



กรมที่ดิน  
Department of Lands



Chula  
Chulalongkorn University

MARS

# Basic AI for Computer Vision

Teerapong Panboonyuen, Ph.D.

[teerapong.pa@chula.ac.th](mailto:teerapong.pa@chula.ac.th)

# Outlines

- Introduction to Basic AI
- Introduction to Computer Vision
- Code and Demo: Image Classification
- Code and Demo: Object Detection
- Code and Demo: Instance Segmentation
- Demo: AI-Based Detection of Markers in Land Title Deeds

# Reference:

1. <https://pytorch.org/tutorials>
2. <https://stanford.edu/~shervine/teaching/cs-229/>
3. <http://introtodeeplearning.com/>
4. <https://www.simplilearn.com/tutorials/deep-learning-tutorial/introduction-to-deep-learning>
5. <https://www.geeksforgeeks.org/introduction-deep-learning/>
6. <https://databasetown.com/supervised-learning-algorithms/>
7. <https://databasetown.com/unsupervised-learning-types-applications/>
8. <https://realpython.com/learning-paths/pandas-data-science/>
9. <https://www.turing.ac.uk/>

# About Me



Kao  
Panboonyuen

kao-panboonyuen   
AI Research Scientist

**Name:** Teerapong Panboonyuen (Kao)

**Contact:** [teerapong.pa@chula.ac.th](mailto:teerapong.pa@chula.ac.th)

[panboonyuen.kao@gmail.com](mailto:panboonyuen.kao@gmail.com)

**Education:** Ph.D. (AI) Chula

**Position:** AI Team Lead, MARS

PostDoc, Chula

**Interests:** Computer Vision, Deep Learning

Machine Learning, Remote Sensing

# Artificial Intelligence Projects

Mango 🍌 Leaf 🍃 Disease Dataset

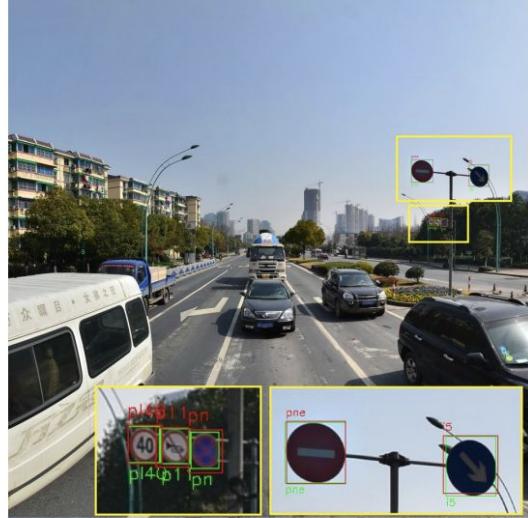
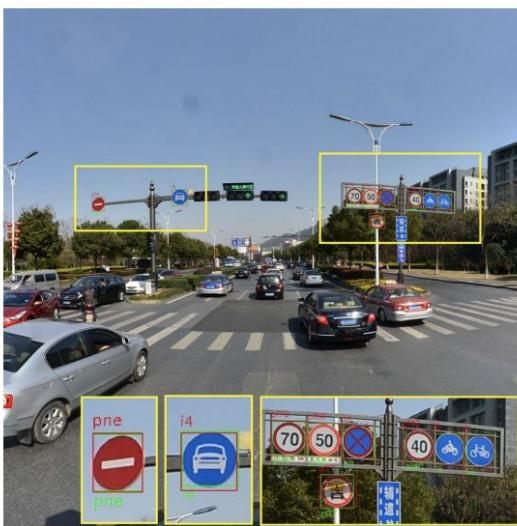


กรมที่ดิน  
Department of Lands

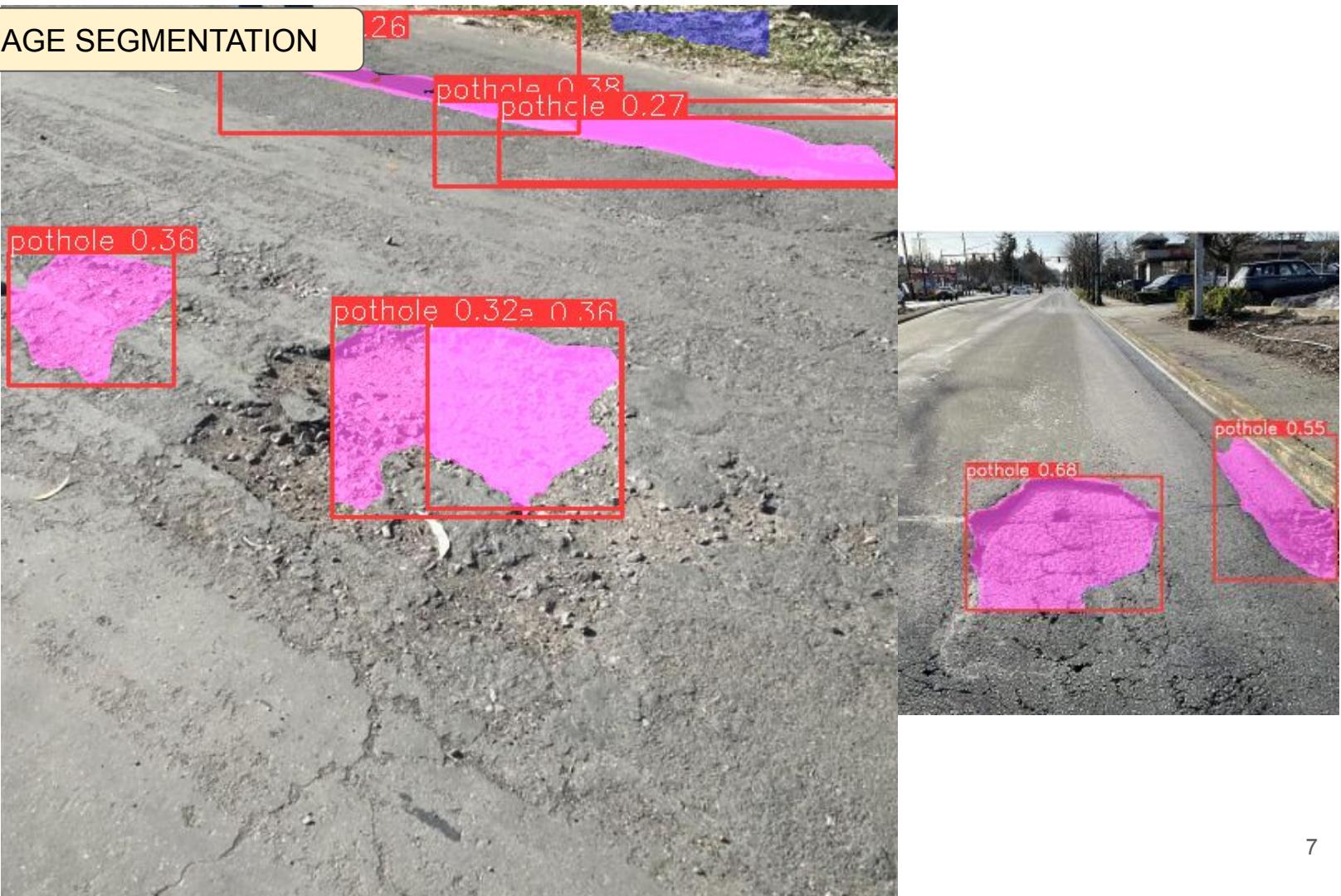


TensorFlow

## LAB 2: AI OBJECT DETECTION



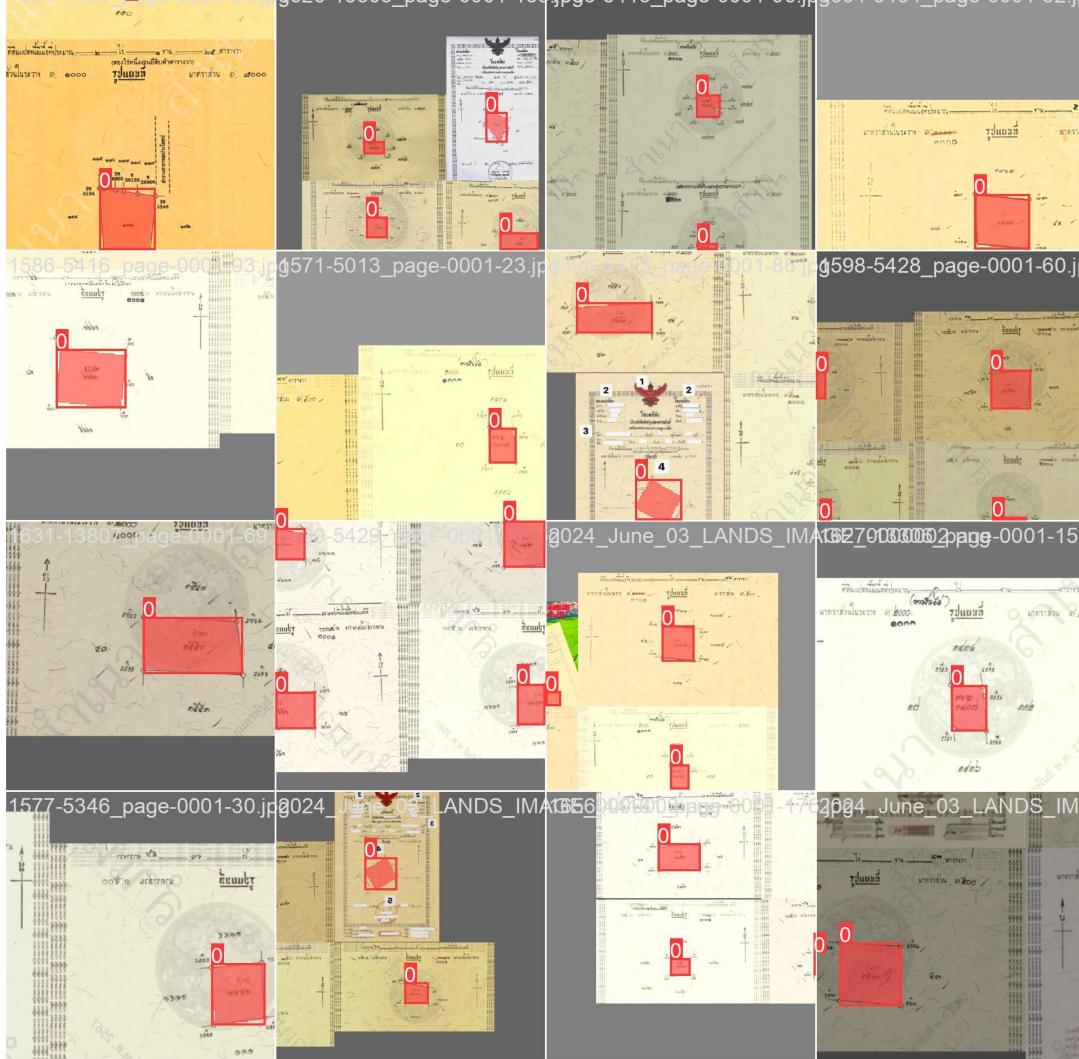
## LAB 3: AI IMAGE SEGMENTATION



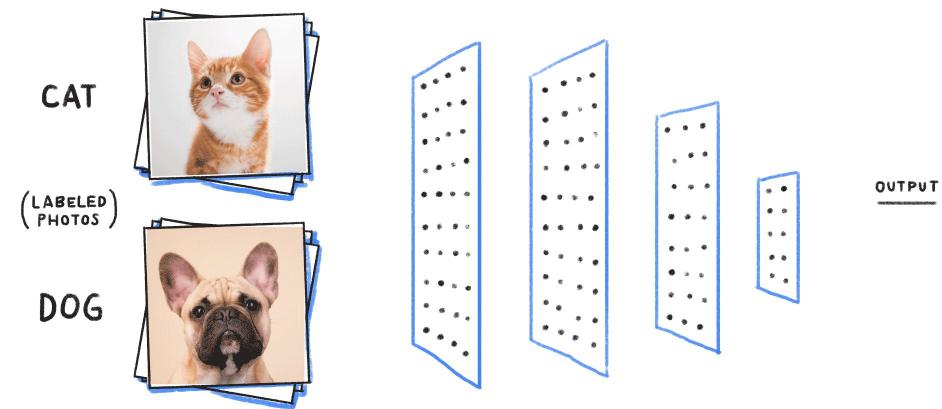
# DEMO



# DEMO



# Basic AI

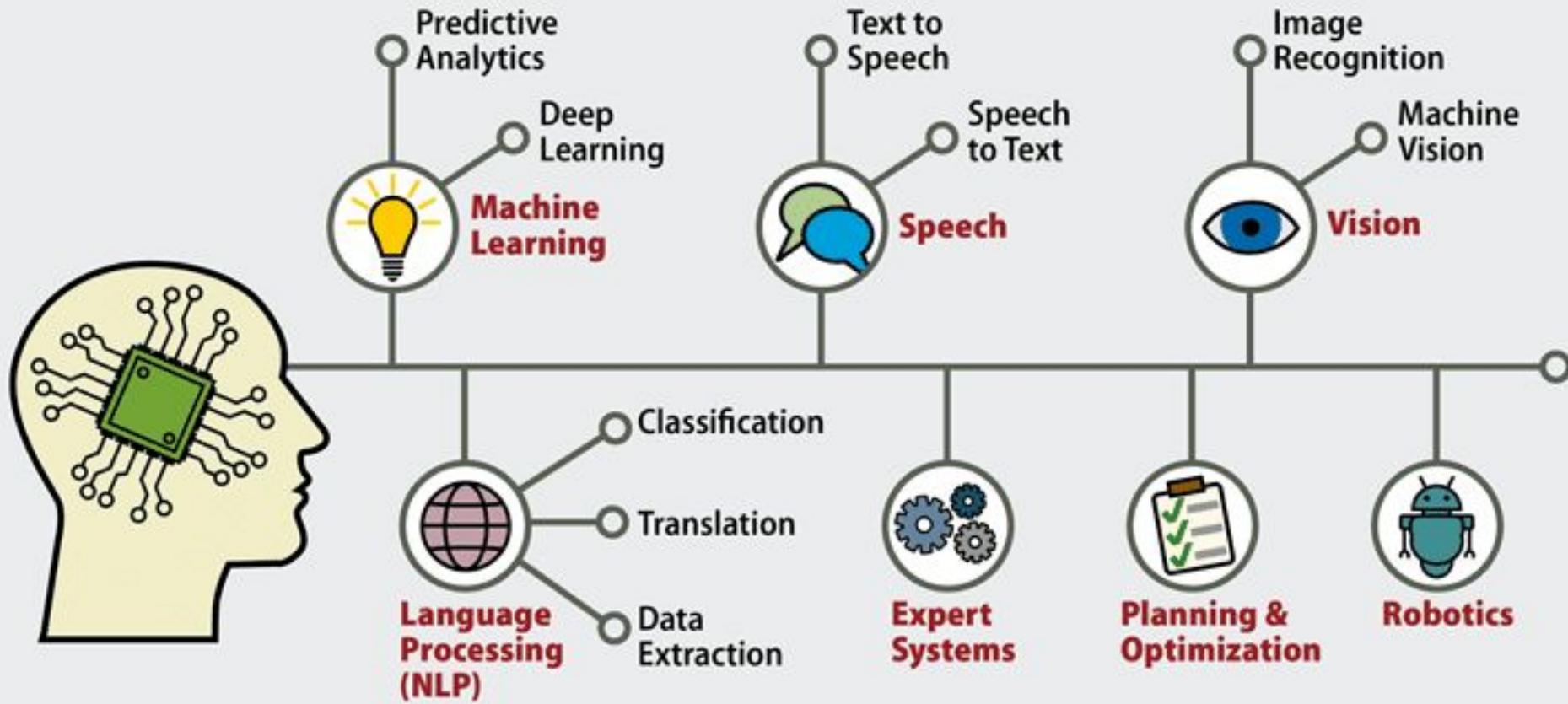




**Artificial Intelligence** (AI) is  
the ability for a computer to  
think and learn.

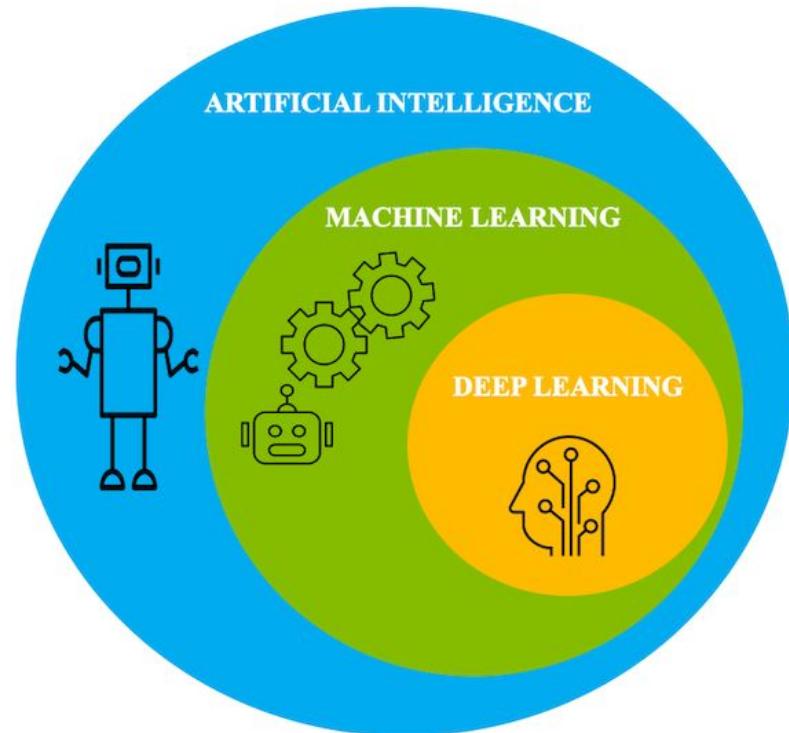


# Artificial Intelligence



# AI vs Machine Learning vs Deep Learning

- **Artificial Intelligence (AI)** is the concept of creating smart intelligent machines.
- **Machine Learning (ML)** is a subset of artificial intelligence that helps you build AI-driven applications.
- **Deep Learning** is a subset of machine learning that uses vast volumes of data and complex algorithms to train a model.



# Machine Learning

- Goal is to learn a mapping from **inputs** to **outputs**
- Simplest technique is **supervised learning**
- Uses **training data**
- How to do the training?
  - **neural nets** and **deep learning** a popular current approach



alan turing



alan turing

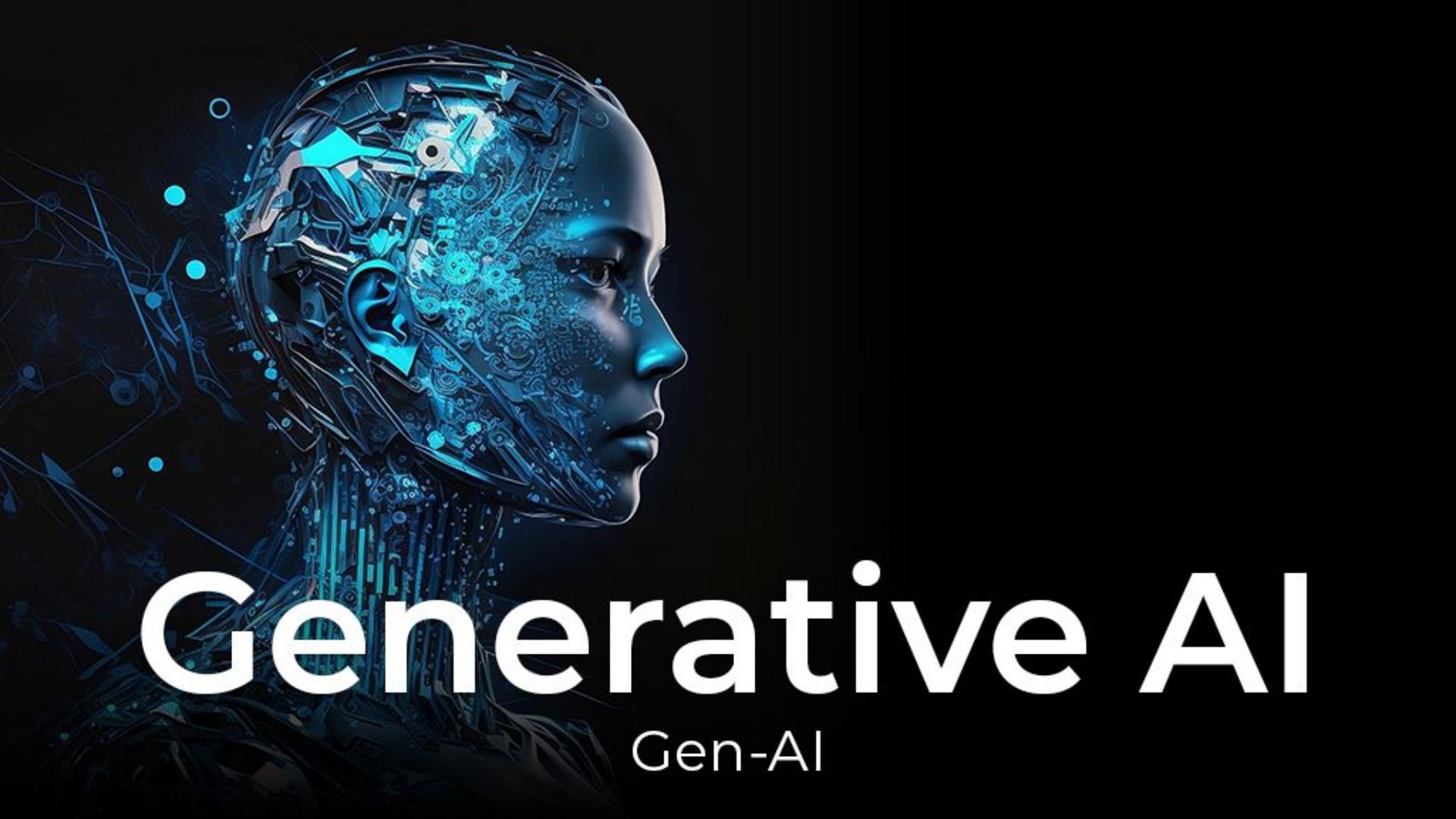


alan turing



?

training  
data



# Generative AI

Gen-AI



<https://openai.com/index/hello-gpt-4o/>

May 13, 2024

# Hello GPT-4o

We're announcing GPT-4o, our new flagship model that can reason across audio, vision, and text in real time.

[Contributions >](#) [Try on ChatGPT ↗](#) [Try in Playground ↗](#) [Rewatch live demos >](#)

All videos on this page are at 1x real time.

# WHAT IS GENERATIVE AI?

VISUALLY EXPLAINED... BY GENERATIVE AI

ChatGPT, DALL-E, MidJourney, DeepMind—generative AI technologies have exploded into mainstream consciousness. With access to these technologies increasing day-by-day, we asked AI to help demonstrate the power and influence of this new tech trend.

This robot—and all the visuals in this infographic, including icons—were generated using AI software like MidJourney and DALL-E.

 The text captions were also generated using AI (ChatGPT).

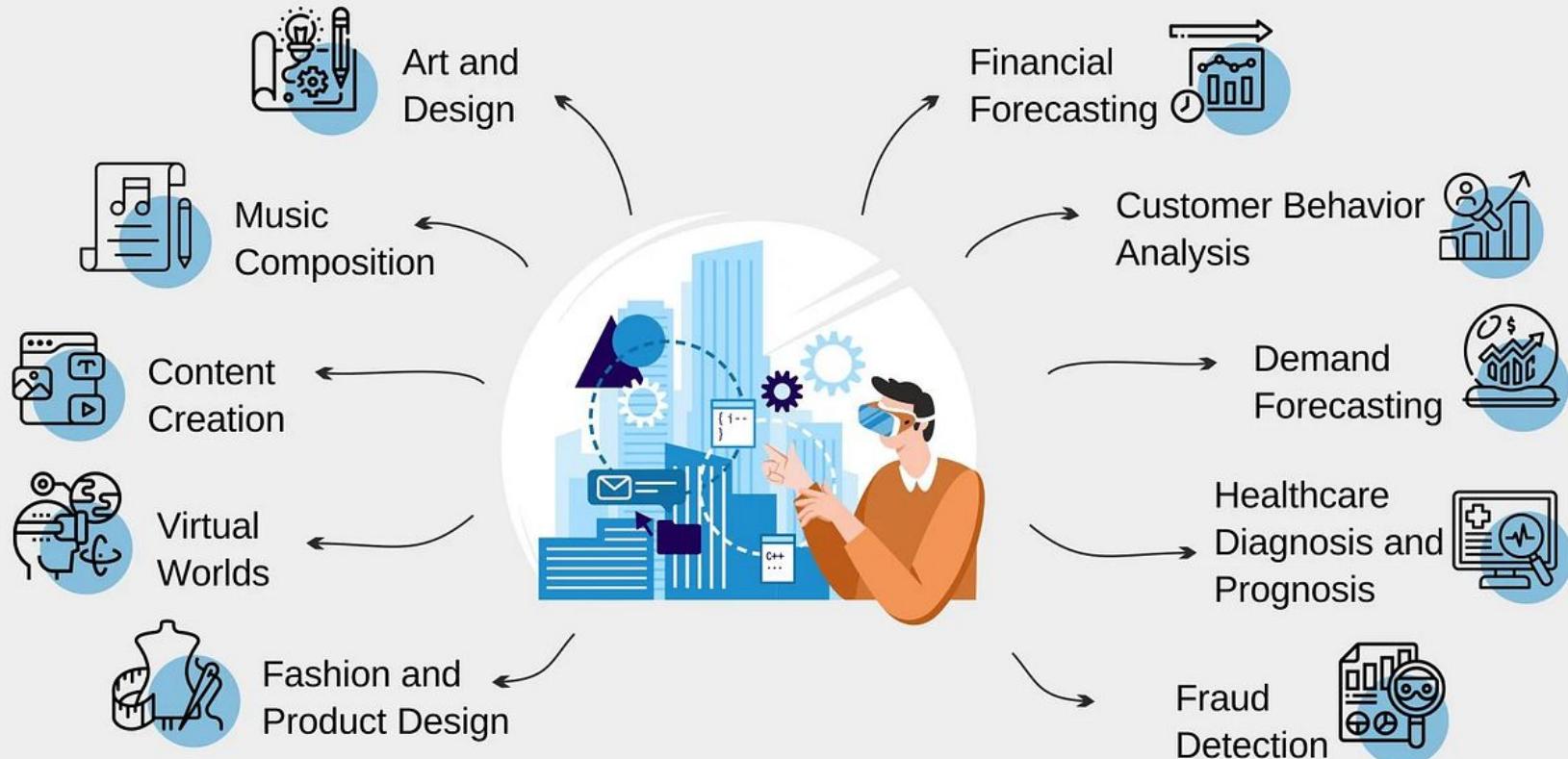
This image was created on MidJourney using the following text prompt:  
a robot head, portrait, no background, artificial intelligence, minimalist and white aesthetic, futuristic, digital, realistic, 4K



# Artificial Intelligence vs. Traditional Machine Learning, Generative AI

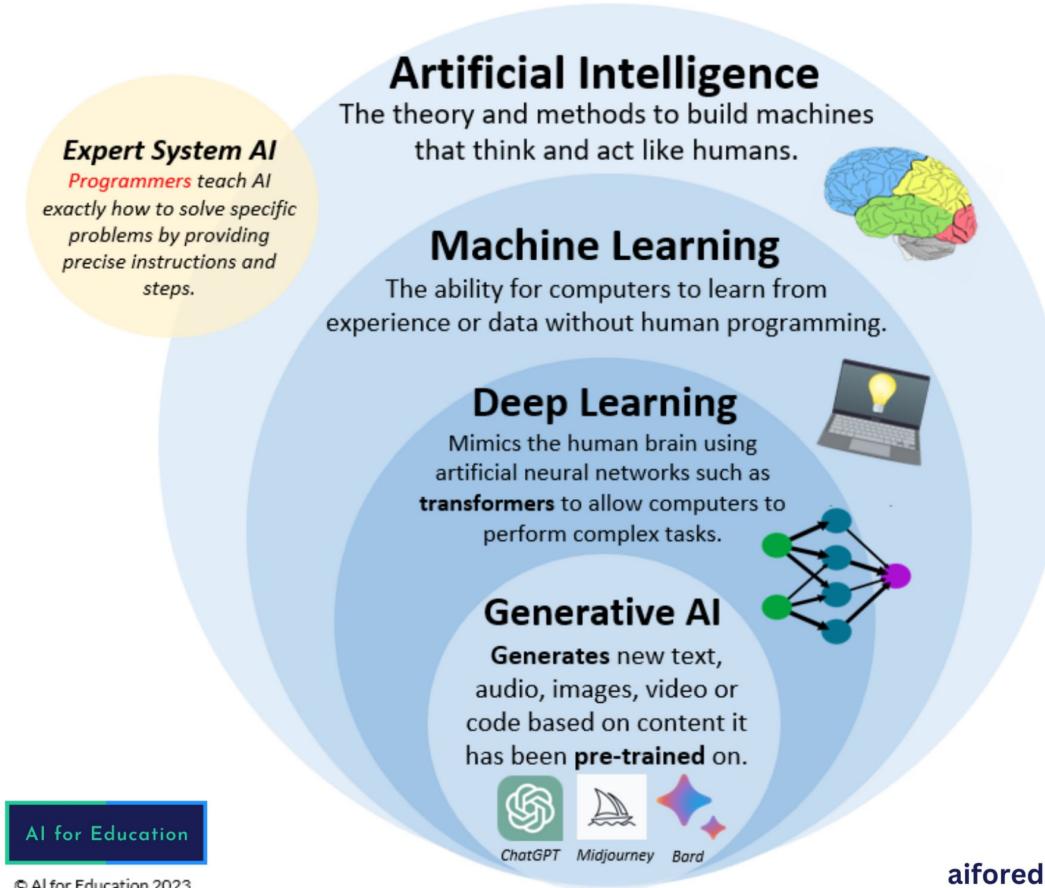
Characteristic	AI	Traditional ML	Generative AI
Purpose	Develop computer systems that can perform tasks that typically require human intelligence.	Make predictions or decisions based on given data.	Generate new data samples that resemble a given set of training data.
Data Interaction	Models use various techniques and strategies designed to mimic human intelligence across a wide range of applications.	Models learn from data to make predictions or decisions on new unseen data.	Models produce new data that weren't part of the original dataset but share similar characteristics.

# Generative AI Applications



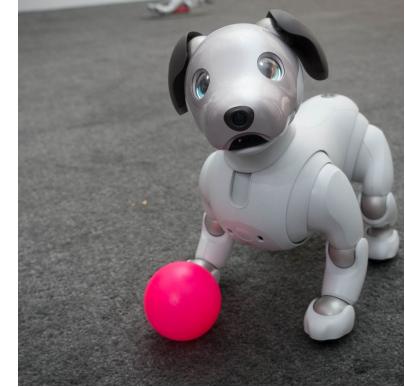
# Defining Generative AI

To understand generative artificial intelligence (GenAI), we first need to understand how the technology builds from each of the AI subcategories listed below.



# Applications of Artificial Intelligence

- Machine Translation such as Google Translate
- Self Driving Vehicles such as Google's Waymo
- AI Robots such as Sophia and Aibo
- Speech Recognition applications like Apple's Siri or OK Google



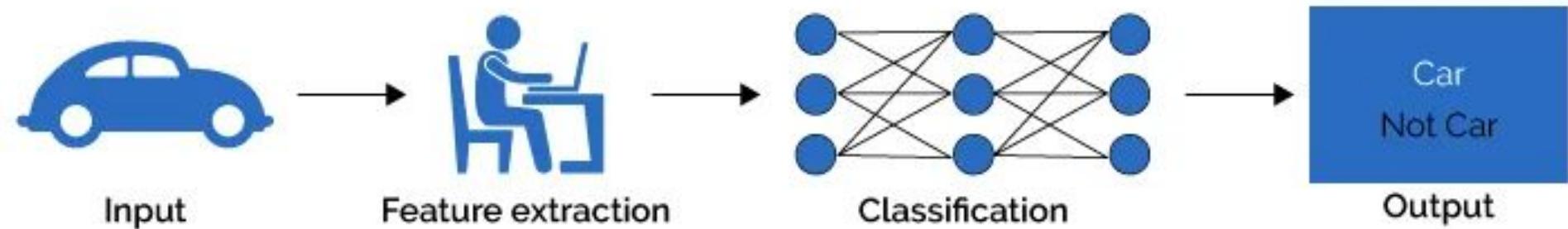
<https://www.youtube.com/watch?v=kJoAcEl2PXQ&t=14s>



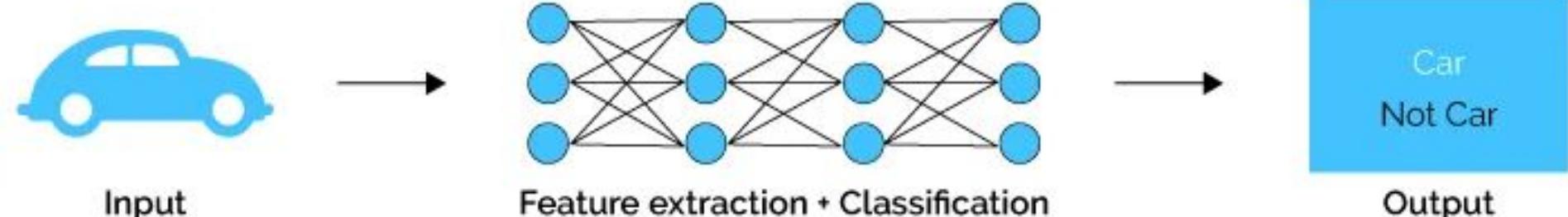
Sophia is a humanoid robot with unique features developed by Hanson Robotics Limited



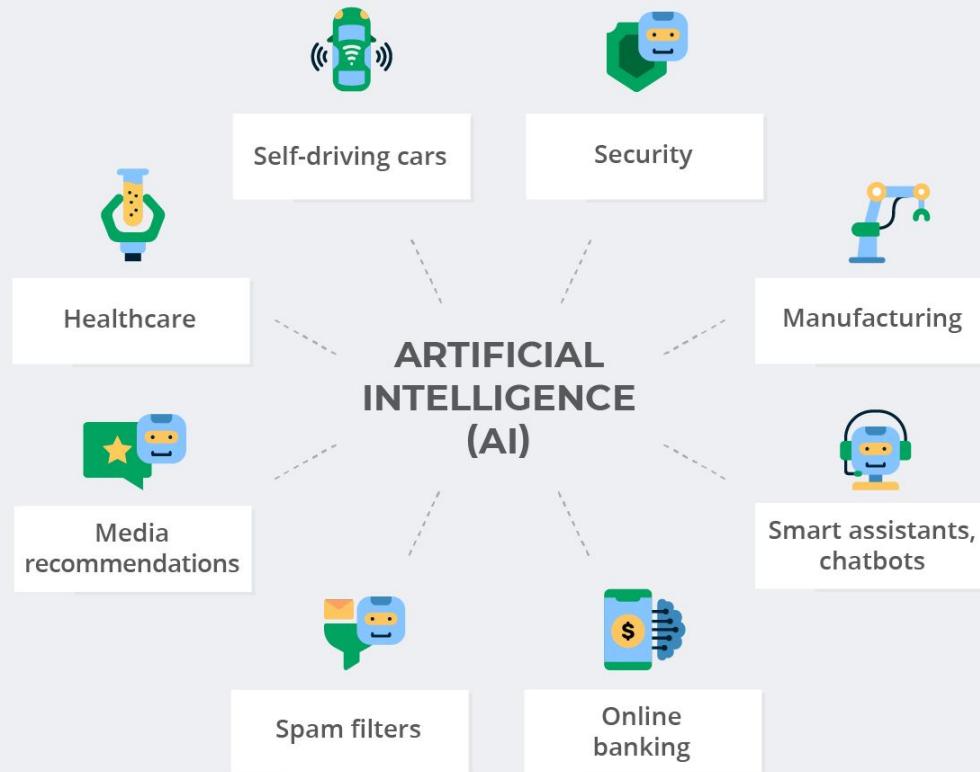
# Machine Learning



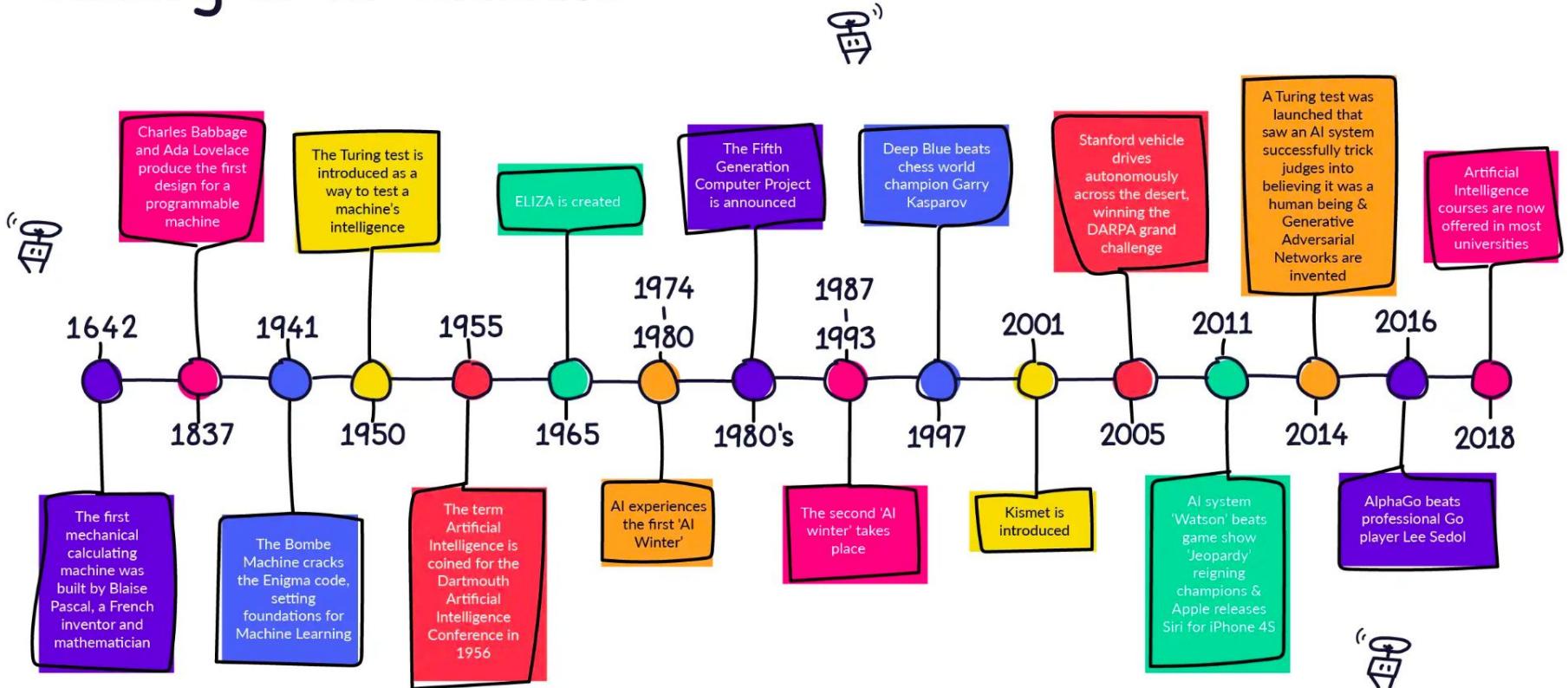
# Deep Learning



## Examples of Artificial Intelligence in Everyday Life



# History of AI Timeline



# 6 ARTIFICIAL INTELLIGENCE JOB TITLES



## Machine Learning Engineers

focus on research, design, and implementation of models to help machines make choices.



## Data Scientists

create predictive models to solve real-world problems, obtaining insights from data and presenting them.



## Data Engineers

cleanse the data, massage the date, pipeline the data, etc. ensuring it is formatted correctly for the intended usage.



## Software Engineers

typically create software, but may have a focus on data and AI – look for keywords like "Python" and "Artificial Intelligence" in the job posting.



## Product Managers

identify what users find valuable in a product. if the product is using AI/machine learning techniques - they would have to understand these concepts.



## Architects

have a deep understanding of the entire machine learning ecosystem – this is a long-term career goal!



Big Data Engineer



AI Data Analyst



Business Intelligence  
Developer



AI Software Engineer



AI Consultant



NLP Processing  
Specialist



AI Research Scientist



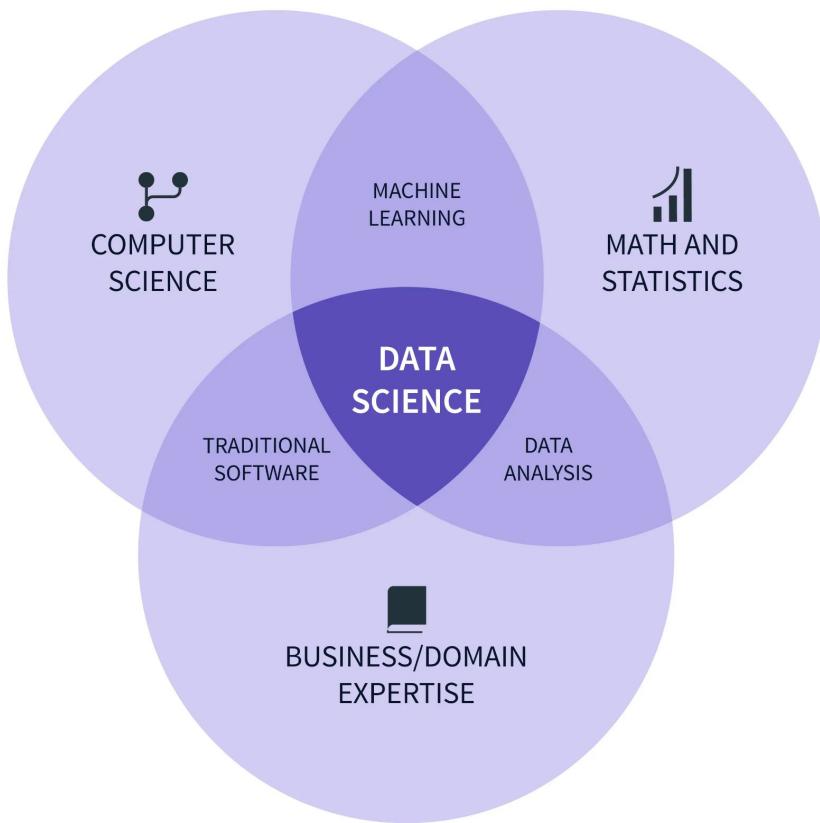
Machine Learning  
Engineer



User Experience  
Specialist

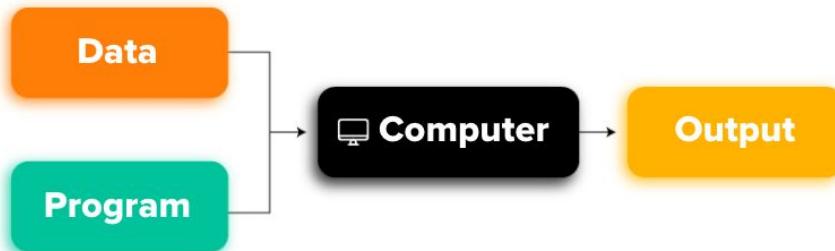


AI Wrangler

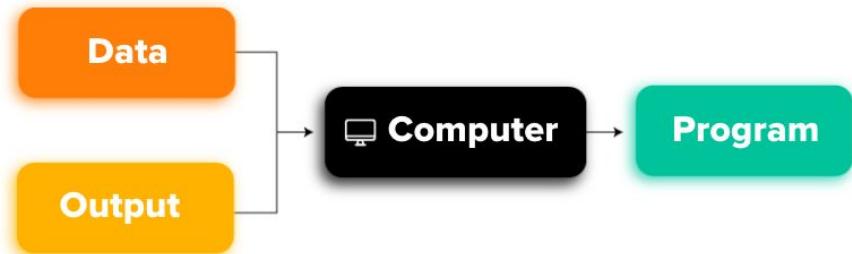




# TRADITIONAL PROGRAMMING



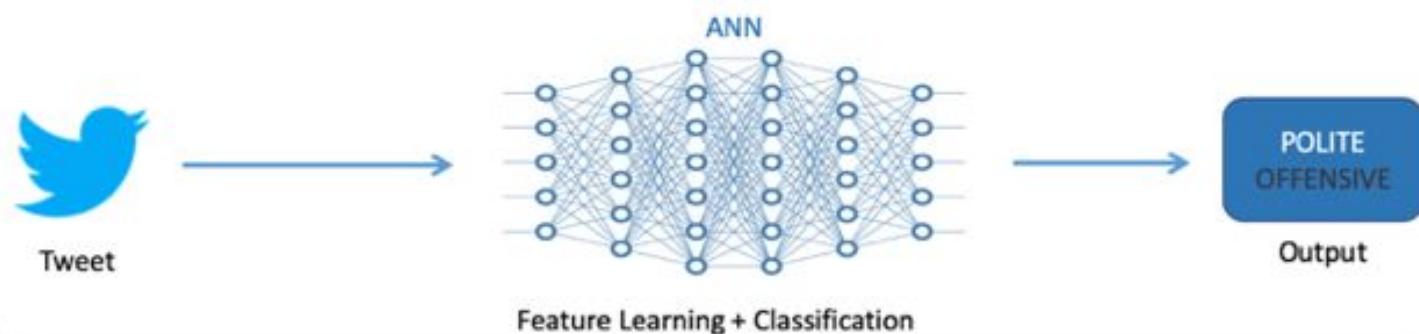
# MACHINE LEARNING

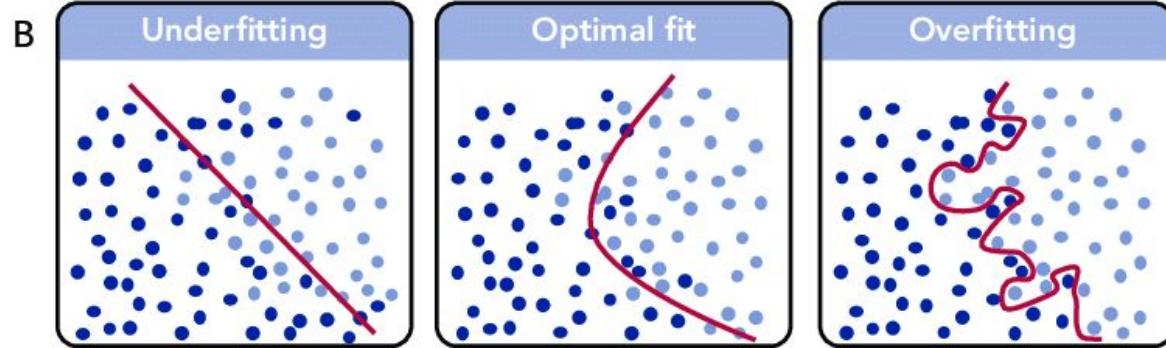
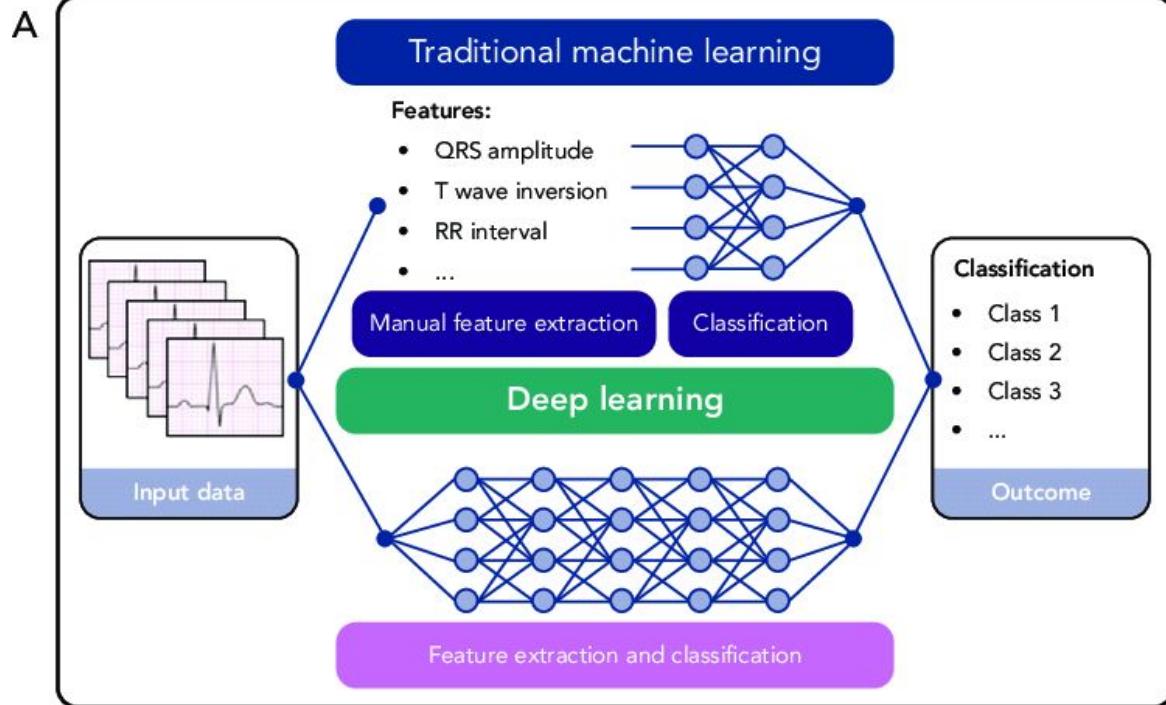


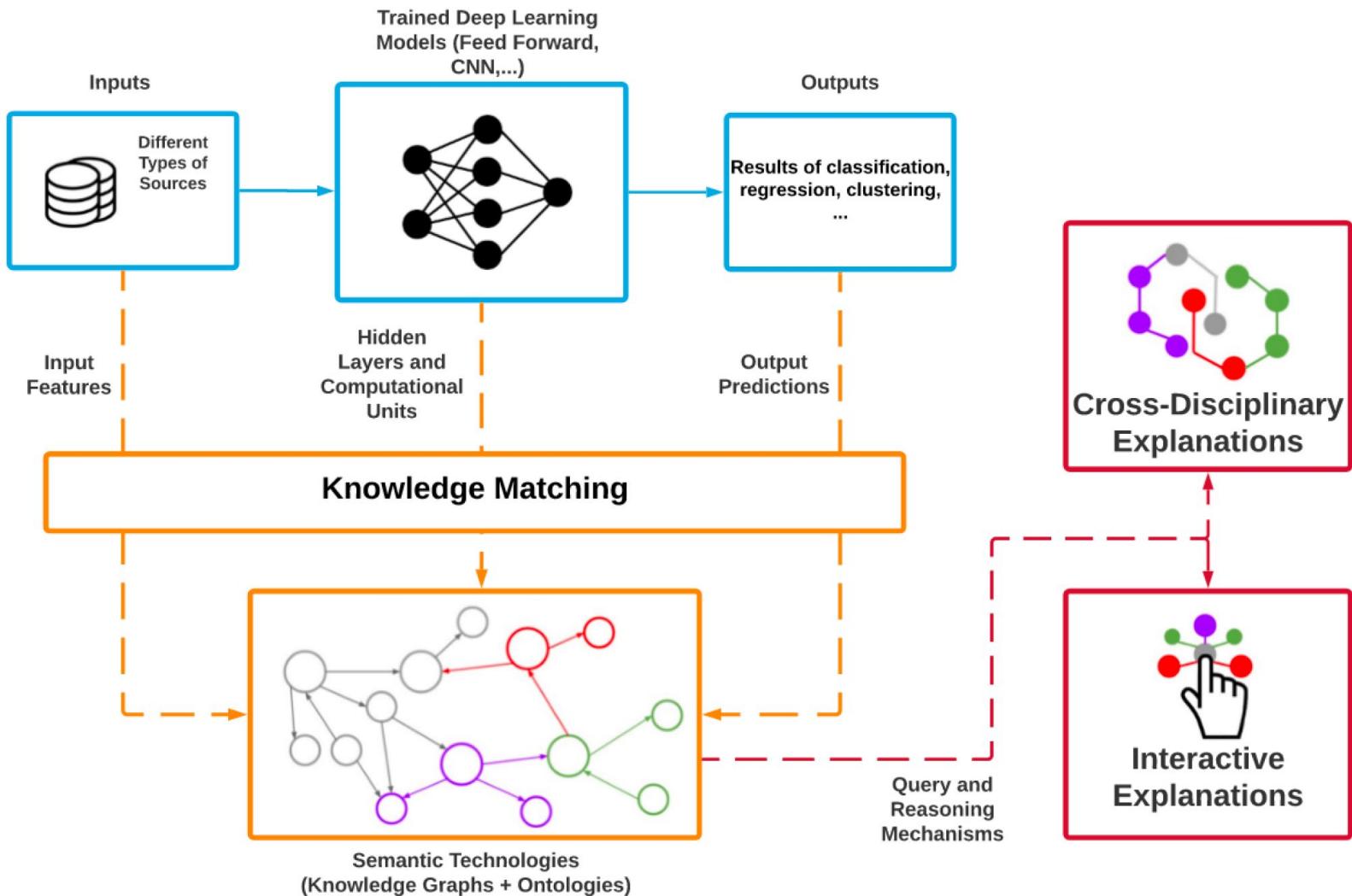
## TRADITIONAL MACHINE LEARNING

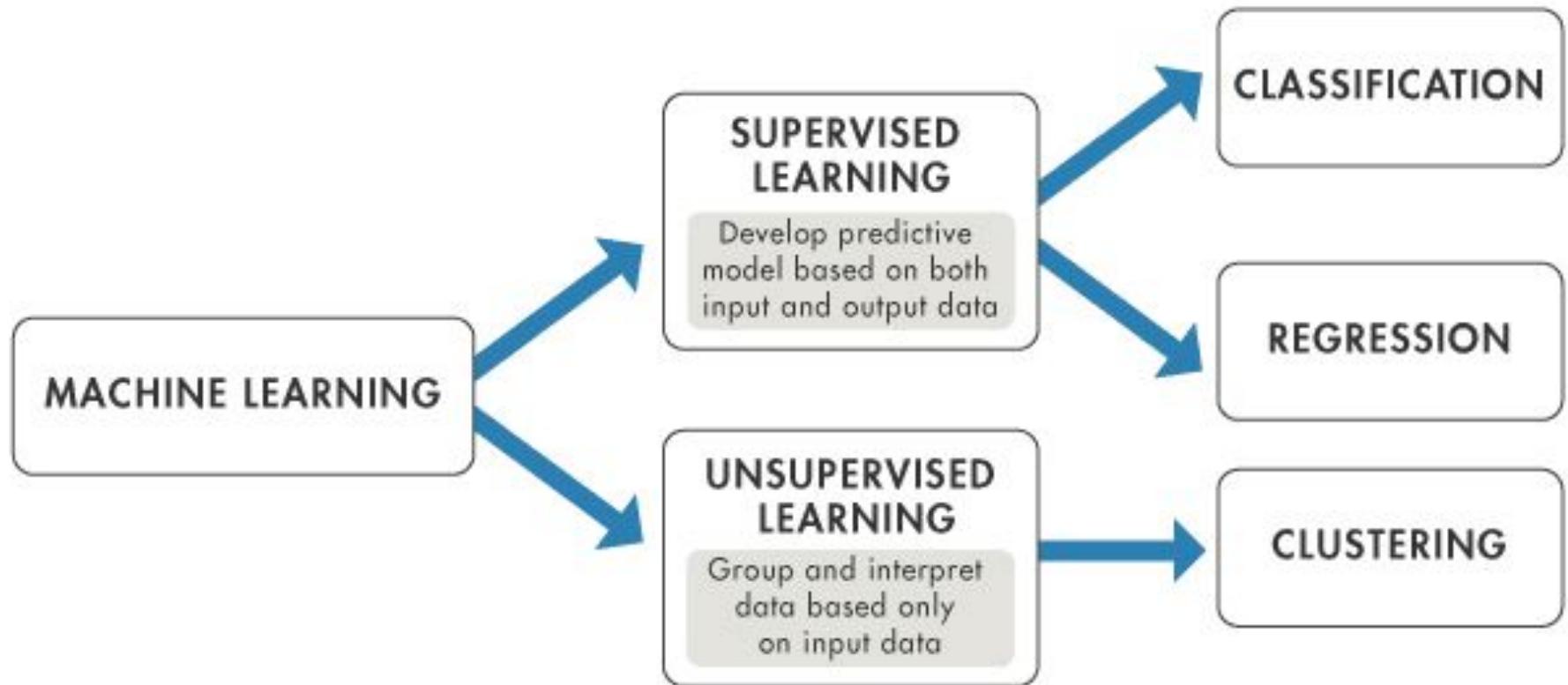


## DEEP LEARNING





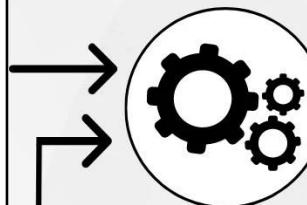
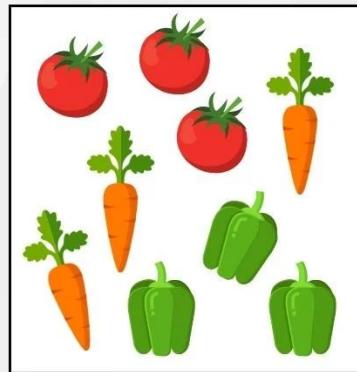




# SUPERVISED LEARNING

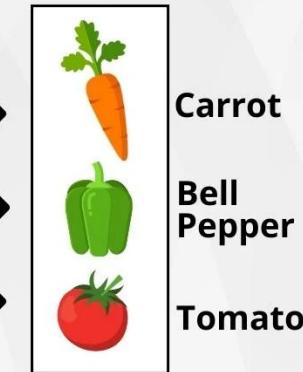
Supervised machine learning is a branch of artificial intelligence that focuses on training models to make predictions or decisions based on labeled training data.

## Labeled Data

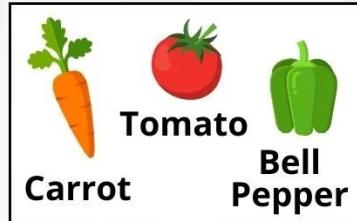


Model Training

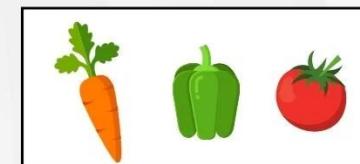
Prediction



## Labels



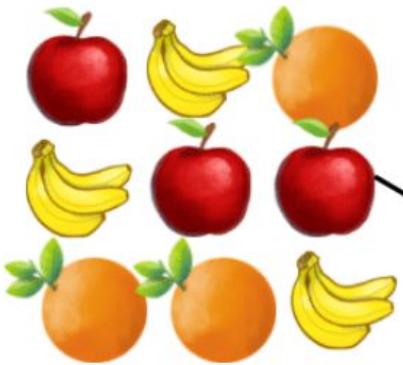
DatabaseTown



Test Data



Input Data



Machine Learning  
Model



Prediction

It's an Apple

Partial Labels

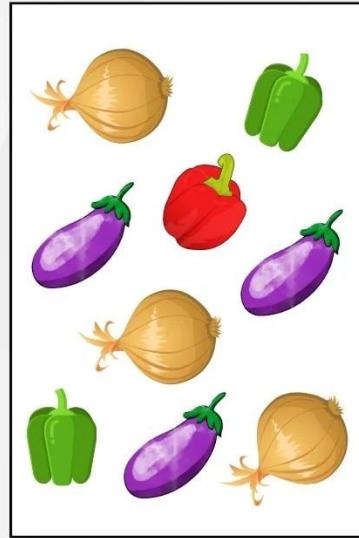


Unlabelled Data

# UNSUPERVISED LEARNING

Unsupervised learning is a type of machine learning where the algorithm learns from unlabeled data without any predefined outputs or target variables.

**Input Raw Data**

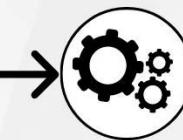


**Unlabeled Data**



**DatabaseTown**

**Interpretation**

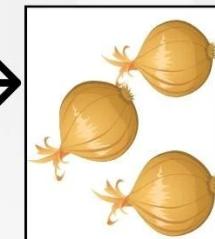
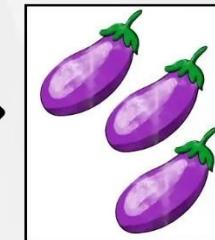
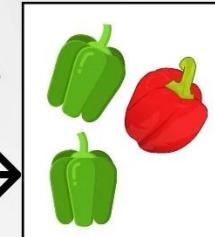


**Algorithms**

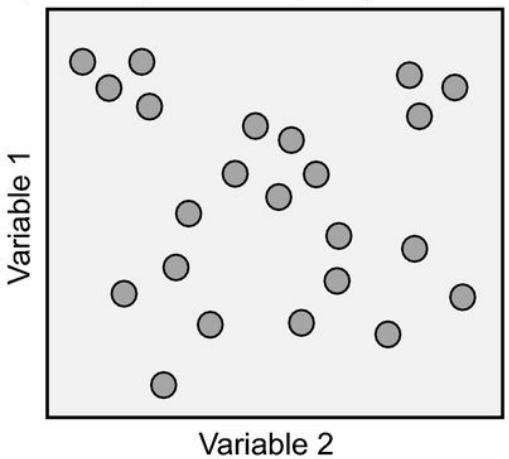


**Processing**

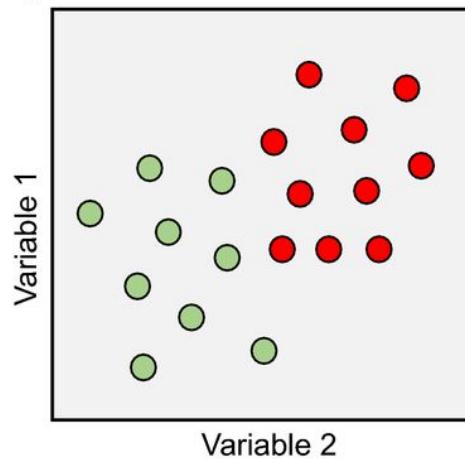
**Outputs**



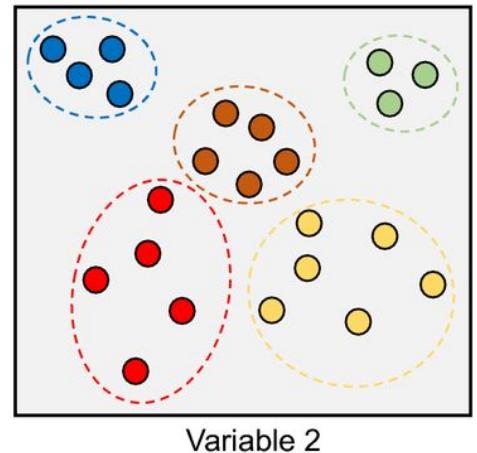
a) Unsupervised learning



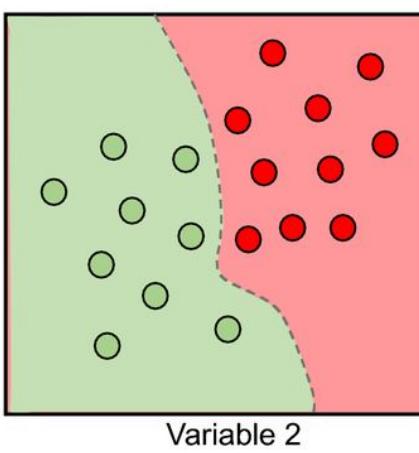
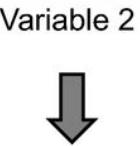
b) Supervised learning



Variable 1



Variable 1





## MACHINE LEARNING



### SUPERVISED LEARNING



### UNSUPERVISED LEARNING



#### CLASSIFICATION



#### REGRESSION



#### CLUSTERING



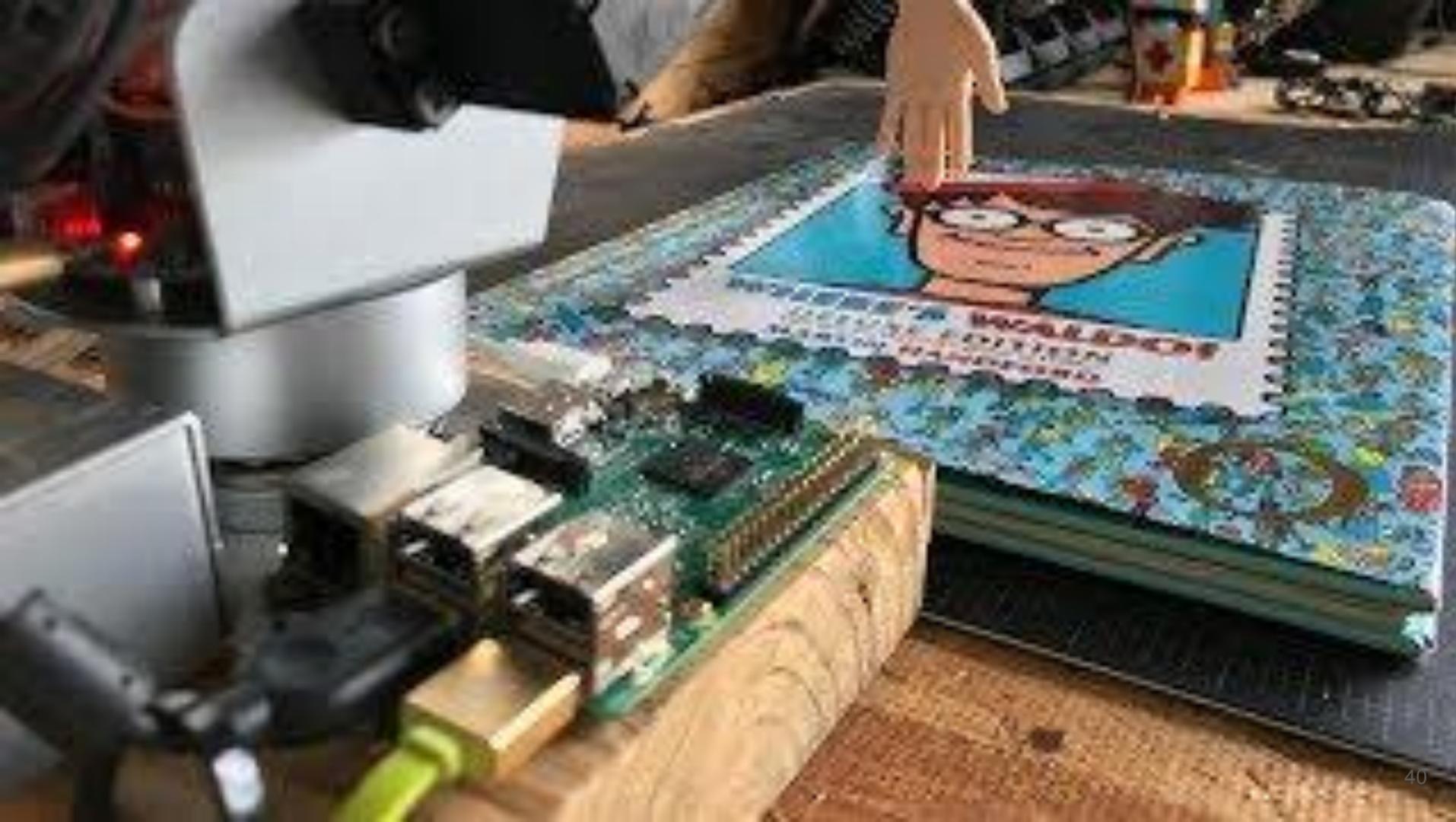
#### ASSOCIATION

## Classic Machine Learning



## Deep Learning





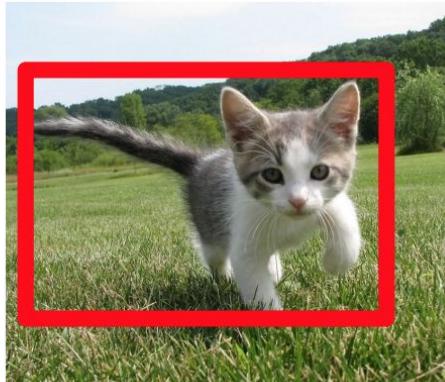
## Semantic Segmentation



GRASS, CAT,  
TREE, SKY

No objects, just pixels

## Classification + Localization



CAT

Single Object

## Object Detection



DOG, DOG, CAT

Multiple Object

## Instance Segmentation

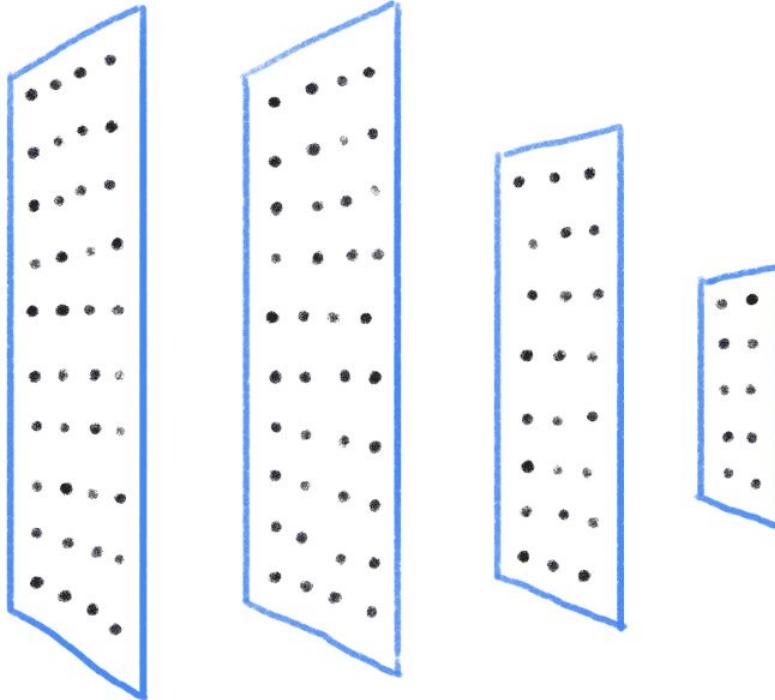


DOG, DOG, CAT

CAT

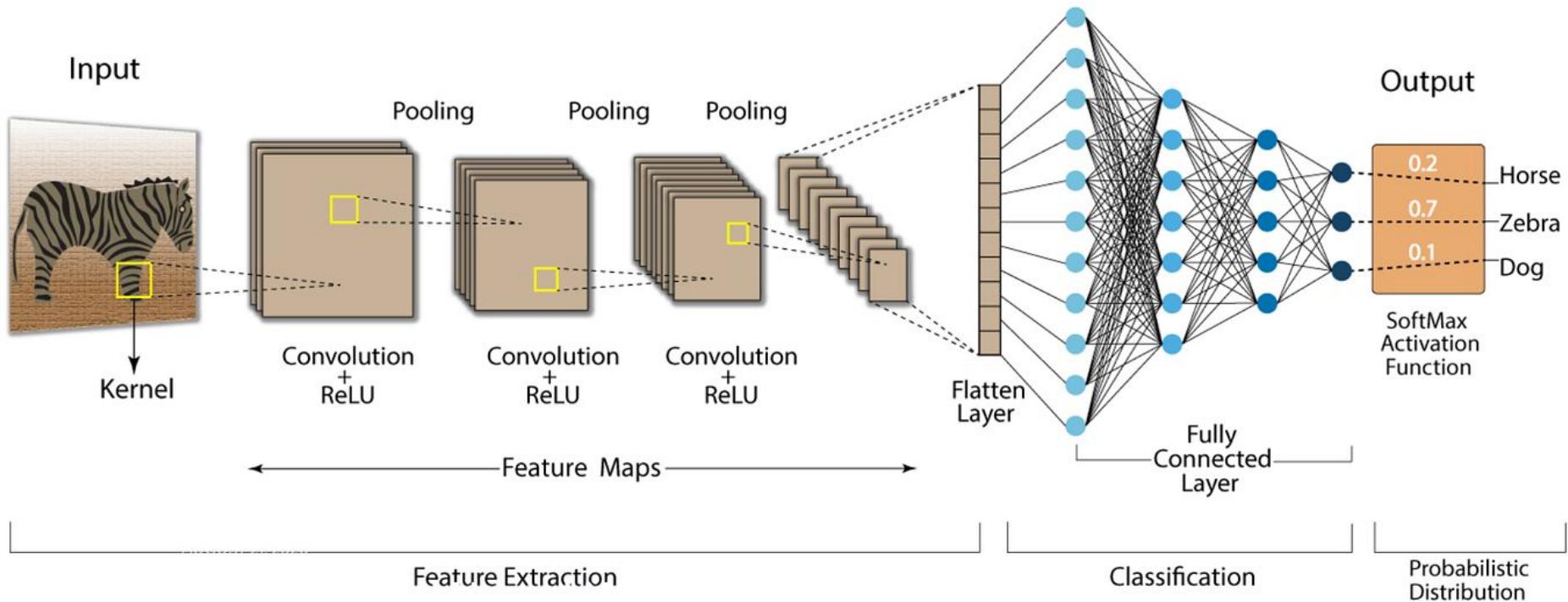
(LABELED)  
PHOTOS

DOG



OUTPUT

# Convolution Neural Network (CNN)



# Architecture of LeNet5

LeNet5 is a network made up of 7 layers. It consists of 3 convolution layers, two subsampling layers, and two fully connected layers.

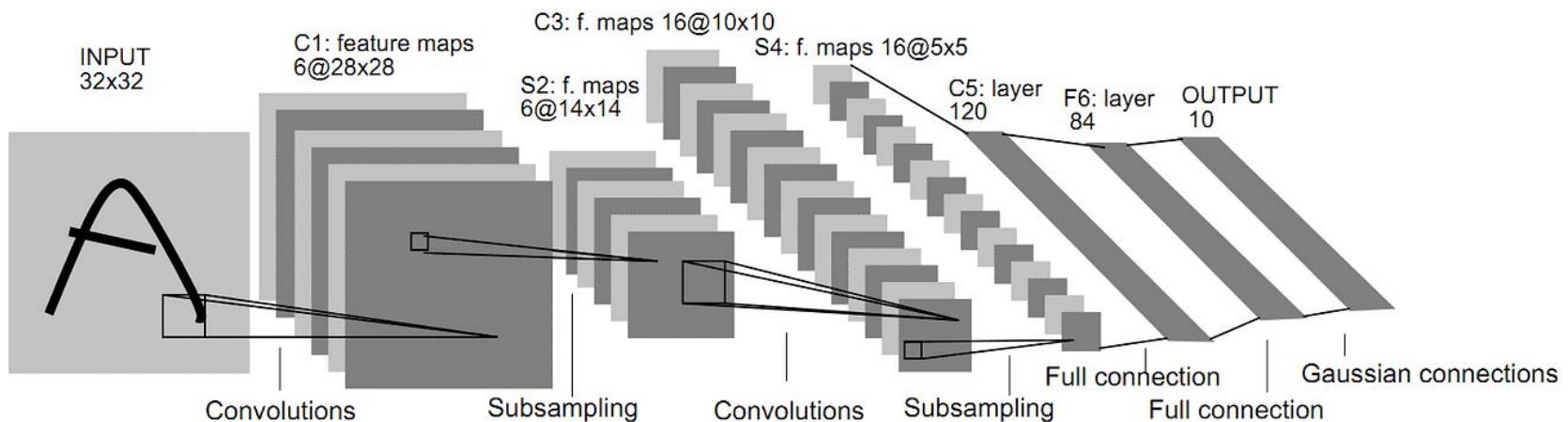
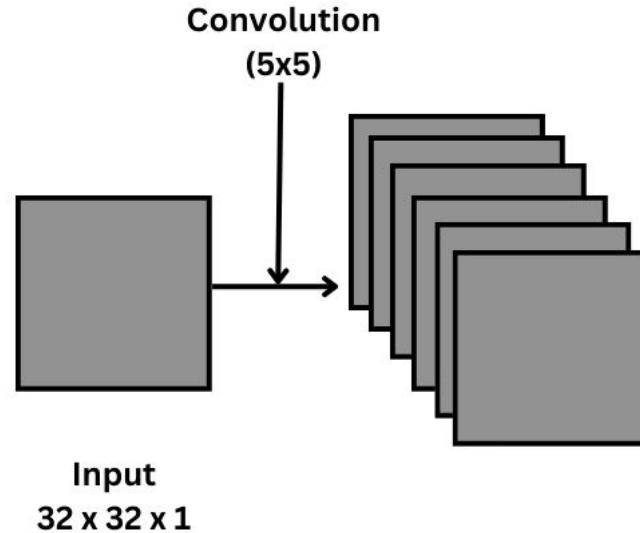


Fig. 2. Architecture of LeNet-5, a Convolutional Neural Network, here for digits recognition. Each plane is a feature map, i.e. a set of units whose weights are constrained to be identical.

# LeNet5: First layer

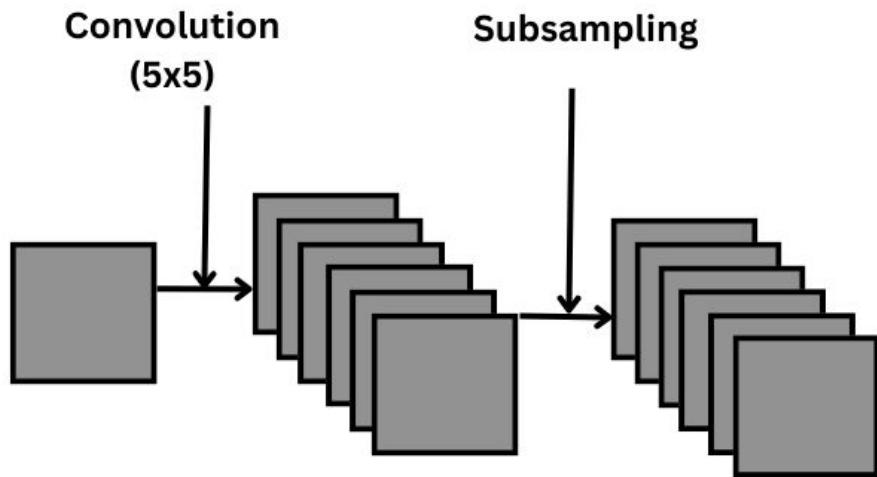
After this, the first convolution step takes place, and the input image is convoluted to the size of 28x28.



$$\begin{aligned}\text{Output Shape} &= ((32 - 5 + 1) \times (32 - 5 + 1) \times 6) \\ &= (28 \times 28 \times 6)\end{aligned}$$

# LeNet5: Second layer

Next is the subsampling layer, in which the size is reduced to half, i.e.,  $14 \times 14$ .



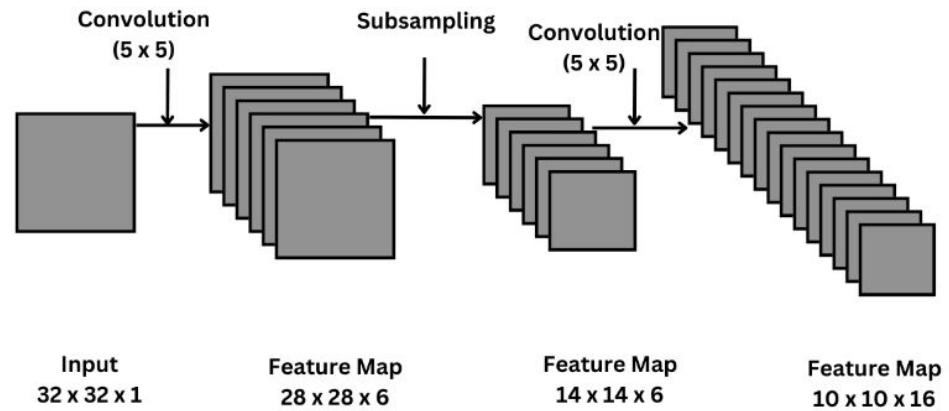
**Input**  
 $32 \times 32 \times 1$

**Feature Map**  
 $28 \times 28 \times 6$

**Feature Map**  
 $14 \times 14 \times 6$

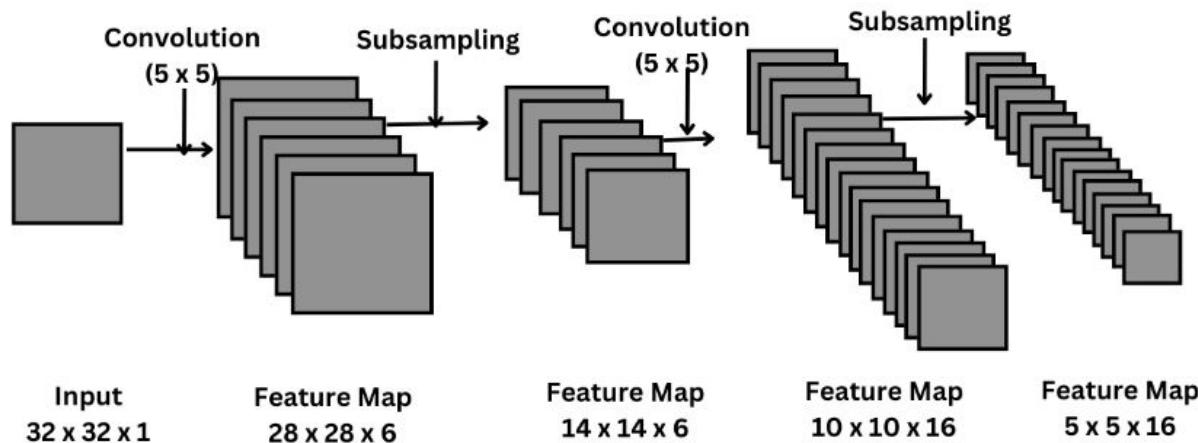
# LeNet5: Third layer

In the third layer, convolution occurs again, but this time with 16 filters of 5x5 size. After this layer, the size of the input image is reduced to 10x10x16.



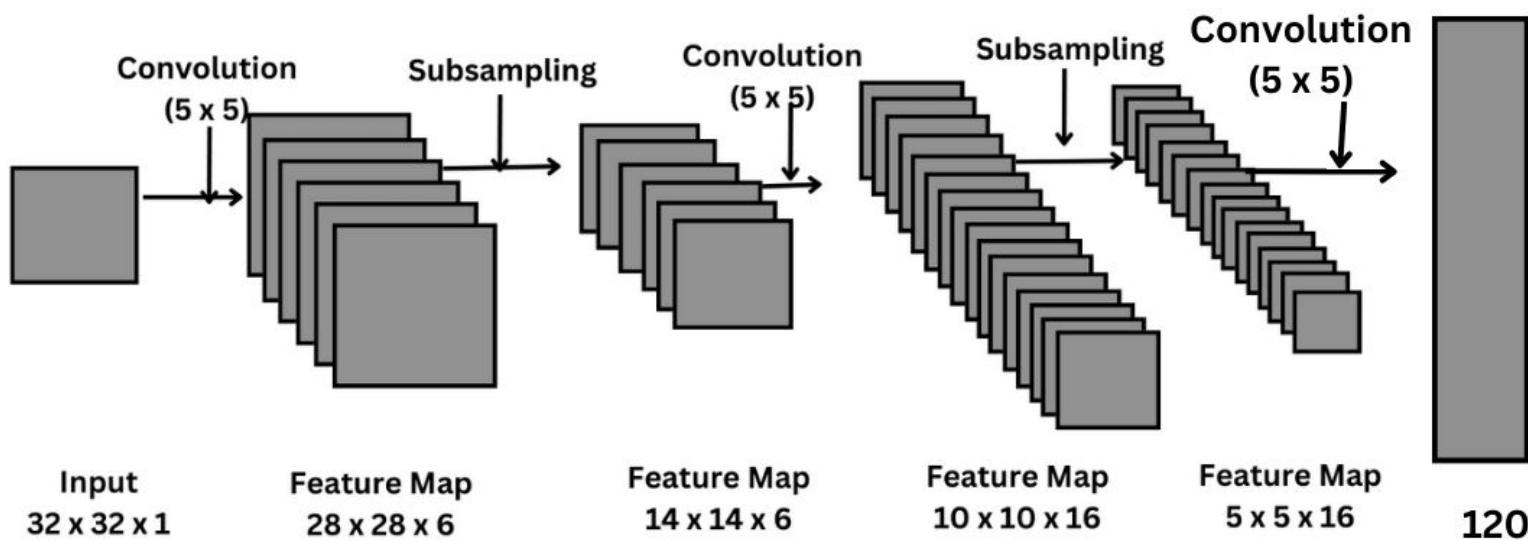
# LeNet5: Fourth layer

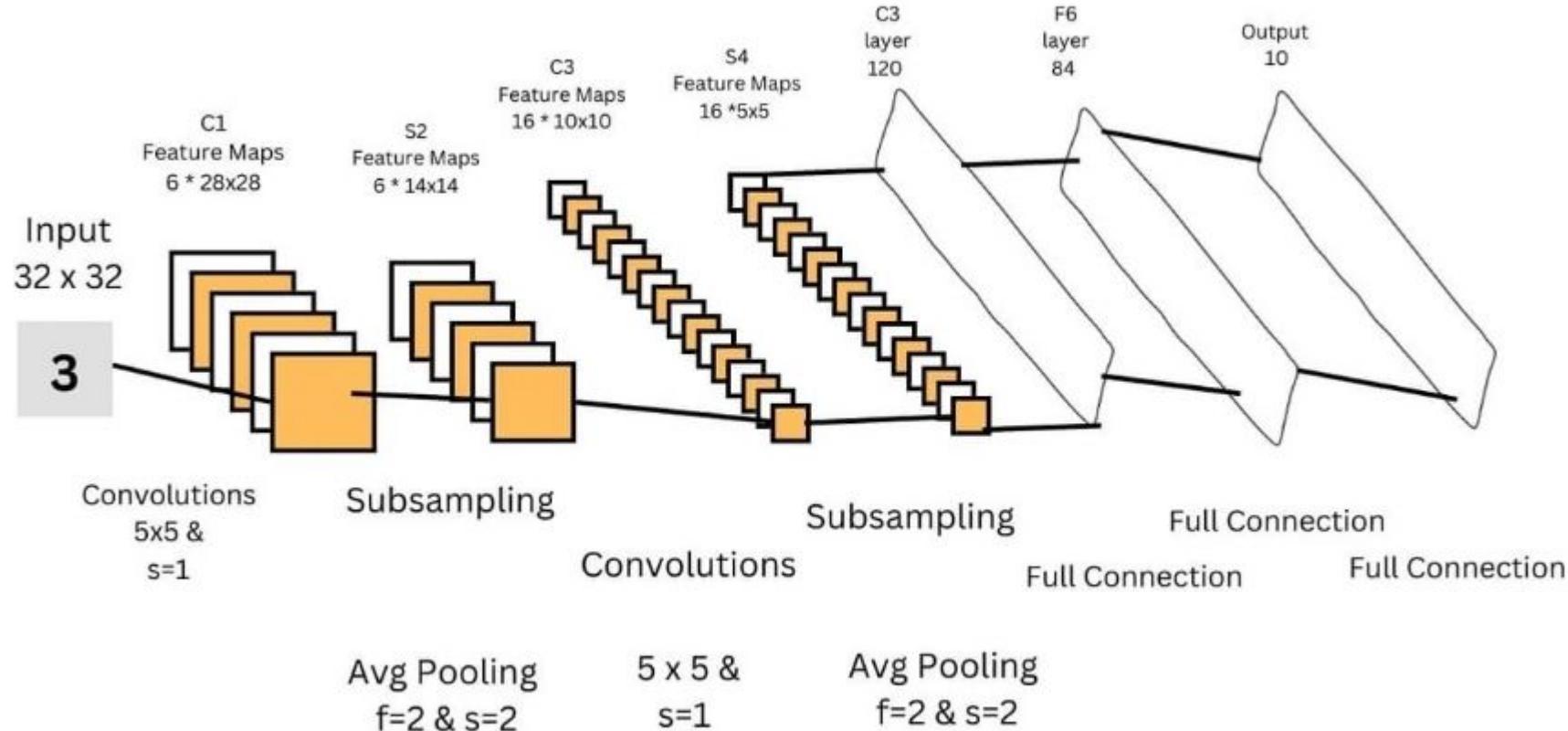
The subsampling takes place, and the image size in this step is reduced to  $5 \times 5 \times 16$ . In this layer, the input for the very last function diagram comes from all the remaining function diagrams.



# LeNet5: Fifth layer

This is the final layer of the convolution-subsampling pair. There are 120 filters in this layer with a convolution size of 5x5. After this layer, the feature map is 1x1x120, and the flattened result is 120 values.





# Summary of the architecture of LeNet5

Layer	Filter	Filter Size	Stride	Size of feature map	Activation Function
Input	-	-	-	$32 \times 32 \times 1$	-
Conv 1	6	$5 \times 5$	1	$28 \times 28 \times 6$	tanh
Pooling 1	-	$2 \times 2$	2	$14 \times 14 \times 6$	-
Conv 2	16	$5 \times 5$	1	$10 \times 10 \times 16$	tanh
Pooling 2	-	$2 \times 2$	2	$5 \times 5 \times 16$	-
Conv 3	120	$5 \times 5$	1	120	tanh
Fully Connected 1	-	-	-	84	tanh
Fully Connected 2	-	-	-	10	Softmax

## Confusion Matrix and ROC Curve

		Predicted Class	
		No	Yes
Observed Class	No	TN	FP
	Yes	FN	TP

### Model Performance

Accuracy =  $(TN+TP)/(TN+FP+FN+TP)$

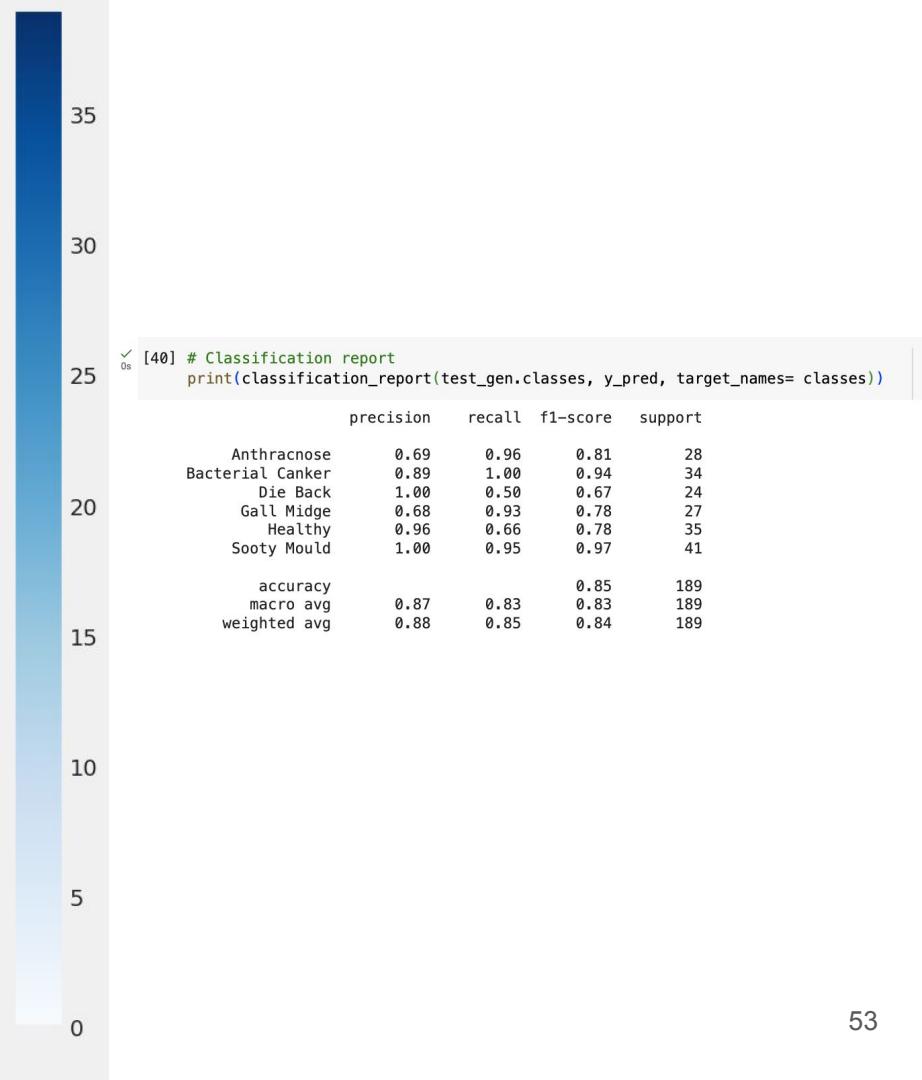
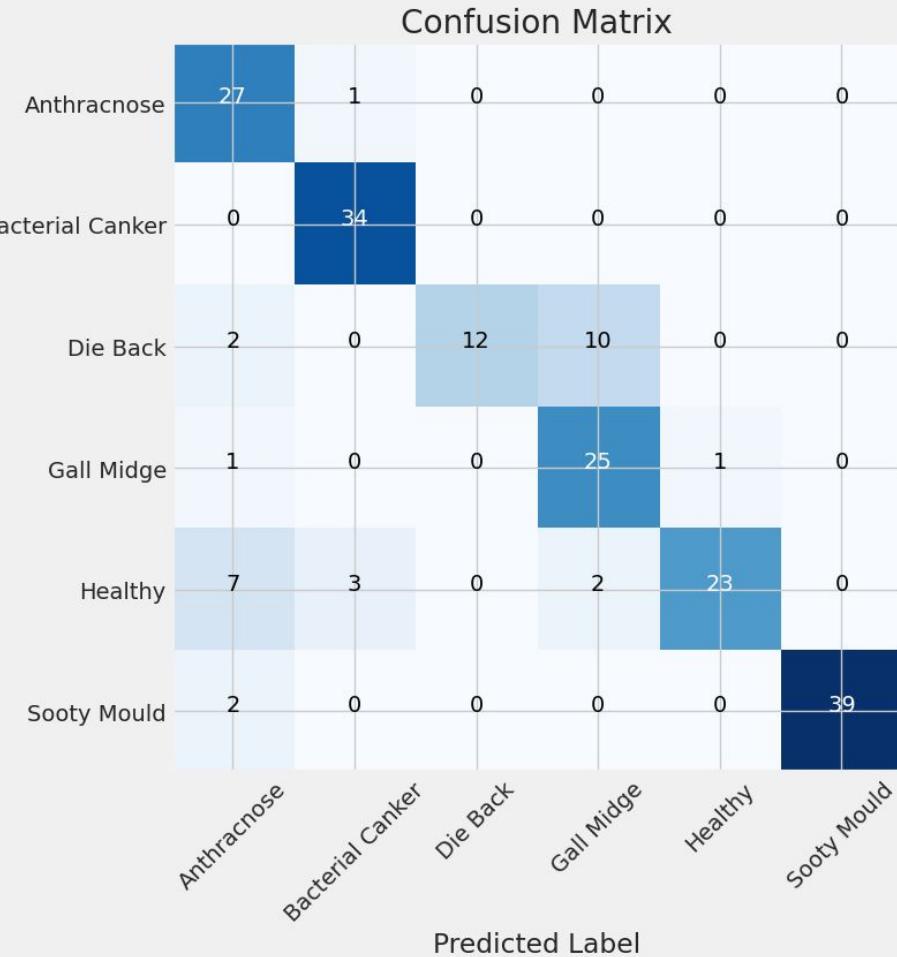
Precision =  $TP/(FP+TP)$

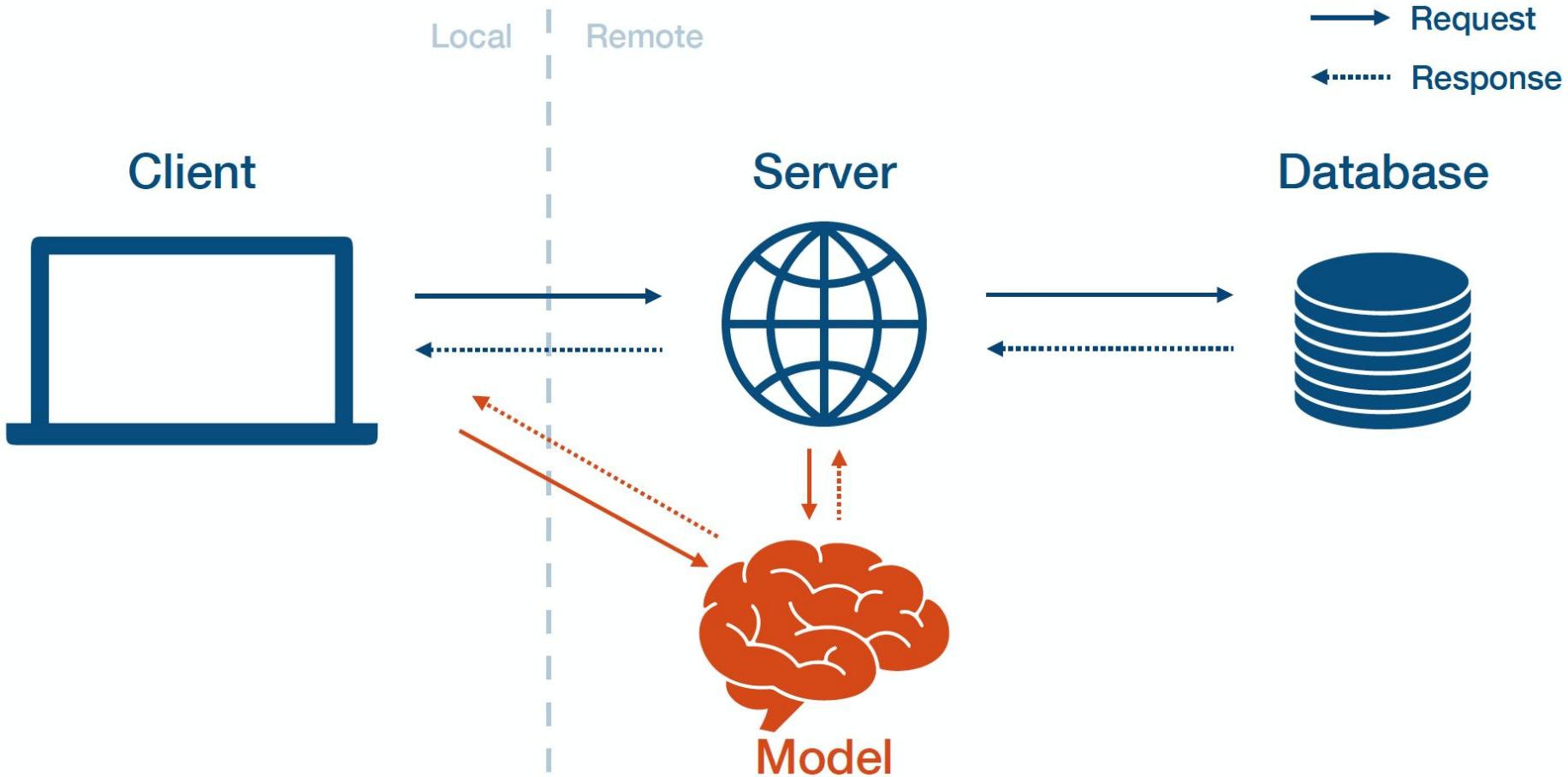
Sensitivity =  $TP/(TP+FN)$

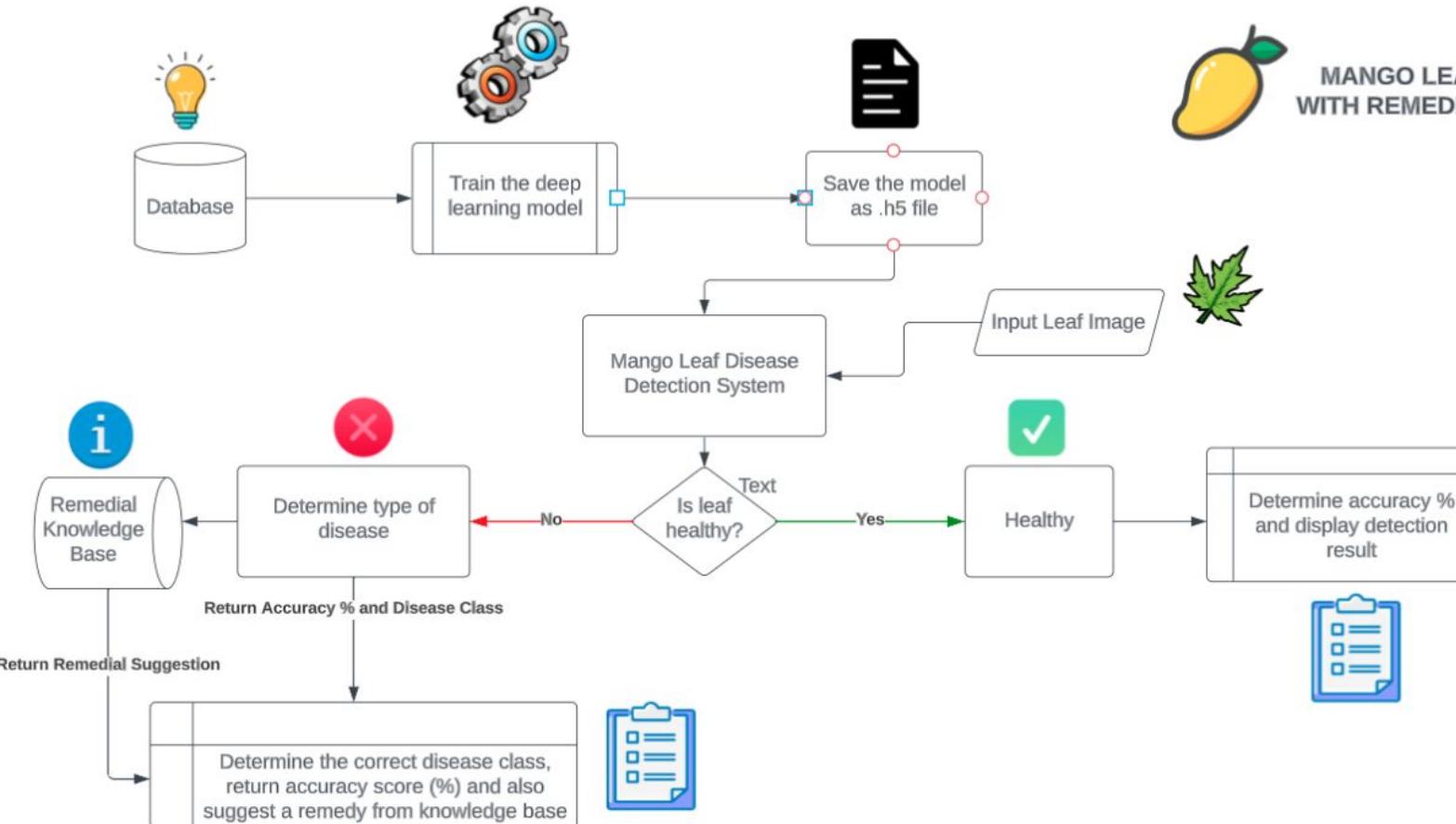
Specificity =  $TN/(TN+FP)$

- TN True Negative
- FP False Positive
- FN False Negative
- TP True Positive

True Label







# Artificial Intelligence Projects

Mango 🍌 Leaf 🍃 Disease Dataset

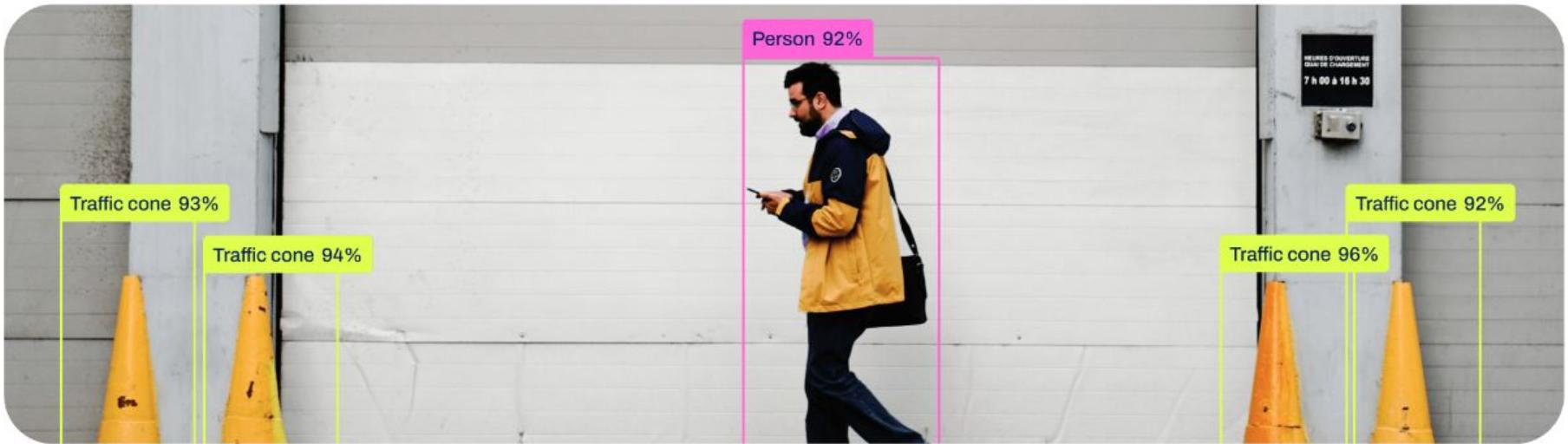


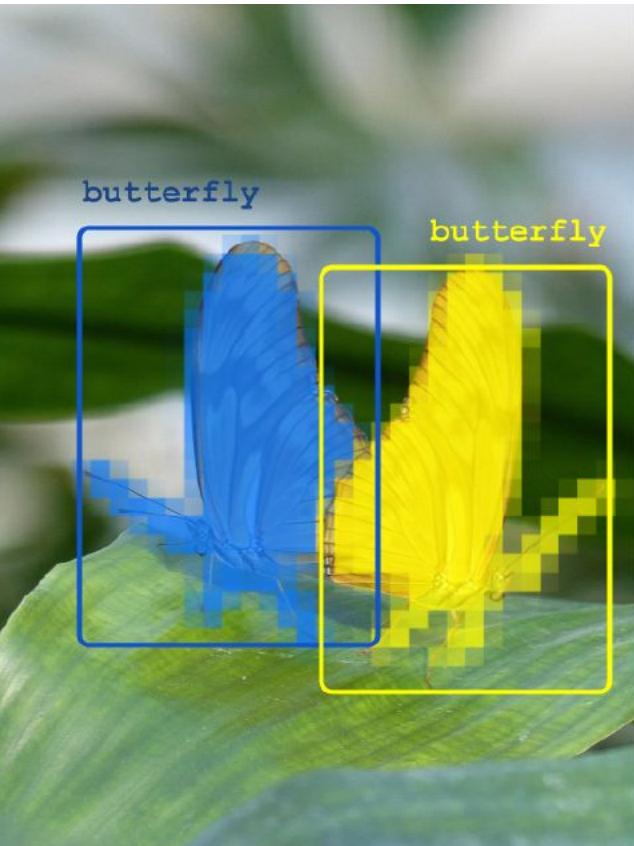
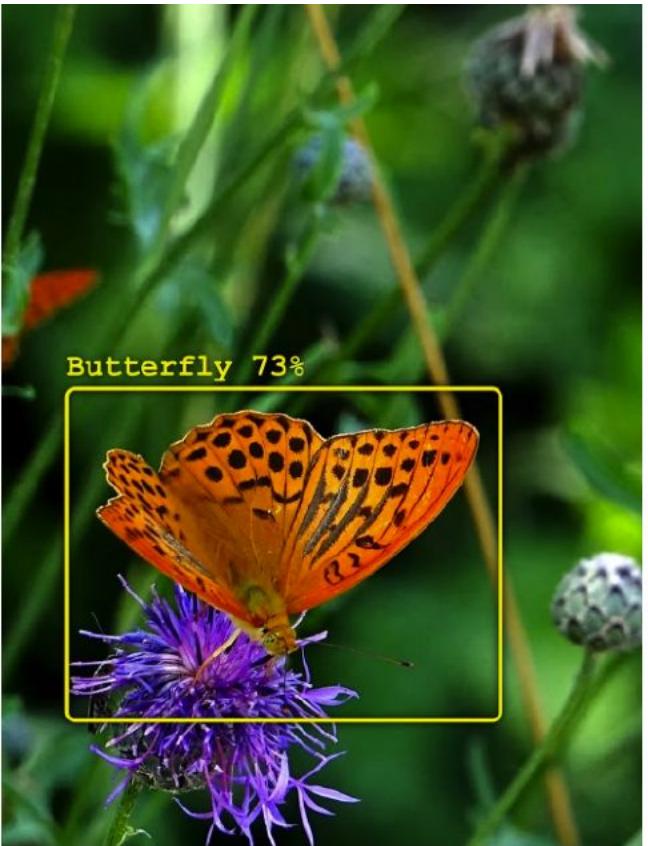
กรมที่ดิน  
Department of Lands



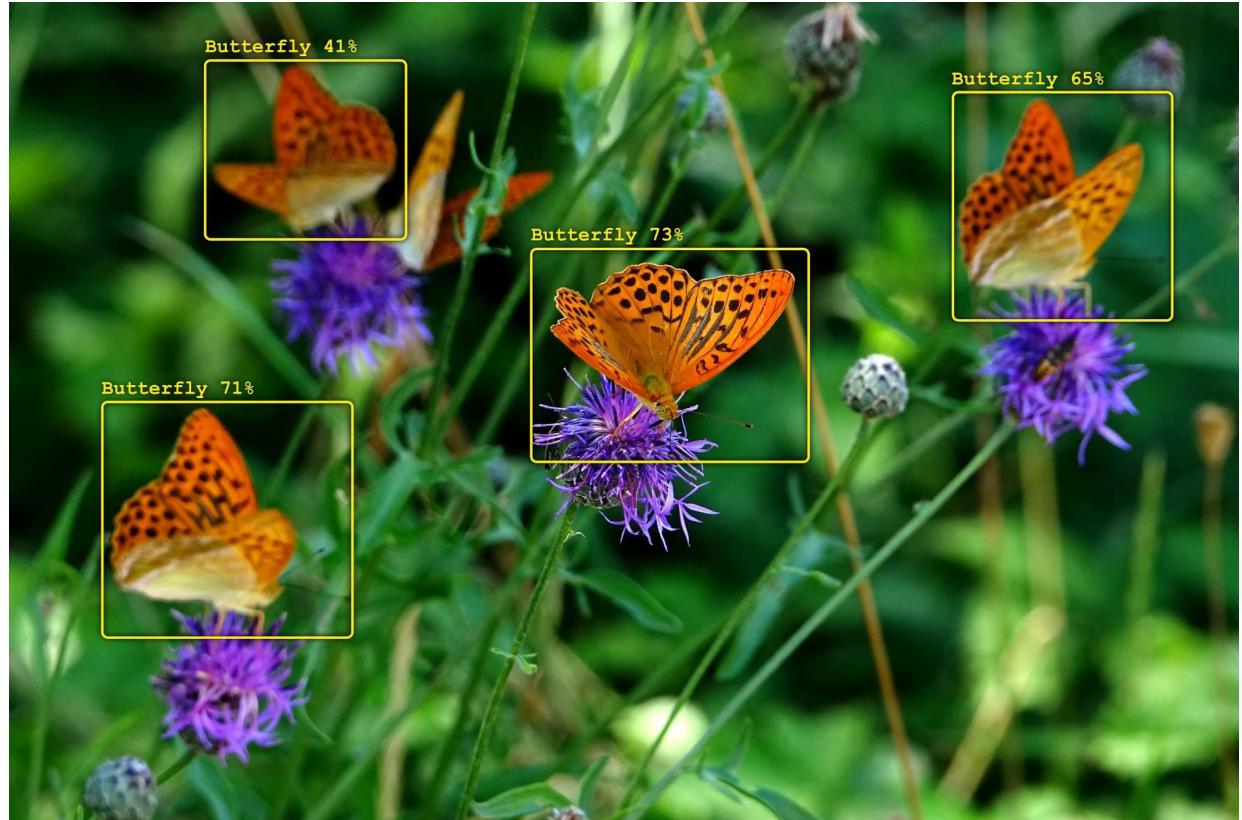
TensorFlow

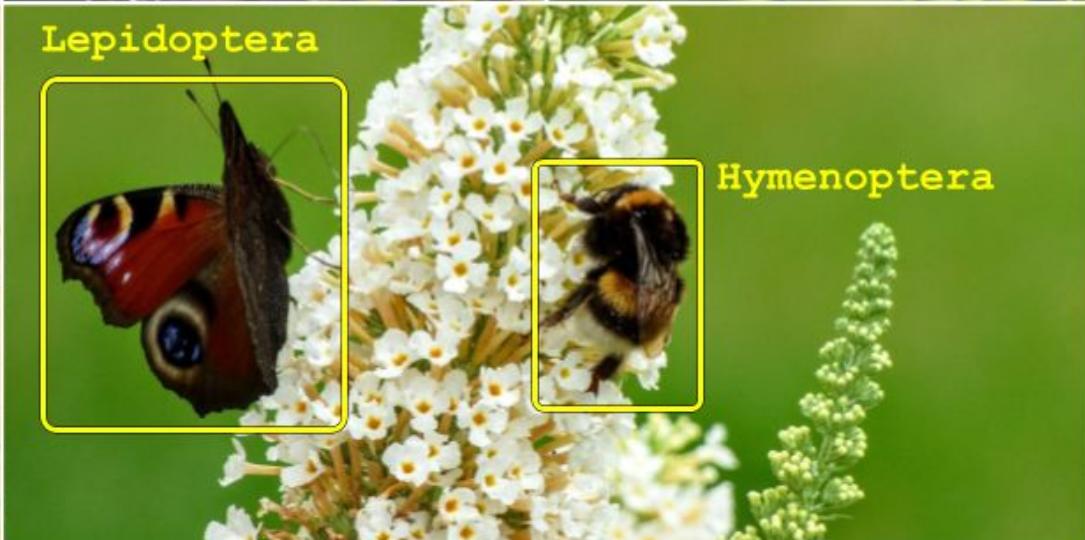
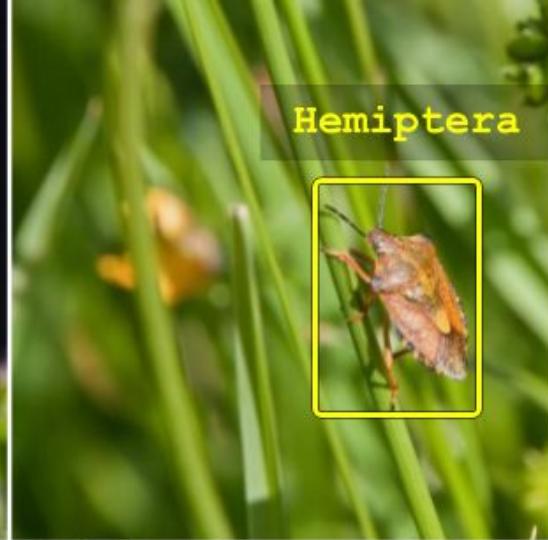
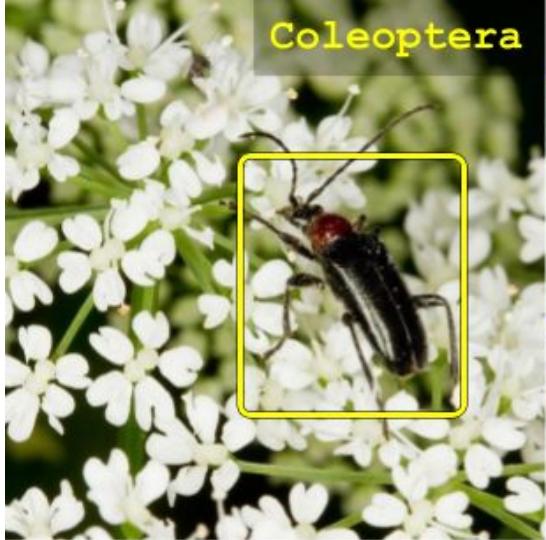
# Object Detection





# Object Detection



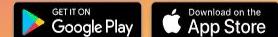


# YOLO

YOLO (you only look once) is the simplest object detection architecture. It is not the most accurate, but it's one of the fastest when it comes to prediction times.

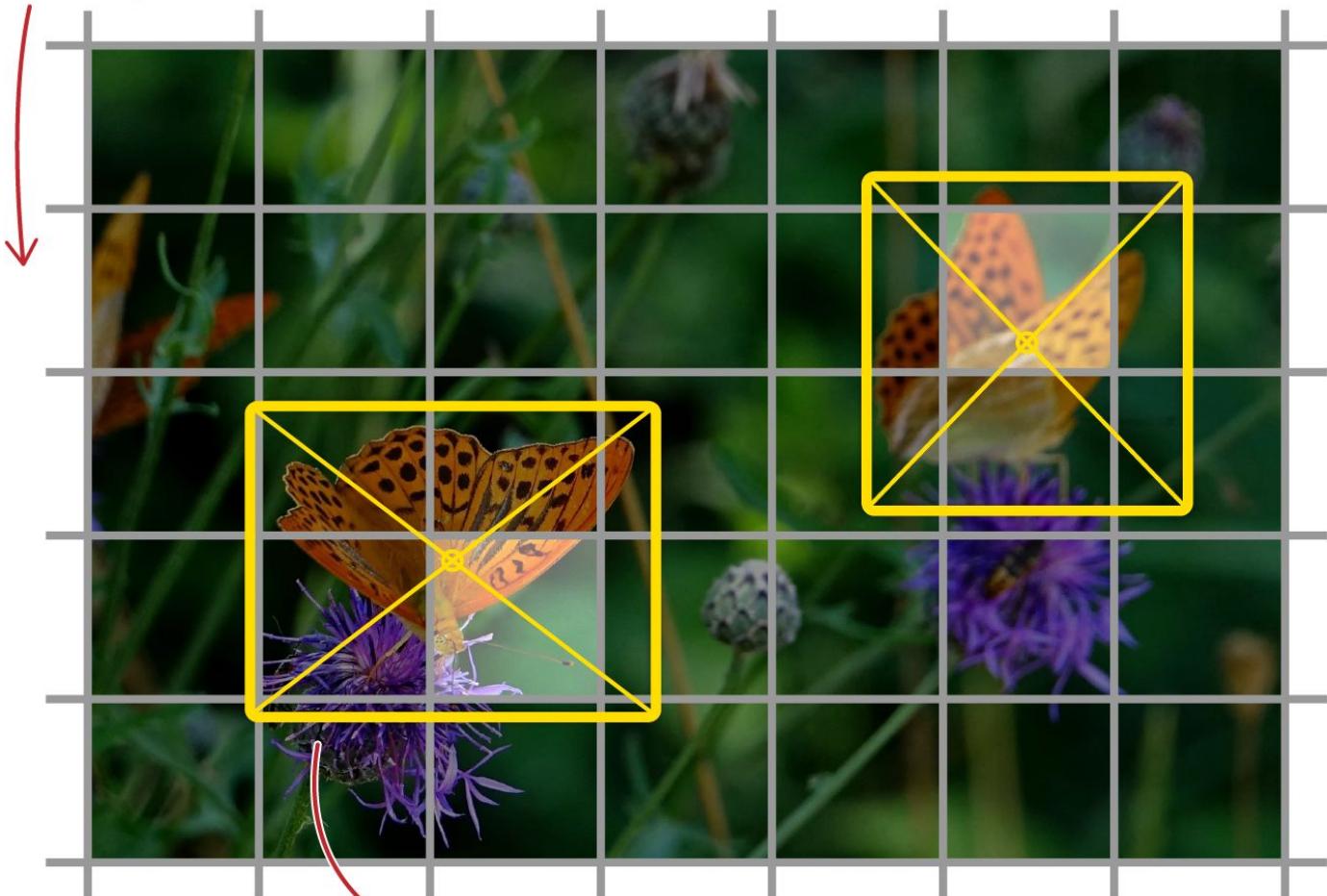


DOWNLOAD THE APP



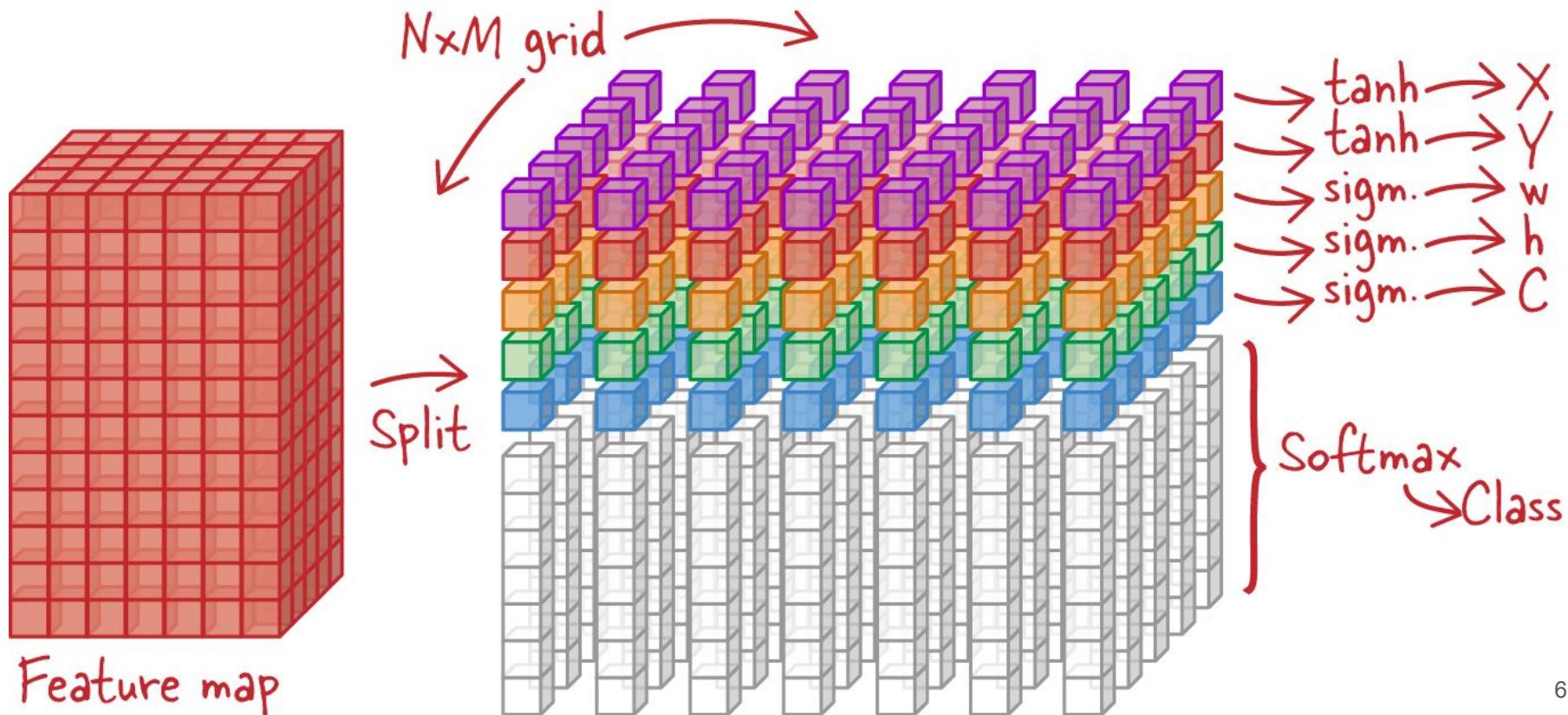
YOLO grid

$N \times M$  grid



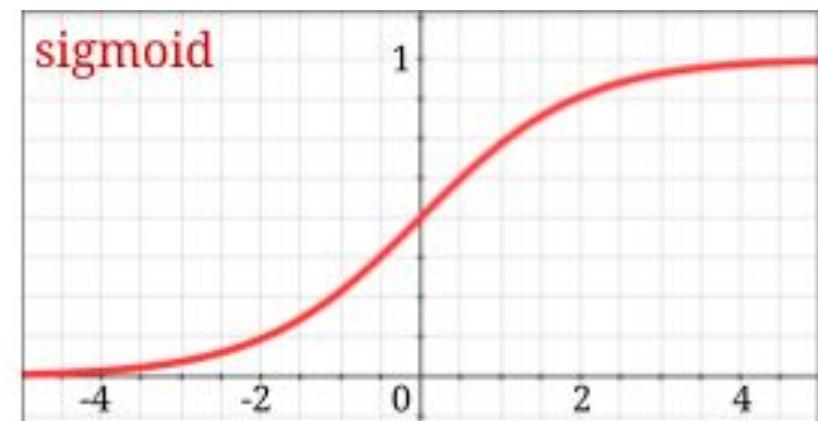
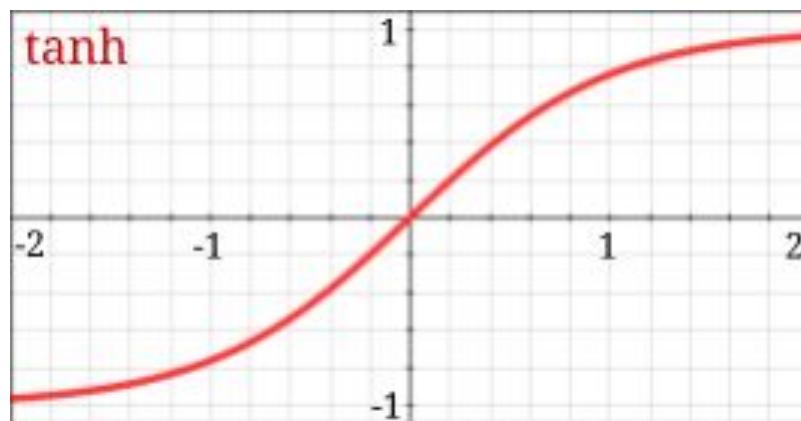
Bounding box:  $x, y, \text{width}, \text{height}, \text{confidence}, \text{class}$

# Object detection head

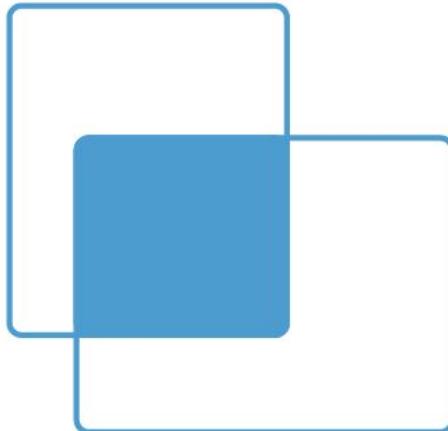


# Activation functions

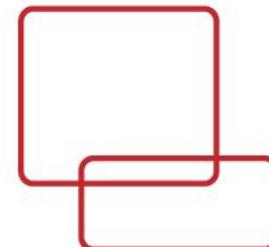
The tanh and sigmoid activation functions. Tanh outputs values in the  $[-1, 1]$  range, while the sigmoid function outputs them in the  $[0, 1]$  range.



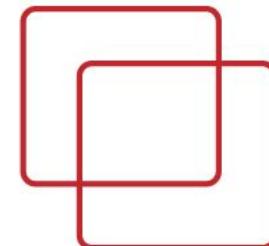
$$\text{IOU} = \frac{\text{Area of intersection}}{\text{Area of union}}$$



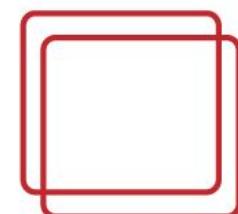
Examples



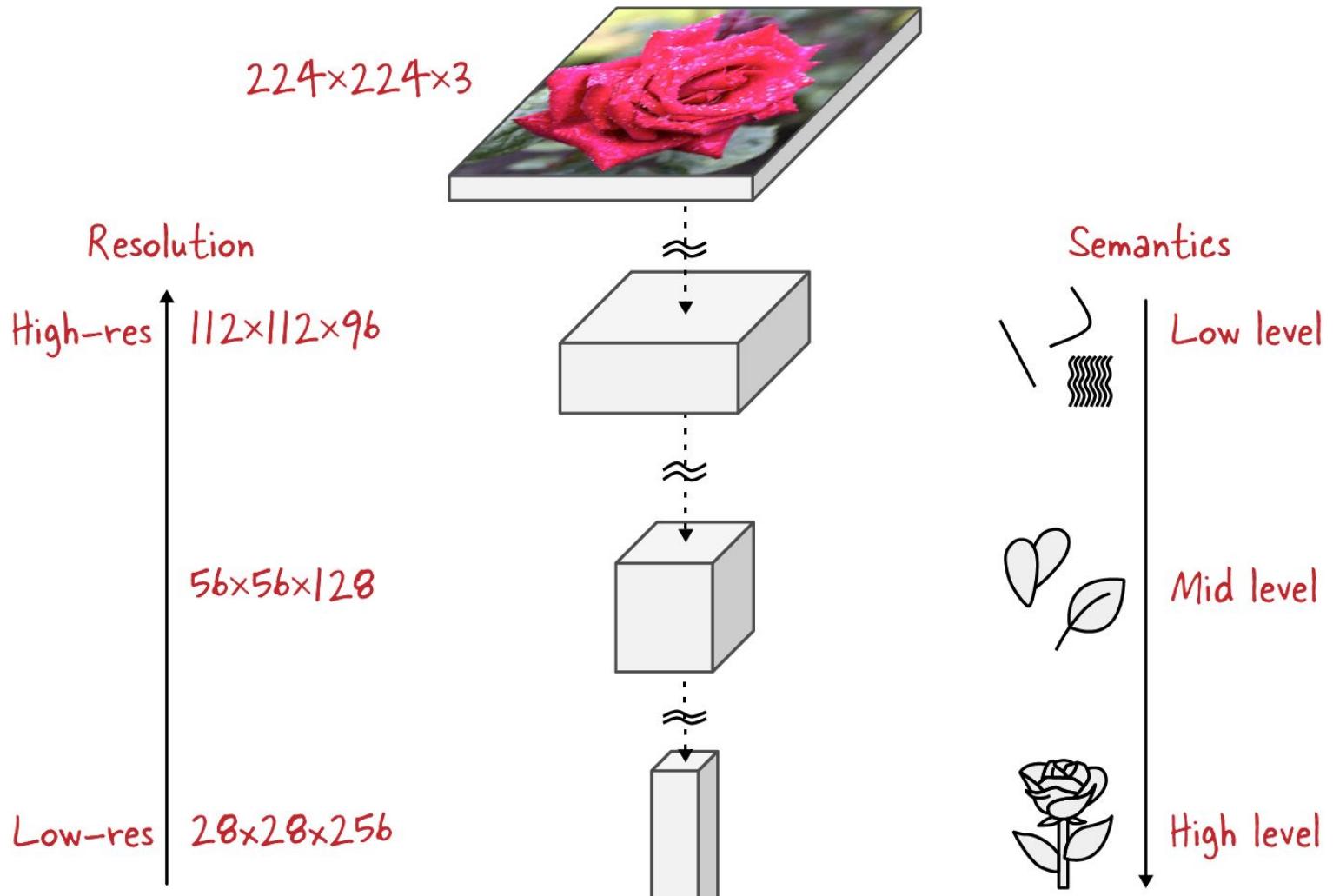
IOU = 0.1

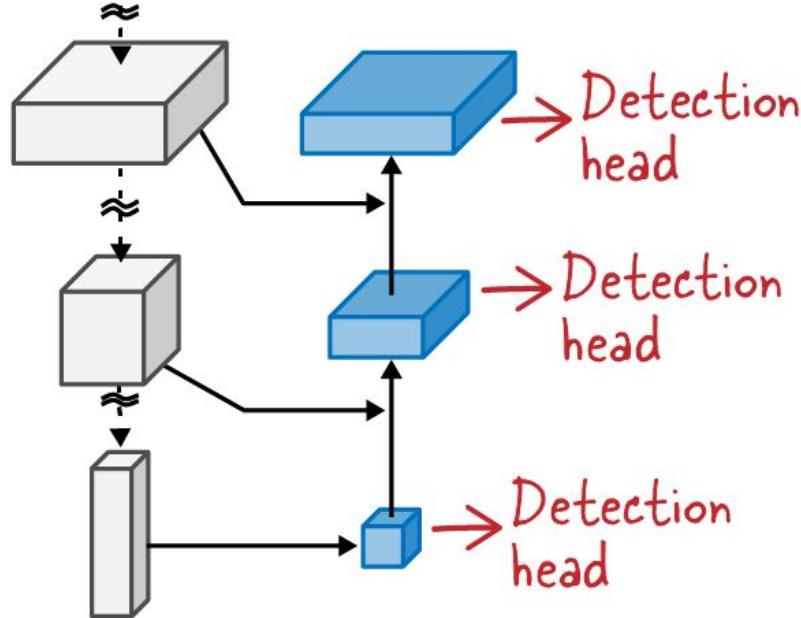
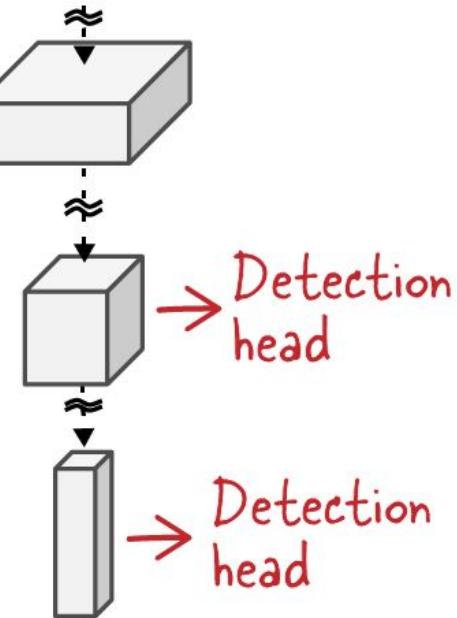
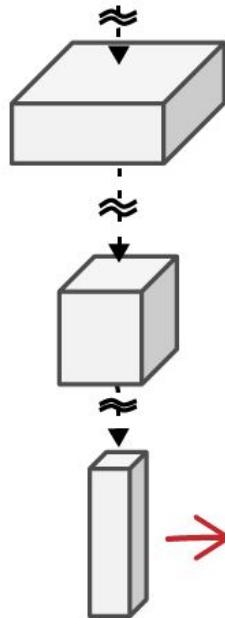


IOU = 0.3



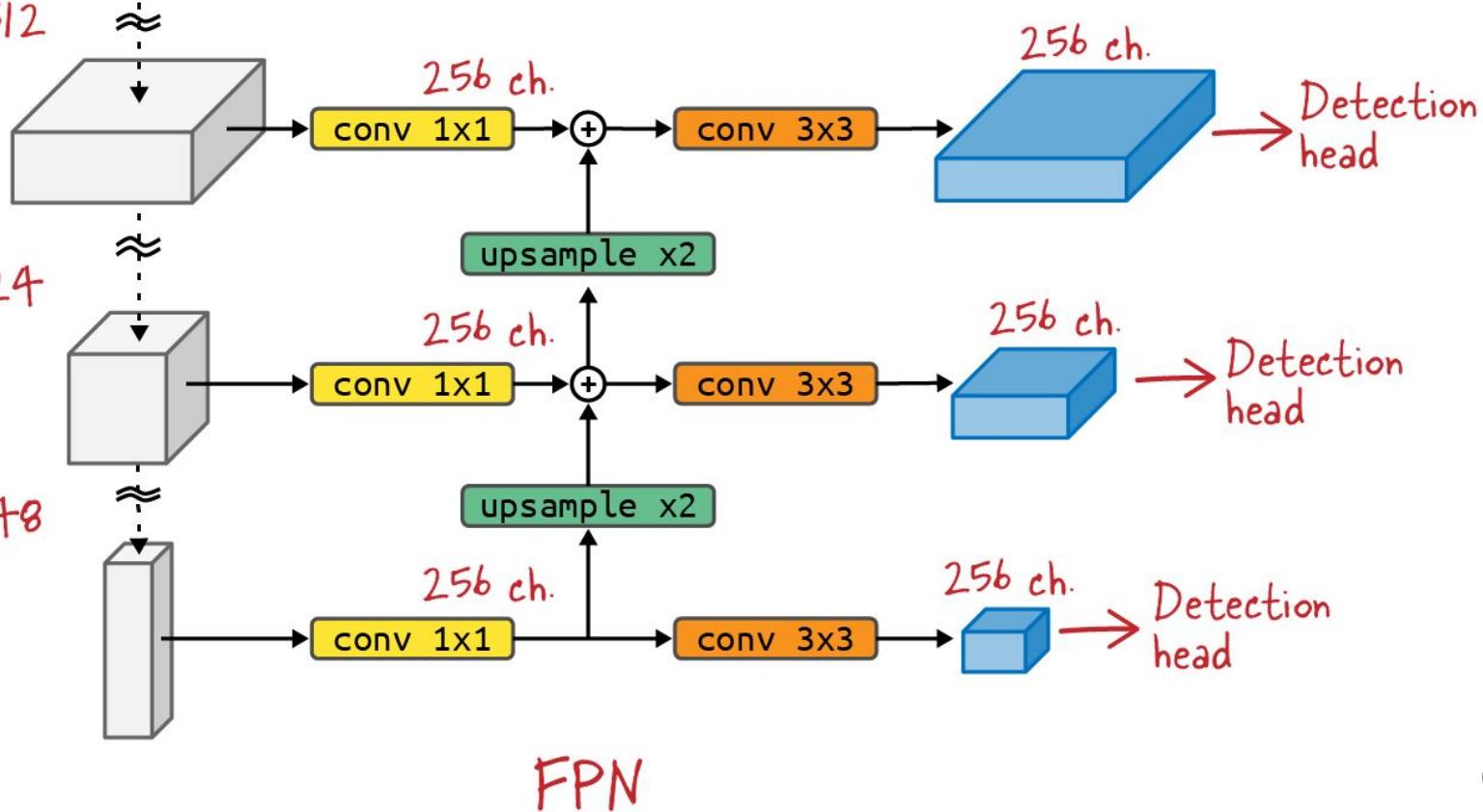
IOU = 0.6



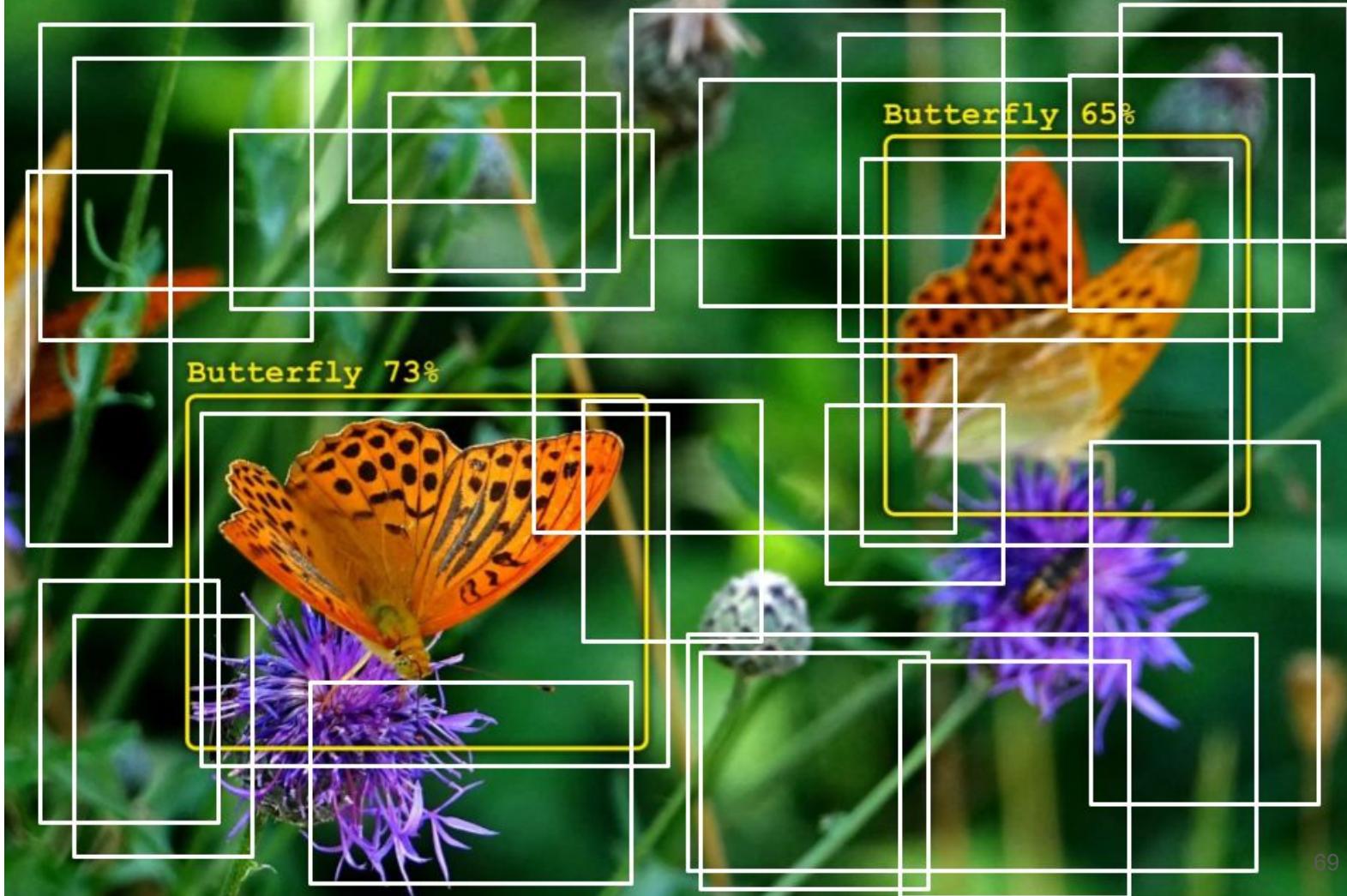




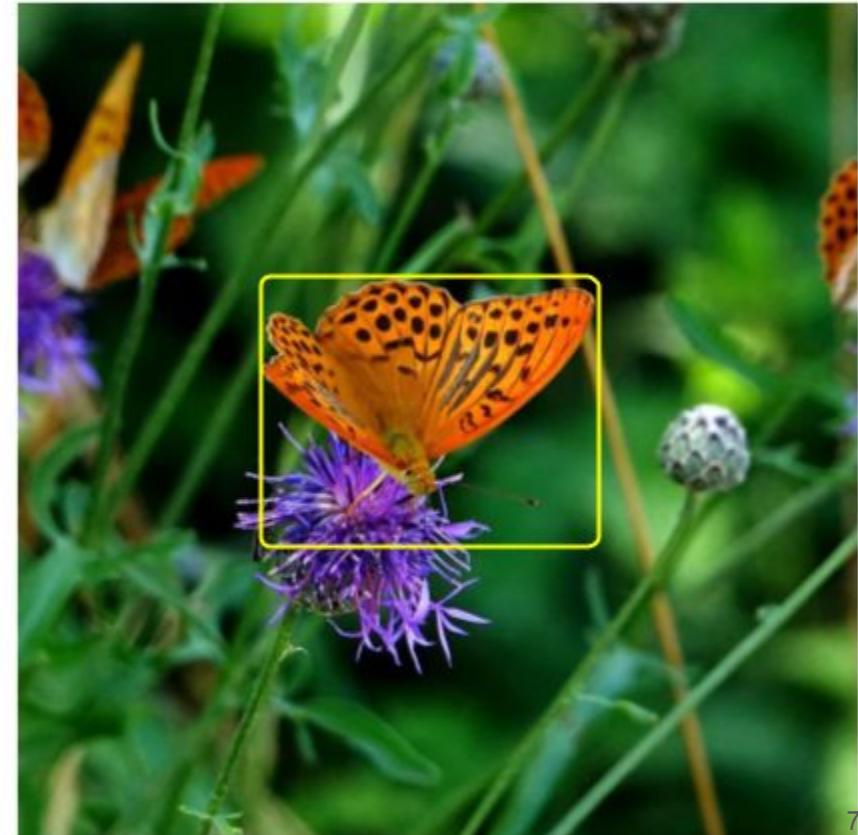
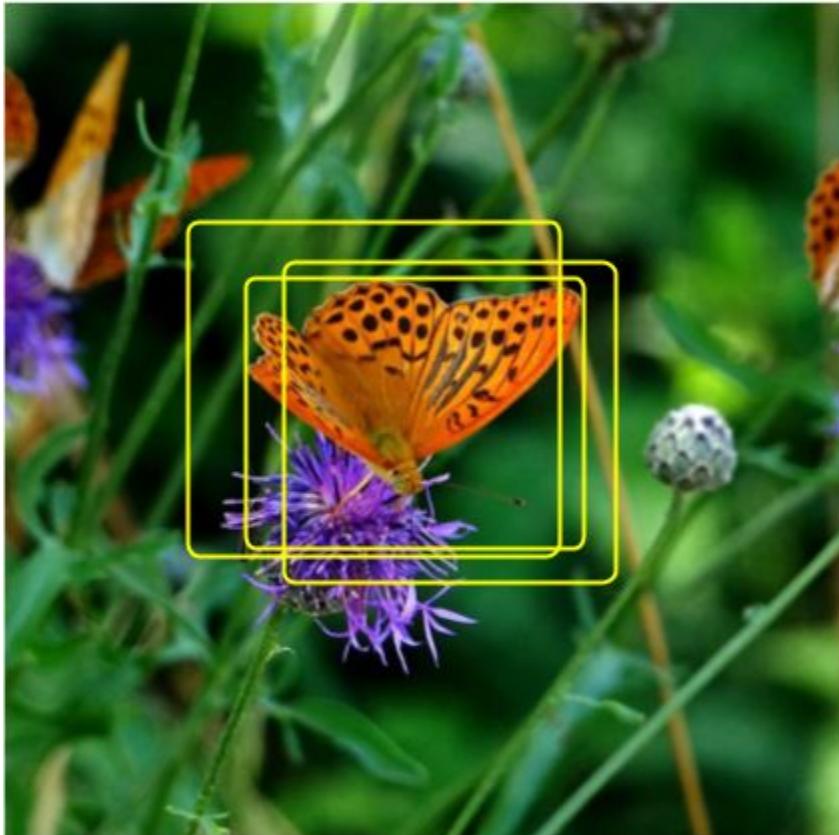
$80 \times 120 \times 512$

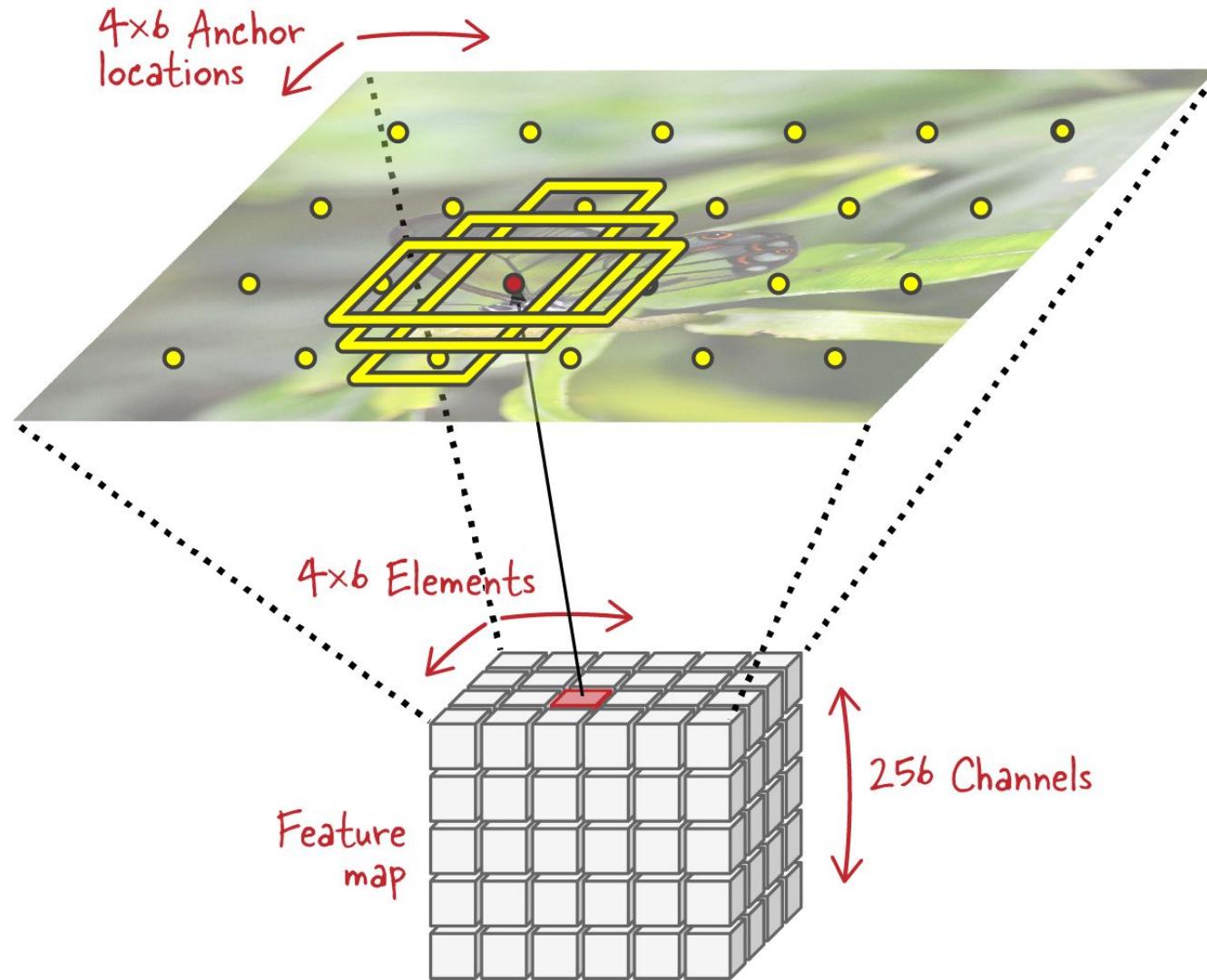


Anchor



# Non-maximum suppression (NMS)





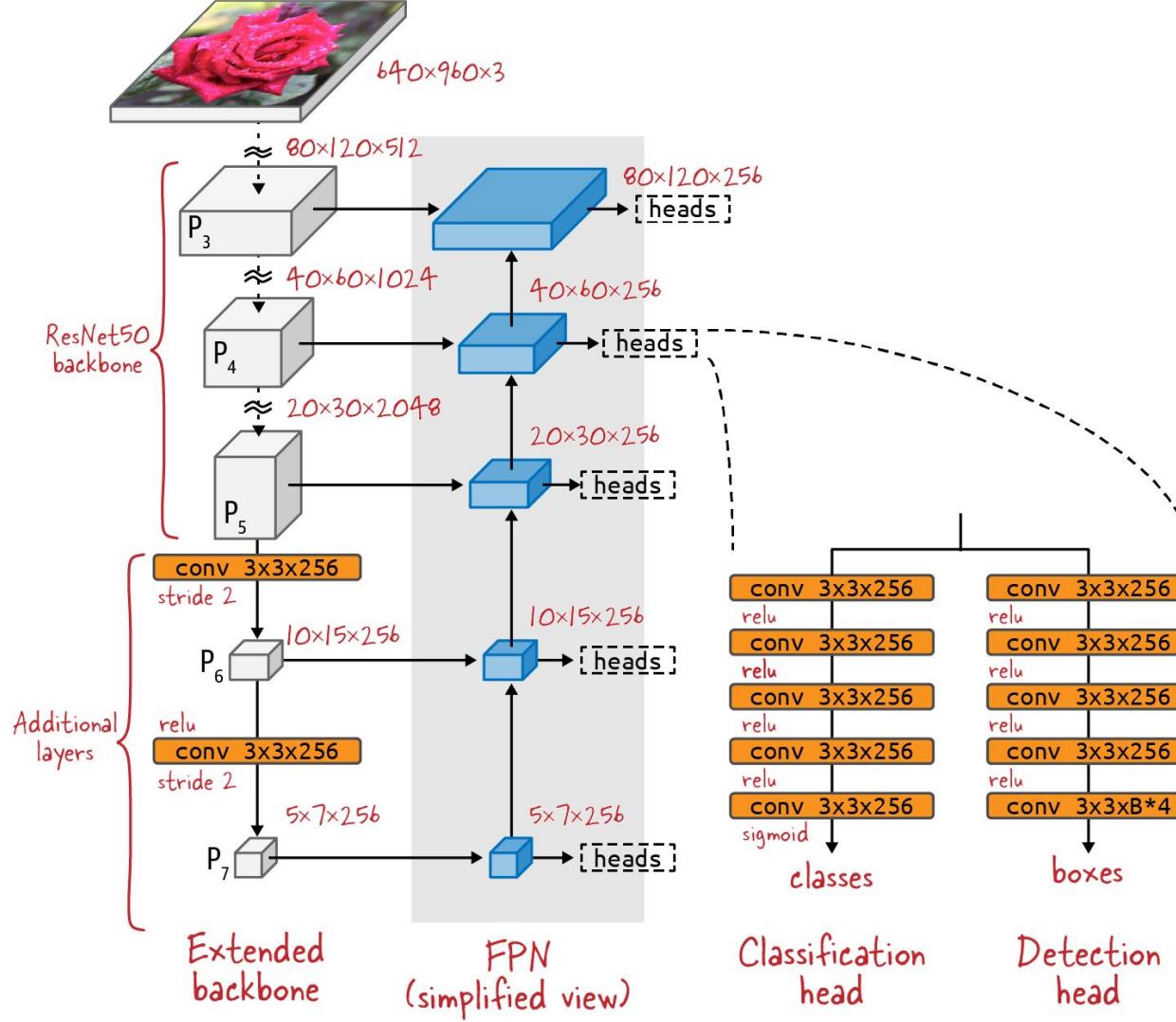
## IoU Matrix Anchors

Ground truth boxes

0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.3	0.1	0.0	0.0	0.0
0.0	0.0	0.0	0.1	0.3	0.6	0.2	0.1	0.0	0.0
0.0	0.0	0.0	0.0	0.1	0.5	0.0	0.1	0.0	0.0
0.0	0.0	0.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Low values everywhere:  
anchor = background

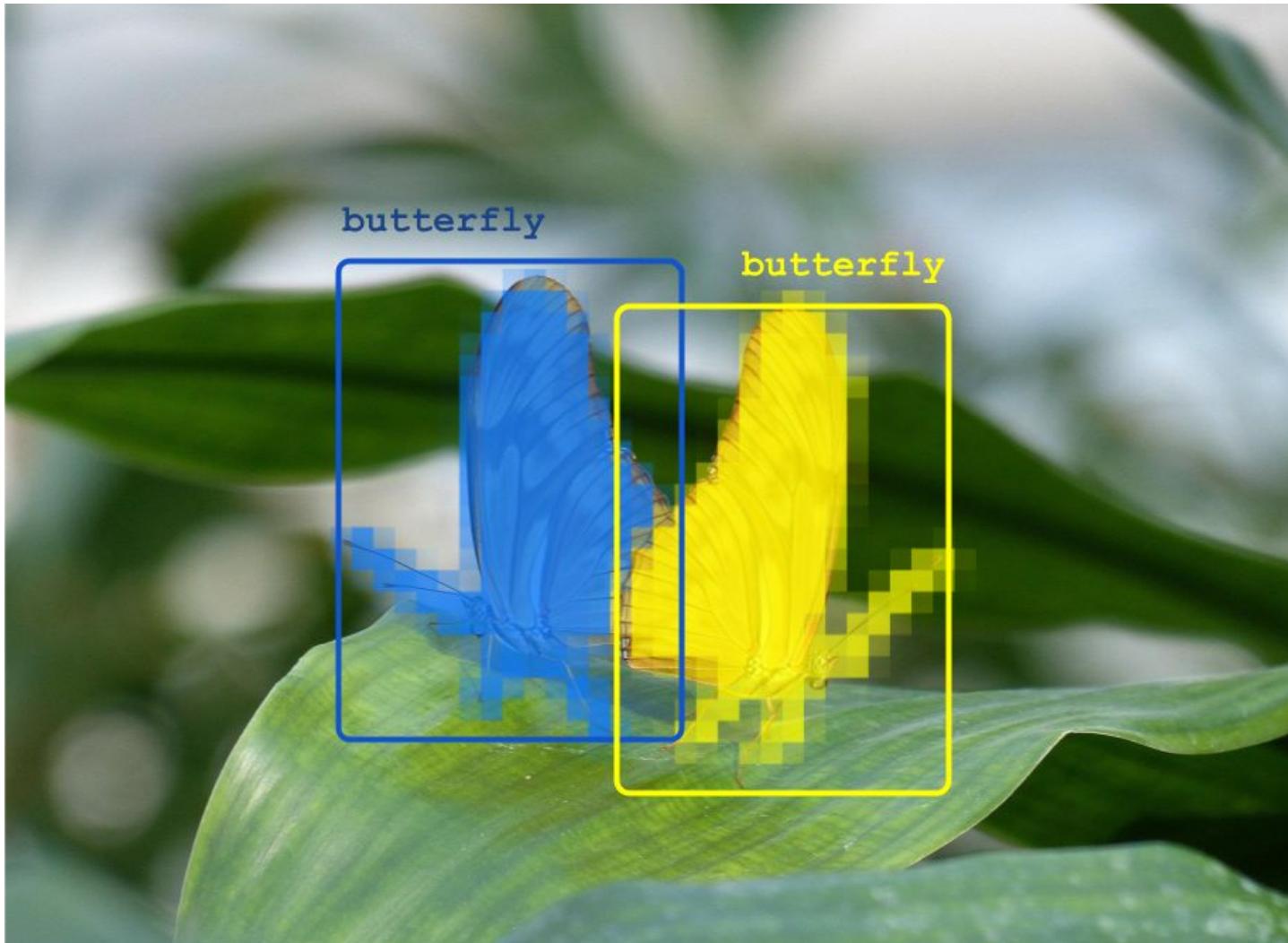
Large max value: anchor  
and ground truth box paired

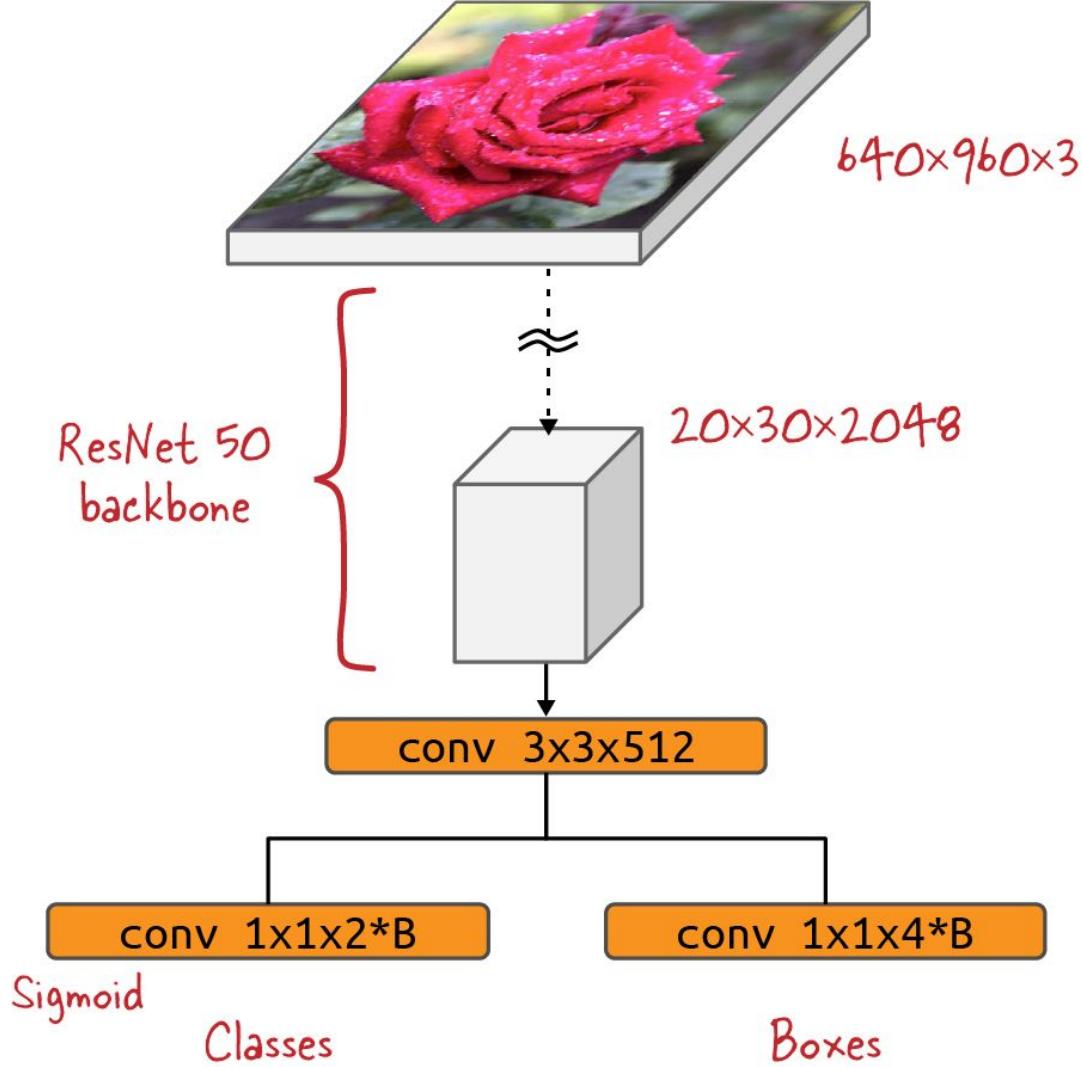






# Mask R-CNN and Instance Segmentation





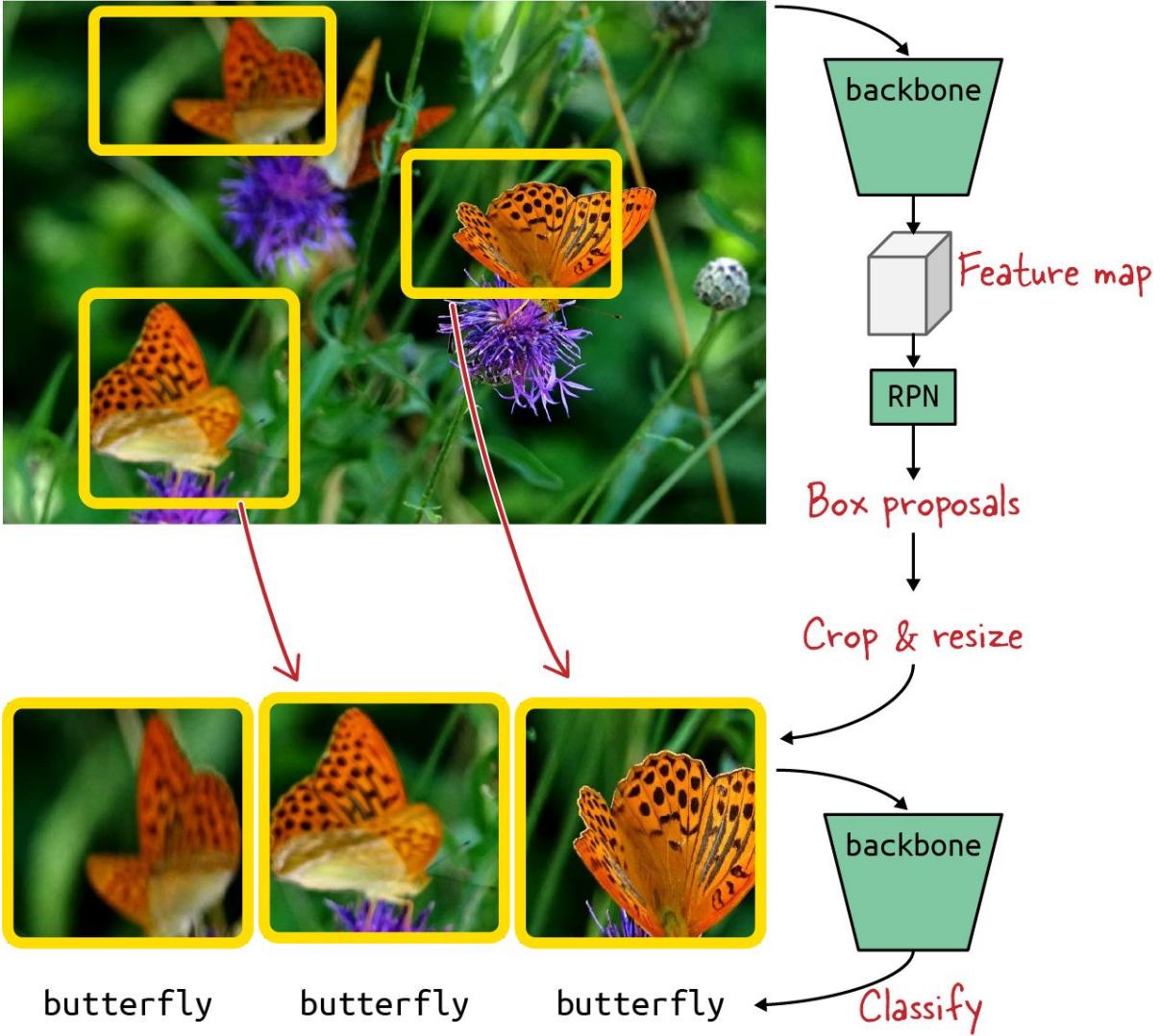
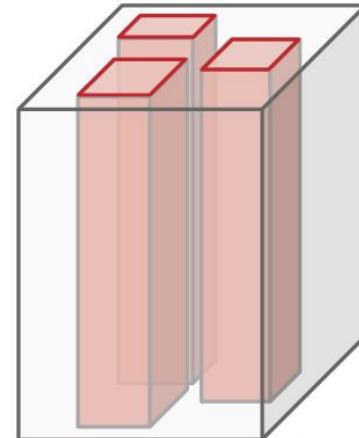


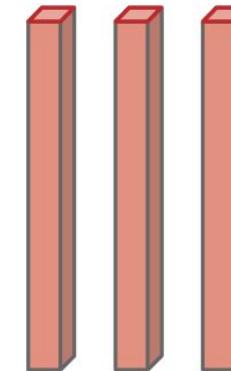
Image with ROIs



Feature map



Extracted feature maps



ROI map & align

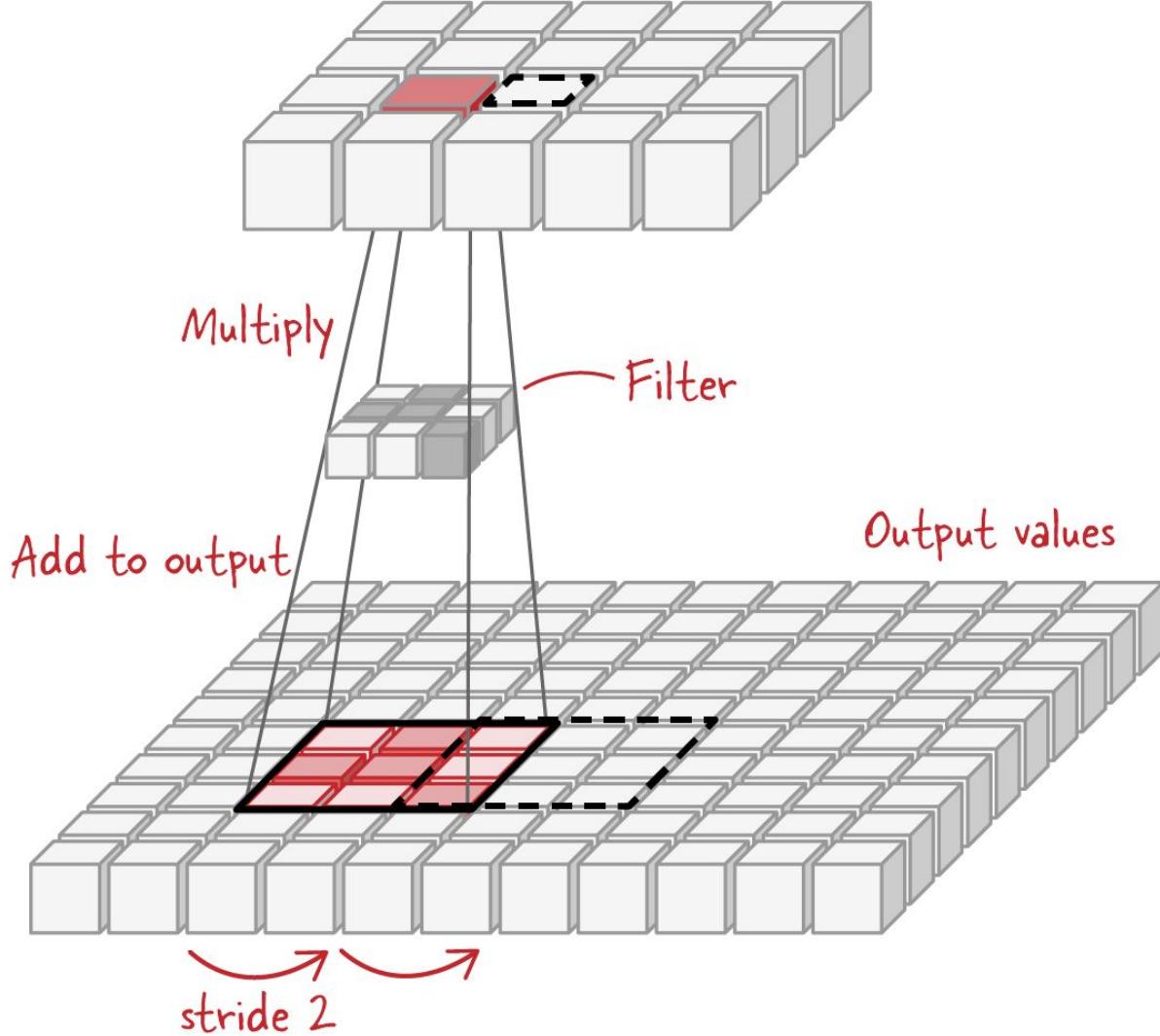
Crop & resize

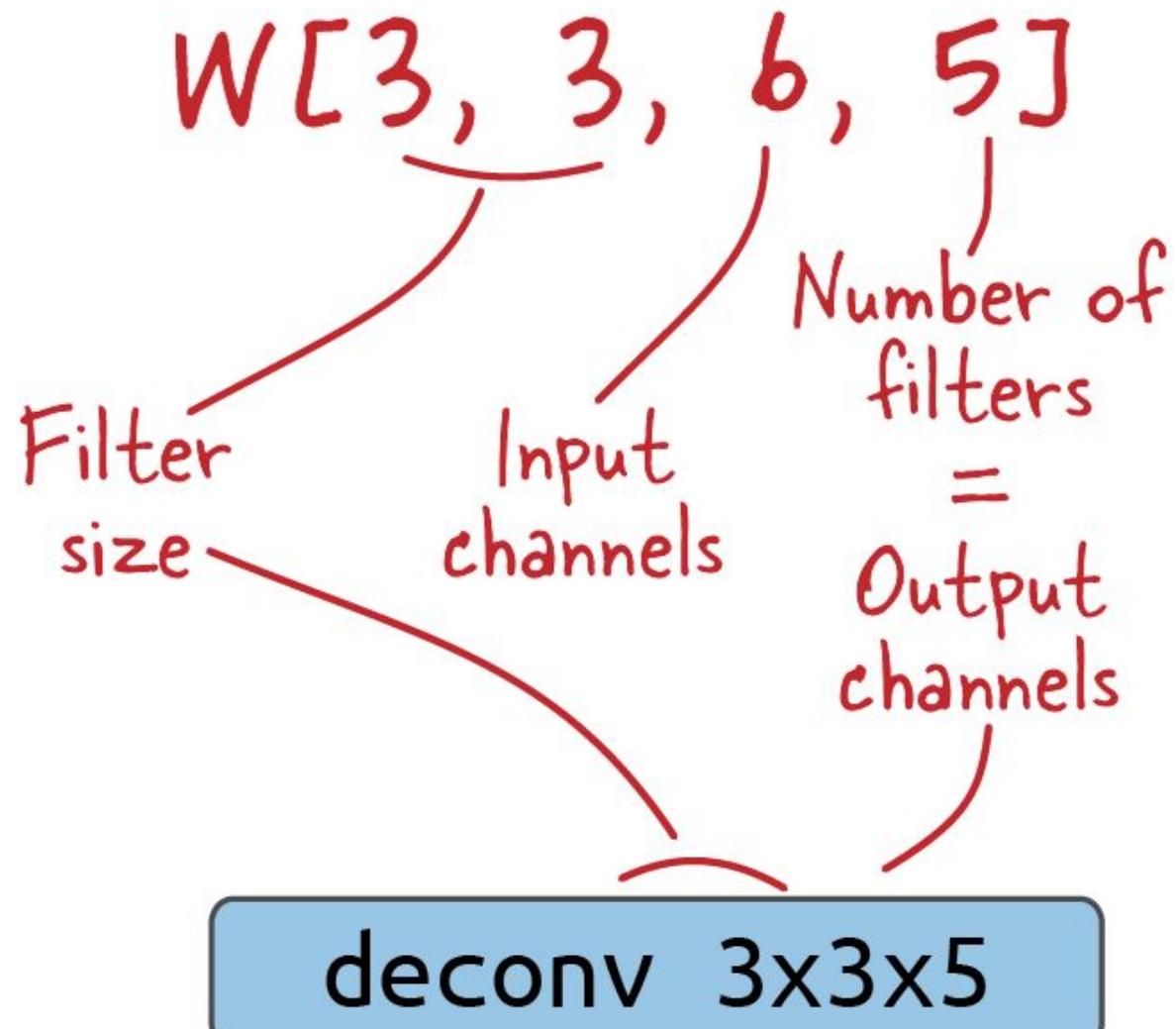
heads

Classification

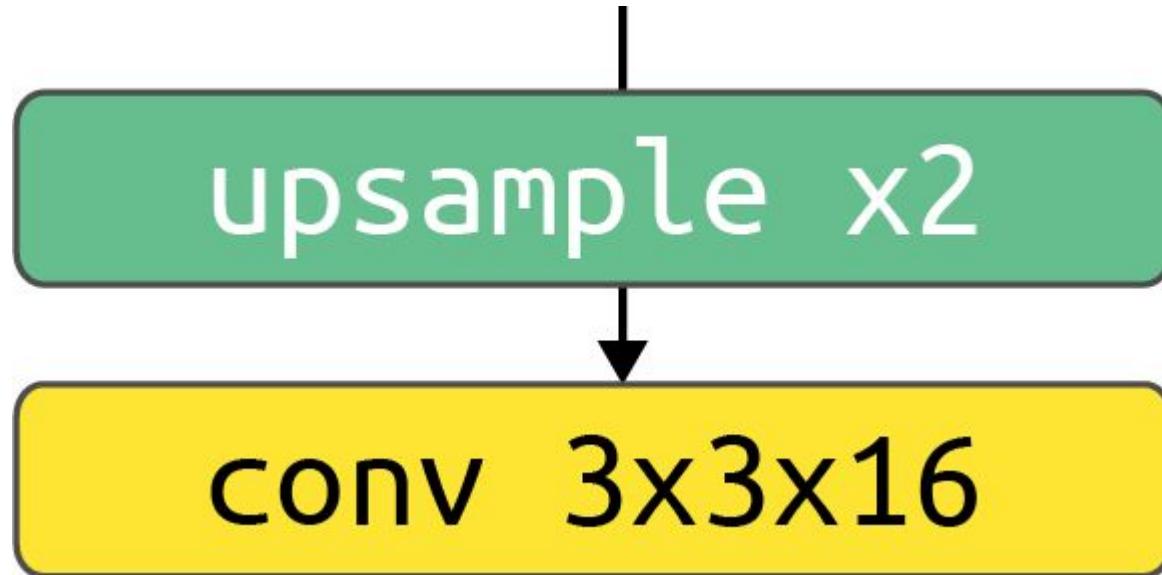
Box refinement

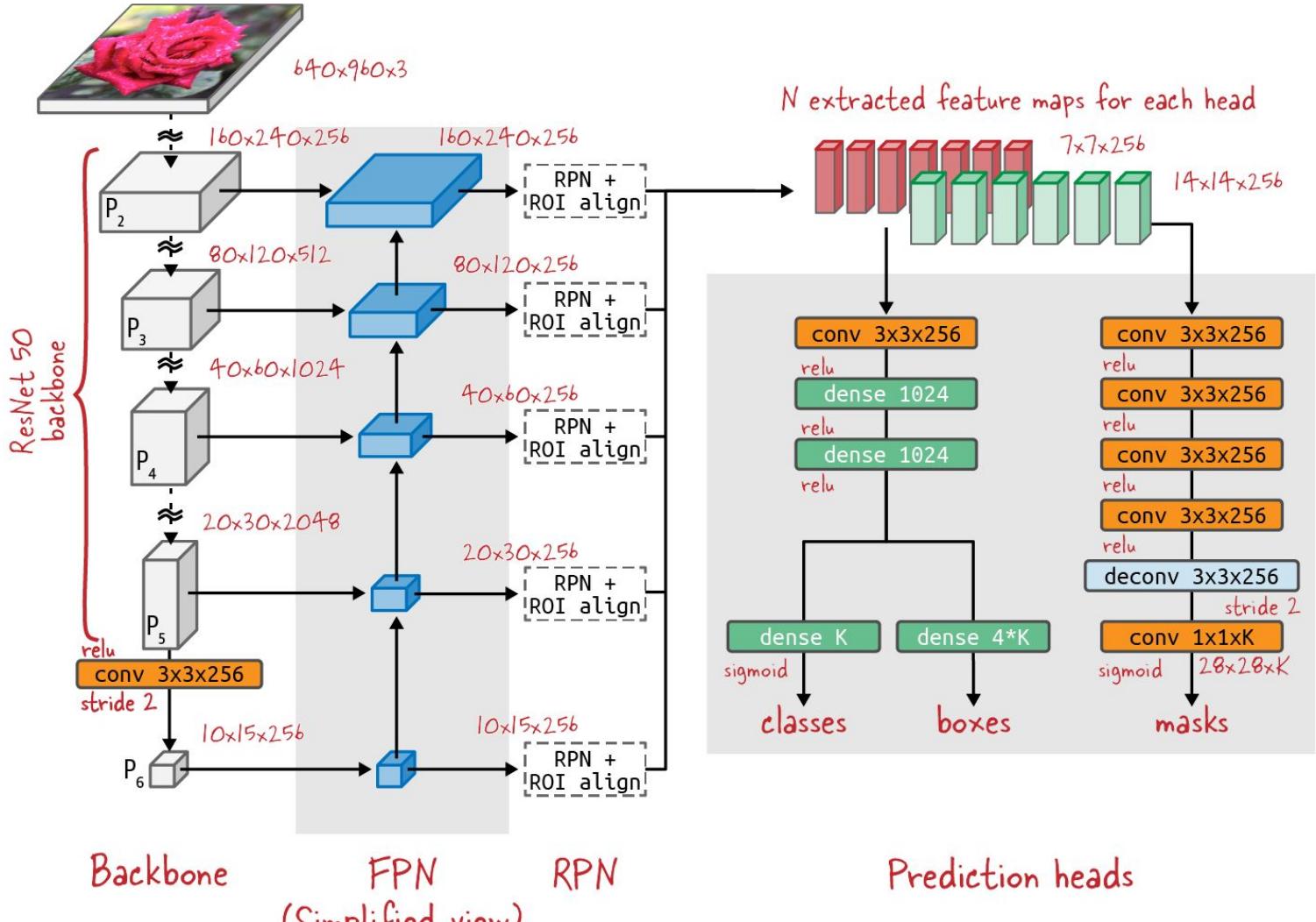
Mask generation





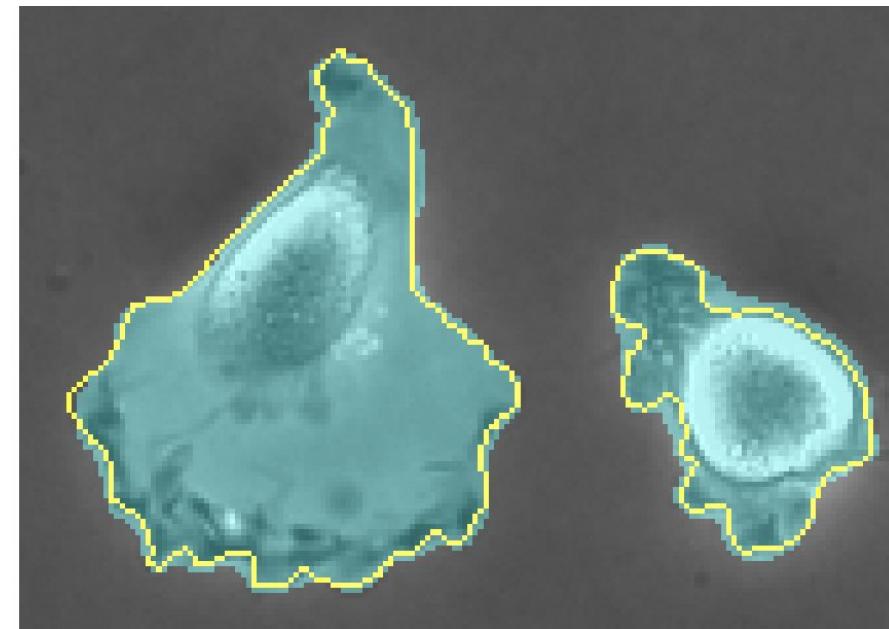
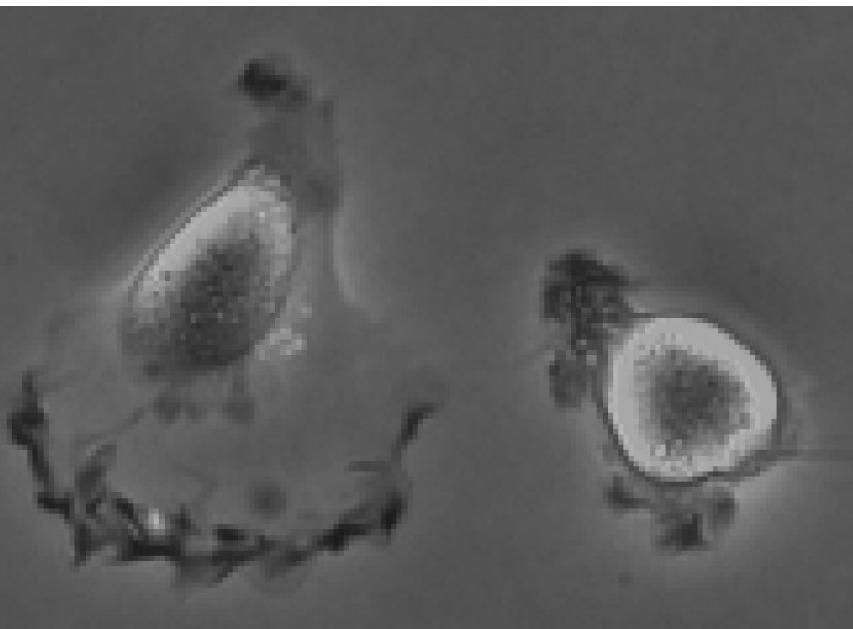
An “up-convolution” is a simple nearest neighbor upsampling operation followed by an ordinary convolutional layer.

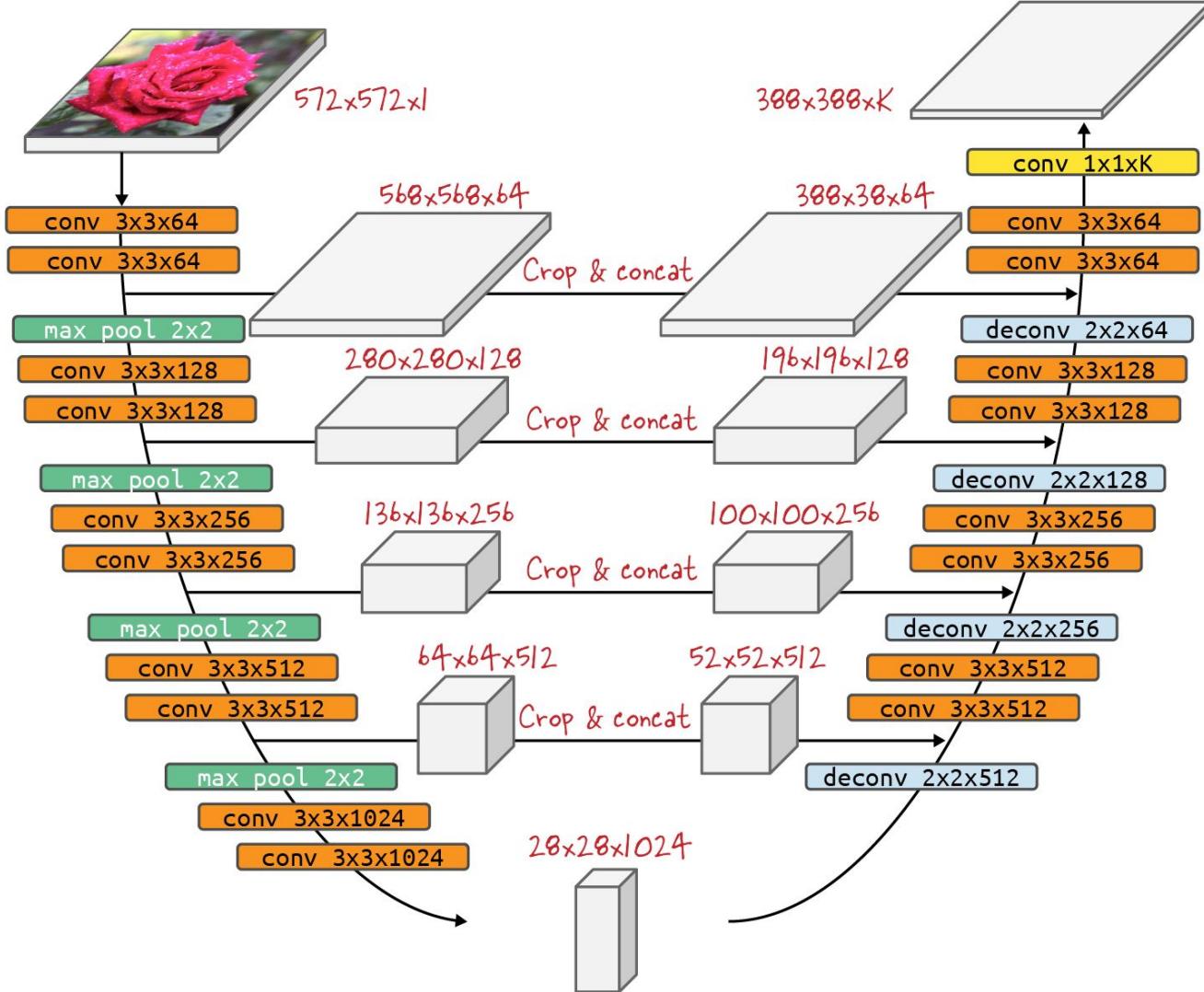


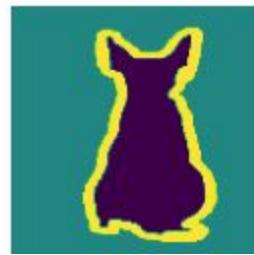
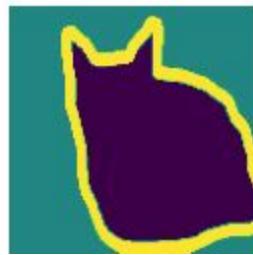
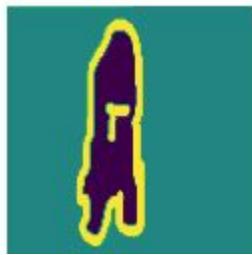
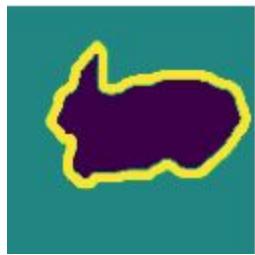


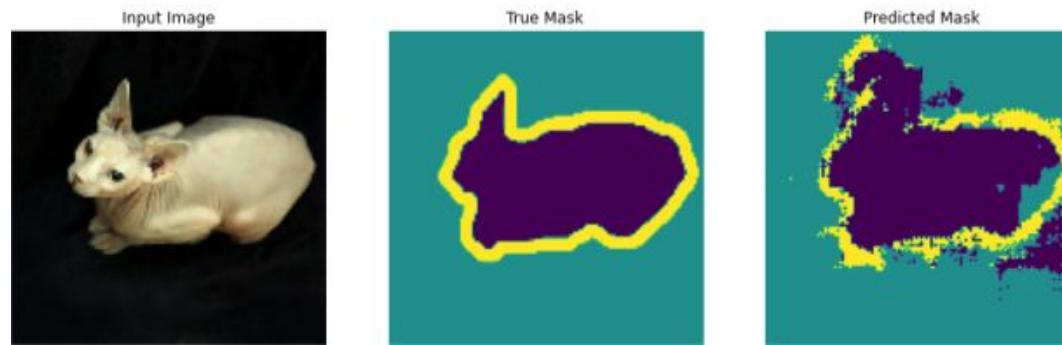
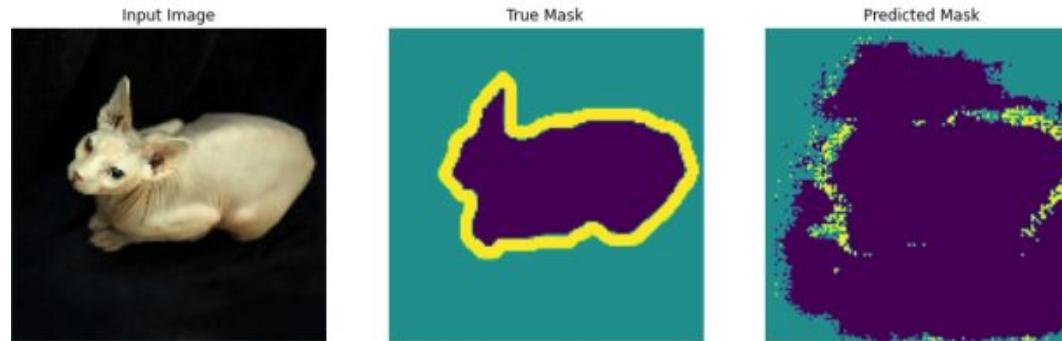
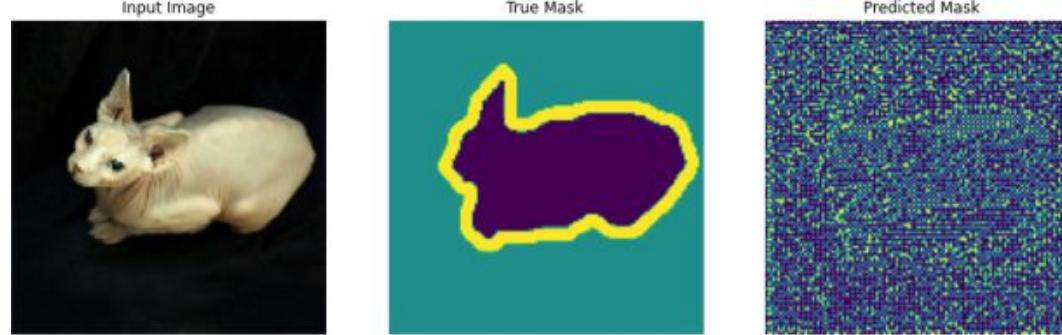


The U-Net architecture was designed to segment biomedical images such as these microscopy cell images.

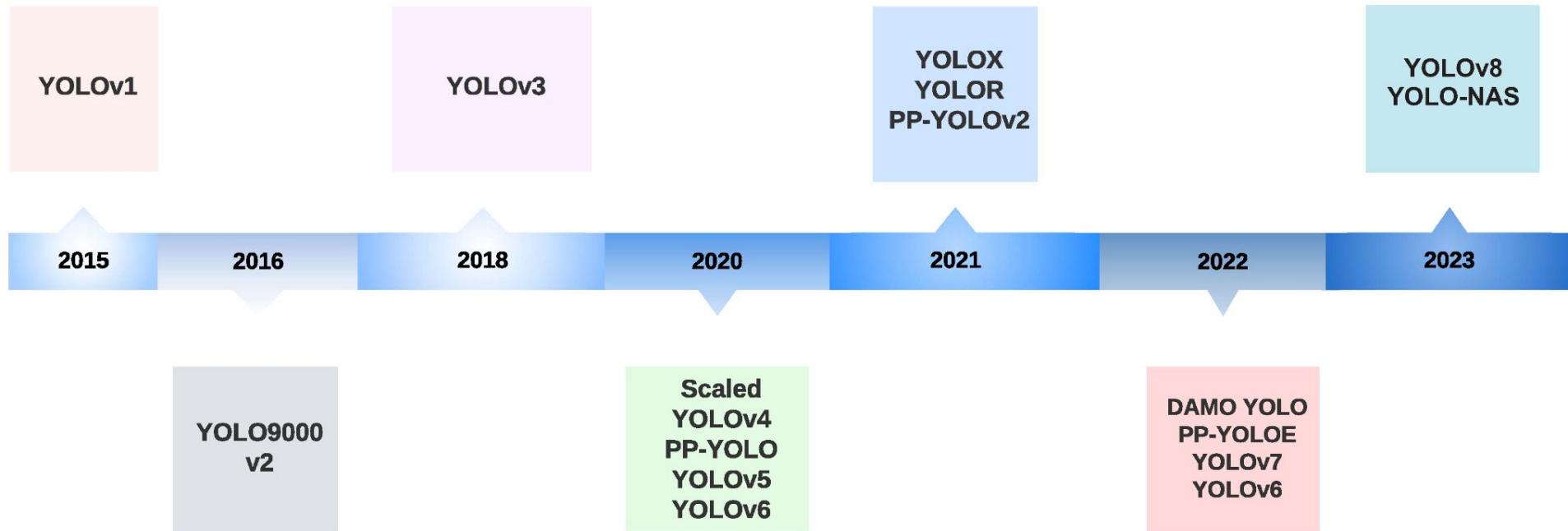




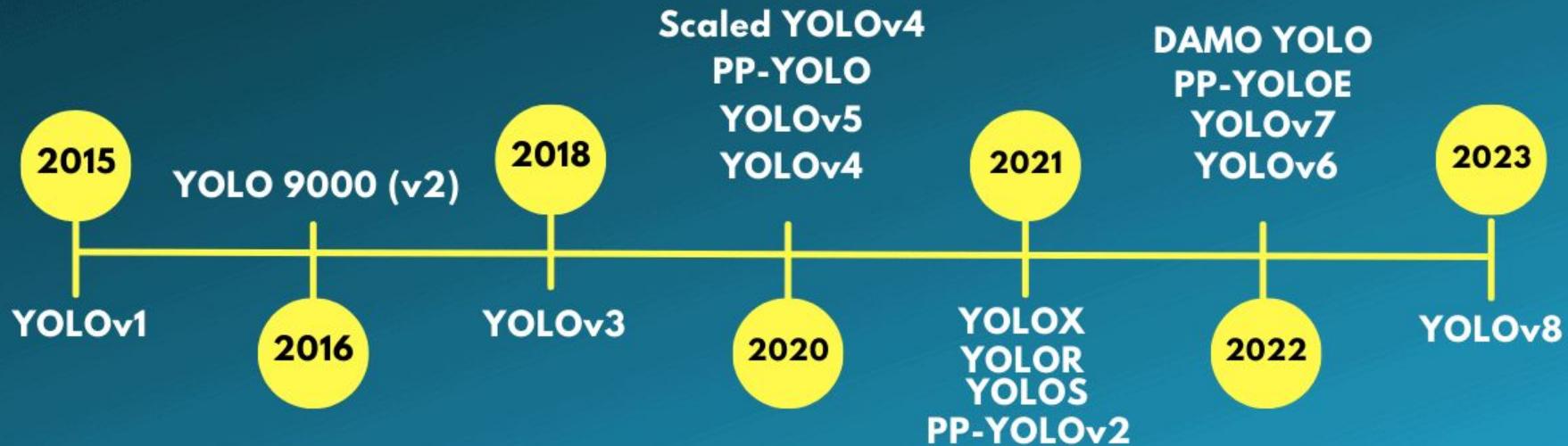


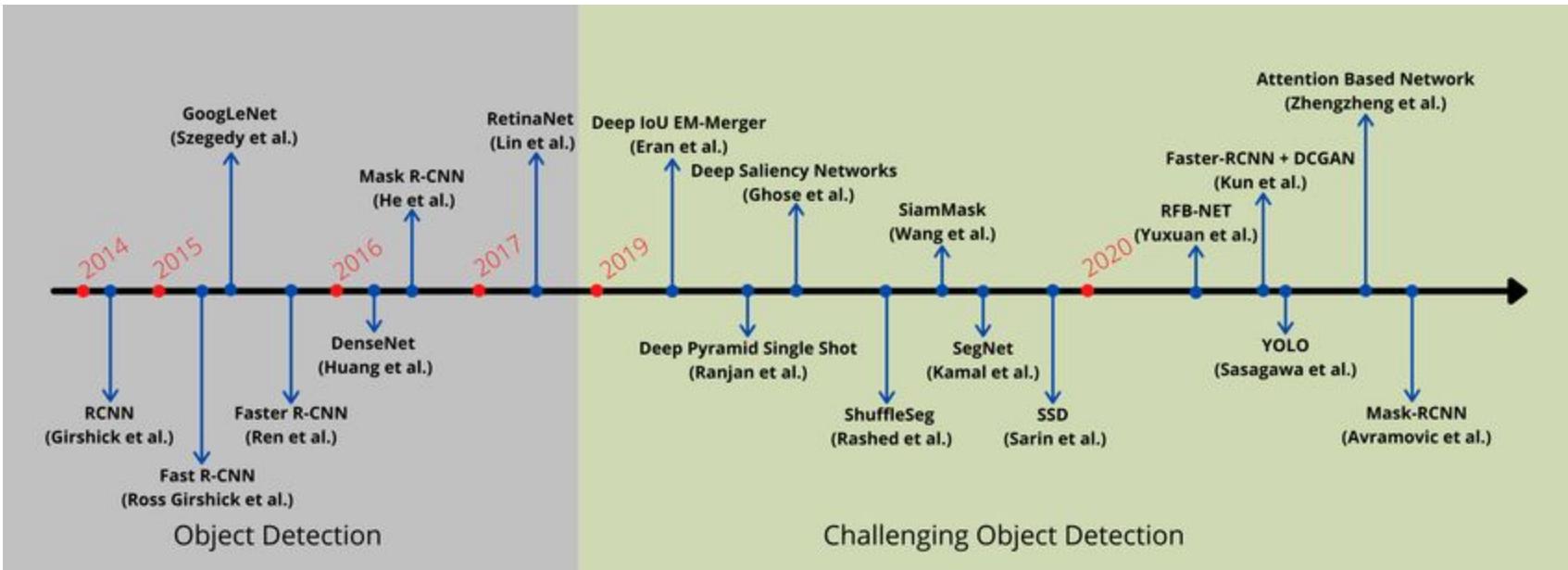


# YOLO



# YOLO Object Detection Models Timeline



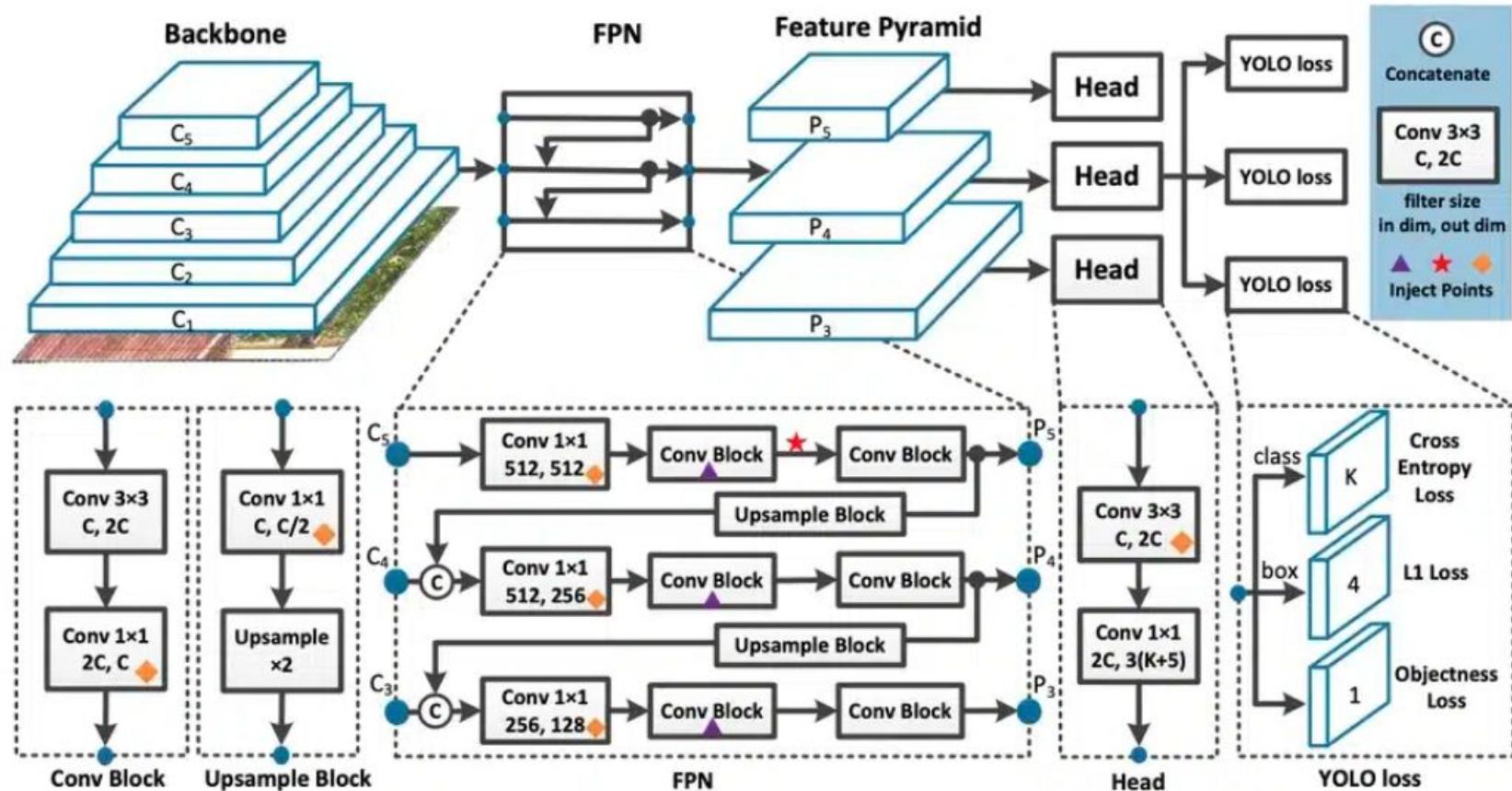




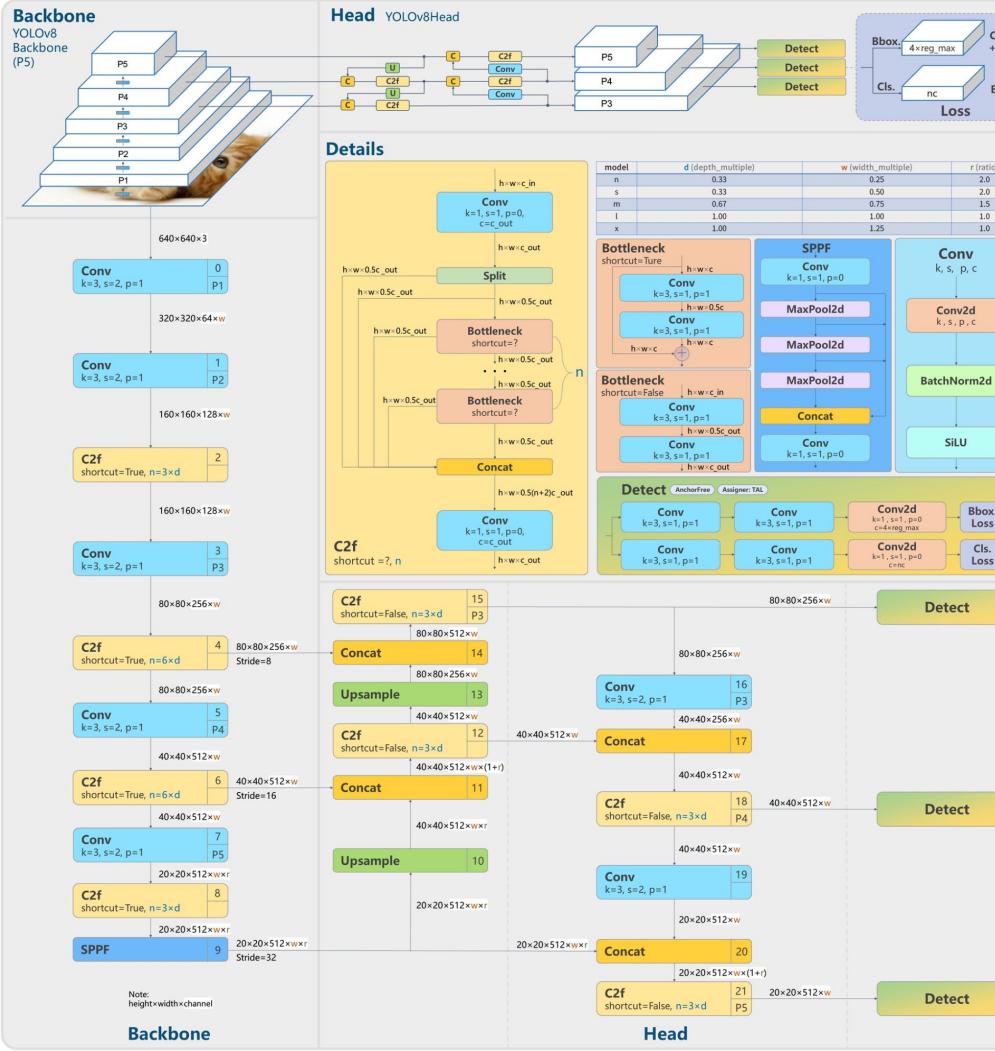
## Mean Average Precision (mAP) Using the COCO Evaluator

$$mAP = \frac{1}{|classes|} \sum_{c \in classes} \frac{|TP_c|}{|FP_c| + |TP_c|}$$

# YOLOv7



## YOLOv8



[Home](#)[Quickstart](#)[Modes](#)[Train](#)[Val](#)[Predict](#)[Export](#)[Track](#)[Benchmark](#)[Tasks](#)[Detect](#)[Segment](#)[Classify](#)[Pose](#)[OBB](#)[Models](#)[Datasets](#)[Guides](#)[NEW Explorer](#)[Languages](#)

English

# Home



## Table of contents

[Where to Start](#)[YOLO: A Brief History](#)[YOLO Licenses: How is Ultralytics YOLO licensed?](#)

Introducing Ultralytics YOLOv8, the latest version of the acclaimed real-time object detection and image segmentation model. YOLOv8 is built on cutting-edge advancements in deep learning and computer vision, offering unparalleled performance in terms of speed and accuracy. Its streamlined design makes it suitable for various applications and easily adaptable to different hardware platforms, from edge devices to cloud APIs.

Explore the YOLOv8 Docs, a comprehensive resource designed to help you understand and utilize its features

main

24 Branches 1 Tags

Go to file

Code

 Laughing-q and glenn-jocher ultralytics 8.1.20 add YOLOv8x-World sup...  59ed47c · 1 hour ago 937 Commits .github Model typehints Docker fix (#8306) last week docker ultralytics 8.1.18 add cmake for building onnxsim on a... last week docs ultralytics 8.1.20 add YOLOv8x-World support (#8539) 1 hour ago examples OBB Docs updates (#7568) 2 months ago tests Add missing single-line docstrings (#8362) last week ultralytics ultralytics 8.1.20 add YOLOv8x-World support (#8539) 1 hour ago .gitignore ultralytics 8.0.163 add new gpu-latest runner to CI ac... 7 months ago .pre-commit-config.yaml YAML reformat (#7669) 2 months ago CITATION.cff YOLOv8.1 blog, Explorer notebook and 2023 > 2024 upda... 2 months ago CONTRIBUTING.md ultralytics 8.0.224 Counting and Heatmaps updates (#... 3 months ago LICENSE Update LICENSE to AGPL-3.0 (#2031) last year README.md Add <https://youtu.be/3VryynorQeo> to README and fix con... 2 months ago README.zh-CN.md Add <https://youtu.be/3VryynorQeo> to README and fix con... 2 months ago mkdocs.yml Add TFLite Docs Integrations Page (#8522) 2 hours ago pyproject.toml Update mkdocs-ultralytics-plugin>=0.0.44 (#8347) last week

## About

NEW - YOLOv8 🎨 in PyTorch > ONNX > OpenVINO > CoreML > TFLite

[docs.ultralytics.com](#)

tracking machine-learning

deep-learning hub pytorch yolo

image-classification object-detection

obb pose instance-segmentation

yolov3 yolov5 ultralytics yolov8

yolo-world

 Readme

 AGPL-3.0 license

 Code of conduct

 Security policy

 Cite this repository

 Activity

 Custom properties

 20.1k stars

 131 watching

 4k forks

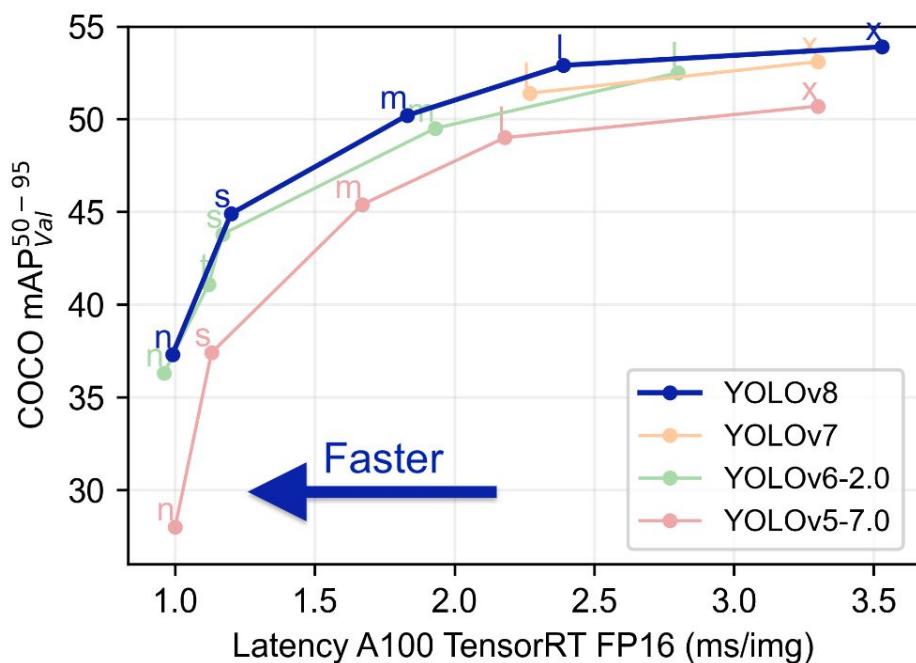
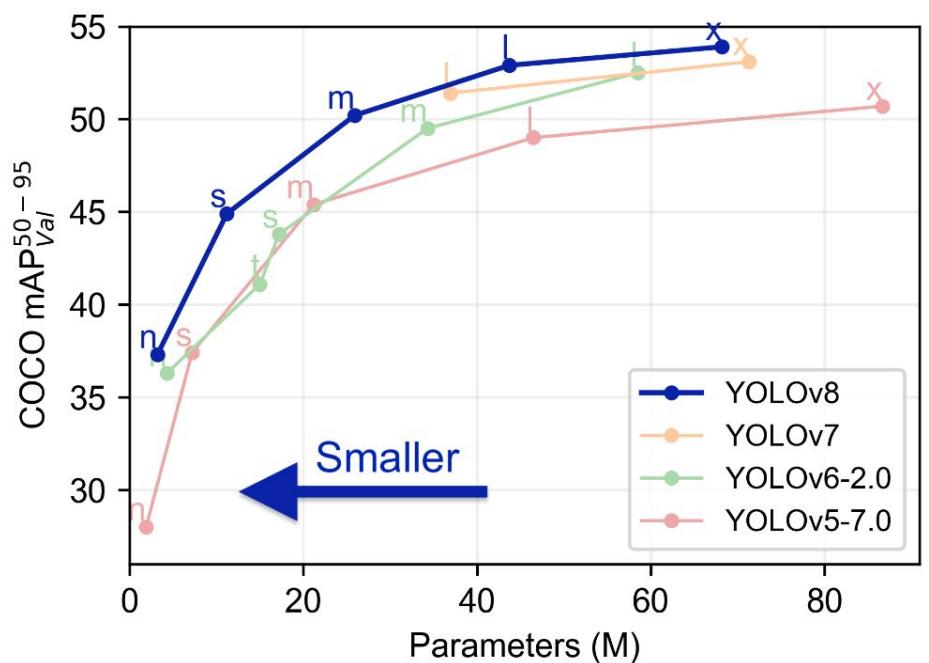
Report repository

## Releases

v8.1.0 - YOLOv8 Oriented Bo...

Latest

on Jan 10



## ▼ Detection (COCO)

See [Detection Docs](#) for usage examples with these models trained on [COCO](#), which include 80 pre-trained classes.

Model	size (pixels)	mAP <sup>val</sup> 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
<a href="#">YOLOv8n</a>	640	37.3	80.4	0.99	3.2	8.7
<a href="#">YOLOv8s</a>	640	44.9	128.4	1.20	11.2	28.6
<a href="#">YOLOv8m</a>	640	50.2	234.7	1.83	25.9	78.9
<a href="#">YOLOv8l</a>	640	52.9	375.2	2.39	43.7	165.2
<a href="#">YOLOv8x</a>	640	53.9	479.1	3.53	68.2	257.8

- mAP<sup>val</sup> values are for single-model single-scale on [COCO val2017](#) dataset.

Reproduce by `yolo val detect data=coco.yaml device=0`

- Speed averaged over COCO val images using an [Amazon EC2 P4d](#) instance.

Reproduce by `yolo val detect data=coco.yaml batch=1 device=0|cpu`

## ▼ Segmentation (COCO)

See [Segmentation Docs](#) for usage examples with these models trained on [COCO-Seg](#), which include 80 pre-trained classes.

Model	size (pixels)	mAP <sup>box</sup> 50-95	mAP <sup>mask</sup> 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	params (M)	FLOPs (B)
<a href="#">YOLOv8n-seg</a>	640	36.7	30.5	96.1	1.21	3.4	12.6
<a href="#">YOLOv8s-seg</a>	640	44.6	36.8	155.7	1.47	11.8	42.6
<a href="#">YOLOv8m-seg</a>	640	49.9	40.8	317.0	2.18	27.3	110.2
<a href="#">YOLOv8l-seg</a>	640	52.3	42.6	572.4	2.79	46.0	220.5
<a href="#">YOLOv8x-seg</a>	640	53.4	43.4	712.1	4.02	71.8	344.1

- mAP<sup>val</sup> values are for single-model single-scale on [COCO val2017](#) dataset.

Reproduce by `yolo val segment data=coco-seg.yaml device=0`

- Speed averaged over COCO val images using an [Amazon EC2 P4d](#) instance.

Reproduce by `yolo val segment data=coco-seg.yaml batch=1 device=0|cpu`

# Train

Train YOLOv8n on the COCO128 dataset for 100 epochs at image size 640. For a full list of available arguments see the [Configuration](#) page.



## Example

[Python](#)    [CLI](#)

```
from ultralytics import YOLO

# Load a model
model = YOLO('yolov8n.yaml') # build a new model from YAML
model = YOLO('yolov8n.pt') # load a pretrained model (recommended for training)
model = YOLO('yolov8n.yaml').load('yolov8n.pt') # build from YAML and transfer weights

# Train the model
results = model.train(data='coco128.yaml', epochs=100, imgsz=640)
```

# Val

Validate trained YOLOv8n model accuracy on the COCO128 dataset. No argument need to passed as the model retains it's training data and arguments as model attributes.

## Example

Python    CLI

```
from ultralytics import YOLO

# Load a model
model = YOLO('yolov8n.pt') # load an official model
model = YOLO('path/to/best.pt') # load a custom model

# Validate the model
metrics = model.val() # no arguments needed, dataset and settings remembered
metrics.box.map # map50-95
metrics.box.map50 # map50
metrics.box.map75 # map75
metrics.box.maps # a list contains map50-95 of each category
```

# Predict

Use a trained YOLOv8n model to run predictions on images.

## Example

**Python**

```
from ultralytics import YOLO

# Load a model
model = YOLO('yolov8n.pt') # load an official model
model = YOLO('path/to/best.pt') # load a custom model

# Predict with the model
results = model('https://ultralytics.com/images/bus.jpg') # predict on an image
```

See full predict mode details in the [Predict](#) page.

# Export

Export a YOLOv8n model to a different format like ONNX, CoreML, etc.

## Example

**Python**    **CLI**

```
from ultralytics import YOLO

# Load a model
model = YOLO('yolov8n.pt')  # load an official model
model = YOLO('path/to/best.pt')  # load a custom trained model

# Export the model
model.export(format='onnx')
```



# Export



Export mode is used for exporting a YOLOv8 model to a format that can be used for deployment. In this mode, the model is converted to a format that can be used by other software applications or hardware devices. This mode is useful when deploying the model to production environments.

**Export**

[Export to ONNX](#)   [Export to TensorRT](#)

---

Export an official YOLOv8n model to TensorRT on `device=0` for acceleration on CUDA devices.

```
from ultralytics import YOLO

model = YOLO('yolov8n.pt')
model.export(format='onnx', device=0)
```

## Export Examples



## Python

```
from ultralytics import YOLO

# Create a new YOLO model from scratch
model = YOLO('yolov8n.yaml')

# Load a pretrained YOLO model (recommended for training)
model = YOLO('yolov8n.pt')

# Train the model using the 'coco128.yaml' dataset for 3 epochs
results = model.train(data='coco128.yaml', epochs=3)

# Evaluate the model's performance on the validation set
results = model.val()

# Perform object detection on an image using the model
results = model('https://ultralytics.com/images/bus.jpg')

# Export the model to ONNX format
success = model.export(format='onnx')
```





## Train

From pretrained(recommended)

**From scratch**

Resume

```
from ultralytics import YOLO  
  
model = YOLO('yolov8n.yaml')  
results = model.train(data='coco128.yaml', epochs=5)
```



**Train Examples**

<https://github.com/ultralytics/JSON2YOLO>

 glenn-jocher	Add docstrings (#79)	✓	5013314 · last week	 98 Commits
 .github/workflows	Ultralytics Actions with OpenAI GPT-4 PR Summary (#77)	2 months ago		
 .gitignore	Update .gitignore	3 years ago		
 LICENSE	Update LICENSE to AGPL-3.0 (#43)	last year		
 README.md	Update Actions with Lychee and GitHub Token (#76)	2 months ago		
 general_json2yolo.py	Add docstrings (#79)	last week		
 labelbox_json2yolo.py	Add docstrings (#79)	last week		
 requirements.txt	Update requirements.txt	3 years ago		
 utils.py	Add docstrings (#79)	last week		

 README  Code of conduct  AGPL-3.0 license  Security



## Introduction

Welcome to the [COCO2YOLO](#) repository! This toolkit is designed to help you convert datasets in JSON format, following the COCO (Common Objects in Context) standards, into YOLO (You Only Look Once) format, which is widely recognized for its efficiency in real-time object detection tasks.

This process is essential for machine learning practitioners looking to train object detection models using the Darknet framework. Our code is flexible and can be utilized across various platforms including Linux, MacOS, and Windows.

# 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.

Classify



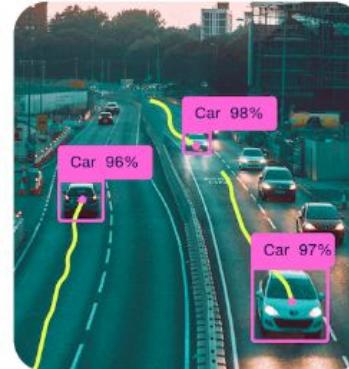
Detect



Segment



Track



Pose



All [Models](#) download automatically from the latest Ultralytics [release](#) on first use.

# Integrations

Our key integrations with leading AI platforms extend the functionality of Ultralytics' offerings, enhancing tasks like dataset labeling, training, visualization, and model management. Discover how Ultralytics, in collaboration with [Roboflow](#), [ClearML](#), [Comet](#), Neural Magic and [OpenVINO](#), can optimize your AI workflow.



Roboflow	ClearML ★ NEW	Comet ★ NEW	Neural Magic ★ NEW
Label and export your custom datasets directly to YOLOv8 for training with <a href="#">Roboflow</a>	Automatically track, visualize and even remotely train YOLOv8 using <a href="#">ClearML</a> (open-source!)	Free forever, <a href="#">Comet</a> lets you save YOLOv8 models, resume training, and interactively visualize and debug predictions	Run YOLOv8 inference up to 6x faster with <a href="#">Neural Magic DeepSparse</a>

## Models

YOLOv3  
YOLOv4  
YOLOv5  
YOLOv6  
YOLOv7  
YOLOv8  
YOLOv9  
SAM (Segment Anything Model)  
MobileSAM (Mobile Segment Anything Model)  
FastSAM (Fast Segment Anything Model)  
YOLO-NAS (Neural Architecture Search)  
RT-DETR (Realtime Detection Transformer)  
YOLO-World (Real-Time Open-Vocabulary Object Detection)

# Featured Models

Here are some of the key models supported:

1. **YOLOv3**: The third iteration of the YOLO model family, originally by Joseph Redmon, known for its efficient real-time object detection capabilities.
2. **YOLOv4**: A darknet-native update to YOLOv3, released by Alexey Bochkovskiy in 2020.
3. **YOLOv5**: An improved version of the YOLO architecture by Ultralytics, offering better performance and speed trade-offs compared to previous versions.
4. **YOLOv6**: Released by Meituan in 2022, and in use in many of the company's autonomous delivery robots.
5. **YOLOv7**: Updated YOLO models released in 2022 by the authors of YOLOv4.
6. **YOLOv8 NEW 🚀**: The latest version of the YOLO family, featuring enhanced capabilities such as instance segmentation, pose/keypoints estimation, and classification.
7. **YOLOv9**: An experimental model trained on the Ultralytics YOLOv5 codebase implementing Programmable Gradient Information (PGI).
8. **Segment Anything Model (SAM)**: Meta's Segment Anything Model (SAM).
9. **Mobile Segment Anything Model (MobileSAM)**: MobileSAM for mobile applications, by Kyung Hee University.
10. **Fast Segment Anything Model (FastSAM)**: FastSAM by Image & Video Analysis Group, Institute of Automation, Chinese Academy of Sciences.
11. **YOLO-NAS**: YOLO Neural Architecture Search (NAS) Models.
12. **Realtime Detection Transformers (RT-DETR)**: Baidu's PaddlePaddle Realtime Detection Transformer (RT-DETR) models.

## Files

main

Go to file

- ✓  cfg
- >  datasets
- ✓  models
- >  rt-detr
- >  v3
- >  v5
- >  v6
- ✓  v8
  - yolov8-cls-resnet101.yaml
  - yolov8-cls-resnet50.yaml
  - yolov8-cls.yaml
  - yolov8-ghost-p2.yaml
  - yolov8-ghost-p6.yaml
  - yolov8-ghost.yaml
  - yolov8-obb.yaml
  - yolov8-p2.yaml
  - yolov8-p6.yaml
  - yolov8-pose-p6.yaml
  - yolov8-pose.yaml
  - yolov8-rtdetr.yaml
  - yolov8-seg-p6.yaml
  - yolov8-seg.yaml
  - yolov8-world-t2i.yaml
  - yolov8-world.yaml
  - yolov8.yaml



glenn-jocher YAML reformat (#7669)



Code

Blame 46 lines (39 loc) · 1.84 KB

```

1  # Ultralytics YOLO 🚀, AGPL-3.0 license
2  # YOLOv8 object detection model with P3-P5 outputs. For Usage examples see https://docs.ultralytics.com/tasks/detect
3
4  # Parameters
5  nc: 80 # number of classes
6  scales: # model compound scaling constants, i.e. 'model=yolov8n.yaml' will
7    # [depth, width, max_channels]
8  n: [0.33, 0.25, 1024] # YOLOv8n summary: 225 layers, 3157200 parameters,
9  s: [0.33, 0.50, 1024] # YOLOv8s summary: 225 layers, 1116560 parameters,
10 m: [0.67, 0.75, 768] # YOLOv8m summary: 295 layers, 25902640 parameters,
11 l: [1.00, 1.00, 512] # YOLOv8l summary: 365 layers, 43691520 parameters,
12 x: [1.00, 1.25, 512] # YOLOv8x summary: 365 layers, 68229648 parameters,
13
14 # YOLOv8.0n backbone
15 backbone:
16  # [from, repeats, module, args]
17  - [-1, 1, Conv, [64, 3, 2]] # 0-P1/2
18  - [-1, 1, Conv, [128, 3, 2]] # 1-P2/4
19  - [-1, 3, C2f, [128, True]]
20  - [-1, 1, Conv, [256, 3, 2]] # 3-P3/8
21  - [-1, 6, C2f, [256, True]]
22  - [-1, 1, Conv, [512, 3, 2]] # 5-P4/16
23  - [-1, 6, C2f, [512, True]]
24  - [-1, 1, Conv, [1024, 3, 2]] # 7-P5/32
25  - [-1, 3, C2f, [1024, True]]
26  - [-1, 1, SPPF, [1024, 5]] # 9
27
28 # YOLOv8.0n head
29 head:
30  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
31  - [[-1, 6], 1, Concat, [1]] # cat backbone P4
32  - [-1, 3, C2f, [512]] # 12
33
34  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
35  - [[-1, 4], 1, Concat, [1]] # cat backbone P3
36  - [-1, 3, C2f, [256]] # 15 (P3/8-small)
37
38  - [-1, 1, Conv, [256, 3, 2]]
39  - [[-1, 12], 1, Concat, [1]] # cat head P4
40  - [-1, 3, C2f, [512]] # 18 (P4/16-medium)
41
42  - [-1, 1, Conv, [512, 3, 2]]
43  - [[-1, 9], 1, Concat, [1]] # cat head P5
44  - [-1, 3, C2f, [1024]] # 21 (P5/32-large)
45
46  - [[15, 18, 21], 1, Detect, [nc]] # Detect(P3, P4, P5) 112

```



## Files

main
cfg
datasets
Argoverse.yaml
DOTAv1.5.yaml
DOTAv1.yaml
GlobalWheat2020.yaml
ImageNet.yaml
Objects365.yaml
SKU-110K.yaml
VOC.yaml
VisDrone.yaml
carparts-seg.yaml
coco-pose.yaml
coco.yaml
coco128-seg.yaml
coco128.yaml
coco8-pose.yaml
coco8-seg.yaml
coco8.yaml
crack-seg.yaml
dota8.yaml
open-images-v7.yaml
package-seg.yaml
tiger-pose.yaml
xView.yaml
models

## ultralytics / ultralytics / cfg / datasets /

3 people ultralytics 8.1.6 revert 8.0.206 box ops box scaling (#7823) ef141af · 2 months ago

113

History

Name	Last commit message	Last commit date
...		
Argoverse.yaml	ultralytics 8.1.4 RTDETR TensorBoard graph visualization fix (#7725)	2 months ago
DOTAv1.5.yaml	YAML reformat (#7669)	2 months ago
DOTAv1.yaml	YAML reformat (#7669)	2 months ago
GlobalWheat2020.yaml	YAML reformat (#7669)	2 months ago
ImageNet.yaml	YAML reformat (#7669)	2 months ago
Objects365.yaml	YAML reformat (#7669)	2 months ago
SKU-110K.yaml	YAML reformat (#7669)	2 months ago
VOC.yaml	YAML reformat (#7669)	2 months ago
VisDrone.yaml	YAML reformat (#7669)	2 months ago
carparts-seg.yaml	ultralytics 8.1.6 revert 8.0.206 box ops box scaling (#7823)	2 months ago
coco-pose.yaml	YAML reformat (#7669)	2 months ago
coco.yaml	YAML reformat (#7669)	2 months ago
coco128-seg.yaml	YAML reformat (#7669)	2 months ago
coco128.yaml	YAML reformat (#7669)	2 months ago
coco8-pose.yaml	YAML reformat (#7669)	2 months ago
coco8-seg.yaml	YAML reformat (#7669)	2 months ago
coco8.yaml	YAML reformat (#7669)	2 months ago
crack-seg.yaml	YAML reformat (#7669)	2 months ago
dota8.yaml	YAML reformat (#7669)	2 months ago
open-images-v7.yaml	YAML reformat (#7669)	2 months ago
package-seg.yaml	YAML reformat (#7669)	2 months ago
tiger-pose.yaml	YAML reformat (#7669)	2 months ago
xView.yaml	ultralytics 8.1.6 revert 8.0.206 box ops box scaling (#7823)	2 months ago
crack-seg.yaml		2 months ago

# Ultralytics HUB

Experience seamless AI with [Ultralytics HUB](#) ★, the all-in-one solution for data visualization, YOLOv5 and YOLOv8 🚀 model training and deployment, without any coding. Transform images into actionable insights and bring your AI visions to life with ease using our cutting-edge platform and user-friendly [Ultralytics App](#). Start your journey for Free now!



## Contribute

We love your input! YOLOv5 and YOLOv8 would not be possible without help from our community. Please see our [Contributing Guide](#) to get started, and fill out our [Survey](#) to send us feedback on your experience. Thank you 🙏 to all our contributors!

