CHULA ENGINEERING
Foundation toward Innovation
COMPUTER

Data Mining Group
Machine Intelligence and Knowledge Discovery Lab
Chulalongkorn University

Introduction to Deep Learning (CNN)

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/

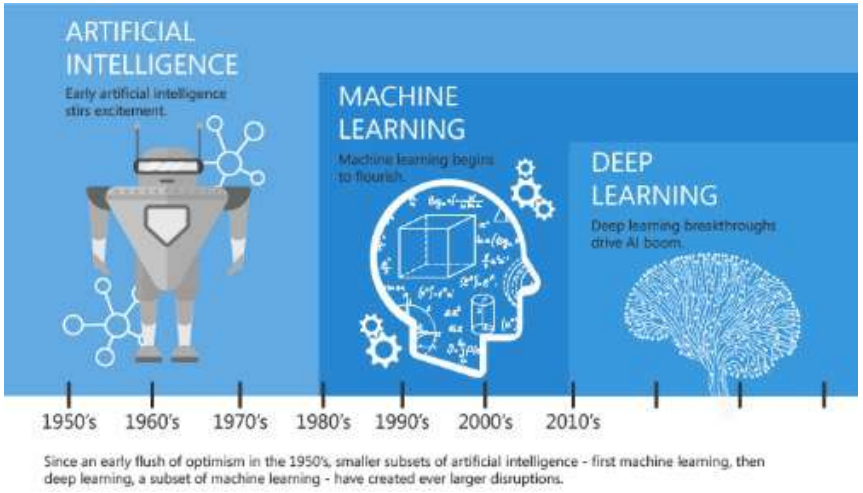
Outline

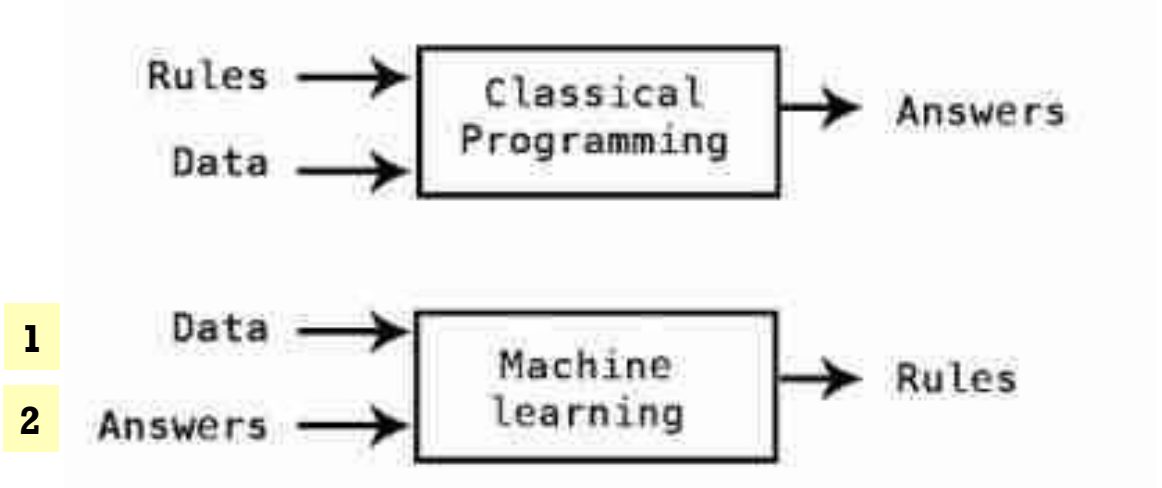
- Introduction to Deep Learning
- Convolutional Neural Networks (CNN) [Imaging Task]
- Image Classification
- Object Detection
- Semantic Segmentation

Introduction to Deep Learning

AI = Automation

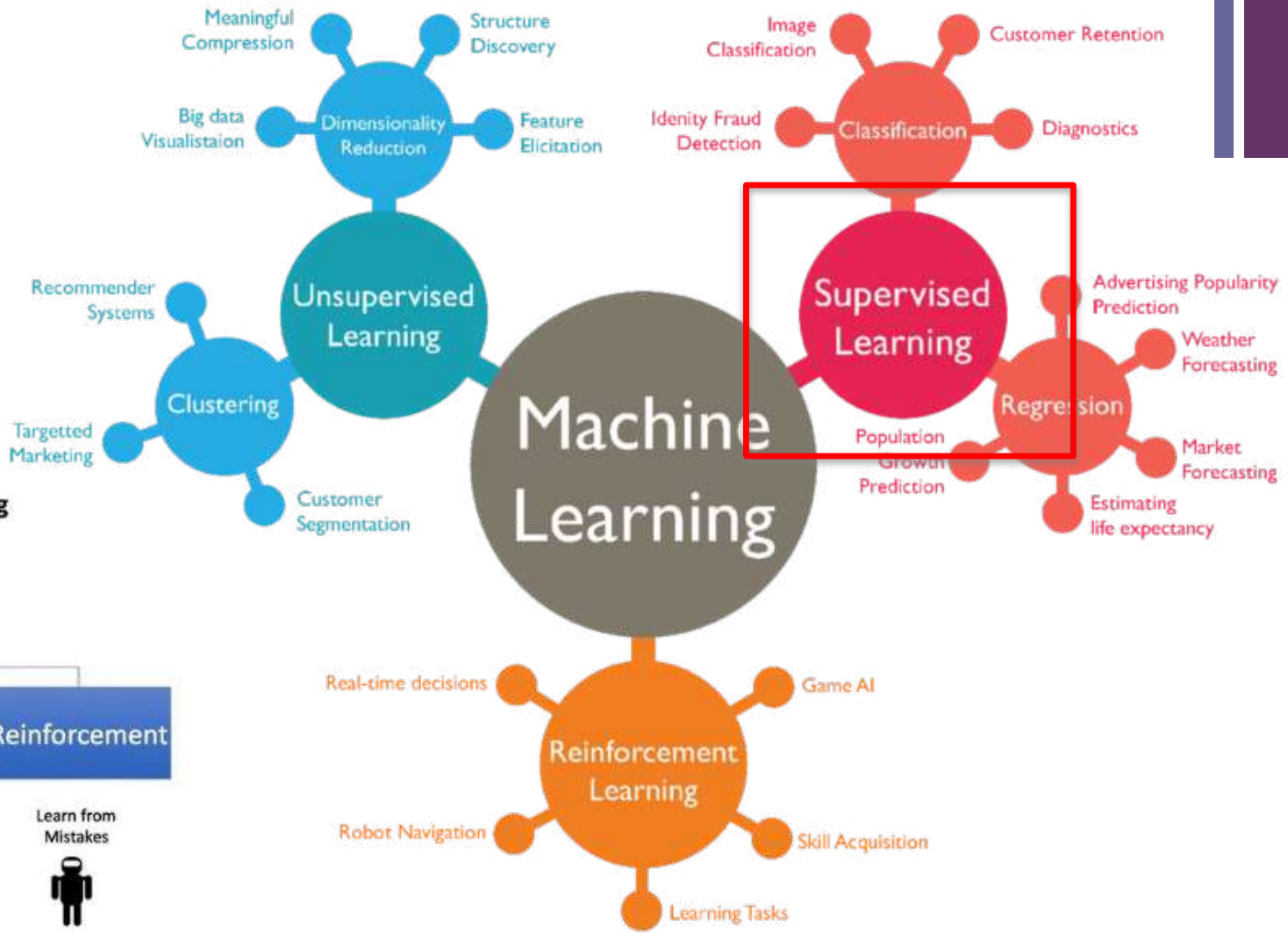
- 1) Rule-based AI (Symbolic AI)
- 2) Machine Learning



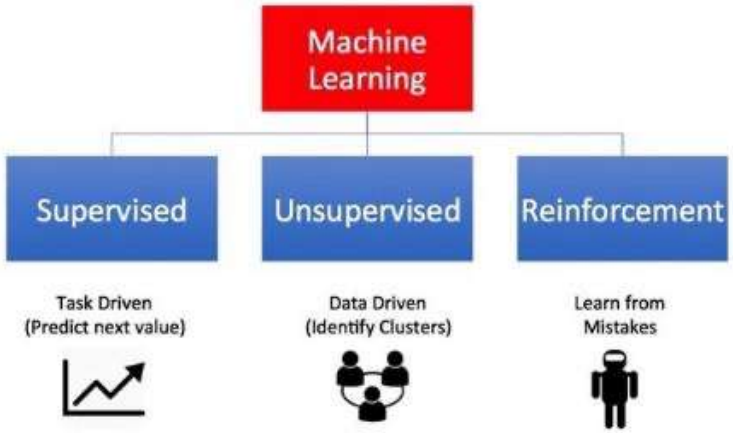


https://mc.ai/machine-learning-basics-artificial-intelligence-machine-learning-and-deep-learning/

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

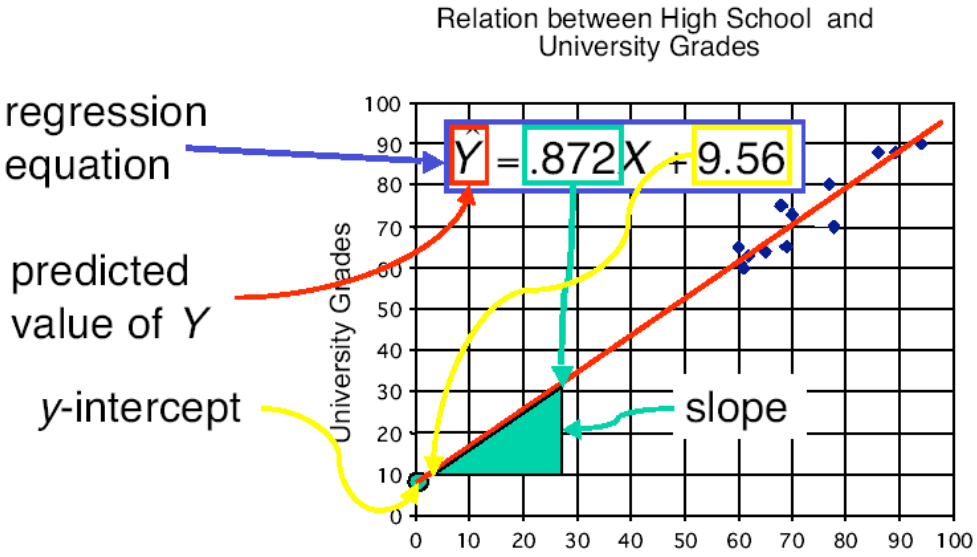
Age	**Gender**	**BodyTemp**	**Cough**	**Corona**
25	Male	40	No	?

Application: Corona Prediction

Prediction algorithms

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

Regression – Linear Relationship



weight, coefficient

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

input

- The least square method aims to minimize the following term

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

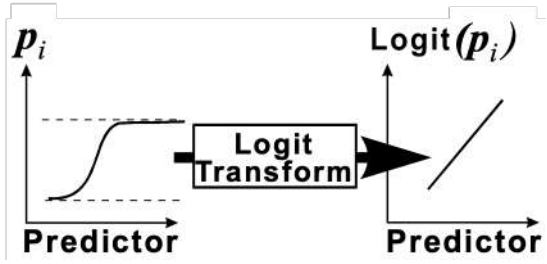
Logistic Regression (cont.)
Linear Relationship

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"

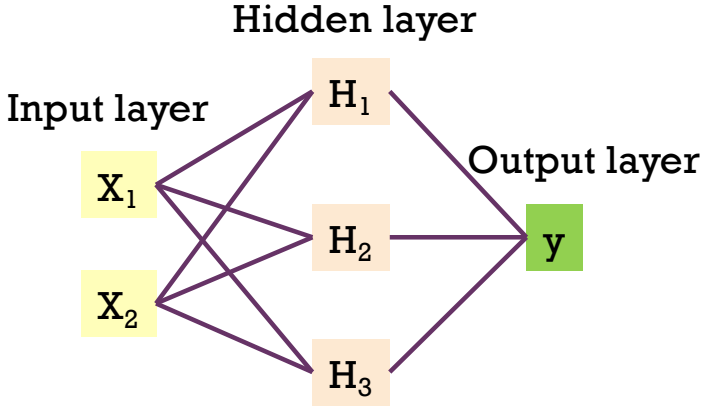


$$\hat{p} = \frac{1}{1 + e^{-\text{logit}(\hat{p})}}$$

Application: Corona Prediction

Neural Networks (universal approximator)

Non-linear relationship

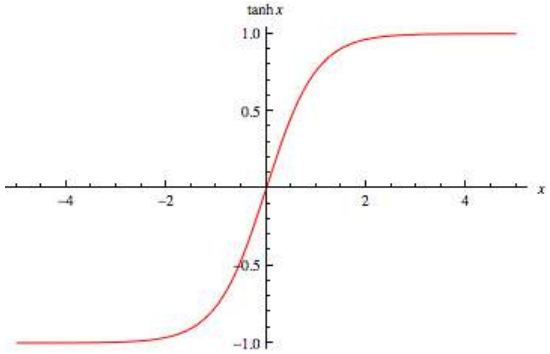


$$log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{w}_0 + \hat{w}_1 H_1 + \hat{w}_2 H_2 + \hat{w}_3 H_3$$

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

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

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

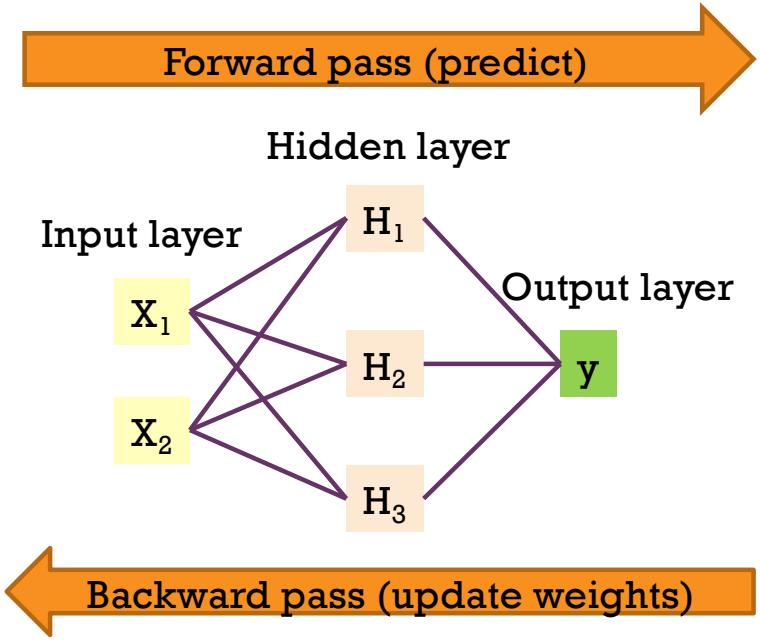


Stop when?
- Converge (no change in loss)
- Max epochs

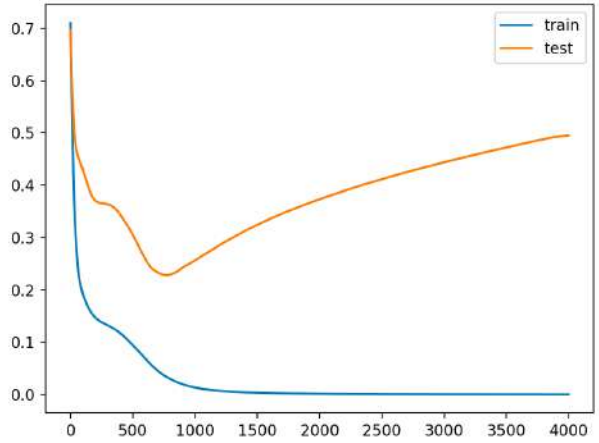
Important Params:
- #hidden units, #hidden layers
- Learning rate, Momentum, decay
- Seed number, etc.

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



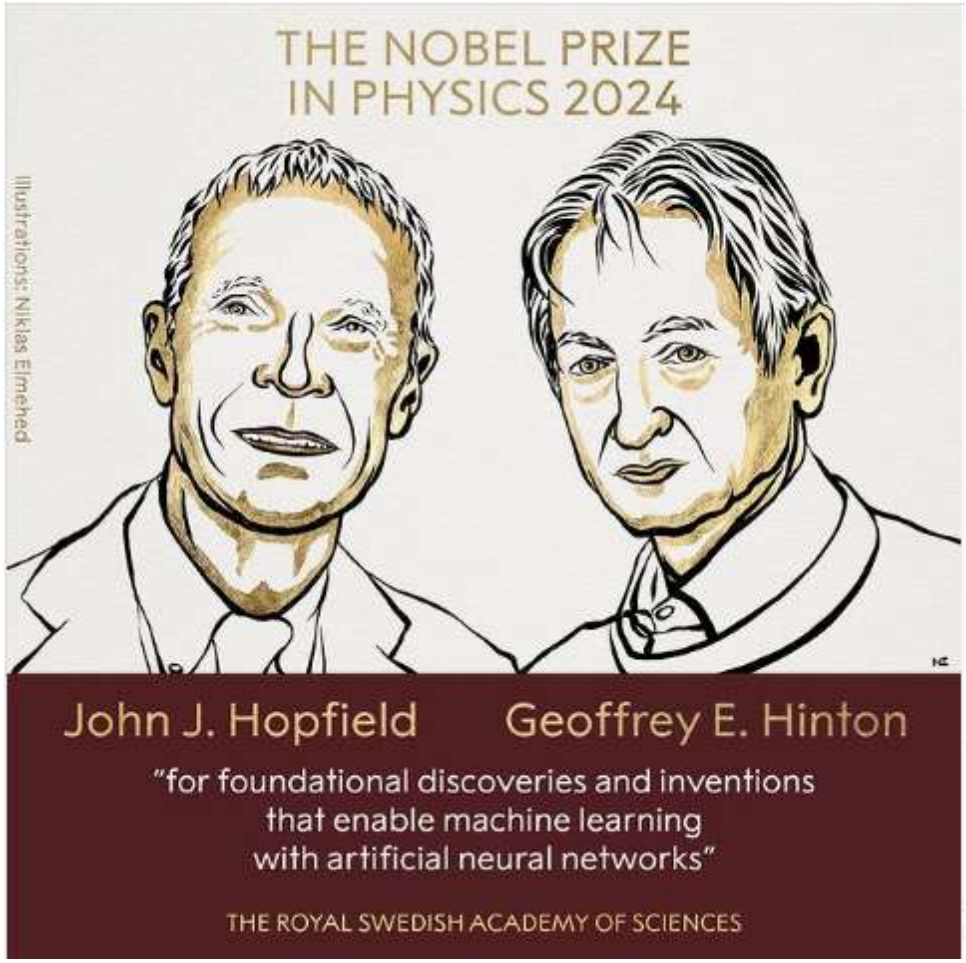
$$log\left(\frac{\hat{p}}{1-\hat{p}}\right) = \hat{w}_0 + \hat{w}_1 H_1 + \hat{w}_2 H_2 + \hat{w}_3 H_3$$

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

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

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

Nobel Prize to Backpropagration's Inventors

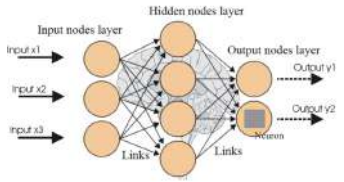


8 October 2024

The Royal Swedish Academy of Sciences has decided to award the Nobel Prize in Physics 2024 to

John J. Hopfield
Princeton University, NJ, USA

Geoffrey E. Hinton
University of Toronto, Canada



"for foundational discoveries and inventions that enable machine learning with artificial neural networks"

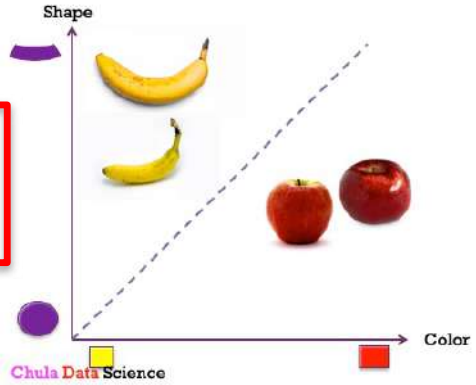
They trained artificial neural networks using physics

John Hopfield invented a network that uses a method for saving and recreating patterns. We can imagine the nodes as pixels. The *Hopfield network* utilises physics that describes a material's characteristics due to its atomic spin – a property that makes each atom a tiny magnet.

Geoffrey Hinton used the Hopfield network as the foundation for a new network that uses a different method: the *Boltzmann machine*. This can learn to recognise characteristic elements in a given type of data.

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

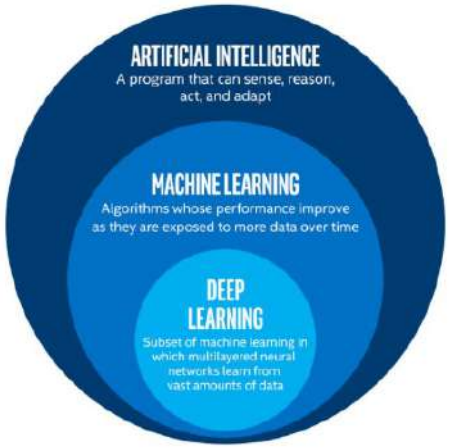


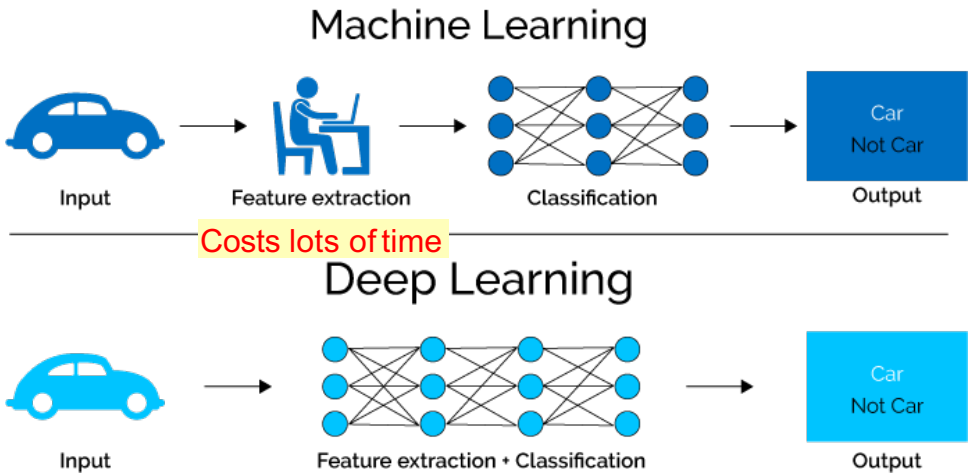
Can we still tell the features (columns)?

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.



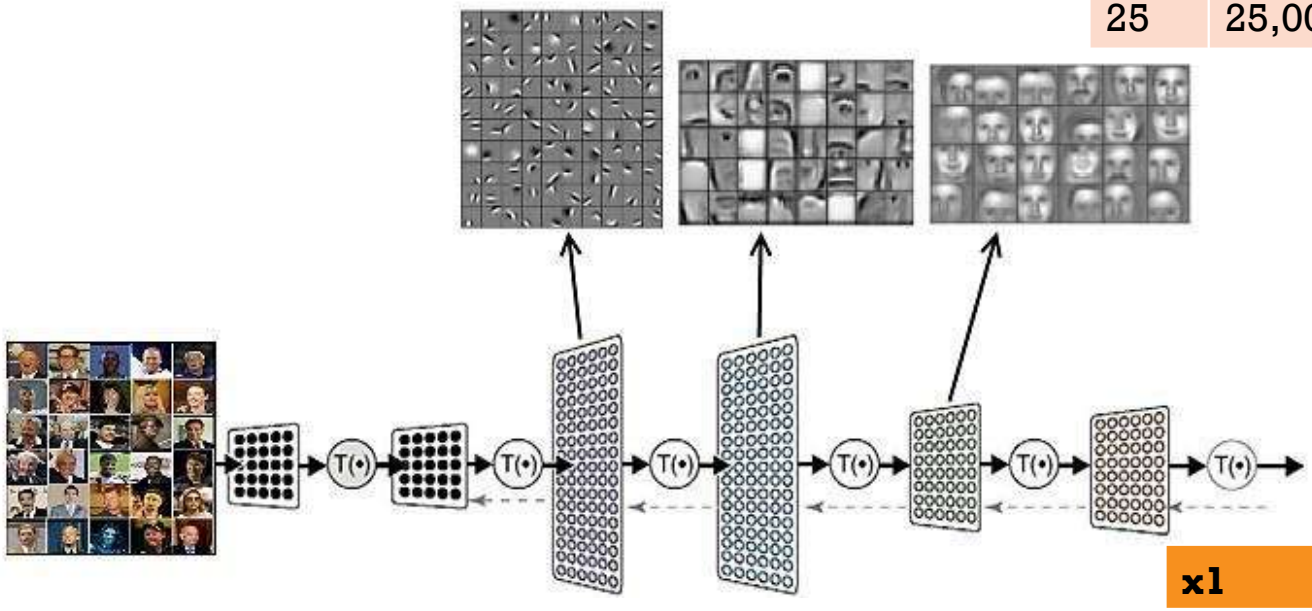


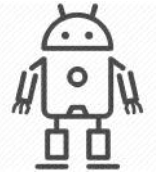
Deep Learning – Basics (cont.)

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

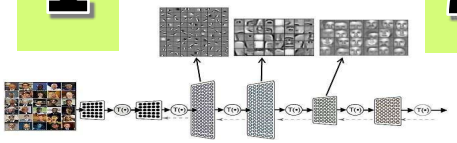
Deep Learning Application

Speech Recognition

Computer Vision

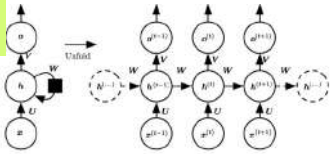
Natural Language Processing

1



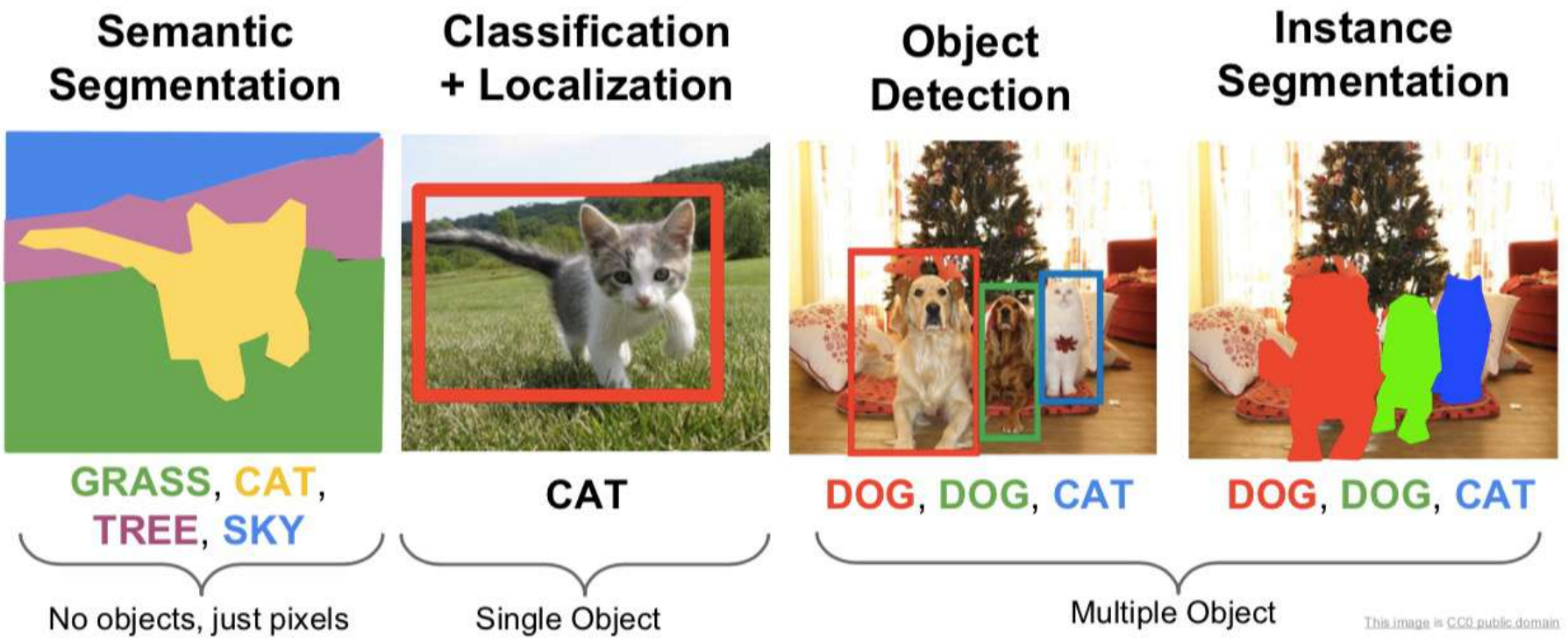
CNN

2



RNN

Type of image tasks

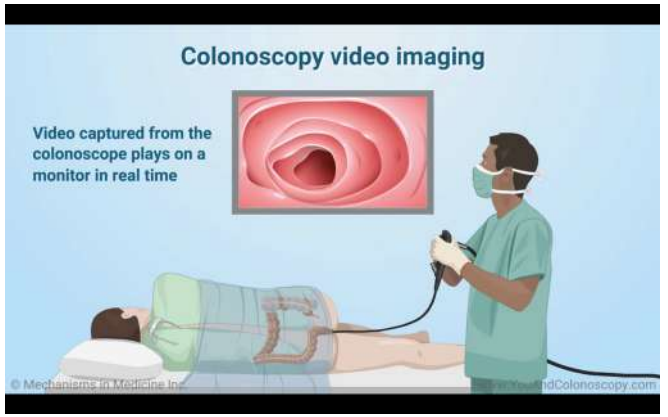


Face Recognition

Smart Medical Diagnosis

CHULA ENGINEERING
Foundation toward Innovation

King Chulalongkorn Memorial Hospital
The Thai Red Cross Society



+

Convolutional Neural Networks (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:

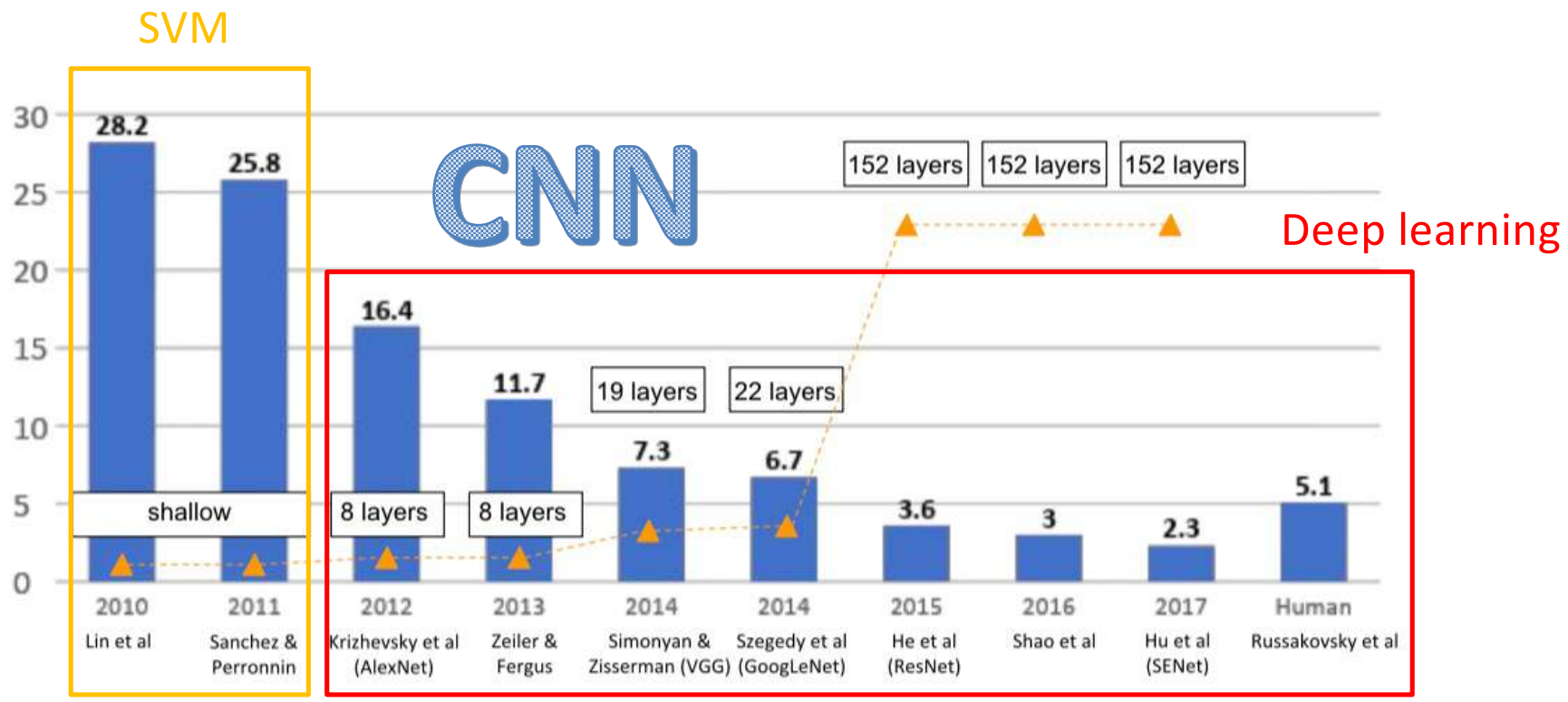
1,000 object classes

1,431,167 images

ImageNet 2017 is the last challenge.

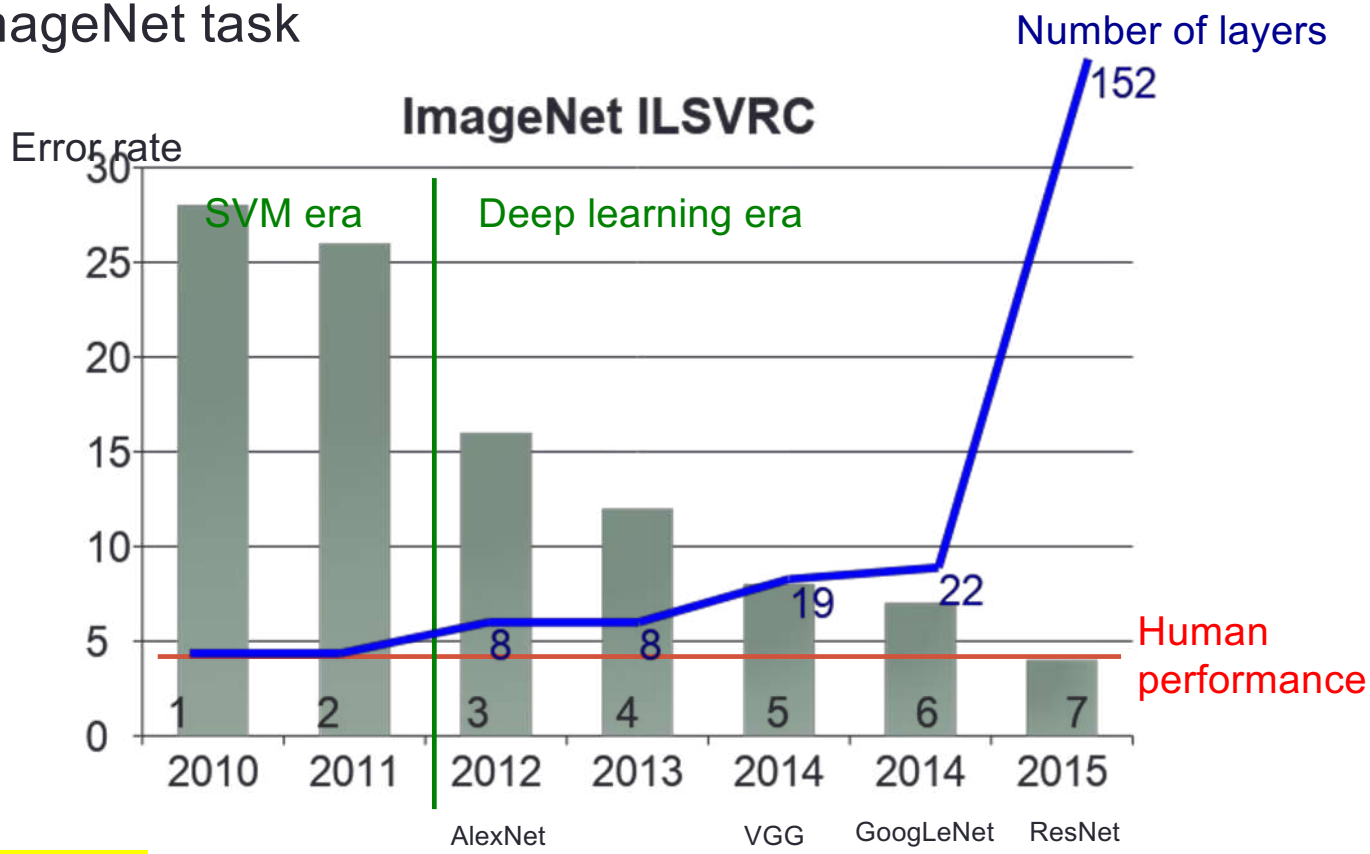
22

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



Wider and deeper networks (Beyond Human)

- ImageNet task

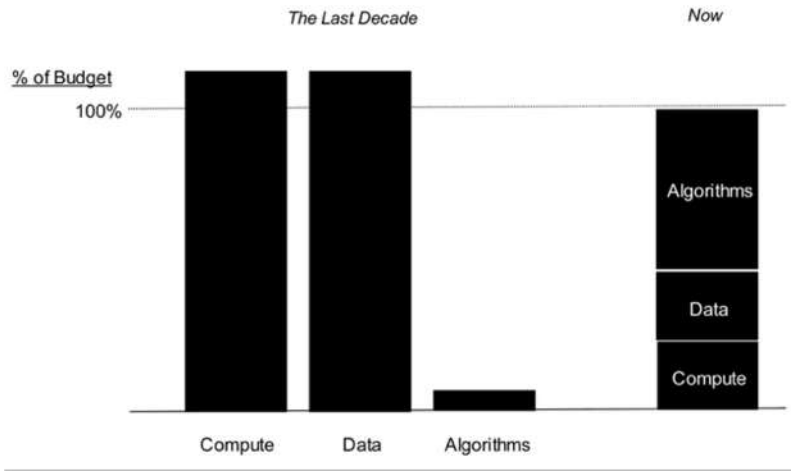


Based on the slide of Aj.Ekapol

Net 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/

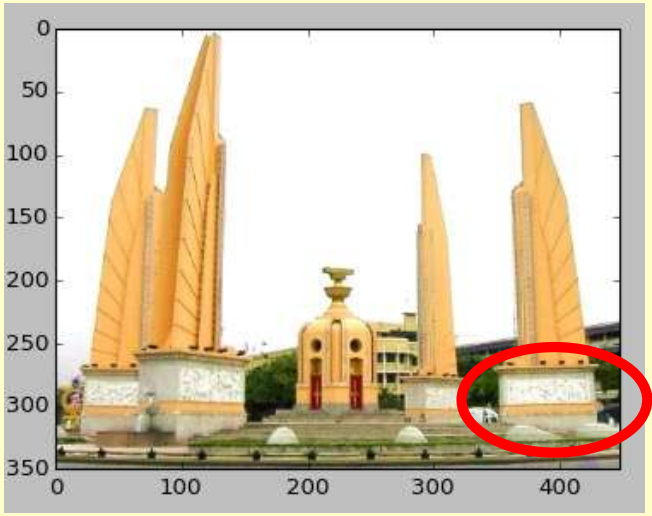
การแสดงรูปภาพใน python ด้วย matplotlib

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

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

plt.show()
```

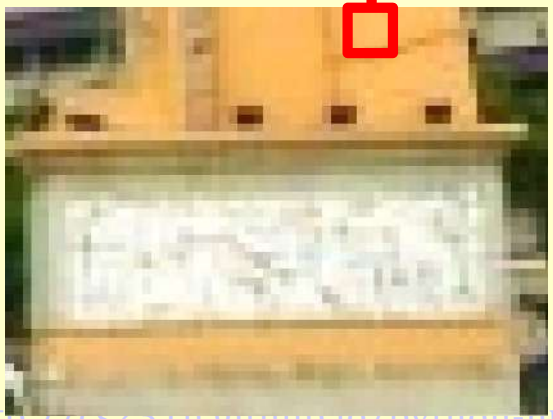
เพื่อความง่ายโปรแกรมที่จะเขียนจากนี้ไปใช้กับแฟ้ม png เท่านั้น

image เป็นอาเรย์ที่แต่ละช่อง
มีค่า 0 ถึง 1 แทนความเข้มของสี

1 pixel = 1 จุดสี มี 3 สีย่อย



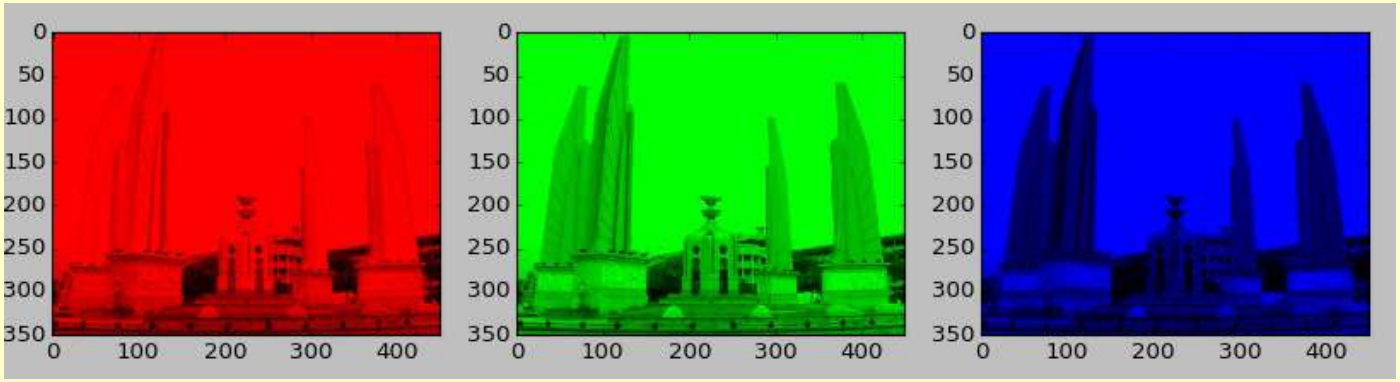
0.98762	0.72549	0.35294



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

It is an array of R,G,B (3 channels)

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



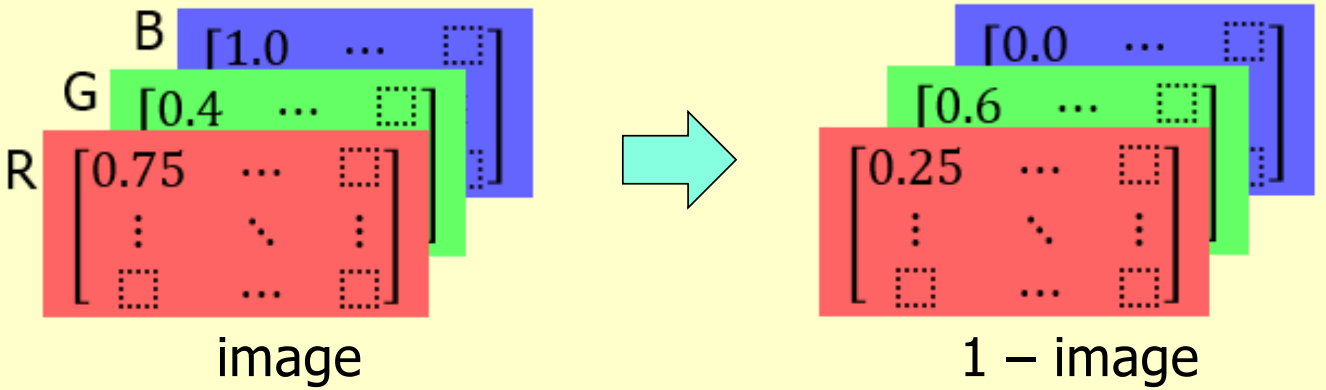
Red
image[:,:,0]

Green
image[:,:,1]

Blue
image[:,:,2]

Basic Image Processing

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

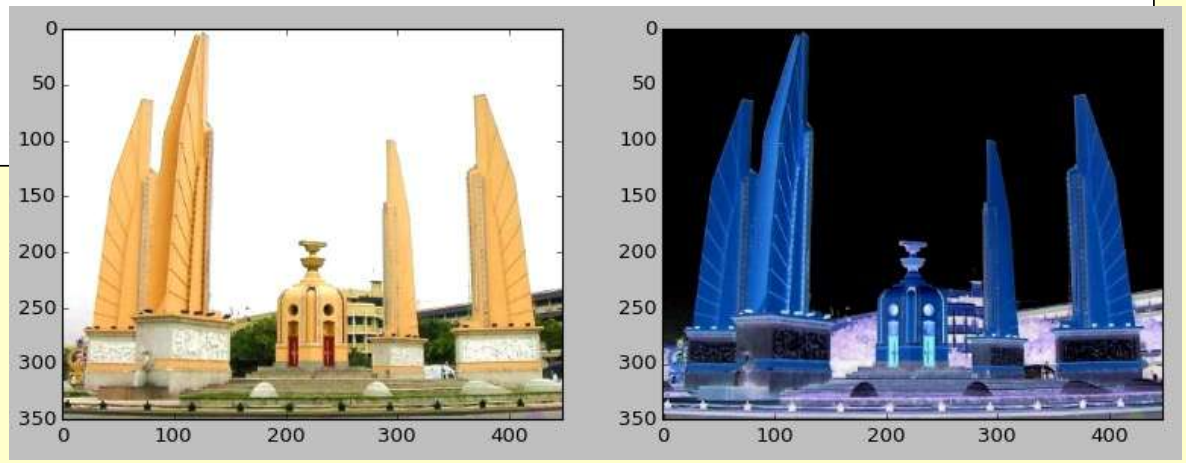


How's it changed?

Basic Image Processing (1): Negative Image

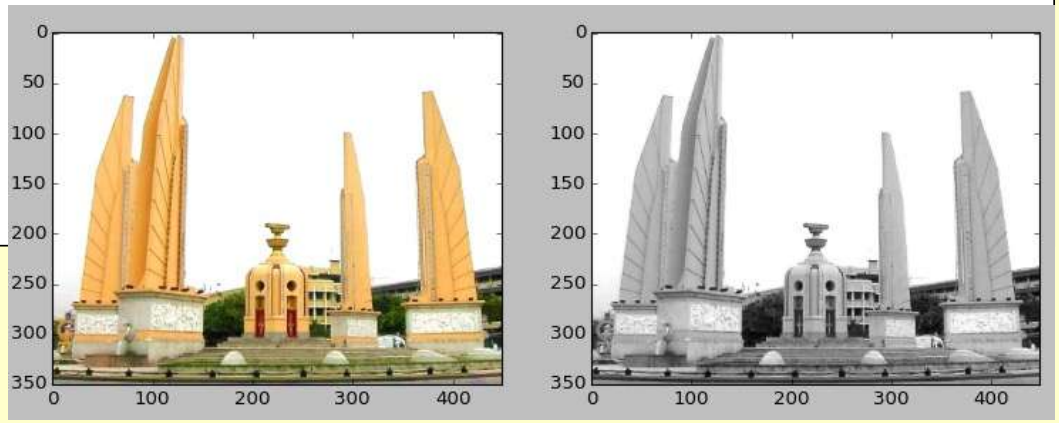
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

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 = Invert Tone
(White → Black, Yellow → Blue, ...)
broadcast & element-wise subtract

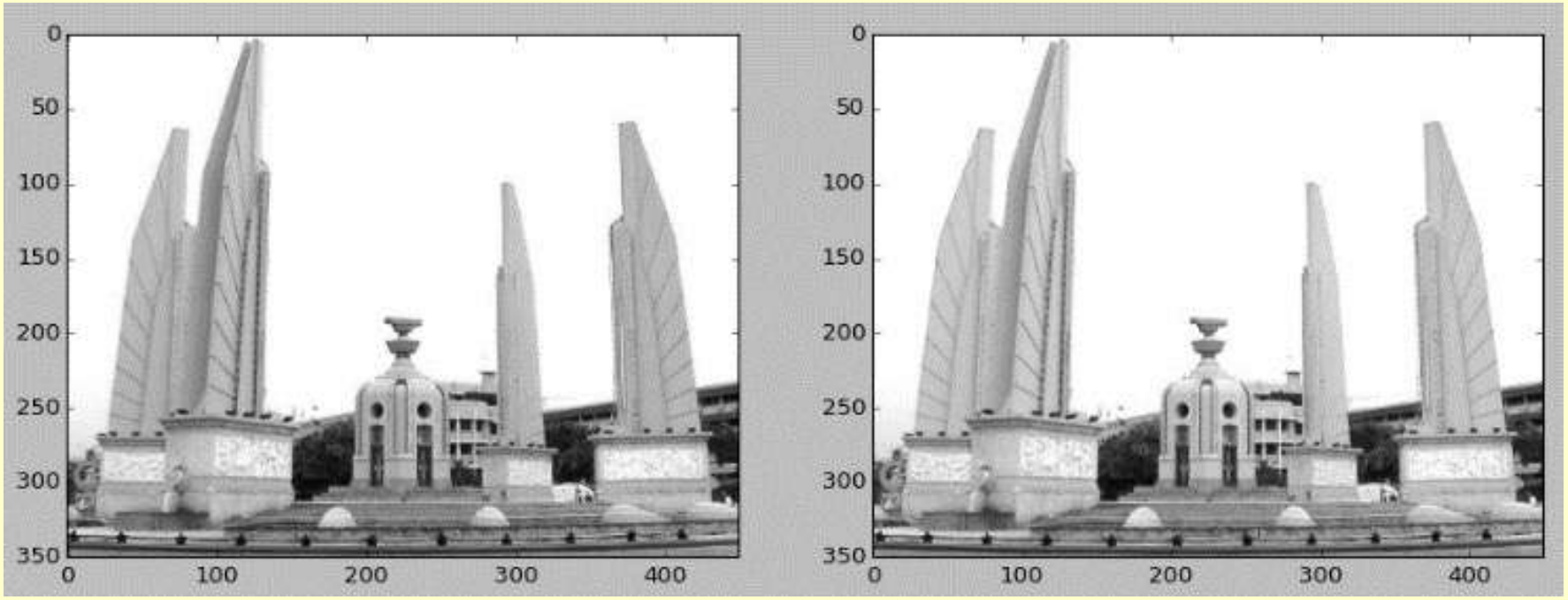


Basic Image Processing (2): Grayscale Image

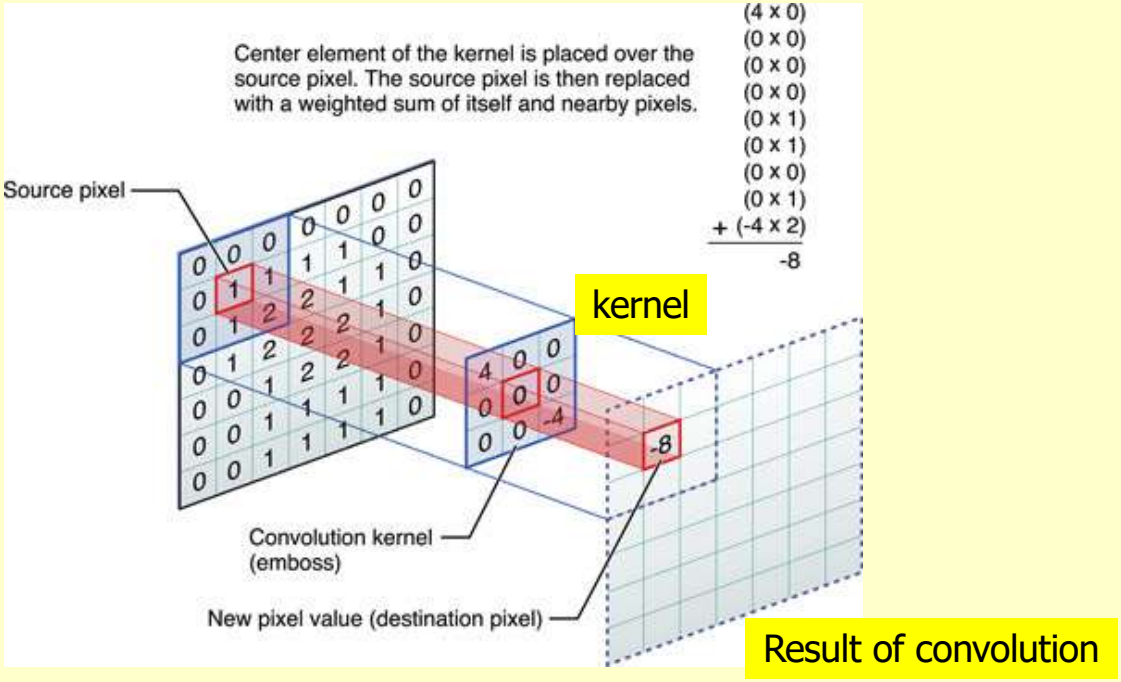
```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

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()
```



Another Grayscale Image: 0.299*R + 0.587*G + 0.114*B

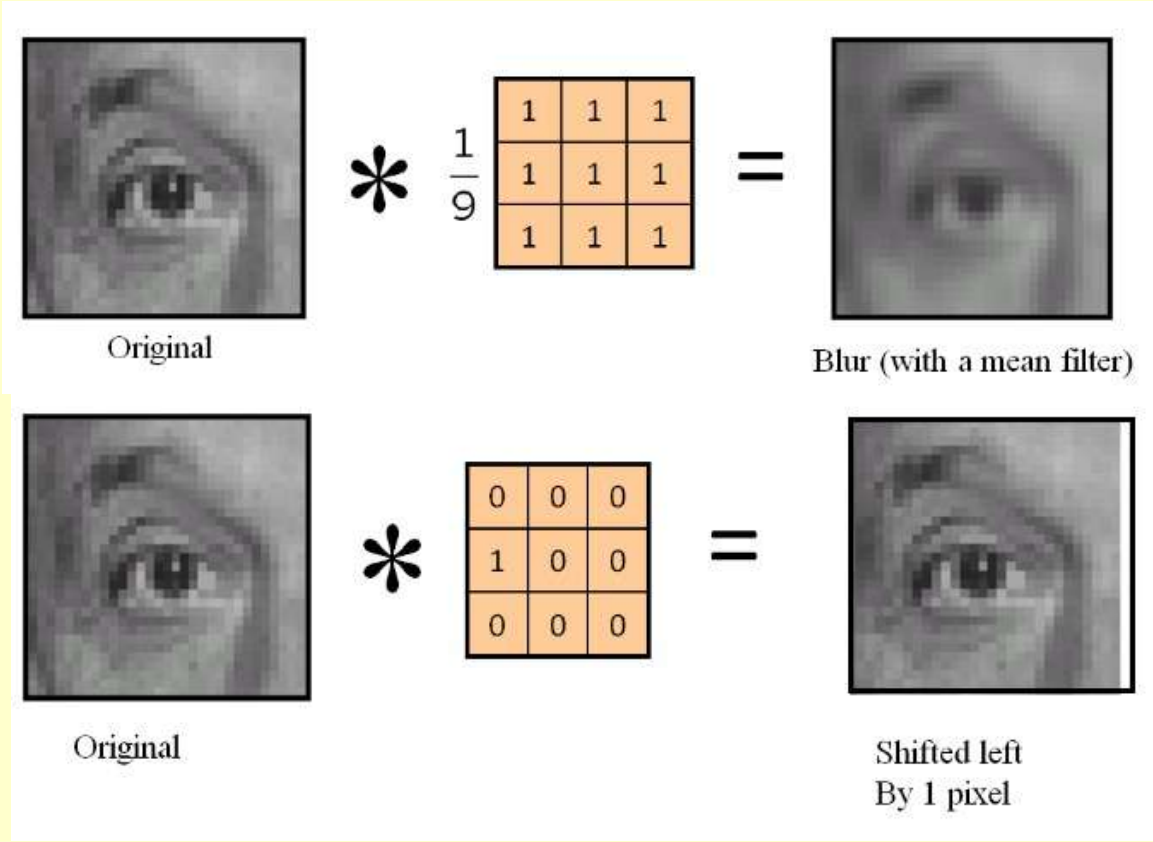


Convolution



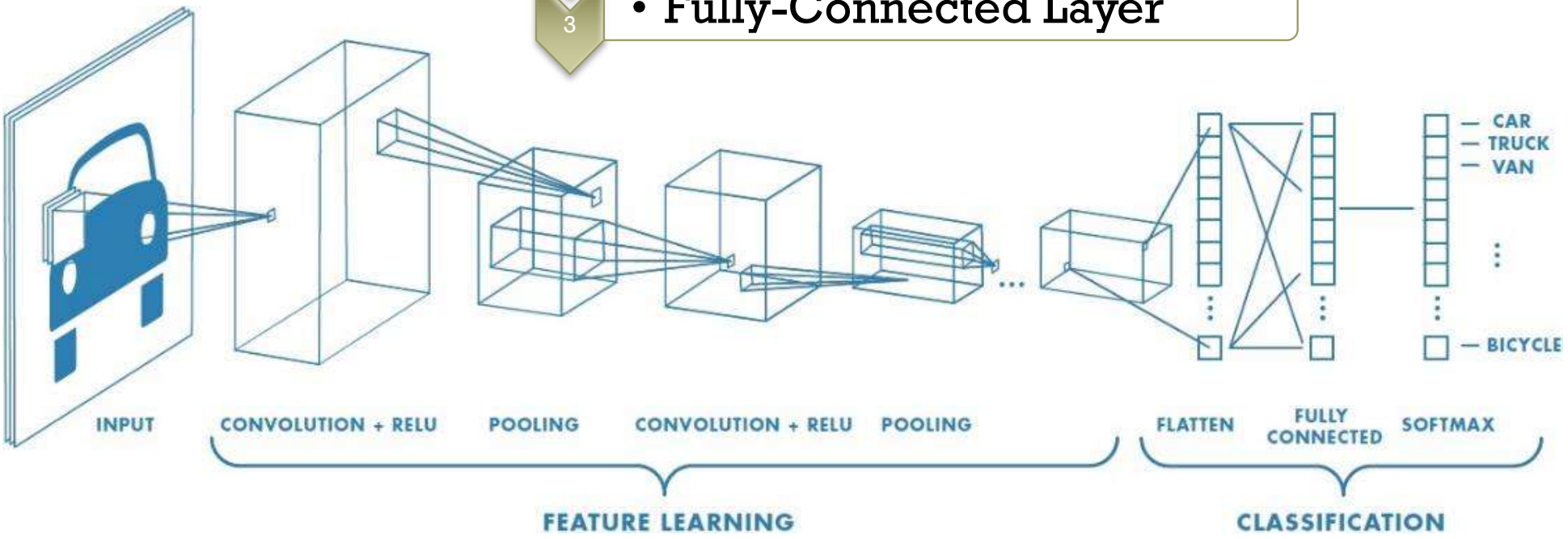
Changing the kernel (filter) results in different image processing outcomes

Image Convolution

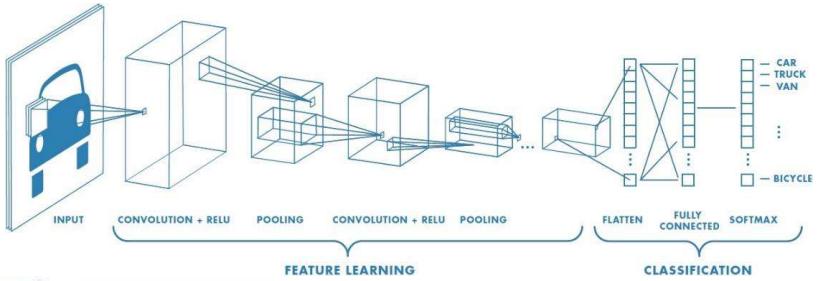


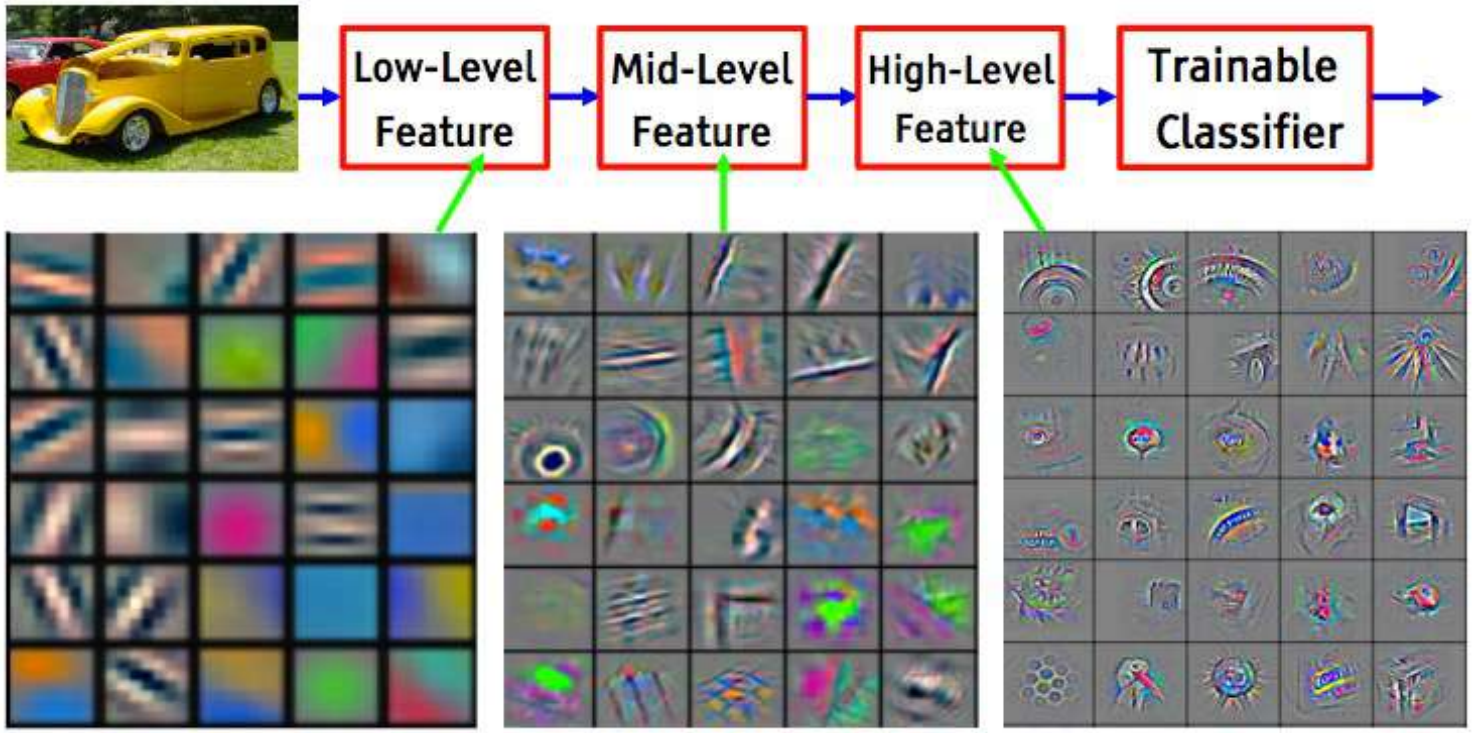
Convolutional Neural Networks (CNN)

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



CNN (cont.)

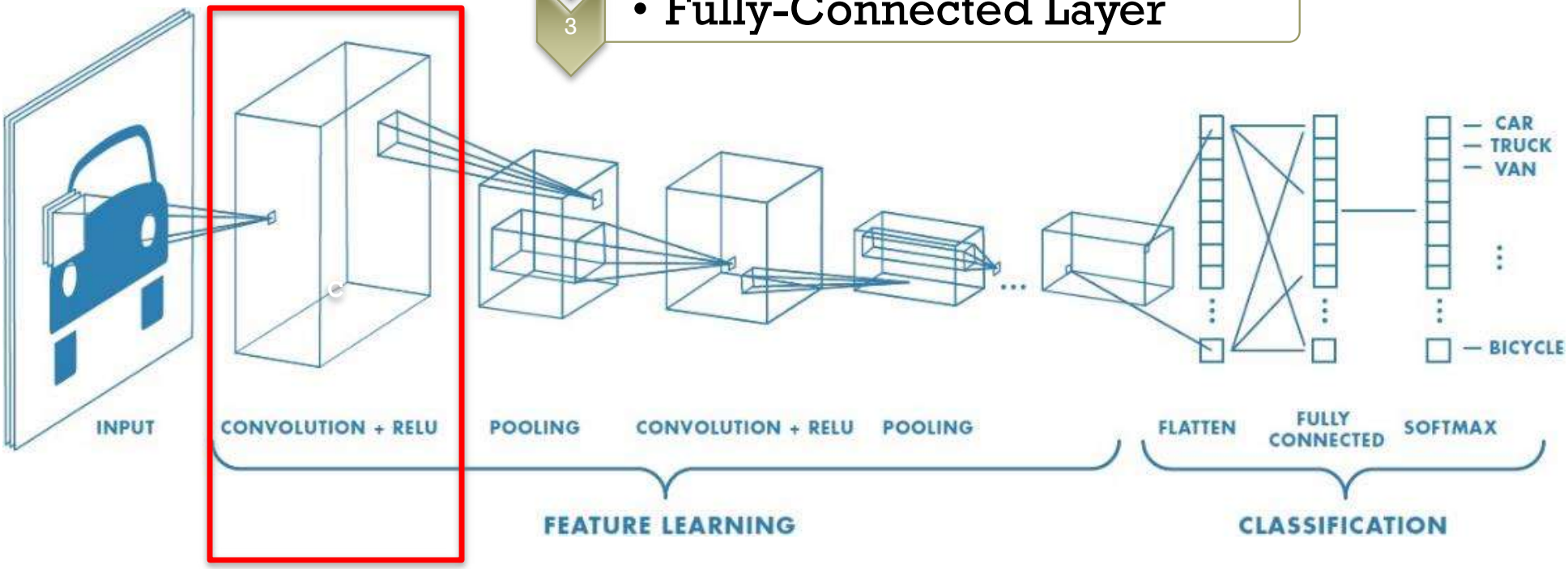

CAR
TRUCK
VAN
BICYCLE
INPUT
CONVOLUTION + RELU
POOLING
CONVOLUTION + RELU
POOLING
FLATTEN
FULLY CONNECTED
SOFTMAX
FEATURE LEARNING
CLASSIFICATION


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

Convolutional Neural Networks (CNN)

1) Convolutional Layer

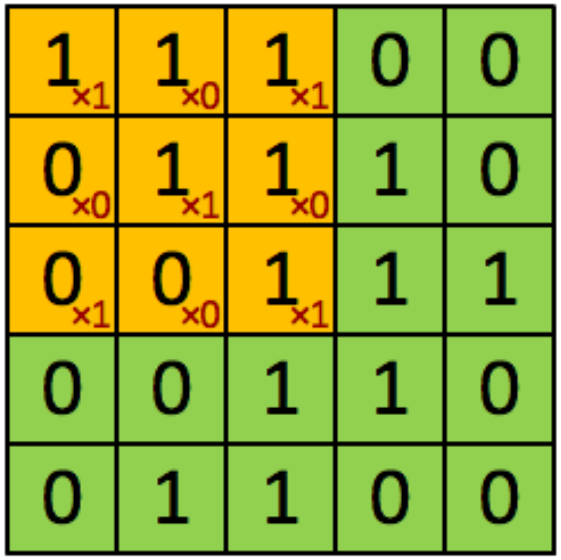
1. Convolutional Layer
2. Pooling Layer
3. Fully-Connected Layer



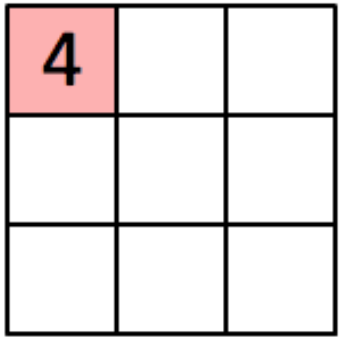
Layer 1: Convolutional Layer

(3*3 filter)

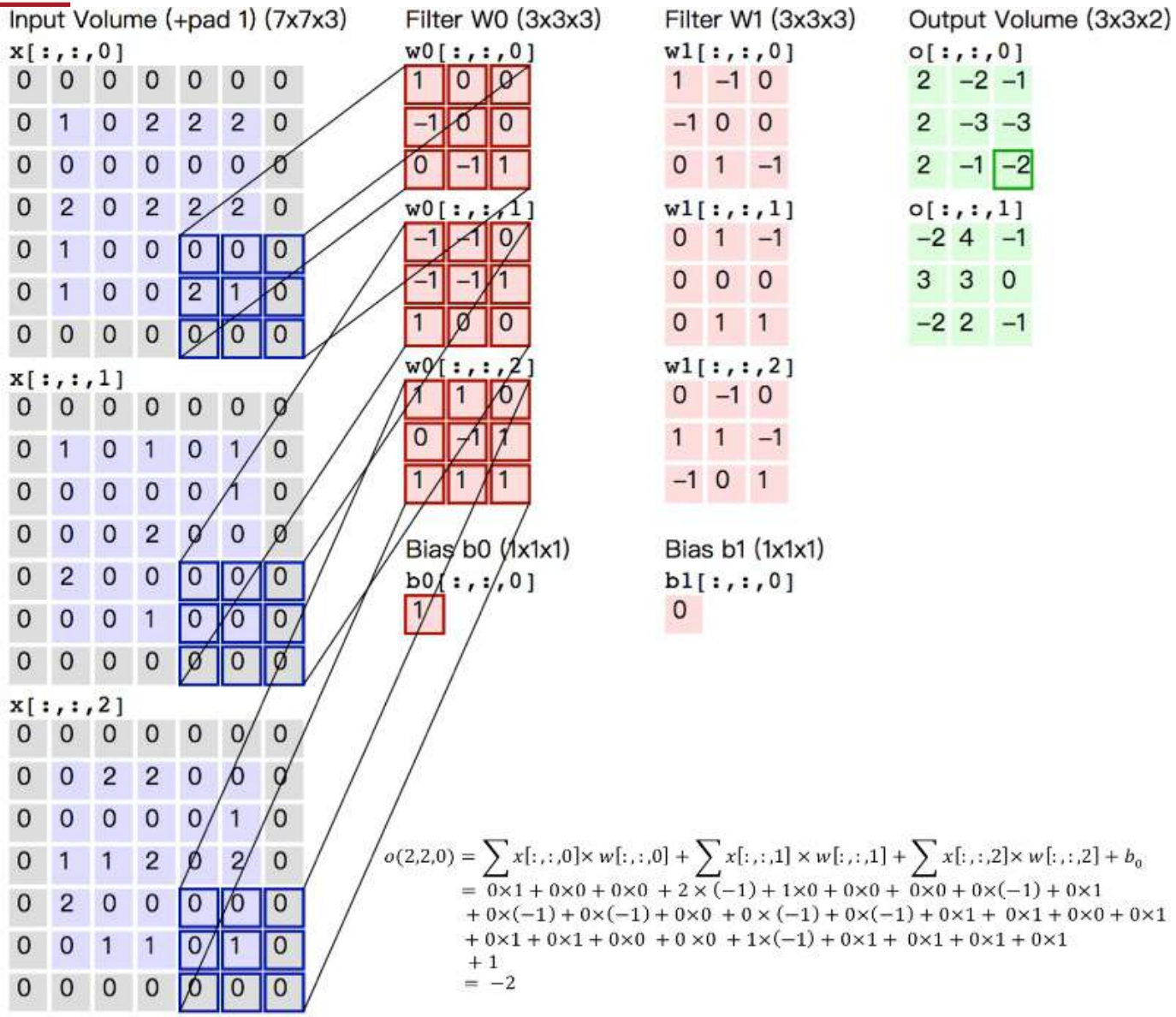
1	0	1
0	1	0
1	0	1



Image



Convolved Feature



$$o(2,2,0) = \sum x[:,:,0] \times w[:,:,0] + \sum x[:,:,1] \times w[:,:,1] + \sum x[:,:,2] \times w[:,:,2] + b_0$$
$$= 0\times1 + 0\times0 + 0\times0 + 2\times(-1) + 1\times0 + 0\times0 + 0\times0 + 0\times(-1) + 0\times1$$
$$+ 0\times(-1) + 0\times(-1) + 0\times0 + 0\times(-1) + 0\times(-1) + 0\times1 + 0\times1 + 0\times0 + 0\times1$$
$$+ 0\times1 + 0\times1 + 0\times0 + 0\times0 + 1\times(-1) + 0\times1 + 0\times1 + 0\times1 + 0\times1$$
$$+ 1$$
$$= -2$$

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		

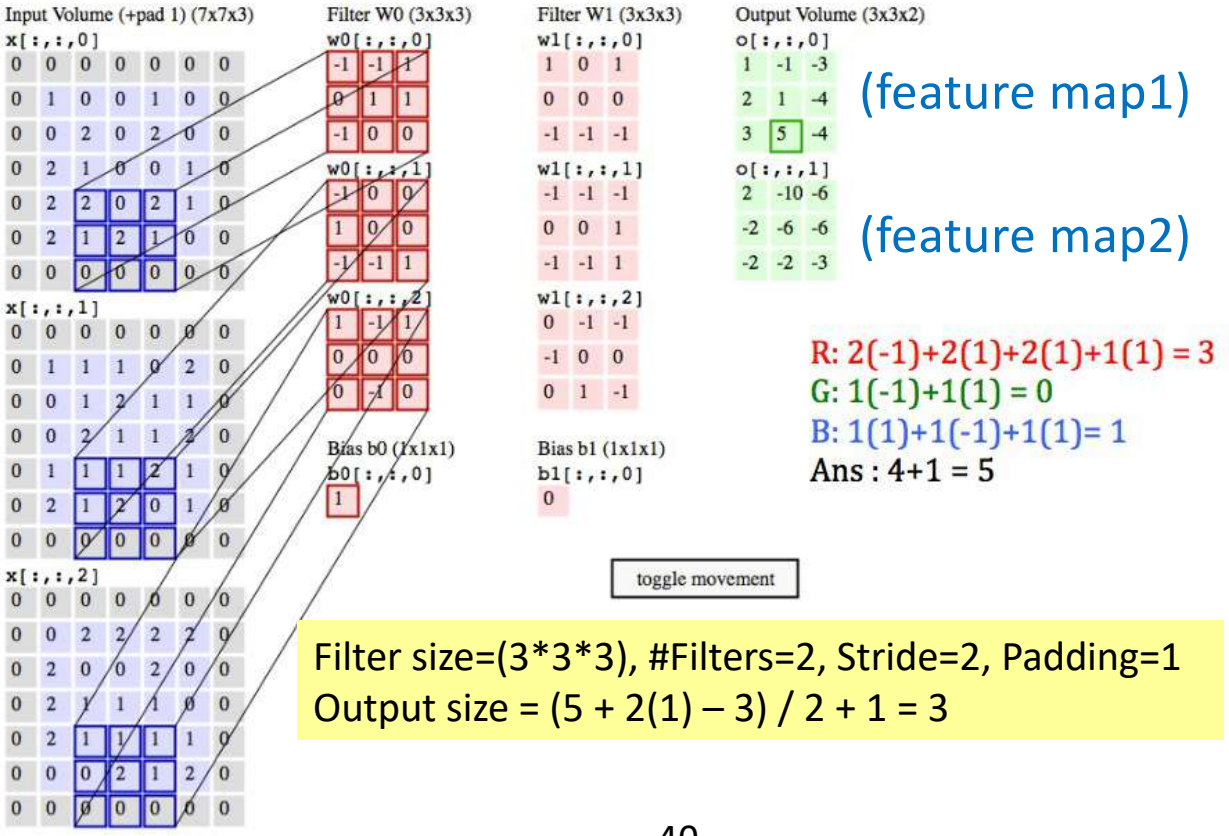
Convolved Feature (feature map)

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)

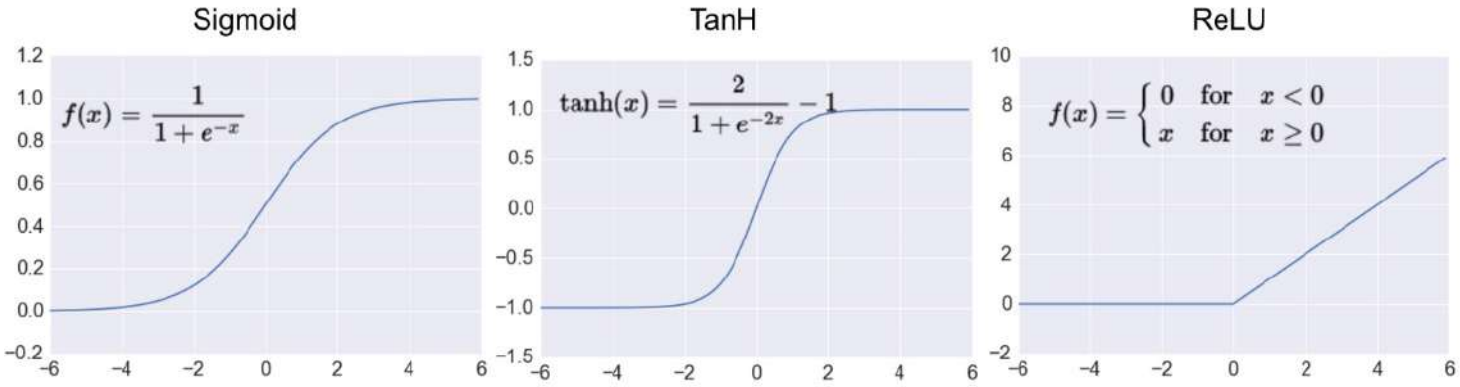
(feature map2)

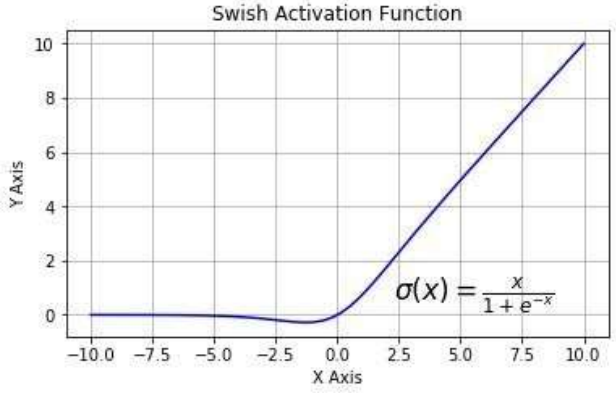
R: 2(-1)+2(1)+2(1)+1(1) = 3
G: 1(-1)+1(1) = 0
B: 1(1)+1(-1)+1(1)= 1
Ans : 4+1 = 5

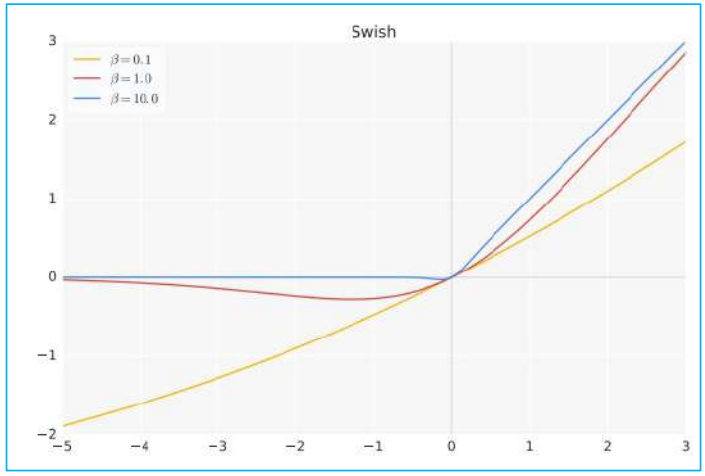
Filter size=(3*3*3), #Filters=2, Stride=2, Padding=1
Output size = (5 + 2(1) – 3) / 2 + 1 = 3

CNNs: 1) Convolutional Layer (cont.)

Non-Linear Activation Function



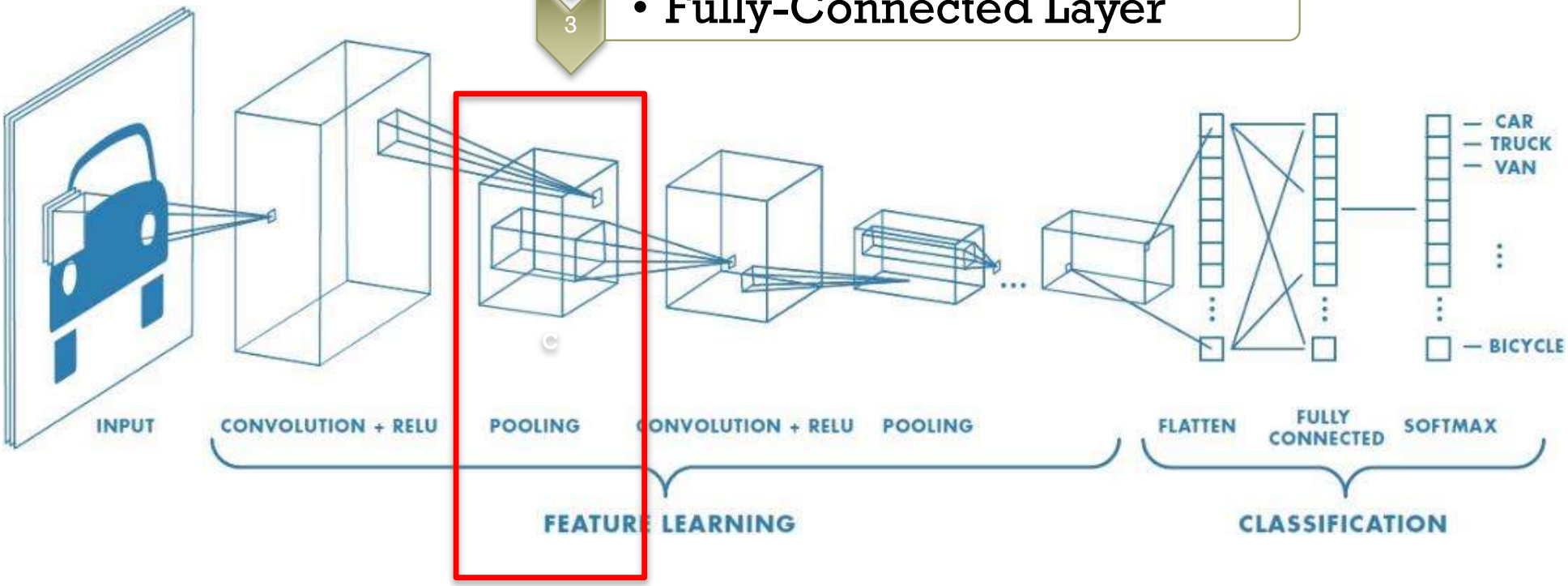




Convolutional Neural Networks (CNN)

2) Pooling Layer

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



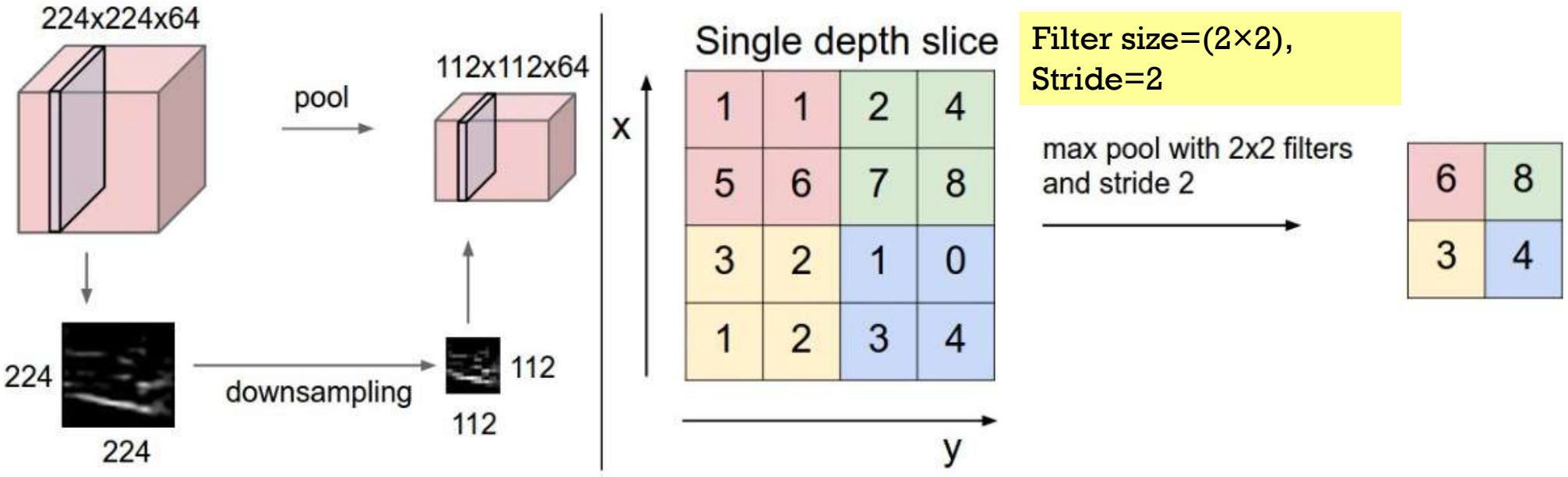
CNNs: 2) Pooling Layer

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

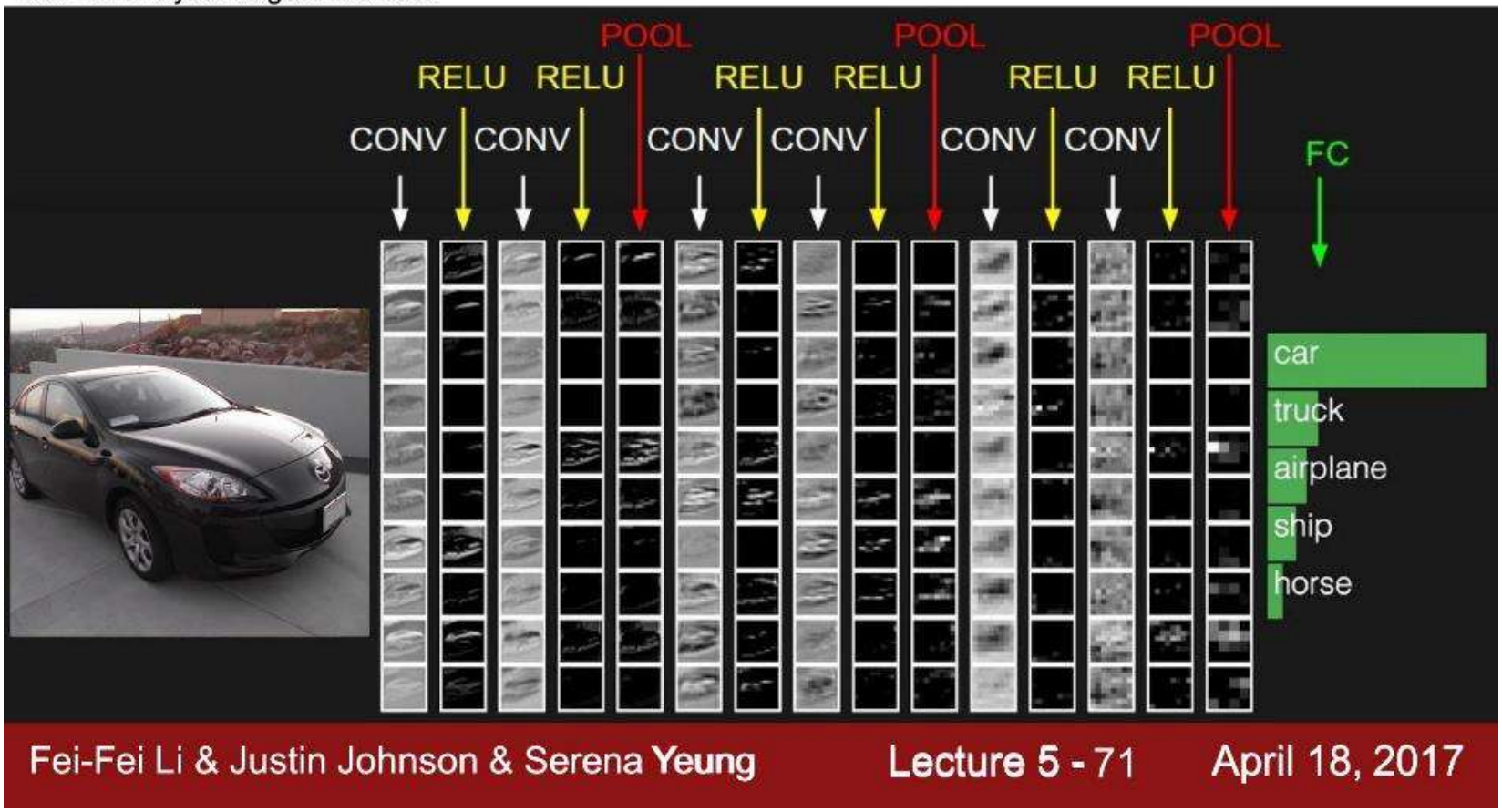
Summary: **2 parameters** for pooling layer

- Filter size, Stride

- Feature selection (reduction); downsampling



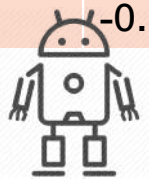
two more layers to go: POOL/FC



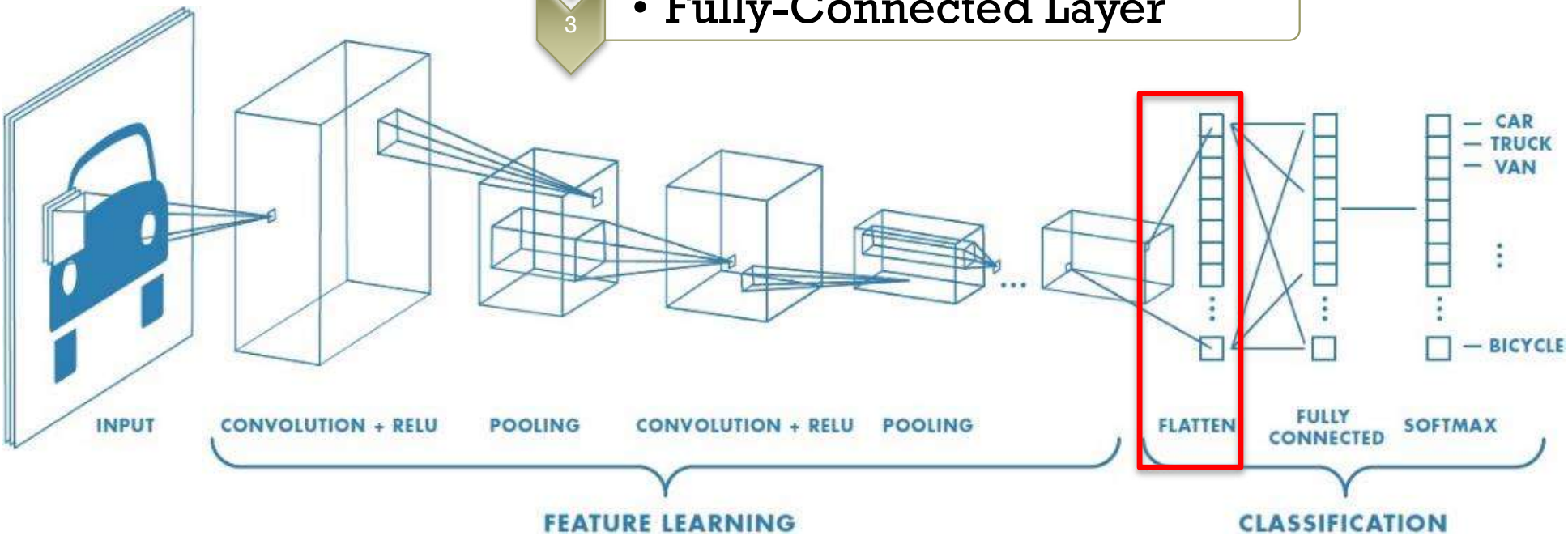
Convolutional Neural Networks

Embedding Vector

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



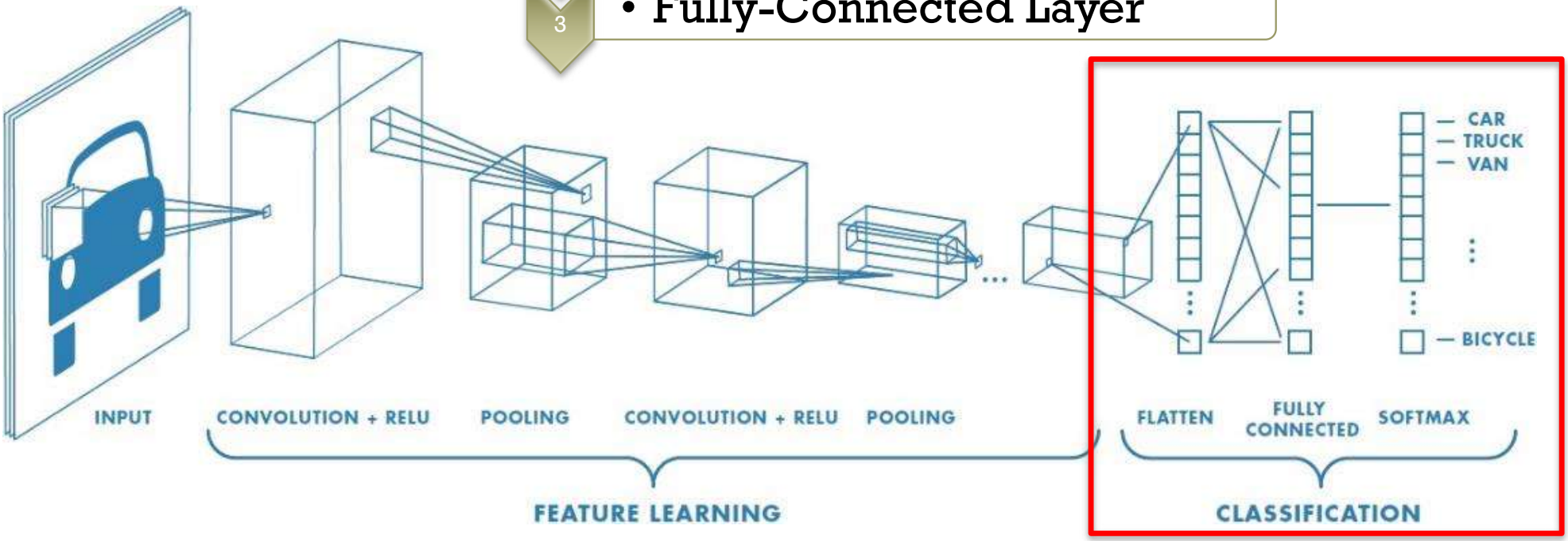
1. • Convolutional Layer
2. • Pooling Layer
3. • Fully-Connected Layer



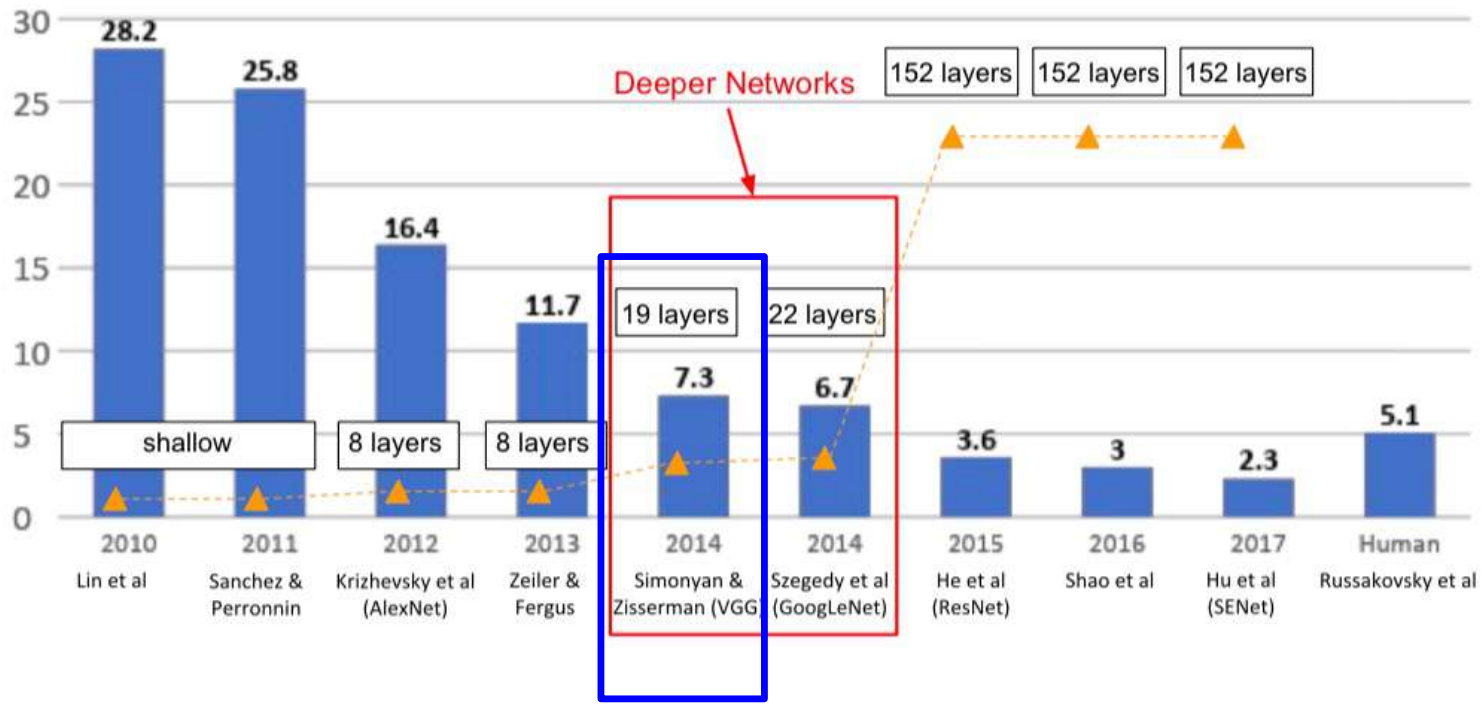
Convolutional Neural Networks (CNN)

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



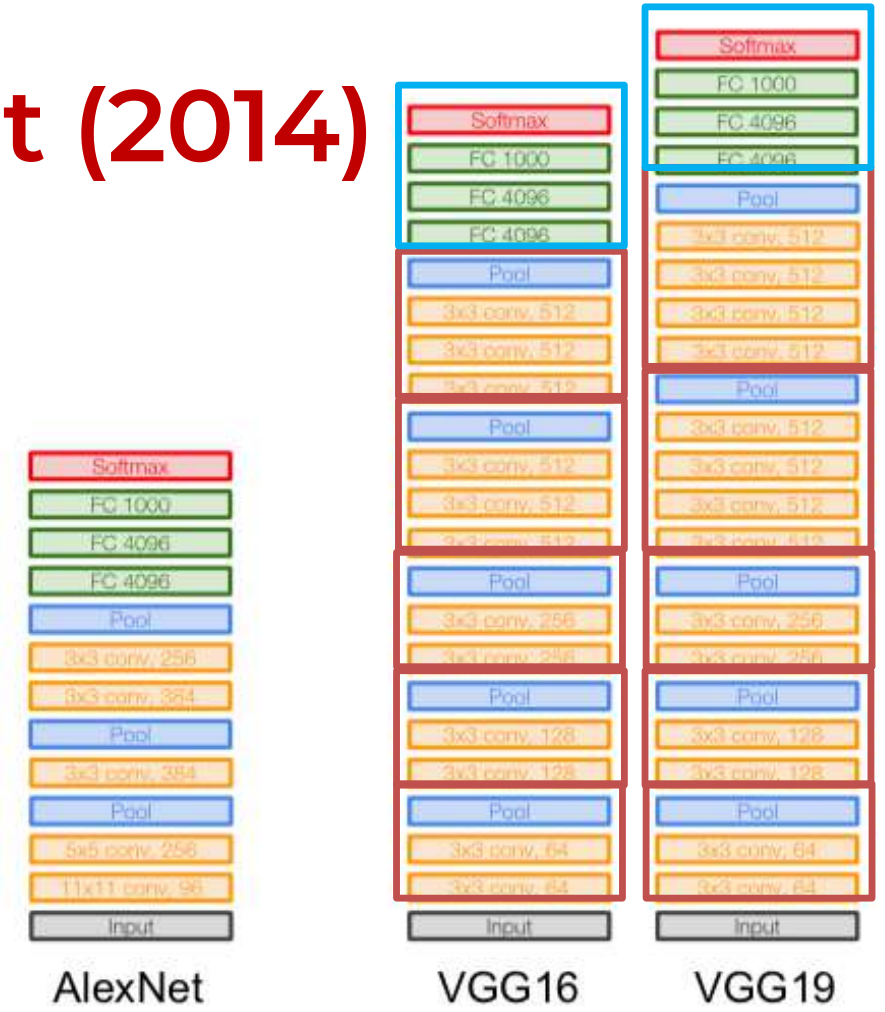
CNN Example: VGGNet (2014)

[Simonyan and Zisserman, 2014]

Small filters, Deeper networks

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

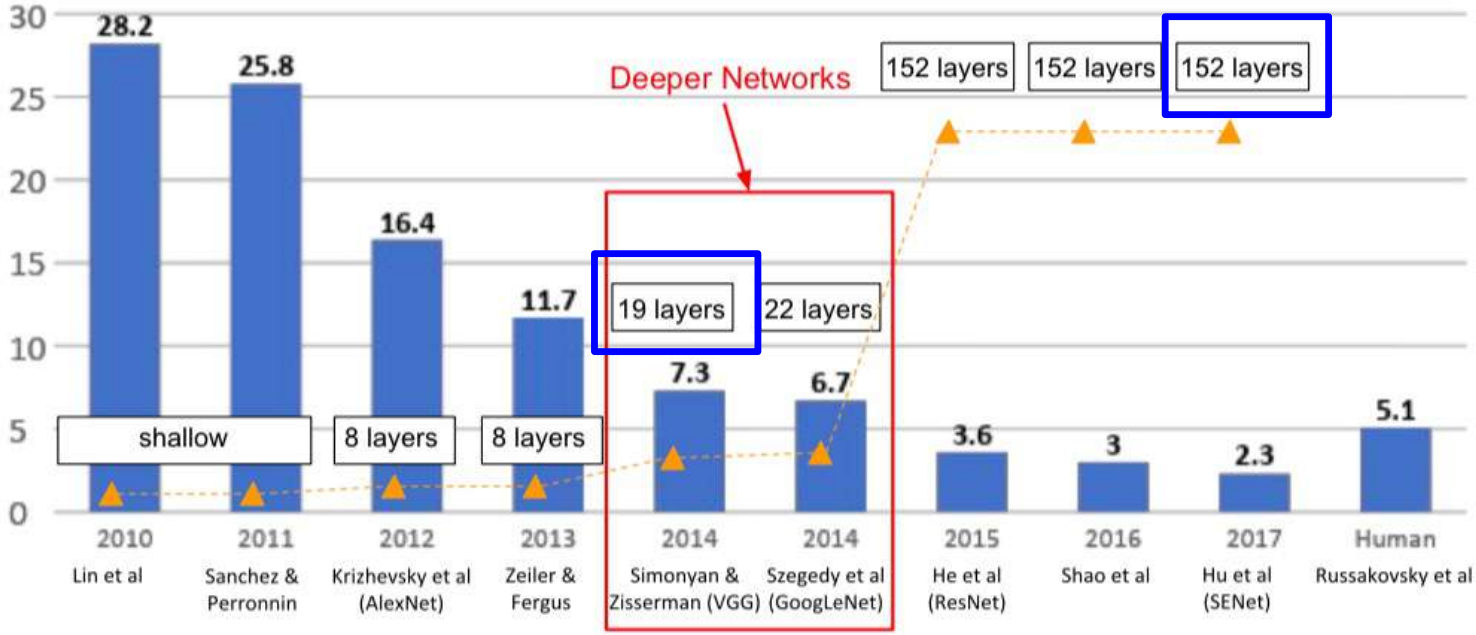
Only 3x3 CONV Stride 1, pad 1
And 2x2 MAX POOL stride 2



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

48

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

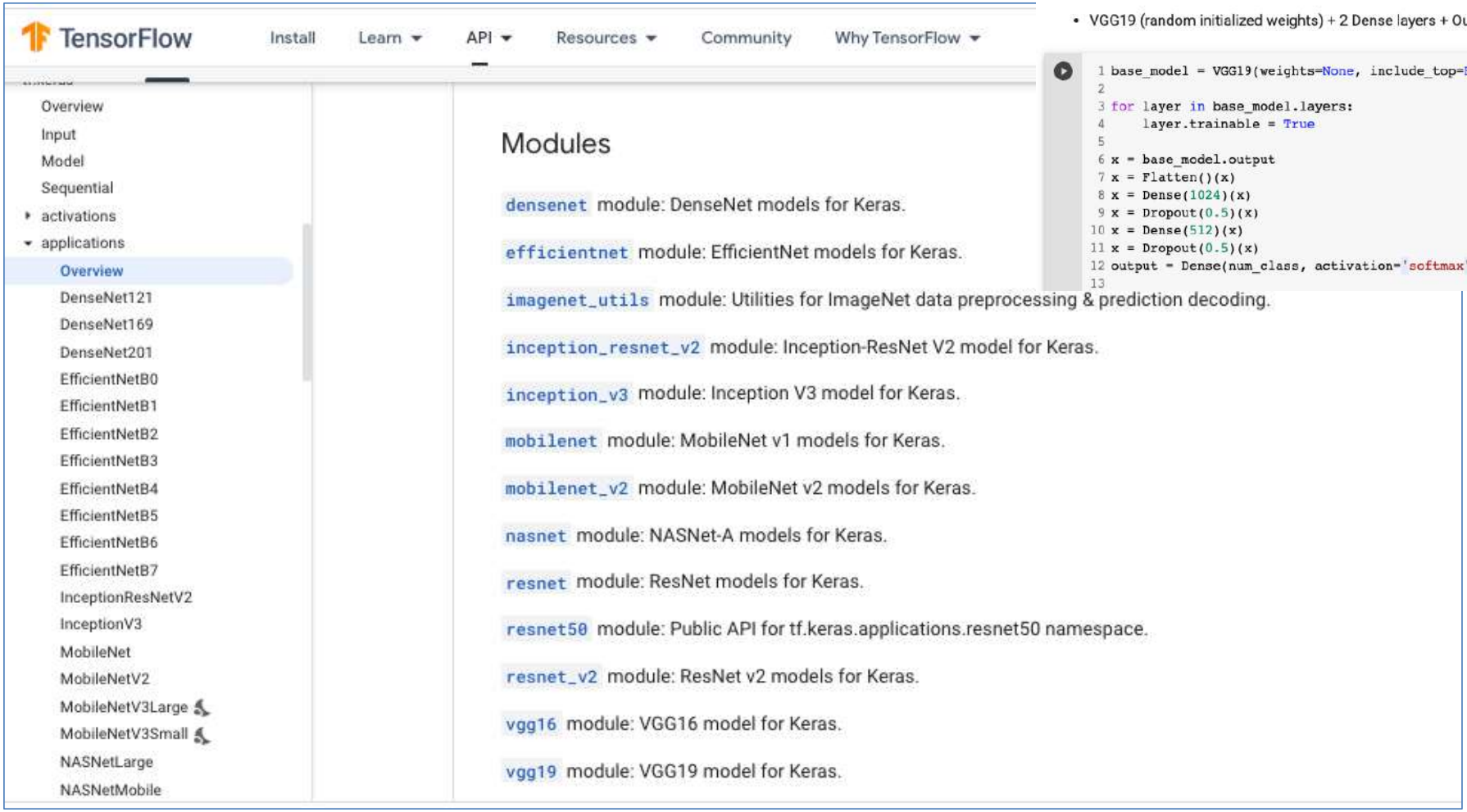


49

- 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

50

PyTorch

TORCHVISION.MODELS

The models subpackage contains definitions of models for addressing different tasks, including: image classification, pixelwise semantic segmentation, object detection, instance segmentation, person keypoint detection and video classification.

Classification

The models subpackage contains definitions for the following model architectures for image classification:

- AlexNet
- VGG
- ResNet
- SqueezeNet
- DenseNet
- Inception v3
- GoogLeNet
- ShuffleNet v2
- MobileNetV2
- MobileNetV3
- ResNeXt
- Wide ResNet
- MNASNet

We provide pre-trained models, using the PyTorch `torch.utils.model_zoo`. These can be constructed by passing `pretrained=True`:

```
import torchvision.models as models
resnet18 = models.resnet18(pretrained=True)
alexnet = models.alexnet(pretrained=True)
squeezenet = models.squeezenet1_0(pretrained=True)
vgg16 = models.vgg16(pretrained=True)
densenet = models.densenet161(pretrained=True)
inception = models.inception_v3(pretrained=True)
googlenet = models.googlenet(pretrained=True)
shufflenet = models.shufflenet_v2_x1_0(pretrained=True)
mobilenet_v2 = models.mobilenet_v2(pretrained=True)
mobilenet_v3_large = models.mobilenet_v3_large(pretrained=True)
mobilenet_v3_small = models.mobilenet_v3_small(pretrained=True)
resnext50_32x4d = models.resnext50_32x4d(pretrained=True)
wide_resnet50_2 = models.wide_resnet50_2(pretrained=True)
mnasnet = models.mnasnet1_0(pretrained=True)
```

https://pytorch.org/vision/0.9/models.html

v1.1.0 ▼

pytorch-transformers

Star 131,473

Search docs

Pytorch-Transformers

PyTorch-Transformers is a library of state-of-the-art pre-trained models for Natural Language Processing (NLP).

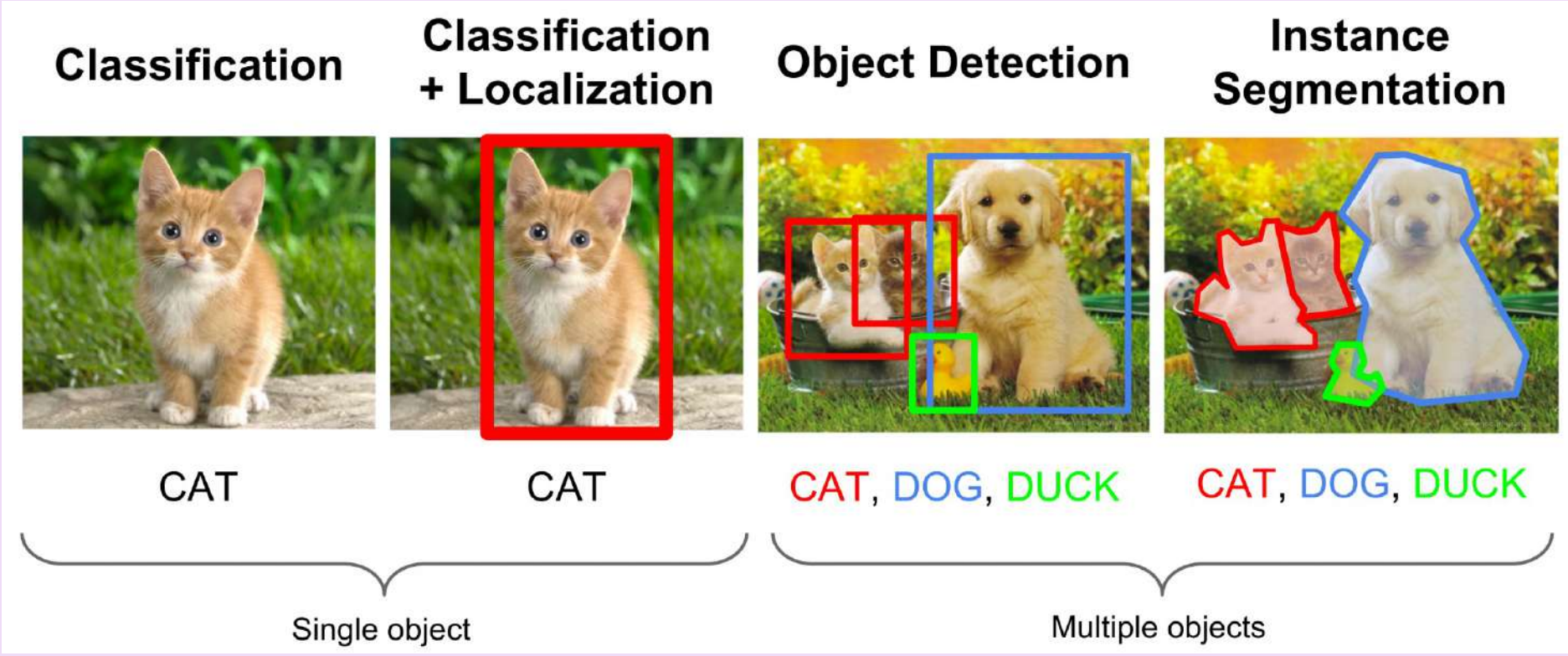
The library currently contains PyTorch implementations, pre-trained model weights, usage scripts and conversion utilities for the following models:

1. BERT (from Google) released with the paper BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding by Jacob Devlin, Ming-Wei Lee and Kristina Toutanova.
2. GPT (from OpenAI) released with the paper Improving Language Understanding by Generative Pre-Training by Alec Radford, Karthik Narasimhan, Tim Salim Sutskever.
3. GPT-2 (from OpenAI) released with the paper Language Models are Unsupervised Multitask Learners by Alec Radford*, Jeffrey Wu*, Rewon Child, David Lua and Ilya Sutskever**.
4. Transformer-XL (from Google/CMU) released with the paper Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context by Zihang Dai*, Z Yang, Jaime Carbonell, Quoc V. Le, Ruslan Salakhutdinov.
5. XLNet (from Google/CMU) released with the paper XLNet: Generalized Autoregressive Pretraining for Language Understanding by Zhilin Yang*, Zihang Dai Jaime Carbonell, Ruslan Salakhutdinov, Quoc V. Le.
6. XLM (from Facebook) released together with the paper Cross-lingual Language Model Pretraining by Guillaume Lample and Alexis Conneau.

Notes

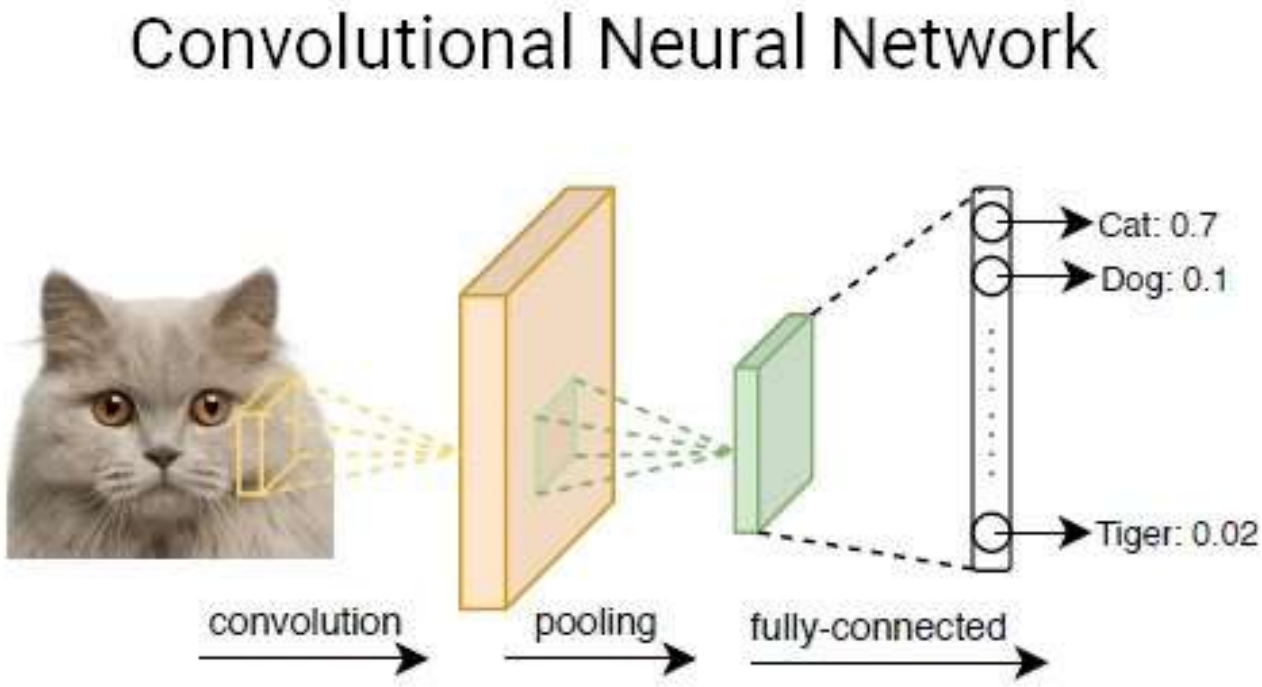
- Installation

Image Classification Tasks

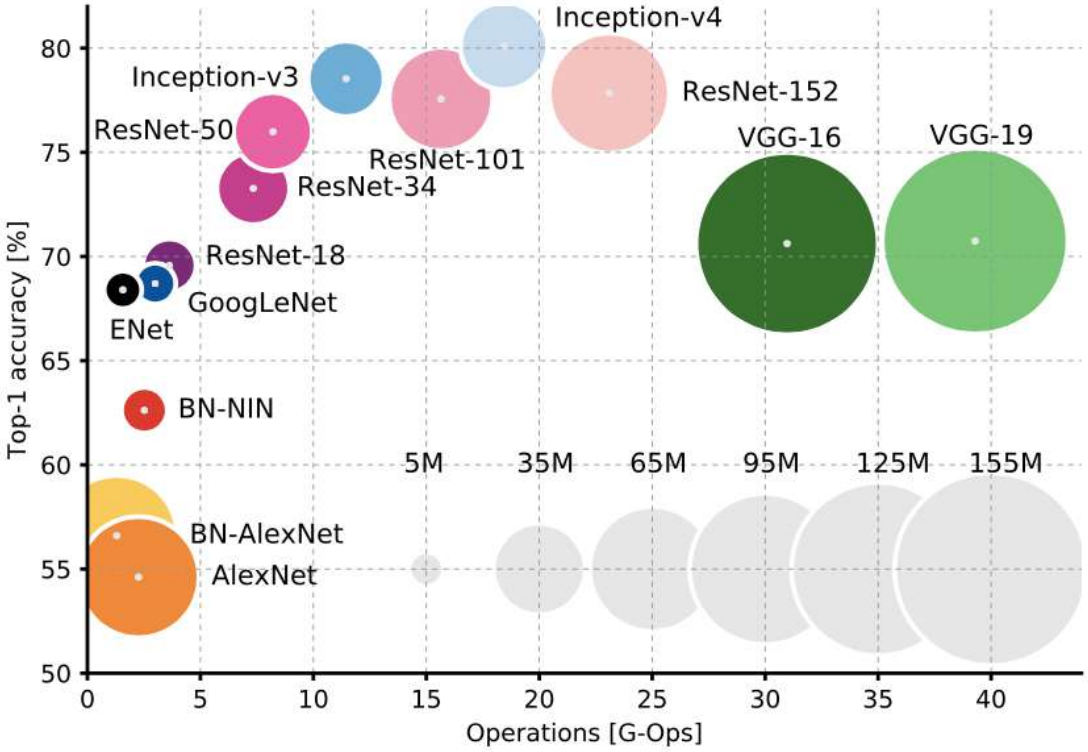


https://analyticsindiamag.com/top-5-image-classification-research-papers-every-data-scientist-should-know/

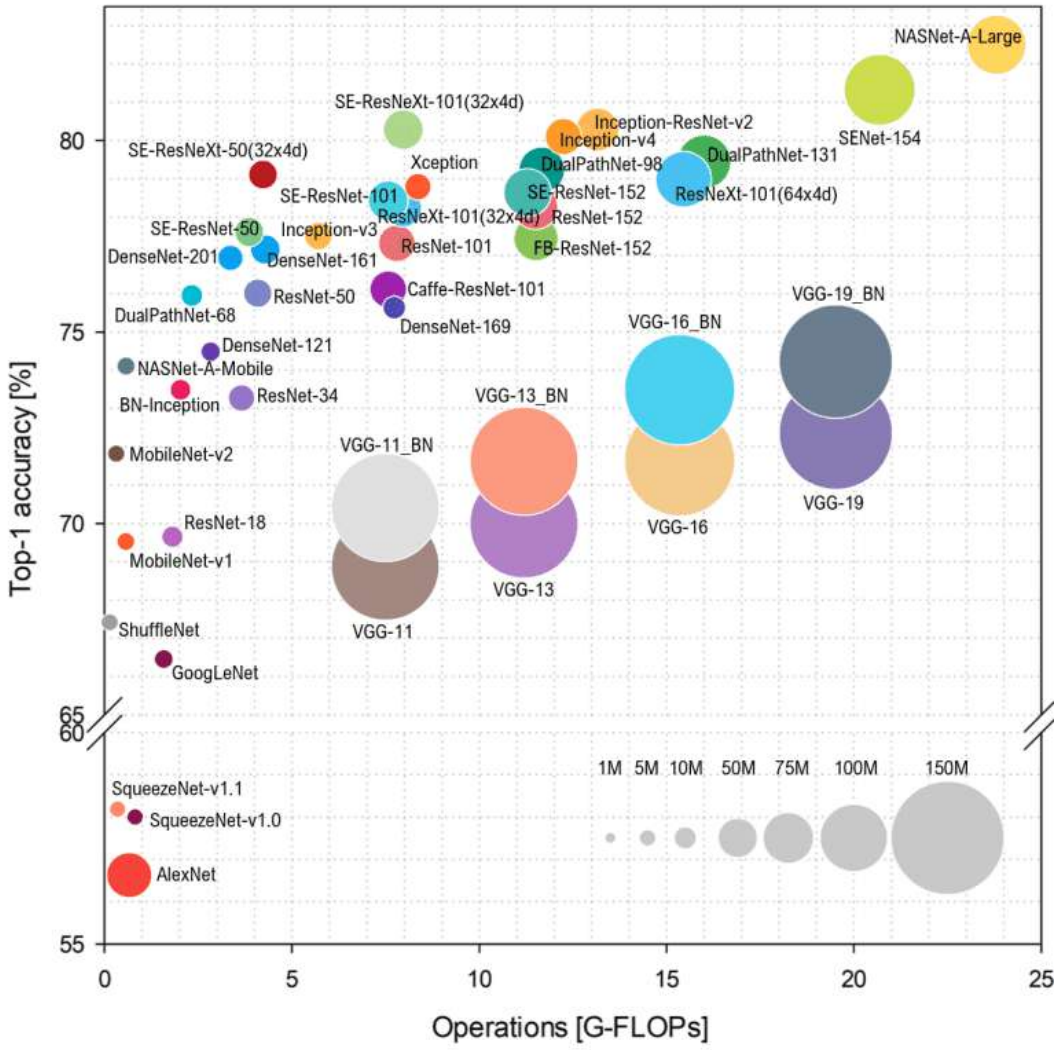
Image Classification (recap)



SOTA of Image Classification



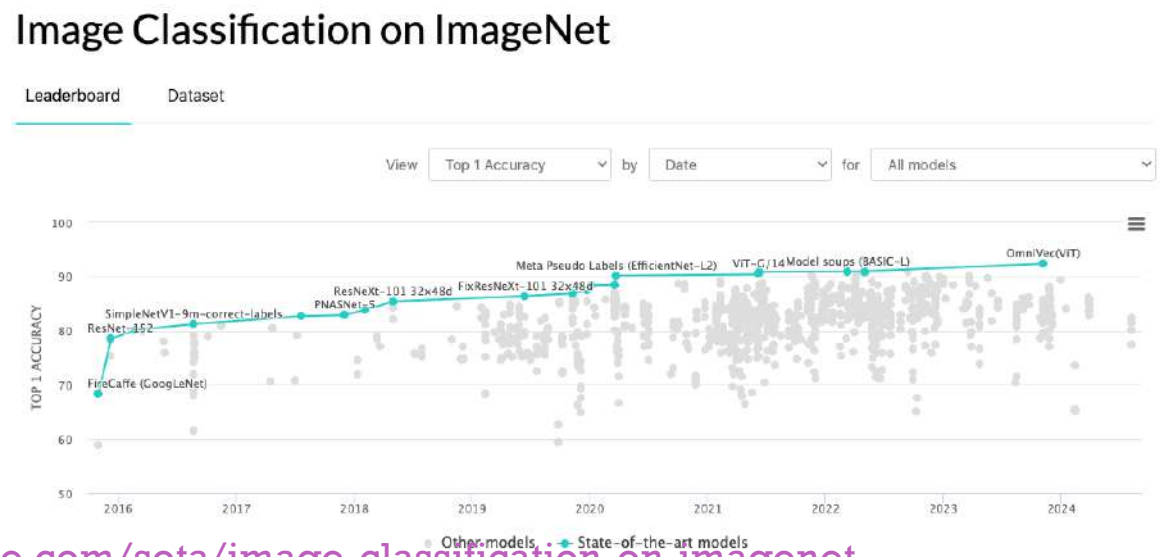
https://blog.csdn.net/qq_34216467/article/details/83061692



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

Common Image Classification Models

Model	Year	Team / Organization
ResNet	2015	Microsoft Research
MobileNet	**2017**	**Google**
EfficientNet	**2019**	**Google**
Vision Transformer (ViT)	2020	Google Research
Swin Transformer	2021	Microsoft Research
ConvNeXt	2022	Meta AI



https://paperswithcode.com/sota/image-classification-on-imagenet

Image classfication dashboard (2025)

- CNN, Transformer (ViT, VLM)
  - 2021–2022 → CNNs (EfficientNetV2) were strong, but Transformers started dominating.
  - 2022–2023 → Vision–Language models (VLM) like CoCa + giant ViTs (EVA, Model Soups) pushed past 90%.
  - 2024 → Specialized ViT variants (DaViT, Florence-CoSwin) consolidated.
  - 2025 → Large-scale foundation models (DINOv3) + fine-tuned CoCa remain at the frontier.
- Note:
  - CoCa / CLIP → Vision–Language (image + text), trained with image–text pairs, suitable for cross-modal tasks such as captioning and retrieval
  - DINOv3 → Vision-only (image), self-supervised ViT, backbone for computer vision

https://paperswithcode.com/sota/image-classification-on-imagenet

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

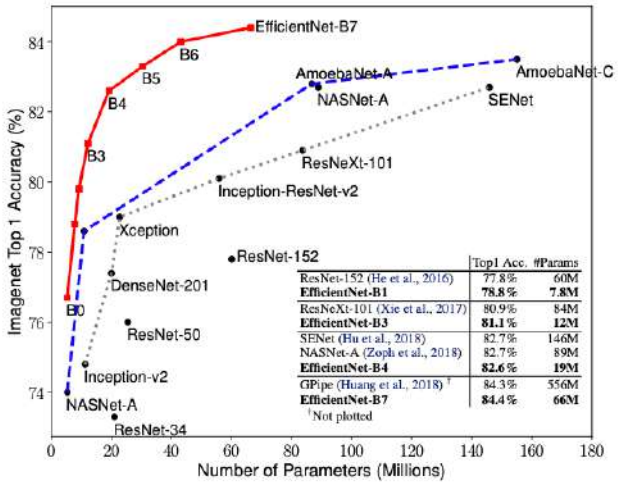
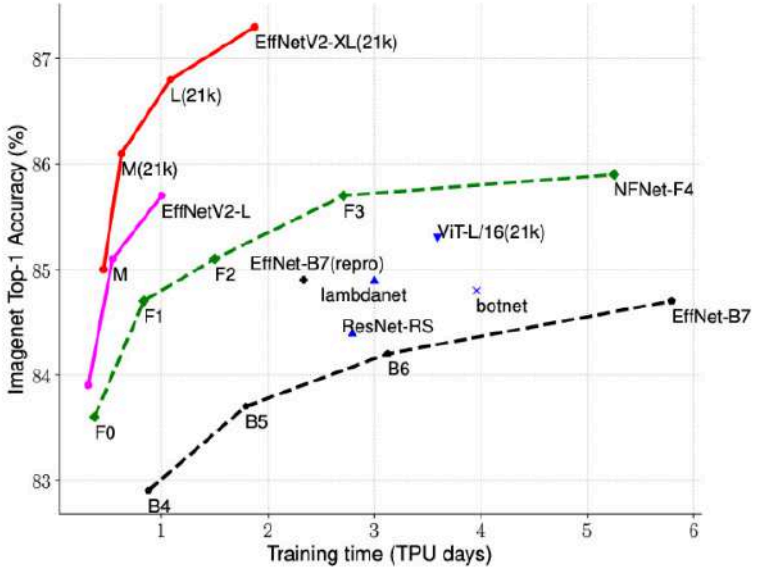


Figure 1. Model Size vs. ImageNet Accuracy. All numbers are



(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

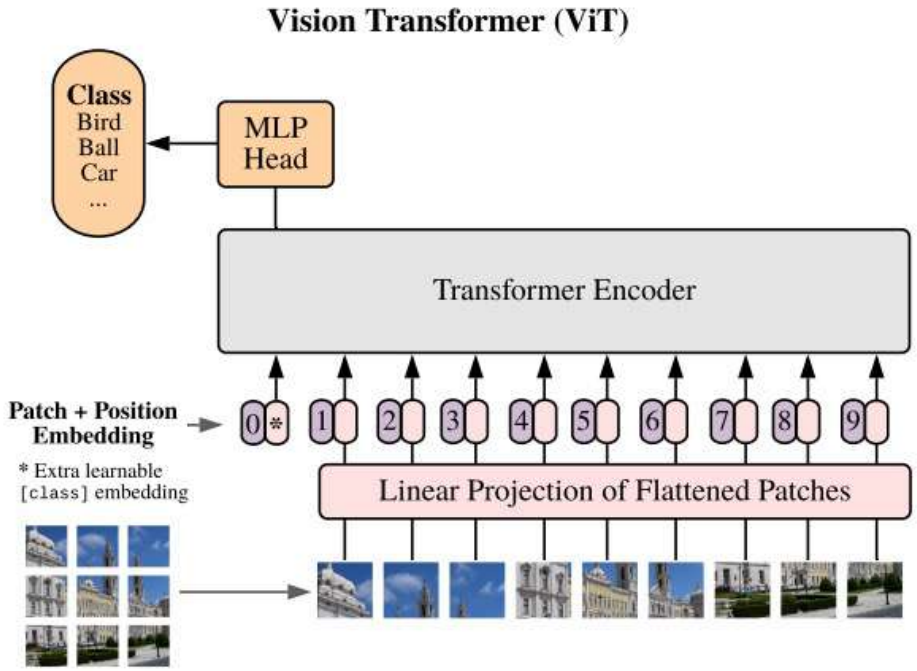
(b) Parameter efficiency.

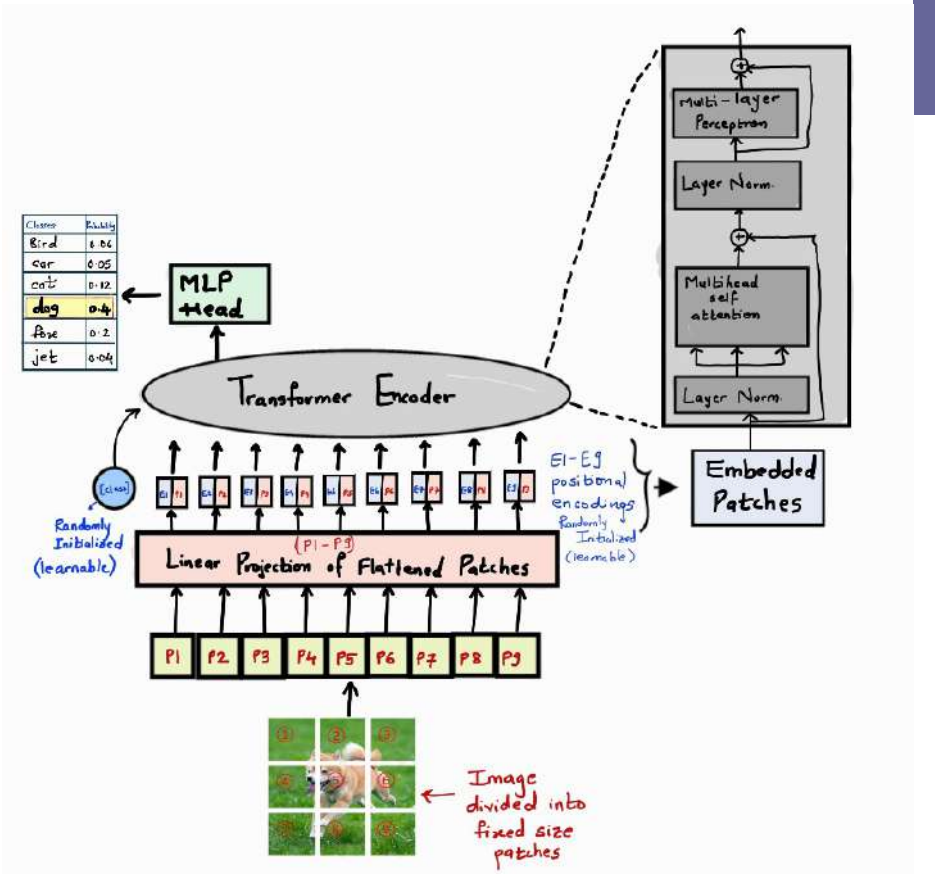
http://proceedings.mlr.press/v97/tan19a/tan19a.pdf

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

Vision Transformer (ViT) (ICLR2021)

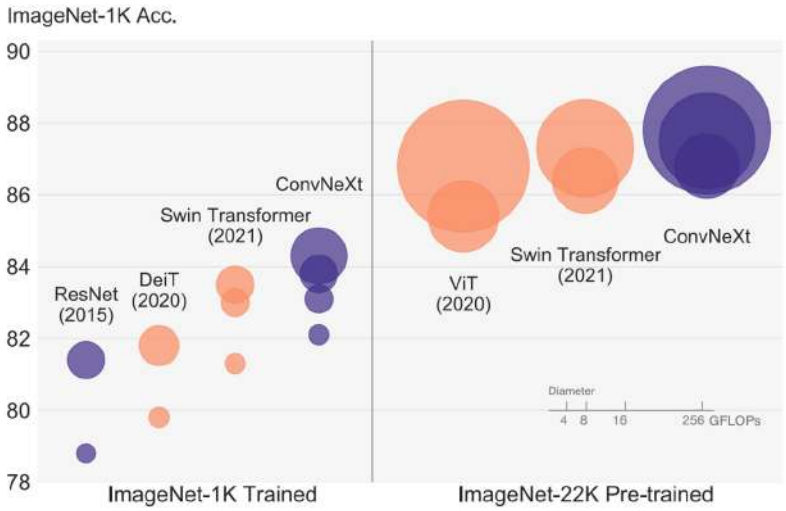
Hugging Face

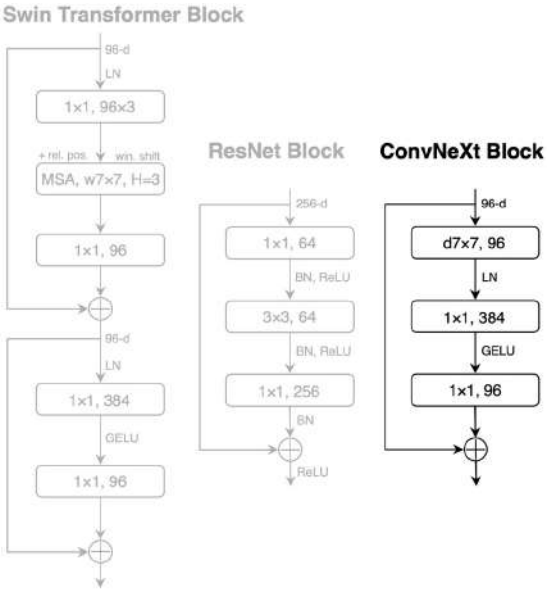


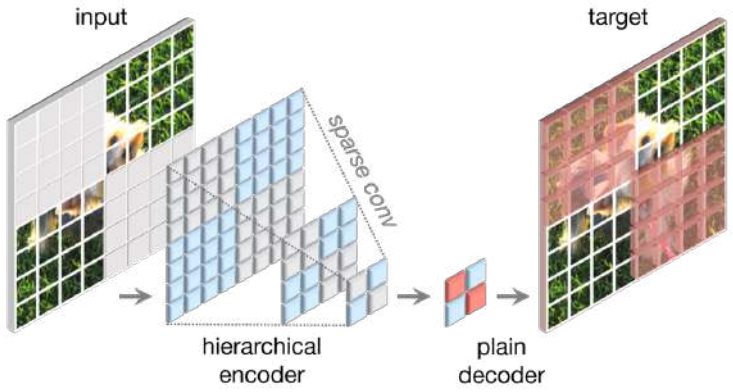


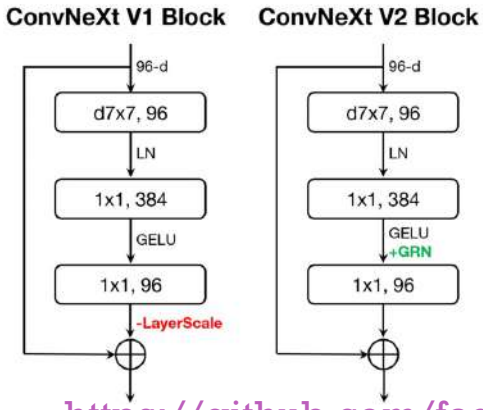
https://medium.com/machine-intelligence-and-deep-learning-lab/vit-vision-transformer-cc56c8071a20

ConvNeXt (CVPR2022) → ConvNeXt V2 (2023)









https://github.com/facebookresearch/ConvNeXt-V2

https://github.com/facebookresearch/ConvNeXt

Open Source

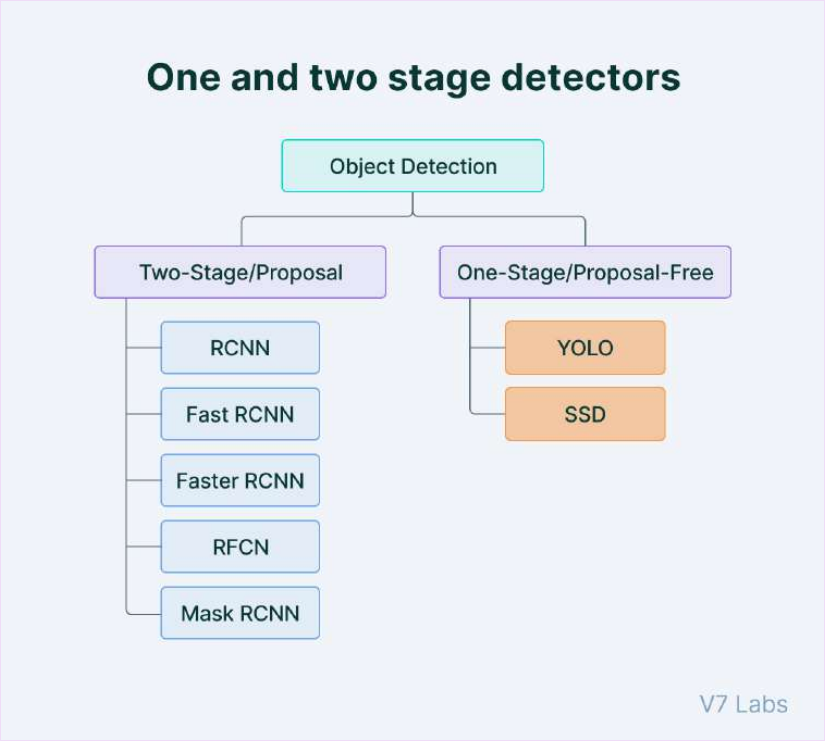
DINOv3: Self-supervised learning for vision at unprecedented scale

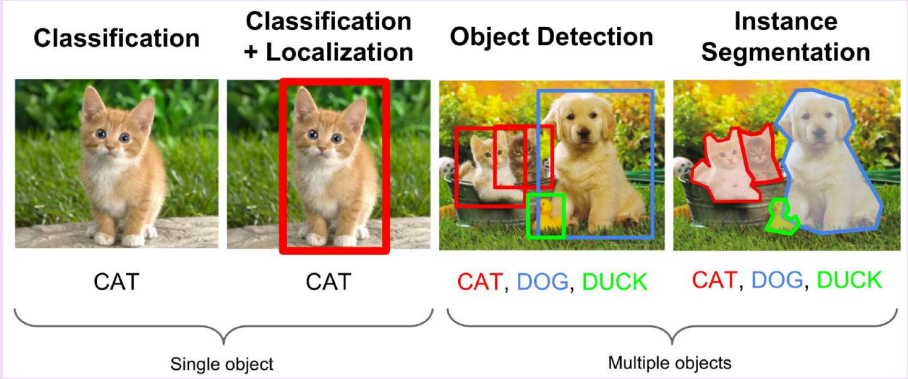
August 14, 2025 · 11 minute read

- DINOv3: Self-Supervised Vision Transformer (Meta AI, 2025)
- Vision-only (encoder-only) backbone (not multimodal, unlike CLIP/CoCa).
- Self-supervised learning → no human labels needed.
- Massive scale: up to 7B parameters, trained on 1.7B images.
- Key innovations:
  - Gram Anchoring → stabilizes feature learning.
  - Axial RoPE → improved positional encoding.
  - Resolution robustness → better handling of varied image sizes.
- Strengths:
  - Provides general-purpose visual representations.
  - Excels in downstream tasks (classification, detection, segmentation).
  - Foundation model role for vision, similar to LLMs in NLP.

Link

SOTA of Object Detection

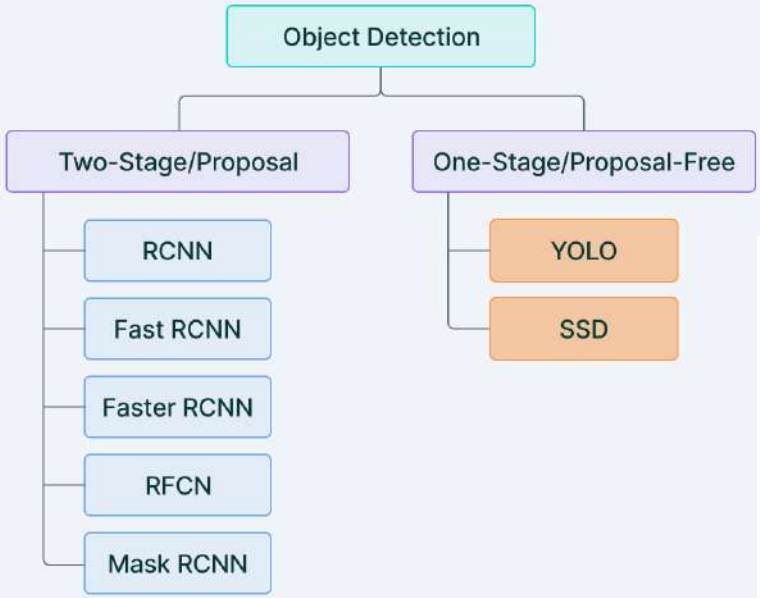




https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

Object Detection

One and two stage detectors



		units	
		absolute	normalized
corner coordinate	(left, top, right, bottom)	Pascal VOC	Albumentations
	(left, top, width, height)	COCO	
	(center_x, center_y, width, height)	CreateML	YOLO

https://dragoneye.ai/blog/a-guide-to-bounding-box-formats/

YOLO Open Images in New York

YOLO Open Images in New York

Joseph Redmon
6.95K subscribers
Subscribe

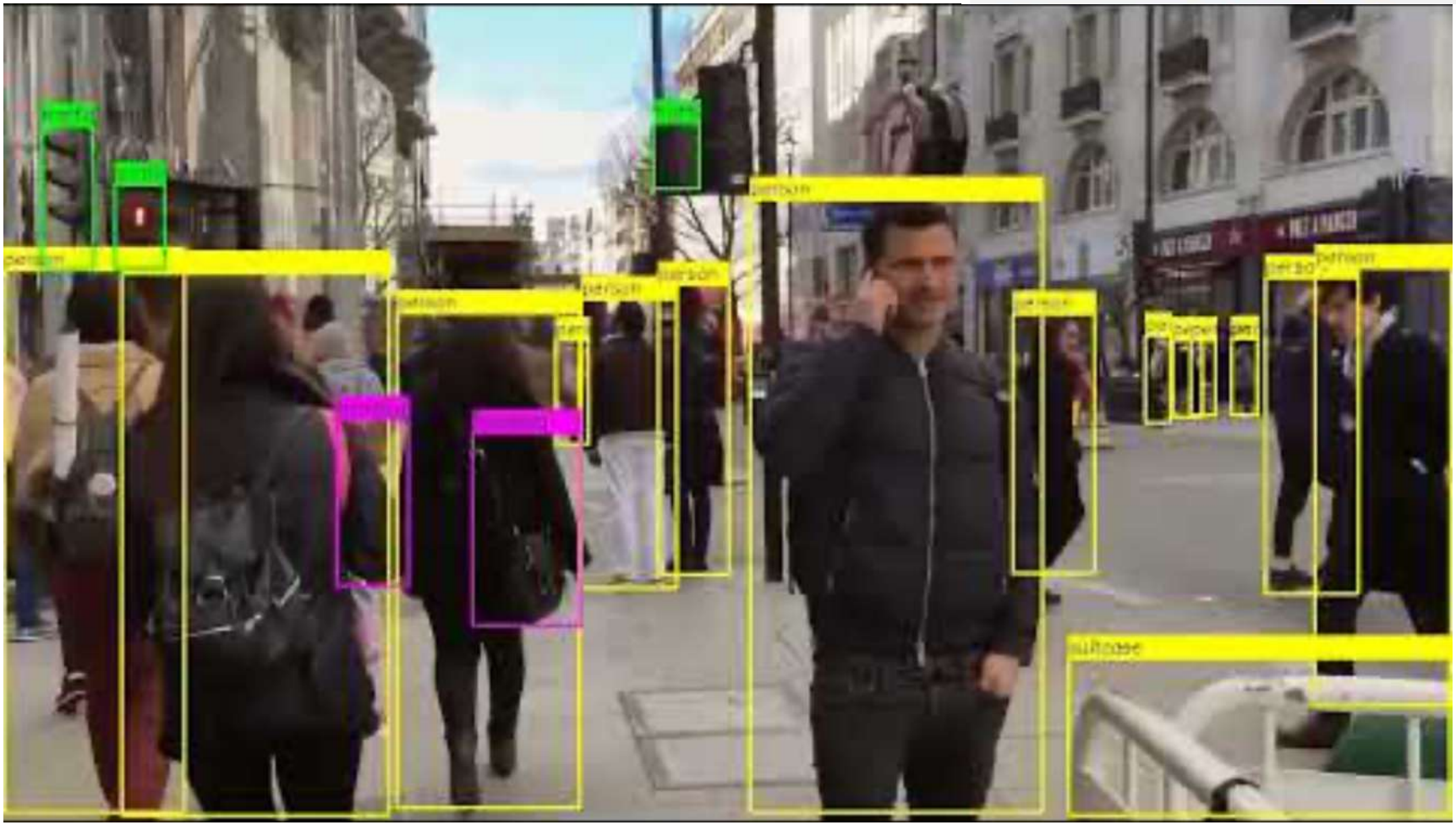
13,911 views Nov 13, 2017
No description has been added to this video.

YOLO v5

Example of YOLO v5 detection on video file

Luiz doleron
127 subscribers
Subscribe

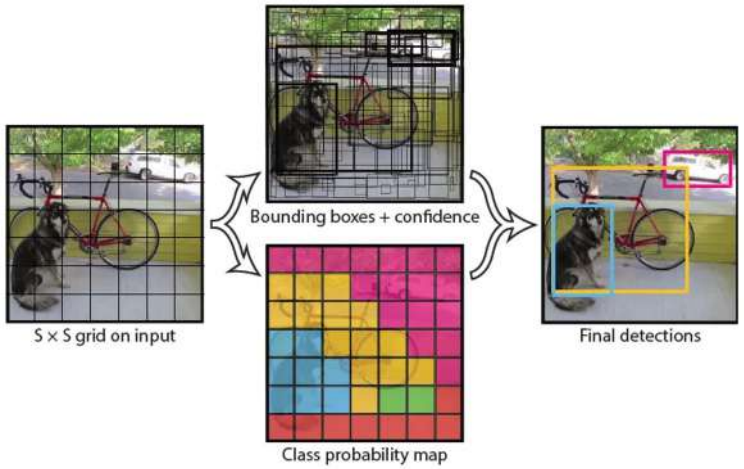
5,781 views Jan 23, 2022
Video example of YOLOv5 performing object detection.
- Check out the explanation at / detecting-objects-with-yolov5-opencv-pytho...
- Github repository is here: https://github.com/doleron/yolov5-ope...



What is YOLO?

You Only Look Once (YOLO) is a one-stage, end-to-end object detection framework that predicts bounding boxes and class probabilities in a single forward pass of a neural network.

Unlike earlier object detection approaches that adapted image classifiers into multi-step detection pipelines, YOLO treats object detection as a single regression problem.



https://medium.com/data-science/yolov1-you-only-look-once-object-detection-e1f3ffec8a89

News

- We are pleased to announce the LVIS 2021 Challenge and Workshop to be held at ICCV.
- Please note that there will not be a COCO 2021 Challenge, instead, we encourage people to participate in the LVIS 2021 Challenge.
- We have partnered with the team behind the open-source tool **FiftyOne** to make it easier to download, visualize, and evaluate COCO
- **FiftyOne** is an open-source tool facilitating visualization and access to COCO data resources and serves as an evaluation tool for model analysis on COCO.

What is COCO?

COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:

- **Object segmentation**
- **Recognition in context**
- **Superpixel stuff segmentation**
- **330K images (>200K labeled)**
- **1.5 million object instances**
- **80 object categories**
- **91 stuff categories**
- **5 captions per image**
- **250,000 people with keypoints**

Collaborators

Tsung-Yi Lin Google Brain
Genevieve Patterson MSR, Trash TV
Matteo R. Ronchi Caltech
Yin Cui Google
Michael Maire TTI-Chicago
Serge Belongie Cornell Tech
Lubomir Bourdev WaveOne, Inc.
Ross Girshick FAIR
James Hays Georgia Tech
Pietro Perona Caltech
Deva Ramanan CMU
Larry Zitnick FAIR
Piotr Dollár FAIR

Sponsors

CVDF
Microsoft
facebook
Mighty AI

https://cocodataset.org/#home

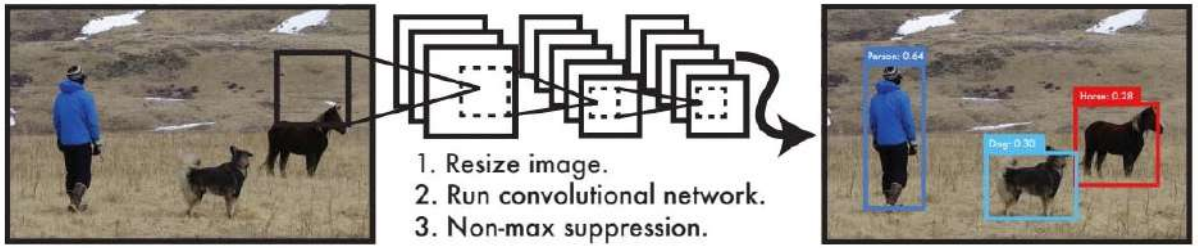
ultralytics/cfg/datasets/coco8.yaml

79 classes

```yaml
# Classes
names:
  0: person
  1: bicycle
  2: car
  3: motorcycle
  4: airplane
  5: bus
  6: train
  7: truck
  8: boat
  9: traffic light
  10: fire hydrant
  11: stop sign
  12: parking meter
  13: bench
  14: bird
  15: cat
  16: dog
  17: horse
  18: sheep
  19: cow
  20: elephant
  21: bear
  22: zebra
  23: giraffe
```

https://docs.ultralytics.com/datasets/detect/#ultralytics-yolo-format

YOLO's output



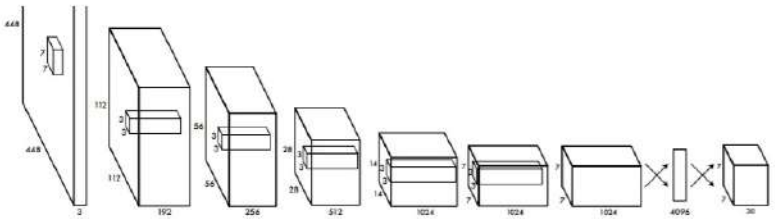
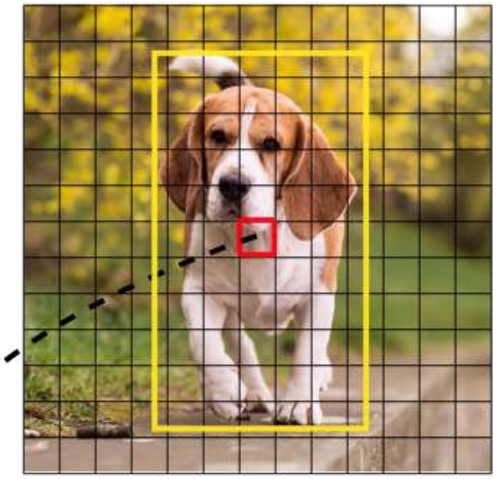
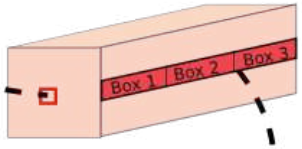


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

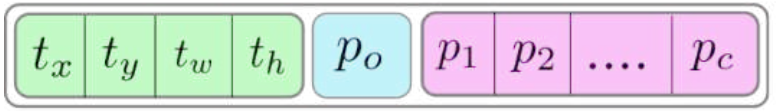
Image Grid. The Red Grid is responsible for detecting the dog



Prediction Feature Map



Attributes of a bounding box

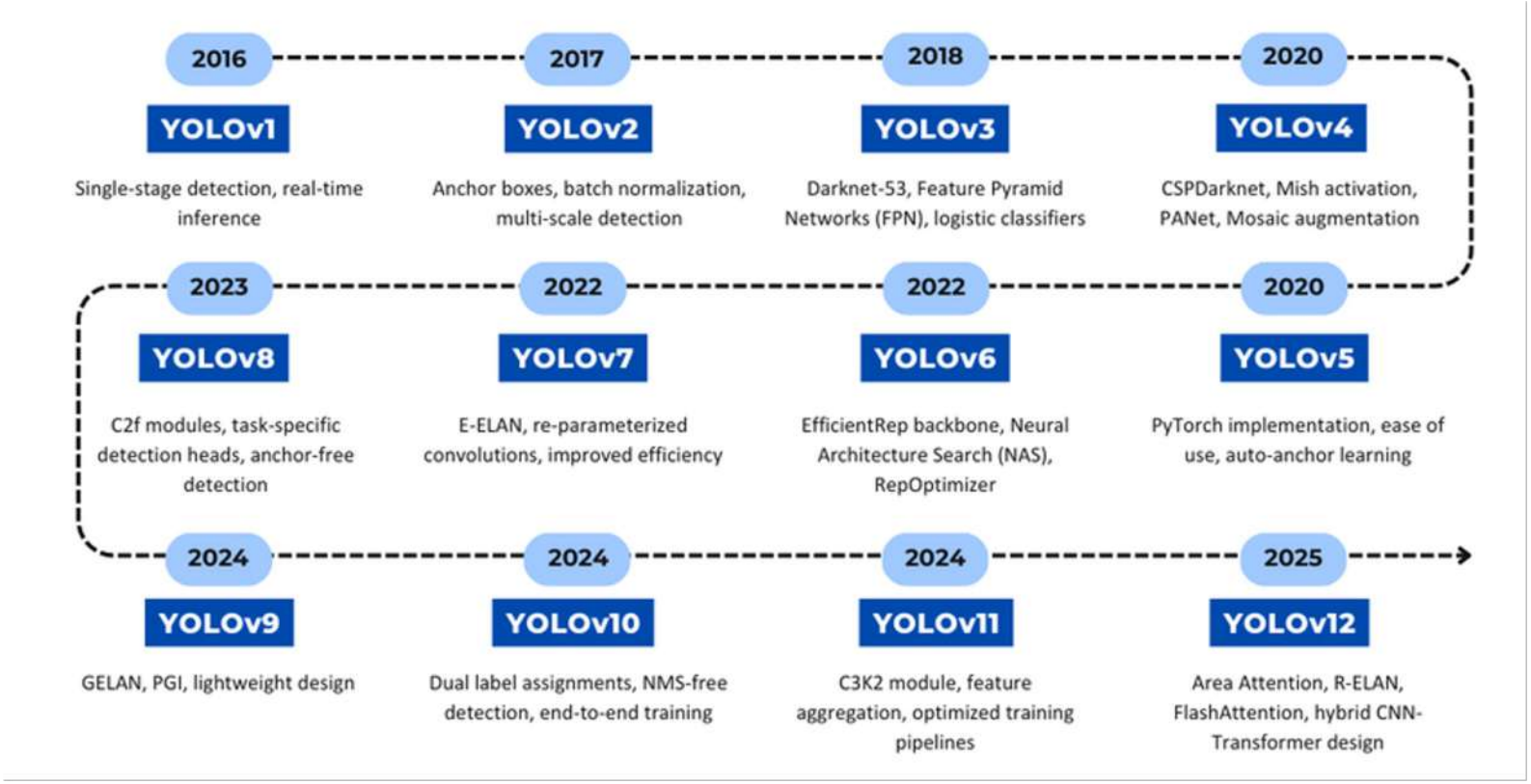


Box Co-ordinates | Objectness Score | Class Scores

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

https://arxiv.org/abs/2504.11995

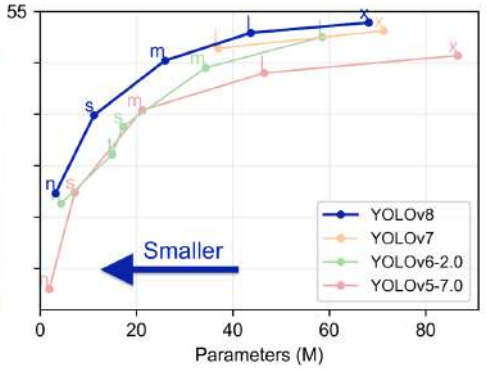
The Evolution of YOLO

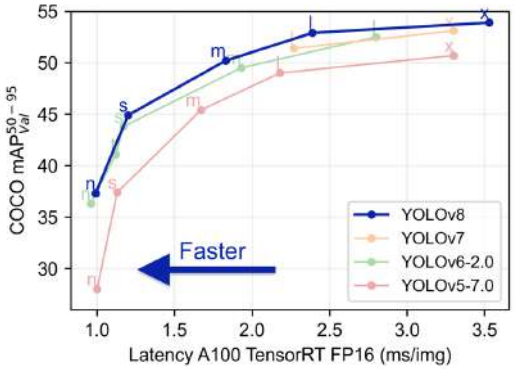


Ultralytics YOLO → production-ready, practical framework (YOLOv5 → YOLOv8 → YOLO11).
Academic YOLO → new research architectures (YOLOv9, v10, v11, v12).

Ultralytic's YOLO

YOLOv5 (2020) → YOLOv8 (2023) → Ultralytics YOLO 11 (2024)





English | 简体中文

Ultralytics CI passing | codecov 87% | DOI 10.5281/zenodo.7347926 | docker pulls 22k

Run on Gradient | Open in Colab | Open in Kaggle

Ultralytics YOLOv8 is a cutting-edge, state-of-the-art (SOTA) model that builds upon the success of previous YOLO versions and introduces new features and improvements to further boost performance and flexibility. YOLOv8 is designed to be fast, accurate, and easy to use, making it an excellent choice for a wide range of object detection and tracking, instance segmentation, image classification and pose estimation tasks.

We hope that the resources here will help you get the most out of YOLOv8. Please browse the YOLOv8 Docs for details, raise an issue on GitHub for support, and join our Discord community for questions and discussions!

To request an Enterprise License please complete the form at Ultralytics Licensing.

Models

YOLOv8 Detect, Segment and Pose models pretrained on the COCO dataset are available here, as well as YOLOv8 Classify models pretrained on the ImageNet dataset. Track mode is available for all Detect, Segment and Pose models.

All Models download automatically from the latest Ultralytics release on first use.

YOLO Models (v9 – v12) Timeline

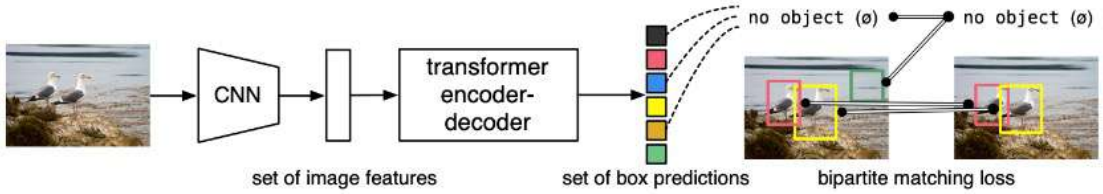
Model	Release Timeframe	Team / Organization	Key Highlight
YOLOv12	Feb 2025	**Google Research + Tsinghua University** (collaboration)	Attention-centric architecture with high accuracy and real-time speed
YOLOv11	Late 2024	**Tencent Youtu Lab**	Enhanced architecture (C3k2, SPPF, C2PSA), versatile across detection, segmentation, pose
YOLOv10	Mid 2024	**Tsinghua University + Huawei Noah's Ark Lab**	End-to-end design without NMS, major speed and efficiency gains
YOLOv9	Early 2024	**Wong Kin Yiu (original YOLO contributor) + Community researchers**	PGI and GELAN for better gradients and efficiency
Ultralytics YOLO11	2024–2025	**Ultralytics**	Practical SOTA release, supports detection, segmentation, pose, classification, track

DETR (Detection Transformer) [Meta, 2020]

DE:TR: End-to-End Object Detection with Transformers

Support Ukraine

PyTorch training code and pretrained models for **DETR** (**DE**tection **TR**ansformer). We replace the full complex hand-crafted object detection pipeline with a Transformer, and match Faster R-CNN with a ResNet-50, obtaining **42 AP** on COCO using half the computation power (FLOPs) and the same number of parameters. Inference in 50 lines of PyTorch.



- 1. Deformable DETR (2020)
- Team: Microsoft Research Asia

- 2. Conditional DETR (2021)
- Team: Chinese Academy of Sciences

- 3. Anchor DETR (2021)
- Team: SenseTime and Shanghai Jiao Tong University

- 4. Efficient DETR (2021)
- Team: UC Berkeley and Google Research

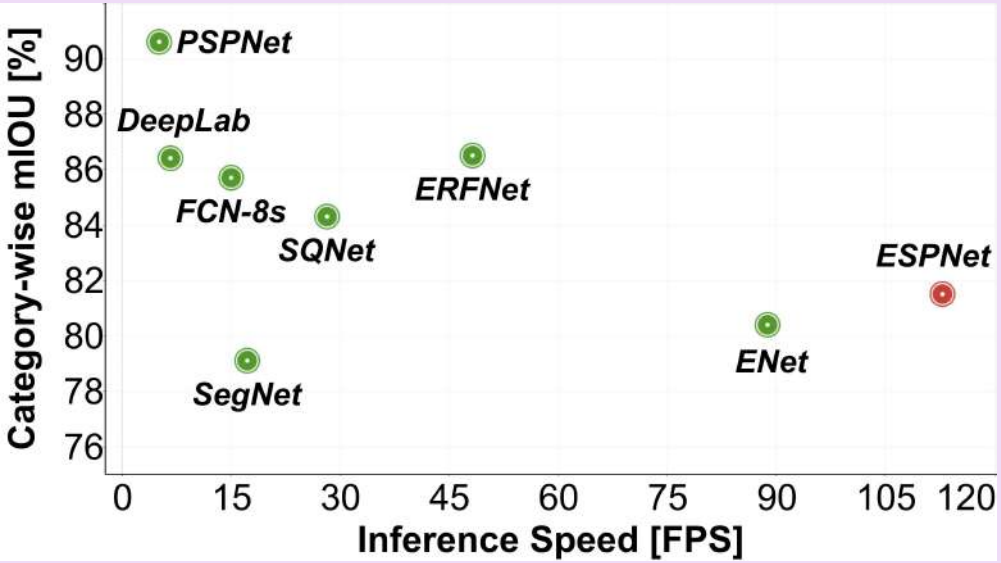
- 5. DINO (DETR with Improved Non-Autoregressive Decoding) (2022)
- Team: Meta AI (formerly Facebook AI Research)

- 6. RT-DETR (Real-Time DETR) (2024)
- Team: Alibaba DAMO Academy

Top Object Detection Models (2025)

Model	Team / Source	Notable Metric (COCO)	Strengths
RF-DETR	Roboflow	73.6 AP_{50} / 54.7 AP_{50:95} (Medium)	Real-time DETR; best mAP at low latency
YOLOv12	YOLO authors (YOLO Series)	—	Latest YOLO, attention-enhanced
YOLO11 (Ultralytics)	Ultralytics	Up to ~50 AP_{50} / ~48.6 AP_{50:95}	Multi-task, production-ready

SOTA of Semantic Segmentation



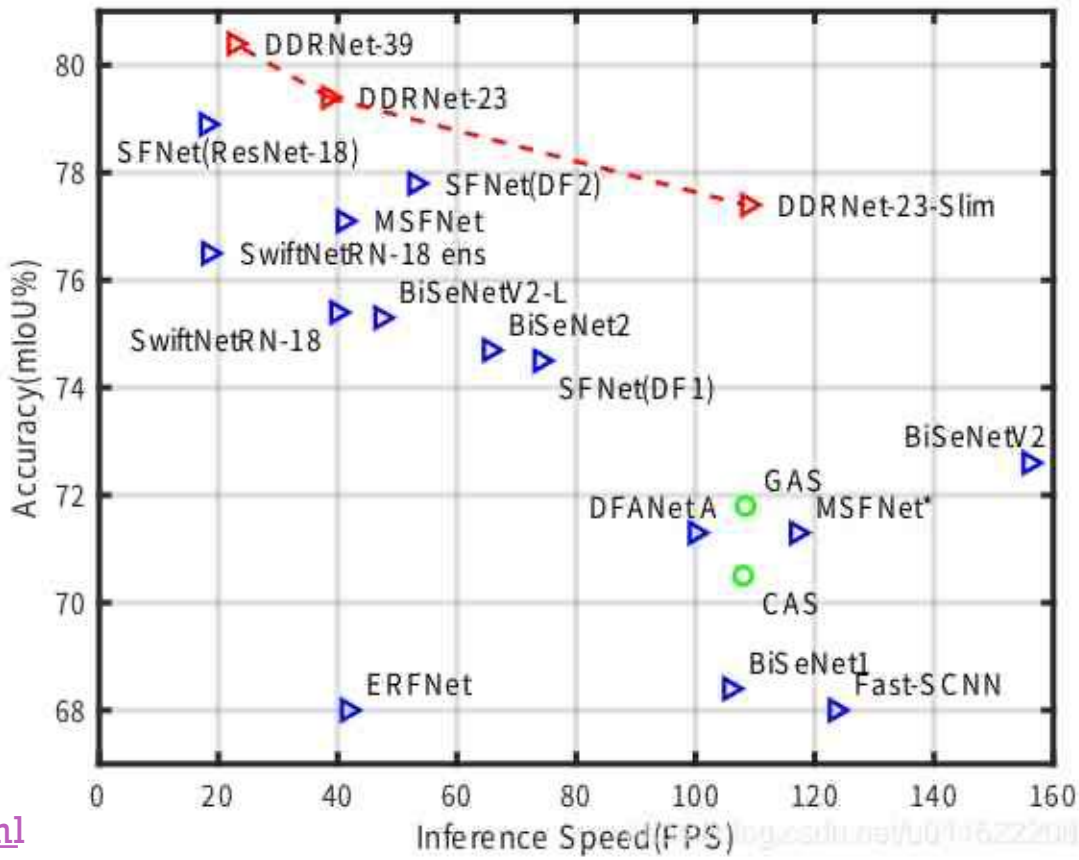
- **Real-time semantic segmentation**
- BiSeNet V1,V2
- DDRNet

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

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

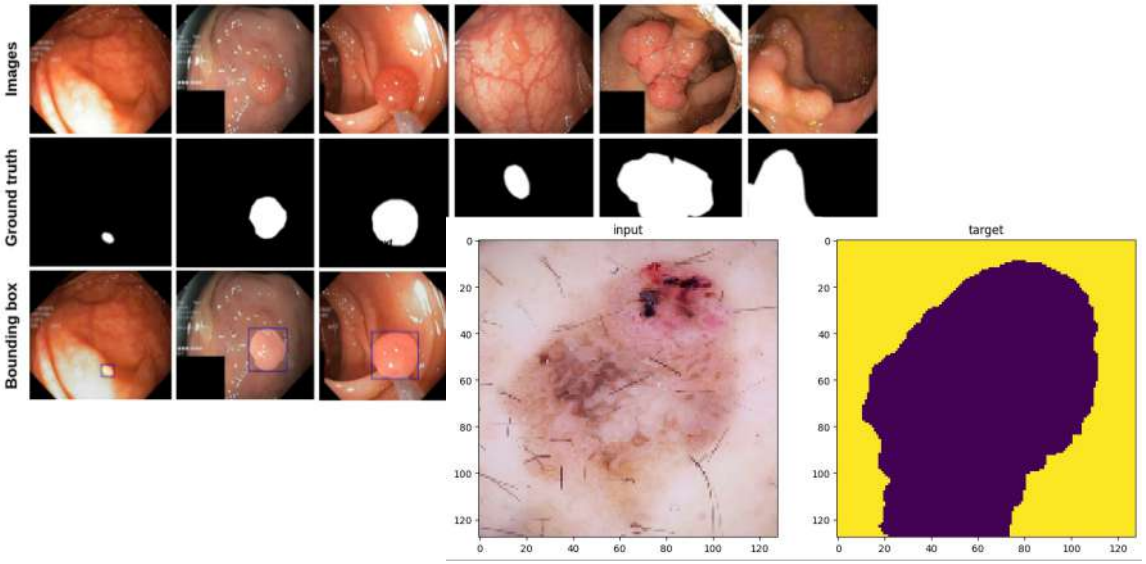
2021-11-09 04:24:01 【Nongfu Mountain Spring 2】

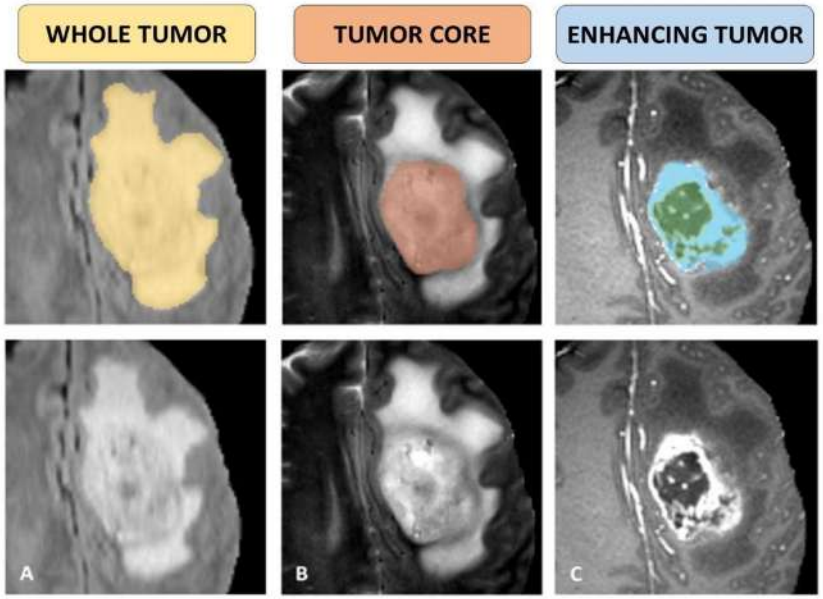
当前位置：网站首页>[Pers



Semantic Segmentation

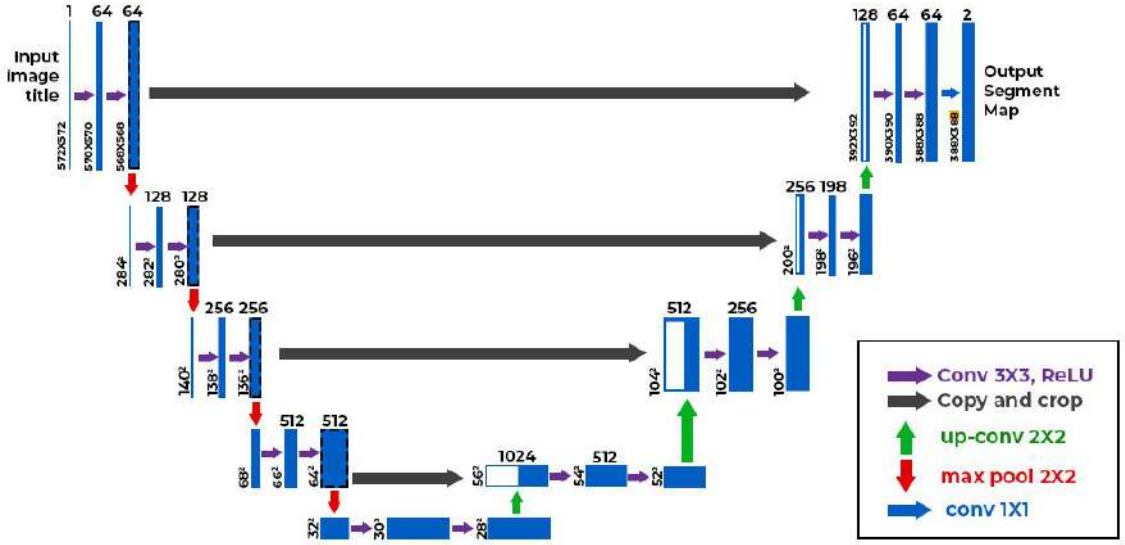
- Semantic segmentation = pixel-wise classification
- Each pixel gets a class label (road, car, polyp, tumor, sky, background).

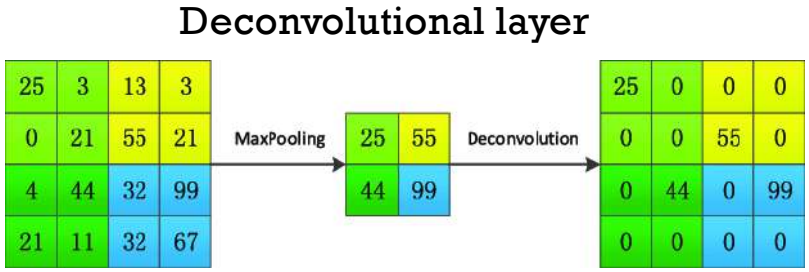




UNet: Encoder-Decoder Network: Encoder, Decoder, Skip Connections

- **U-Net**





$\mathcal{L} = \lambda_1 \cdot \mathrm{CE} + \lambda_2 \cdot \mathrm{Dice}$; CE (pixel loss) & Dice (region loss; semantic)

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)

Recent interesting segmentation networks

- 1. ContextFormer (2025)
- Team: Abid, Mehta, Wu, Dharejo, Timofte
- Architecture: Hybrid CNN + Vision Transformer
- Strengths: Combines high accuracy with real-time efficiency
- Applications: Urban scene understanding, robotics, autonomous driving

- 2. Segment Anything Model (SAM) [Meta, 2023] → SAM V2 [Meta, 2024]
- Explanation: SAM is a highly versatile model designed to segment any object within an image with minimal user input. It uses a prompt-based approach, where the user provides a prompt (e.g., a point or bounding box), and the model generates a mask for the object. SAM is particularly noted for its generalizability across different segmentation tasks.

- 3. Mask2Former [Meta, 2022]
- Explanation: Mask2Former is a unified model for instance, semantic, and panoptic segmentation tasks. It improves upon the original MaskFormer by incorporating transformers into the architecture, enabling it to handle different segmentation tasks within a single framework efficiently.

- 4. Swin Transformer V2 [Microsoft, 2022]
- Explanation: Swin Transformer V2 builds on the hierarchical vision transformer design introduced in the original Swin Transformer. It enhances performance in both segmentation and classification tasks and has achieved SOTA results on benchmarks such as COCO and ADE20K.

- 5. SegFormer [NVIDIA, 2021]
- Explanation: SegFormer is a transformer-based model that is designed to be both efficient and accurate for semantic segmentation tasks. It integrates transformers with lightweight MLP decoders, enabling high performance while maintaining computational efficiency.

- 6. DeepLabV3+ [Google, 2018]
- Explanation: DeepLabV3+ is an extension of the DeepLab series, which combines spatial pyramid pooling and atrous convolutions. The "+" version introduces a decoder module to improve object boundary delineation, making it a strong performer in semantic segmentation.

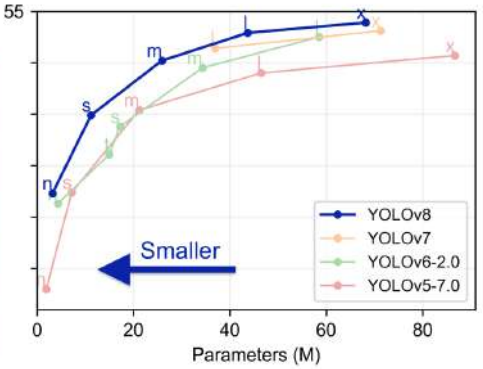
Introducing SAM 2: The next generation of Meta Segment Anything Model for videos and images

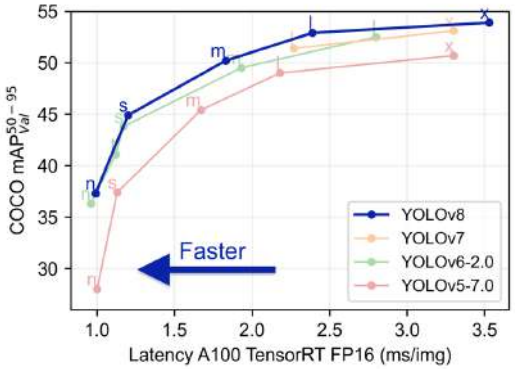
July 29, 2024 · 15 minute read

https://ai.meta.com/blog/segment-anything-2/

Ultralytic's YOLO (extra segmentation model)

YOLOv5 (2020) → YOLOv8 (2023) → Ultralytics YOLO 11 (2024)





English | 简体中文

Ultralytics CI passing | codecov 87% | DOI 10.5281/zenodo.7347926 | docker pulls 22k

Run on Gradient | Open in Colab | Open in Kaggle

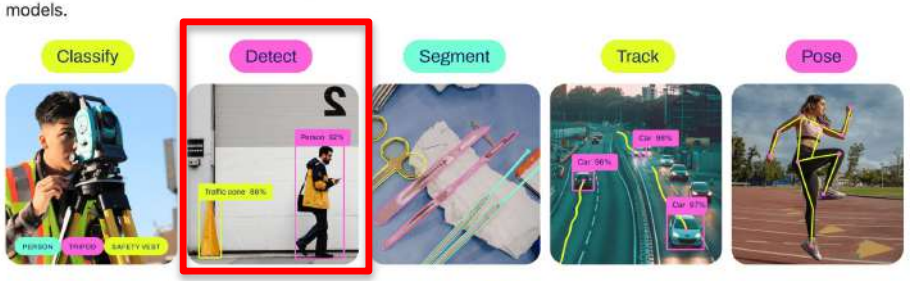
Ultralytics YOLOv8 is a cutting-edge, state-of-the-art (SOTA) model that builds upon the success of previous YOLO versions and introduces new features and improvements to further boost performance and flexibility. YOLOv8 is designed to be fast, accurate, and easy to use, making it an excellent choice for a wide range of object detection and tracking, instance segmentation, image classification and pose estimation tasks.

We hope that the resources here will help you get the most out of YOLOv8. Please browse the YOLOv8 Docs for details, raise an issue on GitHub for support, and join our Discord community for questions and discussions!

To request an Enterprise License please complete the form at Ultralytics Licensing.

Models

YOLOv8 Detect, Segment and Pose models pretrained on the COCO dataset are available here, as well as YOLOv8 Classify models pretrained on the ImageNet dataset. Track mode is available for all Detect, Segment and Pose models.



All Models download automatically from the latest Ultralytics release on first use.

YOLOv8-Seg: Object Detection → Segment

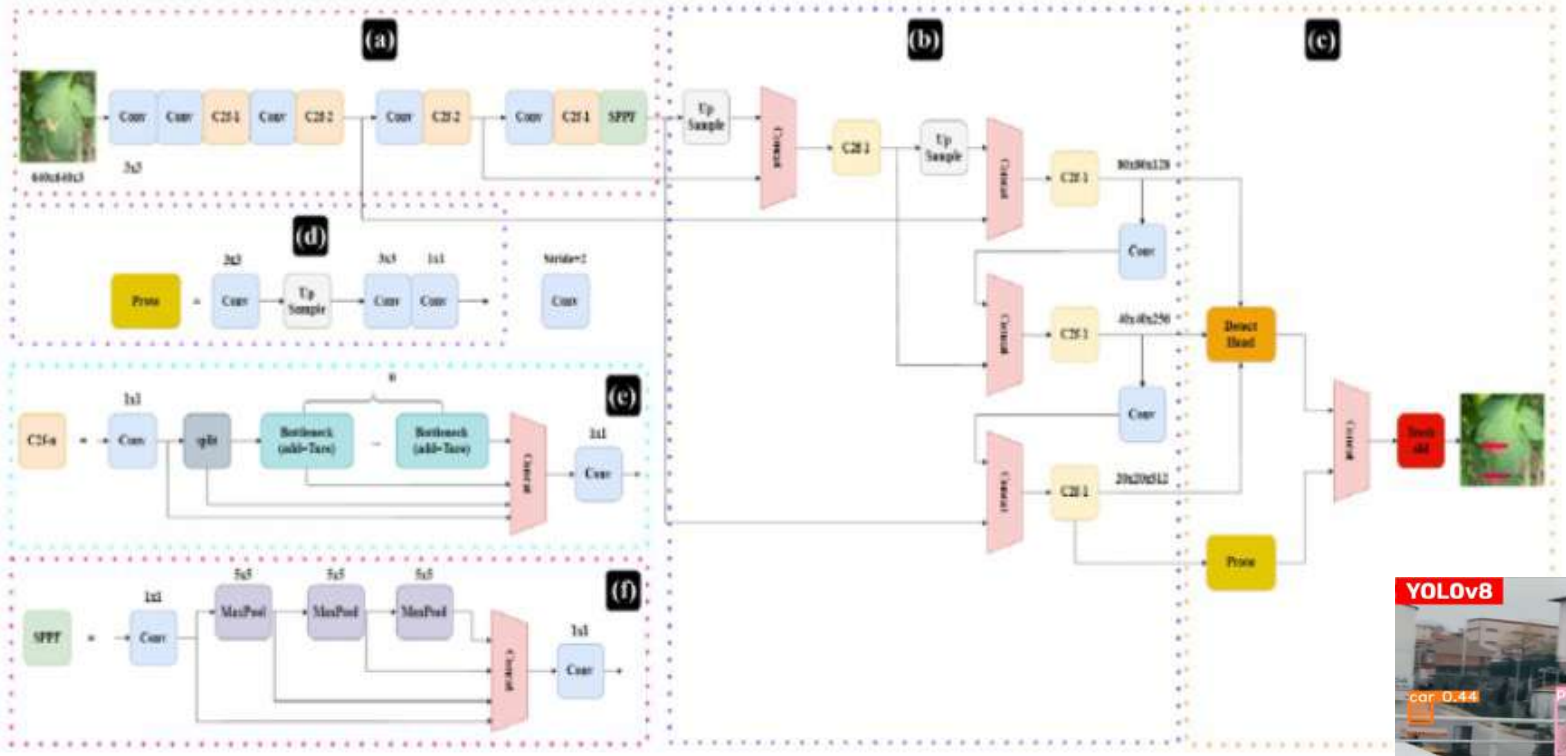
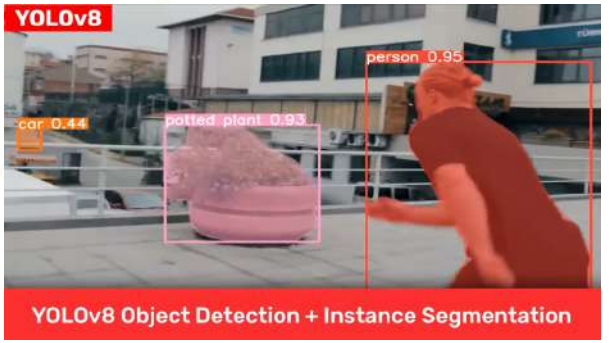


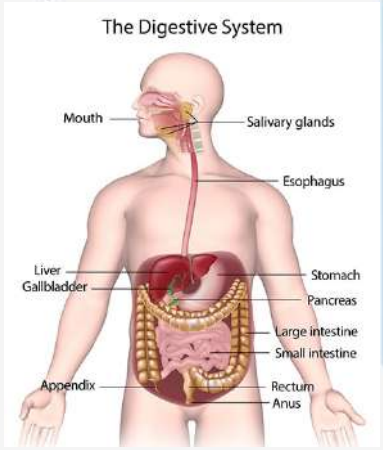
Fig. 3 YOLOv8-Seg architecture; **a** backbone, **b** neck, **c** head, **d** Proto module, **e** C2f module and **f** SPPF module



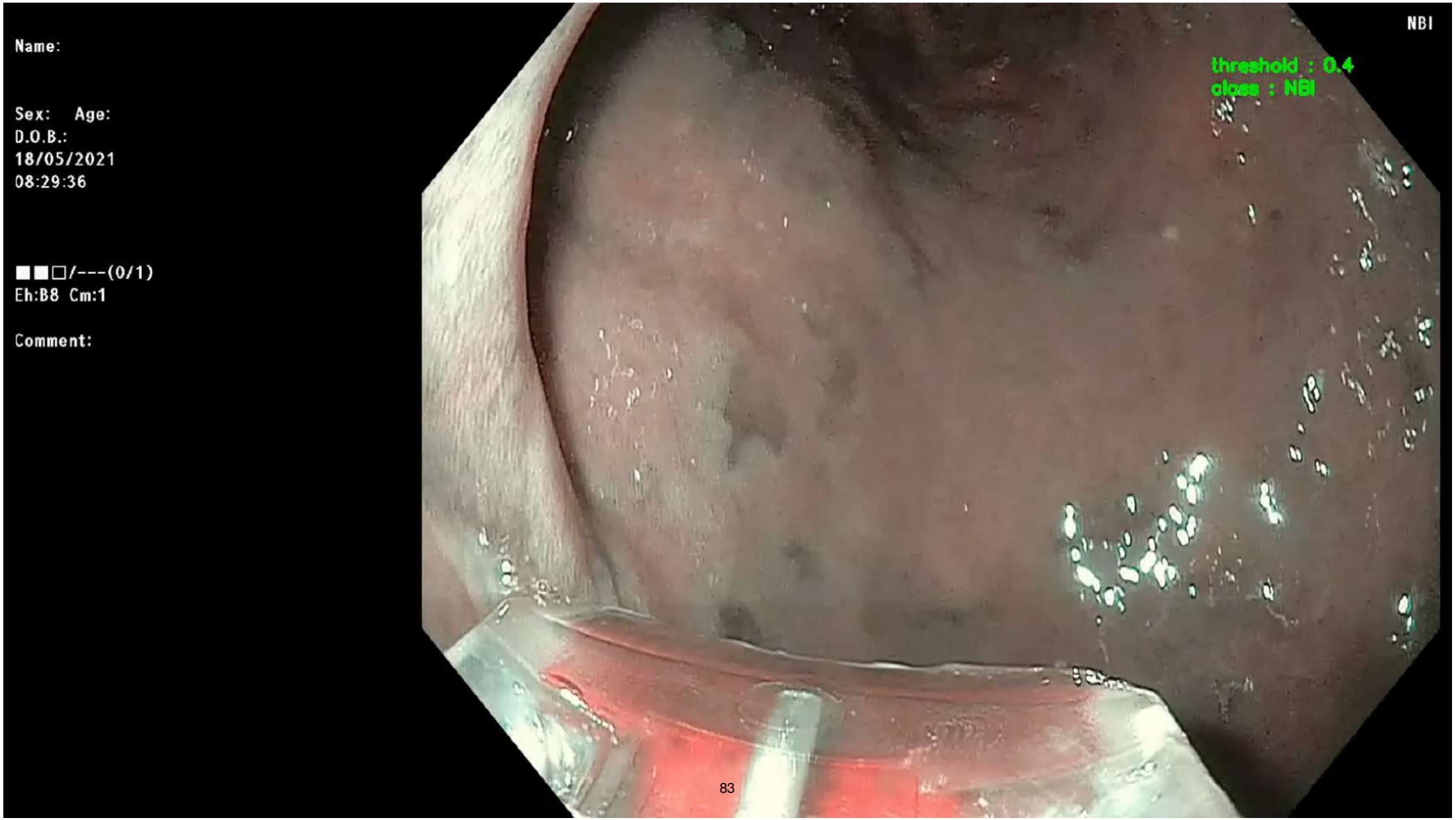
Link

Real-Time Colonic Polyp Detection

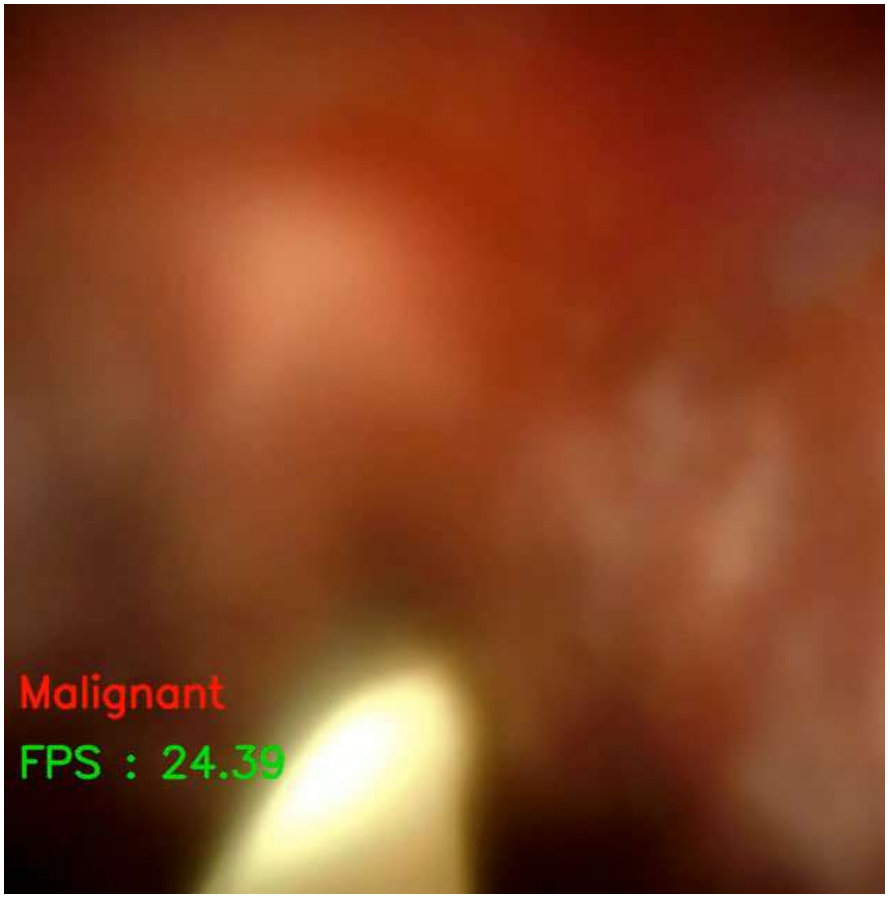
An AI-assisted solution that aims to improve colonic polyp detection in real-time. It is compatible with all standard colonoscope systems.

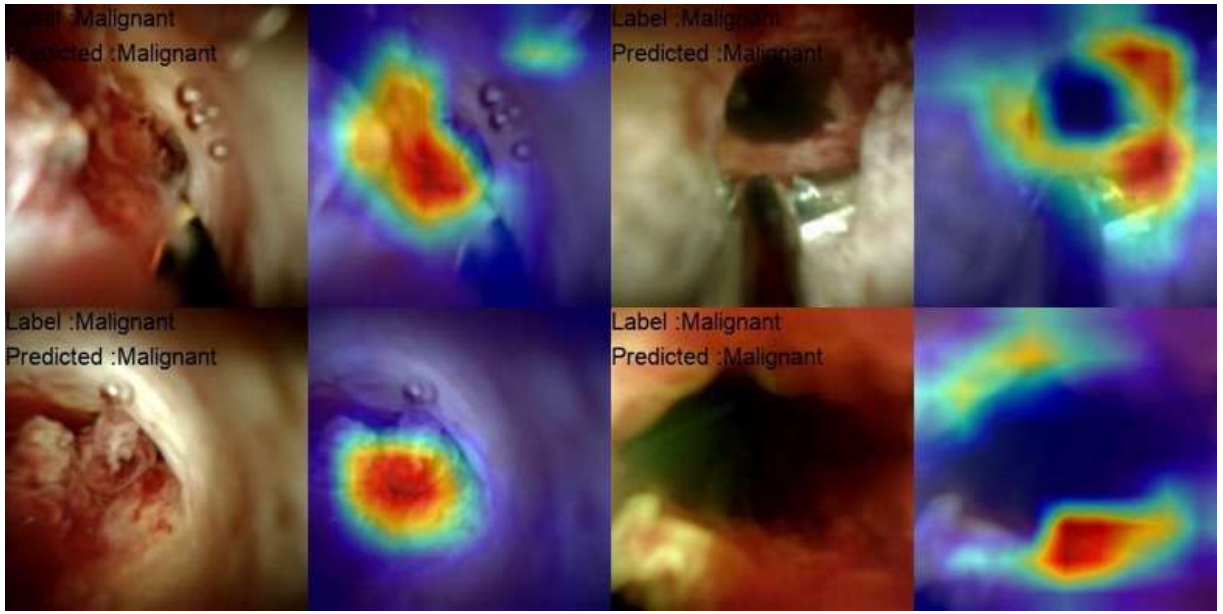



09/11/2021
13:53:24
COLON
*1/200
Lv+2 AUTO
HT NR
SE
3.8
12.0
12.0
S1: F+T
S2: LM
S3: CAD
S4:
EC-760R-V/L
3C729K035
BL-7000
CHULALONGKORN HOS


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





Appendix