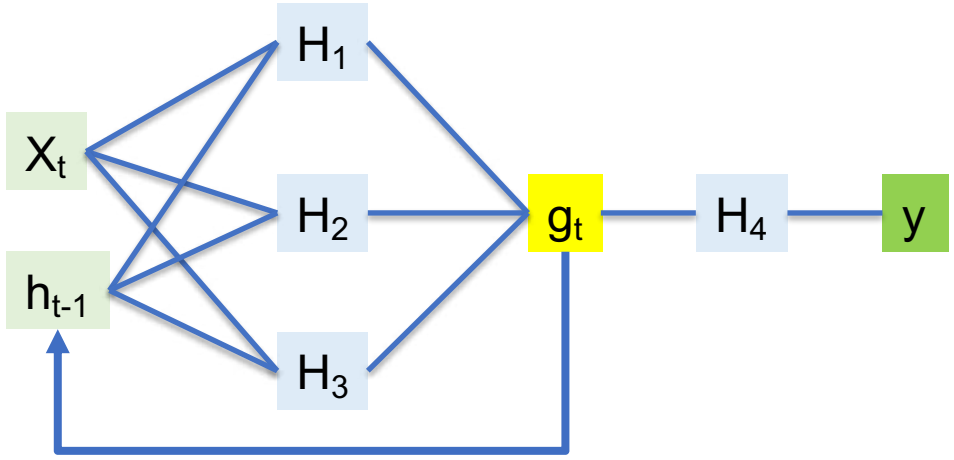

h_{t-1}
h_t
h_{t+1}
A
A
tanh
σ
σ
tanh
σ
X_{t-1}
X_t
X_{t+1}
Neural Network Layer
Pointwise Operation
Vector Transfer
Concatenate
Copy

Inside LSTM


This block's output
h_t
Last block's Memory cell
This block's Memory cell
tanh
x
+
x
x
σ
σ
tanh
σ
Last block's output
This block's output
X_t

Inside LSTM: (1) Cell Memory





There are 3 gates with 2 outputs.

$$\mathbf{i}_{(t)} = \sigma(\mathbf{W}_{xi}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hi}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_i)$$

$$\mathbf{f}_{(t)} = \sigma(\mathbf{W}_{xf}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hf}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_f)$$

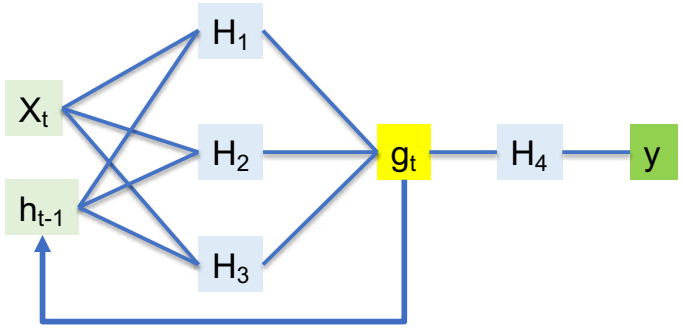
$$\mathbf{o}_{(t)} = \sigma(\mathbf{W}_{xo}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{ho}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_o)$$

$$\mathbf{g}_{(t)} = \tanh(\mathbf{W}_{xg}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_g)$$

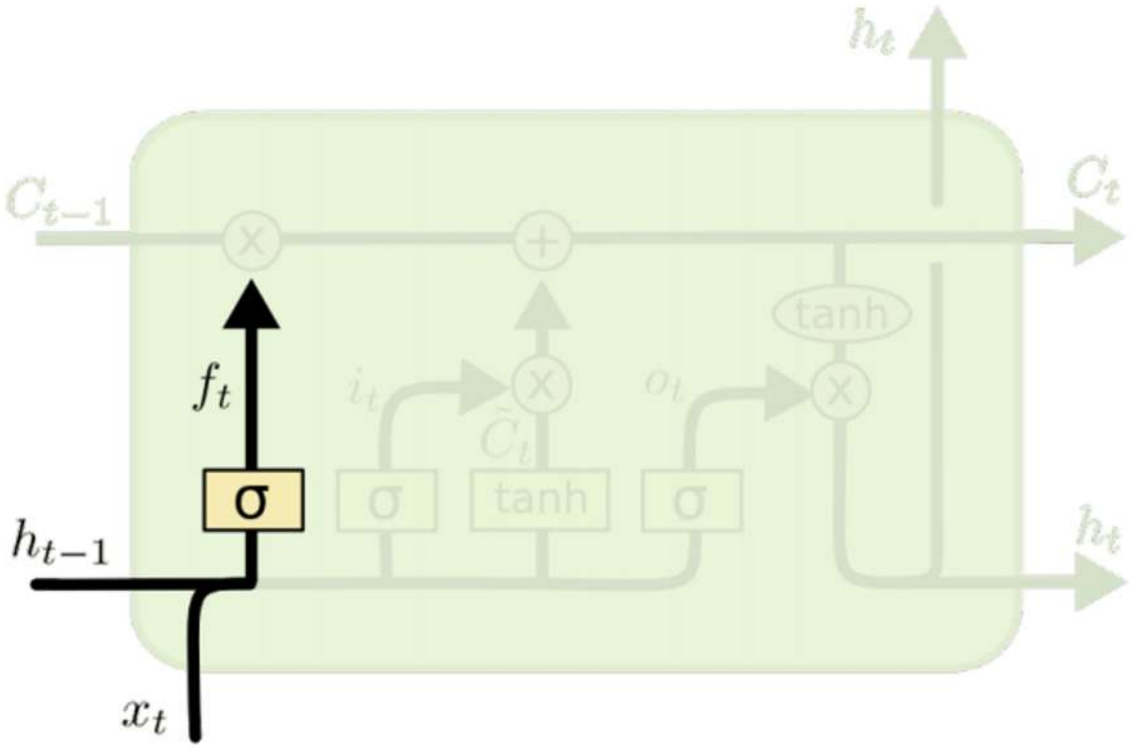
$$\mathbf{c}_{(t)} = \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)}$$

$$\mathbf{y}_{(t)} = \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh(\mathbf{c}_{(t)})$$

Inside LSTM: (2) Forget Gate



Should we continue to remember this “bit” of information or not?



There are 3 gates with 2 outputs.

$$\mathbf{i}_{(t)} = \sigma(\mathbf{W}_{xi}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hi}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_i)$$

$$\mathbf{f}_{(t)} = \sigma(\mathbf{W}_{xf}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hf}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_f)$$

$$\mathbf{o}_{(t)} = \sigma(\mathbf{W}_{xo}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{ho}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_o)$$

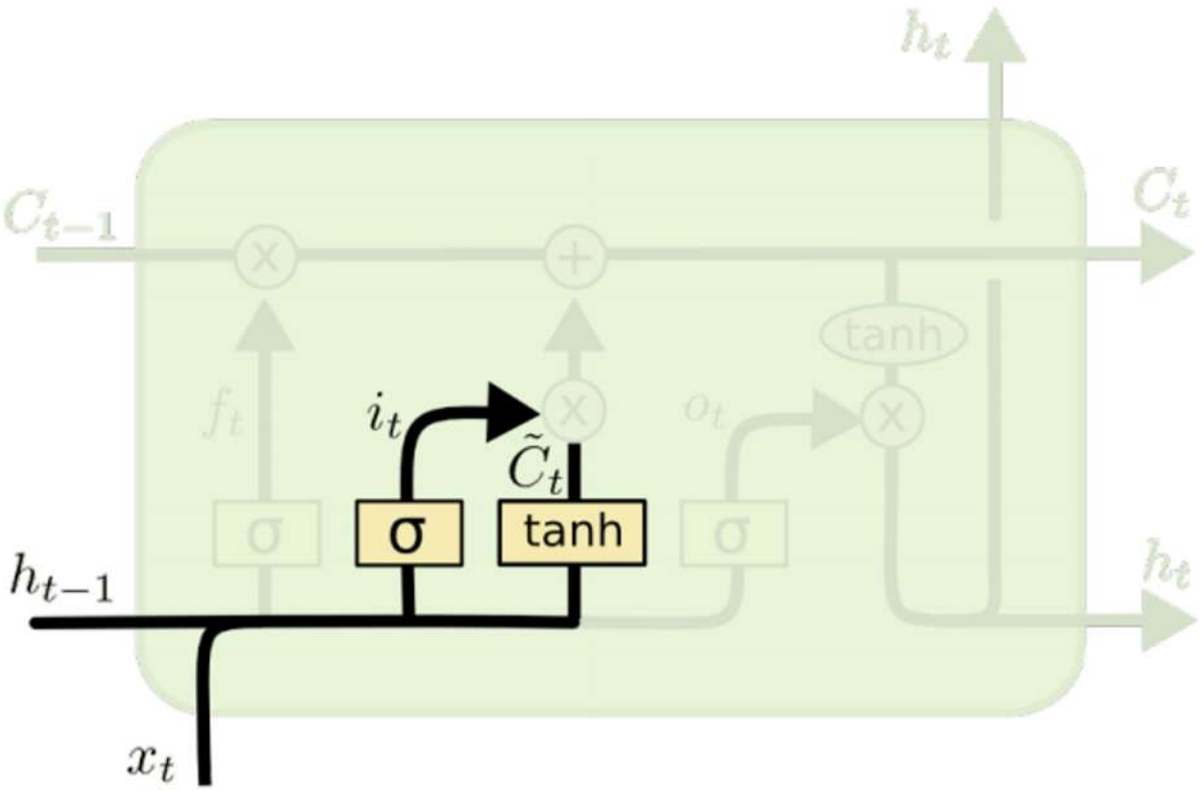
$$\mathbf{g}_{(t)} = \tanh(\mathbf{W}_{xg}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_g)$$

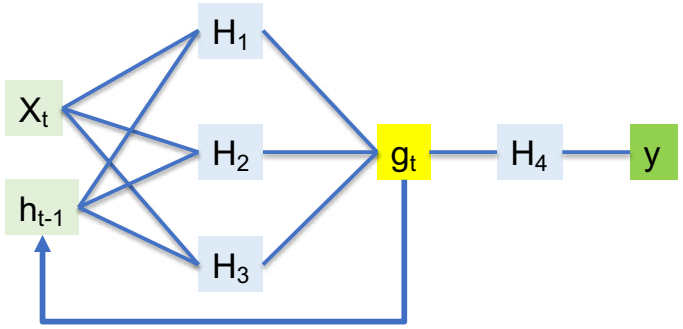
$$\mathbf{c}_{(t)} = \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)}$$

$$\mathbf{y}_{(t)} = \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh(\mathbf{c}_{(t)})$$

Inside LSTM: (3) Input Gate

Should we update this “bit” of information or not?
If yes, then what should we remember?





There are 3 gates with 2 outputs.

$$\mathbf{i}_{(t)} = \sigma(\mathbf{W}_{xi}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hi}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_i)$$

$$\mathbf{f}_{(t)} = \sigma(\mathbf{W}_{xf}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hf}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_f)$$

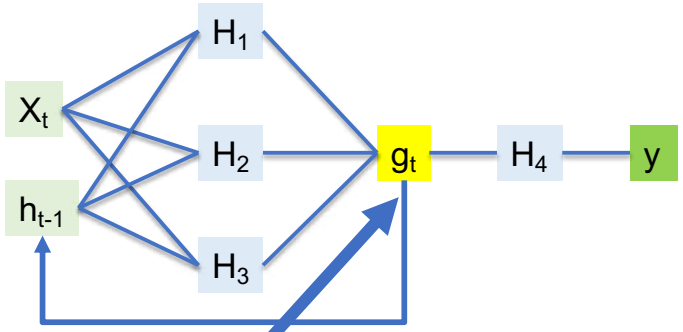
$$\mathbf{o}_{(t)} = \sigma(\mathbf{W}_{xo}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{ho}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_o)$$

$$\mathbf{g}_{(t)} = \tanh(\mathbf{W}_{xg}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_g)$$

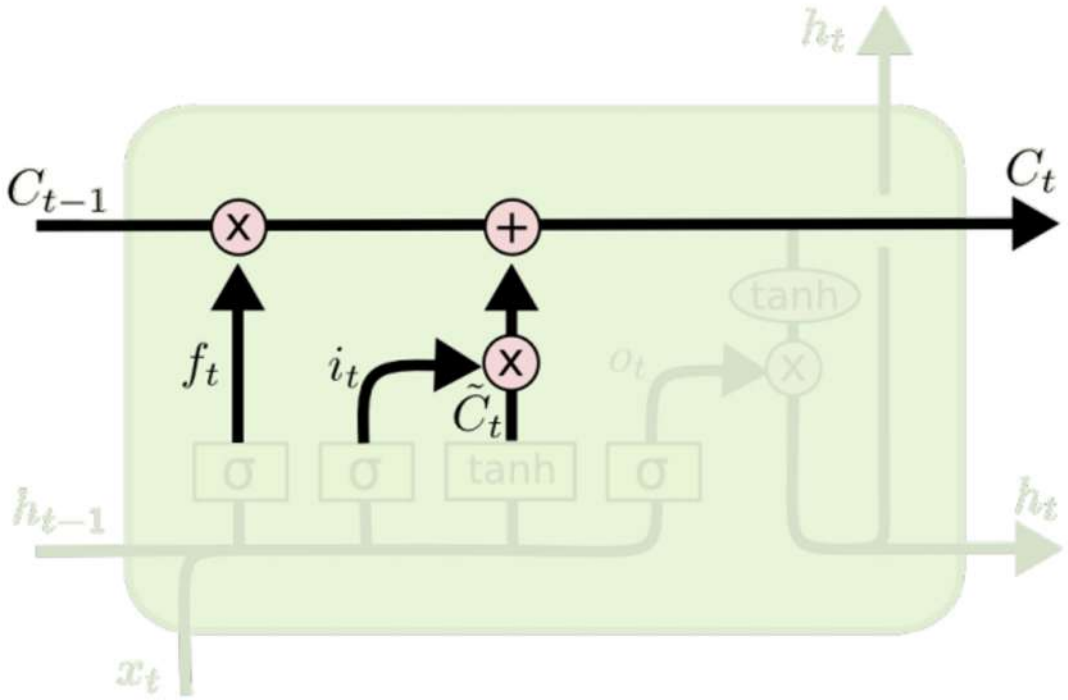
$$\mathbf{c}_{(t)} = \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)}$$

$$\mathbf{y}_{(t)} = \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh(\mathbf{c}_{(t)})$$

Inside LSTM: (4) Memory Update



Forget what needs to be forgotten + memorize what needs to be remembered



There are 3 gates with 2 outputs.

$$\mathbf{i}_{(t)} = \sigma(\mathbf{W}_{xi}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hi}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_i)$$

$$\mathbf{f}_{(t)} = \sigma(\mathbf{W}_{xf}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hf}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_f)$$

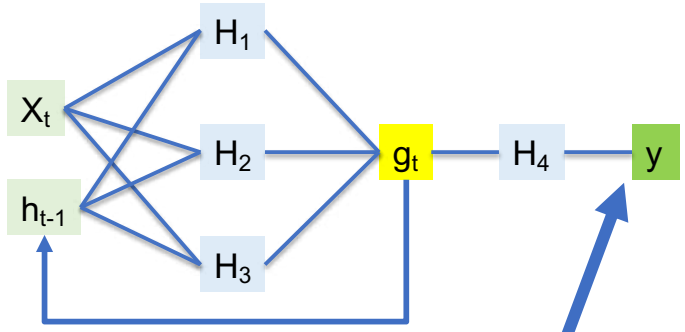
$$\mathbf{o}_{(t)} = \sigma(\mathbf{W}_{xo}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{ho}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_o)$$

$$\mathbf{g}_{(t)} = \tanh(\mathbf{W}_{xg}^{T} \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}^{T} \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_g)$$

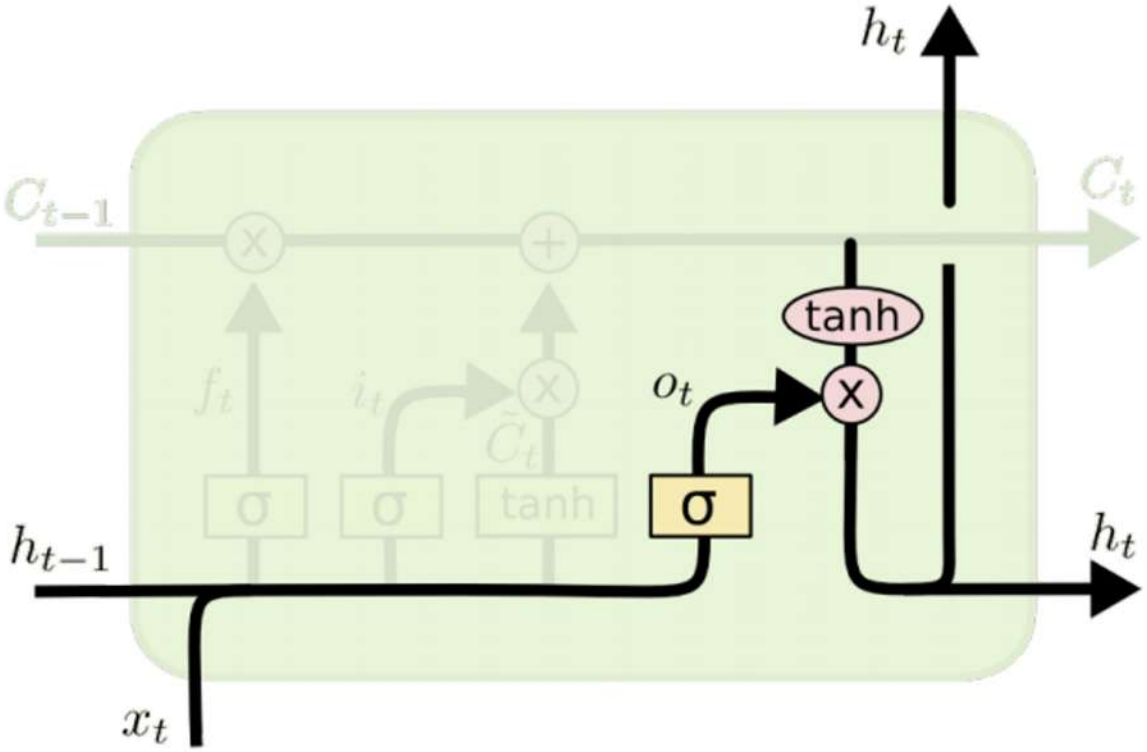
$$\mathbf{c}_{(t)} = \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)}$$

$$\mathbf{y}_{(t)} = \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh(\mathbf{c}_{(t)})$$

Inside LSTM: Output Gate



Should we output this bit of information (e.g., to “deeper” LSTM layers)?



There are 3 gates with 2 outputs.

$$\mathbf{i}_{(t)} = \sigma(\mathbf{W}_{xi}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hi}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_i)$$

$$\mathbf{f}_{(t)} = \sigma(\mathbf{W}_{xf}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hf}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_f)$$

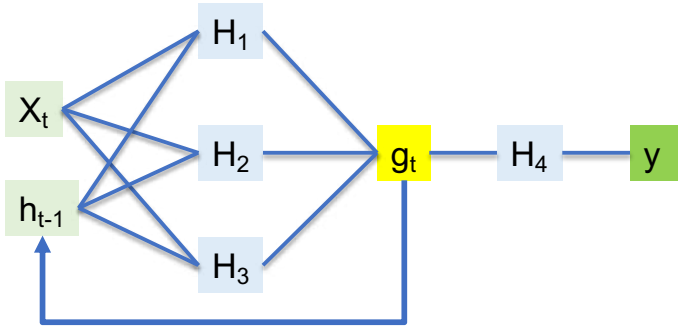
$$\mathbf{o}_{(t)} = \sigma(\mathbf{W}_{xo}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{ho}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_o)$$

$$\mathbf{g}_{(t)} = \tanh(\mathbf{W}_{xg}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_g)$$

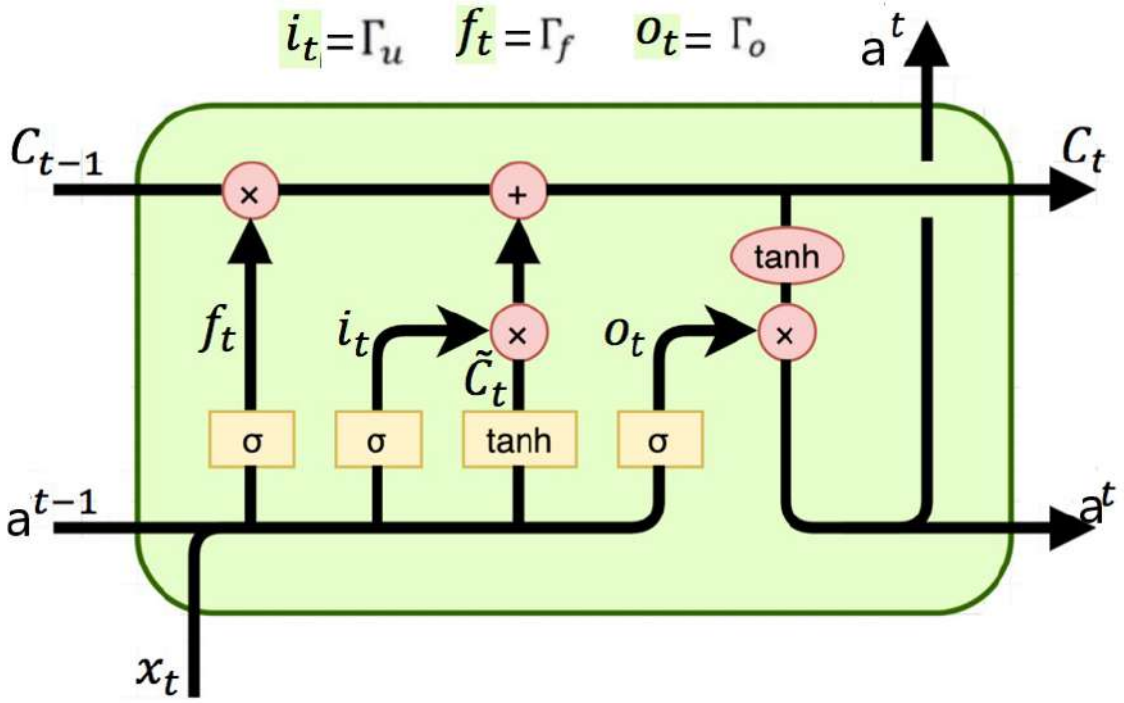
$$\mathbf{c}_{(t)} = \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)}$$

$$\mathbf{y}_{(t)} = \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh(\mathbf{c}_{(t)})$$

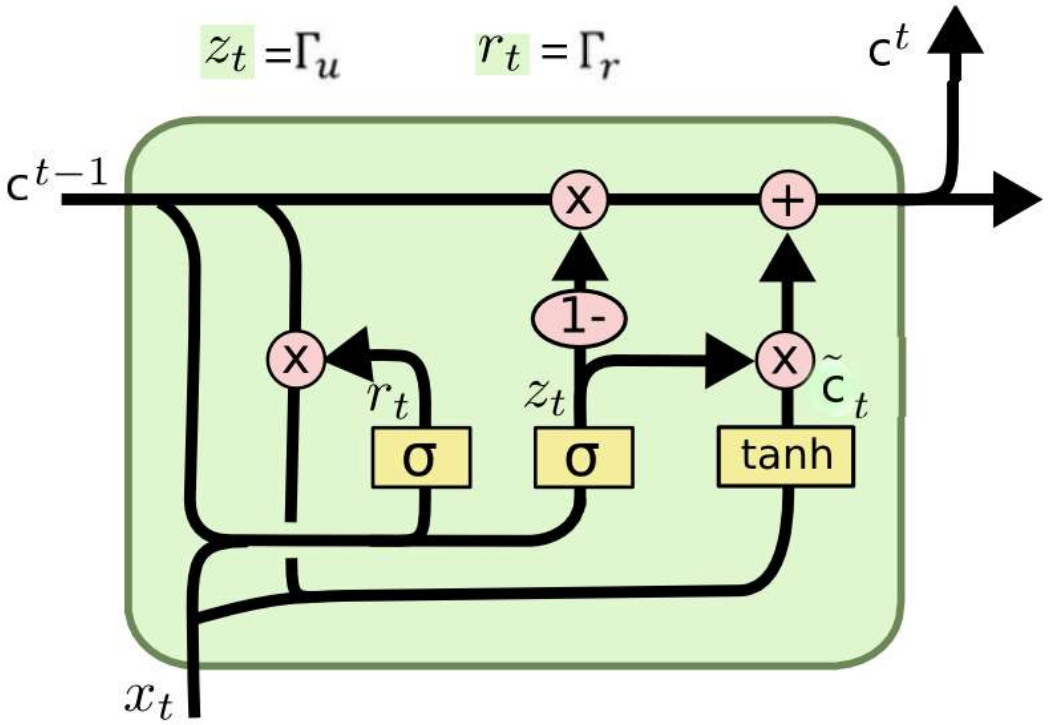
LSTM vs. Gated Recurrent Units (GRU)



There are 3 gates with 2 outputs.
3 inputs (x(t), C(t-1), a(t-1))

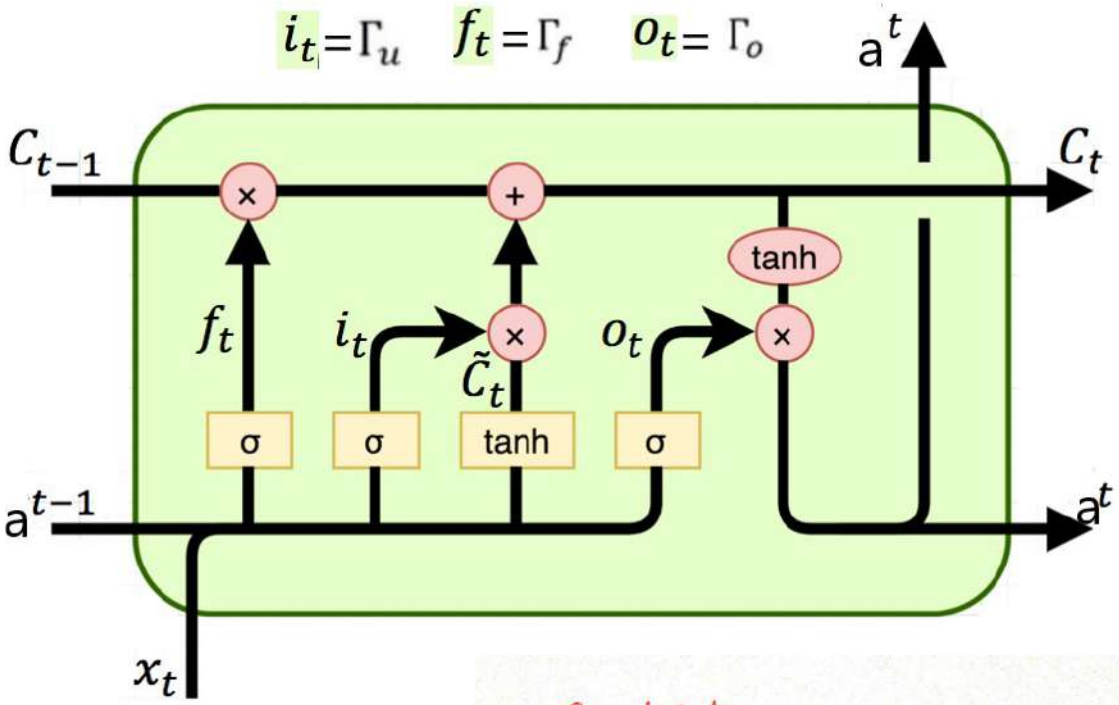


- There are 2 gates (update & reset gates)
- 1 output
- 2 inputs (x(t), C(t-1), ~~a(t-1)~~)



https://medium.com/analytics-vidhya/rnn-vs-gru-vs-lstm-863b0b7b1573

There are 3 gates with 2 outputs.



$$\mathbf{i}_{(t)} = \sigma(\mathbf{W}_{xi}{}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hi}{}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_i)$$
$$\mathbf{f}_{(t)} = \sigma(\mathbf{W}_{xf}{}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hf}{}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_f)$$
$$\mathbf{o}_{(t)} = \sigma(\mathbf{W}_{xo}{}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{ho}{}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_o)$$
$$\mathbf{g}_{(t)} = \tanh(\mathbf{W}_{xg}{}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}{}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_g)$$
$$\mathbf{c}_{(t)} = \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)}$$
$$\mathbf{y}_{(t)} = \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh(\mathbf{c}_{(t)})$$

g(t) Candidate cell → $\tilde{c}^{\langle t \rangle} = \tanh(W_c[a^{\langle t-1 \rangle} + x^{\langle t \rangle}] + b_c)$

i(t) Update / Input gate → $\Gamma_u = \sigma(W_u[a^{\langle t-1 \rangle} + x^{\langle t \rangle}] + b_u)$

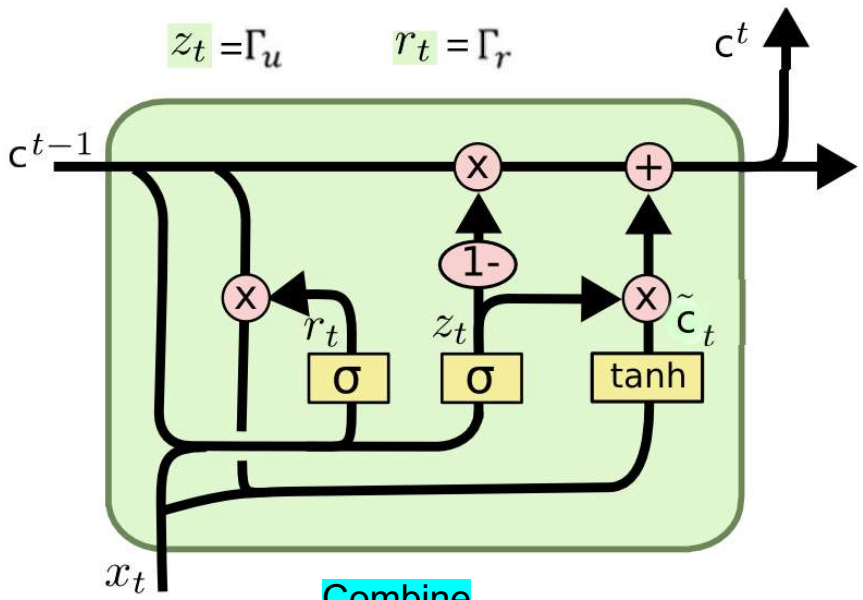
f(t) Forget gate → $\Gamma_f = \sigma(W_f[a^{\langle t-1 \rangle} + x^{\langle t \rangle}] + b_f)$

o(t) Output gate → $\Gamma_o = \sigma(W_o[a^{\langle t-1 \rangle} + x^{\langle t \rangle}] + b_o)$

C(t) Cell state → $c^{\langle t \rangle} = \Gamma_u * \tilde{c}^{\langle t \rangle} + \Gamma_f * c^{\langle t-1 \rangle}$

y(t) Activation → $a^{\langle t \rangle} = \Gamma_o * \tanh c^{\langle t \rangle}$

- There are 2 gates (update & reset gates)
- 1 output
- 2 inputs (x(t), C(t-1), ~~a(t-1)~~)



Combine
- Previous → how much to reset C(t-1)
- Current [x(t)]

g(t) Candidate cell → $\tilde{c}^{\langle t \rangle} = \tanh(W_c[\Gamma_r * c^{\langle t-1 \rangle} + x^{\langle t \rangle}] + b_c)$

i(t) Update gate → $\Gamma_u = \sigma(W_u[c^{\langle t-1 \rangle} + x^{\langle t \rangle}] + b_u)$

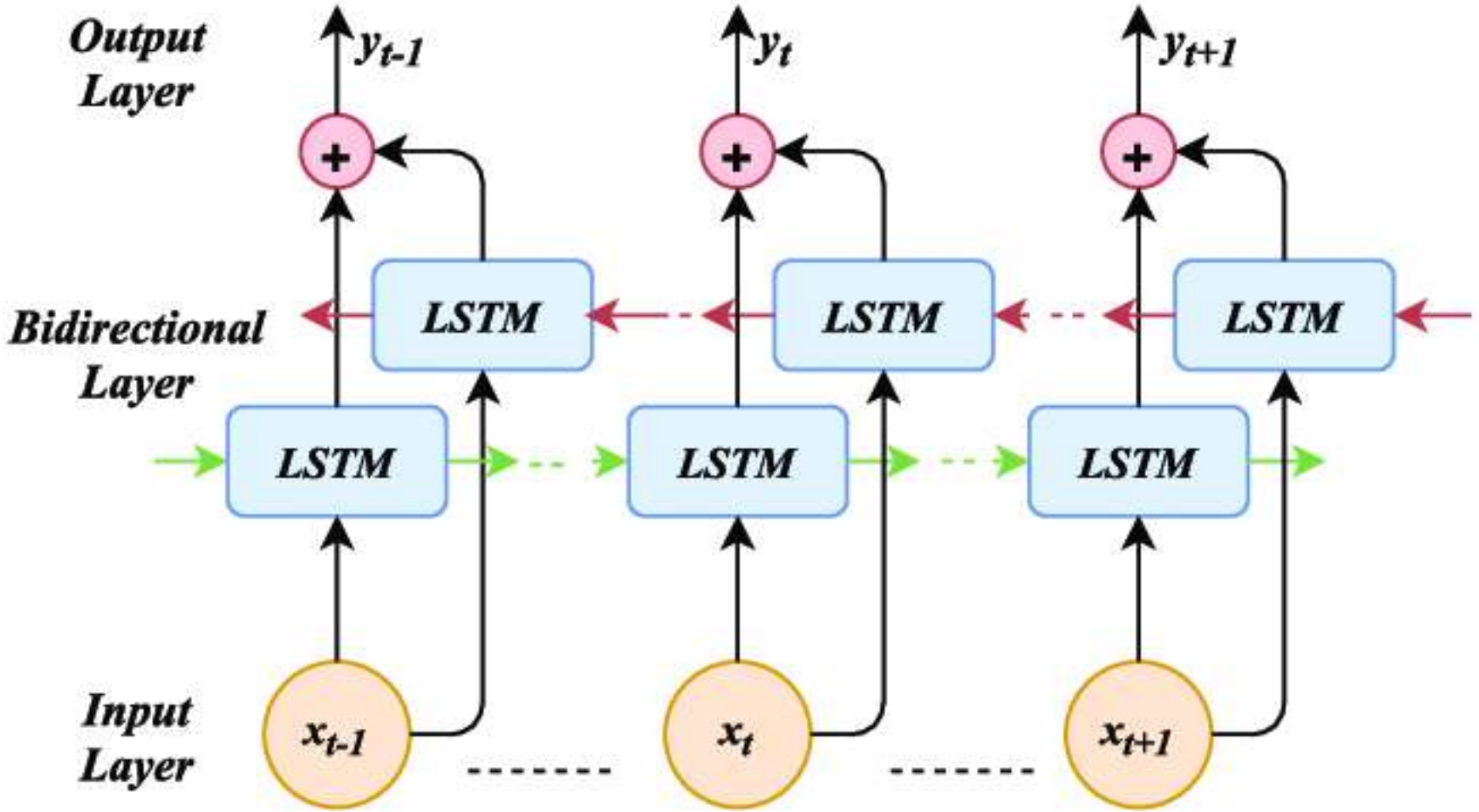
f(t) Reset gate → $\Gamma_r = \sigma(W_r[c^{\langle t-1 \rangle} + x^{\langle t \rangle}] + b_r)$

C(t) Cell state → $c^{\langle t \rangle} = \Gamma_u * \tilde{c}^{\langle t \rangle} + (1 - \Gamma_u) * c^{\langle t-1 \rangle}$

y(t) Activation / Output → $a^{\langle t \rangle} = c^{\langle t \rangle}$

Update g(t) & C(t-1)

Bidirectional LSTM

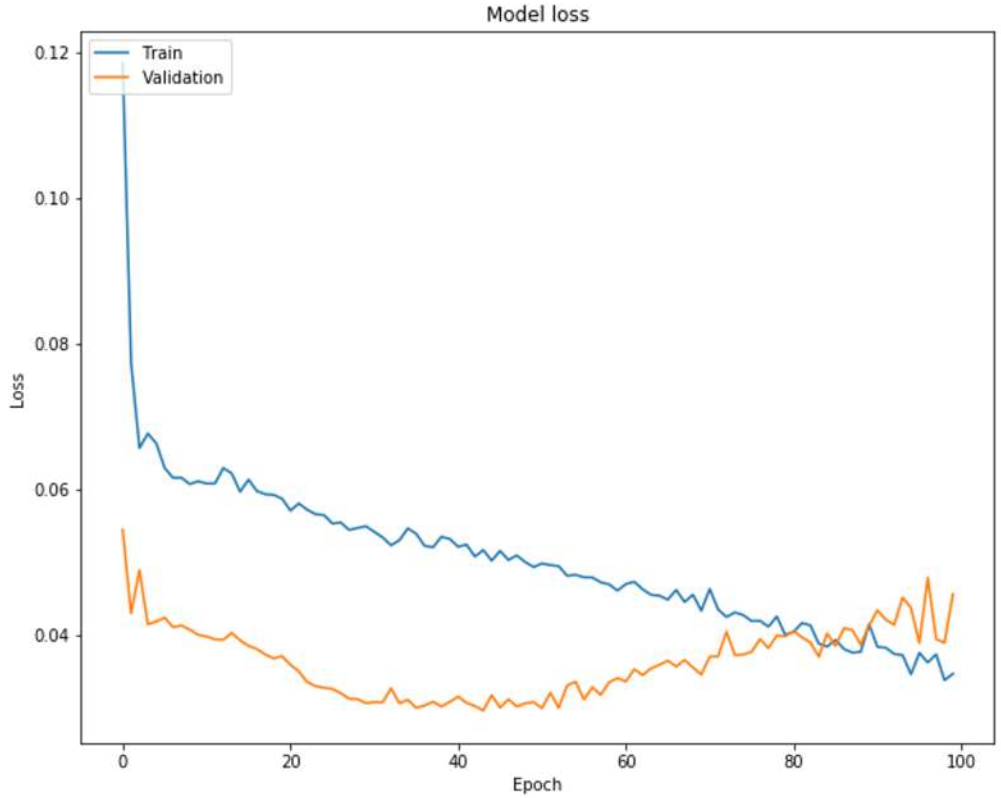


LSTM Parameters Tuning

- Epochs
- Learning Rate
- LSTM Parameters
  - Number of hidden Unit
  - Number of layers
- Loss function

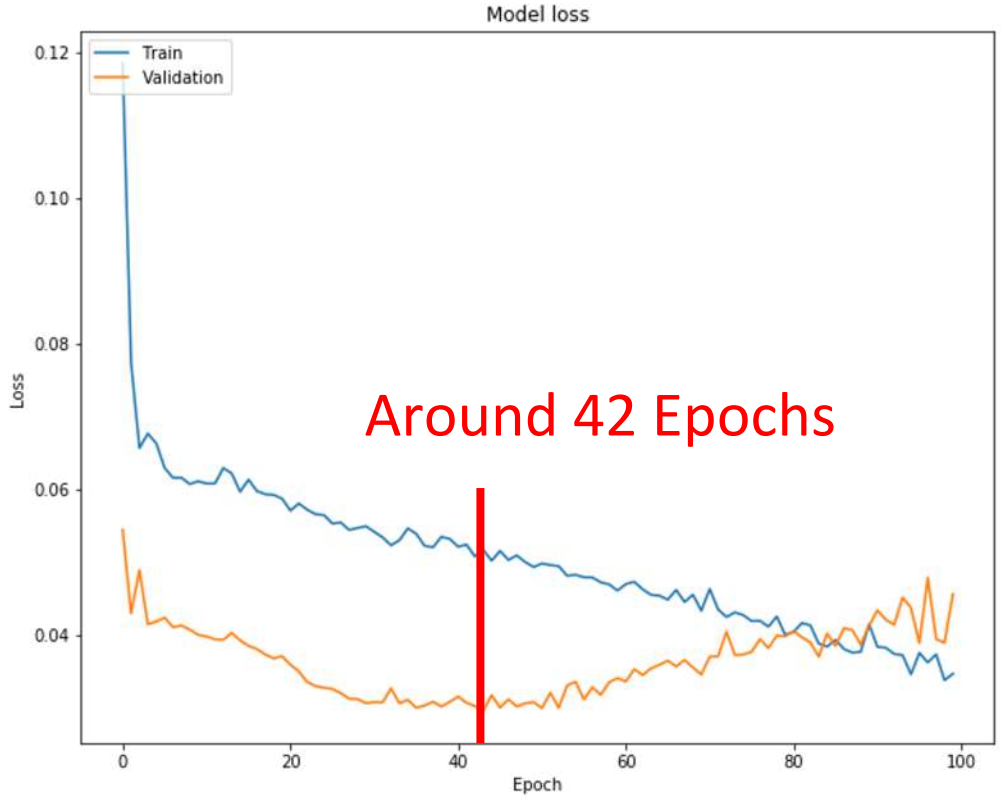
LSTM Parameters Tuning: Epoch

- The single time you see all examples in the dataset.
- Choosing by Plotting Train, Validation loss



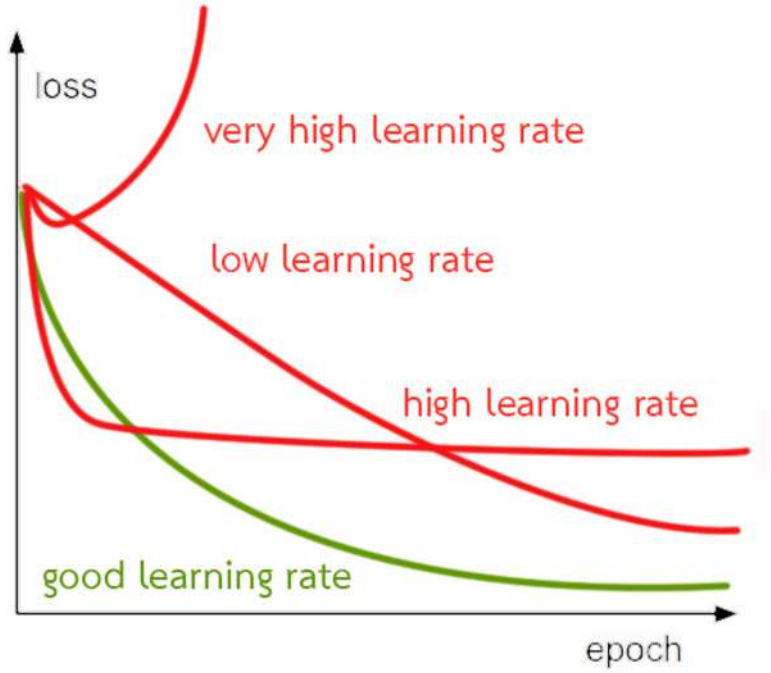
LSTM Parameters Tuning: Epoch (cont.)

- The single time you see all examples in the dataset.
- Choosing by Plotting Train, Validation loss



LSTM Parameters Tuning: Learning Rate

- The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error each time the model weights are updated.

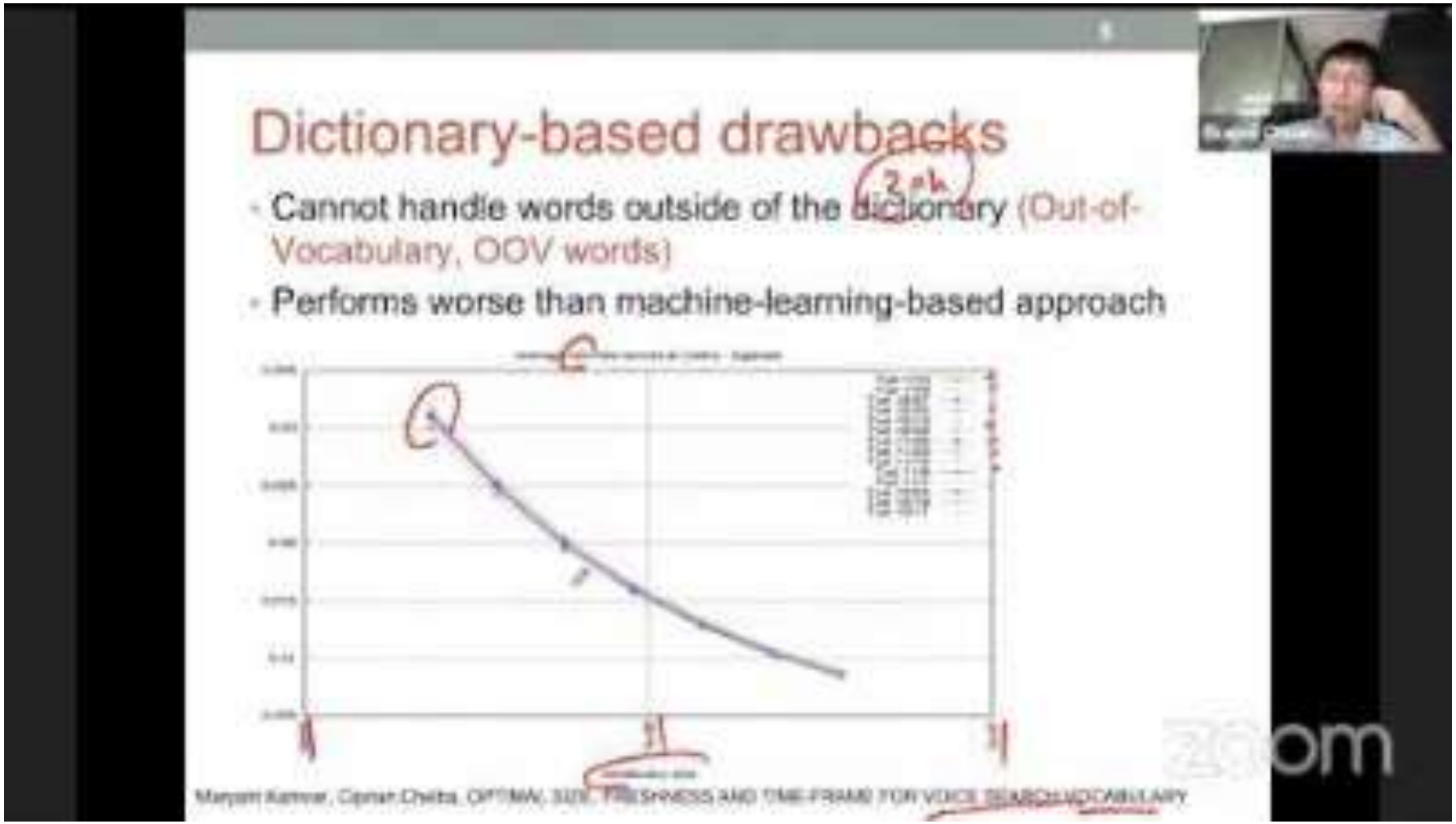


LSTM Parameters Tuning: Loss Function

- We are currently use Mean Square Error

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \tilde{y}_i)^2$$

NLP Lecture2 (2020s2) by Aj.Ekapol
https://youtu.be/WxiO3wvKhCE



Language Technologies

NVIDIA AI TECHNOLOGY CENTER (NVAITC)

2019

NVIDIA AI Tech Centre

NVAITC COLLABORATORS

Singapore

NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

SINGAPORE UNIVERSITY OF TECHNOLOGY AND DESIGN

SGINNOVATE

INFOCOMM MEDIA DEVELOPMENT AUTHORITY

TEMASEK POLYTECHNIC

NUS National University of Singapore

NATIONAL RESEARCH FOUNDATION PRIME MINISTER'S OFFICE SINGAPORE

EDB singapore

Taiwan

National Taiwan University

TSING HUA UNIVERSITY

南科AIROBOT自造基地

NCHC

China

The University of Nottingham

UNITED KINGDOM · CHINA · MALAYSIA

Hong Kong

香港科技大學 THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

HKSTP

ASTRI

LSGM

Europe

UNIVERSITÉ DU LUXEMBOURG

CSC

epcc

University of Reading

cini consorzio interuniversitario nazionale per l'informatica

Hartree Centre

Science & Technology Facilities Council

Thailand

Australia

MONASH University

Indonesia

epsindo

PT. EPSINDO PRIMA SINERGI

AI Acceleration Advisory Board

India

IIT Hyderabad

NVIDIA AI Tech Centre

#AI #GTC20

Intelligent End-to-End AI Chatbot with Audio-Driven Facial Animation

39,101 views • May 16, 2020

1

NVIDIA
689K subscribers

During the GTC 2020 virtual keynote, NVIDIA CEO and founder Jensen Huang interacted with Misty, a conversational AI weather chatbot, to demonstrate an end-to-end pipeline to create an AI driven 3D digital avatar. The NVIDIA Jarvis multimodal application framework includes pre-

SHOW MORE

NVIDIA
NEMO

ASR

NLP

TTS

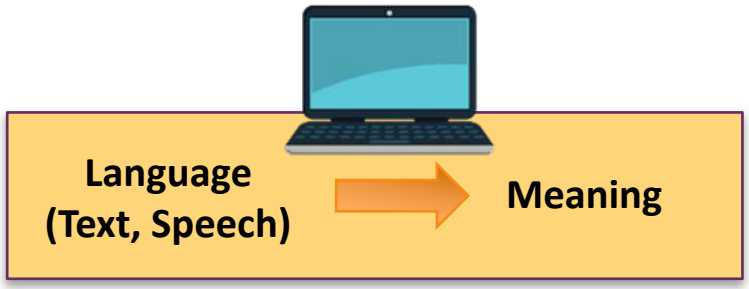
Outlines

- Part1: Overview of NLP & Its Tasks
- Part2: Word Segmentation
- Part3: Text Classification

Part 1: Overview of NLP & Its Tasks

Natural Language Processing (NLP)

NLP is a subfield in AI, where the goal is to bridge the gap between how people communicate and what machines understand in order to perform useful tasks, e.g. making appointments, buying things, question answering, etc.



* Not just string matching

Level of understanding in NLP

https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_natural_language_processing.htm

Lexical Analysis:

Text→ Paragraphs, Sentences, and Words

Syntactic Analysis (Parsing):

Grammar/Relationship between words

Semantic Analysis:

Exact meaning of the sentence

Discourse Integration:

Meaning of the sentence **based on the previous sentence (pronouns)**

Pragmatic Analysis:

Actual Meaning based on **the context** and real-world knowledge

Discourse

Semantics

Syntax: Constituents

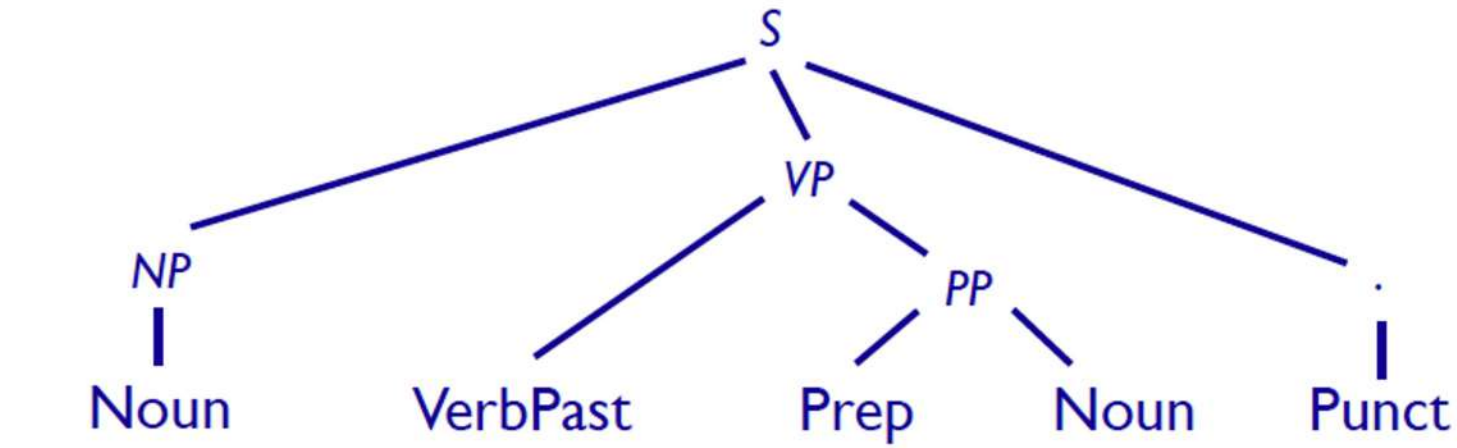
Syntax: Part of Speech

Words

Morphology

Characters

CommunicationEvent(e)
Agent(e, Alice)
Recipient(e, Bob)
SpeakerContext(s)
TemporalBefore(e, s)



Alice talked to Bob .

talk -ed [VerbPast]

Alice talked to Bob.

Tokenization

- Input:ขสมก.เล็งจัดหารถ
- Output: ขสมก.,เล็ง,จัดหา,รถ

Part of Speech tagging

- Input:[ขสมก.,เล็ง,จัดหา,รถ]
- Output:[(ขสมก.,NR),(เล็ง,VV),(จัดหา,VV),(รถ,NN)]

PENN Part Of Speech Tags

- NR – proper noun
- VV - Main verbs in clauses, verb-form
- NN – Non-proper noun

Ref: BEST2010 dataset

Named Entity Tags

- PER –Person
- LOC – Location
- ORG – Organization
- O – Other

NER

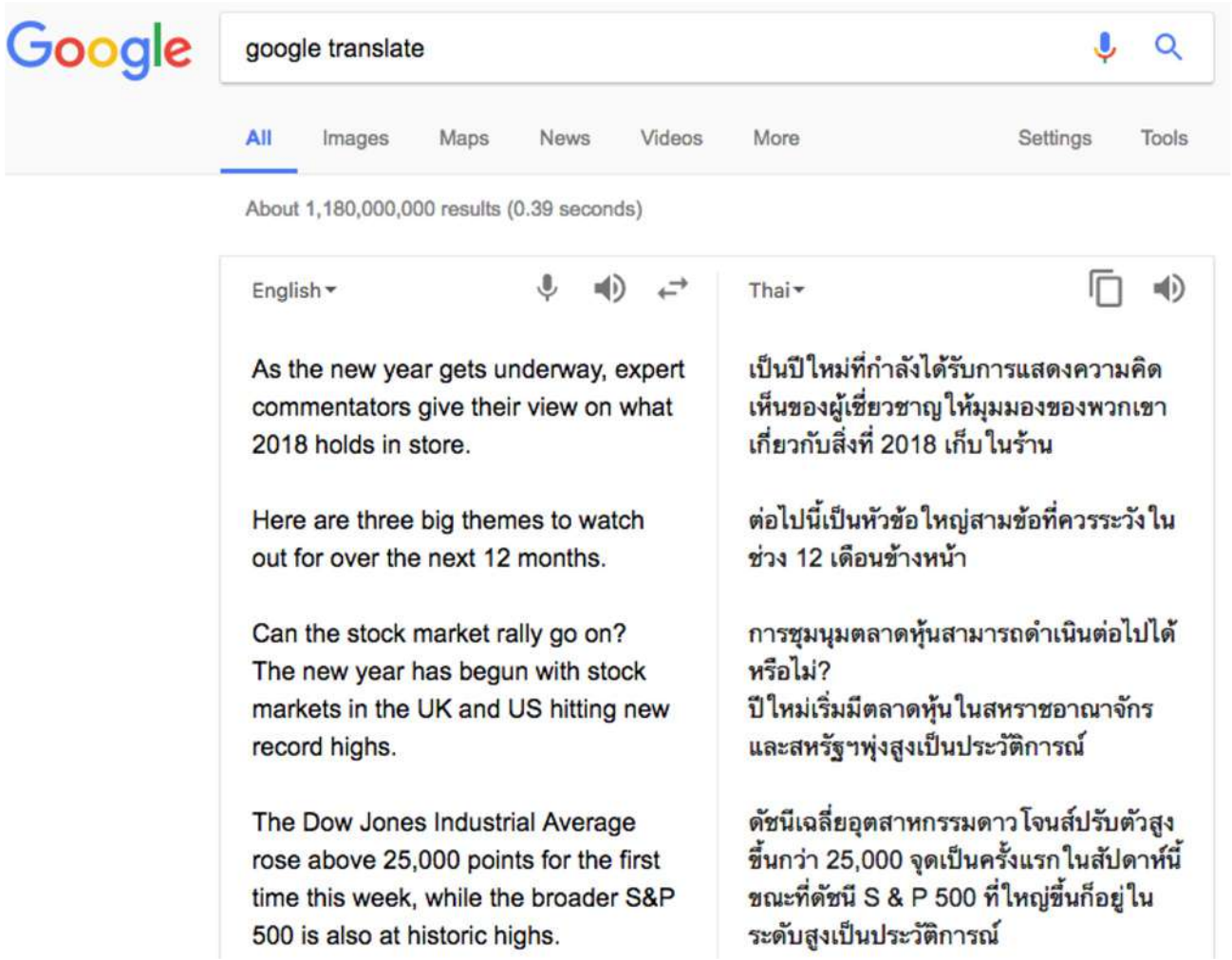
- Input:[(ขสมก.,NR),(เล็ง,VV),(จัดหา,VV),(รถ,NN)]
- Output:[(ขสมก.,NR,ORG),(เล็ง,VV,O),(จัดหา,VV,O),(รถ,NN,O)]

Application

- Word Cloud (Named Entity Only)
- Text Classification
- Etc.

Downstream task

Machine Translation (MT)



Markets, Brexit and Bitcoin: 2018's themes

By Chris Johnston
Business reporter

5 January 2018 Share

As the new year gets underway, expert commentators give their view on what 2018 holds in store.

http://www.bbc.com/news/business-42581934

Question Answering (QA)

IBM Watson wowed the tech industry and a corner of U.S. pop culture with its 2011 win against two of Jeopardy's greatest champions. Here's how IBM pulled it off and a look at what Watson's real career is going to be.

https://www.techrepublic.com/article/ibm-watson-the-inside-story-of-how-the-jeopardy-winning-supercomputer-was-born-and-what-it-wants-to-do-next/

Ref: Prof. Regina Barzilay, NLP @MIT

Chatbots

Number of Websites with a Chatbot by Country

Country	Number of Chatbot Sites	Population	Chatbots/1000 people
USA	15,515	325,000,000	0.0477
Australia	1009	24,000,000	0.0420
UK	2667	65,000,000	0.0410
Netherlands	684	17,000,000	0.0402
Canada	1265	36,000,000	0.0351
France	569	66,000,000	0.0086
Germany	476	82,000,000	0.0058
Brazil	399	207,000,000	0.0019
India	592	1,300,000,000	0.0005

Sites Using Chatbots Gathered from NerdyData

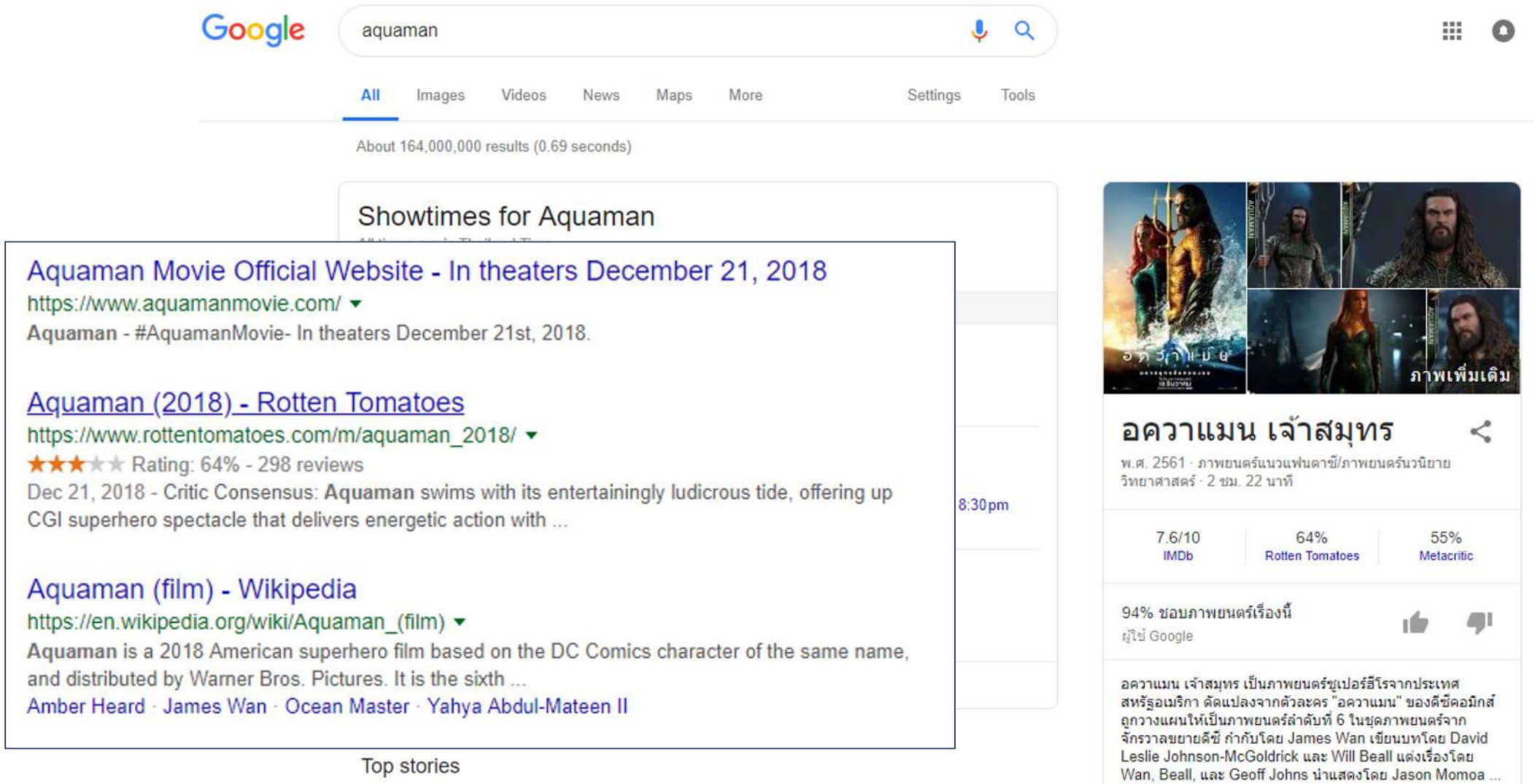
Boomtown

https://research.aimultiple.com/chatbot-stats/

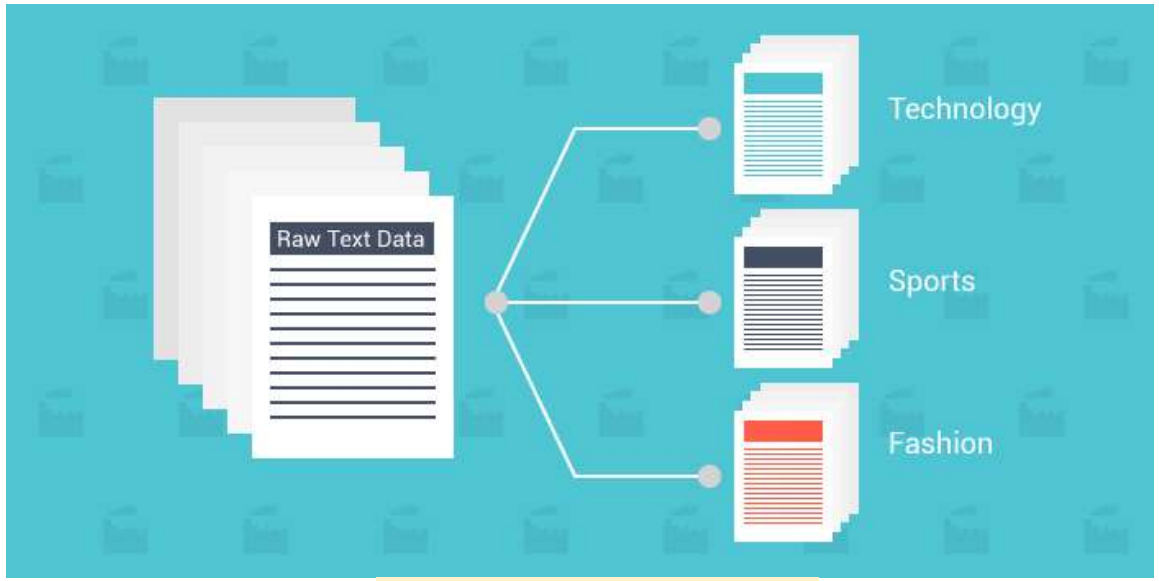
Botnoi Consulting

https://botnoigroup.com/

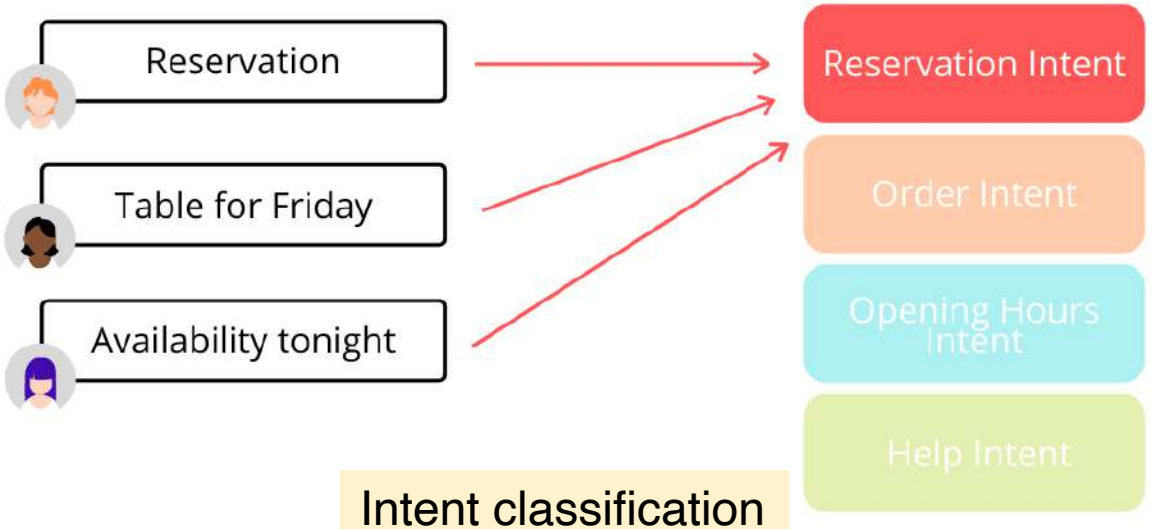
Search/Summarization



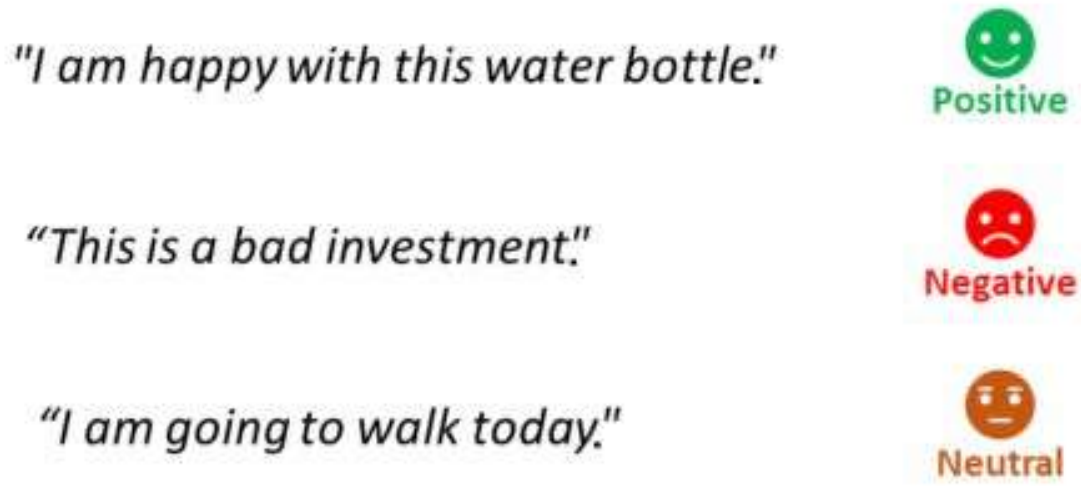
Text Classification



Text categorization



Intent classification

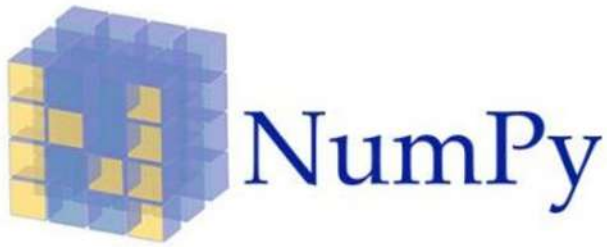


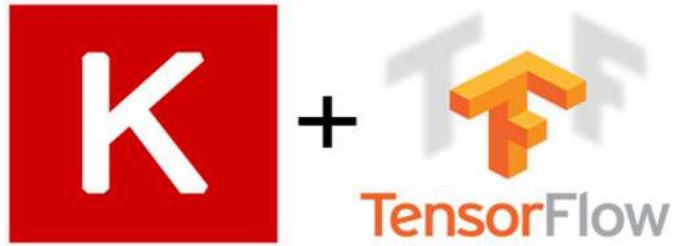
Sentiment classification

Demo


python™


Google
colab


NumPy


+
TensorFlow


scikits
learn
machine learning in Python


PyTorch

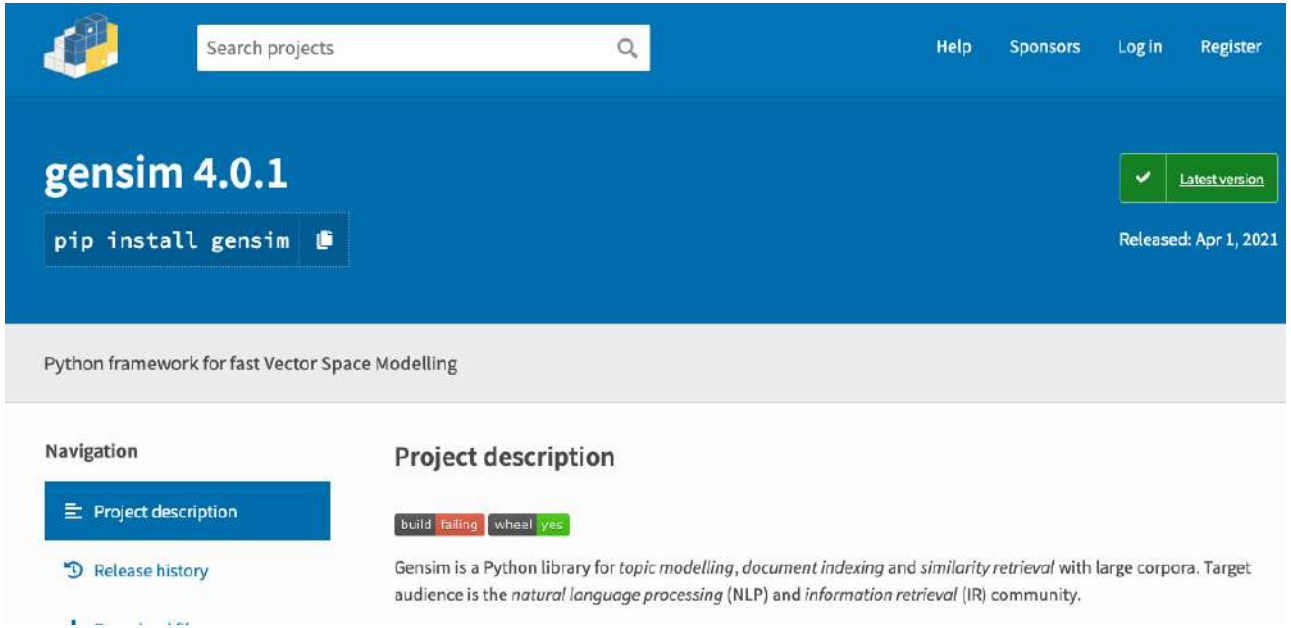
NLP Libraries

NLTK 3.5 documentation

NEXT | MODULES | INDEX

Natural Language Toolkit

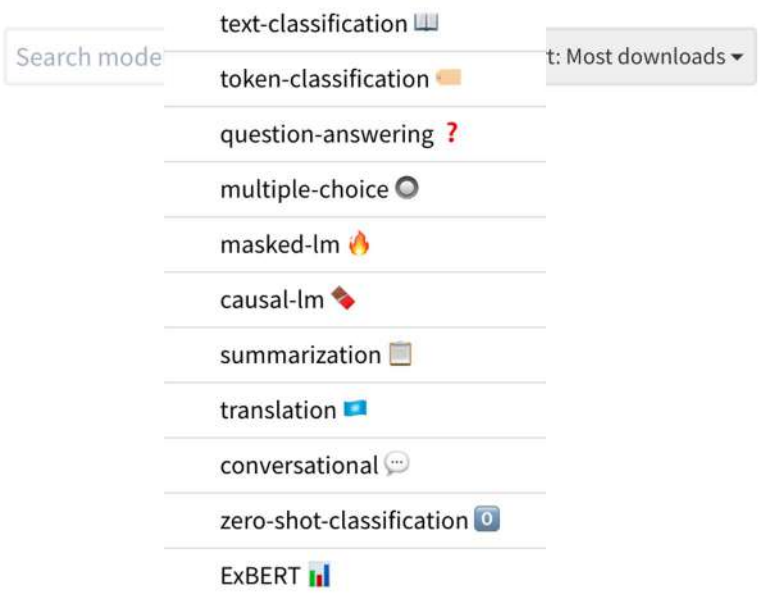
NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.



HUGGING FACE

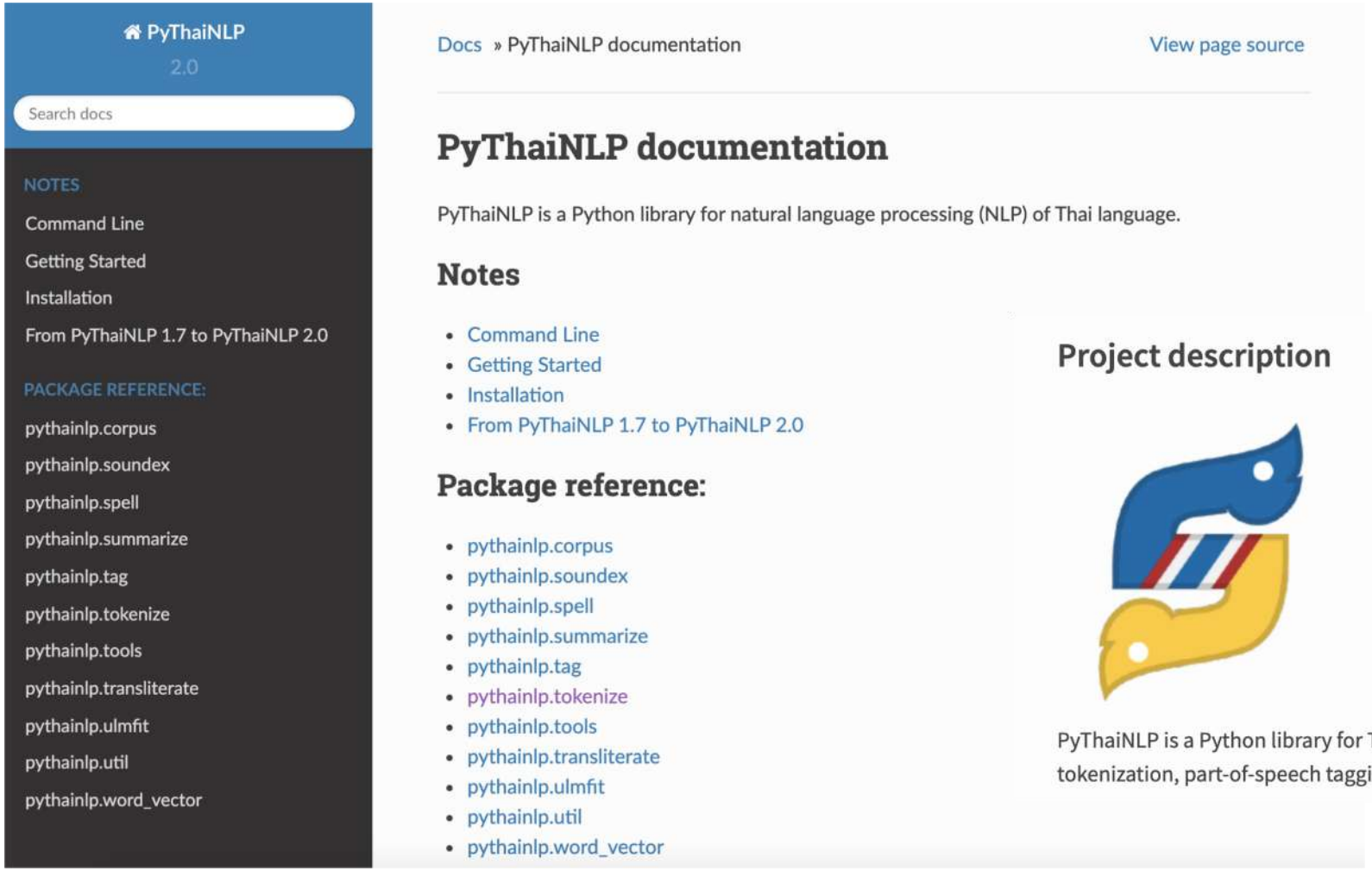
On a mission to solve NLP,
one commit at a time.

Star 39,335



NLP Libraries for Thai

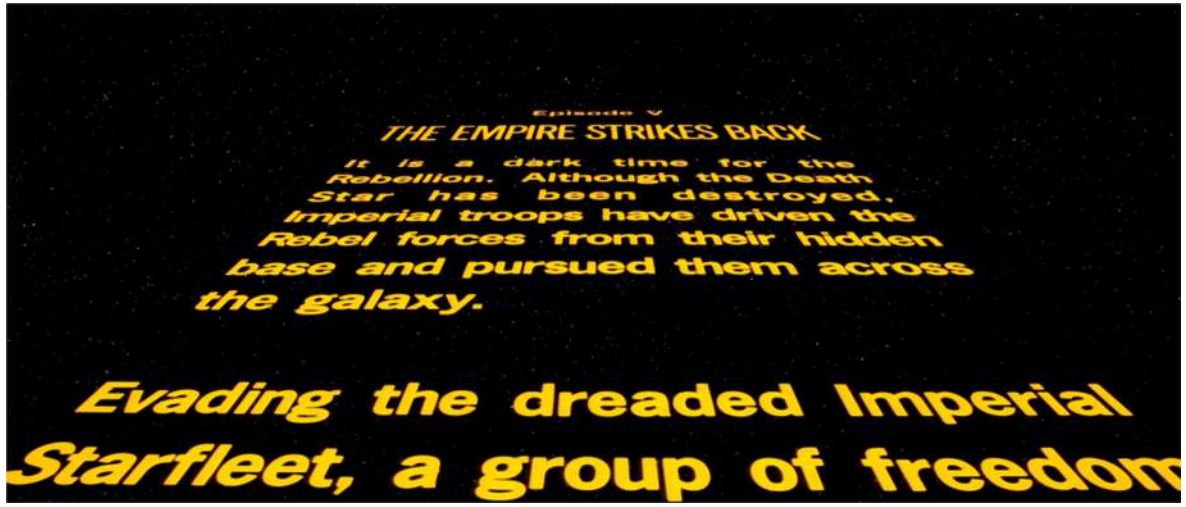
https://thainlp.org/pythainlp/docs/2.0/index.html



Part2: Word Segmentation

The need for segmentation

- Text as a stream of characters



- We need a way to understand the meaning of text
  - Break into words (assign meaning to word) ← Tokenization
  - Break into sentences (put word meanings back to sentence meaning)

http://starwars.wikia.com/wiki/Opening_crawl

Tokenization - Thai

- Thai has no space between words
- Thai has no clear sentence boundaries

- เริ่มจากชนวนของสงครามแรกสุด สมาพันธ์การค้า ทำการปิดล้อม-รุกรานดาวนาบู ส่งผลต่อมาขยายเป็นสงครามโคลนอันมีฝ่ายแบ่งแยกดินแดนเป็นผู้ชักใยสงคราม หมายโค่นล้มฝ่ายสาธารณรัฐ ขนานไปกับเรื่องราวพัฒนาการของของเด็กน้อยคนหนึ่งผู้มีพลังสถิตแรงมาก ชาวดาวทะเลทรายทาทูอินนาม "อนาคิน สกายวอล์คเกอร์" ผู้ถูกคาดการณ์ว่าคือผู้ถูกเลือกในตำนานของเจได หลังจากสงครามยุทธการดาวนาบู อนาคิน ก็ได้รับฝึกฝนในวิถีเจได โดย อ.เจได "โอบีวัน" แต่ เมื่ออนาคินโตเป็นหนุ่มก็ดันแหกกฎเจไดโดยแอบมีความสัมพันธ์ชู้สาวลับๆกับ ราชินี "อมิดาลา" แห่งดาวนาบู จนเธอตั้งครรภ์ลูกแฝด ... และอนาคินก็ค่อยๆถูกกลืนเข้าสู่ด้านมืดของพลังจนกลายเป็นซิธลอร์ด ได้ฉายา "ดาร์ธ เวเดอร์" ภายใต้การโน้มน้าวชี้นำของ ซิธลอร์ดลึกลับนาม "ดาร์ธ ซีเดียส" ซึ่งเผยในตอนท้ายว่า ซีเดียส ไม่ใช่ใครที่ไหน แต่คือท่าน "พัลพาทีน" สมุหนายกผู้นำสูงสุดของฝ่ายสาธารณรัฐเสียเอง และแท้จริงก็ยังเป็นผู้นำลับชักใยฝ่ายแบ่งแยกดินแดนด้วยอีกต่างหาก ... สงครามจบลงที่ฝ่ายสาธารณรัฐพ่ายแพ้ล่มสลาย อมิดาลาก็ตายหลังคลอดลูกแฝด ทั้งเหล่าอัศวินเจไดก็ถูกฆ่ากวาดล้างสิ้นแบบไม่ทันตั้งตัว เหลือแต่ อ.โอบีวัน กับ อ.โยดา ต้องลี้ภัยหลบหนีซ่อนตัว โดยลูกแฝดของอนาคินได้ถูกส่งไปสู่ที่หลบซ่อนลับเช่นกัน ... และแล้ว พัลพาทีน ก็กินรวบทั้งกระดาน เปลี่ยนการปกครองจาก ระบอบสาธารณรัฐเดิมไปเป็น ระบอบเผด็จการจักรวรรดิซิธแทน ตั้งตนเป็นจักรพรรดิปกครองแกแลคซีทั้งปวง โดยมี ดาร์ธ เวเดอร์ เป็นขุนพลซิธลอร์ดเคียงข้างนับแต่นั้นมา

http://www.starwars.siligon.com/swcong.html

pythainlp.tokenize.word_tokenize(*text: str, custom_dict: marisa_trie.Trie = None, engine: str = 'newmm', keep_whitespace: bool = True*) → List[str] [source]

This function tokenizes running text into words.

Parameters

- **text** (*str*) – text to be tokenized
- **engine** (*str*) – name of the tokenizer to be used
- **custom_dict** (*marisa_trie.Trie*) – marisa dictionary trie
- **keep_whitespace** (*bool*) – True to keep whitespaces, a common mark for end of phrase in Thai. Otherwise, whitespaces are omitted.

Returns

list of words

Return type

list[str]

Options for engine

- *newmm* (default) - dictionary-based, Maximum Matching + Thai Character Cluster
- *longest* - dictionary-based, Longest Matching
- *deepcut* - wrapper for deepcut, language-model-based
- *icu* - wrapper for ICU (International Components for Unicode, using PyICU), dictionary-based
- *ulmfit* - for thai2fit

1) Longest Matching

- Scan a sentence from left to right
- Keep finding a word from the starting point, until no word matched (longest), then move to the next point
- Backtrack if current segmentation leads to an un-segmentable chunk

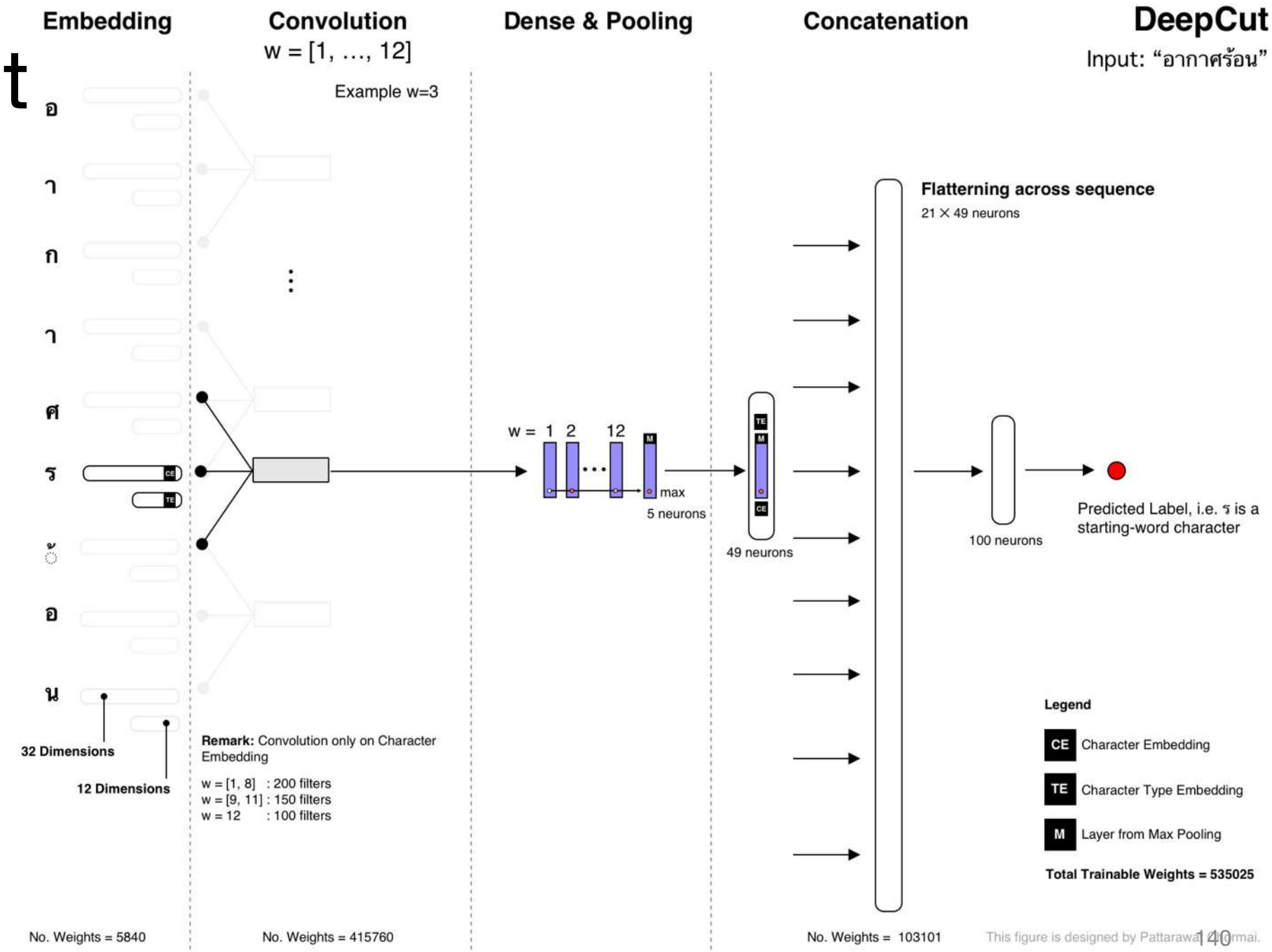
- ป้ายกลับรถ — Start scanning with “ป” as the starting point
- ป้ายกลับรถ — Keep scanning …
- ป้าย/กลับรถ — No more words start with “ป้าย”, move to the next point
- …
- ป้าย/กลับ/รถ

2) Maximal Matching

- Generate all possible segmentations
- Select the segmentations with the fewest words

Haruechaiyasak, Choochart, Sarawoot Kongyoung, and Matthew Dailey. "A comparative study on thai word segmentation approaches." Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, 2008. ECTI-CON 2008. 5th International Conference on. Vol. 1. IEEE, 2008.

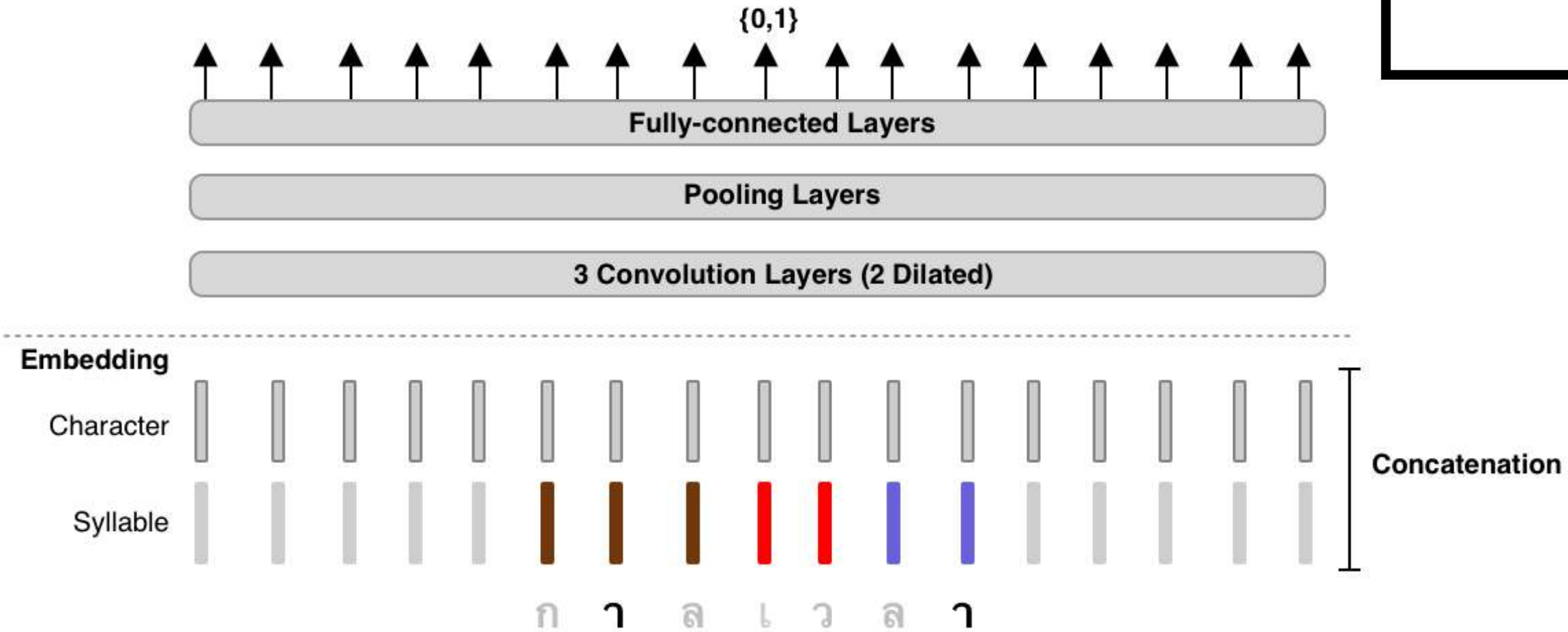
3) DeepCut



https://github.com/rkcosmos/deepcut

4) AttaCut

อัต|ตะ|คัด
ˈadə̀ˌkət



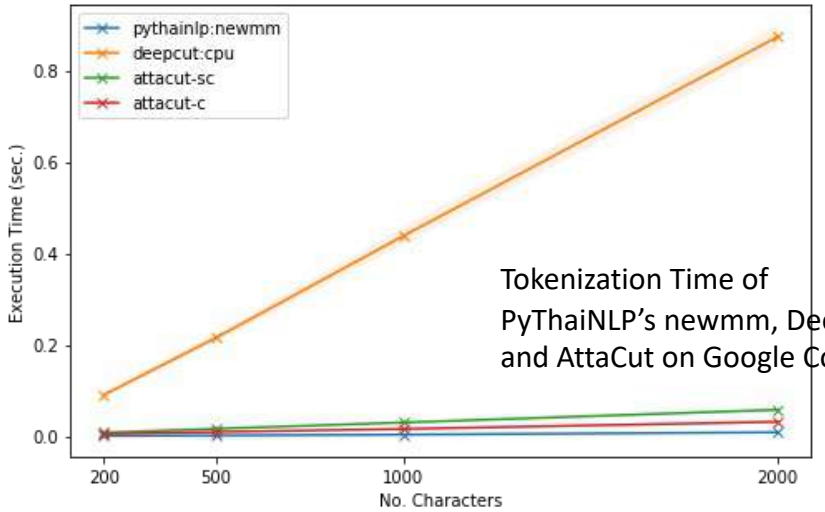
TL;DR: 3-Layer Dilated CNN on syllable and character features. It's **6x faster** than DeepCut (SOTA) while its WL-f1 on BEST is **91%**, only 2% lower.

https://github.com/PyThaiNLP/attacut

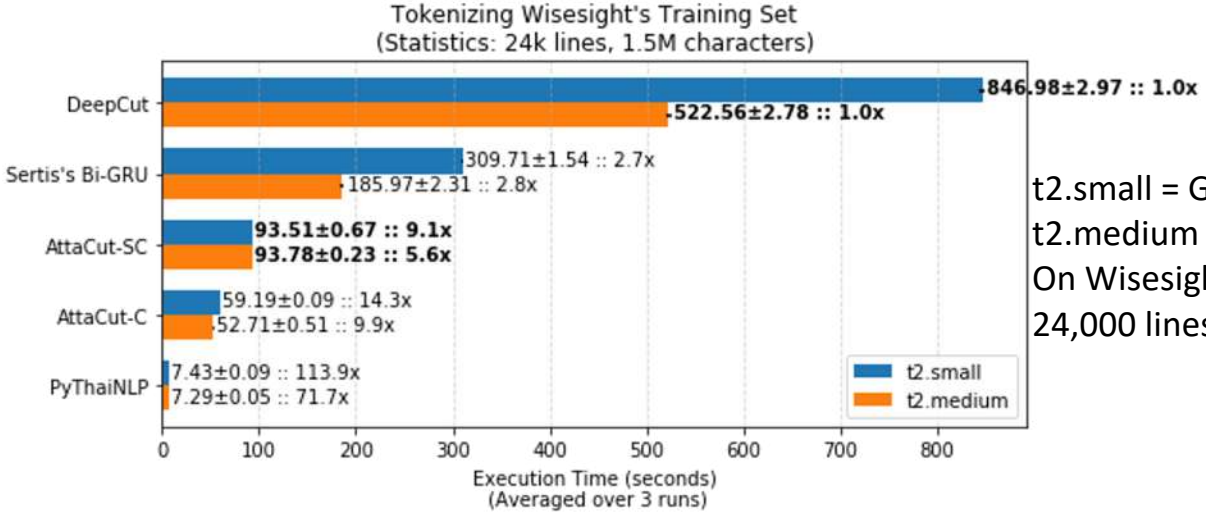
https://github.com/PyThaiNLP/attacut

NECTEC. BEST: Benchmark for Enhancing the Standard of Thai language processing, 2010.

		Others			Ours	
Last Updated: 29/08/2019		**PyThaiNLP newmm**	**Sertis Bi-GRU**	**DeepCut**	**AttaCut-C**	**AttaCut-SC**
BEST Validation Set						
Character-Level	precision	0.94±0.11	0.95±0.10	0.99±0.05	0.97±0.07	0.98±0.05
	recall	0.83±0.09	0.99±0.02	0.99±0.03	0.98±0.04	0.99±0.03
	f1	0.88±0.08	0.97±0.07	**0.99±0.04**	0.98±0.05	0.99±0.04
Word-Level	precision	0.73±0.16	0.91±0.14	0.97±0.07	0.94±0.10	0.96±0.08
	recall	0.65±0.16	0.94±0.10	0.97±0.07	0.94±0.09	0.97±0.08
	f1	0.68±0.15	0.93±0.12	**0.97±0.07**	0.94±0.10	**0.97±0.08**
BEST Test Set						
Character-Level	precision	0.91±0.15	0.92±0.11	0.96±0.08	0.94±0.10	0.95±0.09
	recall	0.85±0.09	0.98±0.04	0.98±0.04	0.98±0.04	0.98±0.04
	f1	0.86±0.11	0.95±0.08	**0.97±0.06**	0.96±0.07	0.96±0.07
Word-Level	precision	0.70±0.19	0.85±0.18	0.92±0.14	0.88±0.17	0.91±0.15
	recall	0.64±0.18	0.90±0.14	0.93±0.12	0.91±0.14	0.92±0.13
	f1	0.67±0.19	0.87±0.16	**0.93±0.13**	0.89±0.16	0.91±0.14



Tokenization Time of PyThaiNLP's newmm, DeepCut, and AttaCut on Google Colab



t2.small = Google Colab
t2.medium = AWS's EC2
On Wisesight Sentiment Corpus
24,000 lines & 1.5M characters

Part3: Text Classification

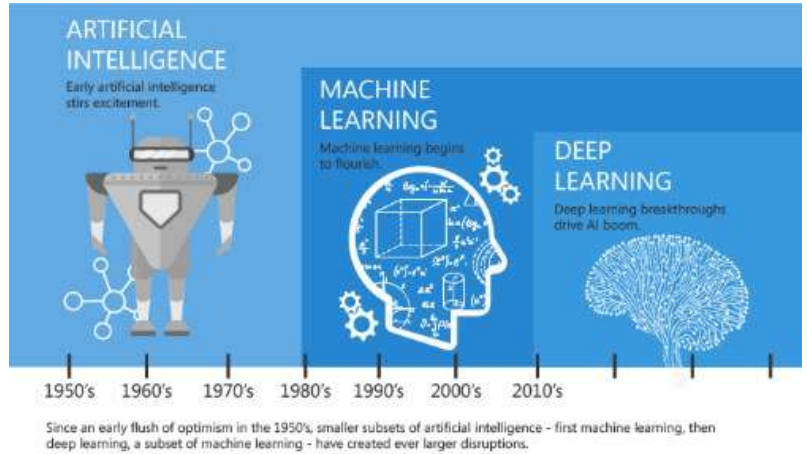
(Common) Text classification pipeline

1) Data preparation

2) Document representation

3) Supervised learning model

Comments	Good	Like	Hate	Sentiment
Tweet1	7	8	0	
Tweet2	1	0	10	
Tweet3	2	9	1	



Truevoice-intent (action 7 classes)

truevoice-intent: destination

We benchmark truevoice-intent by using `destination` as target and construct a 7-class multi-class classification. The performance is measured by micro-averaged and macro-averaged accuracy and F1 score. Codes can be run to confirm performance at this notebook. We also provide performance metrics by class in the notebook.

```
1 display(data_df['cleaned_labels'].unique())
```

```
array(['enquire', 'report', 'cancel', 'buy', 'activate', 'request',
       'garbage', 'change'], dtype=object)
```

```
1 # Show first 10 lines of the data
2 !head clean-phone-data-for-students.csv
```

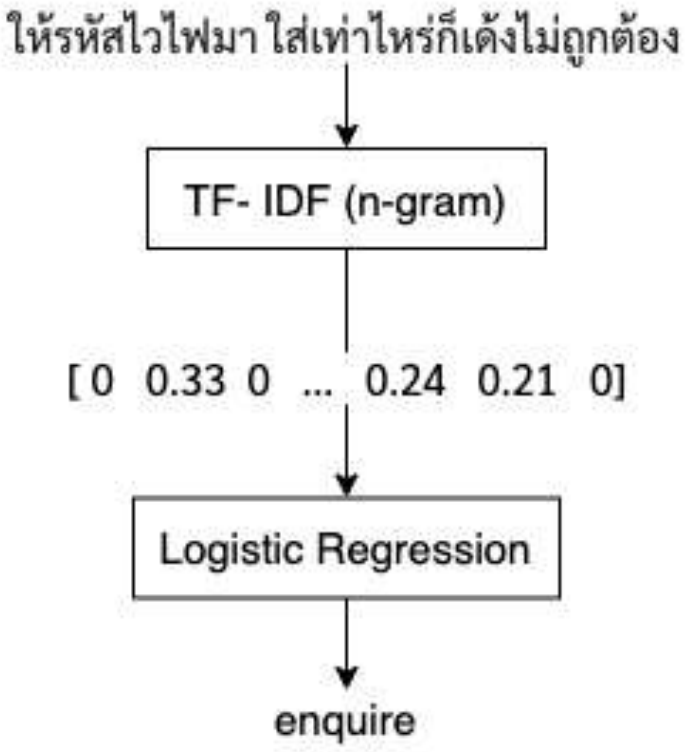
```
Sentence Utterance,Action,Object
 <PHONE_NUMBER_REMOVED> ผมไปจ่ายเงินที่ Counter Services เค้าเช็ต 3276.25 บาท เมื่อวานที่ผมเช็คที่ศูนย์บอกมียอด 3057.79 บาท,enquire,payment
 internet ยังความเร็วอยู่เท่าไหร ครับ,enquire,package
 ตะกี้ไปชำระค่าบริการไปแล้ว แต่ยังใช้งานไม่ได้ ค่ะ,report,suspend
 พี่ค่ะยังใช้ internet ไม่ได้เลยค่ะ เป็นเครื่อง โกลไล,enquire,internet
 ฮาโหล คะ พอดีว่าเมื่อวานเปิดซิมทรูมูฟ แต่มันโทรออกไม่ได้คะ แต่เล่นเนตได้คะ,report,phone_iss
*2222 ใช้งานยังไง ขอรายละเอียดการสมัครหน่อย,enquire,service
<PHONE_NUMBER_REMOVED> เคยมีช่างมาซ่อมที่บ้าน แล้วโทรศัพท์ใช้งานไม่ได้ครับ,enquire,nonTrue
<PHONE_NUMBER_REMOVED> ค้างค่าบริการเท่าไหร่ครับ,enquire,balance
<PHONE_NUMBER_REMOVED> อินเตอร์เน็ตไฟ Adsl ไม่มีสัญญาณครับ,enquire,nonTrueMove
```

	Sentence Utterance	Action	Object
0	<PHONE_NUMBER_REMOVED> ผมไปจ่ายเงินที่ Counte...	enquire	payment
1	internet ยังความเร็วอยู่เท่าไหร ครับ	enquire	package
2	ตะกี้ไปชำระค่าบริการไปแล้ว แต่ยังใช้งานไม่ได้...	report	suspend
3	พี่ค่ะยังใช้ internet ไม่ได้เลยค่ะ เป็นเครื่อ...	enquire	internet
4	ฮาโหล คะ พอดีว่าเมื่อวานเปิดซิมทรูมูฟ แต่มันโ...	report	phone_issues

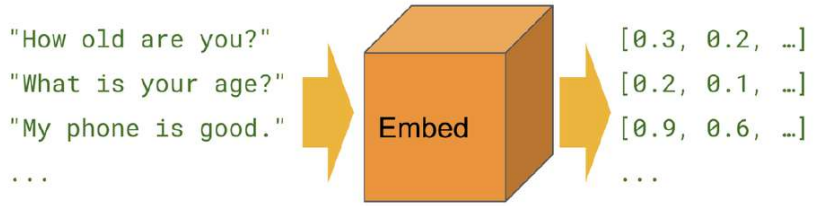
https://github.com/PyThaiNLP/classification-benchmarks

Text Classification Approach

1) TF-IDF + Classifier



2) USE + Classifier



3) BERT

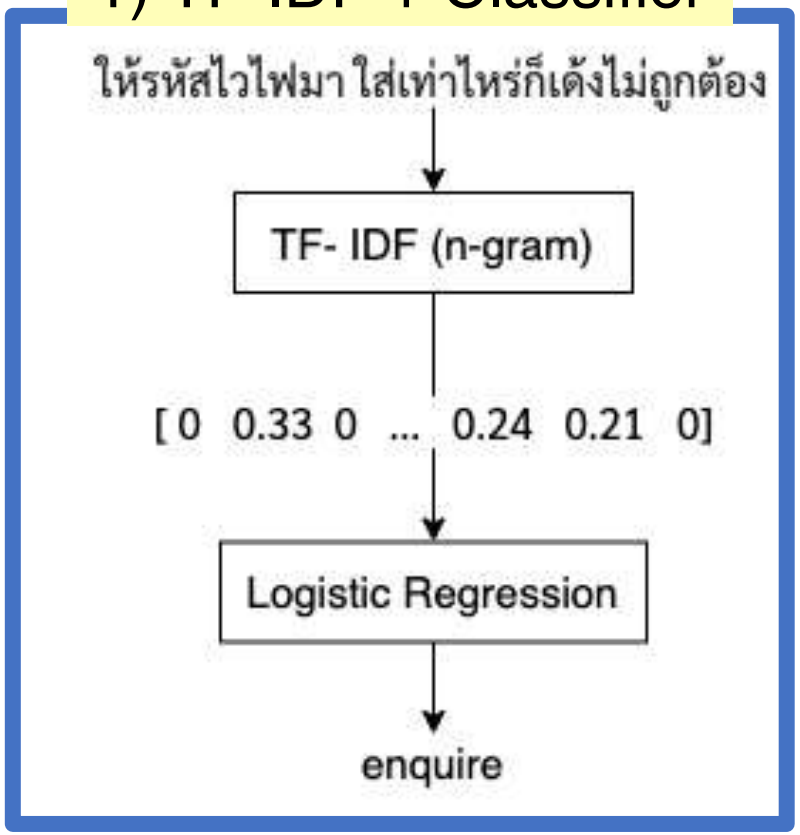
The AI community building the future.

Build, train and deploy state of the art models powered by the reference open source in natural language processing.

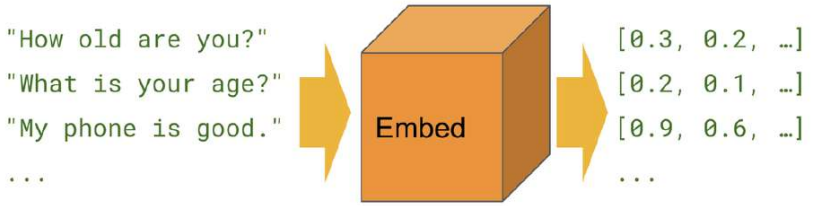
Star 47,792

Text Classification Approach

1) TF-IDF + Classifier



2) USE + Classifier



3) BERT

The AI community building the future.

Build, train and deploy state of the art models powered by the reference open source in natural language processing.

Star 47,792

Sparse representation: Term Frequency (TF)

- Each row represents a word in the vocabulary and term-document matrix
- Each column represents a document.

vocabulary

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	5	117	0	0

document

Figure 15.1 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

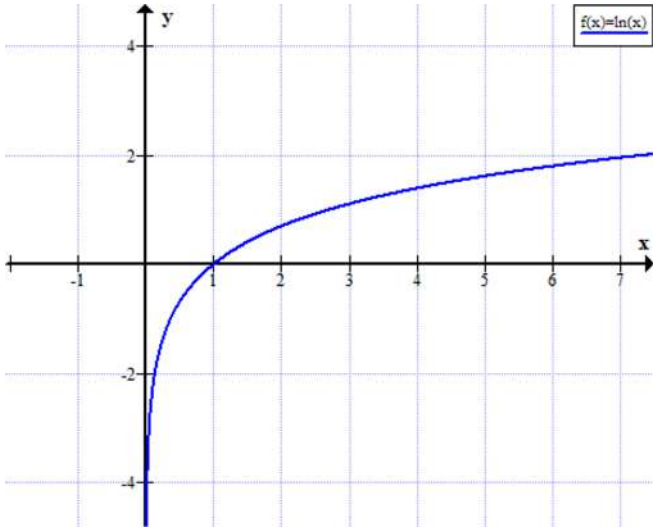
Reference: Jurafsky, Dan, and James H. Martin. Speech and language processing. 3rd edition draft, https://web.stanford.edu/~jurafsky/slp3/, August 2017

Sparse representation: TF-IDF

Need for normalization in TF

Doc1
cat = 5/10

Doc2
cat = 50/1000



- Term Frequency (TF) – per each document

$$TF(w) = \frac{\text{Frequency of word } w \text{ in a document}}{\text{Total number of words in the document}}$$

- Inverse Document Frequency (IDF) – per corpus (all documents)

$$IDF(w) = \log_e\left(\frac{\text{Total number of documents}}{\text{Number of documents that contain word } w}\right)$$

penalty score
i.e., a, an, the

- TF-IDF

$$TFIDF(w) = TF(w) * IDF(w)$$

```
# TF-IDF
tfidf = TfidfVectorizer(
    ngram_range=(1,2),        # Use unigram and bigram
    tokenizer=word_tokenize, # Use `word_tokenize` method from pythainlp for tokenizer
    min_df=2,                 # The word found less than three times in dataset is ignored
    max_df=0.9,               # The word found more than 90% of entries is ignore
    use_idf=True,
    smooth_idf=True,
    sublinear_tf=True
)
# Logistic regresstion
model = LogisticRegression(C=4, max_iter=300, random_state=42)
```

Sparse representation: TF-IDF (cont.)

TF

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	5	117	0	0

TF-IDF

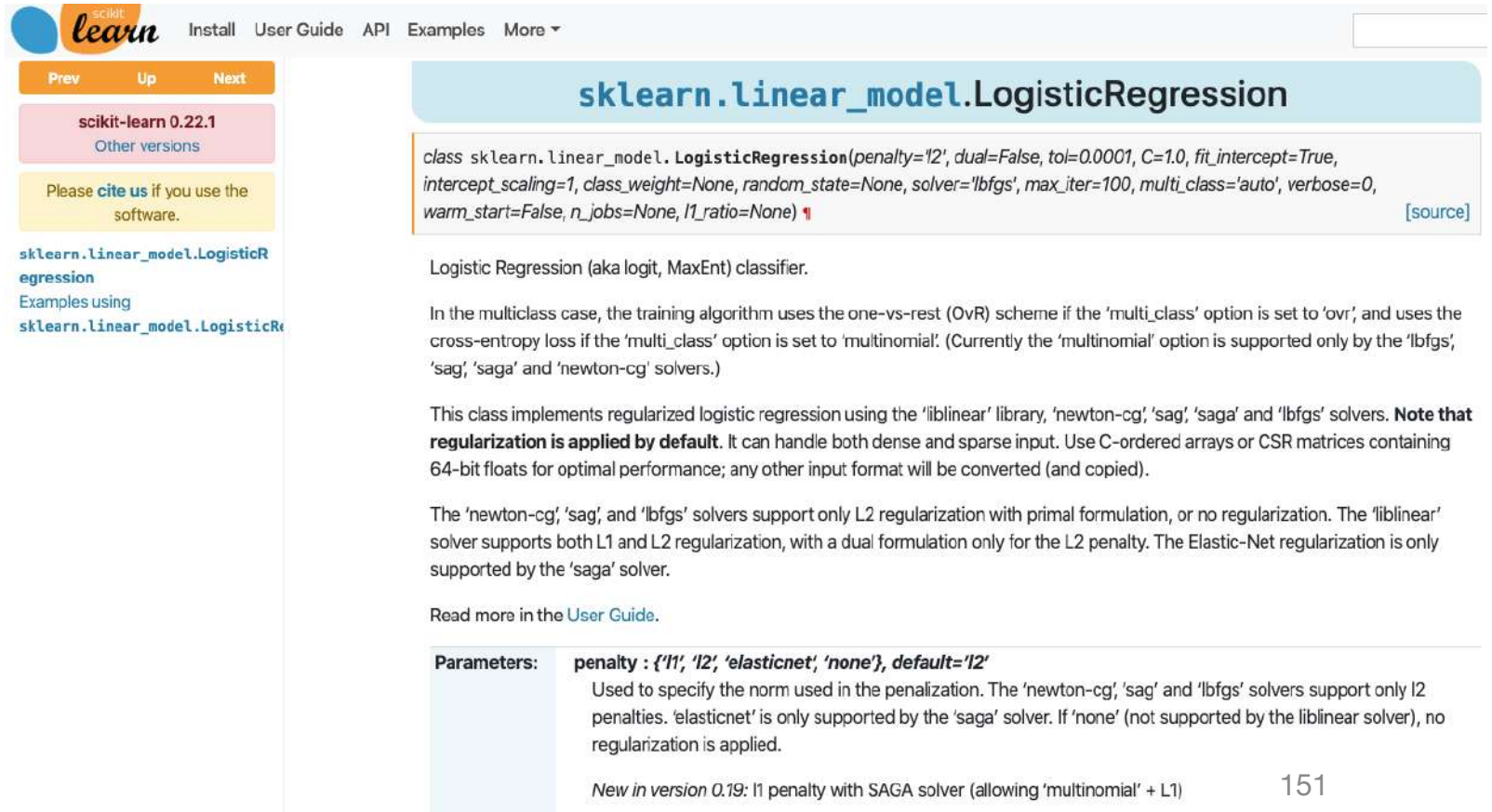
	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

Reference: Jurafsky, Dan, and James H. Martin. Speech and language processing. 3rd edition draft, https://web.stanford.edu/~jurafsky/slp3/6.pdf, Feb 2020

What classifier?

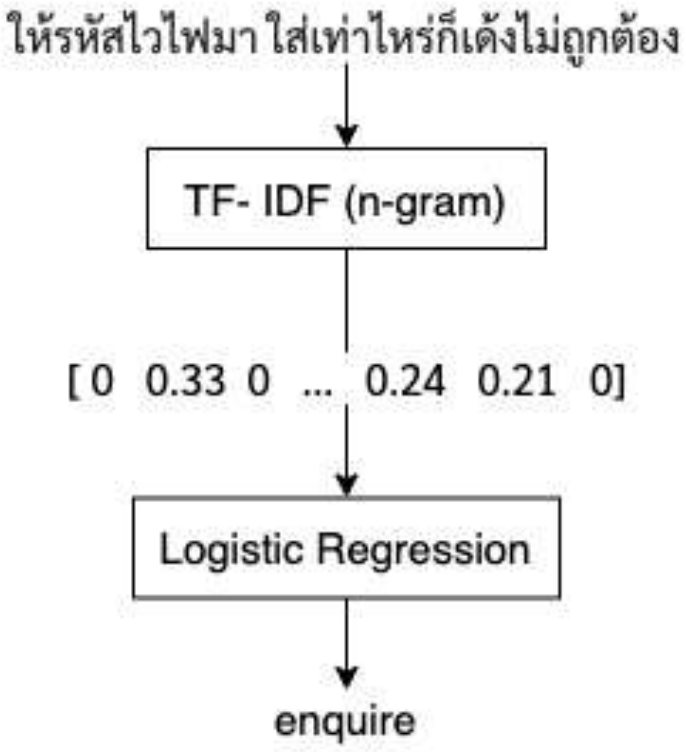
- Any classifier you like
- k-NN
- Naïve Bayes
- Logistic regression
- SVM
- Neural networks

```python
# TF-IDF
tfidf = TfidfVectorizer(
    ngram_range=(1,2),       # Use unigram and bigram
    tokenizer=word_tokenize, # Use `word_tokenize` method from pythainlp for tokenizer
    min_df=2,                # The word found less than three times in dataset is ignored
    max_df=0.9,              # The word found more than 90% of entries is ignore
    use_idf=True,
    smooth_idf=True,
    sublinear_tf=True
)
# Logistic regresstion
model = LogisticRegression(C=4, max_iter=300, random_state=42)
```

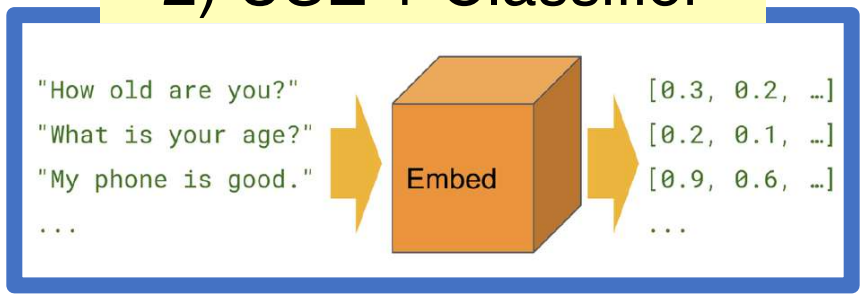


Text Classification Approach

1) TF-IDF + Classifier



2) USE + Classifier



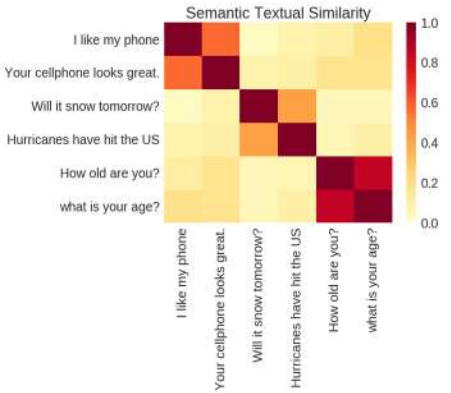
3) BERT

The AI community building the future.

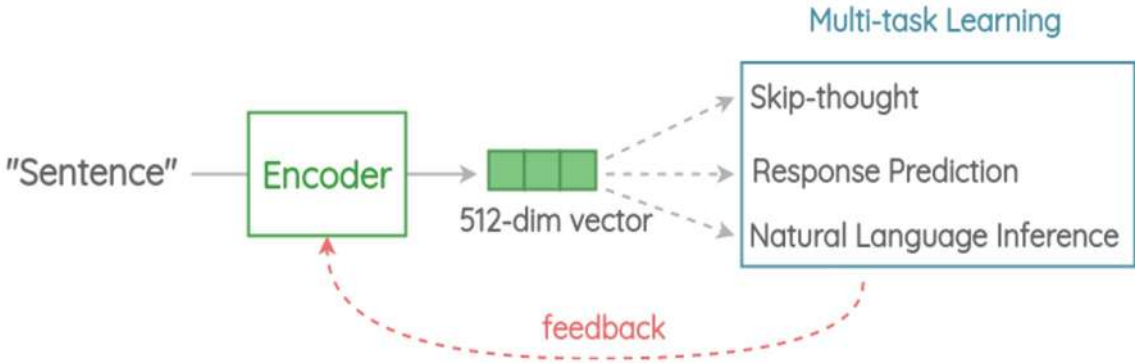
Build, train and deploy state of the art models powered by the reference open source in natural language processing.

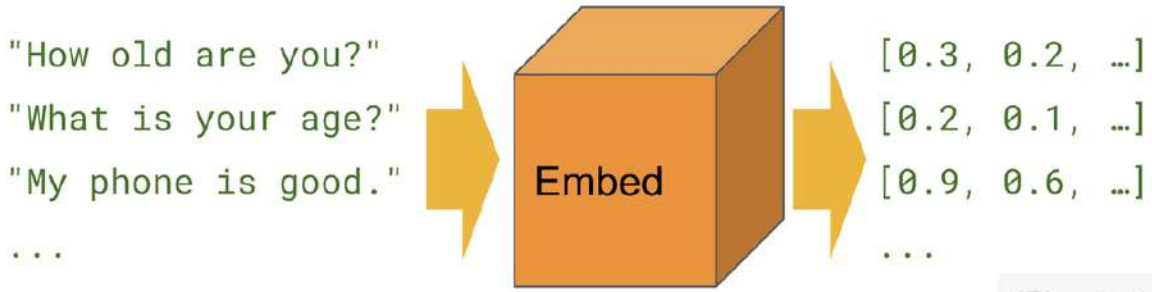
Star 47,792

Universal Sentence Encoder (USE)



A model focusing on sentence representation
Use sentencepiece tokenization
Pre-trained then used anywhere
Based on (1) DAN (lite version) or (2) Transformer






```
1 test_str = "\u0E2D\u0E22\u0E32\u0E01\u0E40\u0E23\u
2 input = [ test_str ] # List of strings
3 output = use_model(input)
4 print("Vector size:", output.shape)
5 print("Object type:", type(output))
```

test_str: "อยากเรียน NLP จังเลยจ้า

```
Vector size: (1, 512)
Object type: <class 'tensorflow.python.framework.ops.EagerTensor'>
```

Official implementation with pretrained weights
https://tfhub.dev/google/collections/universal-sentence-encoder/1
https://ai.googleblog.com/2018/05/advances-in-semantic-textual-similarity.html

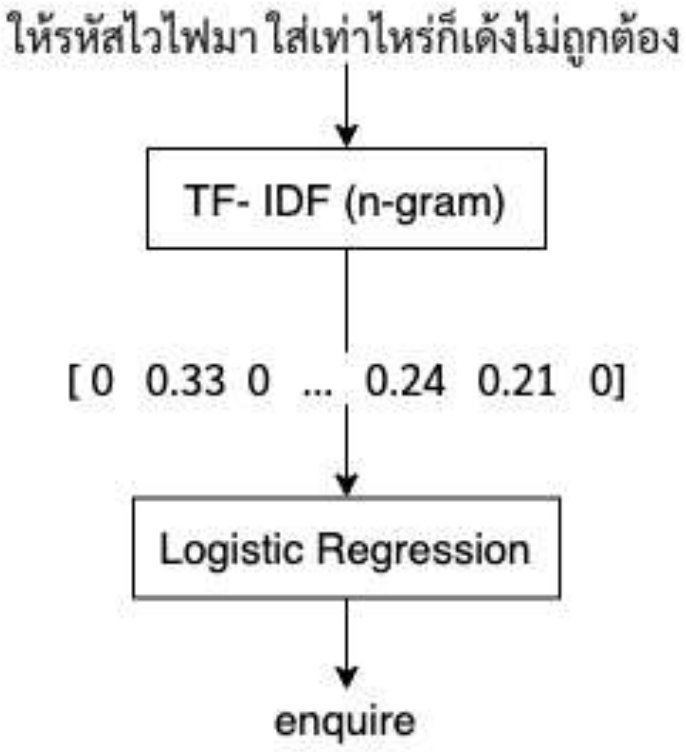
Download-ables

Model	Comments
universal-sentence-encoder	
universal-sentence-encoder-large	
universal-sentence-encoder-lite	
universal-sentence-encoder-qa	Question answering
universal-sentence-encoder-multilingual	16 languages
universal-sentence-encoder-multilingual-large	16 languages
universal-sentence-encoder-multilingual-qa	16 languages , Question answering

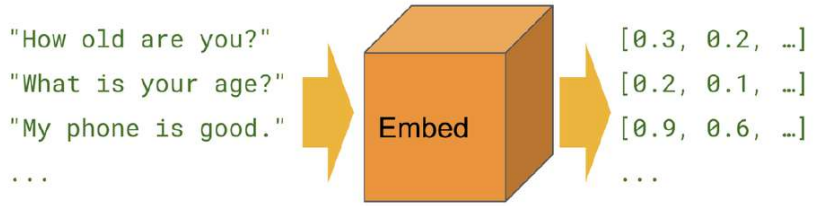
https://tfhub.dev/google/collections/universal-sentence-encoder/1

Text Classification Approach

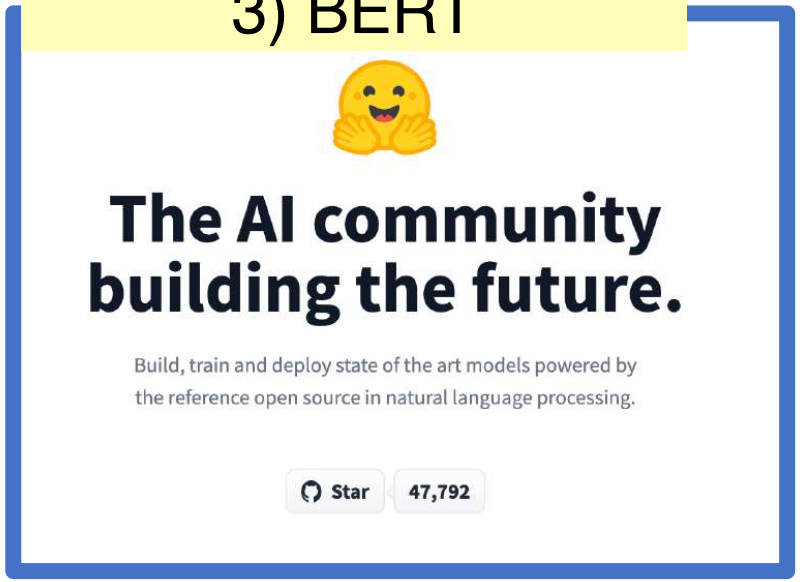
1) TF-IDF + Classifier



2) USE + Classifier

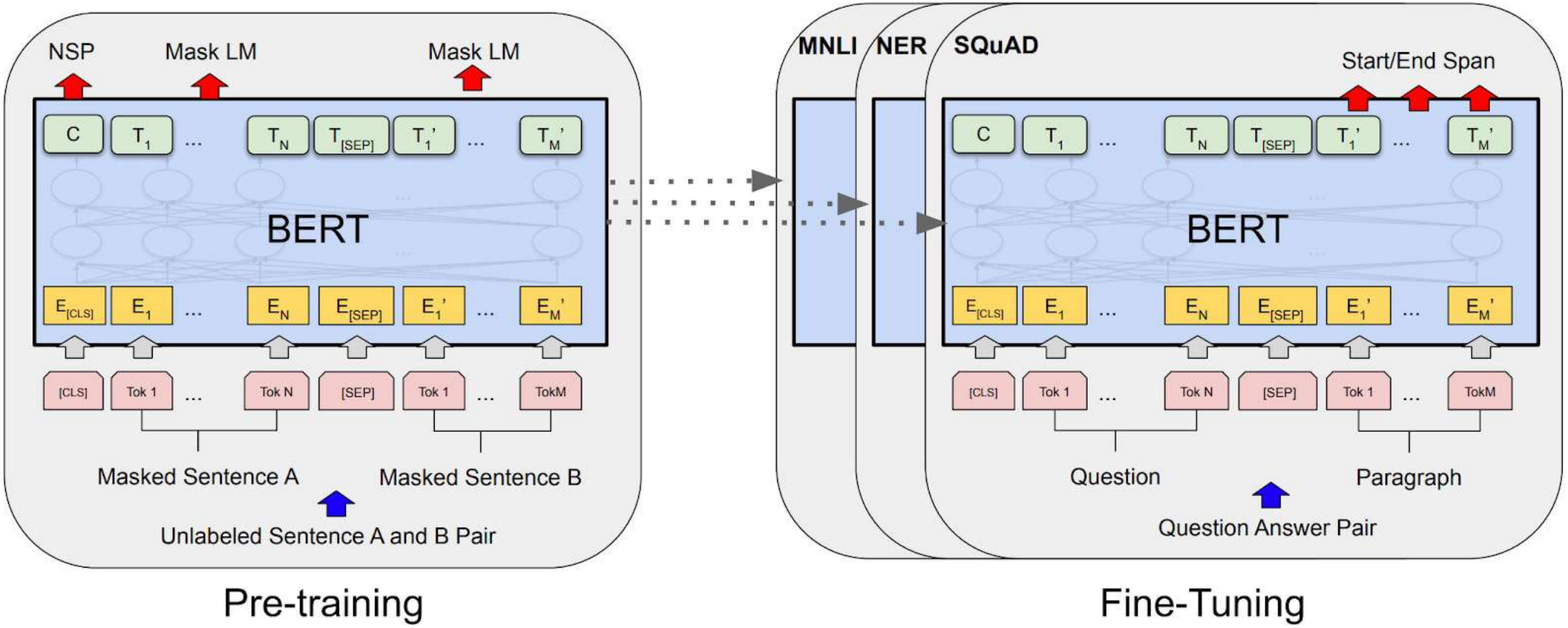


3) BERT



BERT

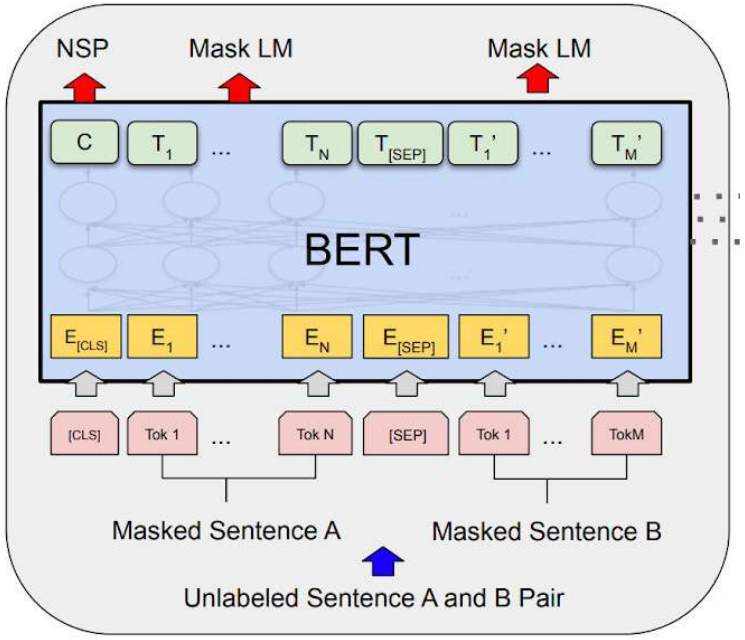
- Pretrained language model based on transformers
  - Can be used in many NLP tasks



BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2018 https://arxiv.org/abs/1810.04805

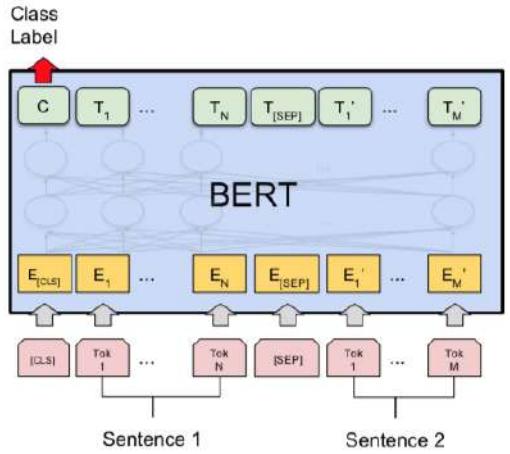
Pre-training BERT

- Predicting masked words in a sentence
  - The quick brown fox jumps over the [MASK]
  - Variants: predict correct word or not, predict swapped words, etc.
- Next sentence prediction
  - A: The cat is scared. B: It hides under the table.
  - A: The apple is on the table. B: It always rain.
  - Variants: sentence order prediction

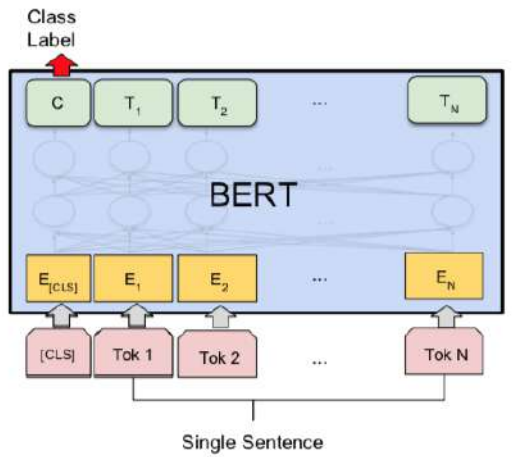


Pre-training

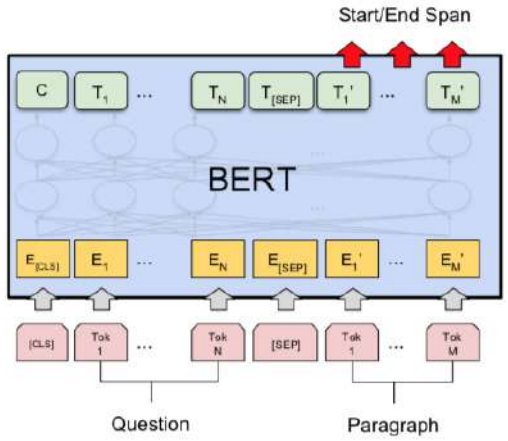
Downstream tasks with BERT



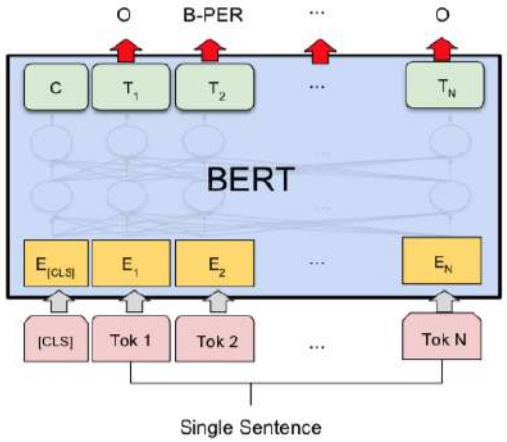
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



(b) Single Sentence Classification Tasks: SST-2, CoLA



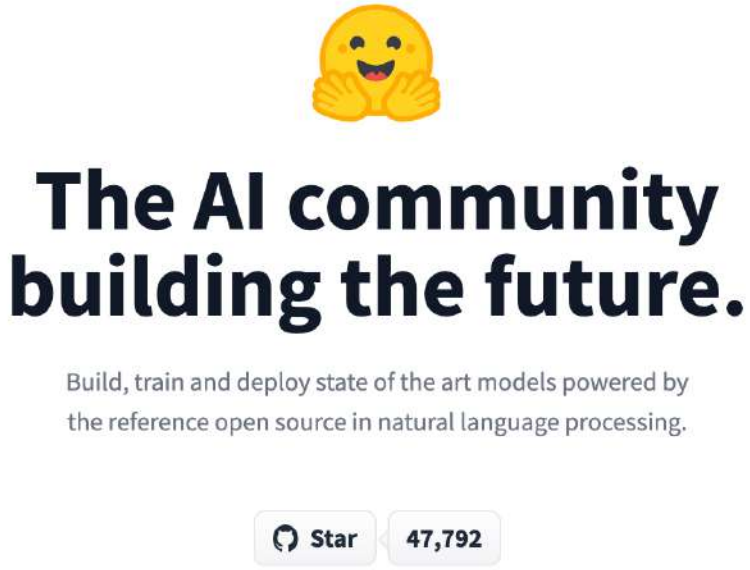
(c) Question Answering Tasks: SQuAD v1.1



(d) Single Sentence Tagging Tasks: CoNLL-2003 NER

Huggingface

- An opensource library for transformer-related models
- Has datasets, models, scripts, deployment solutions
- New official online course https://huggingface.co/course/chapter1



Thai pre-trained BERTs

- WangchanBERTa
- XLM-RoBERTa
- MT5

https://huggingface.co/airesearch/wangchanberta-base-att-spm-uncased
https://huggingface.co/transformers/model_doc/xlmroberta.html
https://huggingface.co/transformers/model_doc/mt5.html

Results

- WangchanBERTa is the winner.
- Bidirectional LSTM is the second rank, but it is the slowest one.
- Fine-tuning can improve a performance.
- USE is pretty easy to use and can show a promising performance.
- NB-SVM is a strong traditional ML model.

#	Model	Accuracy	macro F1		speed (ms)
Model1	n-grams + NB	0.825	0.739		2.4
Model2	TF-IDF + Logistic	0.864	0.760		2.9
Model3	NB-SVM	0.858	0.764	4	17.2
Model3.1	LTW2V + CNN	0.854	0.725		58.3
Model3.2	LTW2V + Bidirectional LSTM	0.870	0.779	2	63.3
Model3.3	LTW2V + DAN	0.736	0.472		
Model4	LTW2V + USE	0.870	0.766	3	23.9
Model3.1 (f)	LTW2V + CNN (fine-tune)	0.849	0.741		
Model3.2 (f)	LTW2V + Bidirectional LSTM (fine-tune)	0.872	0.791	2	
Model5	WangchanBERTa	0.891	0.814	1	10.3