資料探勘
(Data Mining)

卷積神經網絡
(Convolutional Neural Networks)

1092DM09
MBA, IM, NTPU (M5026) (Spring 2021)
Tue 2, 3, 4 (9:10-12:00) (B8F40)

Min-Yuh Day
戴敏育
Associate Professor
副教授
Institute of Information Management, National Taipei University
國立臺北大學 資訊管理研究所
https://web.ntpu.edu.tw/~myday

2021-05-11

課程大綱 (Syllabus)

週次 (Week)　日期 (Date)　內容 (Subject/Topics)

1 2021/02/23 資料探勘介紹 (Introduction to data mining)

2 2021/03/02 ABC：人工智慧，大數據，雲端運算
(ABC: AI, Big Data, Cloud Computing)

3 2021/03/09 Python資料探勘的基礎
(Foundations of Data Mining in Python)

4 2021/03/16 資料科學與資料探勘：發現，分析，可視化和呈現數據
(Data Science and Data Mining:
Discovering, Analyzing, Visualizing and Presenting Data)

5 2021/03/23 非監督學習：關聯分析，購物籃分析
(Unsupervised Learning: Association Analysis,
Market Basket Analysis)

6 2021/03/30 資料探勘個案研究 I
(Case Study on Data Mining I)

課程大綱 (Syllabus)

週次 (Week) 日期 (Date) 內容 (Subject/Topics)

7 2021/04/06 放假一天 (Day off)

8 2021/04/13 非監督學習：集群分析，行銷市場區隔
(Unsupervised Learning: Cluster Analysis, Market Segmentation)

9 2021/04/20 期中報告 (Midterm Project Report)

10 2021/04/27 監督學習：分類和預測
(Supervised Learning: Classification and Prediction)

11 2021/05/04 機器學習和深度學習
(Machine Learning and Deep Learning)

12 2021/05/11 卷積神經網絡
(Convolutional Neural Networks)

課程大綱 (Syllabus)

週次 (Week) 日期 (Date) 內容 (Subject/Topics)

13 2021/05/18 資料探勘個案研究 II (Case Study on Data Mining II)

14 2021/05/25 遞歸神經網絡 (Recurrent Neural Networks)

15 2021/06/01 強化學習 (Reinforcement Learning)

16 2021/06/08 社交網絡分析 (Social Network Analysis)

17 2021/06/15 期末報告 I (Final Project Report I)

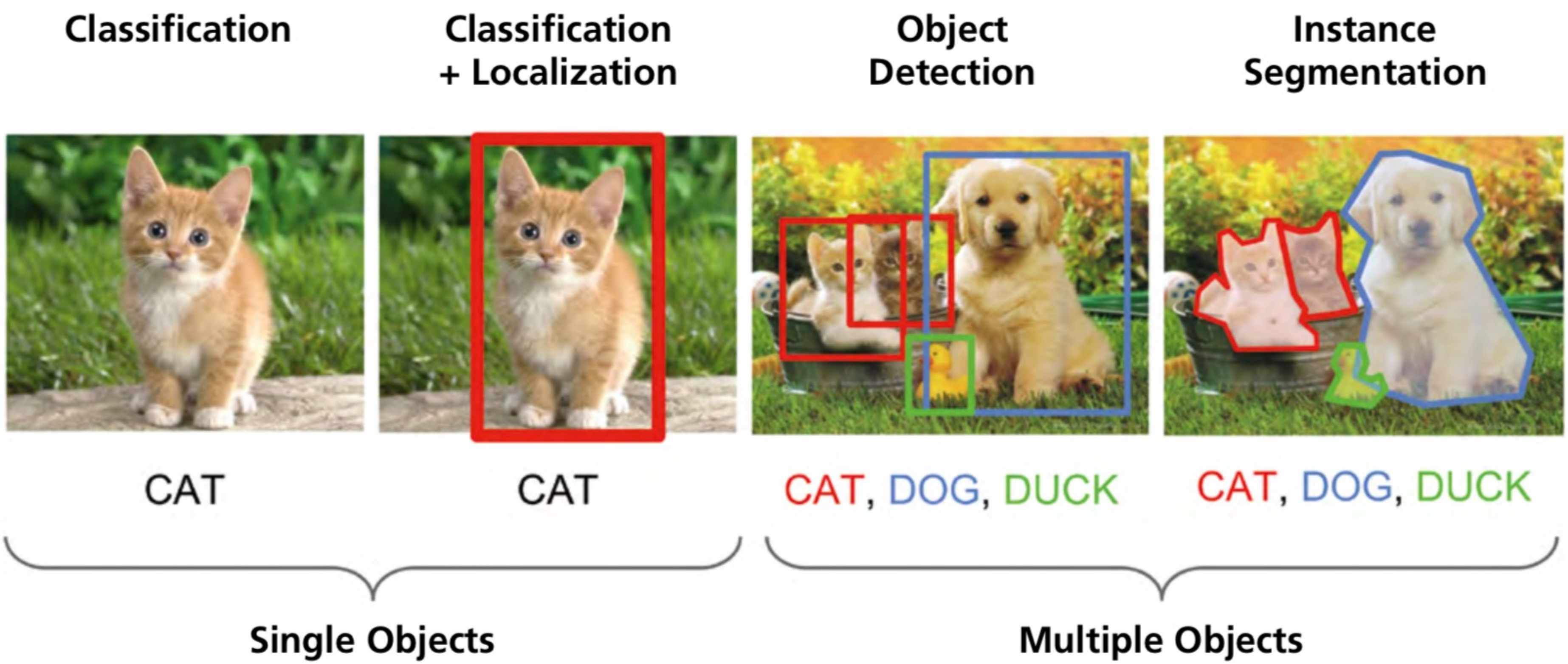
18 2021/06/22 期末報告 II (Final Project Report II)

Convolutional Neural Networks (CNN)

Outline

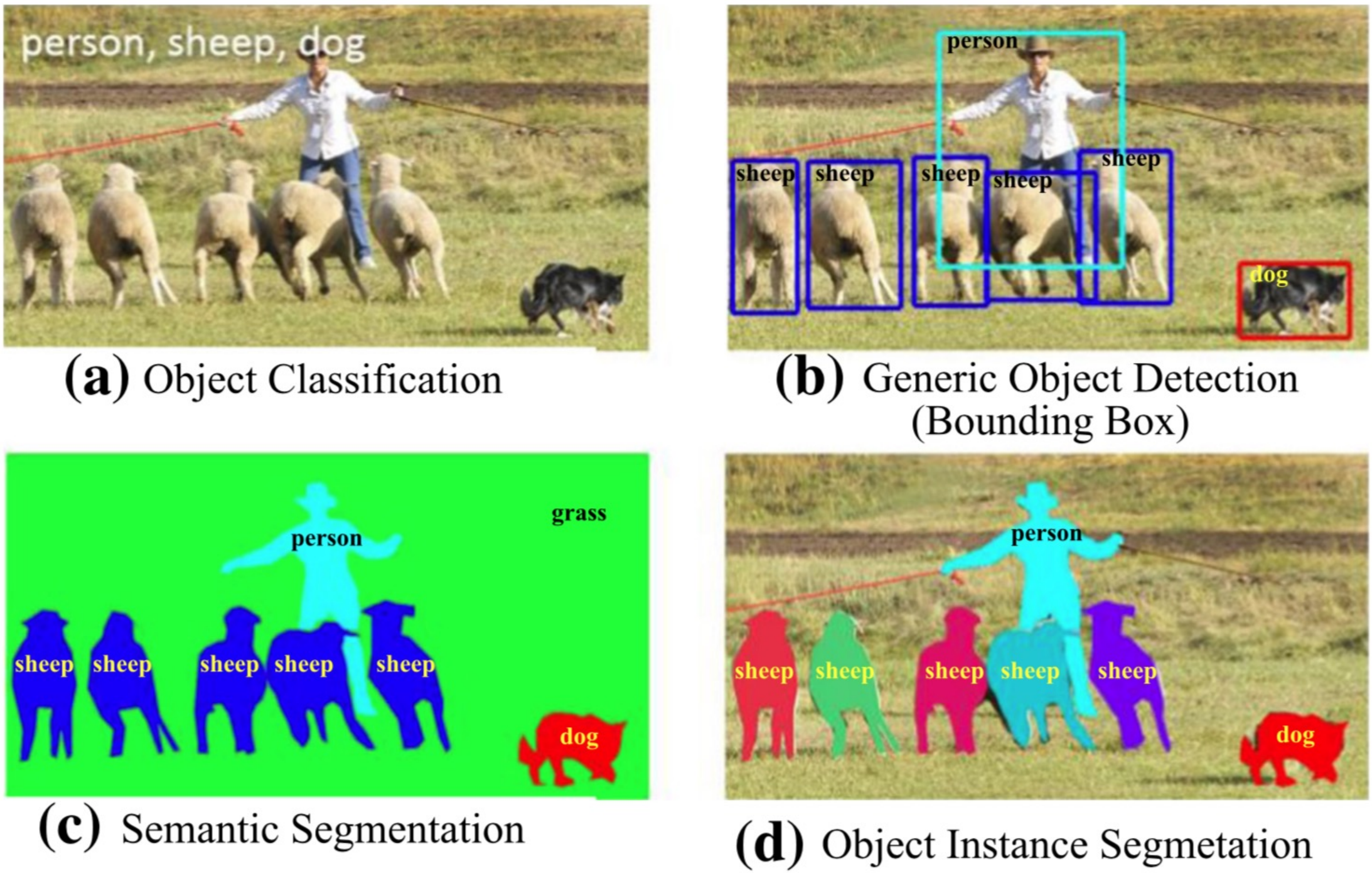
- Convolutional Neural Networks (CNN)
  – Convolution
  – Pooling
  – Fully Connection (FC) (Flattening)
- Computer Vision
  – Image Classification
  – Object Detection

Computer Vision: Image Classification, Object Detection, Object Instance Segmentation



Source: DHL (2018), Artificial Intelligence in Logistics, http://www.globalhha.com/doclib/data/upload/doc_con/5e50c53c5bf67.pdf/

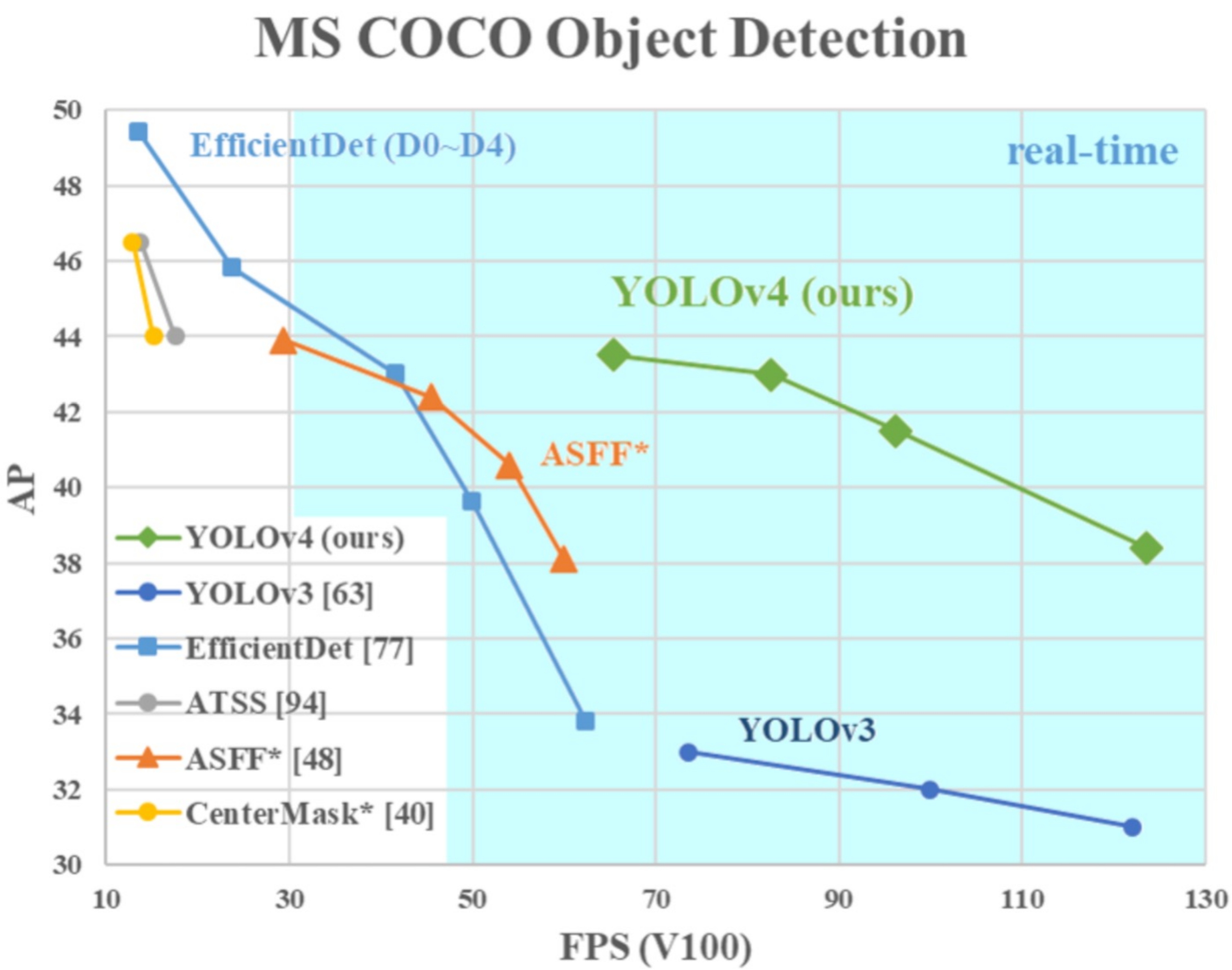
Computer Vision: Object Detection



Source: Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. "Deep learning for generic object detection: A survey." International journal of computer vision 128, no. 2 (2020): 261-318.

YOLOv4:

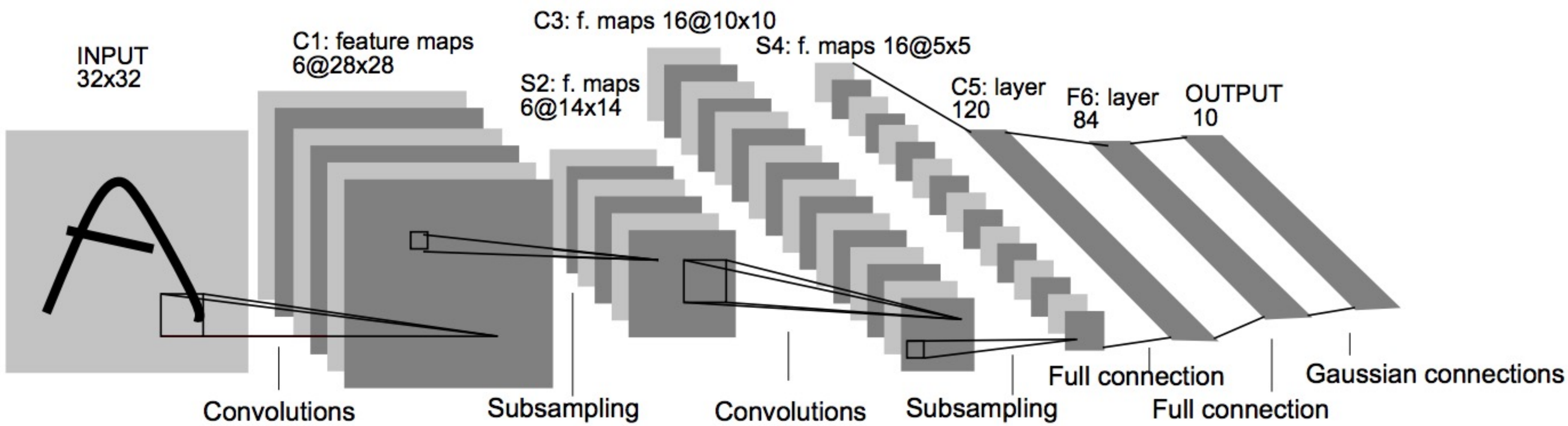
Optimal Speed and Accuracy of Object Detection



Source: Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection." arXiv preprint arXiv:2004.10934 (2020).

Convolutional Neural Networks (CNN)

Convolutional Neural Networks (CNN)



Architecture of LeNet-5 (7 Layers) (LeCun et al., 1998)

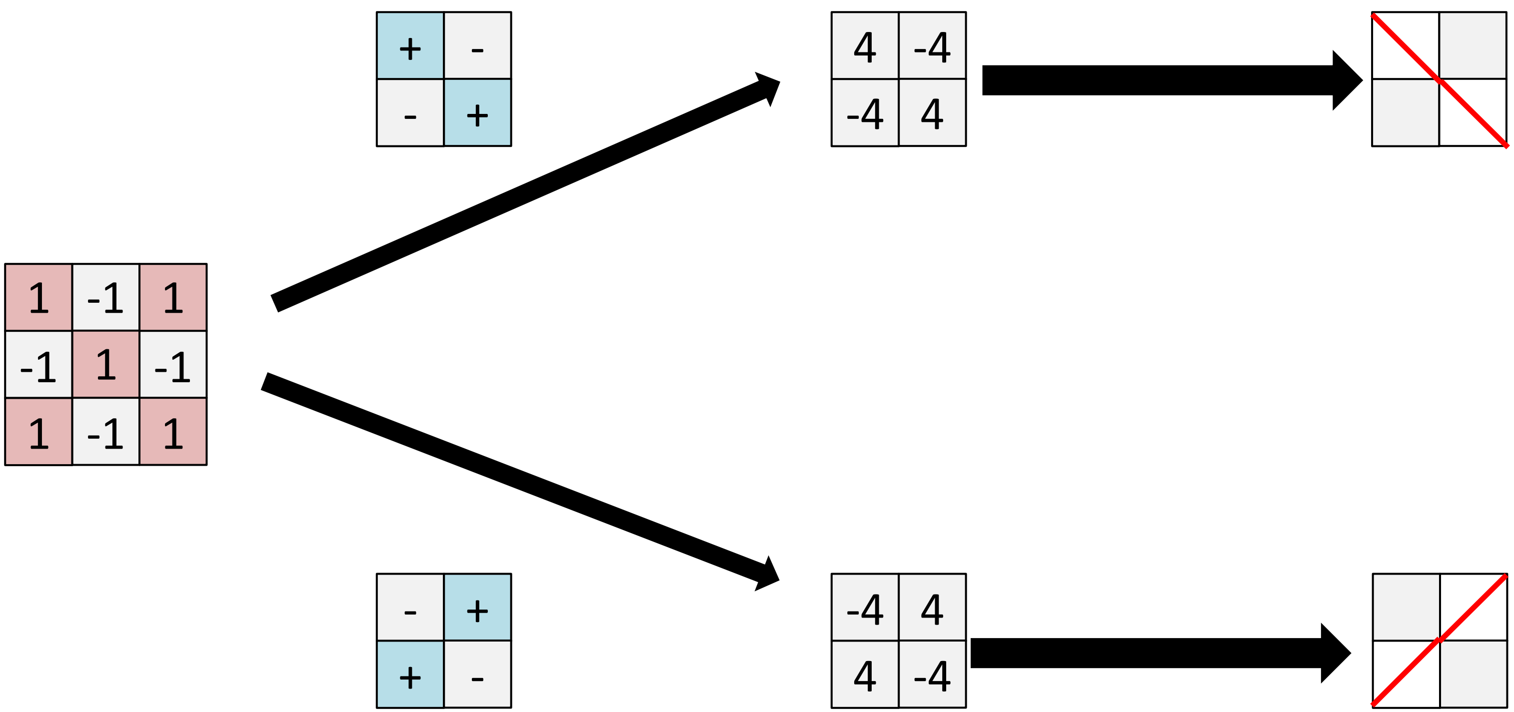
Source: http://yann.lecun.com/exdb/publis/pdf/lecun-01a.pdf

Source: LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition." *Proceedings of the IEEE* 86, no. 11 (1998): 2278-2324.

Convolutional Neural Networks (CNN)

- Convolution
- Pooling
- Fully Connection (FC) (Flattening)

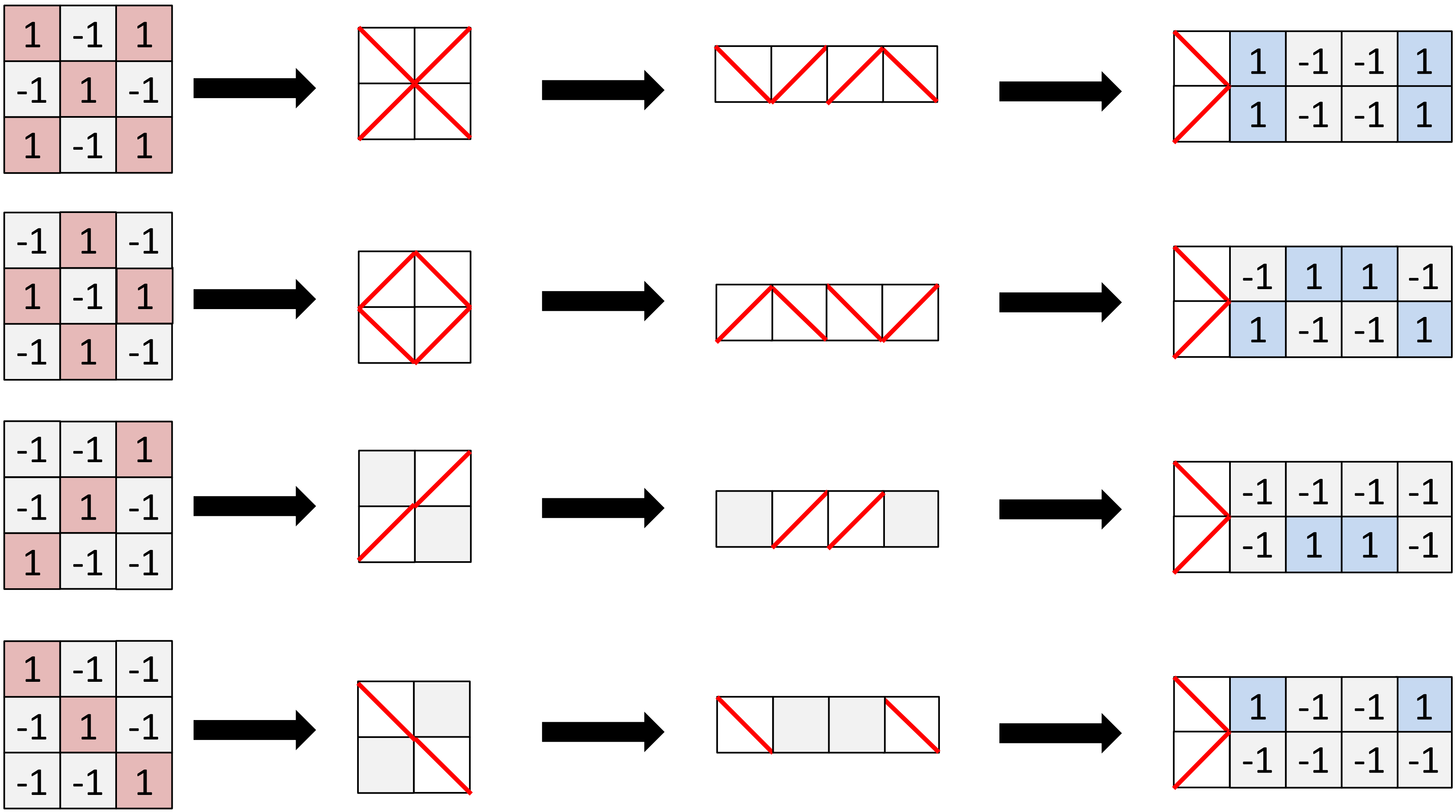
A friendly introduction to Convolutional Neural Networks and Image Recognition



Convolution Layer
Pooling Layer

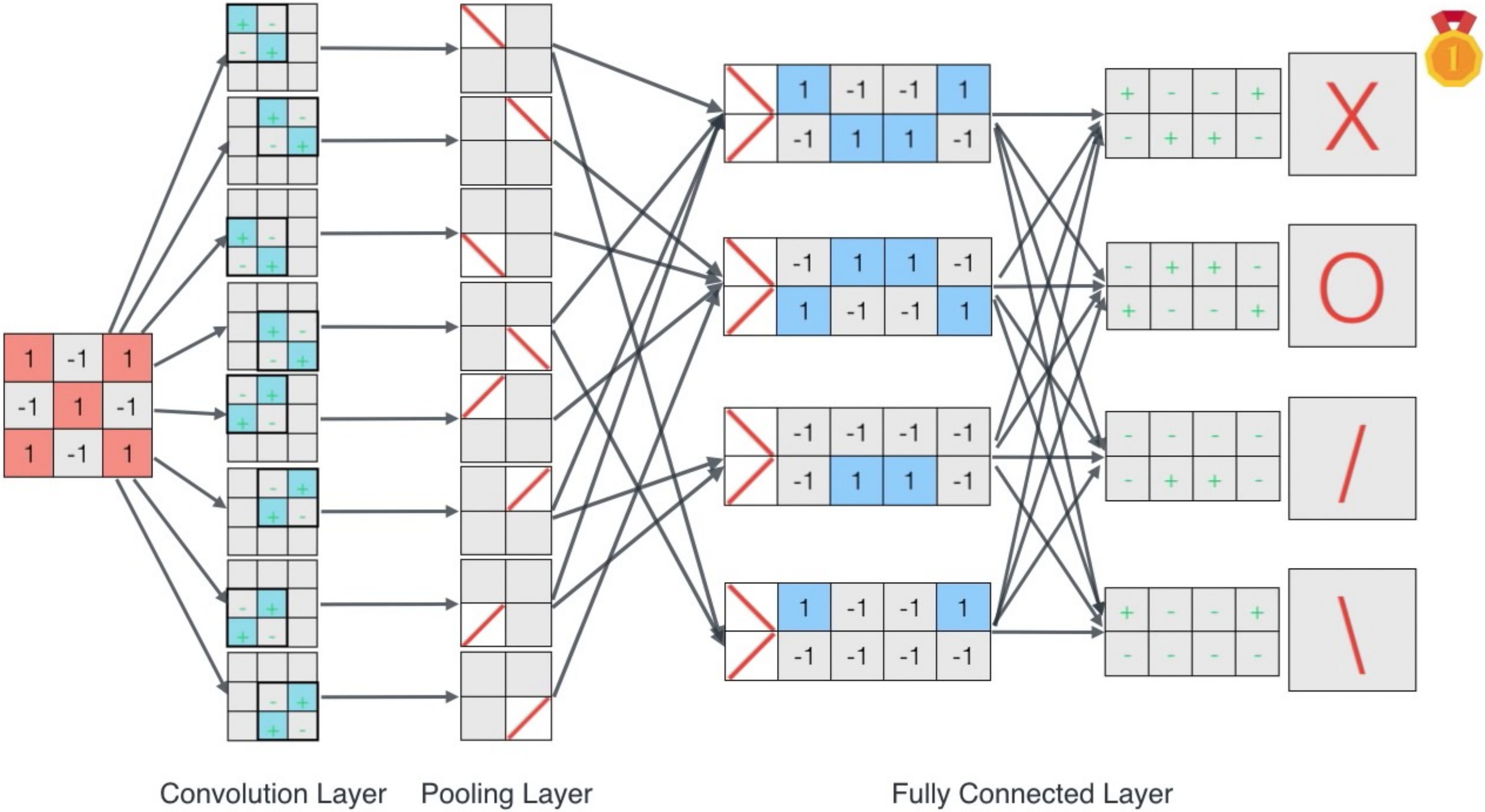
Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-Ol7ZB0MmU

A friendly introduction to Convolutional Neural Networks and Image Recognition



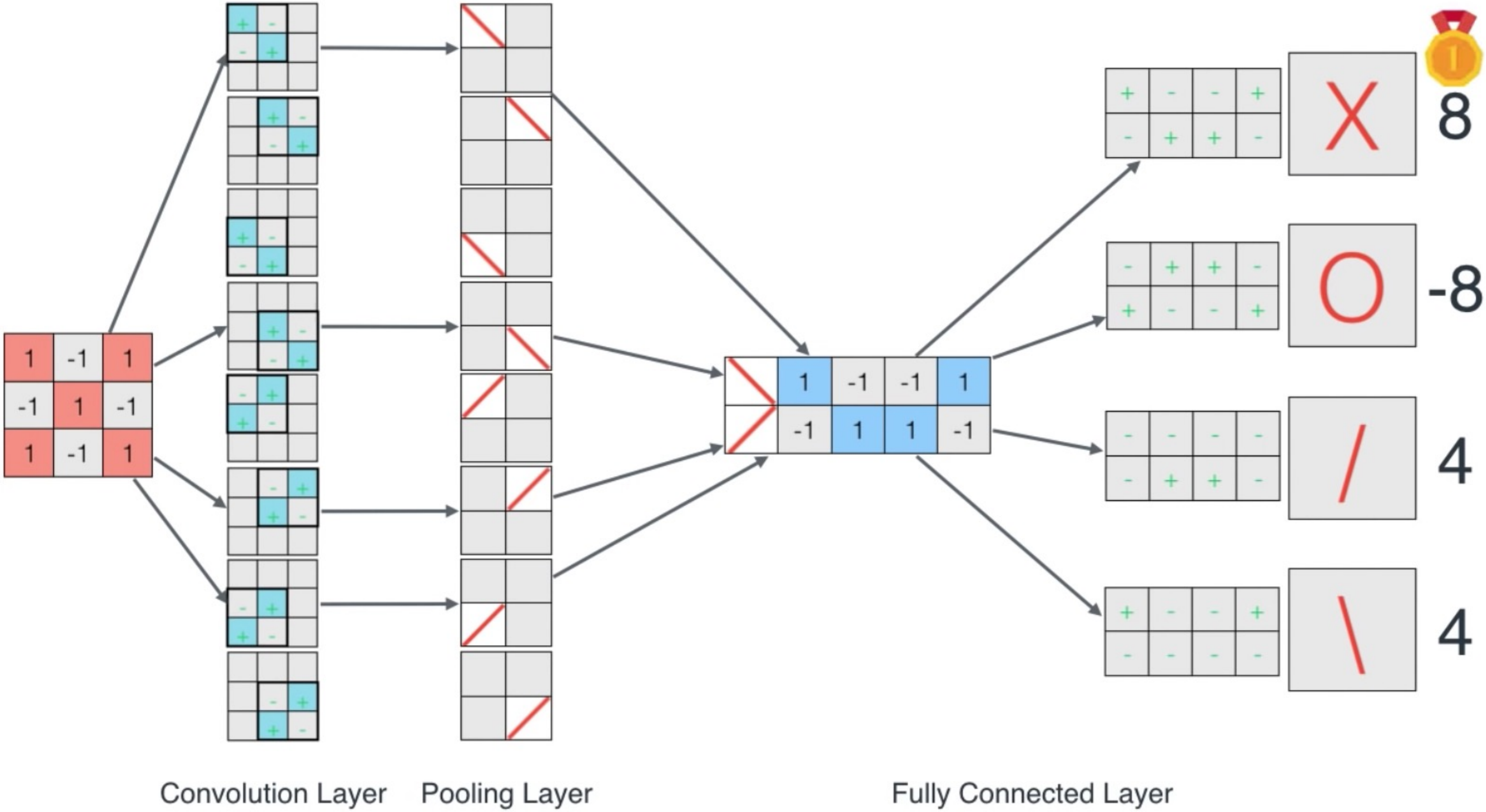
Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-Ol7ZB0MmU

A friendly introduction to Convolutional Neural Networks and Image Recognition



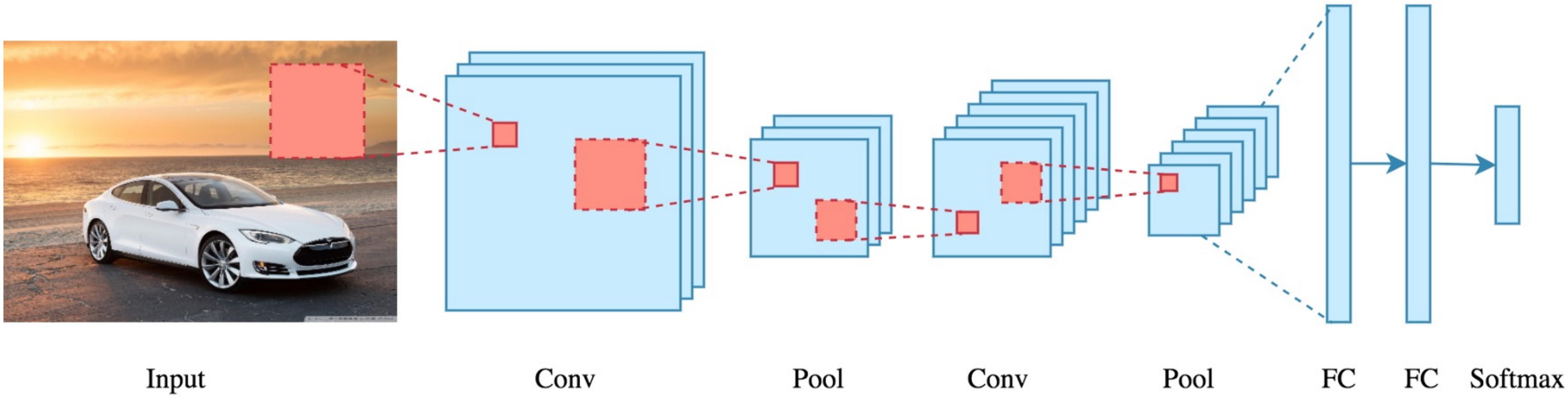
Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-Ol7ZB0MmU

A friendly introduction to Convolutional Neural Networks and Image Recognition



Source: Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-Ol7ZB0MmU

CNN Architecture



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer

Convolution is a **mathematical operation** to **merge two sets** of information

3x3 convolution

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Input

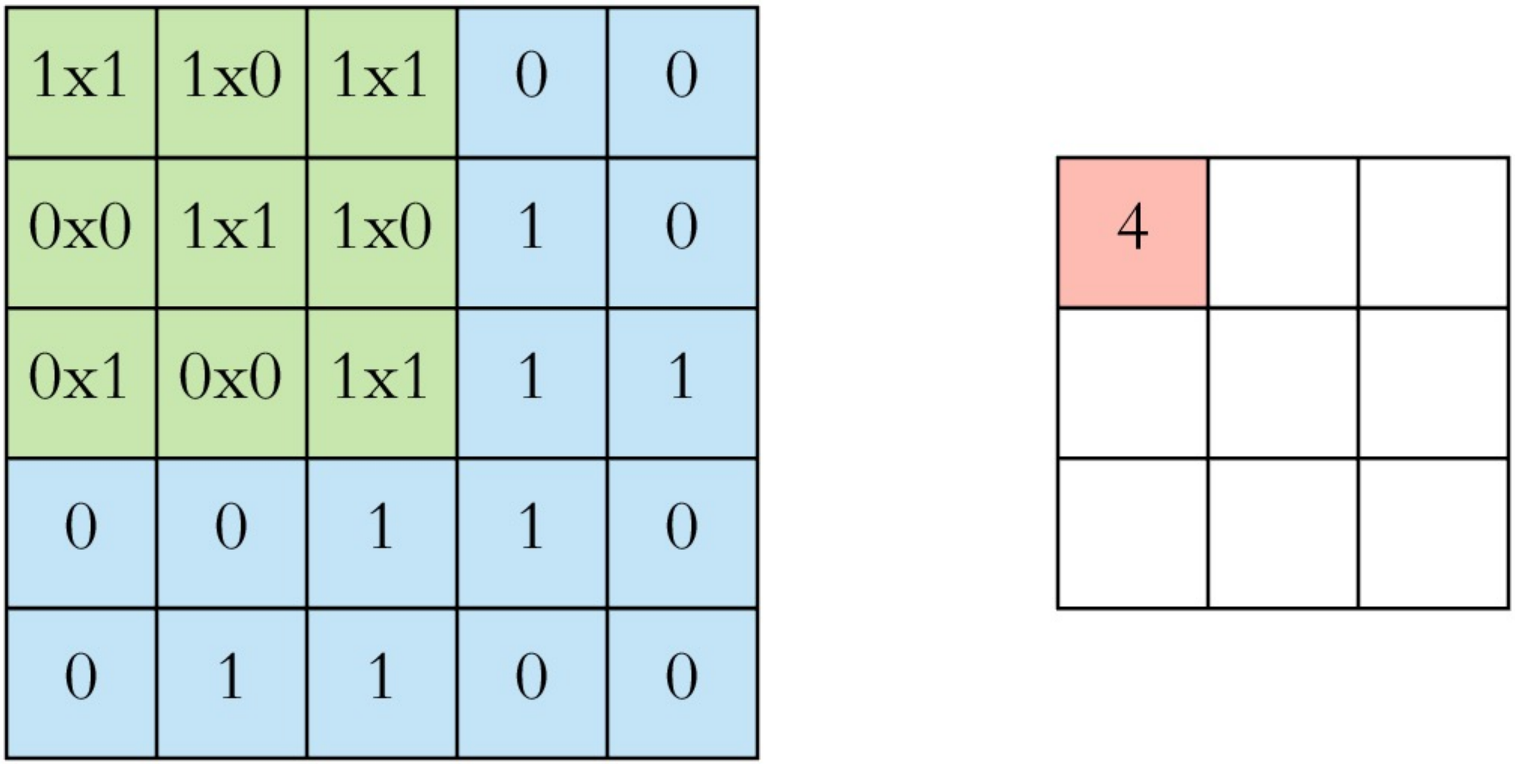
1	0	1
0	1	0
1	0	1

Filter / Kernel

Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer
Input x Filter --> Feature Map

receptive field: 3x3



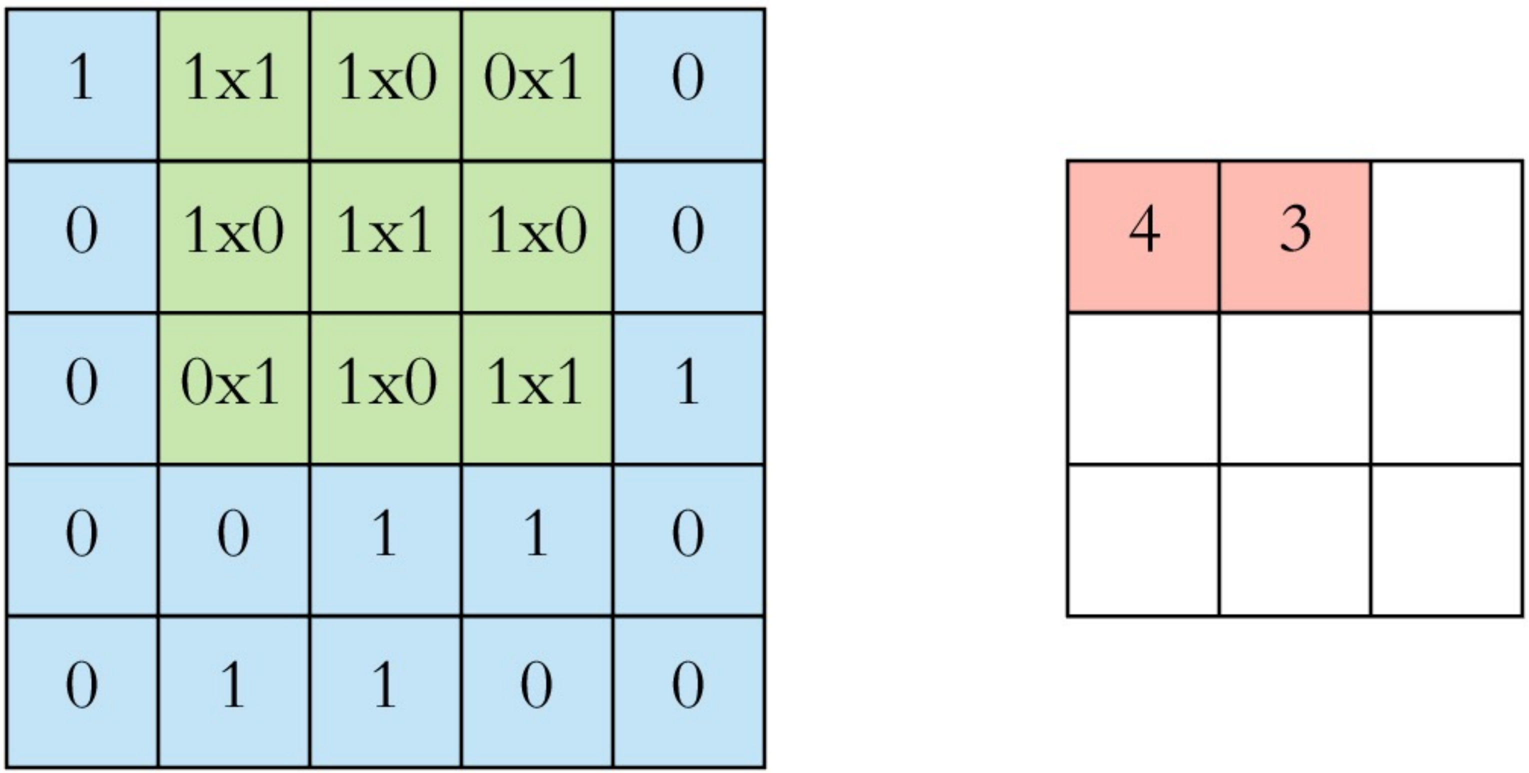
Input x Filter

Feature Map

Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer
Input x Filter --> Feature Map

receptive field: 3x3

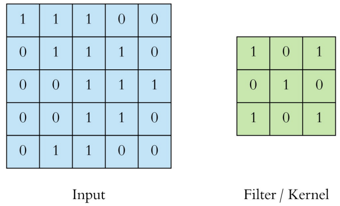


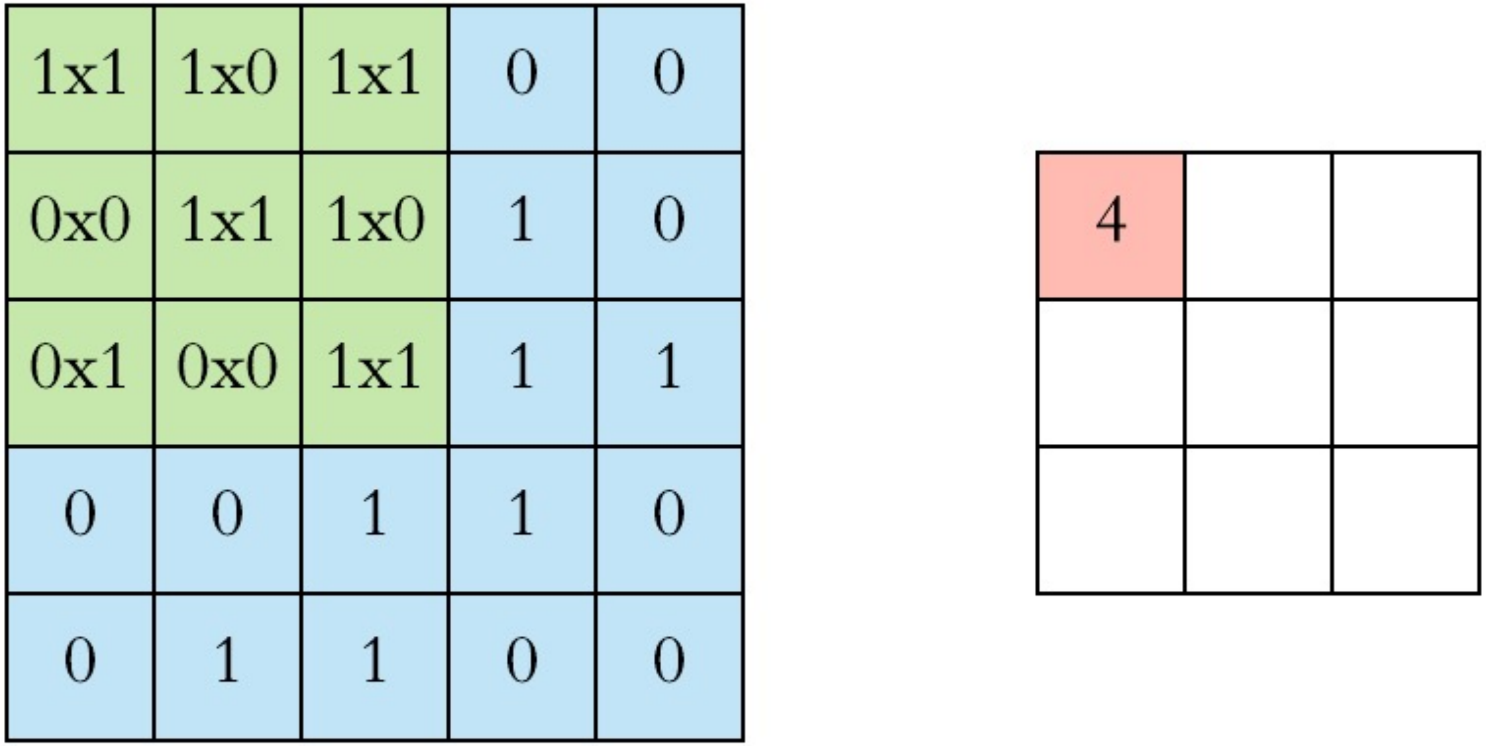
Input x Filter

Feature Map

Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer



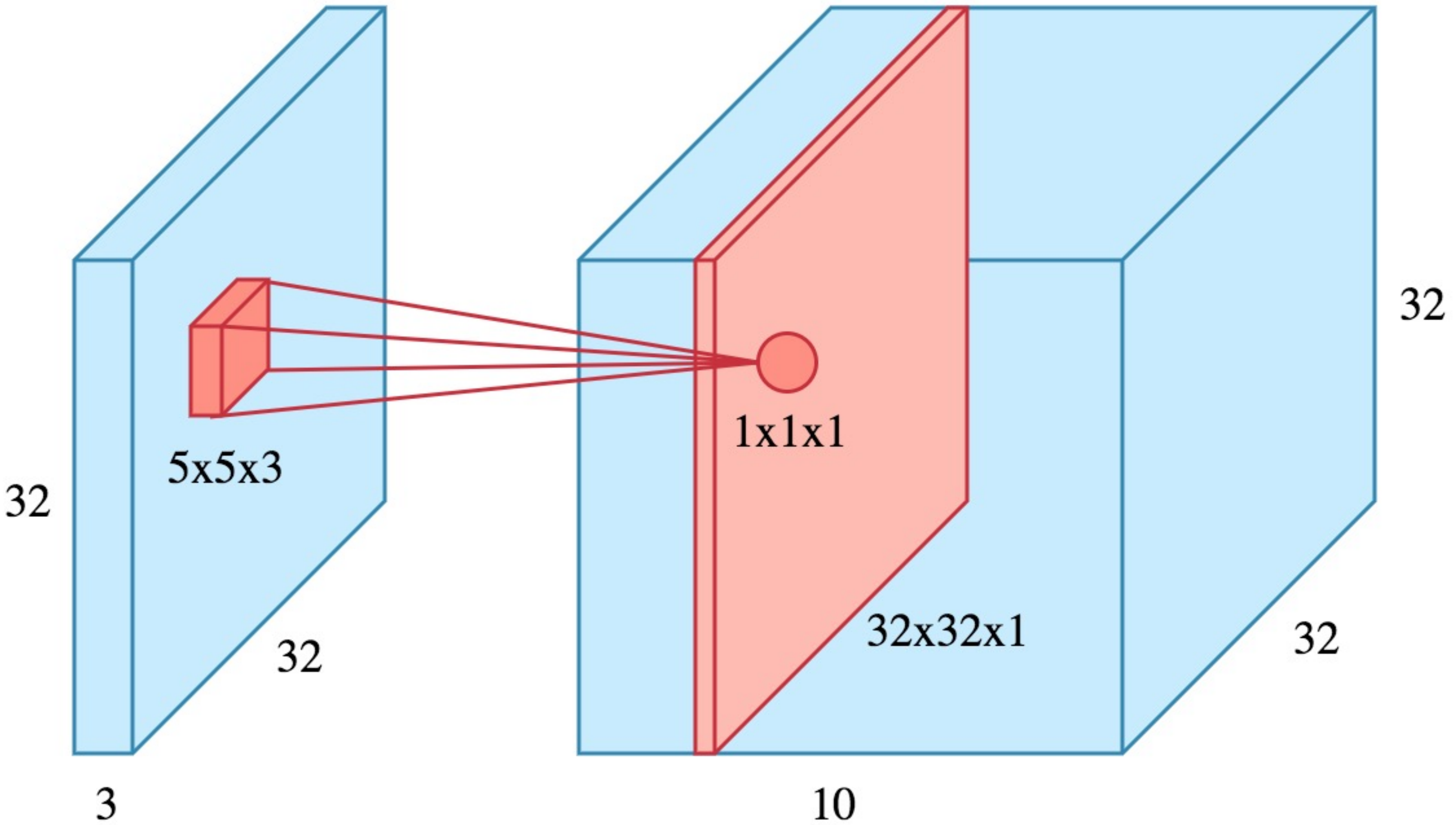


Example convolution operation shown in 2D using a 3x3 filter

Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer

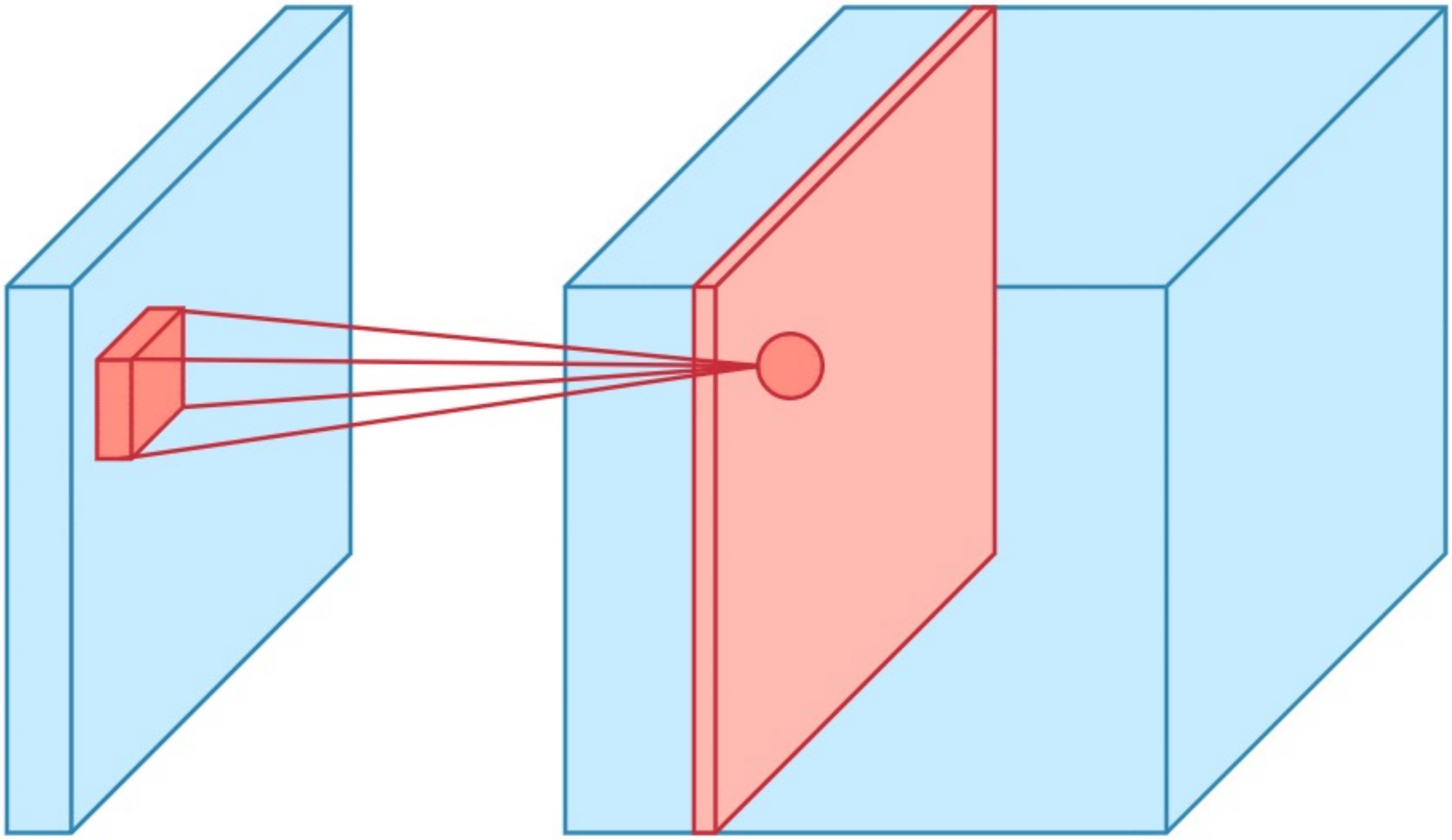
10 different filters 10 feature maps of size 32x32x1



final output of the convolution layer:
a volume of size 32x32x10

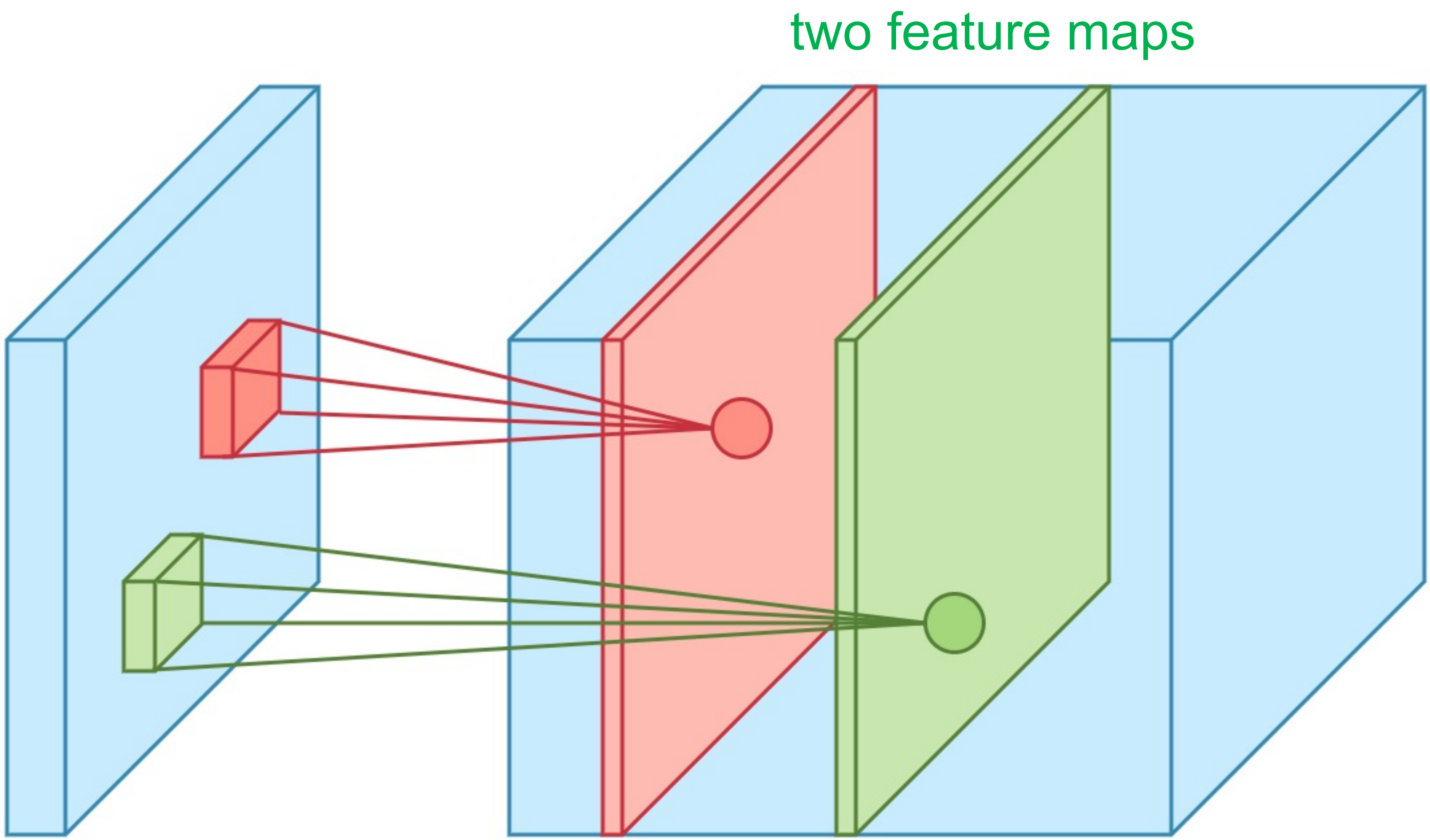
Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer
Sliding operation at 4 locations



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

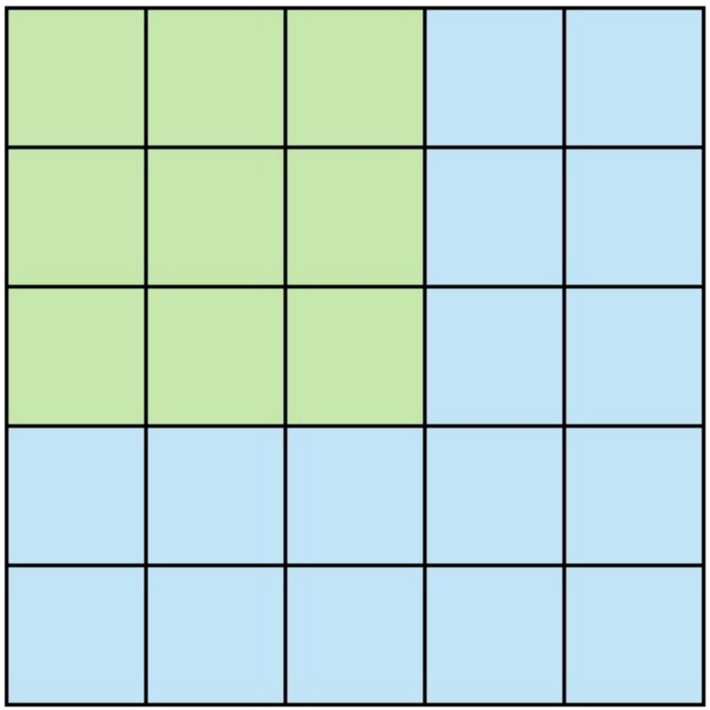
CNN Convolution Layer



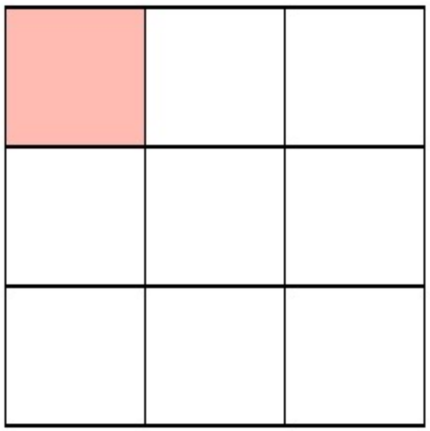
Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer

Stride specifies how much
we move the convolution filter at each step



Stride 1

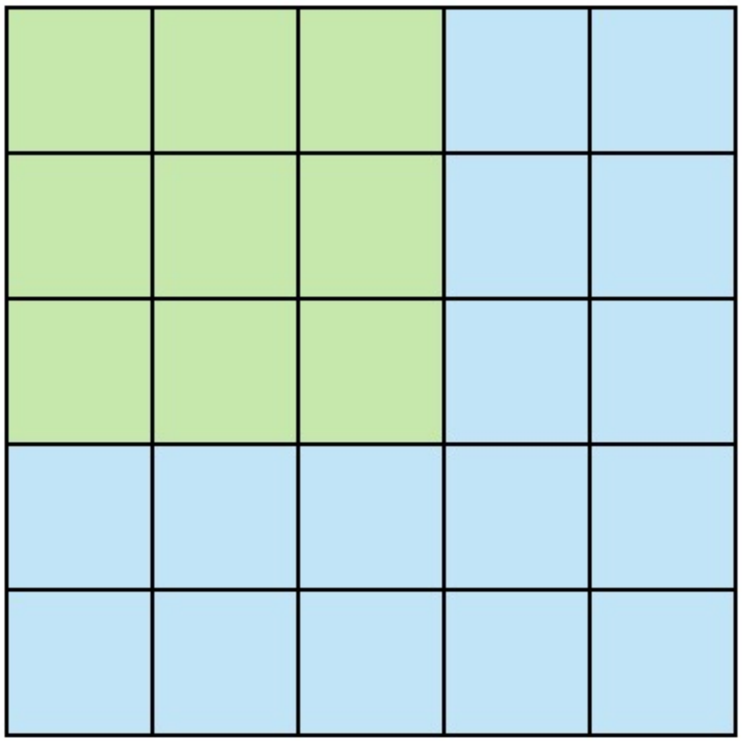


Feature Map

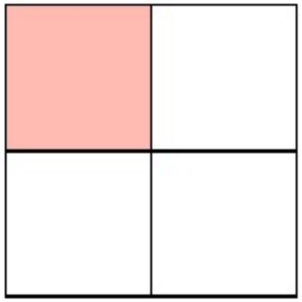
Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer

Stride specifies how much
we move the convolution filter at each step



Stride 2

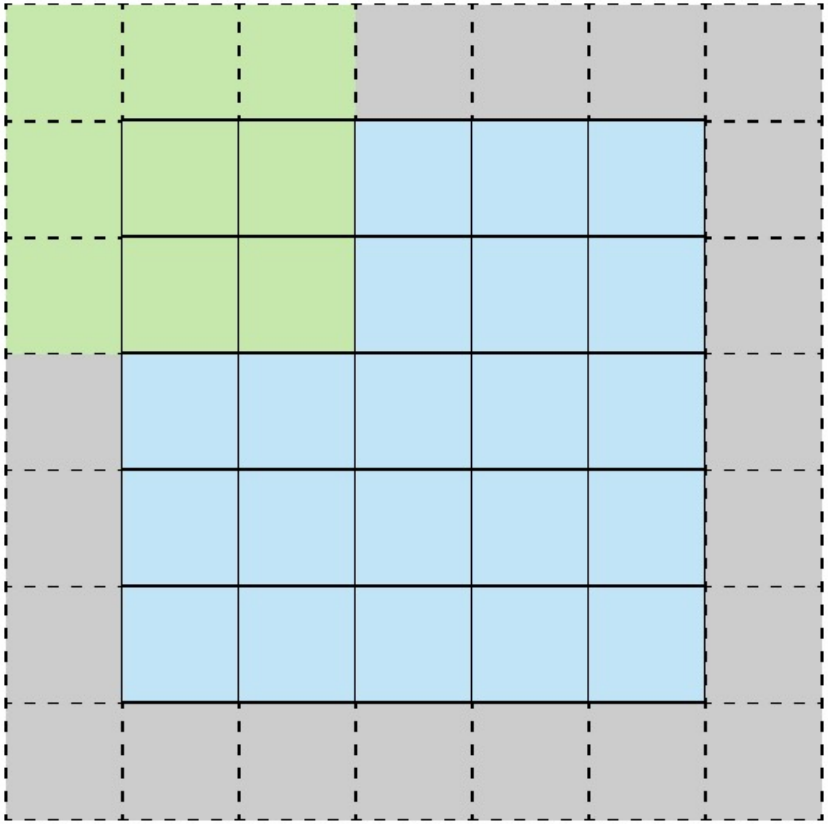


Feature Map

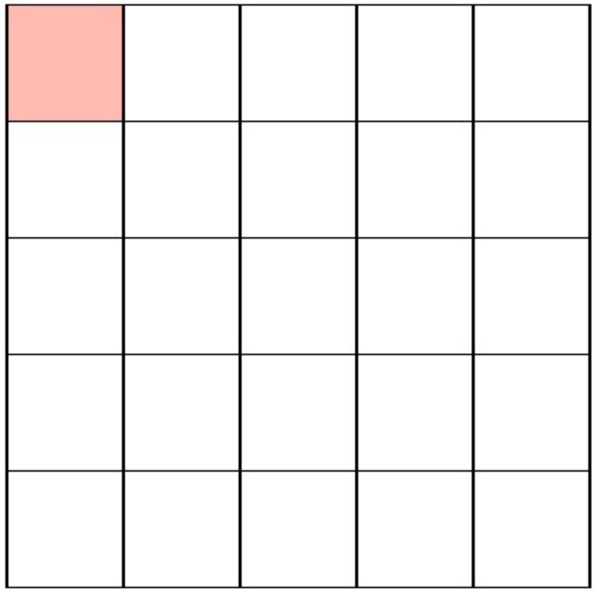
Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Convolution Layer

Stride 1 with Padding



Stride 1 with Padding

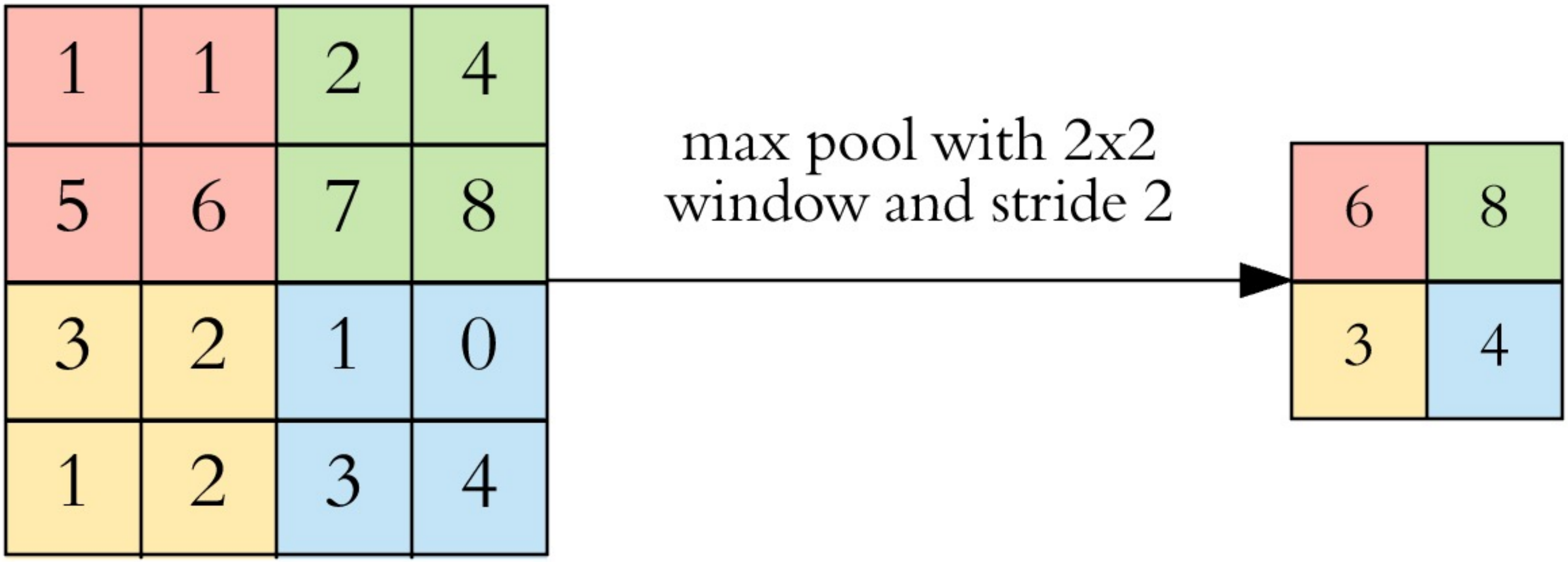


Feature Map

Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

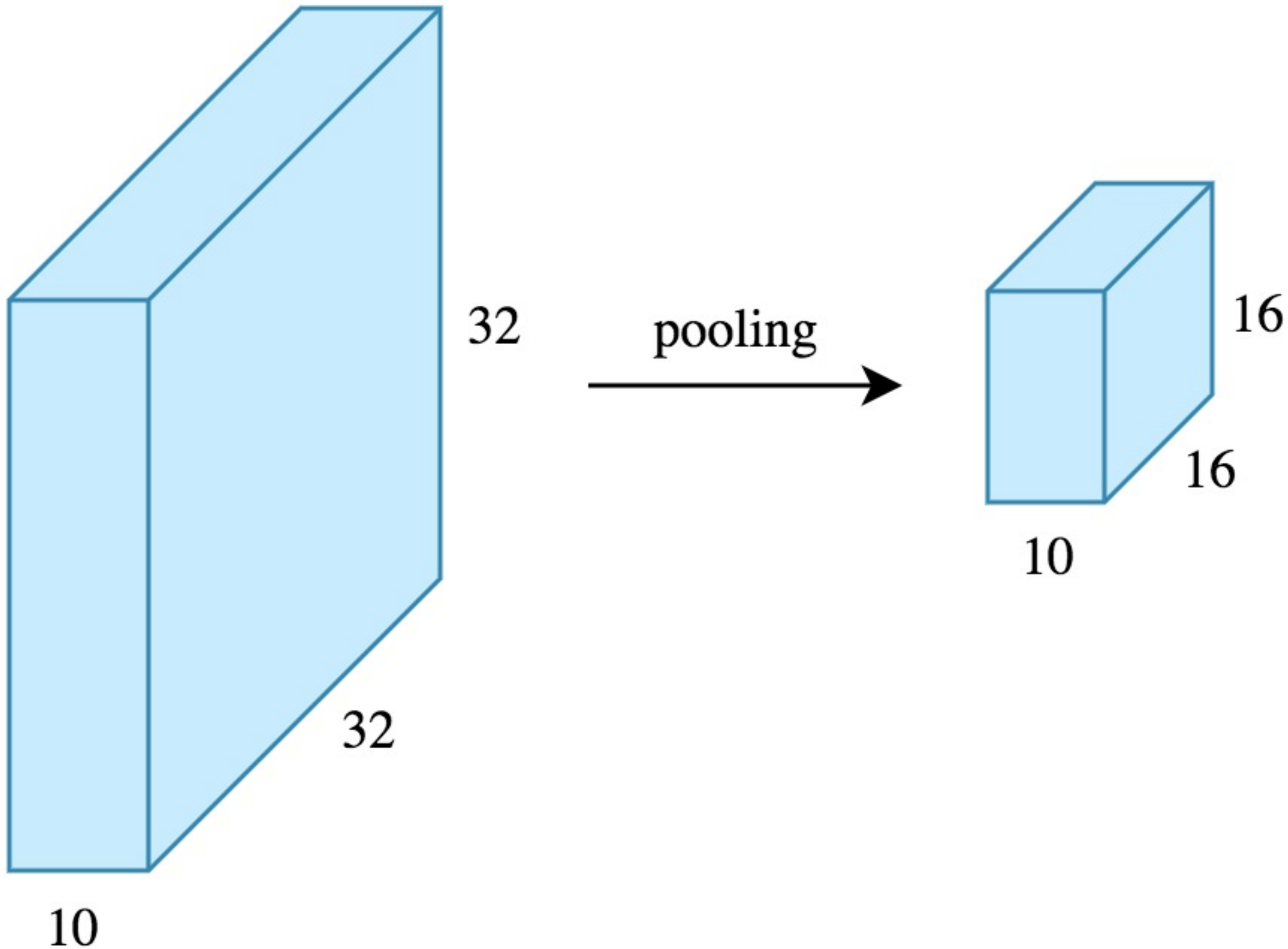
CNN Pooling Layer

Max Pooling



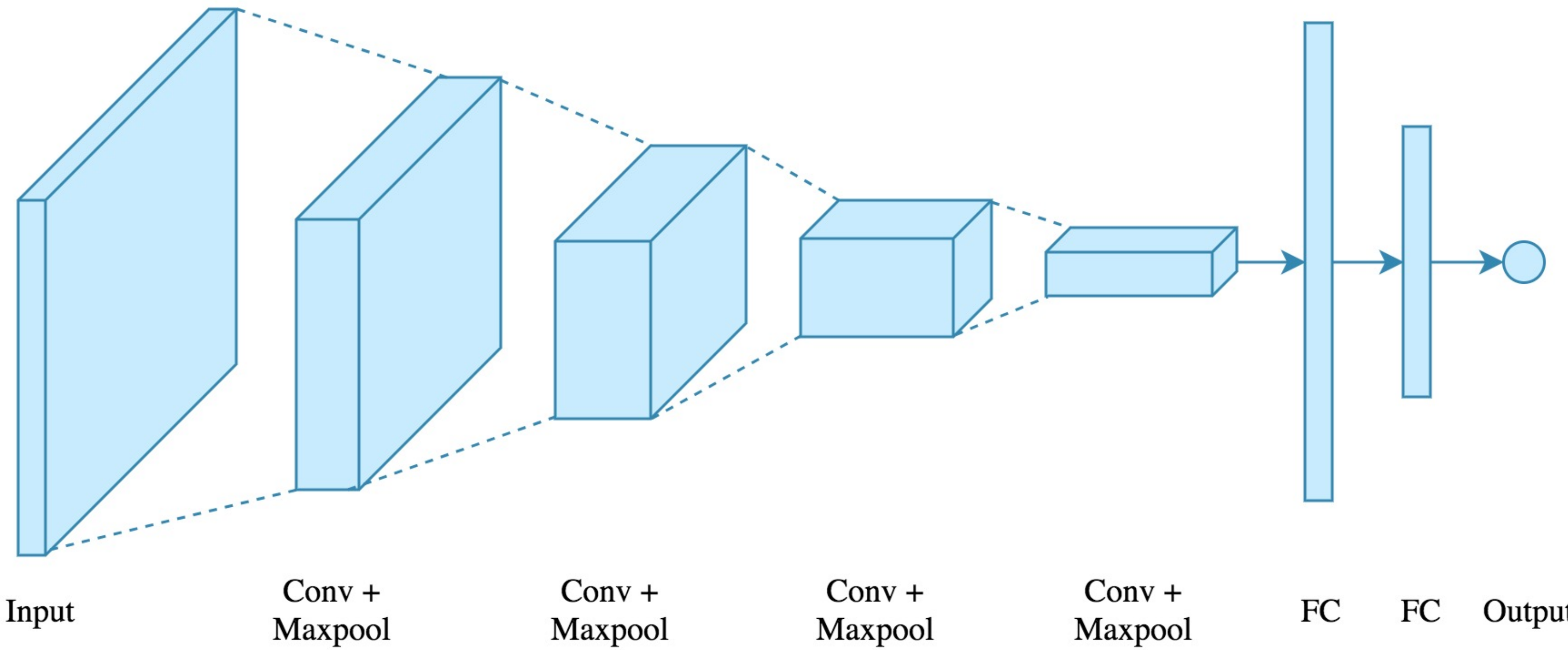
Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks,
https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Pooling Layer



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

CNN Architecture
4 convolution + pooling layers, followed by 2 fully connected layers



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

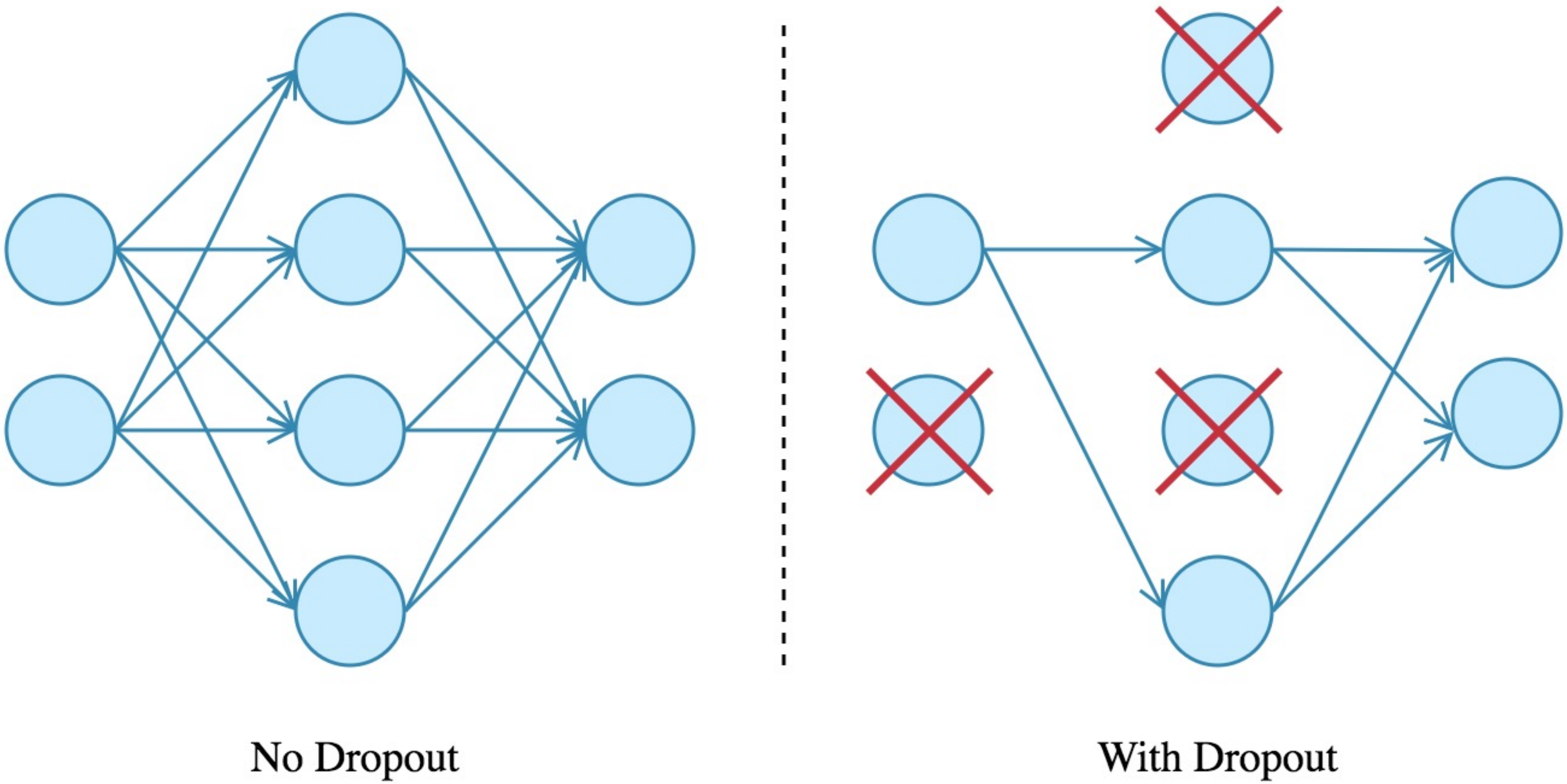
CNN Architecture
4 convolution + pooling layers, followed by 2 fully connected layers

https://gist.github.com/ardendertat/0fc5515057c47e7386fe04e9334504e3

```
model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', padding='same', name='conv_1',
                 input_shape=(150, 150, 3)))
model.add(MaxPooling2D((2, 2), name='maxpool_1'))
model.add(Conv2D(64, (3, 3), activation='relu', padding='same', name='conv_2'))
model.add(MaxPooling2D((2, 2), name='maxpool_2'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', name='conv_3'))
model.add(MaxPooling2D((2, 2), name='maxpool_3'))
model.add(Conv2D(128, (3, 3), activation='relu', padding='same', name='conv_4'))
model.add(MaxPooling2D((2, 2), name='maxpool_4'))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu', name='dense_1'))
model.add(Dense(128, activation='relu', name='dense_2'))
model.add(Dense(1, activation='sigmoid', name='output'))
```

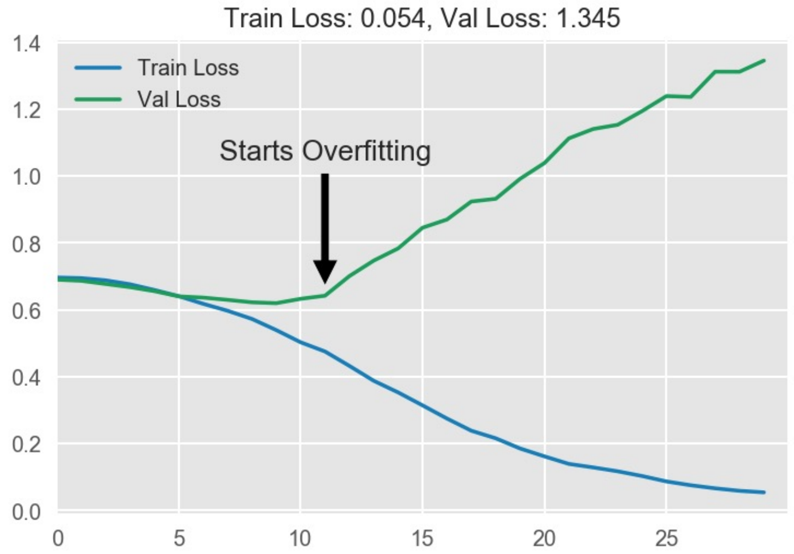
Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

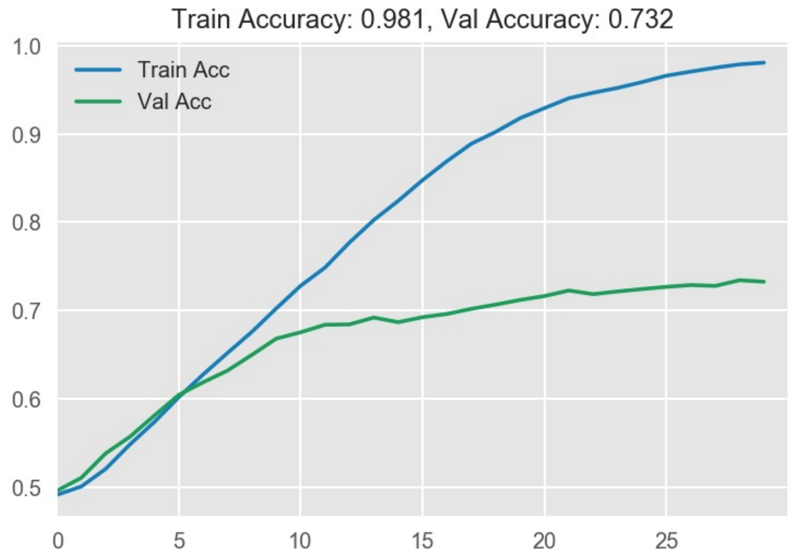
Dropout



Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Model Performance

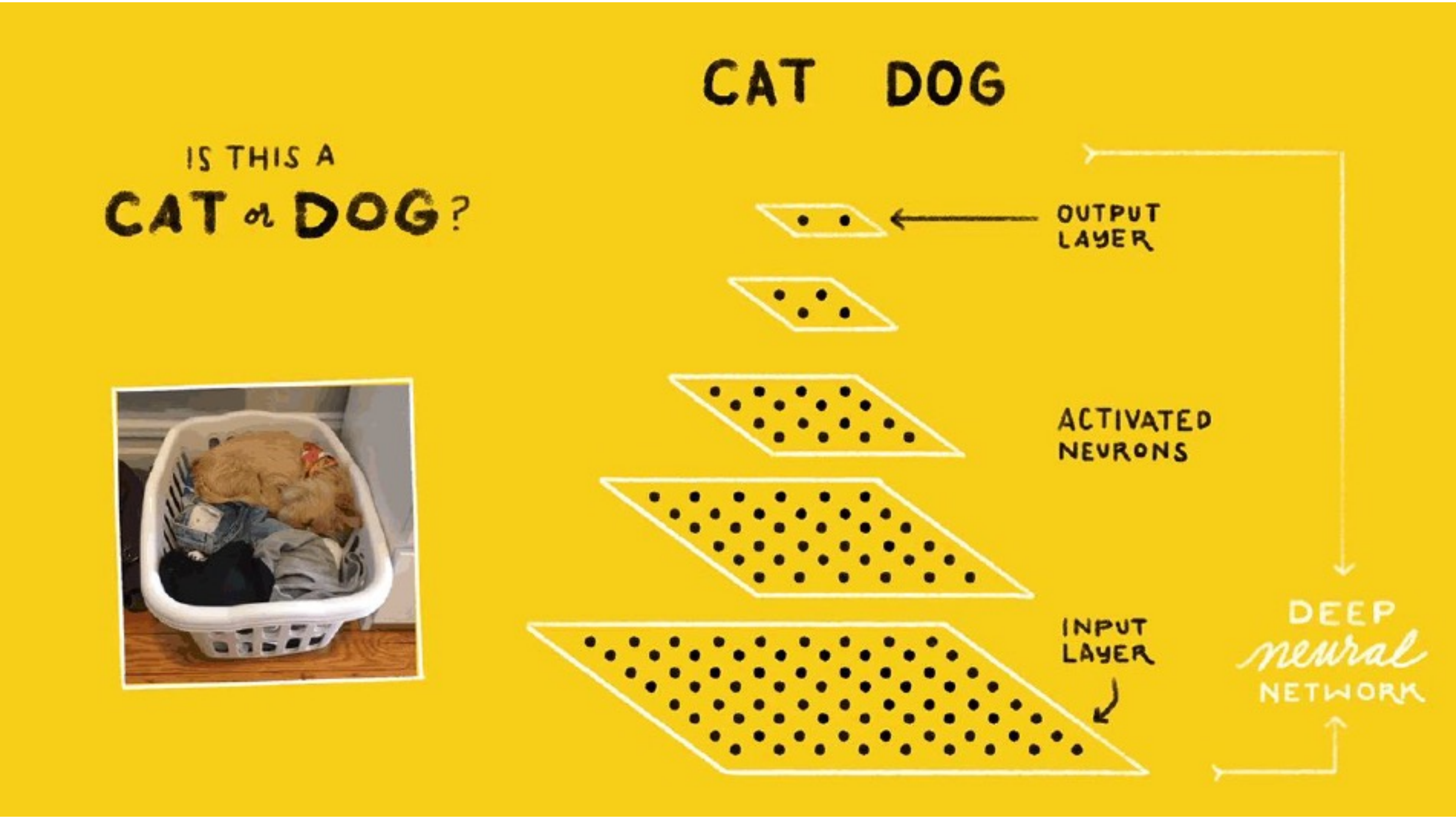




Source: Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

Visual Recognition

Image Classification

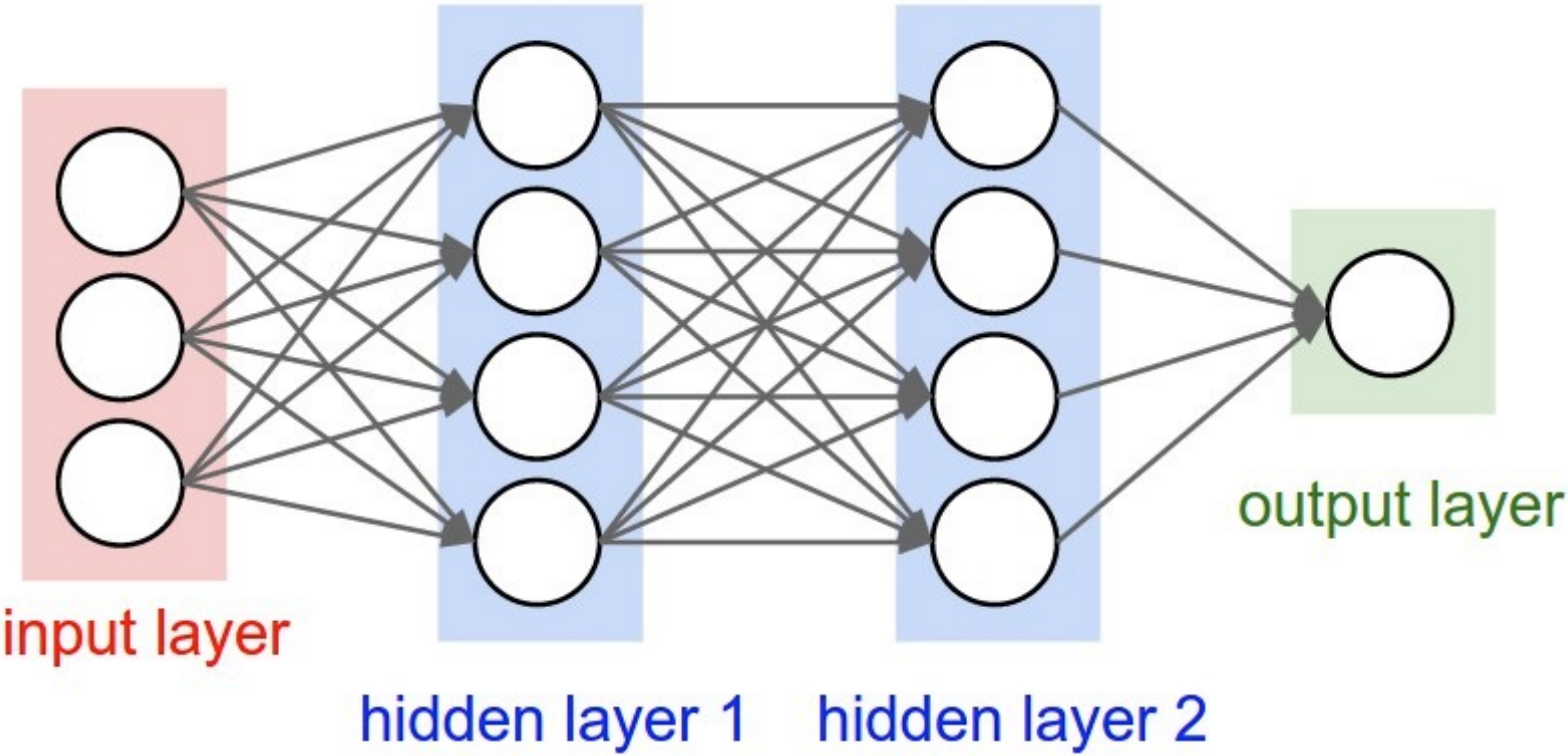


Source: Jeff Dean (2016), Large-Scale Deep Learning For Building Intelligent Computer Systems, WSDM 2016

Convolutional Neural Networks
(CNNs / ConvNets)

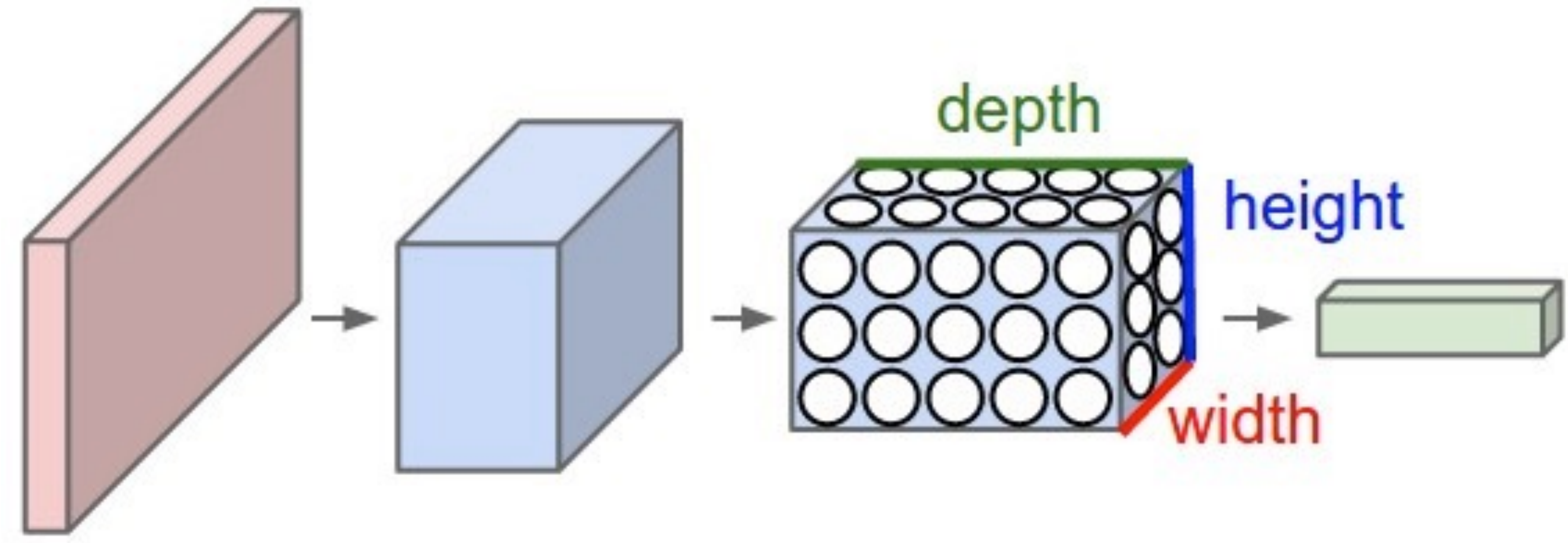
http://cs231n.github.io/convolutional-networks/

A regular 3-layer Neural Network



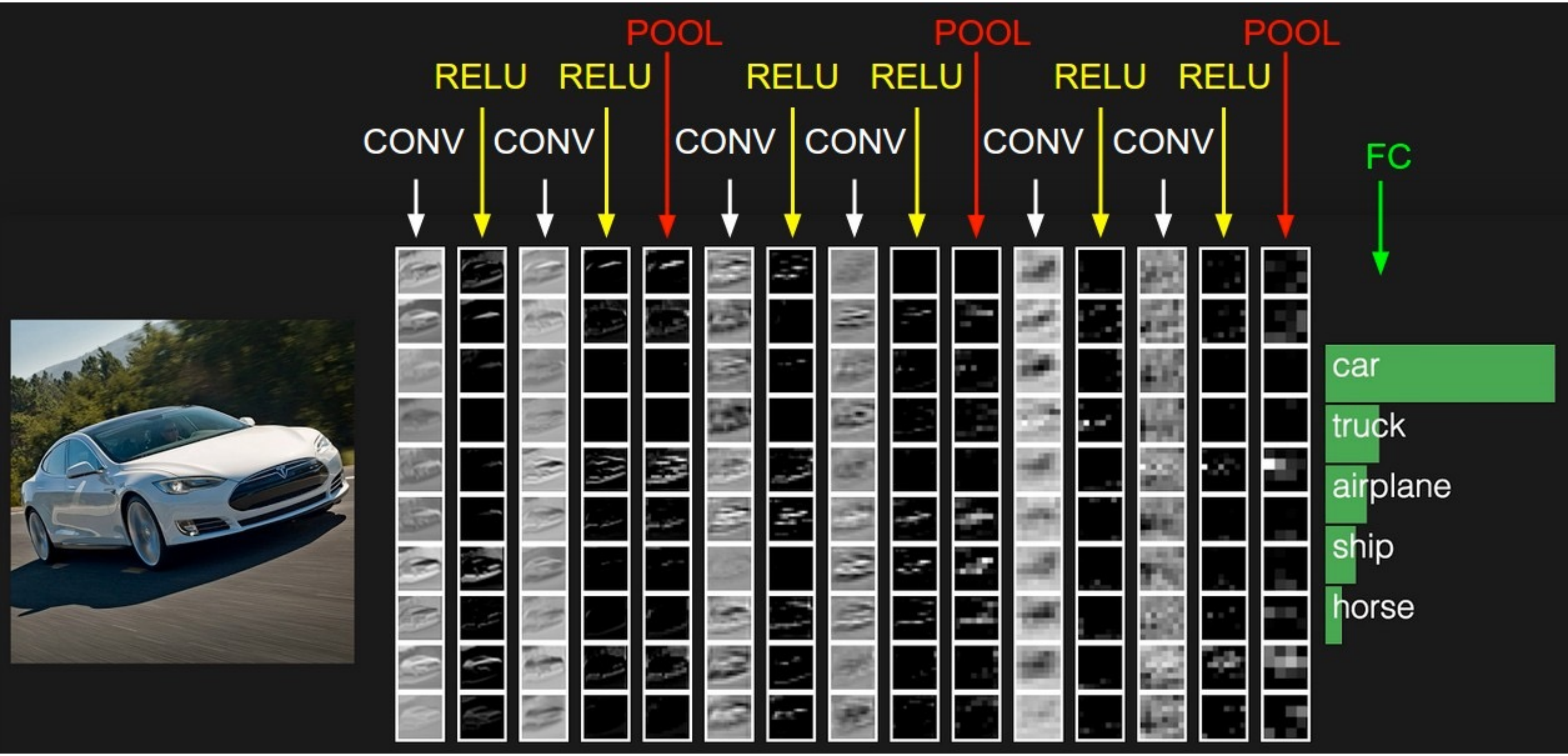
http://cs231n.github.io/convolutional-networks/

A ConvNet arranges its neurons in three dimensions (width, height, depth)



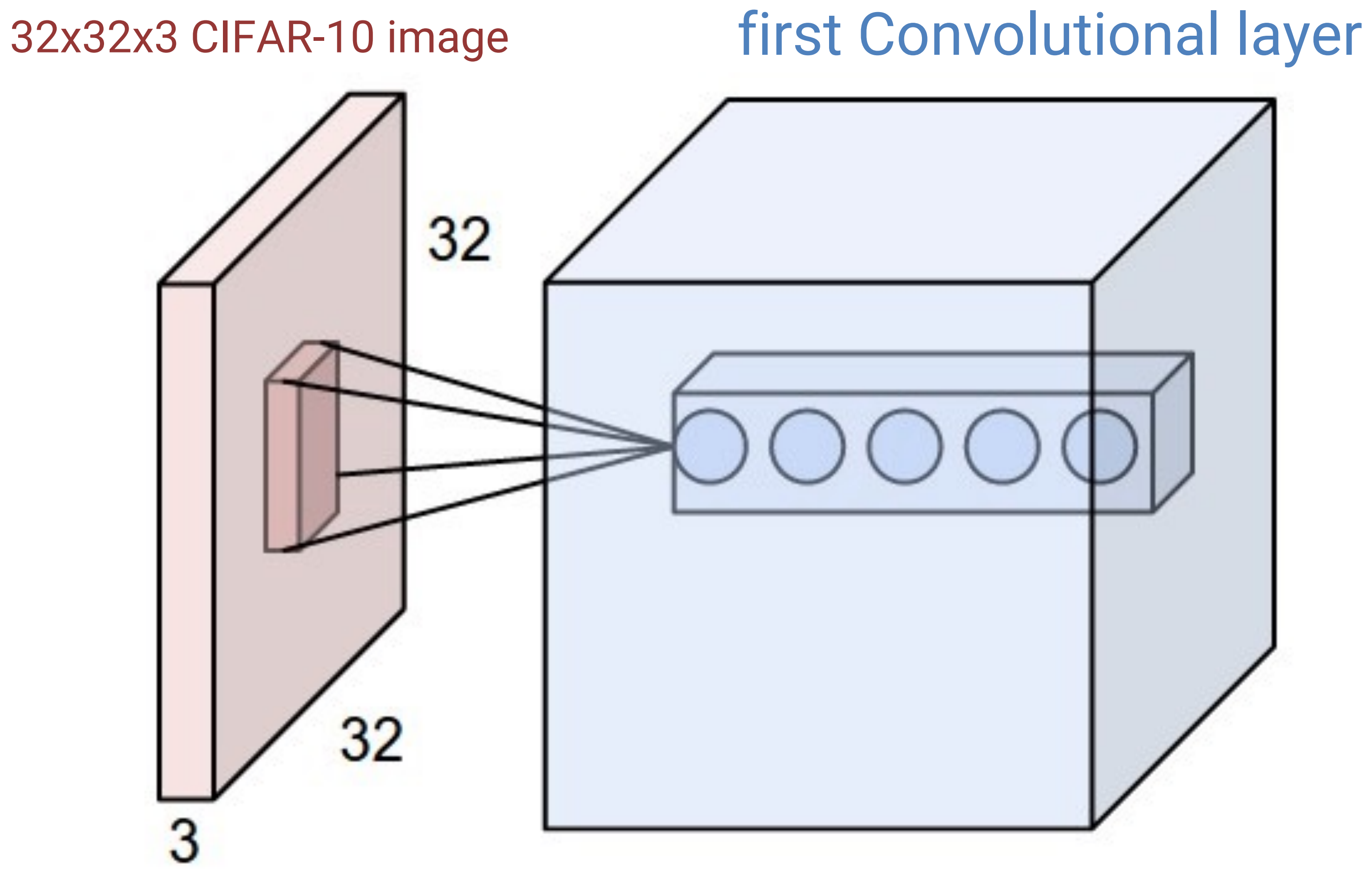
http://cs231n.github.io/convolutional-networks/

The activations of an example ConvNet architecture.



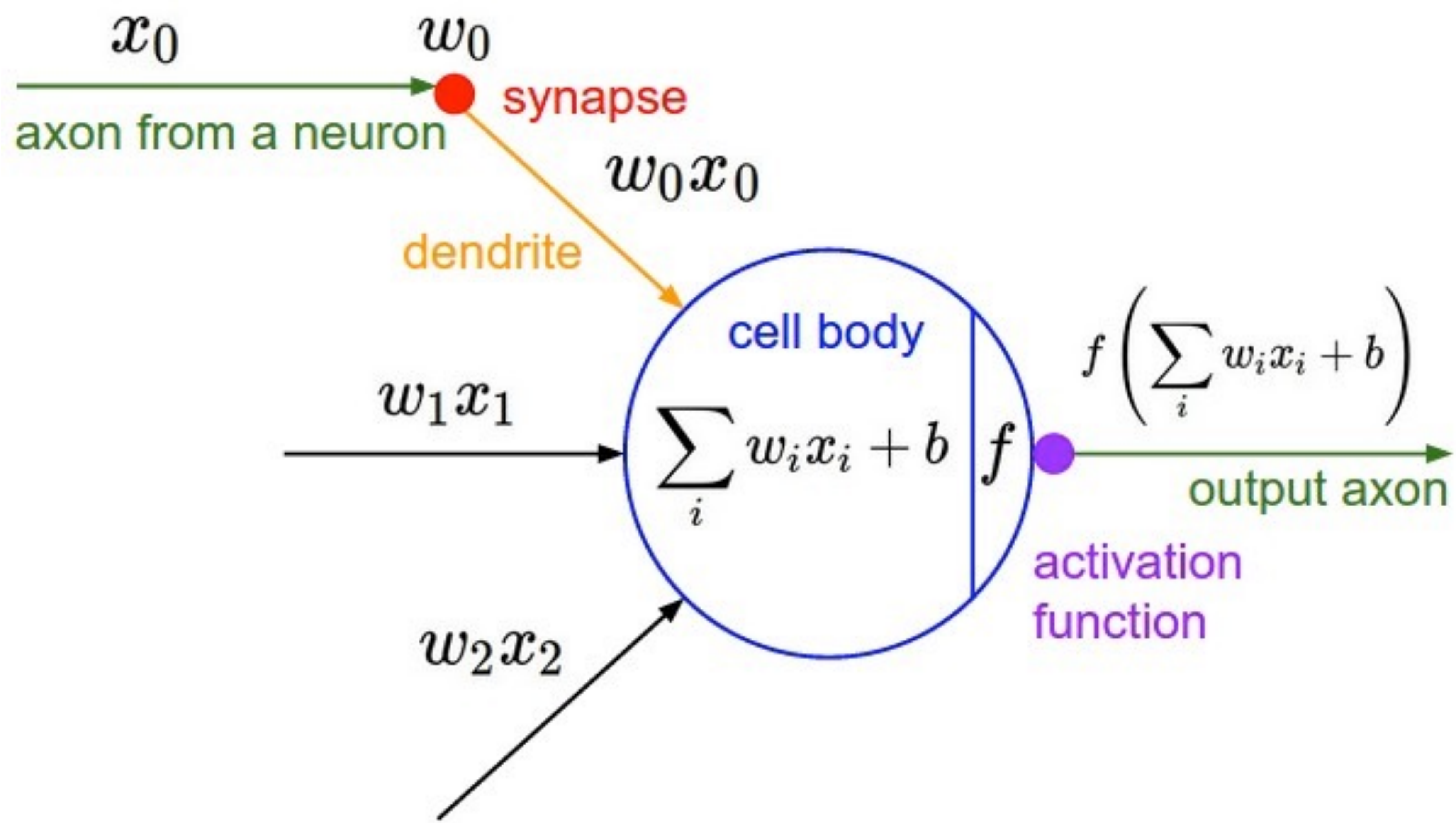
http://cs231n.github.io/convolutional-networks/

ConvNets



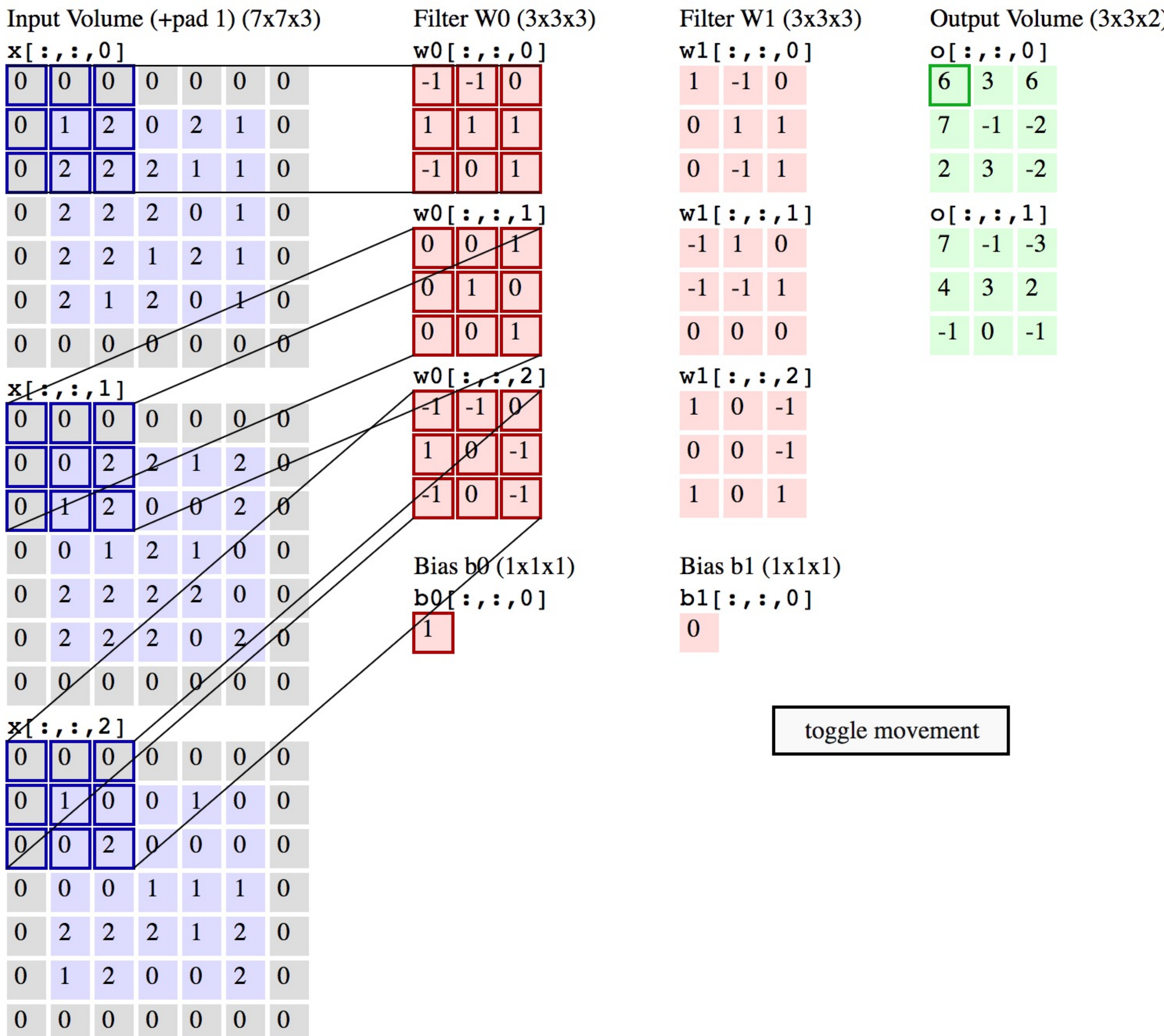
http://cs231n.github.io/convolutional-networks/

ConvNets



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

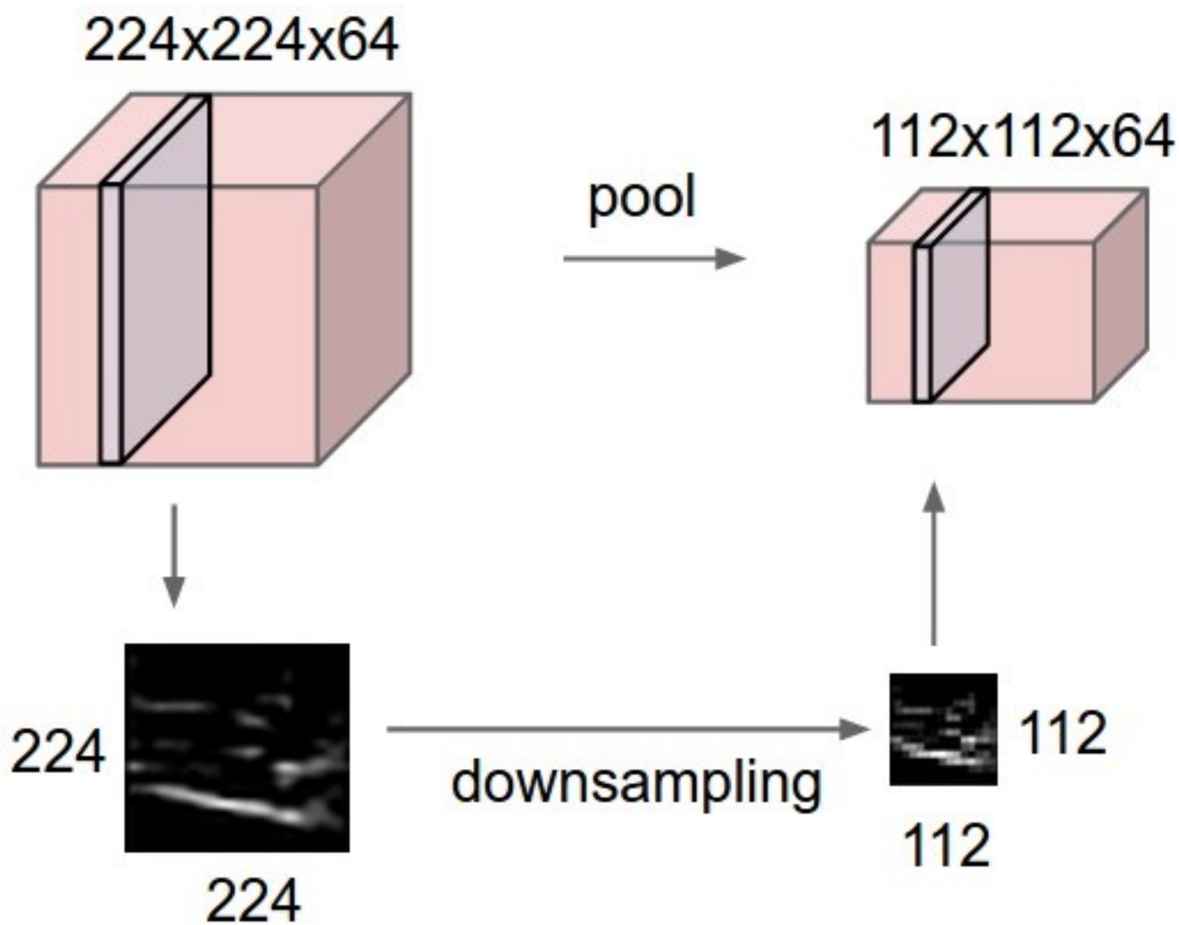
Convolution Demo



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

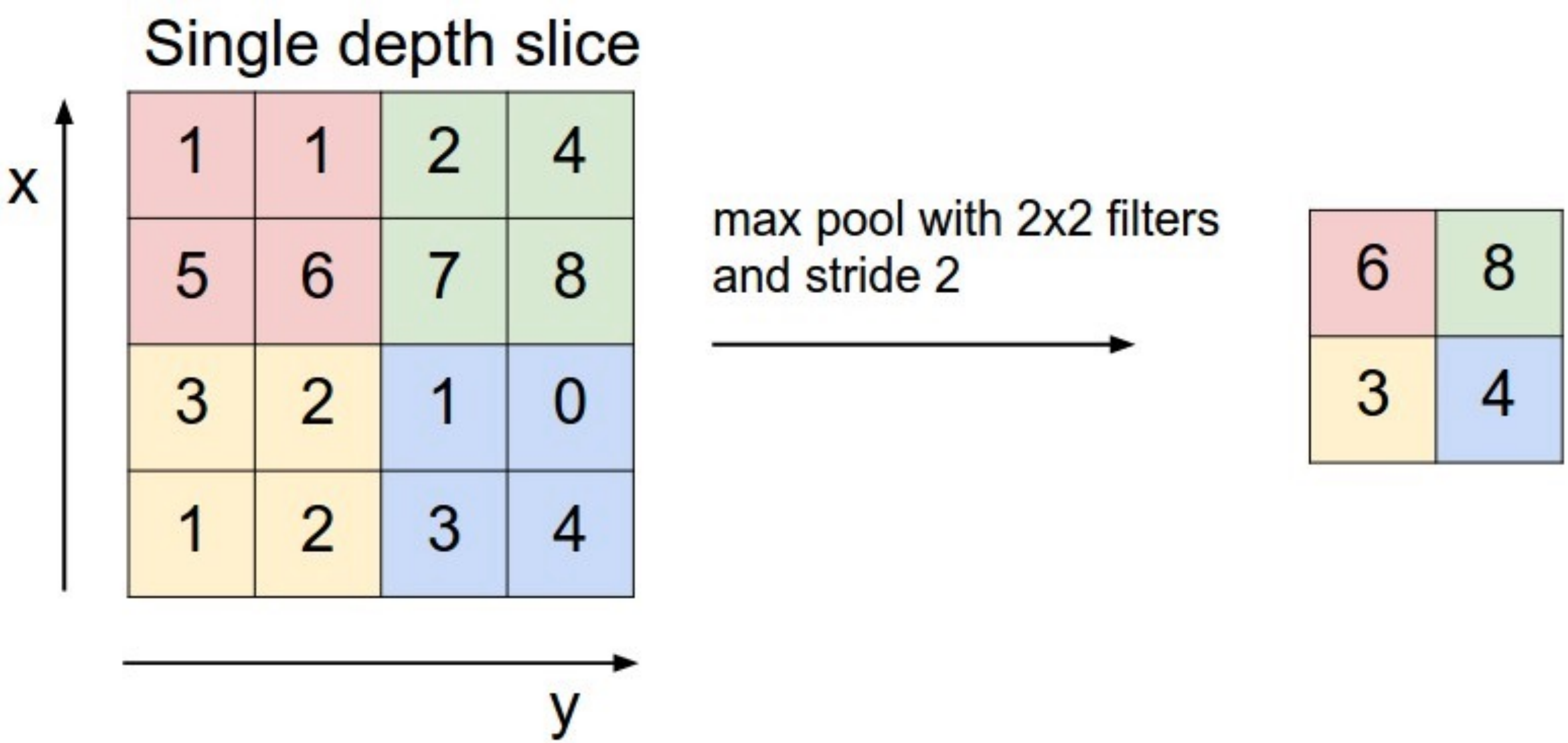
ConvNets

input volume of size [224x224x64]
is pooled with **filter** size 2, **stride** 2
into output volume of size [112x112x64]



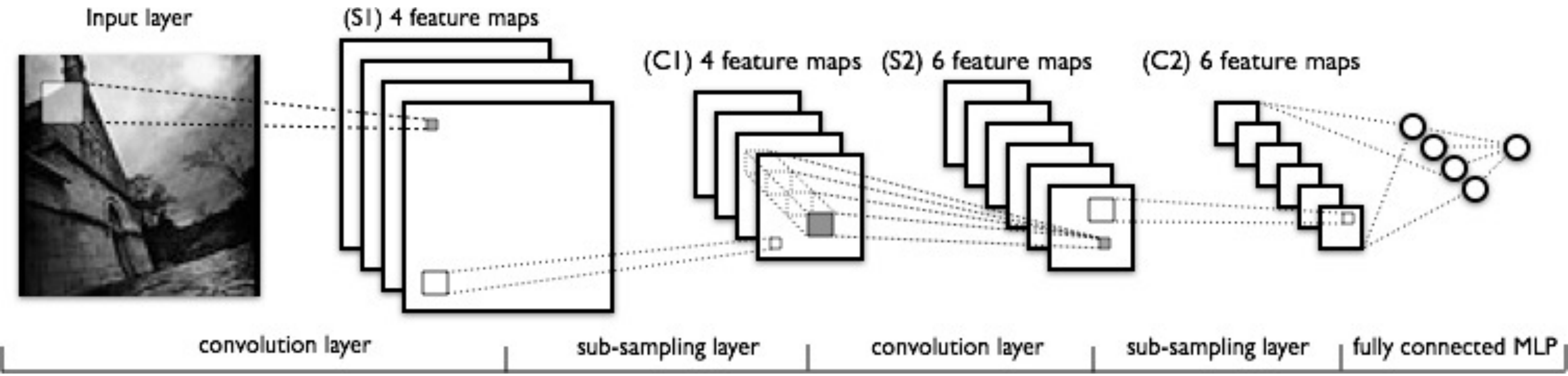
http://cs231n.github.io/convolutional-networks/

ConvNets
max pooling



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

Convolutional Neural Networks (CNN) (LeNet)



Source: http://deeplearning.net/tutorial/lenet.html

You Only Look Once
YOLO

You Only Look Once: Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

Abstract

We present YOLO, a new approach to object detection. Prior work on object detection repurposes classifiers to perform detection. Instead, we frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance.

Our unified architecture is extremely fast. Our base YOLO model processes images in real-time at 45 frames

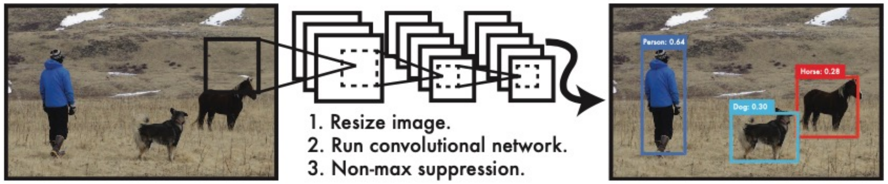
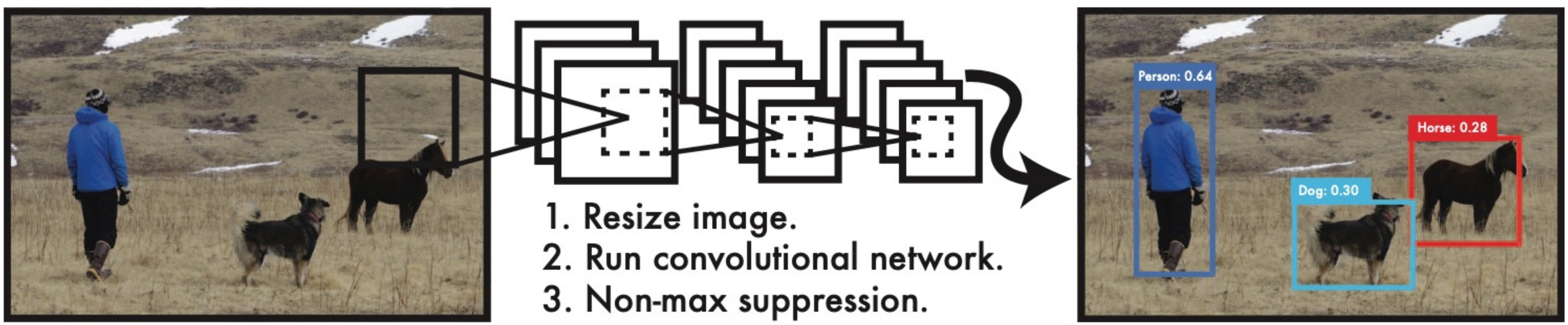


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

Source: Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi (2016). "You only look once: Unified, real-time object detection." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788. 2016.

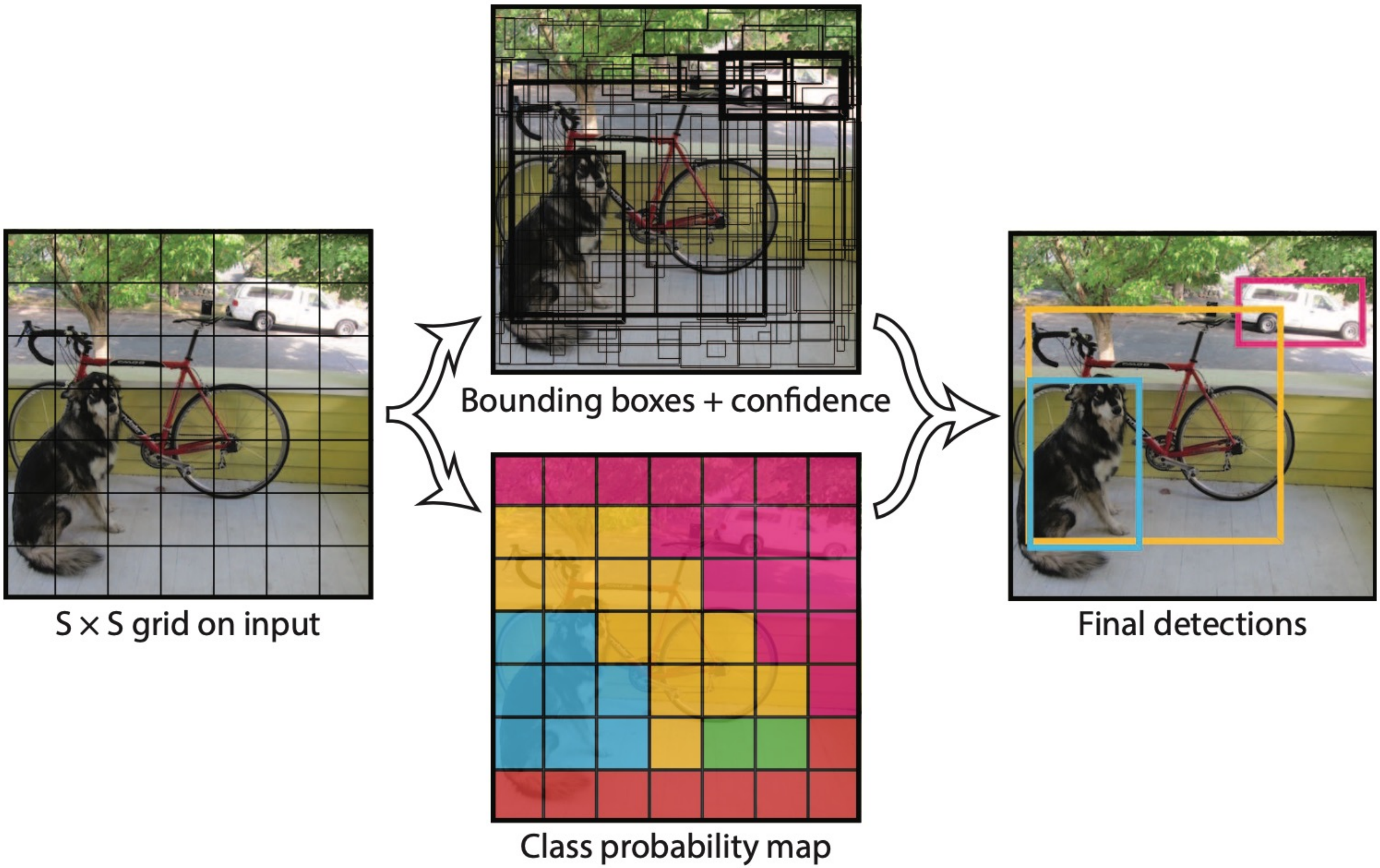
You Only Look Once
YOLO
The YOLO Detection System



(1) resizes the input image to 448 × 448,
(2) runs a single convolutional network on the image
(3) thresholds the resulting detections by the model's confidence.

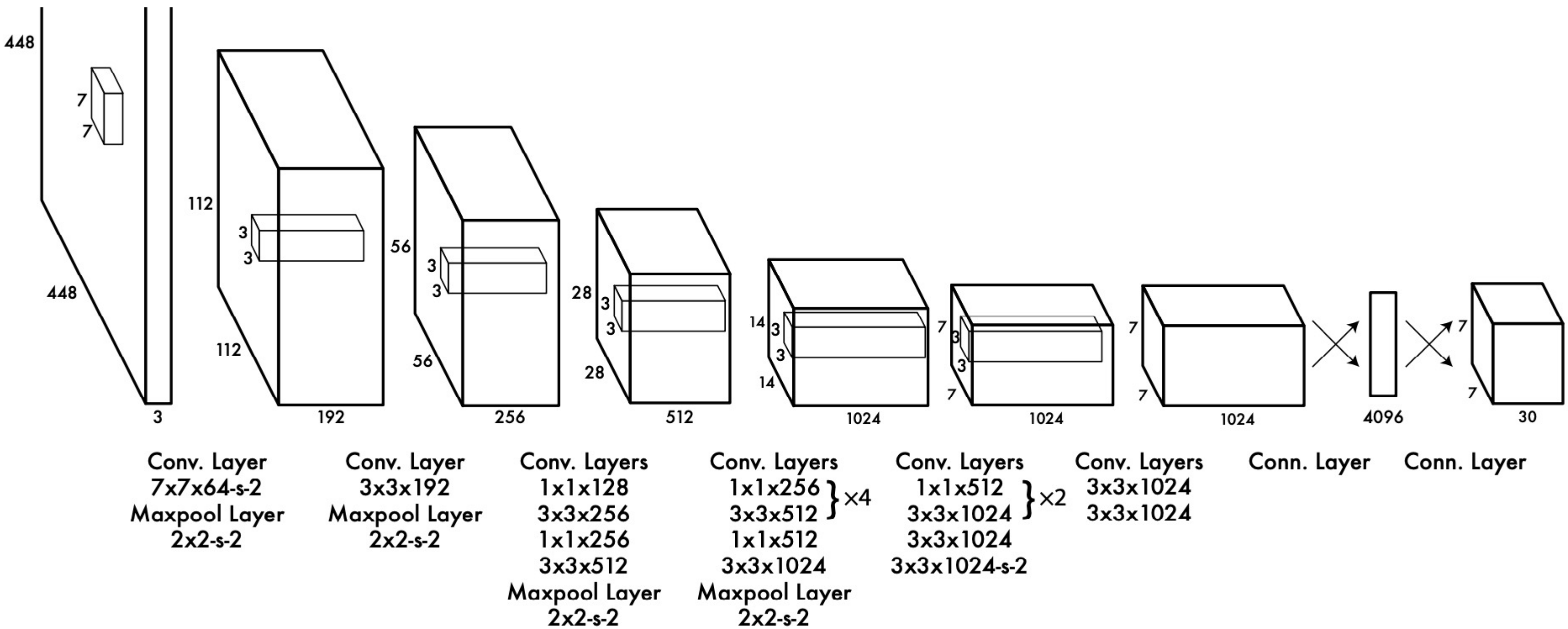
Source: Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi (2016). "You only look once: Unified, real-time object detection." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788. 2016.

You Only Look Once (YOLO) Model



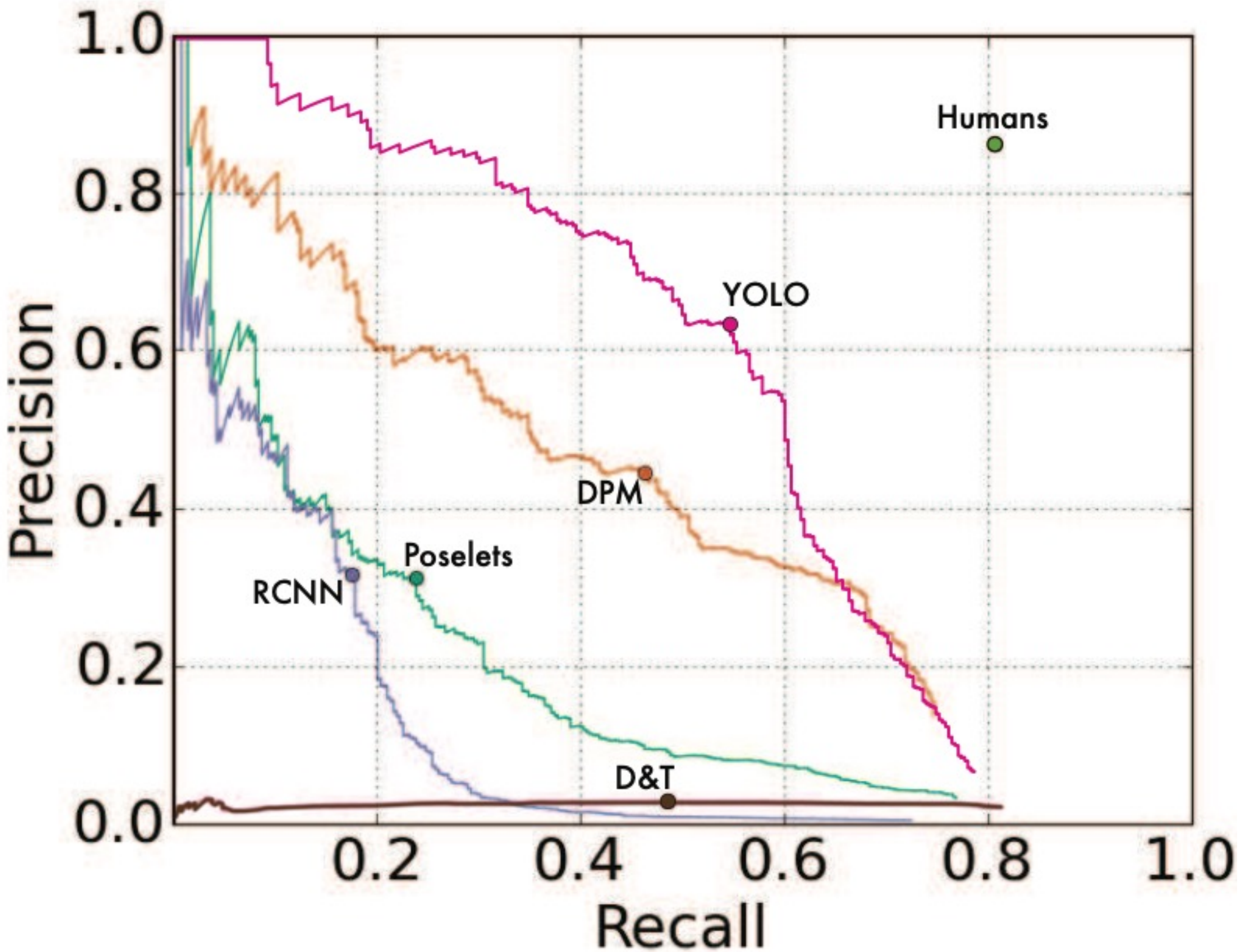
Source: Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi (2016). "You only look once: Unified, real-time object detection." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788. 2016.

You Only Look Once (YOLO) Unified, Real-Time Object Detection Architecture



Source: Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi (2016). "You only look once: Unified, real-time object detection." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788. 2016.

You Only Look Once (YOLO)
Picasso Dataset precision-recall curves



Source: Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi (2016). "You only look once: Unified, real-time object detection." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788. 2016.

YOLOv4

YOLOv4: Optimal Speed and Accuracy of Object Detection

Alexey Bochkovskiy*
alexeyab84@gmail.com

Chien-Yao Wang*
Institute of Information Science
Academia Sinica, Taiwan
kinyiu@iis.sinica.edu.tw

Hong-Yuan Mark Liao
Institute of Information Science
Academia Sinica, Taiwan
liao@iis.sinica.edu.tw

Abstract

There are a huge number of features which are said to improve Convolutional Neural Network (CNN) accuracy. Practical testing of combinations of such features on large datasets, and theoretical justification of the result, is required. Some features operate on certain models exclusively and for certain problems exclusively, or only for small-scale datasets; while some features, such as batch-normalization and residual-connections, are applicable to the majority of models, tasks, and datasets. We assume that such universal features include Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), Cross mini-Batch Normalization (CmBN), Self-adversarial-training (SAT) and Mish-activation. We use new features: WRC, CSP, CmBN, SAT, Mish activation, Mosaic data augmentation, CmBN, DropBlock regularization, and CIoU loss, and combine some of them to achieve state-of-the-art results: 43.5% AP (65.7% AP_{50}) for the MS COCO dataset at a real-time speed of ~65 FPS on Tesla V100. Source code is at https://github.com/AlexeyAB/darknet.

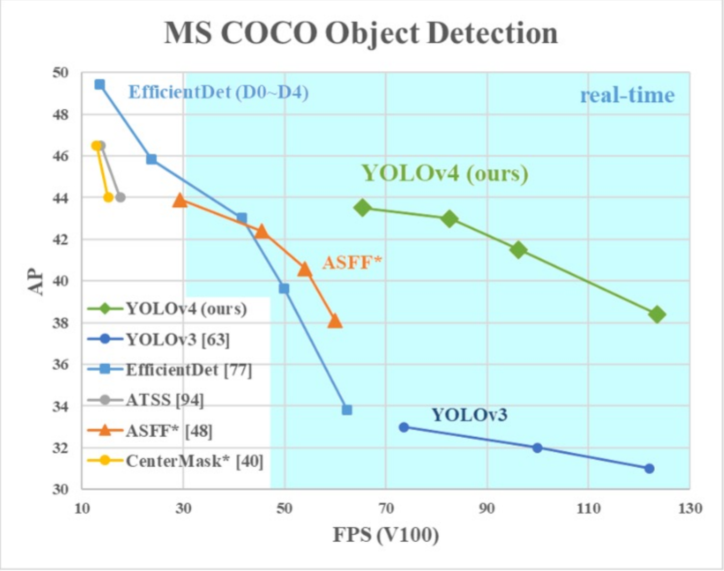
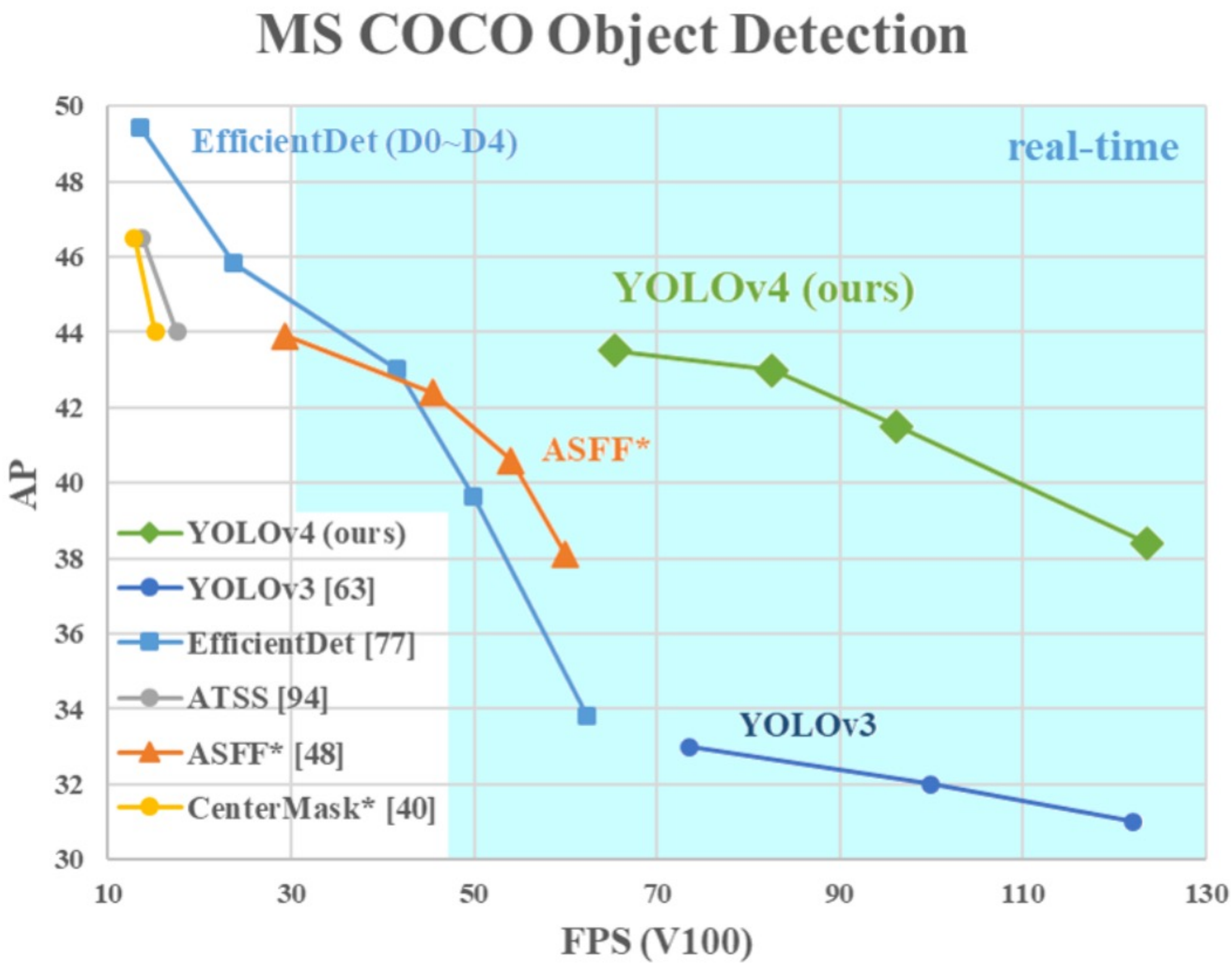


Figure 1: Comparison of the proposed YOLOv4 and other state-of-the-art object detectors. YOLOv4 runs twice faster than EfficientDet with comparable performance. Improves YOLOv3's AP and FPS by 10% and 12%, respectively.

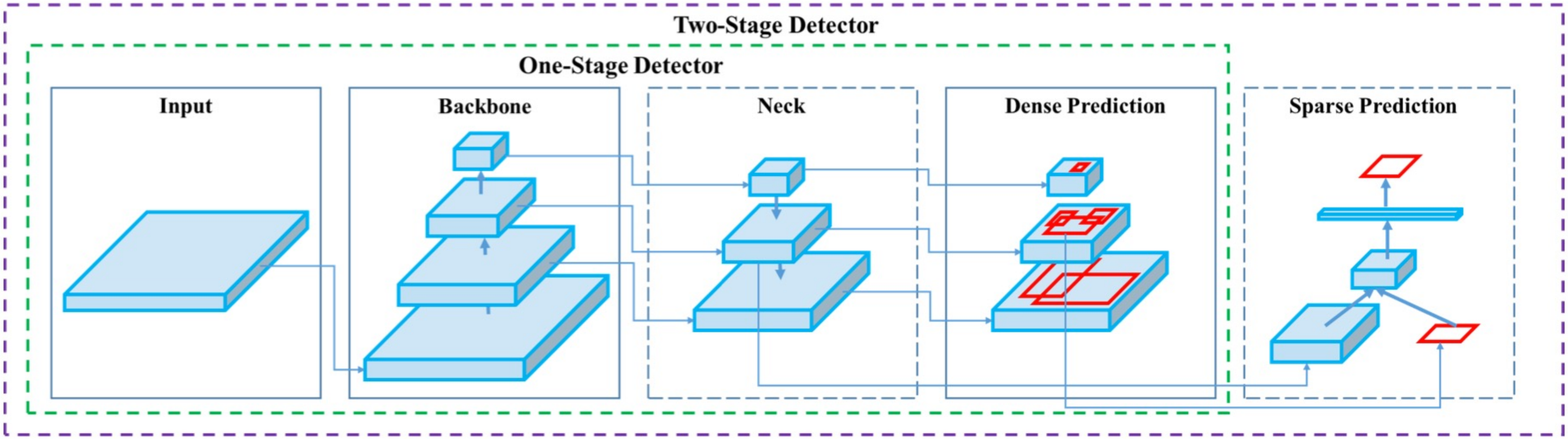
Source: Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao (2020). "Yolov4: Optimal speed and accuracy of object detection." arXiv preprint arXiv:2004.10934 (2020).

YOLOv4



Source: Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao (2020). "Yolov4: Optimal speed and accuracy of object detection." arXiv preprint arXiv:2004.10934 (2020).

YOLOv4:
Optimal speed and accuracy of object detection



Input: { Image, Patches, Image Pyramid, … }

Backbone: { VGG16 [68], ResNet-50 [26], ResNeXt-101 [86], Darknet53 [63], … }

Neck: { FPN [44], PANet [49], Bi-FPN [77], … }

Head:

- **Dense Prediction:** { RPN [64], YOLO [61, 62, 63], SSD [50], RetinaNet [45], FCOS [78], … }
- **Sparse Prediction:** { Faster R-CNN [64], R-FCN [9], … }

Source: Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao (2020). "Yolov4: Optimal speed and accuracy of object detection." arXiv preprint arXiv:2004.10934 (2020).

EfficientDet

EfficientDet: Scalable and Efficient Object Detection

Mingxing Tan Ruoming Pang Quoc V. Le
Google Research, Brain Team
{tanmingxing, rpang, qvl}@google.com

Abstract

Model efficiency has become increasingly important in computer vision. In this paper, we systematically study neural network architecture design choices for object detection and propose several key optimizations to improve efficiency. First, we propose a weighted bi-directional feature pyramid network (BiFPN), which allows easy and fast multi-scale feature fusion; Second, we propose a compound scaling method that uniformly scales the resolution, depth, and width for all backbone, feature network, and box/class prediction networks at the same time. Based on these optimizations and EfficientNet backbones, we have developed a new family of object detectors, called EfficientDet, which consistently achieve much better efficiency than prior art across a wide spectrum of resource constraints. In particular, with single-model and single-scale, our EfficientDet-D7 achieves state-of-the-art ***52.2 AP*** *on COCO* `test-dev` *with 52M parameters and 325B FLOPs*[1]*, being* ***4x – 9x*** *smaller and using* ***13x – 42x*** *fewer FLOPs than previous detector. Code is available at* `https://github.com/google/automl/tree/master/efficientdet`.

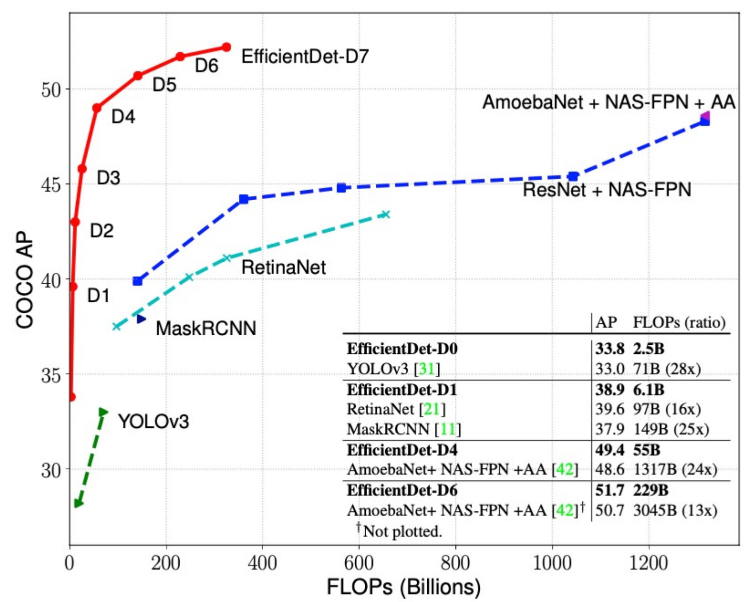
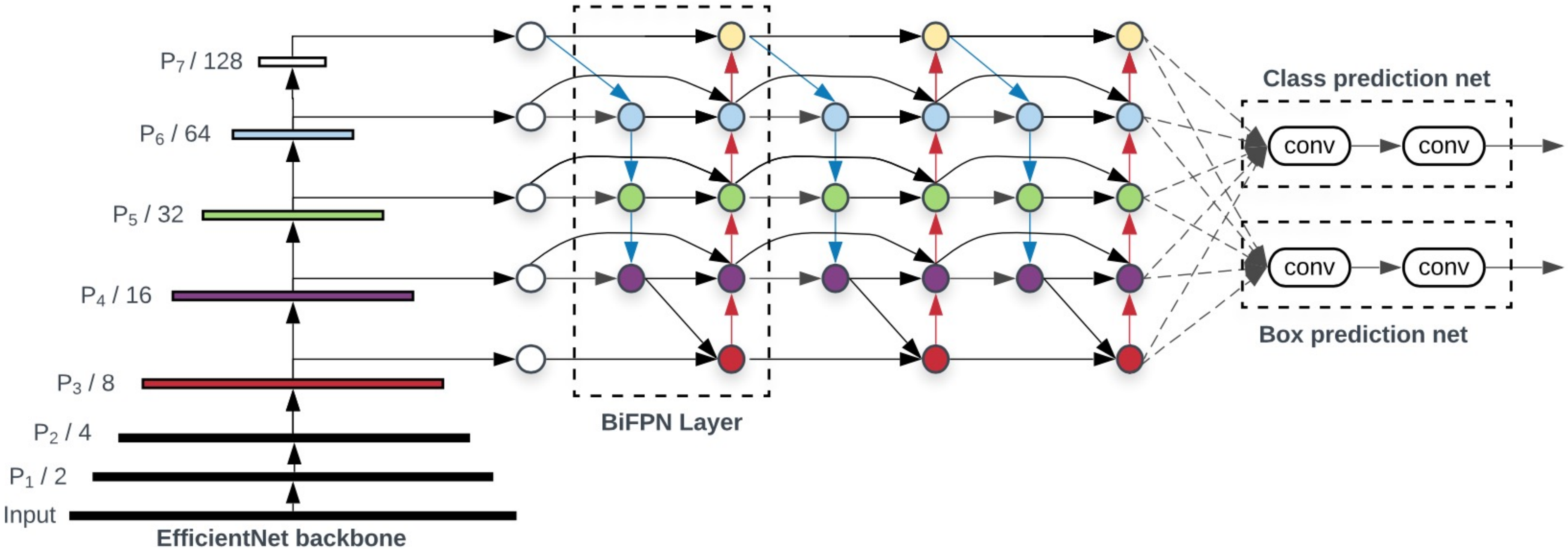


Figure 1: **Model FLOPs vs. COCO accuracy** – All numbers are for single-model single-scale. Our EfficientDet achieves new state-of-the-art 52.2% COCO AP with much fewer parameters and FLOPs than previous detectors. More studies on different backbones and FPN/NAS-FPN/BiFPN are in Table 4 and 5. Complete results are in Table 2.

Source: Mingxing Tan, Ruoming Pang, and Quoc V. Le (2020). "Efficientdet: Scalable and efficient object detection." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 10781-10790. 2020.

EfficientDet:
Scalable and Efficient Object Detection

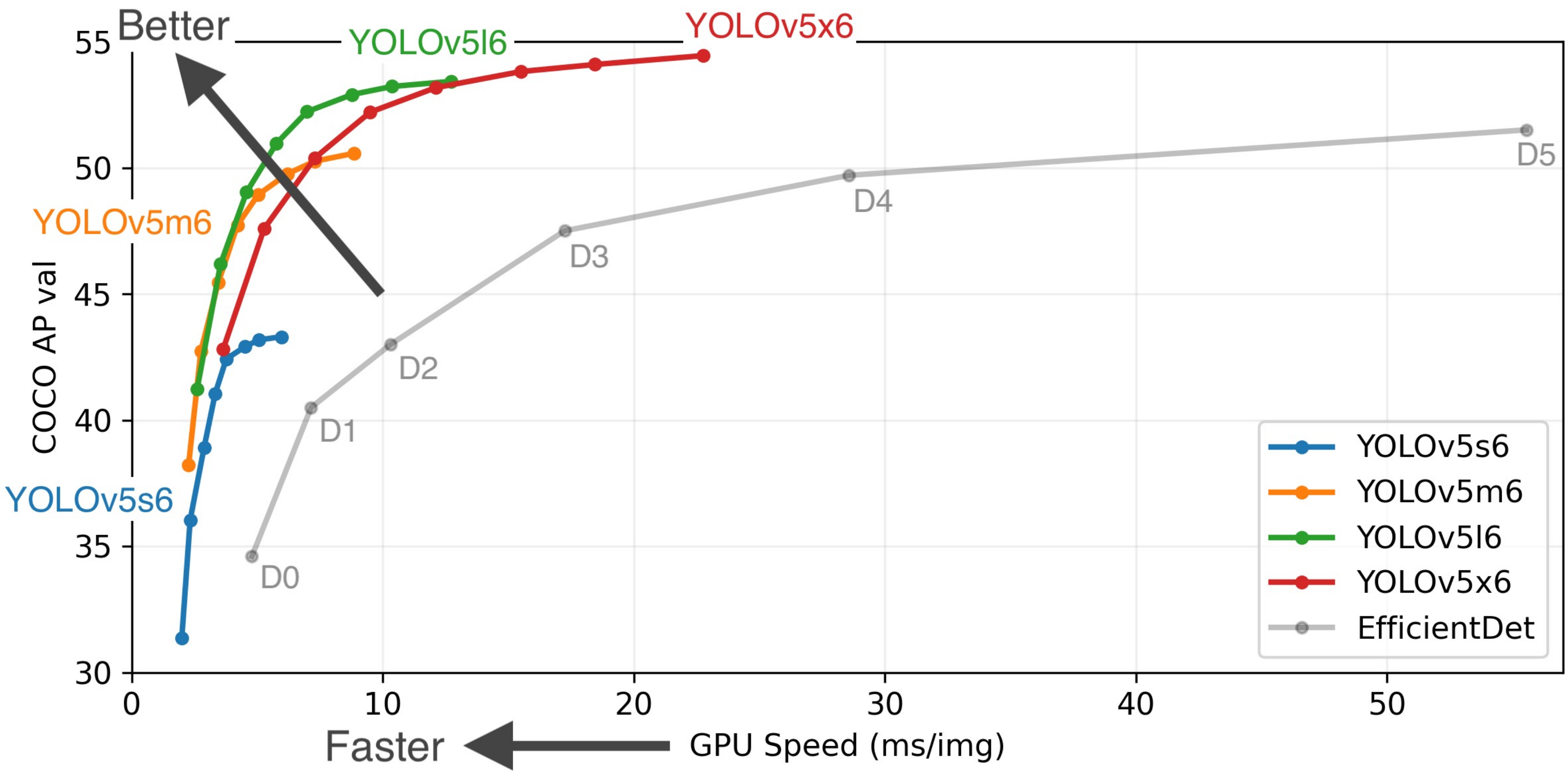


Source: Mingxing Tan, Ruoming Pang, and Quoc V. Le (2020). "Efficientdet: Scalable and efficient object detection." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 10781-10790. 2020.

YOLOv5

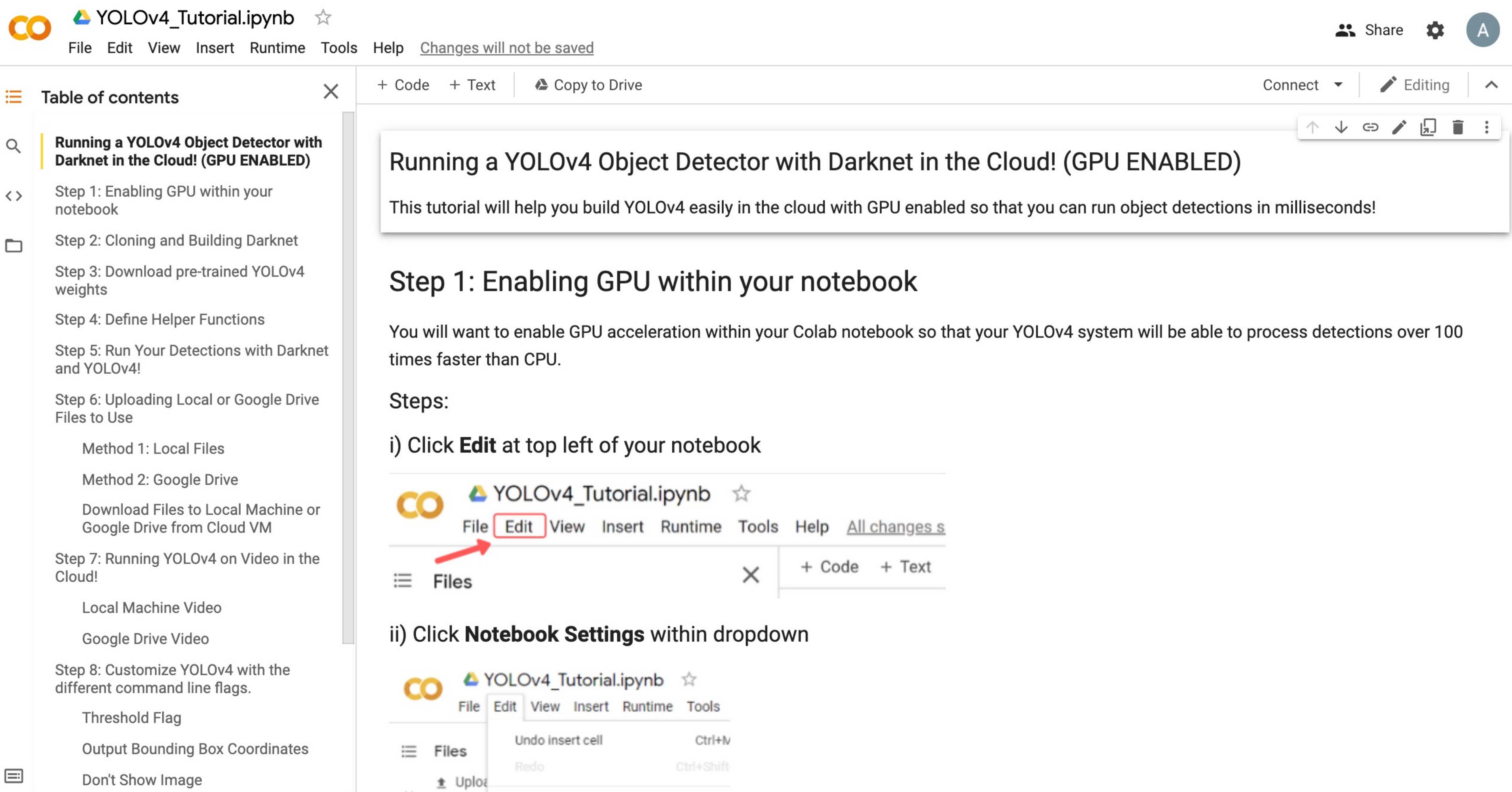
Source: https://github.com/ultralytics/yolov5

YOLOv5



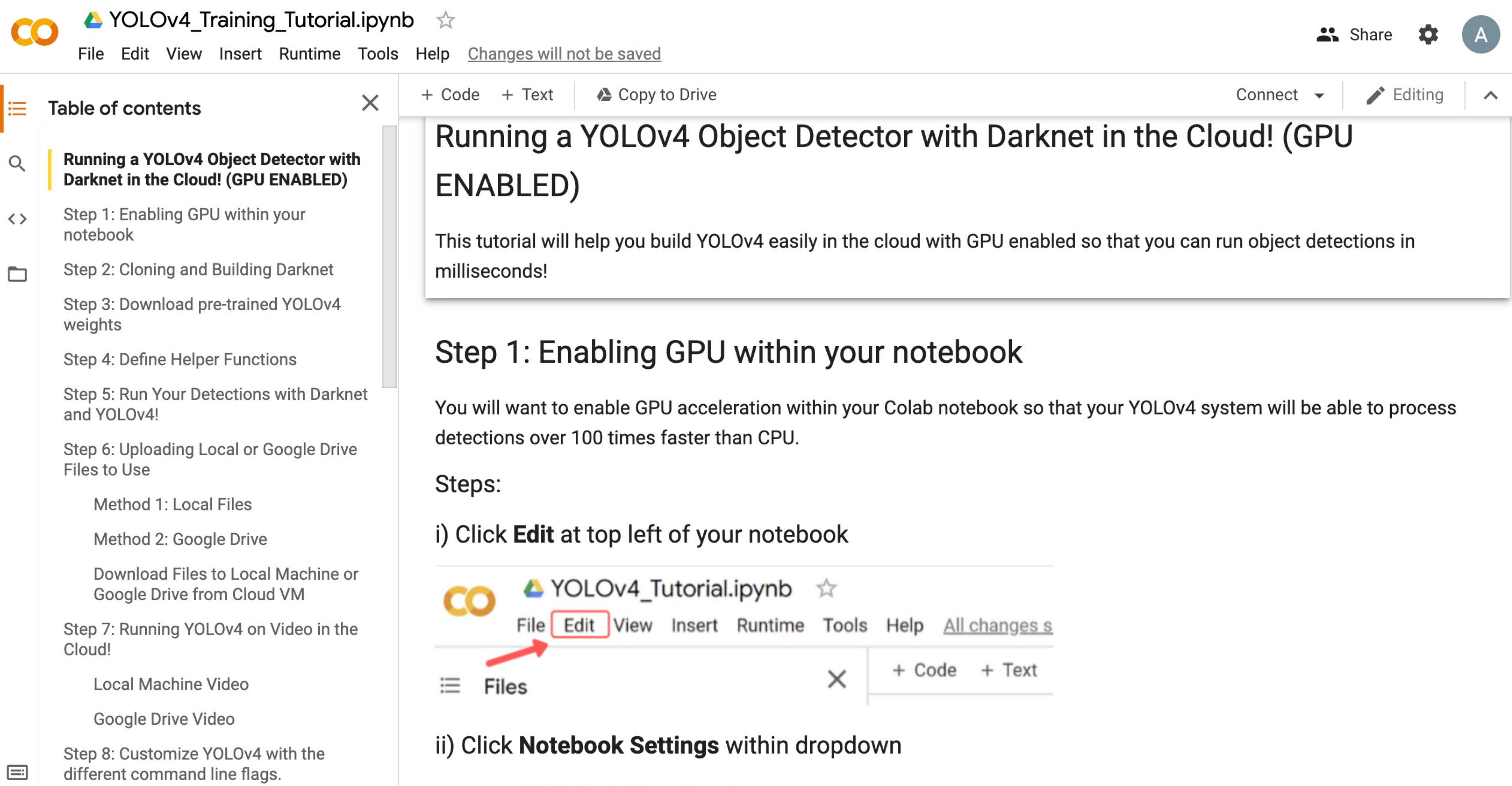
Source: https://github.com/ultralytics/yolov5

YOLOv4 Object Detector in Google Colab



Source: theAIGuysCode, YOLOv4 Cloud Tutorial, https://github.com/theAIGuysCode/YOLOv4-Cloud-Tutorial

Train Custom YOLOv4 Model in Google Colab



Source: theAIGuysCode, YOLOv4 Cloud Tutorial, https://github.com/theAIGuysCode/YOLOv4-Cloud-Tutorial

YOLOv5 Tutorial

Source: https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb

TensorFlow 2.0 MNIST

```
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

https://www.tensorflow.org/overview/

TensorFlow

TensorFlow Image Classification

TensorFlow Install Learn API Resources More Search Language GitHub Sign in

TensorFlow > Learn > TensorFlow Core > Tutorials

TensorFlow tutorials
Quickstart for beginners
Quickstart for experts

BEGINNER

ML basics with Keras
- Basic image classification
- Text classification with TF Hub
- Text classification with preprocessed text
- Regression
- Overfit and underfit
- Save and load

Load and preprocess data
Estimator

ADVANCED

Customization
Distributed training
Images
Text

Basic classification: Classify images of clothing

Run in Google Colab | View source on GitHub | Download notebook

This guide trains a neural network model to classify images of clothing, like sneakers and shirts. It's okay if you don't understand all the details; this is a fast-paced overview of a complete TensorFlow program with the details explained as you go.

This guide uses tf.keras, a high-level API to build and train models in TensorFlow.

```
from __future__ import absolute_import, division, print_function, unicode_literals

# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt

print(tf.__version__)
```

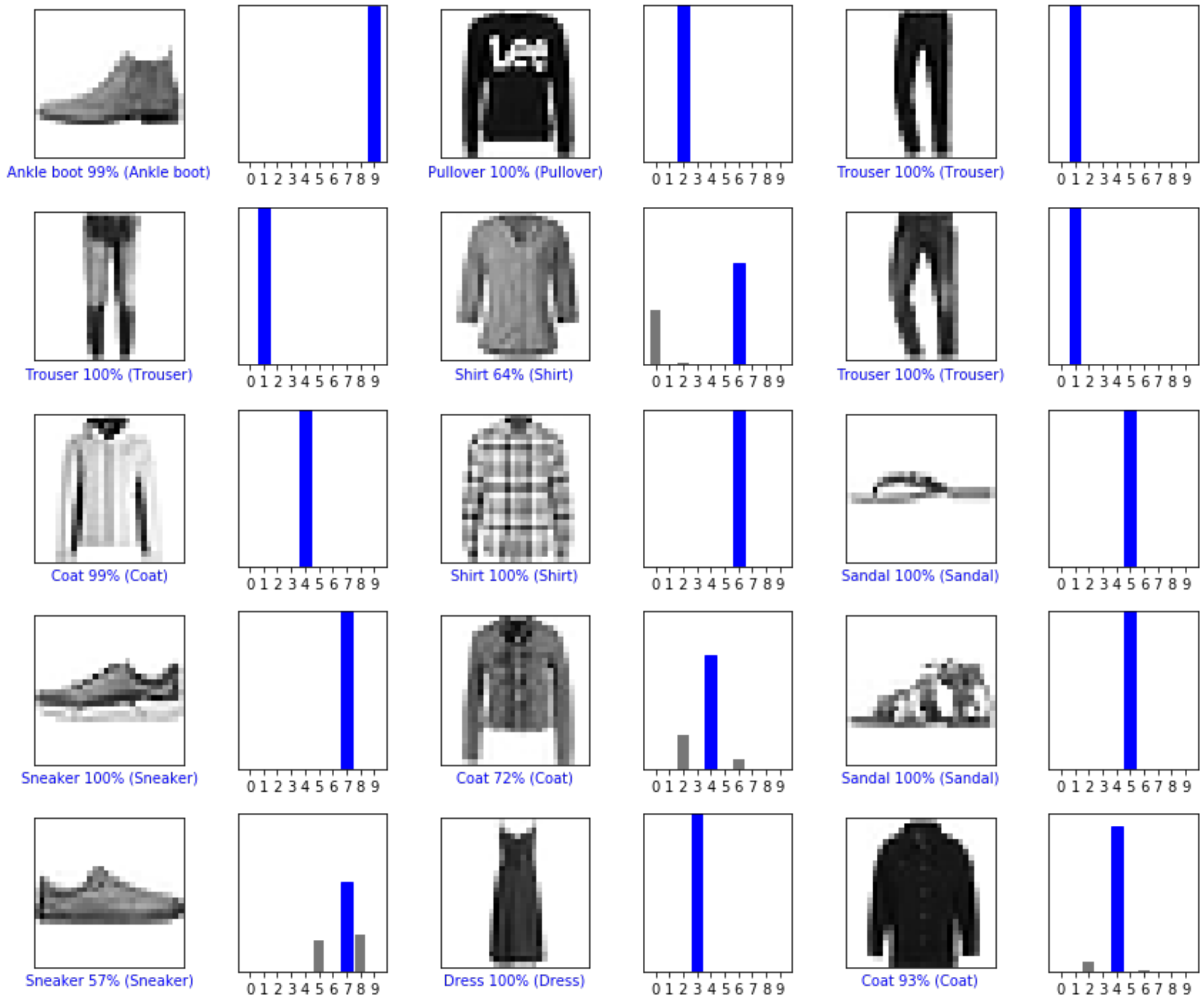
```
2.0.0
```

Contents
- Import the Fashion MNIST dataset
- Explore the data
- Preprocess the data
- Build the model
  - Set up the layers
  - Compile the model
- Train the model
- Evaluate accuracy
- Make predictions

https://www.tensorflow.org/tutorials/keras/classification

TensorFlow

Image Classification
Fashion MNIST dataset



https://www.tensorflow.org/tutorials/keras/classification

Basic Classification
Fashion MNIST Image Classification

https://colab.research.google.com/drive/19PJOJi1vn1kjcutIzNHjRSLbeVl4kd5z

tf01_basic_classification.ipynb

File Edit View Insert Runtime Tools Help

COMMENT SHARE

CODE TEXT CELL CELL CONNECT EDITING

Table of contents | Code snippets | Files

SECTION

Copyright 2018 The TensorFlow Authors.

↳ *2 cells hidden*

Train your first neural network: basic classification

View on TensorFlow.org | Run in Google Colab | View source on GitHub

This guide trains a neural network model to classify images of clothing, like sneakers and shirts. It's okay if you don't understand all the details, this is a fast-paced overview of a complete TensorFlow program with the details explained as we go.

This guide uses tf.keras, a high-level API to build and train models in TensorFlow.

```
# memory footprint support libraries/code
!ln -sf /opt/bin/nvidia-smi /usr/bin/nvidia-smi
!pip install gputil
!pip install psutil
!pip install humanize
import psutil
import humanize
import os
import GPUtil as GPU
GPUs = GPU.getGPUs()
gpu = GPUs[0]
def printm():
    process = psutil.Process(os.getpid())
    print("Gen RAM Free: " + humanize.naturalsize( psutil.virtual_memory().available ), " | Pro
    print("GPU RAM Free: {0:.0f}MB | Used: {1:.0f}MB | Util {2:3.0f}% | Total {3:.0f}MB".format
printm()
```

Source: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/keras/basic_classification.ipynb

Papers with Code
State-of-the-Art (SOTA)

Search for papers, code and tasks | Browse State-of-the-Art | Follow | Discuss | Trends | About | Log In/Register

Browse State-of-the-Art

1509 leaderboards • 1327 tasks • 1347 datasets • 17810 papers with code

Follow on Twitter for updates

Computer Vision

Semantic Segmentation
33 leaderboards
667 papers with code

Image Classification
52 leaderboards
564 papers with code

Object Detection
54 leaderboards
467 papers with code

Image Generation
51 leaderboards
231 papers with code

Pose Estimation
40 leaderboards
231 papers with code

See all 707 tasks

Natural Language Processing

Machine Translation

Language Modelling

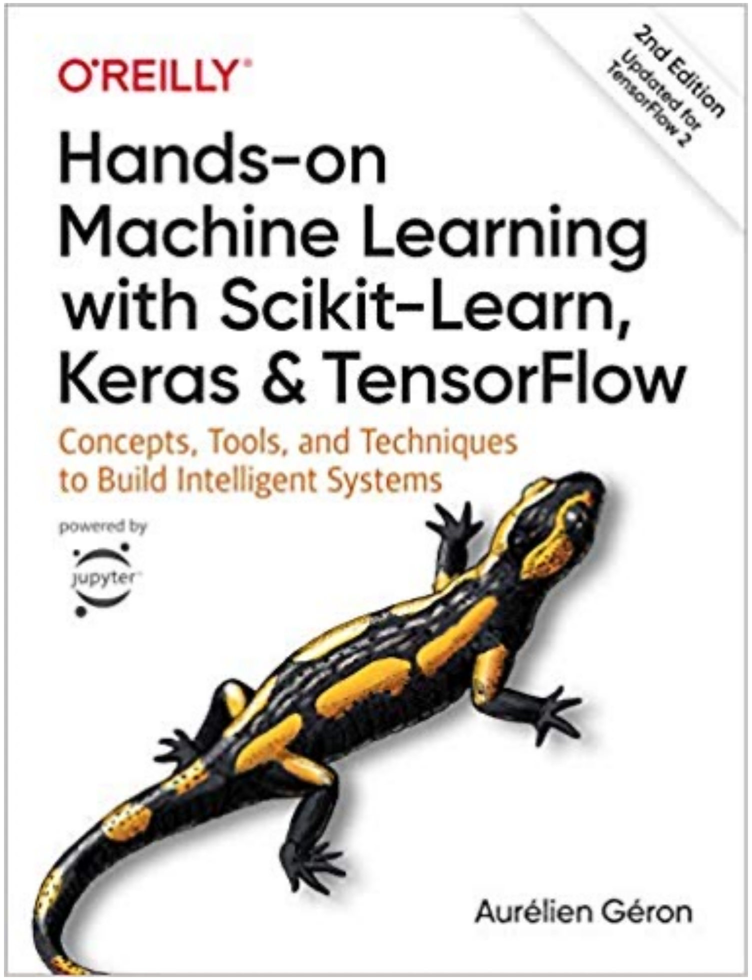
Question Answering

Sentiment Analysis

Text Generation

https://paperswithcode.com/sota

Aurélien Géron (2019),
Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd Edition
O'Reilly Media, 2019



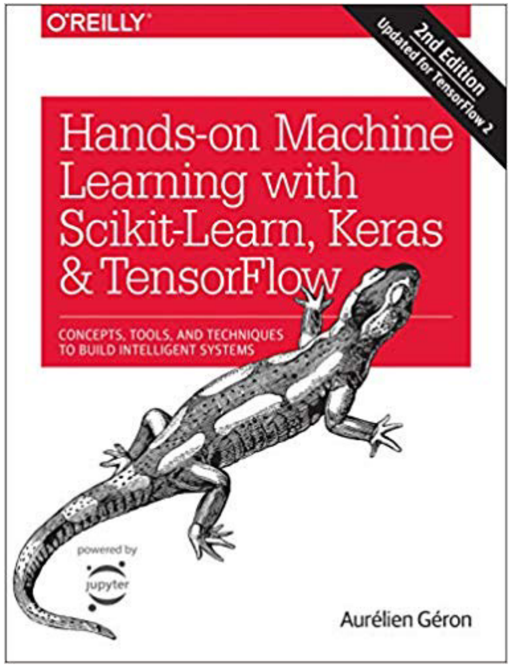
https://github.com/ageron/handson-ml2

Source: https://www.amazon.com/Hands-Machine-Learning-Scikit-Learn-TensorFlow/dp/1492032646/

Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow

Notebooks

1. The Machine Learning landscape
2. End-to-end Machine Learning project
3. Classification
4. Training Models
5. Support Vector Machines
6. Decision Trees
7. Ensemble Learning and Random Forests
8. Dimensionality Reduction
9. Unsupervised Learning Techniques
10. Artificial Neural Nets with Keras
11. Training Deep Neural Networks
12. Custom Models and Training with TensorFlow
13. Loading and Preprocessing Data
14. Deep Computer Vision Using Convolutional Neural Networks
15. Processing Sequences Using RNNs and CNNs
16. Natural Language Processing with RNNs and Attention
17. Representation Learning Using Autoencoders
18. Reinforcement Learning
19. Training and Deploying TensorFlow Models at Scale



https://github.com/ageron/handson-ml2

Python in Google Colab (Python101)

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT

python101.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share A

+ Code + Text RAM Disk Editing

Table of contents

Deep Learning

Image Classification

- Source: https://www.tensorflow.org/overview/

```python
import tensorflow as tf
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(128, activation='relu'),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test)
```

```
Epoch 1/5
1875/1875 [==============================] - 4s 2ms/step - loss: 0.4790 - accuracy: 0.8606
```

https://tinyurl.com/aintpupython101

Summary

- Convolutional Neural Networks (CNN)
  - Convolution
  - Pooling
  - Fully Connection (FC) (Flattening)
- Computer Vision
  - Image Classification
  - Object Detection

References

- Aurélien Géron (2019), Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems, 2nd Edition, O'Reilly Media. https://github.com/wesm/pydata-book
- Stuart Russell and Peter Norvig (2020), Artificial Intelligence: A Modern Approach, 4th Edition, Pearson.
- Luis Serrano (2017), A friendly introduction to Convolutional Neural Networks and Image Recognition, https://www.youtube.com/watch?v=2-Ol7ZB0MmU
- Arden Dertat (2017), Applied Deep Learning - Part 4: Convolutional Neural Networks, https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2
- CS231n: Convolutional Neural Networks for Visual Recognition, https://cs231n.github.io/convolutional-networks/
- Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao (2020). "Scaled-YOLOv4: Scaling Cross Stage Partial Network." arXiv preprint arXiv:2011.08036 (2020).
- Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao (2020). "Yolov4: Optimal speed and accuracy of object detection." arXiv preprint arXiv:2004.10934 (2020).
- Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi (2016). "You only look once: Unified, real-time object detection." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788. 2016.
- Mingxing Tan, Ruoming Pang, and Quoc V. Le (2020). "EfficientDet: Scalable and efficient object detection." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 10781-10790. 2020.
- theAIGuysCode, YOLOv4 Cloud Tutorial, https://github.com/theAIGuysCode/YOLOv4-Cloud-Tutorial
- YOLOv5, https://github.com/ultralytics/yolov5
- Min-Yuh Day (2021), Python 101, https://tinyurl.com/aintpupython101