3099704: AI for Digital Health

Pretrained Models & Object Detection

Prof. Peerapon Vateekul, Ph.D.
Peerapon.v@chula.ac.th

Outline

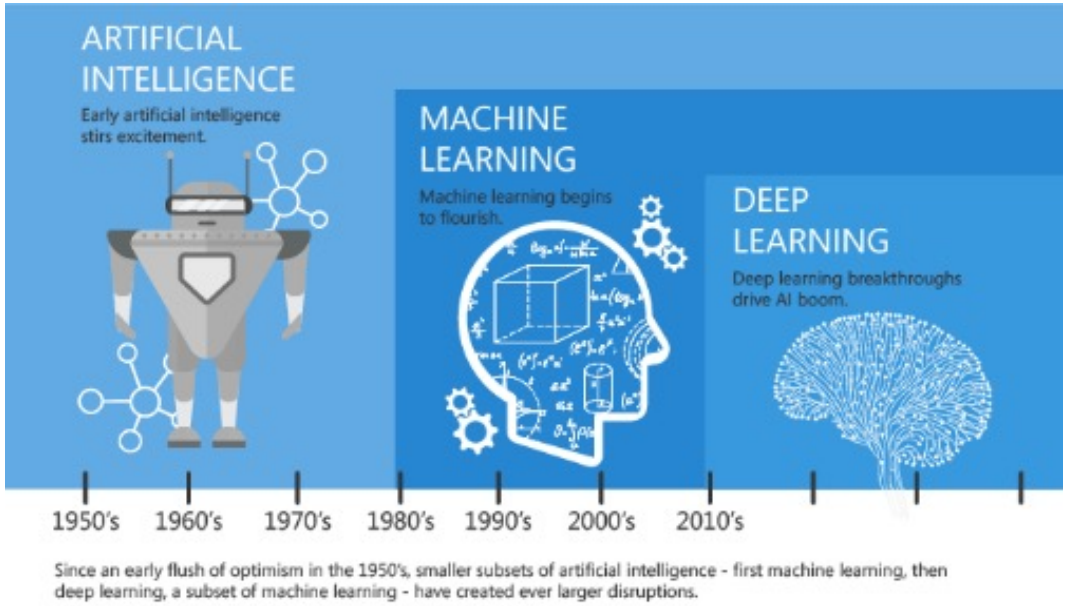
- DL & Image classification (recap)
- Pretrained models
- Object detection

DL & Image classification (recap)

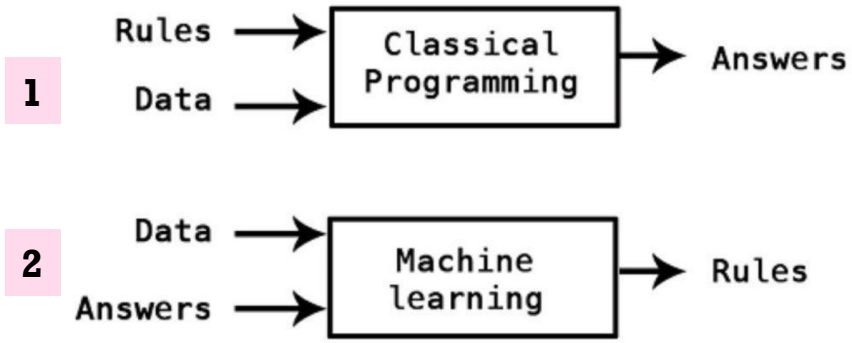
CHULA ENGINEERING COMPUTER
Foundation toward Innovation

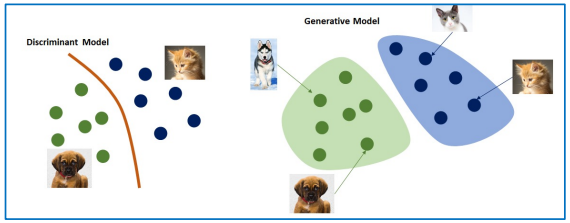
AI = Automation

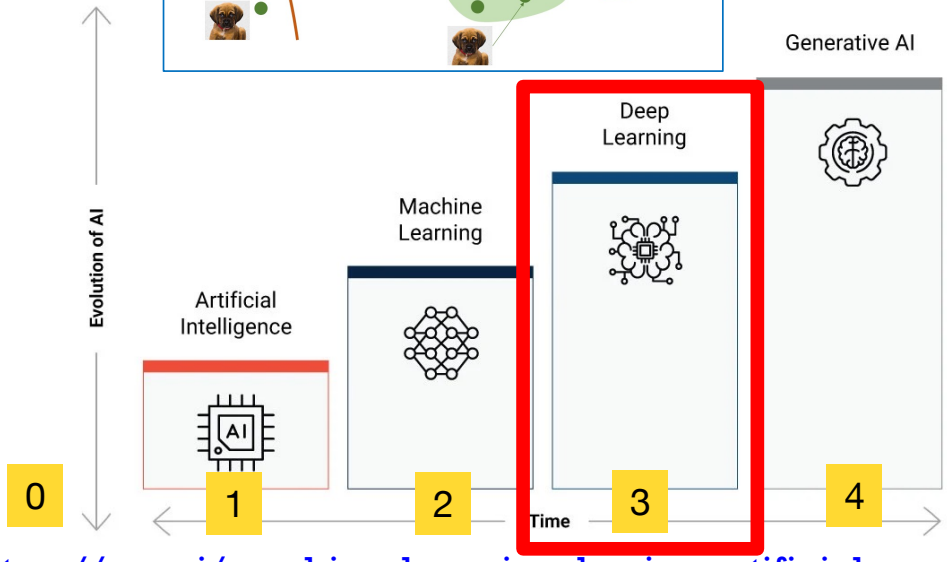
- 0) Not AI Solution (not automatic)
- 1) Rule-based AI
- 2) Machine Learning (ML)



Since an early flush of optimism in the 1950's, smaller subsets of artificial intelligence - first machine learning, then deep learning, a subset of machine learning - have created ever larger disruptions.

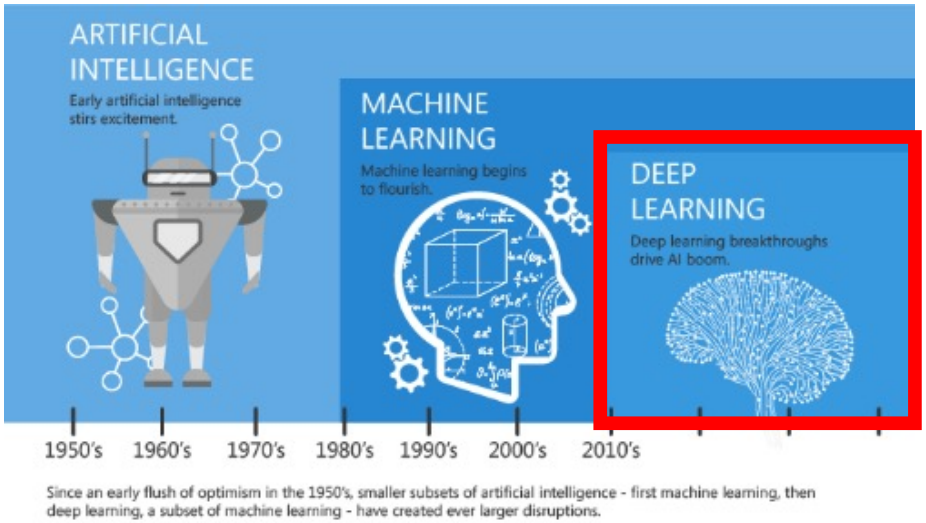




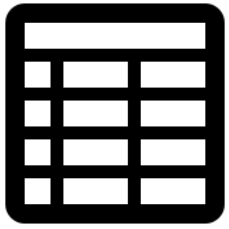


https://mc.ai/machine-learning-basics-artificial-intelligence-machine-learning-and-deep-learning/

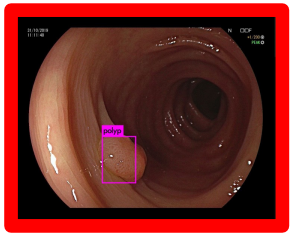
Arise of Deep Learning

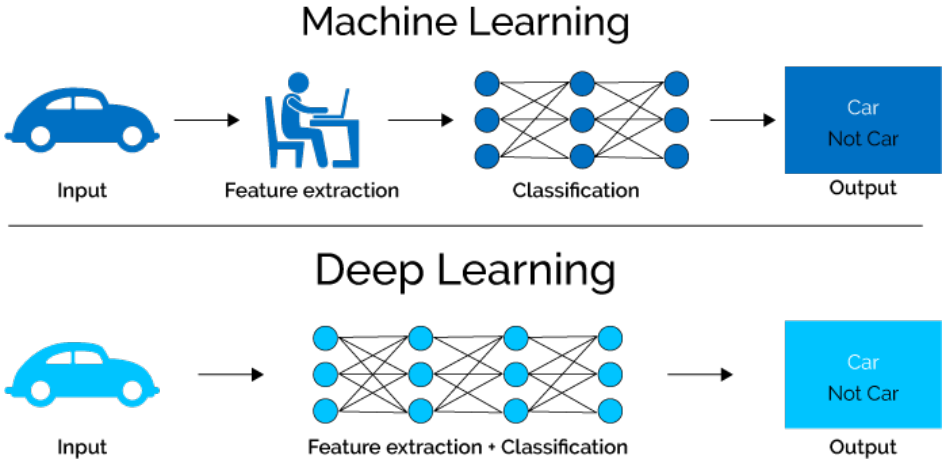


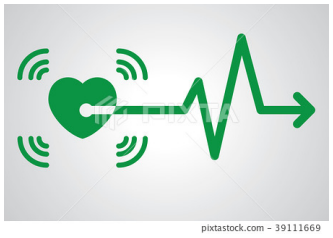
Since an early flush of optimism in the 1950's, smaller subsets of artificial intelligence - first machine learning, then deep learning, a subset of machine learning - have created ever larger disruptions.

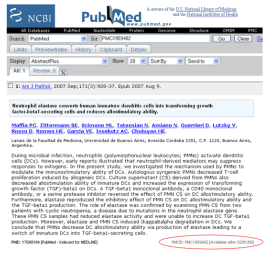


VS







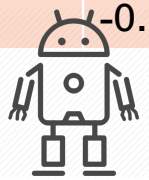


Image, video signals, voice NLP

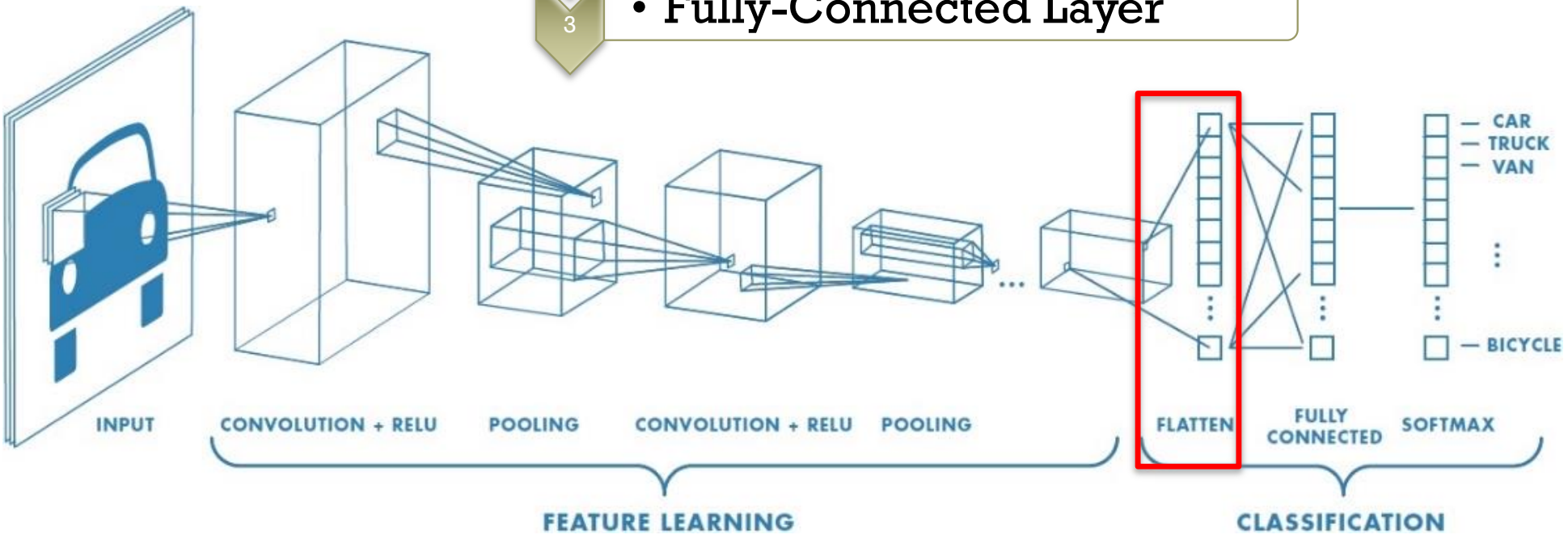
Convolutional Neural Networks

Embedding Vector

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

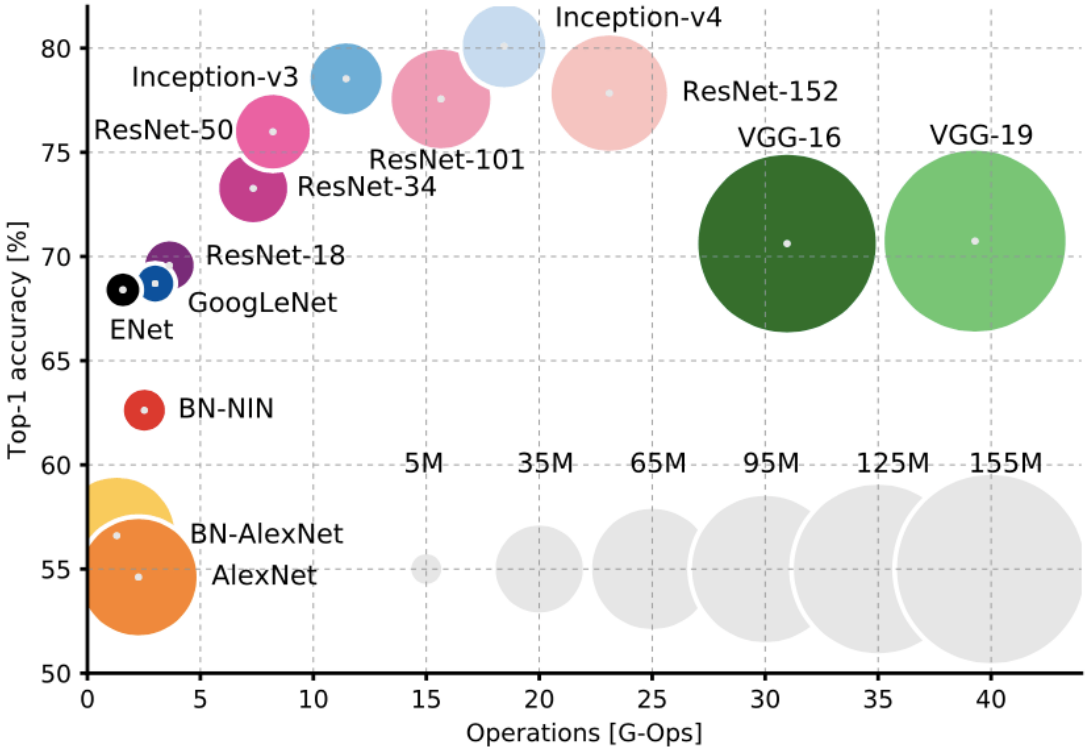


1. • Convolutional Layer
2. • Pooling Layer
3. • Fully-Connected Layer

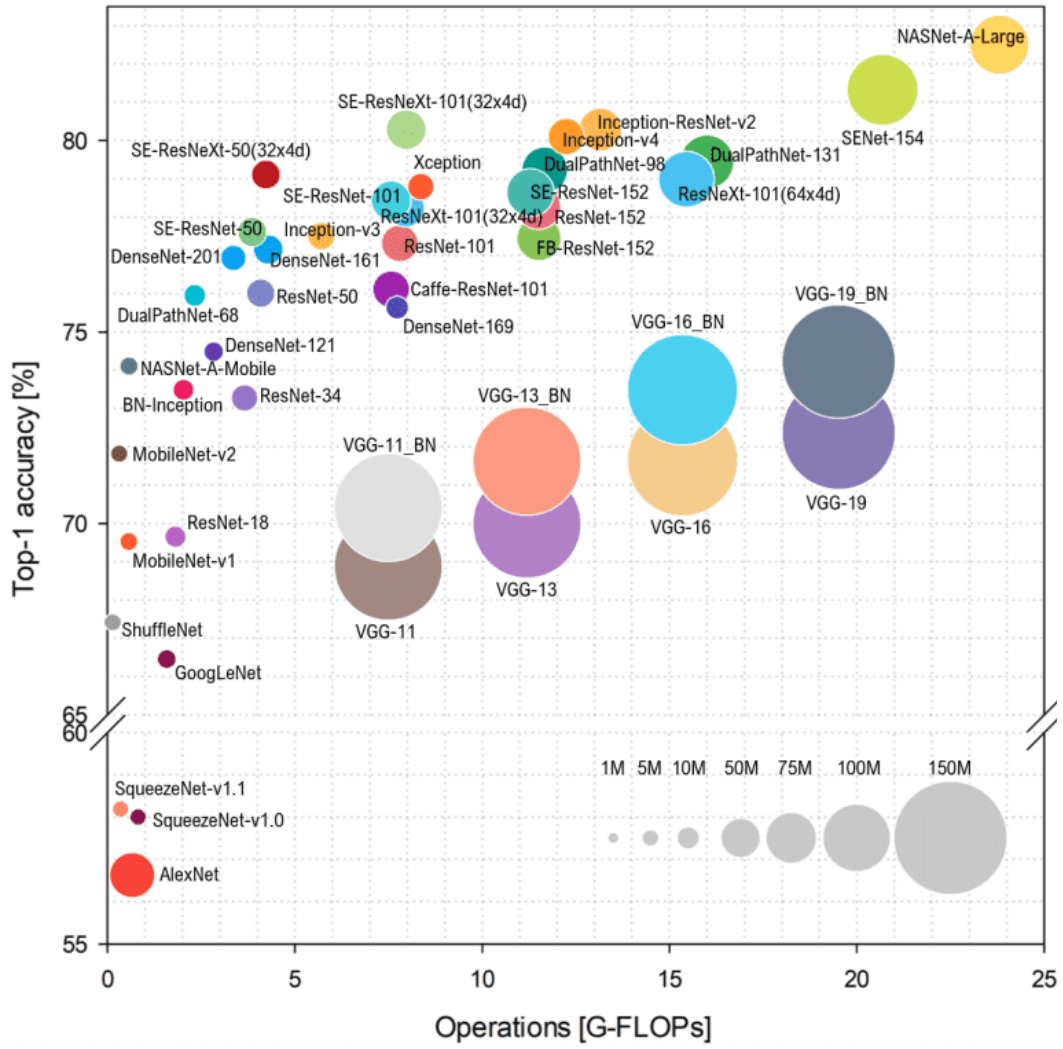


SOTA of Image Classification

Millions of parameters!



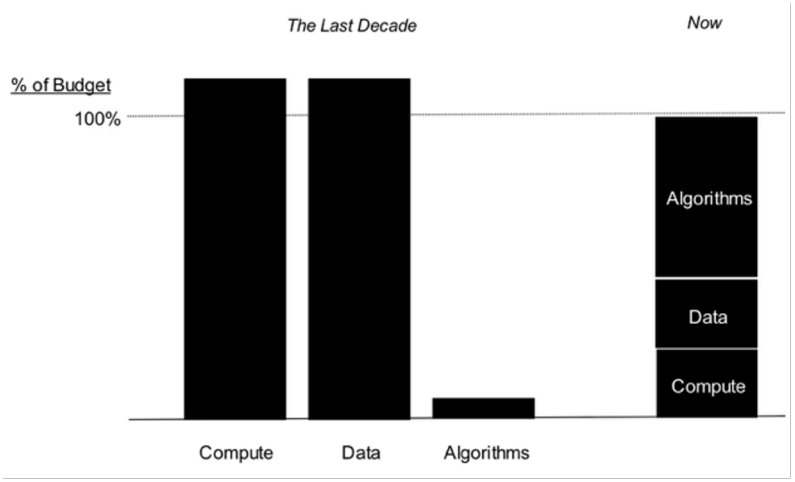
https://blog.csdn.net/qq_34216467/article/details/83061692

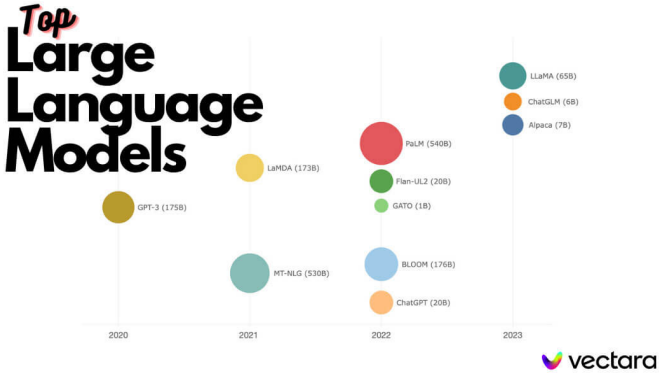


https://theaisummer.com/cnn-architectures/

Why now?

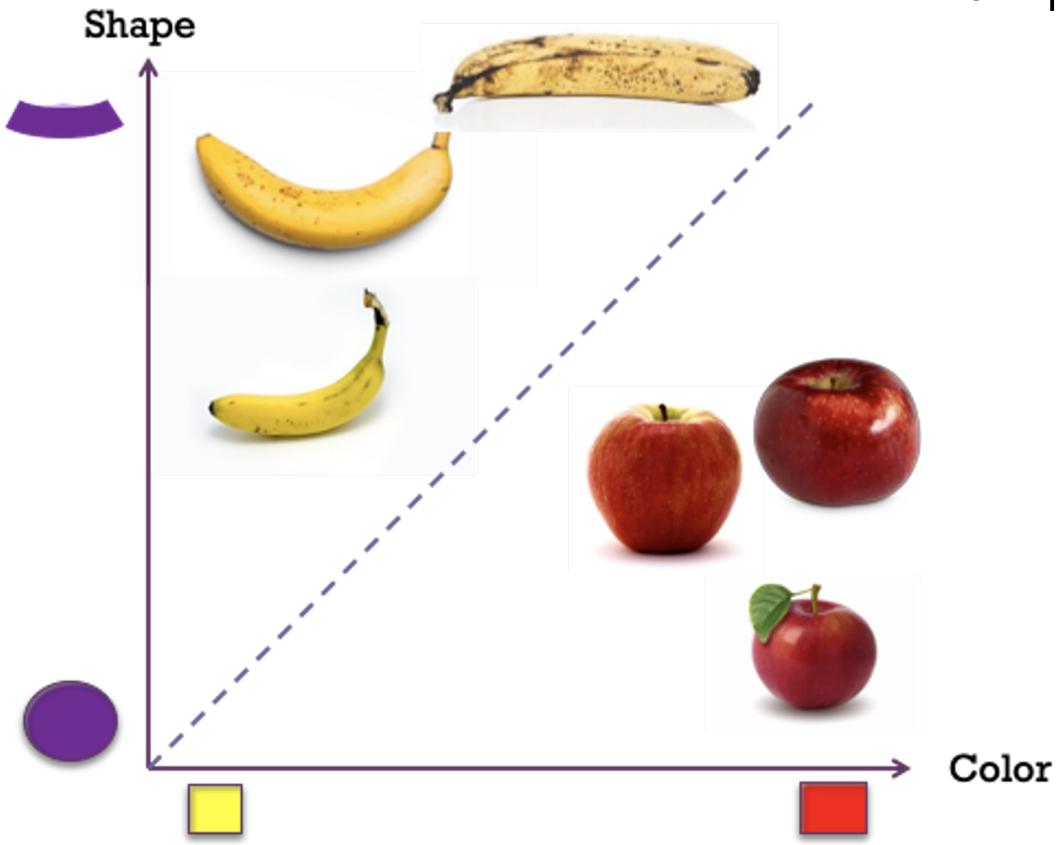
- Neural Networks has been around since 1990s
- 1) Big data – DNN can take advantage of large amounts of data better than other models
- 2) GPU – Enable training bigger models possible
- 3) Deep (pretrained models) – Easier to avoid bad local minima when the model is large





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

We need enough training data to infer the data pattern and to create model

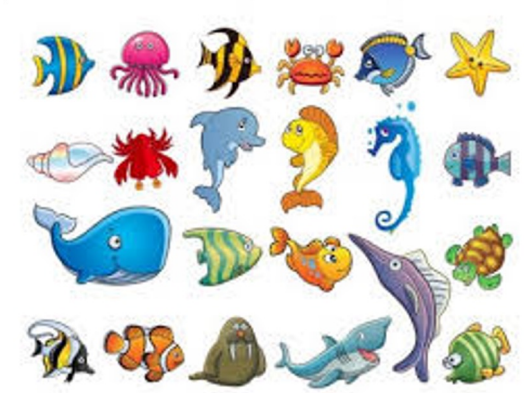


A simple problem requires less training data.

Pretrained models

Have you ever seen this creature before?
Can you guess whether it is a land animal or a water animal?

Have you ever seen this creature before?
Can you guess whether it is a land animal or a water animal?
You can transfer your knowledge from the past



Transfer learning

Myth: you can't do deep learning unless you have a million labelled examples for your problem.

Reality:

- You can **transfer** learned representations from **a related task**
- You can train on a nearby **surrogate objective** for which it is easy to generate labels

Transfer learning: idea

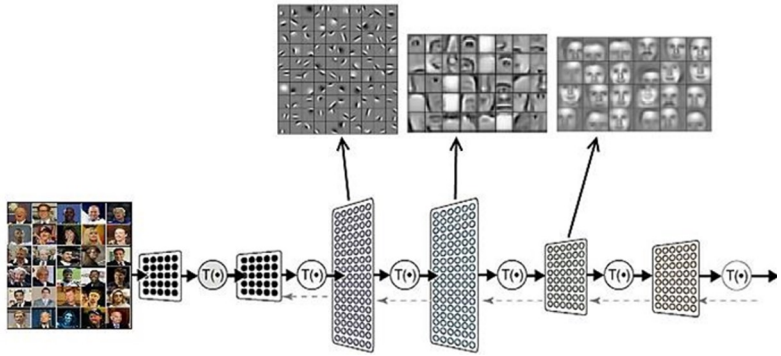
Instead of training a deep network from scratch for your task, you can

- **Take** a network trained on a different domain (data) for a different source task (e.g., LM)
- **Adapt** it for your domain (data) and your target task (e.g., classification)

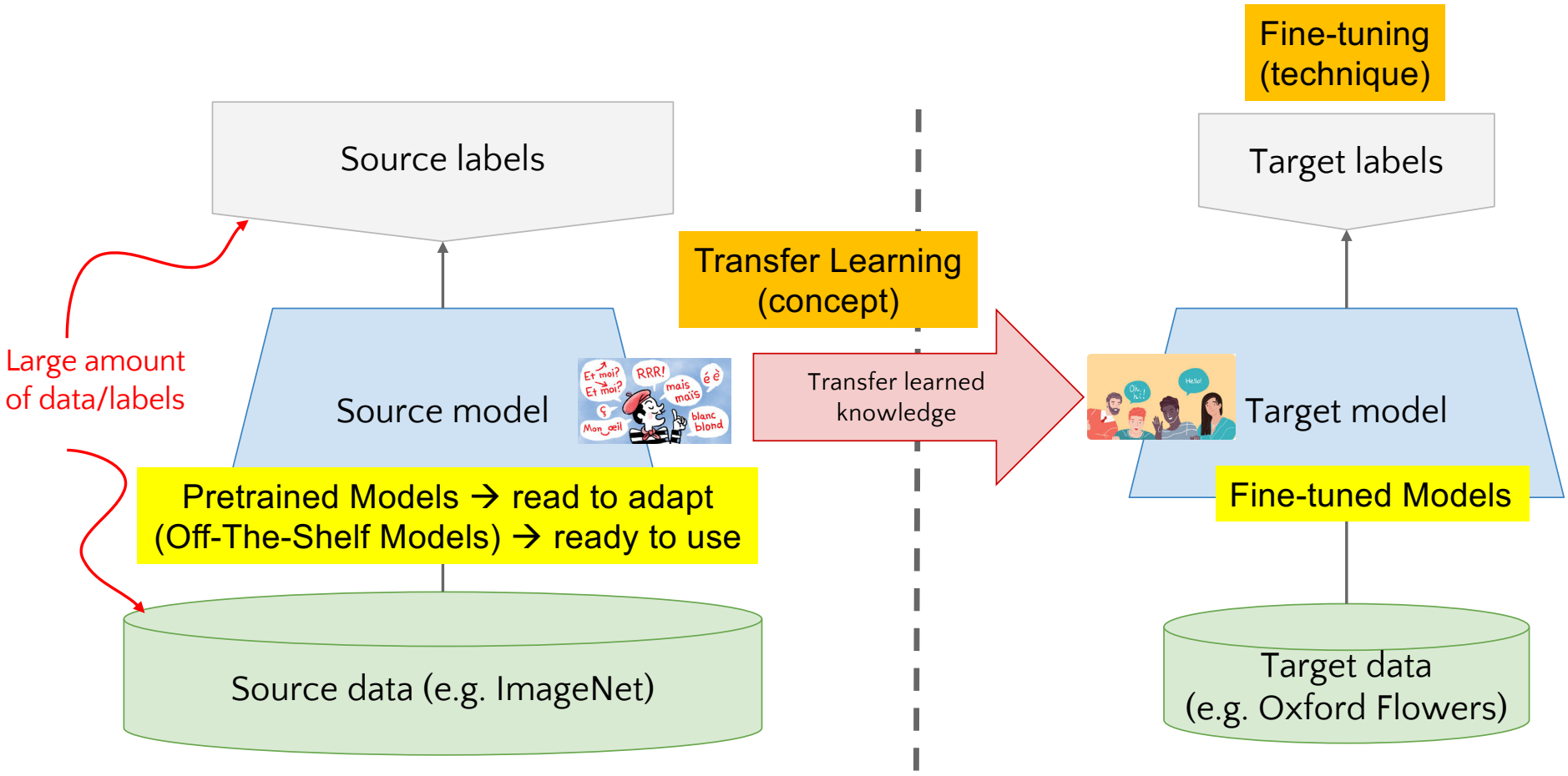
This lecture will talk about how to do this.

Variations:

- Different domain (data), same task
- Different domain (data), different task



Transfer learning: idea



http://www.image-net.org/

IMAGENET

14,197,122 images, 21841 synsets indexed

Explore Download Challenges Publications CoolStuff About

Not logged in. Login | Signup

Object recognition: 14M++ images on 20K++ categories

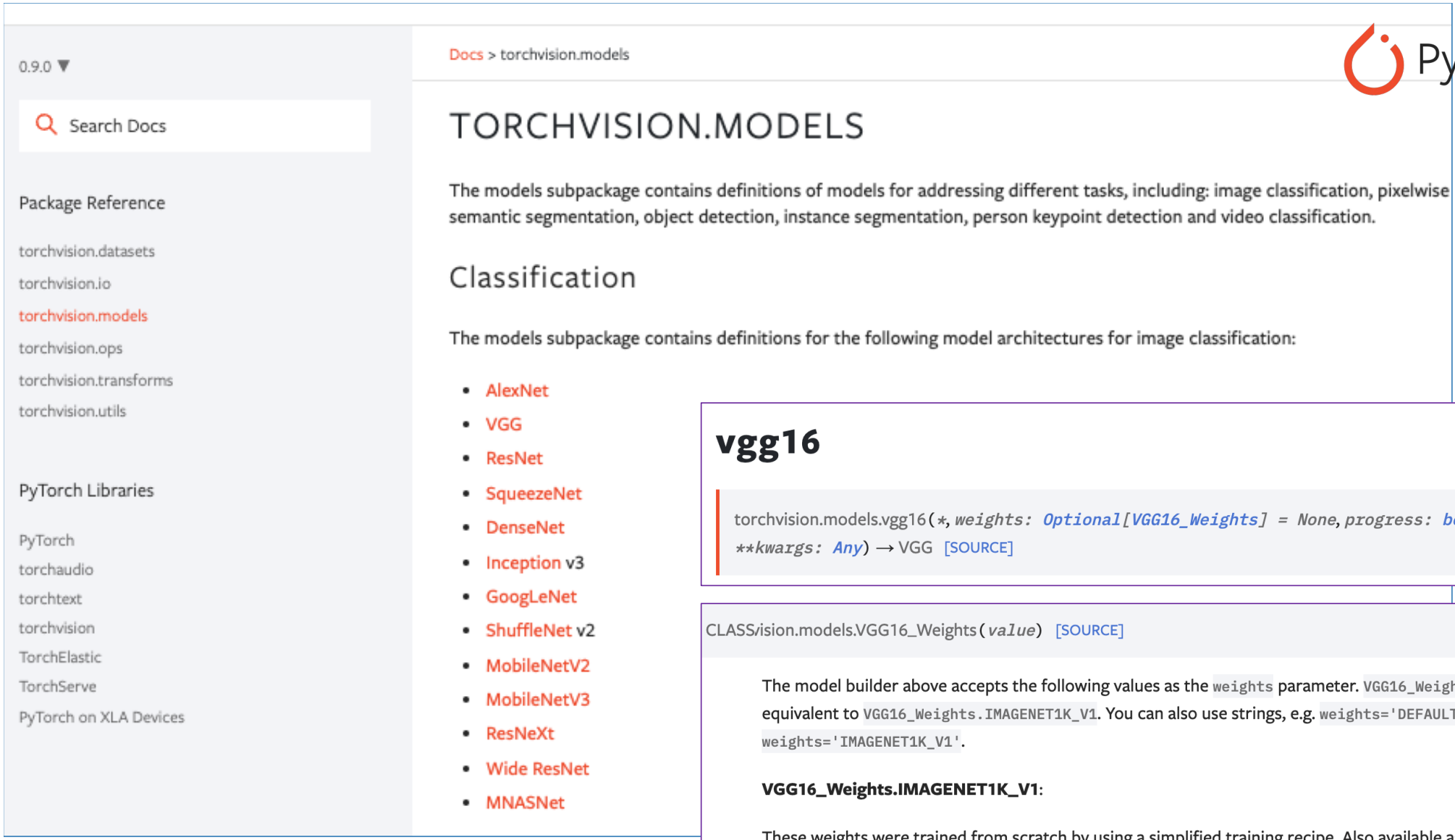
ImageNet is an image database organized according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by hundreds and thousands of images. Currently we have an average of over five hundred images per node. We hope ImageNet will become a useful resource for researchers, educators, students and all of you who share our passion for pictures.

Click here to learn more about ImageNet, Click here to join the ImageNet mailing list.

ImageNet

From Wikipedia, the free encyclopedia

The **ImageNet** project is a large visual database designed for use in visual object recognition software research. More than 14 million[1][2] images have been hand-annotated by the project to indicate what objects are pictured and in at least one million of the images, bounding boxes are also provided.[3] ImageNet contains more than 20,000 categories[2] with a typical category, such as "balloon" or "strawberry", consisting of several hundred images.[4] The database of annotations of third-party



PyTorch

vgg16

torchvision.models.vgg16(*, *weights: Optional[VGG16_Weights] = None*, *progress: bool = True*, ***kwargs: Any*) → VGG [SOURCE]

CLASS*v*ision.models.VGG16_Weights(*value*) [SOURCE]

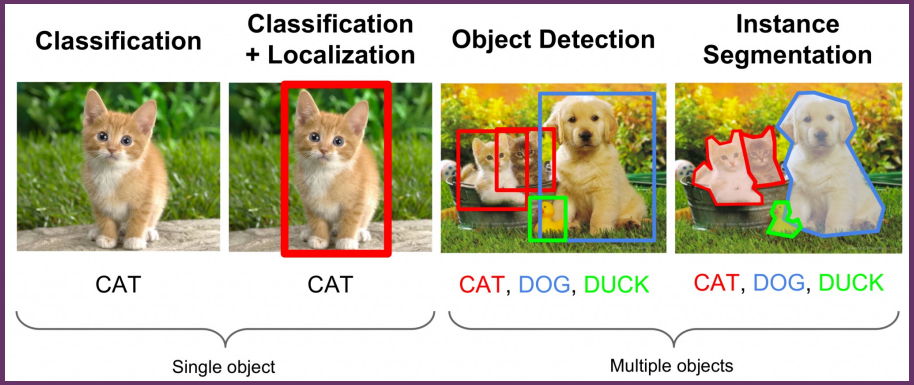
The model builder above accepts the following values as the `weights` parameter. `VGG16_Weights.DEFAULT` is equivalent to `VGG16_Weights.IMAGENET1K_V1`. You can also use strings, e.g. `weights='DEFAULT'` or `weights='IMAGENET1K_V1'`.

VGG16_Weights.IMAGENET1K_V1:

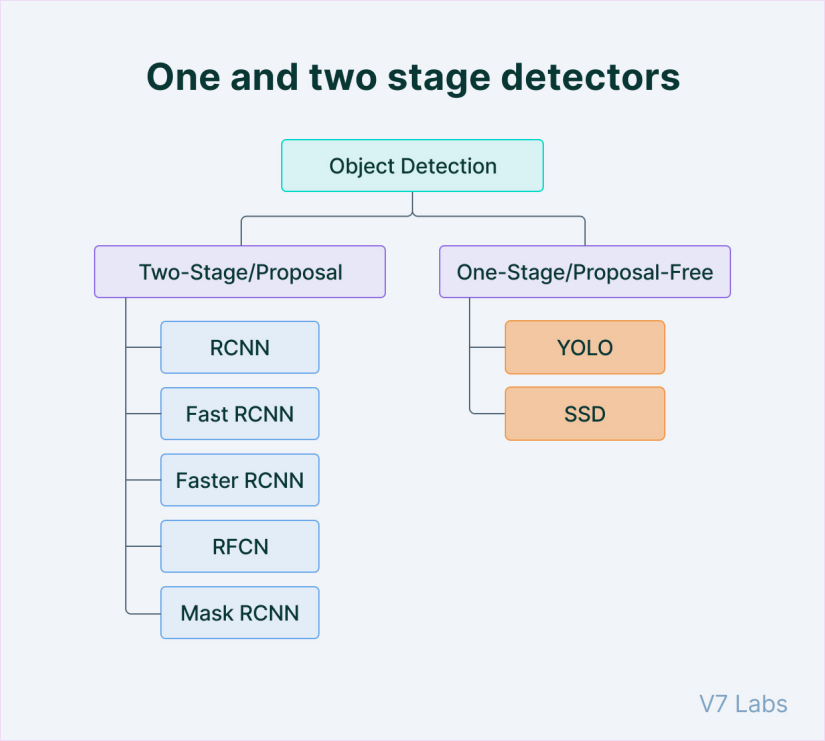
These weights were trained from scratch by using a simplified training recipe. Also available as `VGG16_Weights.DEFAULT`.

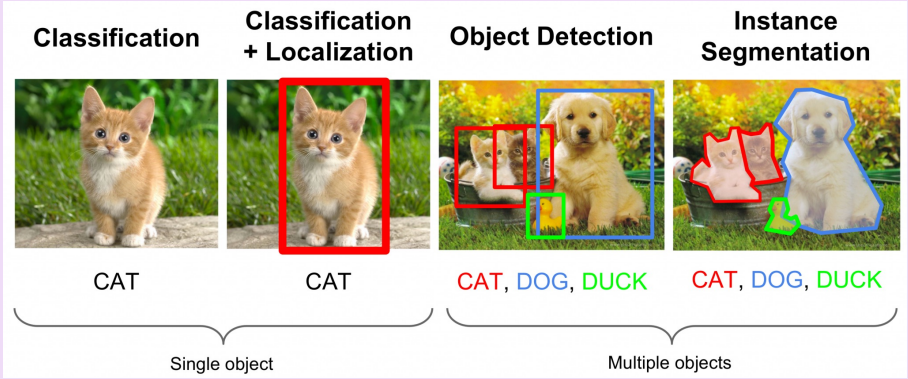
https://pytorch.org/vision/0.9/models.html

Object Detection



SOTA of Object Detection

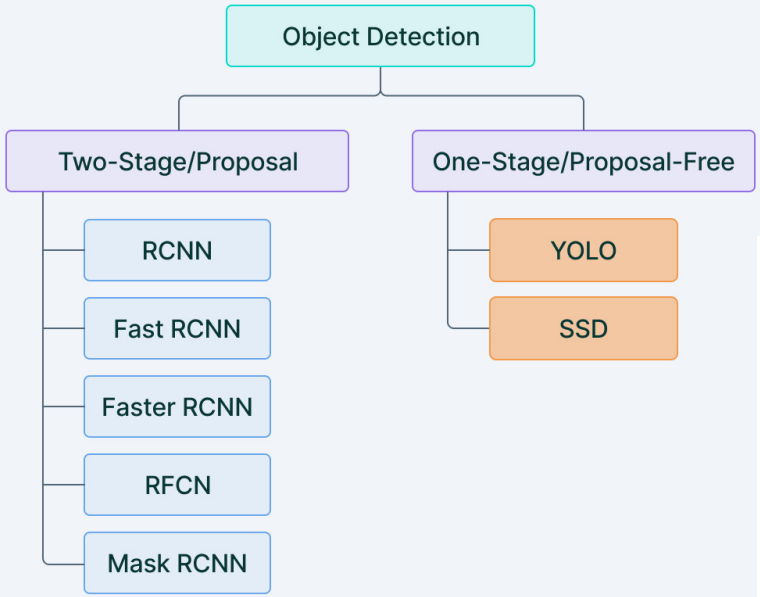




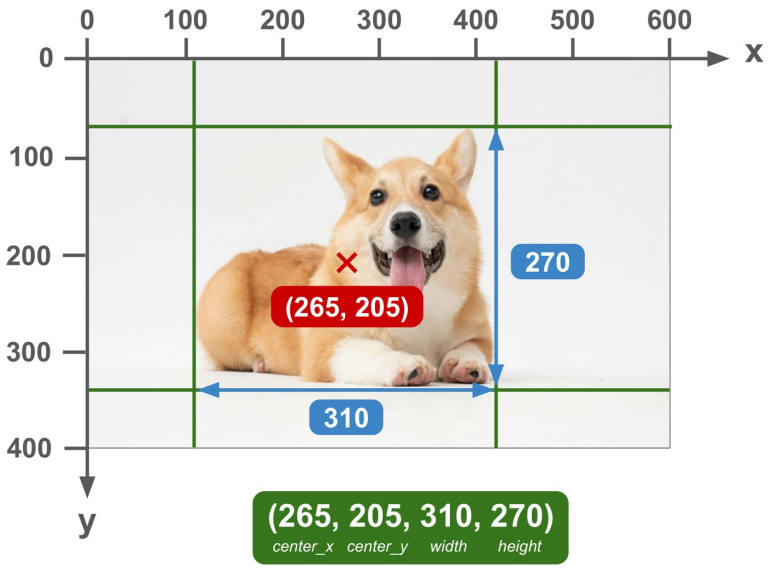
https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088

Object Detection

One and two stage detectors



V7



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

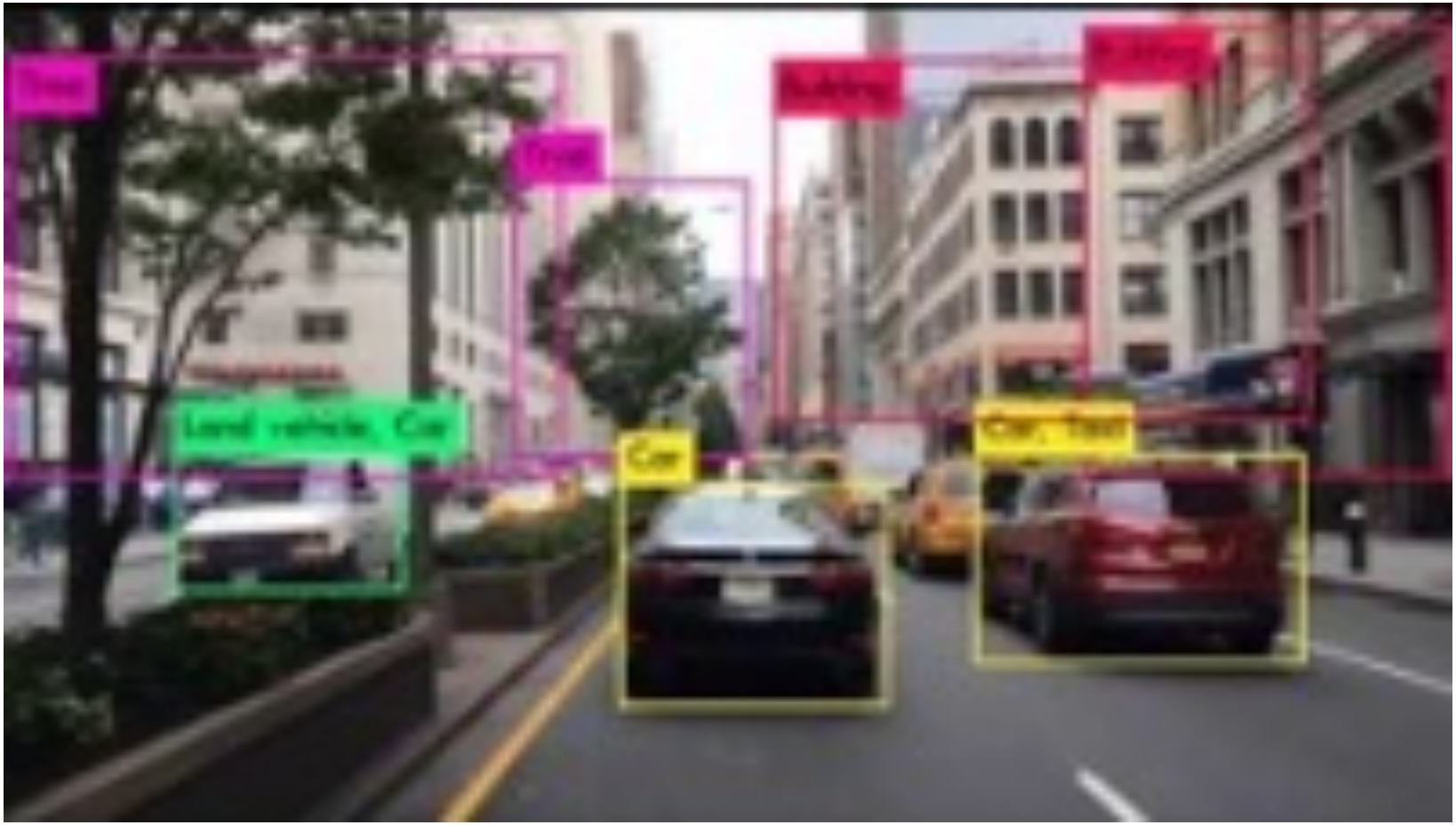
https://dragoneye.ai/blog/a-guide-to-bounding-box-formats/

YOLO Open Images in New York

YOLO Open Images in New York

Joseph Redmon
6.95K subscribers
Subscribe

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

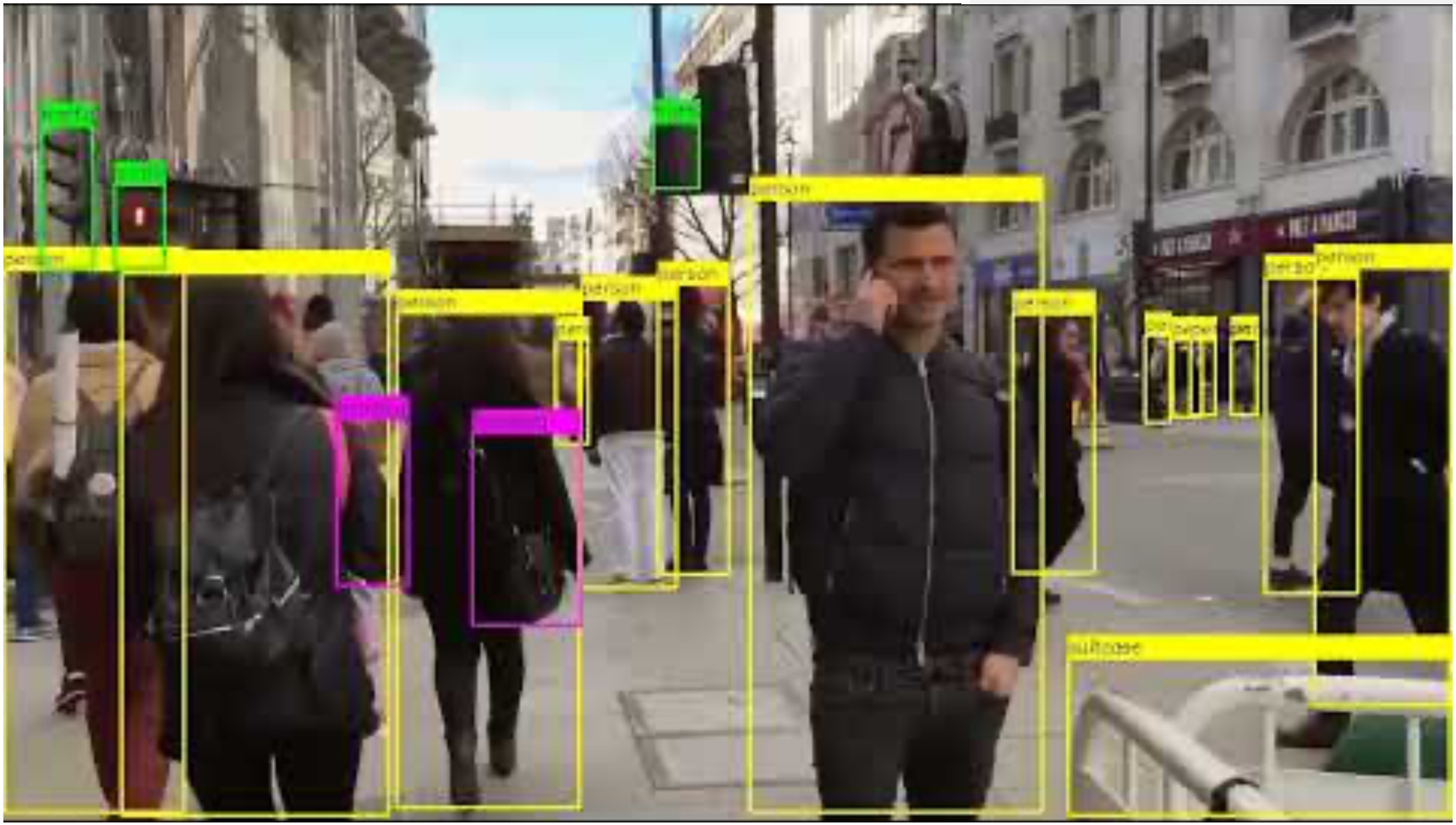


YOLO v5

Example of YOLO v5 detection on video file

Luiz doleron
127 subscribers
Subscribe

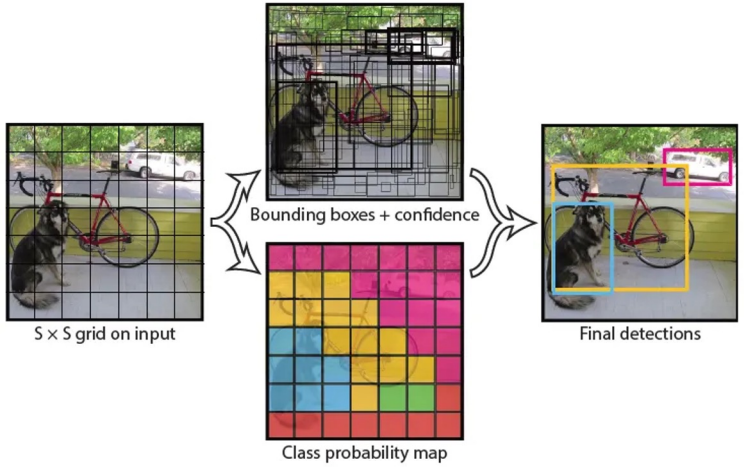
5,781 views Jan 23, 2022
Video example of YOLOv5 performing object detection.
- Check out the explanation at / detecting-objects-with-yolov5-opencv-pytho...
- Github repository is here: https://github.com/doleron/yolov5-ope...



What is YOLO?

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

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



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

COCO
Common Objects in Context

info@cocodataset.org

Home People Dataset Tasks Evaluate

News

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

What is COCO?

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

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

Collaborators

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

Sponsors

CVDF
Microsoft
facebook
Mighty AI

https://cocodataset.org/#home

ultralytics/cfg/datasets/coco8.yaml

79 classes

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

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

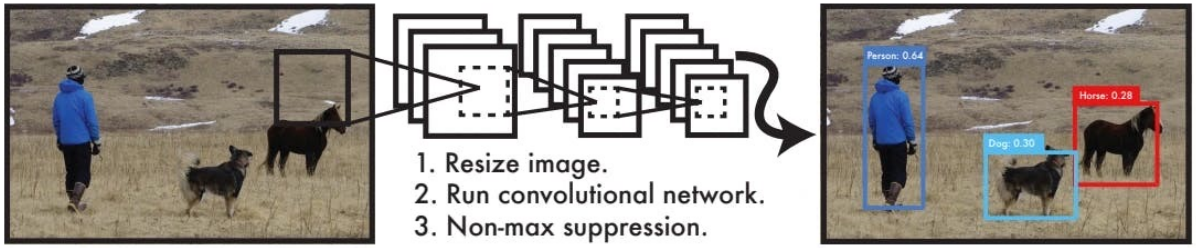
YOLO Dataset Format

The label file corresponding to the above image contains 2 persons (class 0) and a tie (class 27):

zidane.txt

```
0  0.481719 0.634028 0.690625 0.713278
0  0.741094 0.524306 0.314750 0.933389
27 0.364844 0.795833 0.078125 0.400000
```

YOLO's output



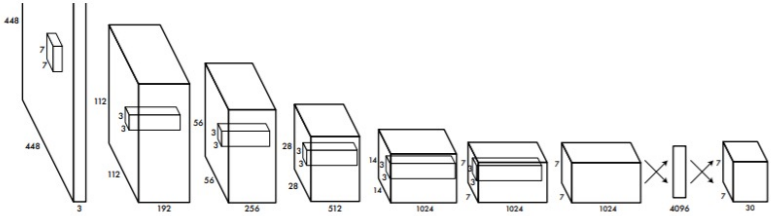
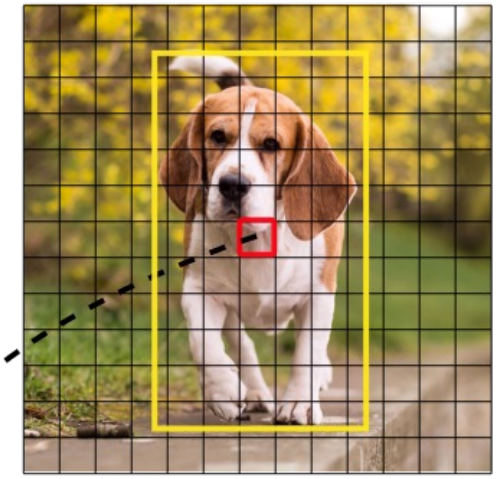
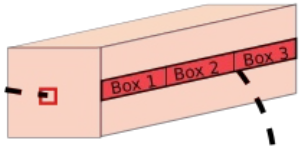


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

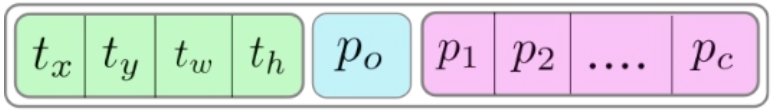
Image Grid. The Red Grid is responsible for detecting the dog



Prediction Feature Map



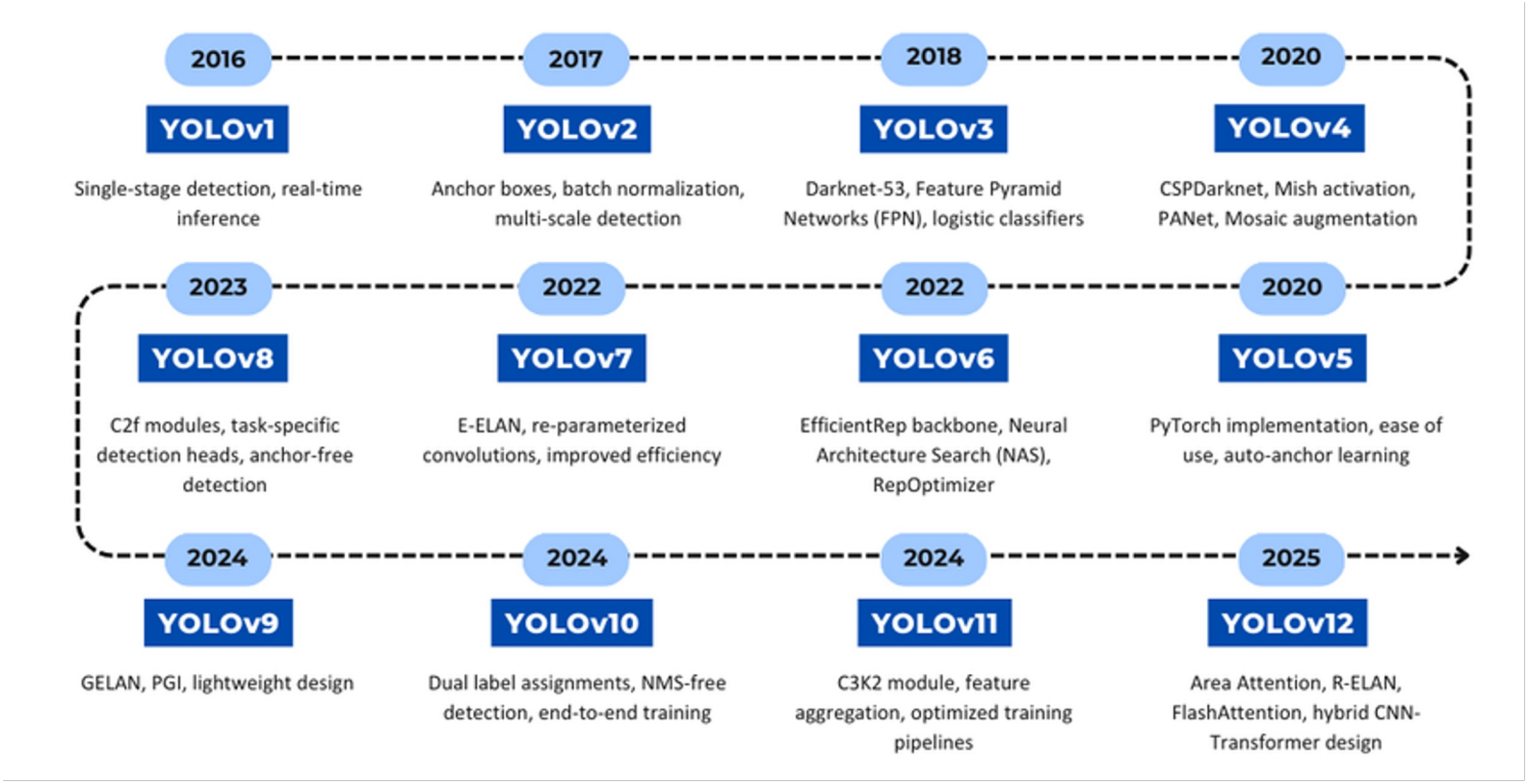
Attributes of a bounding box



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

https://arxiv.org/abs/2504.11995

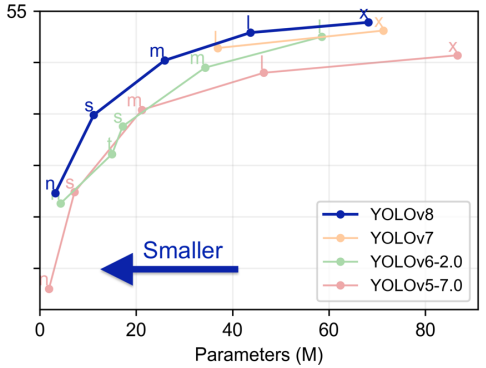
The Evolution of YOLO

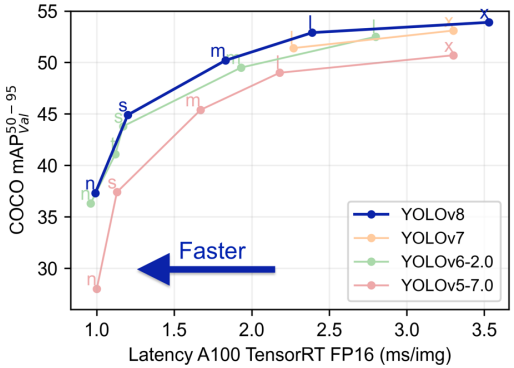


Ultralytics YOLO → production-ready, practical framework (YOLOv5 → YOLOv8 → YOLO11).
Academic YOLO → new research architectures (YOLOv9, v10, v11, v12).

Ultralytic's YOLO

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





English | 简体中文

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

Run on Gradient | Open in Colab | Open in Kaggle

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

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

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

Models

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

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

YOLO Models (v9 – v12) Timeline

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

Top Object Detection Models (2025)

Model	Team / Source	Notable Metric (COCO)	Strengths
RF-DETR	Roboflow	73.6 AP_{50} / 54.7 AP_{50:95} (Medium)	Real-time DETR; best mAP at low latency
YOLOv12	YOLO authors (YOLO Series)	—	Latest YOLO, attention-enhanced
YOLOv8 (Ultralytics) **YOLO11 (Ultralytics)**	Ultralytics	Up to ~50 AP_{50} / ~48.6 AP_{50:95}	Multi-task, production-ready

Explore Ultralytics YOLOv8

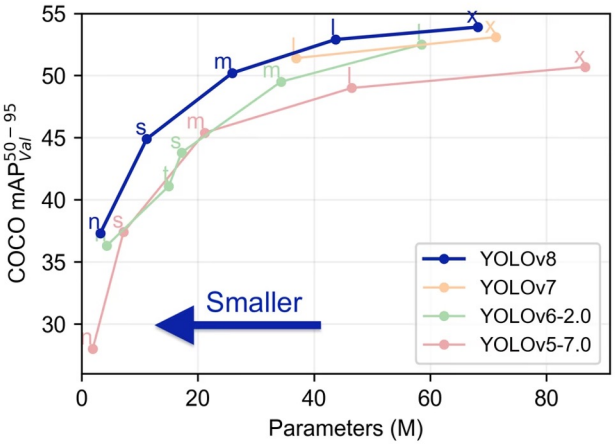
YOLOv5 (2020) → Anchor-based baseline
YOLOv8 (2023) → Anchor-free & simplified
YOLO11 (2025) → Refined, state-of-the-art

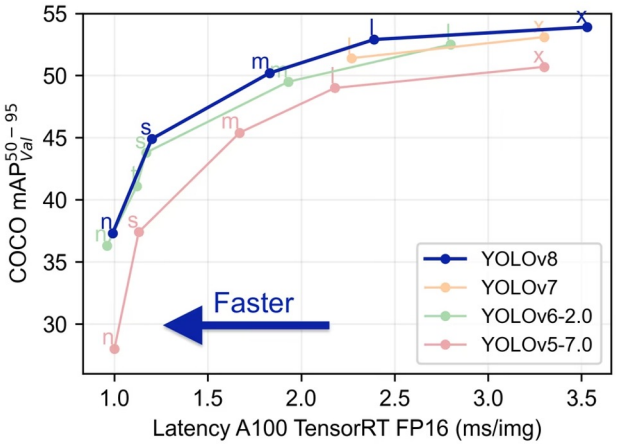
Overview

YOLOv8, launched by Ultralytics on January 10, 2023, enhances accuracy and speed while introducing features that optimize it for various object detection applications.

Key Features of YOLOv8

- Advanced Backbone and Neck Architectures
- Anchor-free Split Ultralytics Head
- Optimized Accuracy-Speed Tradeoff
- Variety of Pretrained Models





https://docs.ultralytics.com/models/yolov8/

Parameters (1)

Training

- YOLO training settings rely on hyperparameters that directly influence performance and training speed, making careful tuning essential for optimal results.

Example of training parameters:

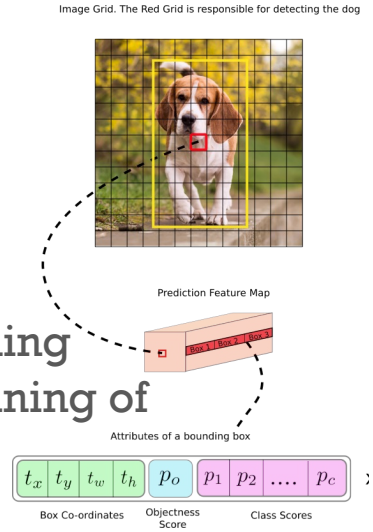
Argument	Type	Default	Description
epochs	int	100	Total number of training epochs (full dataset passes).
batch	int or float	16	Batch size. Supports auto modes (-1 or fraction like 0.7).
imgsz	int	640	Target training image size. Affects accuracy and compute cost.
val	bool	TRUE	Enables validation during training.
patience	int	100	Early stopping patience (epochs without improvement).
seed	int	0	Random seed for reproducibility.

For more details on training parameters: https://docs.ultralytics.com/usage/cfg/#modes:~:text=Modes%20Guide-,Train%20Settings,-Training%20settings%20for

Parameters (2)

Prediction

- YOLO prediction settings control inference performance and accuracy, including confidence threshold, NMS threshold, input size, and class handling. Proper tuning of these parameters is essential for optimal results.



Example of prediction parameters:

Parameter	Type	Default	Description
conf	float	0.25	Minimum confidence threshold. Detections below this value are discarded to reduce false positives.
iou	float	0.7	IoU threshold for Non-Maximum Suppression (NMS). Lower values remove more overlapping boxes.
imgsz	int or tuple	640	Inference image size. Can be a single integer (square) or (height, width) tuple.
device	str	None	Compute device selection (e.g., cpu, cuda:0, 0).
max_det	int	300	Maximum number of detections per image to prevent excessive outputs in dense scenes.
agnostic_nms	bool	FALSE	Applies class-agnostic NMS, merging overlapping boxes across different classes.

For more details on training parameters: https://docs.ultralytics.com/usage/cfg/#train-settings:~:text=Train%20Guide-,Predict%20Settings,-Prediction%20settings%20for

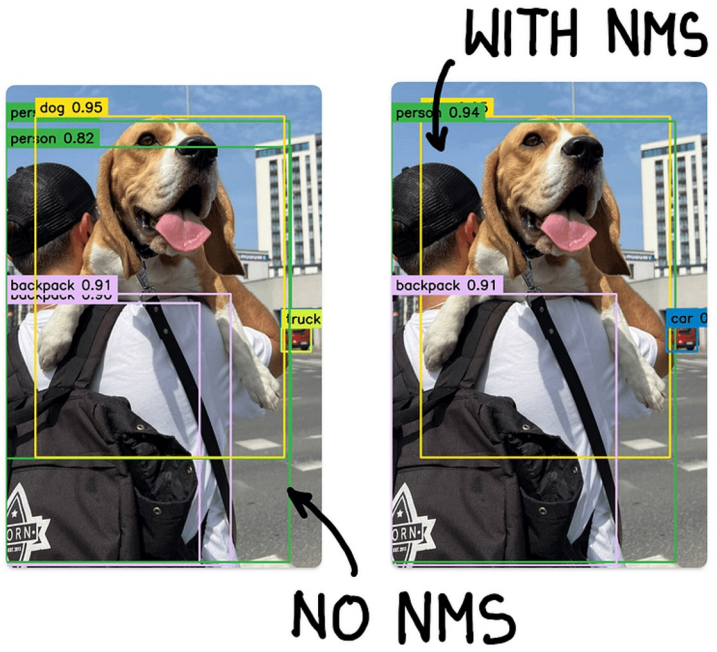
Non-Maximum Suppression (NMS)

What is NMS?

- Removes **duplicate overlapping boxes**
- Keeps the box with the highest confidence
- Controlled mainly by **IoU threshold**

Key Parameter

- Lower IoU → stricter suppression
- Higher IoU → keep more overlapping boxes

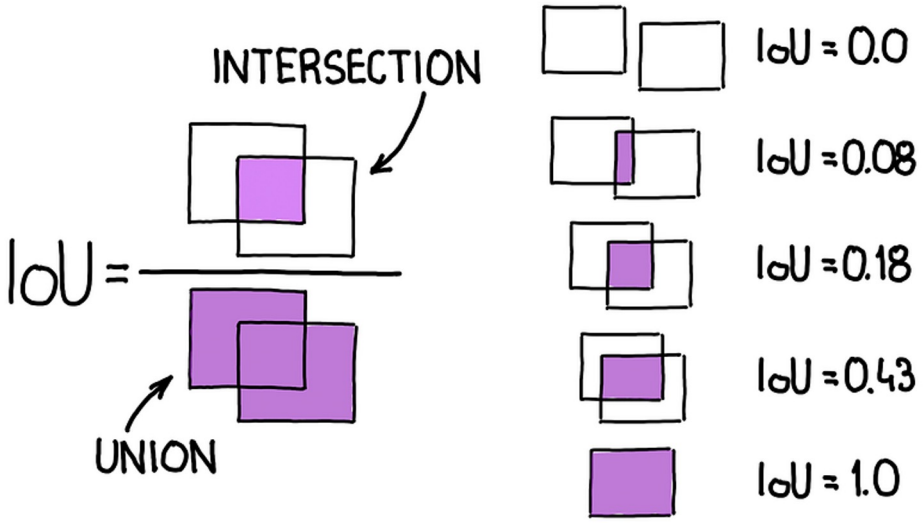


https://blog.roboflow.com/how-to-code-non-maximum-suppression-nms-in-plain-numpy/

IoU Threshold (iou)

Intersection over Union

- Measures how well a predicted bounding box overlaps with a ground-truth bounding box.



https://blog.roboflow.com/how-to-code-non-maximum-suppression-nms-in-plain-numpy/

YOLOv8 Tracking

Multi-Object Tracking with Ultralytics YOLO

- Tracks object locations and classes while preserving unique IDs across video frames
- Enables applications such as surveillance, security, and real-time sports analytics

Object Tracking with YOLOv8

- Ultralytics YOLO supports the following tracking algorithms:
  - ByteTrack
  - BoT-SORT

https://docs.ultralytics.com/modes/track/

YOLOv8 Tracking

Tracker Arguments

- Some tracking behaviors can be fine-tuned by editing the tracking algorithm–specific YAML configuration files.
  - botsort.yaml
  - bytetrack.yaml

Parameter	Valid Values or Ranges	Description
`tracker_type`	`botsort`, `bytetrack`	Specifies the tracker type. Options are `botsort` or `bytetrack`.
`track_high_thresh`	`0.0-1.0`	Threshold for the first association during tracking used. Affects how confidently a detection is matched to an existing track.
`track_low_thresh`	`0.0-1.0`	Threshold for the second association during tracking. Used when the first association fails, with more lenient criteria.
`new_track_thresh`	`0.0-1.0`	Threshold to initialize a new track if the detection does not match any existing tracks. Controls when a new object is considered to appear.
`track_buffer`	`>=0`	Buffer used to indicate the number of frames lost tracks should be kept alive before getting removed. Higher value means more tolerance for occlusion.
`match_thresh`	`0.0-1.0`	Threshold for matching tracks. Higher values makes the matching more lenient.
`fuse_score`	`True`, `False`	Determines whether to fuse confidence scores with IoU distances before matching. Helps balance spatial and confidence information when associating.
`gmc_method`	`orb`, `sift`, `ecc`, `sparseOptFlow`, `None`	Method used for global motion compensation. Helps account for camera movement to improve tracking.
`proximity_thresh`	`0.0-1.0`	Minimum IoU required for a valid match with ReID (Re-identification). Ensures spatial closeness before using appearance cues.
`appearance_thresh`	`0.0-1.0`	Minimum appearance similarity required for ReID. Sets how visually similar two detections must be to be linked.
`with_reid`	`True`, `False`	Indicates whether to use ReID. Enables appearance-based matching for better tracking across occlusions. Only supported by BoTSORT.
`model`	`auto`, `yolo11[nsmlx]-cls.pt`	Specifies the model to use. Defaults to `auto`, which uses native features if the detector is YOLO, otherwise uses `yolo11n-cls.pt`.

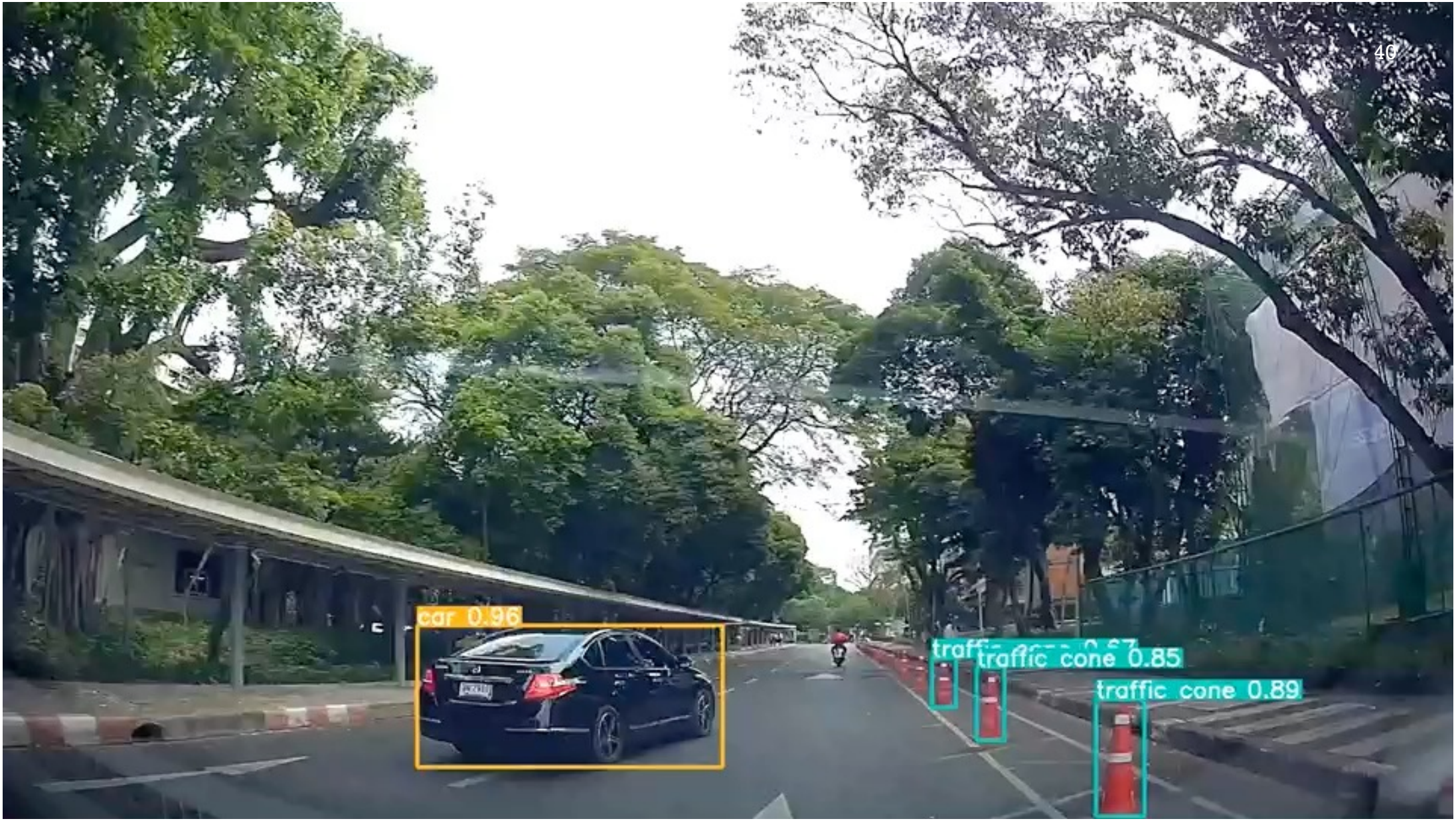
Example Usage

```
from ultralytics import YOLO

# Load an official or custom model
model = YOLO("yolov8n.pt")  # Load an official Detect model

# Perform tracking with the model
results = model.track("https://youtu.be/LNwODJXcvt4", show=True)  # Tracking with default tracker
results = model.track("https://youtu.be/LNwODJXcvt4", show=True, tracker="bytetrack.yaml")  # with ByteTrack
```

https://docs.ultralytics.com/modes/track/

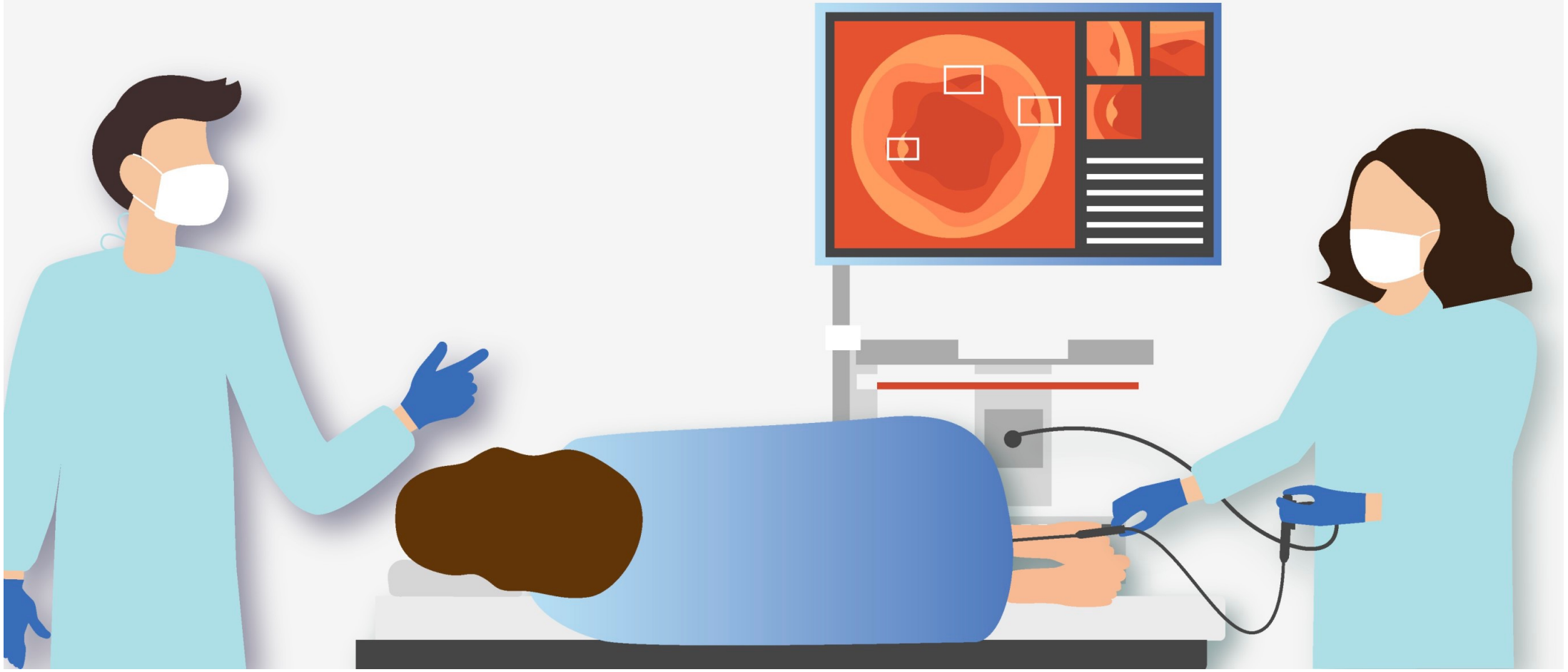

boat 0.55
boat 0.40


In Frame: 3
Identified: 3
workpoint
ID: Orn
DB: 88 (img)
ID: Cherprang
DB: 176 (img)
ID: Pun
DB: 48 (img)
ตัวอย่างความสนุกวันนี้
www.youtube.com/workpointofficial


car 0.96
traffic cone 0.85
traffic cone 0.89

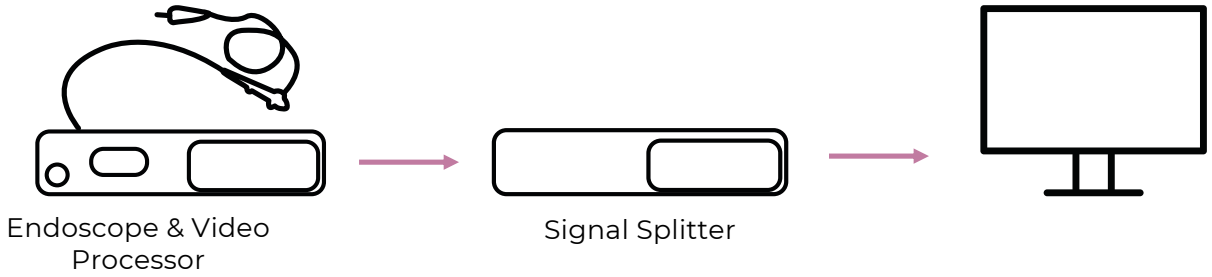
DeepGI: Real-Time Colonic Polyp Detection

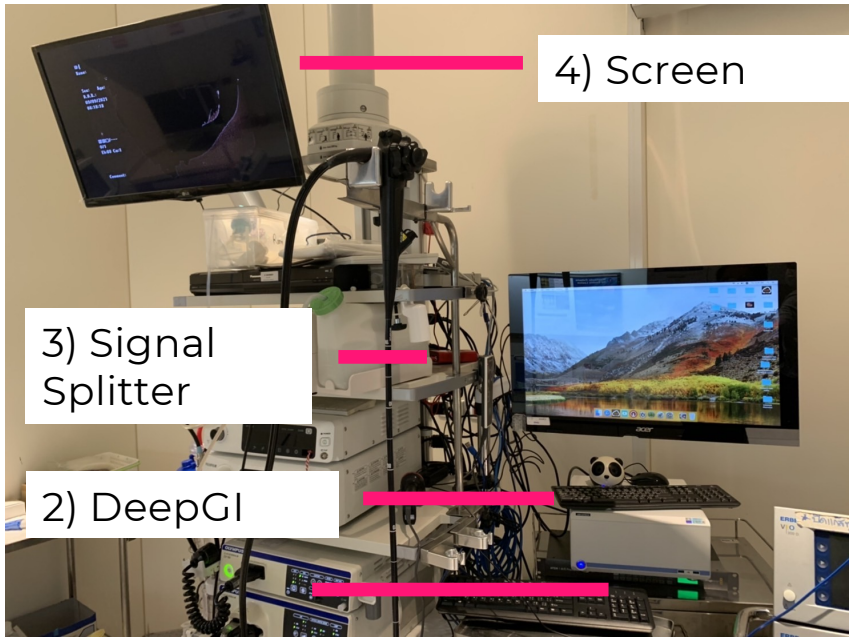
An AI-assisted solution that aims to improve colonic polyp detection in real-time. It is compatible with all standard colonoscope systems.



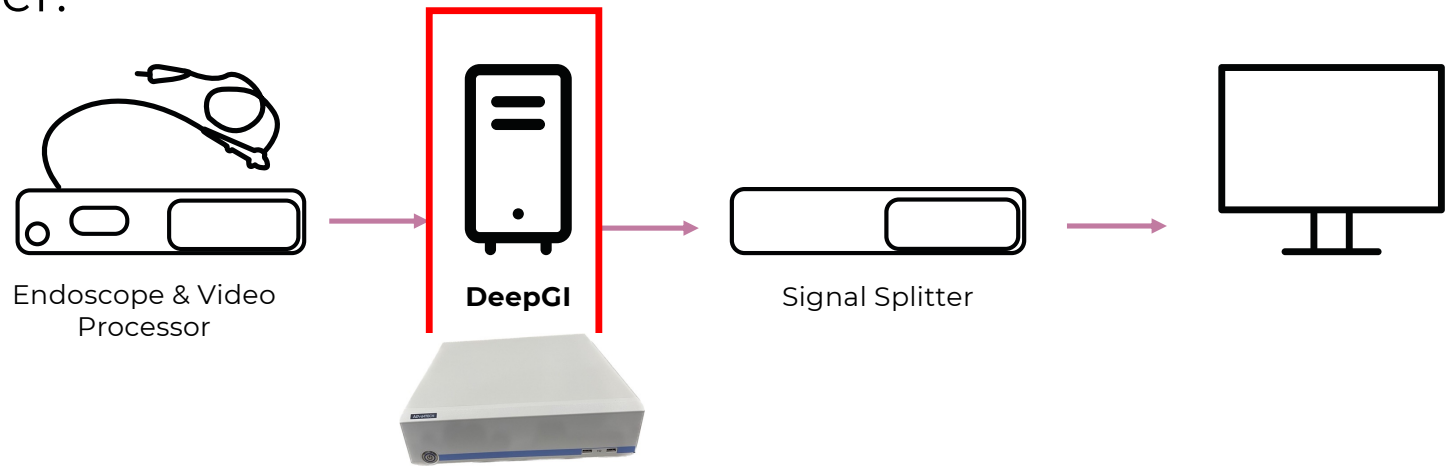
System Architecture

Before:

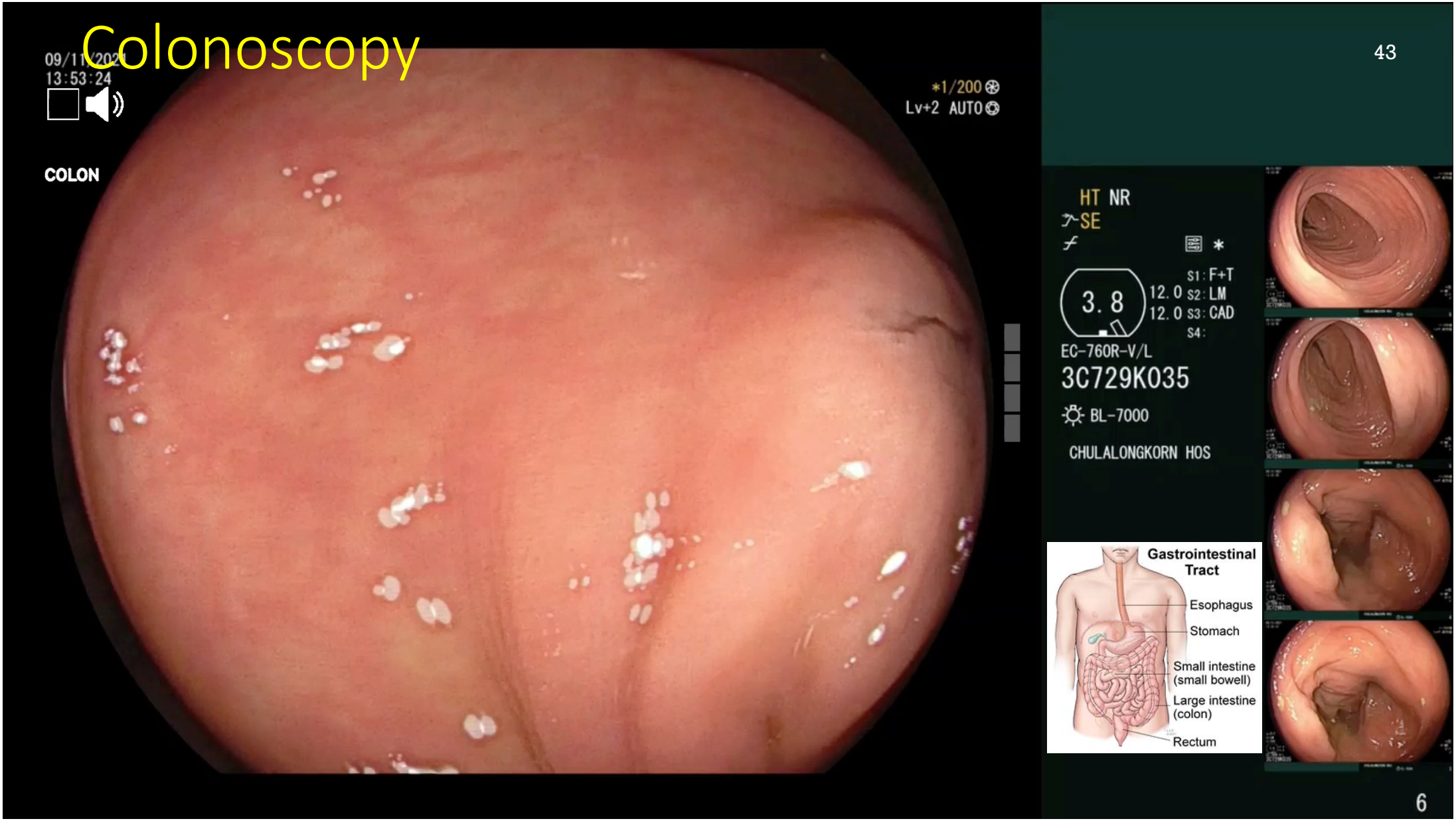




After:



Colonoscopy



+ Thank you
& any questions