Mechatronics

Day11

CNN and Hyperparameters Tuning

Train – Validate – Test Dataset

Re-initialize Image Data Generator

```
tf.random.set_seed(42)
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale = 1./255,validation_split=0.2)
test_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale = 1./255)
train_dir = "pizza_steak/train/"
test_dir = "pizza_steak/test/"
```

Re-generate dataset

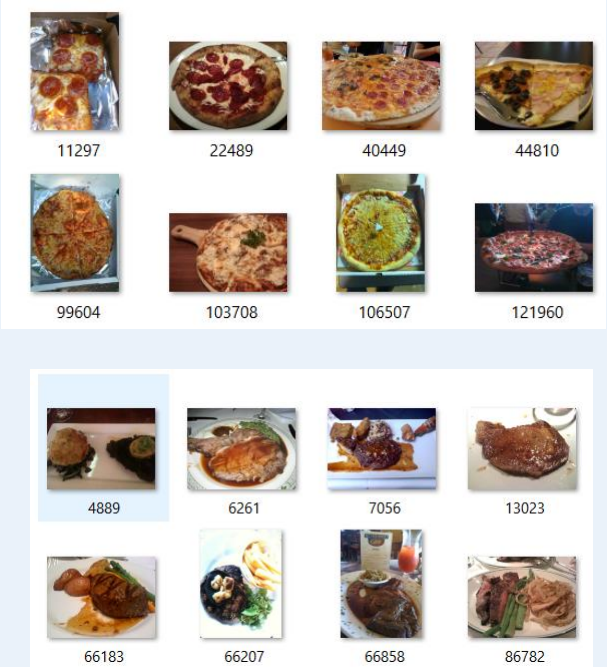
```
train_data = train_datagen.flow_from_directory(train_dir, batch_size = 32, target_size = (224,224),
                                               class_mode= "binary",subset='training')
valid_data = train_datagen.flow_from_directory(train_dir, batch_size = 32, target_size = (224,224),
                                               class_mode= "binary",subset='validation')
test_data = test_datagen.flow_from_directory(test_dir, batch_size = 32, target_size = (224,224),
                                             class_mode= "binary")
```

Plotting Confusion Matrix

Import library and define directory and classes of the images.

```
import os
import numpy as np
import tensorflow as tf
import seaborn as sns
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing import image
from sklearn.metrics import confusion_matrix, accuracy_score

# Define test directory
test_dir = "/content/pizza_steak/test"

# Class names (ensure they match your training labels)
class_names = ['pizza', 'steak']
```

Test Folder



Plotting Confusion Matrix

Import library and define directory and classes of the images.

List to store the labels

```
y_true = []
y_pred = []

# Loop through each class folder (pizza, steak)
for class_name in class_names:
    class_folder = os.path.join(test_dir, class_name)

    for img_name in os.listdir(class_folder):
        img_path = os.path.join(class_folder, img_name)

        # Load and preprocess image
        img = image.load_img(img_path, target_size=(224, 224))
        img_array = image.img_to_array(img) / 255.0
        img_array = np.expand_dims(img_array, axis=0)  # Add batch dimension
```

Iterate all images

Normalize images

Plotting Confusion Matrix

Prediction

```
# Predict using model
prediction = model.predict(img_array).flatten()[0]
```

```
# Convert probability to class (0 = steak, 1 = pizza)
predicted_class = 1 if prediction > 0.5 else 0
actual_class = class_names.index(class_name)  # Get actual class index
```

Interprete class

Append prediction values

```
# Store results
y_pred.append(predicted_class)
y_true.append(actual_class)
```

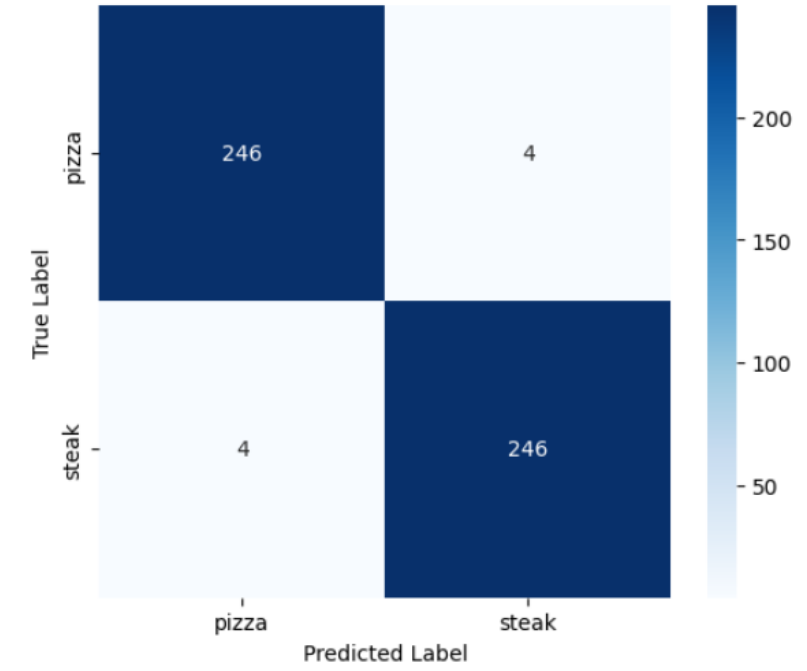
Plotting Confusion Matrix

Calculate accuracy

```
# Calculate accuracy
accuracy = accuracy_score(y_true, y_pred)
print(f"Model Accuracy on Test Set: {accuracy * 100:.2f}%")
```

```
# Compute confusion matrix
cm = confusion_matrix(y_true, y_pred)
```

Create confusion matrix




```
# Plot confusion matrix
plt.figure(figsize=(6, 5))
sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", xticklabels=class_names, yticklabels=class_names)
plt.xlabel("Predicted Label")
plt.ylabel("True Label")
plt.title("Confusion Matrix")
plt.show()
```

TensorFlow Early Stop

- Stop training when a monitored metric has stopped improving.

```
from tensorflow.keras.callbacks import EarlyStopping

es_callback = EarlyStopping(
    monitor='val_loss',
    patience=5,
    restore_best_weights=True,
    verbose=1
)
```

Import Tool

Target Metric

Waiting Epochs

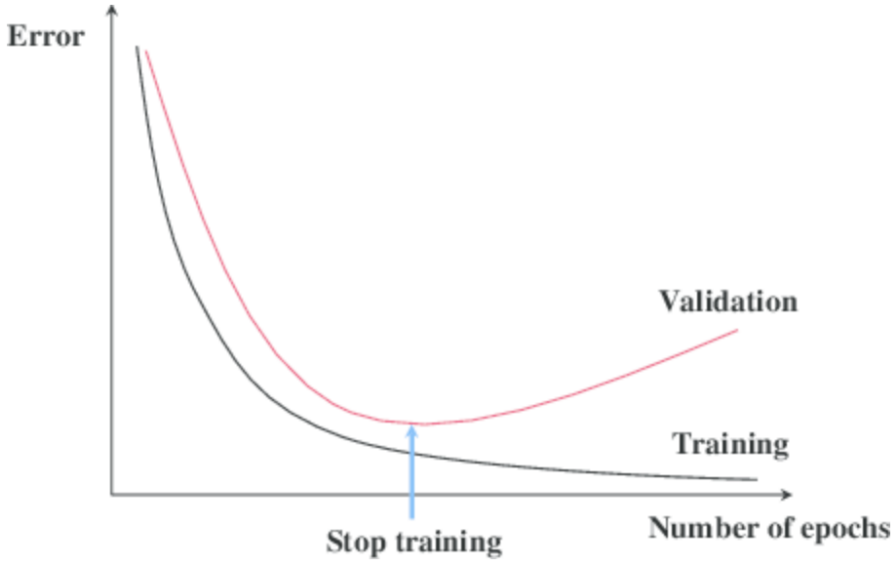
Restore best weight

TensorFlow Early Stop

- Stop training when a monitored metric has stopped improving.

```
hist = model.fit(train_data,
                 epochs=100,
                 batch_size=32,
                 validation_data= valid_data,
                 callbacks=[es_callback],

                 )
```

Add callback function to the training



Learning Rate Scheduler

- A learning rate scheduler in TensorFlow is a mechanism to adjust the learning rate dynamically during training to improve model performance and stability. Instead of using a fixed learning rate, a scheduler modifies the learning rate over epochs or iterations based on a predefined strategy.

```
from tensorflow.keras.callbacks import LearningRateScheduler
def lr_schedule(epoch, lr):
    return lr * 0.95  # Reduce learning rate by 5% each epoch

scheduler_callback = tf.keras.callbacks.LearningRateScheduler(lr_schedule)
```

Epoch	1	2	3	4	5	6
LR	0.001	0.00095	0.00090	0.00085	0.000814	0.00077

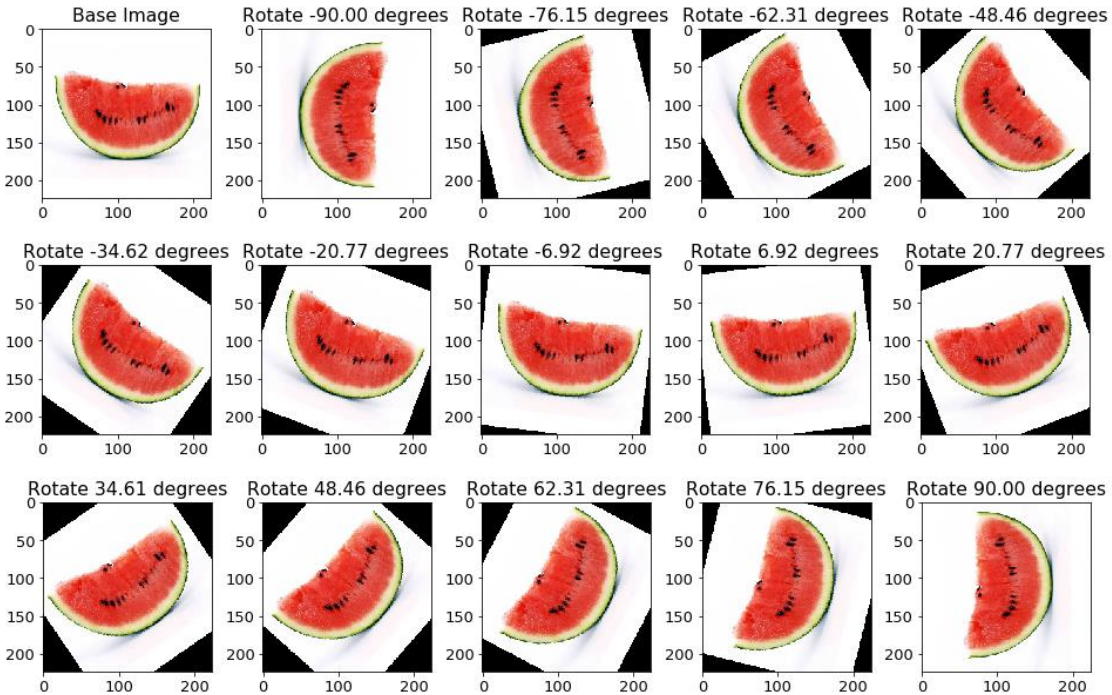
Learning Rate Scheduler

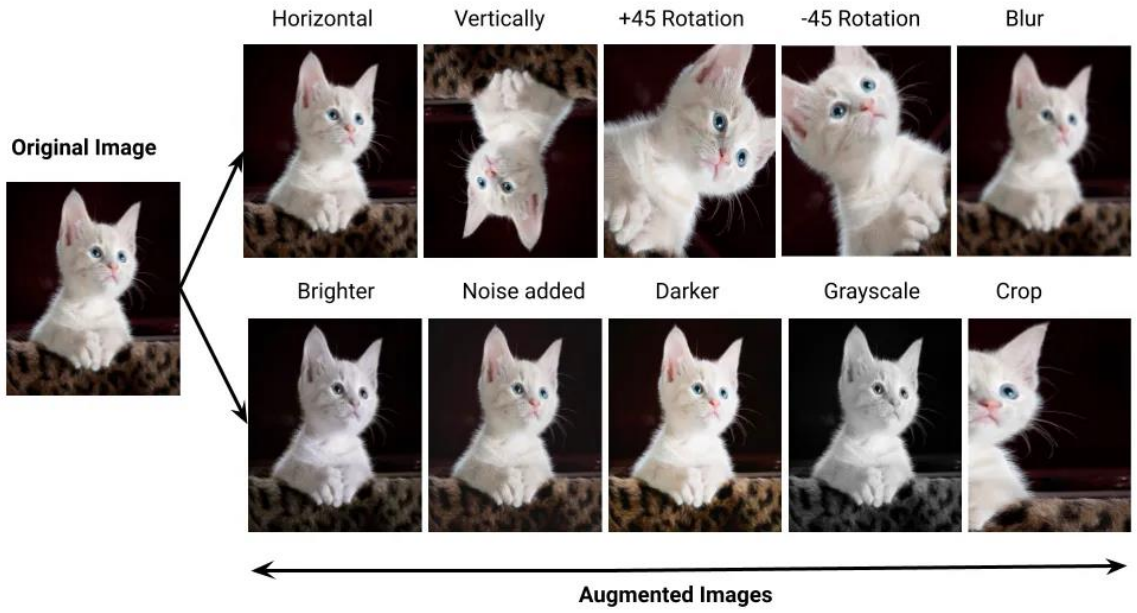
- **Prevents Overshooting**: A high learning rate may cause the model to oscillate and never converge.
- **Avoids Getting Stuck in Local Minima**: A decaying learning rate helps fine-tune the model by taking smaller steps.
- **Speeds Up Convergence**: A higher initial learning rate allows the model to learn faster in the early stages.

```
hist = model.fit(train_data,
                 epochs=100,
                 batch_size=32,
                 validation_data= valid_data,
                 callbacks=[es_callback, scheduler_callback],

                 )
```

Data Augmentation

- **Data augmentation** is a technique used to artificially expand a dataset by applying various transformations to training images. This helps improve model generalization and prevent overfitting.





Data Augmentation

```
tf.random.set_seed(42) Number of Universe
train_datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=40,          # Random rotations
    width_shift_range=0.2,      # Horizontal shifting
    height_shift_range=0.2,     # Vertical shifting
    shear_range=0.2,            # Shear transformation
    zoom_range=0.2,             # Zooming
    horizontal_flip=True,       # Horizontal flip
    fill_mode='nearest'         # Fill mode for pixels after transformation
)

test_datagen = tf.keras.preprocessing.image.ImageDataGenerator(rescale = 1./255)
train_dir = "pizza_steak/train/"
test_dir = "pizza_steak/test/"
```

Augmentation Setup

Example8.1 Pizza-Steak Model Tuning

- From previous EX8, tuning the model to get accuracy more that 80%
- Scoring: $score = (\%Accuracy_{val} - 80\%) \times 3$

Practice Pill QC Model Tuning

- From Pill QC practice problem, tuning the model to get accuracy more that 80%
- Scoring: $score = (\%Accuracy_{val} - 80\%) \times 3$

Save and Load model

Save trained model in the Keras file format

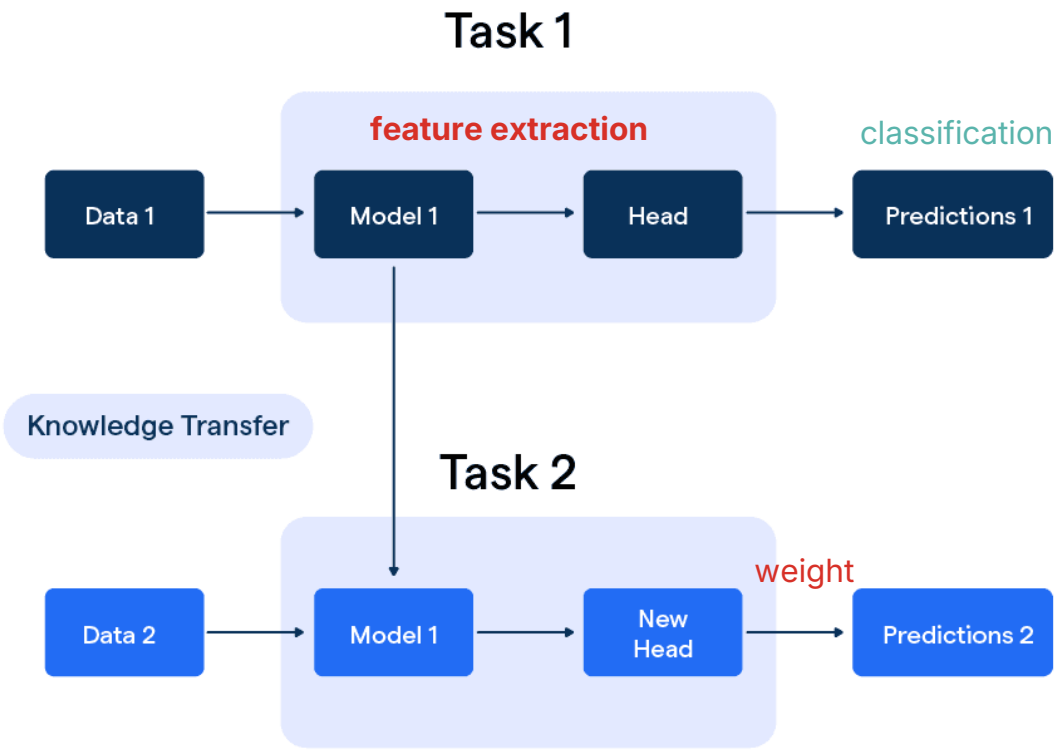
```
model.save("my_model.keras")
```

Load Keras model

```
loaded_model = tf.keras.models.load_model("my_model.keras")
```

Transfer Learning

- Transfer learning is an approach to machine learning where a model trained on one task is used as the starting point for a model on a new task.
- This is done by transferring the knowledge that the first model has learned about the features of the data to the second model.



pre-trained model

Ex:

YOLO

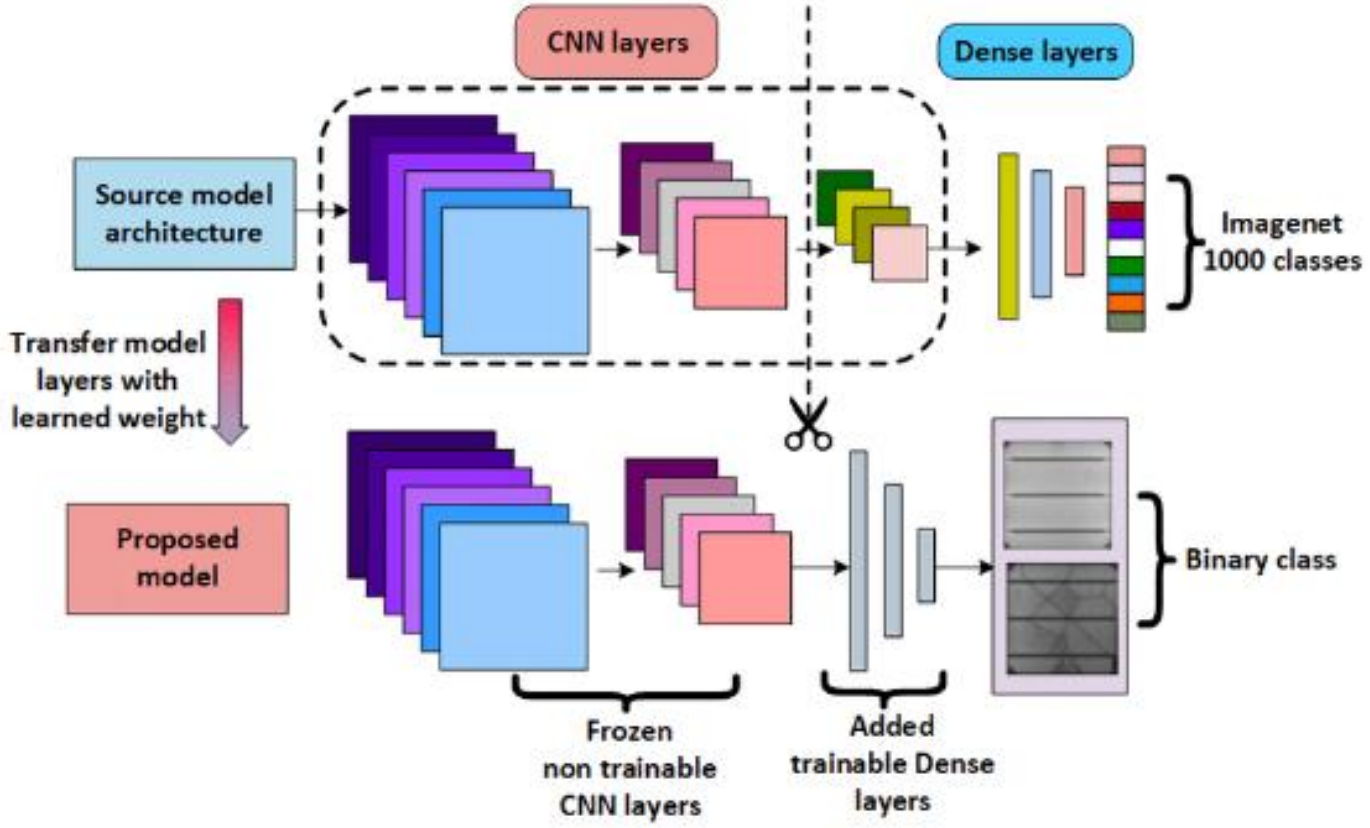
CNN Resnet, MobileNet

Training dataset ⇒ imageNet, COCO

Why use Transfer Learning?

reduce epoch

- **Reduces Training Time**: Leveraging pre-trained weights eliminates the need to train from scratch.
- **Works with Limited Data**: Pre-trained models are less prone to overfitting when fine-tuned on small datasets.
- **Improves Performance**: Models trained on massive datasets like ImageNet capture rich feature hierarchies.

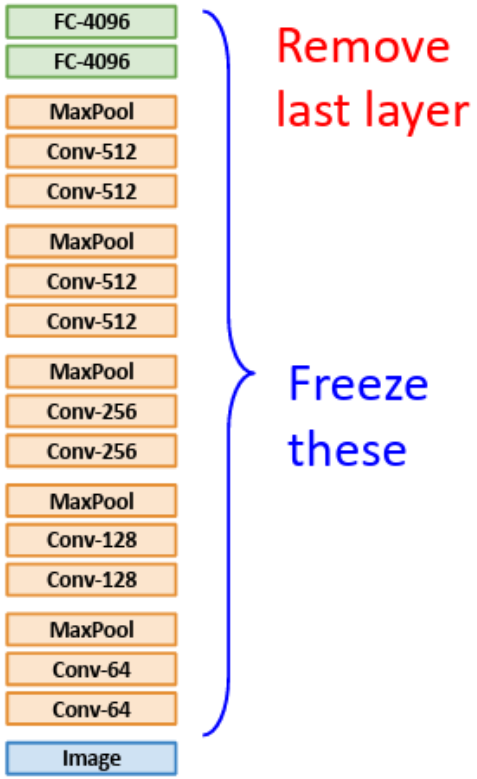


https://www.researchgate.net/figure/Proposed-framework-of-transfer-learning-model_fig3_359273663?__cf_chl_tk=nDMrGe8Wu_Tb0_AvF43KOm2Fy7ytjLGOjTZdPmcgc0s-1739294038-1.0.1.1-UvhuTfizISoDcKEIfJYdHN30FRckec44Ag54ruLhva0

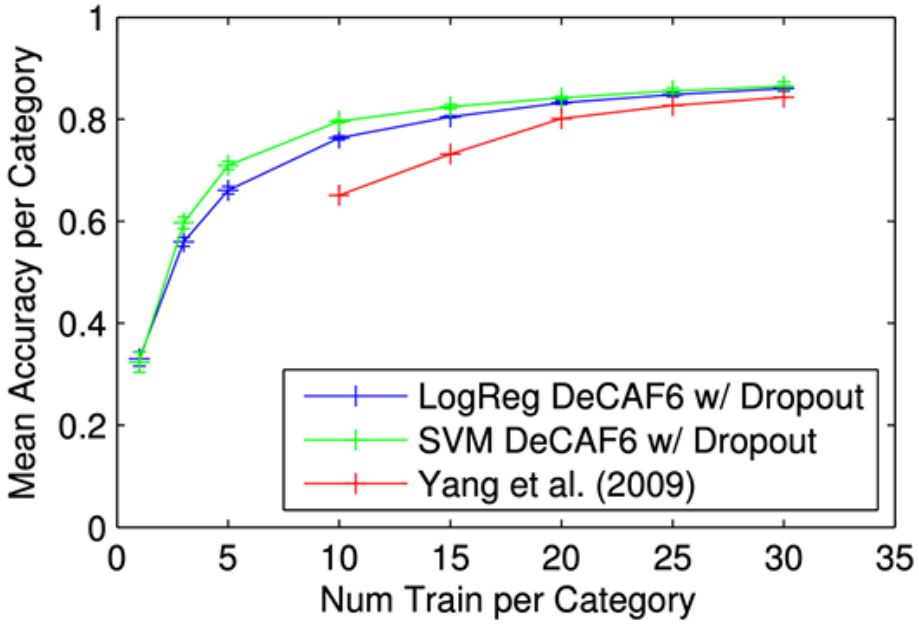
Transfer Learning: Feature Extraction

1. Train on ImageNet

2. Extract features with CNN, train linear model



Classification on Caltech-101



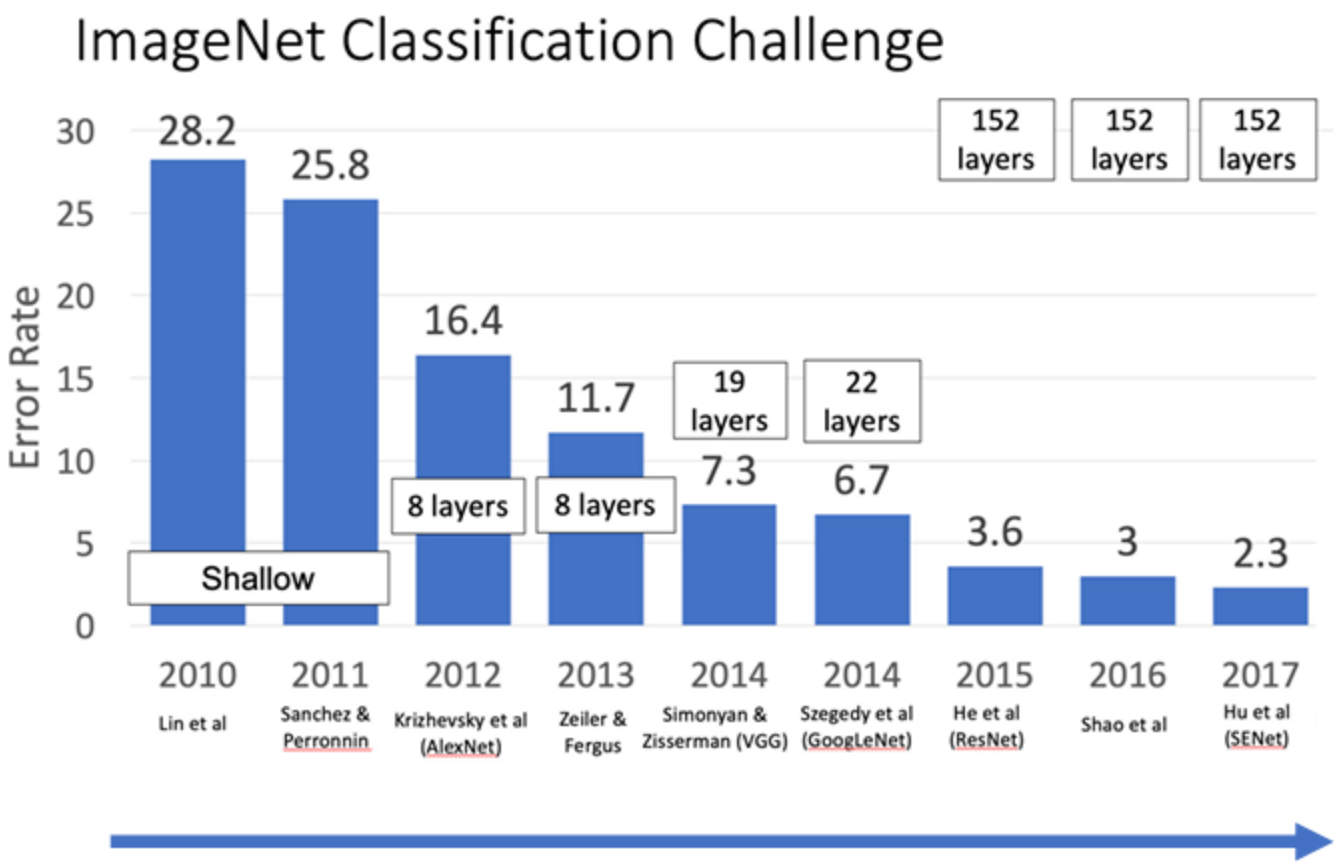
Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014

Transfer Learning: Tricks

- Train with feature extraction first before fine-tuning
- Lower the learning rate: use ~1/10 of LR used in original training

Required fine tuning so:
- should not update weight too much (because it is already trained)
-use low learning rate ex. lr of pre-trained model/10

Transfer Learning



Improvements in CNN architectures lead to improvements in many downstream tasks

TensorFlow

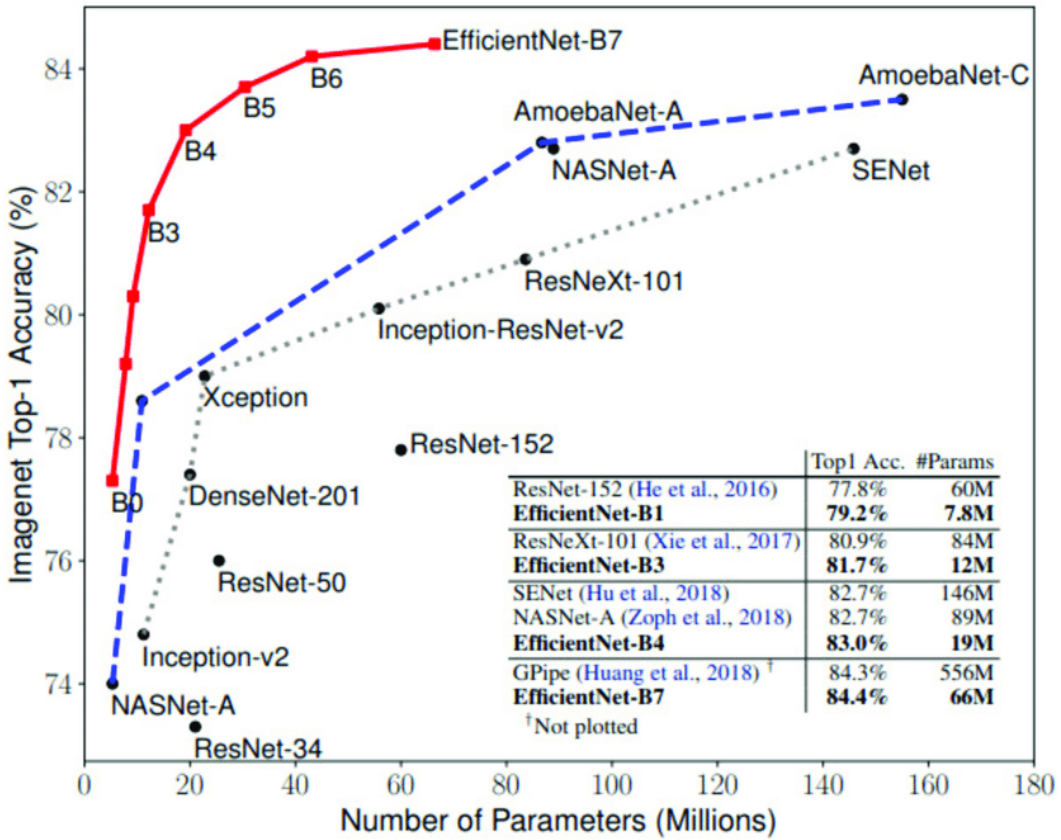
Popular pre-trained model

every model has more variation e.g ResNet18 VG16

- **ResNet (Residual Network):** Excellent for image classification tasks.
- **VGGNet:** Simpler, but computationally expensive.
- **InceptionNet:** Suitable for image data with high variability.
- **EfficientNet:** Balances accuracy and efficiency.
- **BERT and GPT:** For natural language processing tasks.

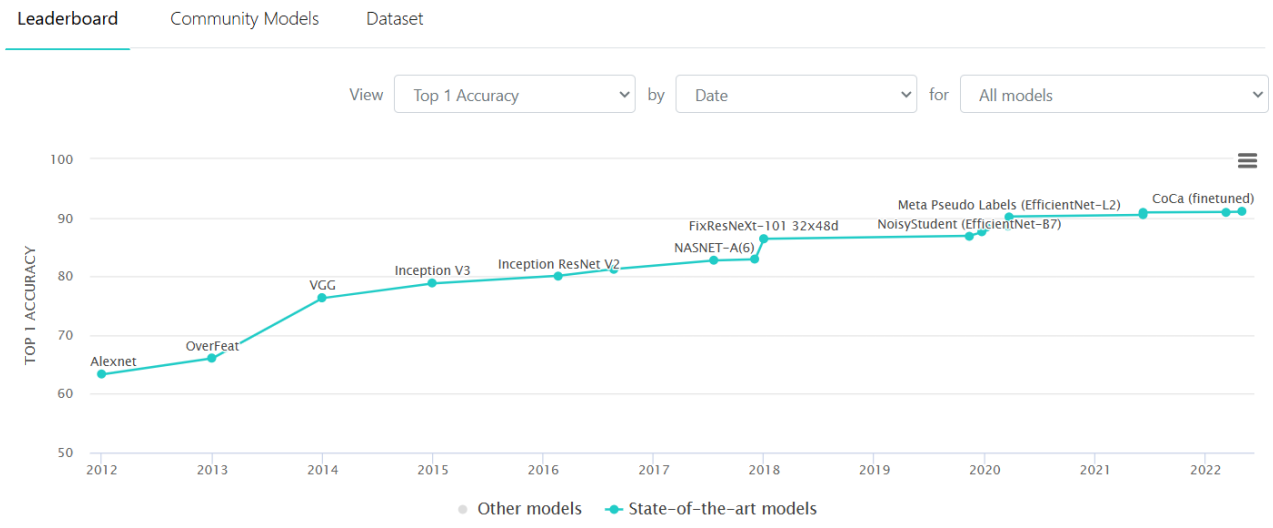
MobileNet

CNN Network Comparison



https://www.researchgate.net/publication/361291112_Waste_Classification_for_Sustainable_Development_Using_Image_Recognition_with_Deep_Learning_Neural_Network_Models/figures?lo=1&utm_source=google&utm_medium=organic

Image Classification on ImageNet



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

EX9: Pizza Steak Transfer Learning

- From Pizza-Steak CNN training, use transfer learning to improve model accuracy.

EX9: Pizza Steak Transfer Learning

- Create base_model from ResNet50

True = top network(decision/prediction)

Not include top head

model name

dataset

```
base_model = tf.keras.applications.ResNet50(weights = 'imagenet', include_top = False)
```

```
base_model.summary()
```

conv5_block3_2_bn (BatchNormalization)	(None, None, None, 512)	2,048	conv5_block3_2_conv[0…
conv5_block3_2_relu (Activation)	(None, None, None, 512)	0	conv5_block3_2_bn[0][…
conv5_block3_3_conv (Conv2D)	(None, None, None, 2048)	1,050,624	conv5_block3_2_relu[0…
conv5_block3_3_bn (BatchNormalization)	(None, None, None, 2048)	8,192	conv5_block3_3_conv[0…
conv5_block3_add (Add)	(None, None, None, 2048)	0	conv5_block2_out[0][0… conv5_block3_3_bn[0][…
conv5_block3_out (Activation)	(None, None, None, 2048)	0	conv5_block3_add[0][0]

Total params: 23,587,712 (89.98 MB)
Trainable params: 23,534,592 (89.78 MB)
Non-trainable params: 53,120 (207.50 KB)

EX9: Pizza Steak Transfer Learning

- Apply base model with custom model

```
x = base_model.output
x = tf.keras.layers.GlobalAveragePooling2D()(x)


x = tf.keras.layers.Dense(5, activation = 'relu')(x)
preds = tf.keras.layers.Dense(1, activation = 'sigmoid')(x)
```

conv5_block3_add (Add)	(None, None, None, 2048)	0	conv5_block2_out[0][0… conv5_block3_3_bn[0][…
conv5_block3_out (Activation)	(None, None, None, 2048)	0	conv5_block3_add[0][0]
global_average_pooling2d… (GlobalAveragePooling2D)	(None, 2048)	0	conv5_block3_out[0][0]
dense_24 (Dense)	(None, 5)	10,245	global_average_poolin…
dense_26 (Dense)	(None, 5)	30	dense_24[0][0]
dense_27 (Dense)	(None, 1)	6	dense_26[0][0]

Total params: 23,597,993 (90.02 MB)
Trainable params: 23,544,873 (89.82 MB)
Non-trainable params: 53,120 (207.50 KB)

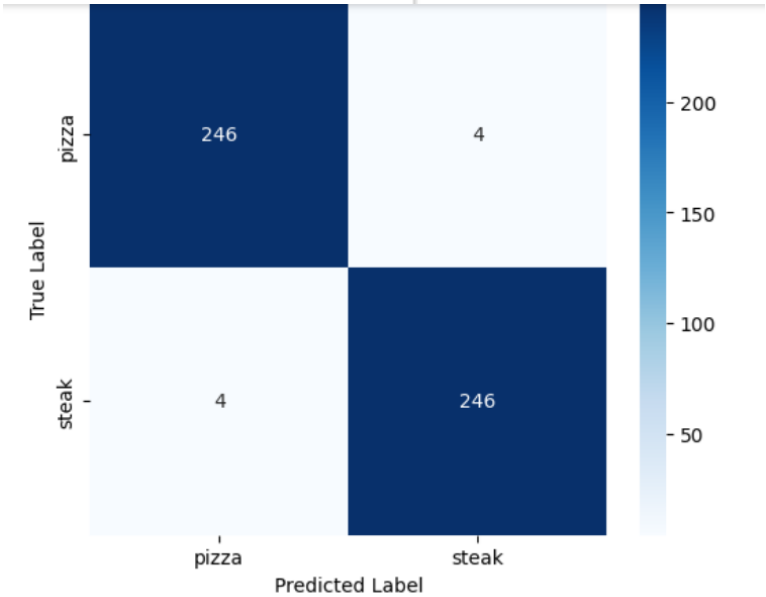
EX9: Pizza Steak Transfer Learning

- Compile model

```
model.compile(loss="binary_crossentropy",optimizer=tf.keras.optimizers.Adam(0.0001),
              metrics=["accuracy", "precision","recall"])
```

- Train the model

```
hist = model.fit(train_data,
                 epochs=30,
                 batch_size=32,
                 validation_data= valid_data)
```




```
Epoch 30/30
38/38 ━━━━━━━━━━━━━━━━━━━━ 13s 344ms/step - accuracy: 1.0000 - loss: 0.3261 - precision: 1.0000 - recall: 1.0000 - val_accuracy: 0.9500 - val_loss: 0.4050
```

Practice: PillQC transfer learning

- From PillQC datasets, use transfer learning to improve model accuracy up to 92%
- Scoring: $score = (\%accuracy_{val} - 90\%) * 2$

ANN guideline

input layer -> neuron size = input size
output layer -> depend on demand(regression - follow whatever you want, classification - binary 0&1, category [encode])
hidden layer
- less qty of layer/neuron size ⇒ small model, more time consuming
- more qty of layer/neuron size ⇒ can solve more complex problems
- neuron size in each hidden layer ⇒ pyramid(historical), equal(modern)
- can see more research e.g. numbers of neuron in hidden should less than 2 times of input

Practice: HW168 Anomaly Classification

- From HW168 datasets, create model for anomaly classification with accuracy up to 90%
- Scoring: $score = (\%accuracy_{val} - 89\%) * 2$

CNN guideline

Feature Extraction

Classification/ Regression

input -> Conv2D -> Pooling -> Flatten -> Dense -> output

initial
Input - Classification(pixel)
224 x 224
256 x 256
128 x 128
20 x 20
512 x 512
YOLO 640 x 640
1280 x 1280

Feature Extraction
- Small img 16,32
- Large img 32,64,128
next layer x2 (normally not over 2 layers)
*Can use transfer learning network with GlobalMaxPooling
--Use lower lr(soft update)

Dense 64, 128,256,512

Practice: HC-SR04 Anomaly Classification

- From HC-SR04 ultrasonic datasets, create model for anomaly classification with accuracy up to 94%
- Scoring: $score = (\%accuracy_{val} - 92\%) * 2$

Mechatronics

Day12

Object Detection

Computer Vision Tasks

classify

Classification

identify area
-object tracking(self-driving car)
-check defect(plant disease)

Semantic Segmentation

classification
+localization
(car driving on the road)

Object Detection

only object

Instance Segmentation

CAT

GRASS, CAT, TREE, SKY

DOG, DOG, CAT

DOG, DOG, CAT

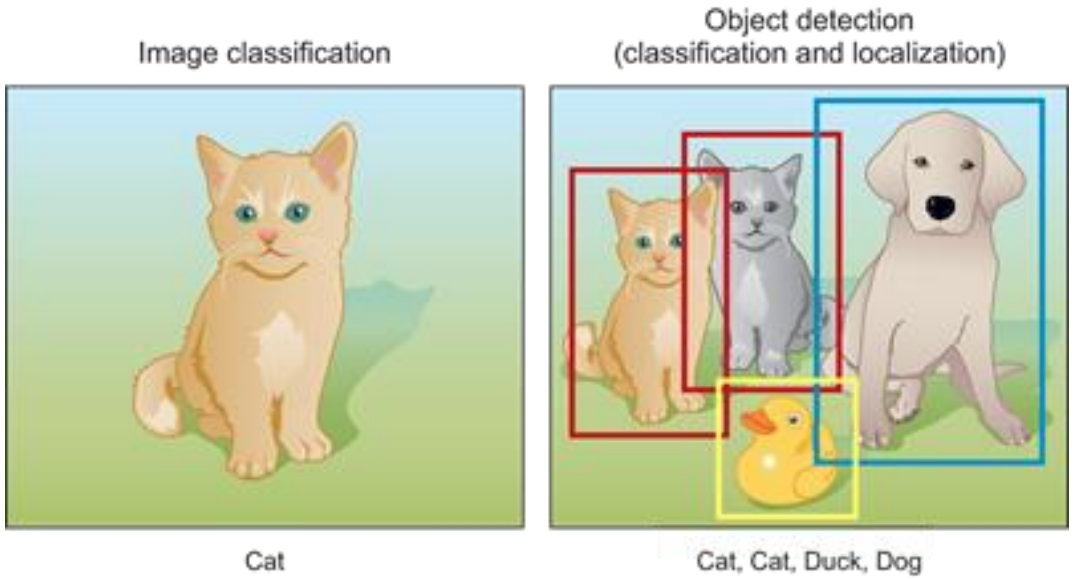
No spatial extent

No objects, just pixels

Multiple Objects

Object detection: Task definition

- **Input**: Single RGB Image
- **Output**: A set of detected objects; For each object predict:
  - Category label (from fixed, known set of categories)
  - Bounding box (four numbers: x, y, width, height)



https://livebook.manning.com/book/deep-learning-for-vision-systems/chapter-7/v-7/

bounding box
- x1,y1 & x2 y2
- x1,y1 & h*w

regression ⇒ find bounding box
classification ⇒ identify object

Object detection: Challenges

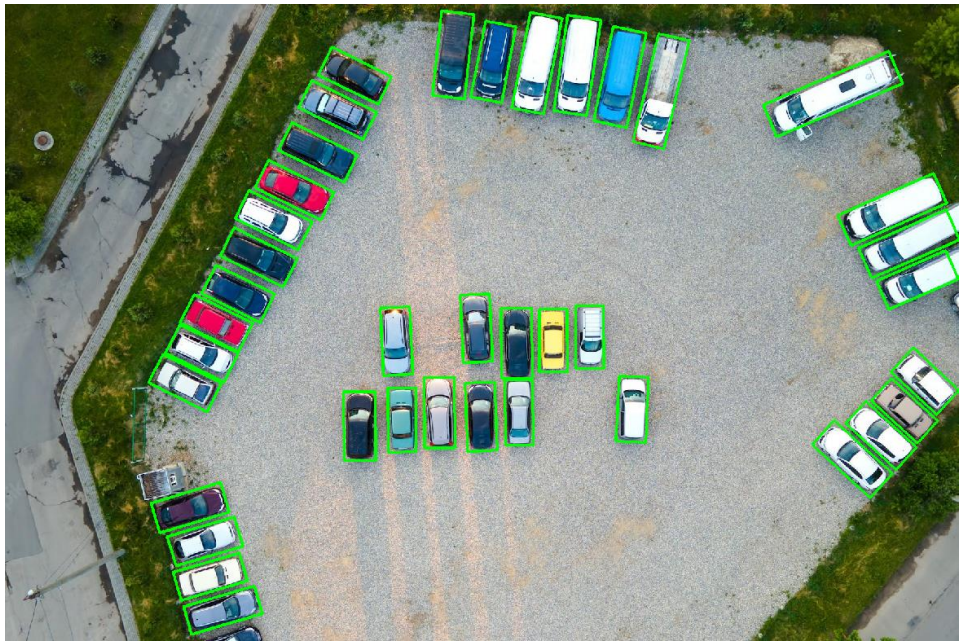
- Multiple outputs: Need to output variable numbers of objects per image
- Multiple types of output: Need to predict "what" (category label) as well as "where" (bounding box)
- Large images: Classification works at 224x224; need higher resolution for detection, often ~800x600

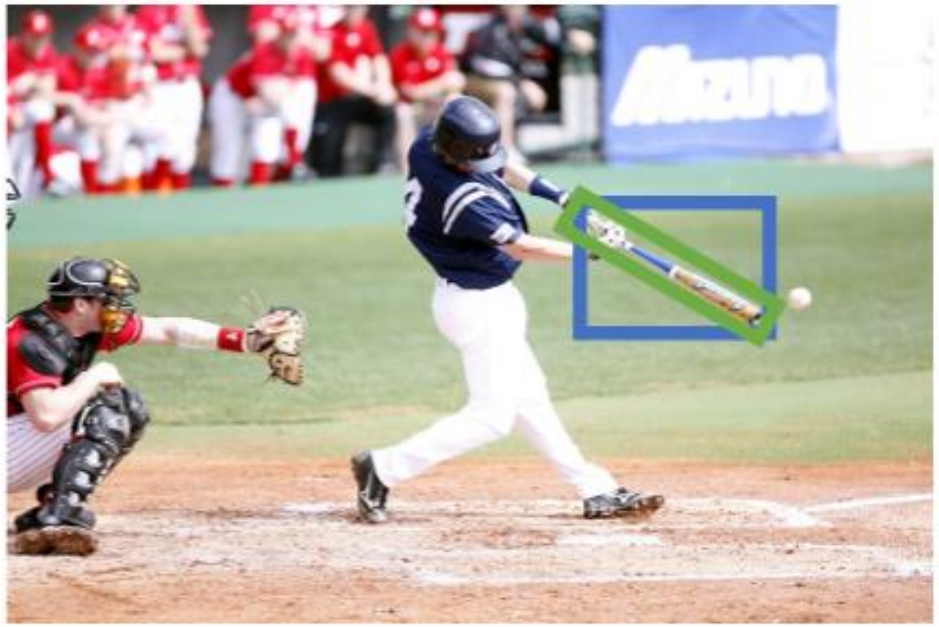
Bounding Boxes

- Bounding boxes are typically axis-aligned
- Oriented boxes are much less common

follow object alignment(Oriented- limited model; YOLO OBB)

require an angle,
e.g. needle meter reading,
bin picking robot





follow global
axis(general)

Modal vs Amodal Boxes

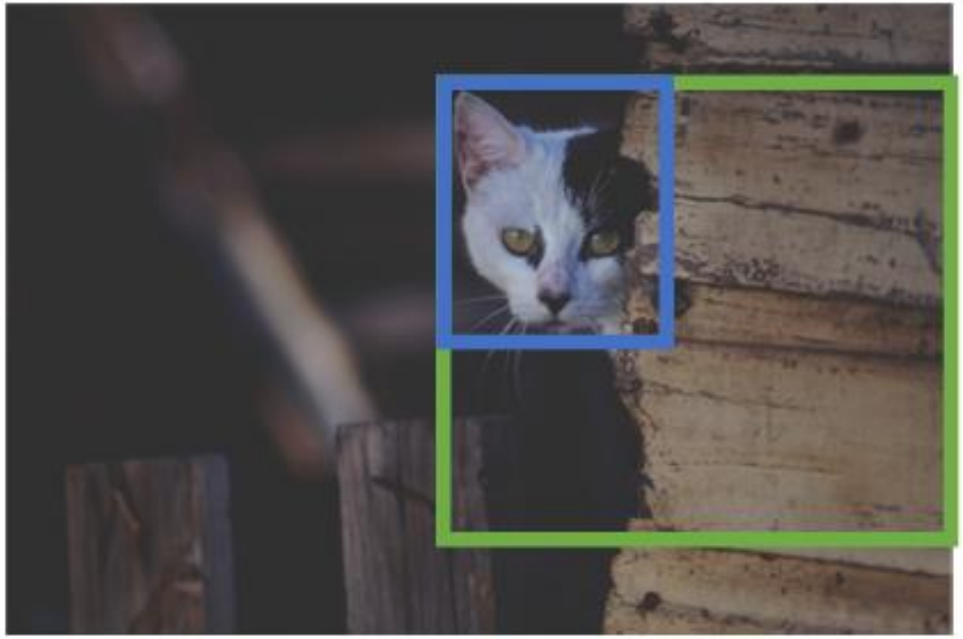
Blue box ⇒ create bounding box only visible portion

- **Modal detection**: Bounding boxes (usually) cover only the visible portion of the object

Green box

- **Amodal detection**: box covers the entire extent of the object, even occluded parts

Can estimate object movement
*with state estimation model(e.g. Kal man Filter)

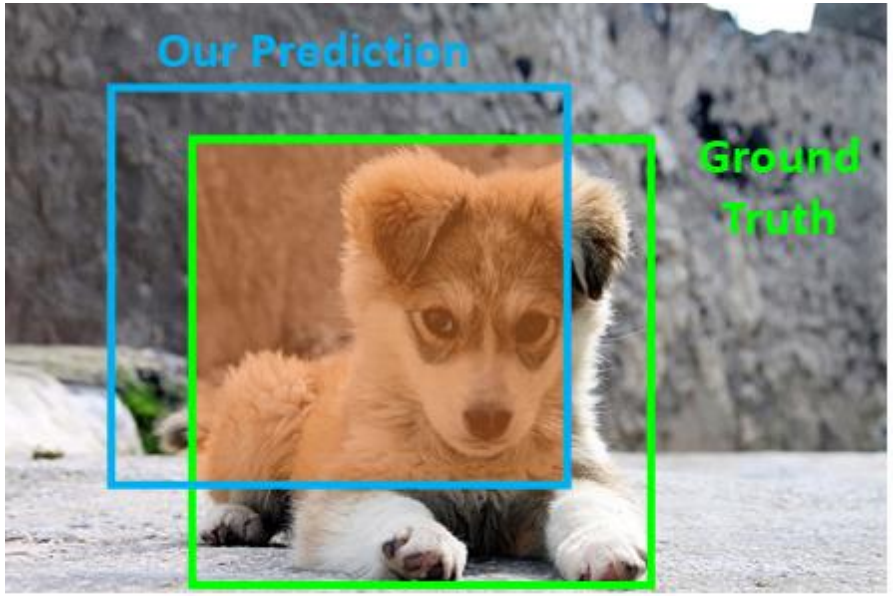


Intersection over Union (IoU)

- Intersection over Union (IoU) (Also called "Jaccard similarity" or "Jaccard index"):

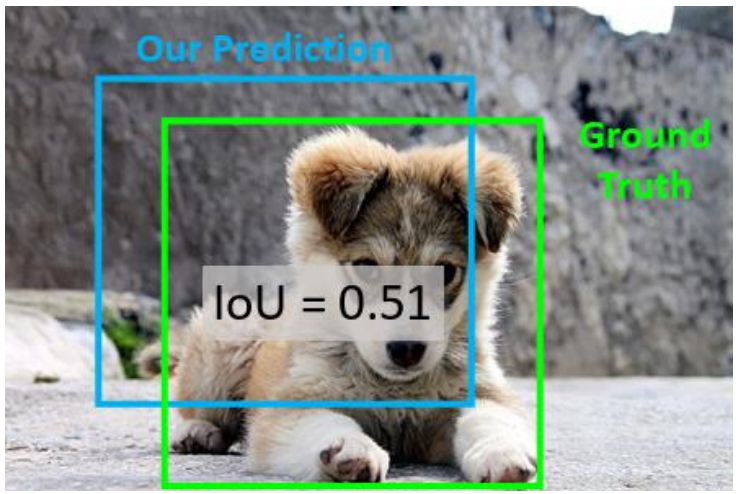
- $IoU = \frac{Area\ of\ Intersection}{Area\ of\ Union}$

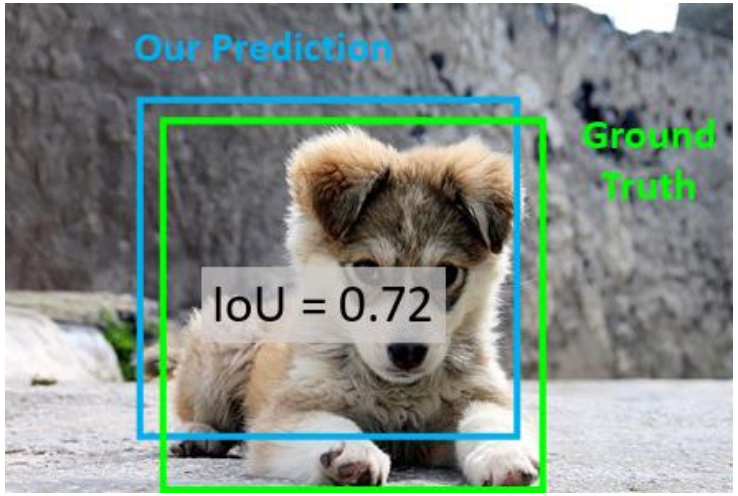
metric for object detection model evaluate
more IoU ⇒ more Precision (accuracy for classification task)

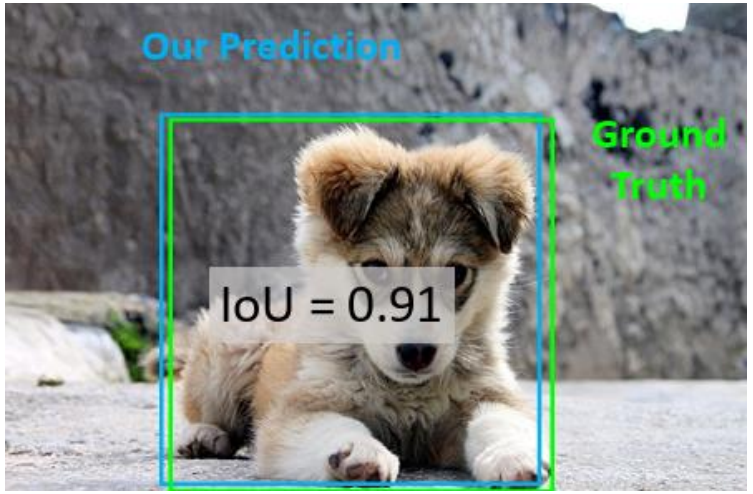


Puppy image is licensed under CC-A 2.0 Generic license. Bounding boxes and text added by Justin Johnson.

Intersection over Union (IoU)





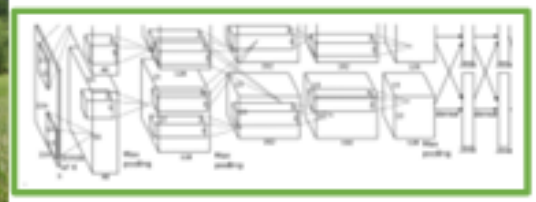


Detecting Single Object

This image is CC0 public domain

feature extraction

Often pretrained on ImageNet (Transfer learning)



Vector: 4096

Treat localization as a regression problem!

Fully Connected: 4096 to 1000

"What"

classification

Class Scores

Cat: 0.9

Dog: 0.05

Car: 0.01

...

Correct label: Cat

class label

Softmax Loss

Multitask Loss

Weighted Sum → Loss

$$L = L_{cls} + \lambda L_{reg}$$

Fully Connected: 4096 to 4

bounding box regression

Box Coordinates (x, y, w, h)

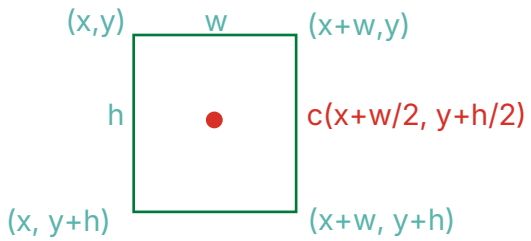
L2 Loss

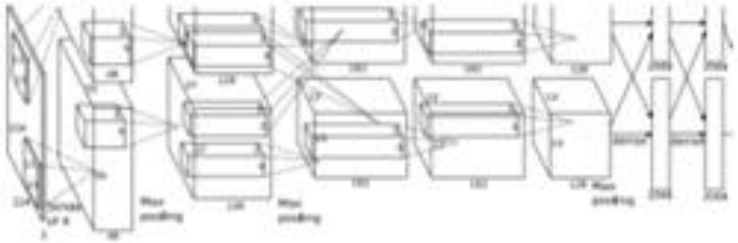
"Where"

Correct box: (x', y', w', h')

location label

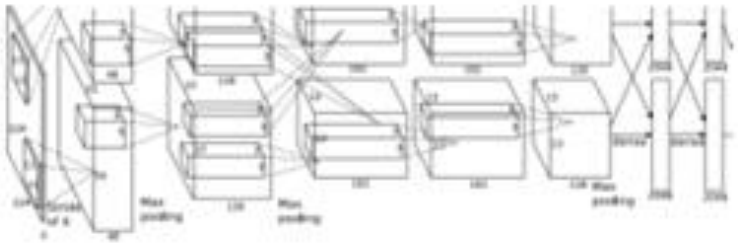
Detecting Multiple Object





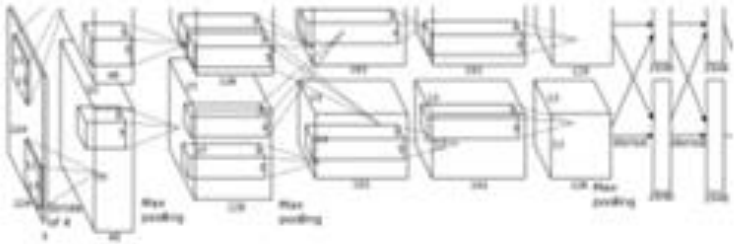
CAT: (x, y, w, h)

4 outputs



DOG: (x, y, w, h)
DOG: (x, y, w, h)
CAT: (x, y, w, h)

12 outputs



DUCK: (x, y, w, h)
DUCK: (x, y, w, h)
....

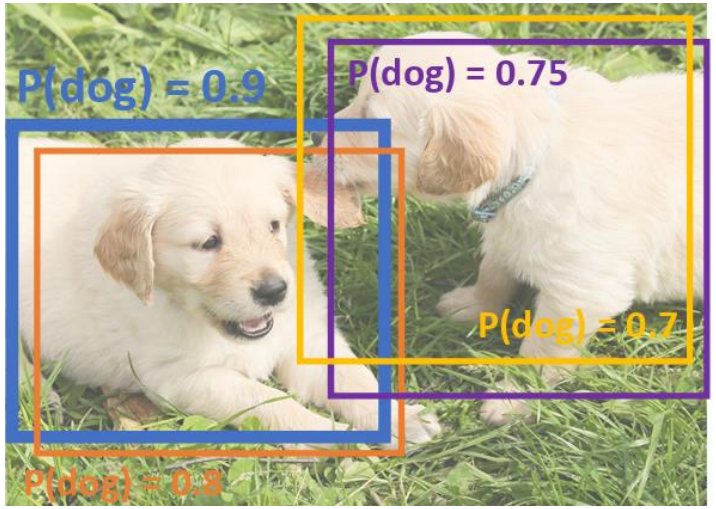
4x outputs

Non-Max Suppression (NMS)

- **Problem**: Object detectors often output many overlapping detections:
- **Solution**: Post-process raw detections using Non-Max Suppression(NMS)

1. Select next highest-scoring box
2. Eliminate lower-scoring boxes with IoU > threshold (e.g. 0.7)
3. If any boxes remain, GOTO 1

high IoU ⇒ select max prob

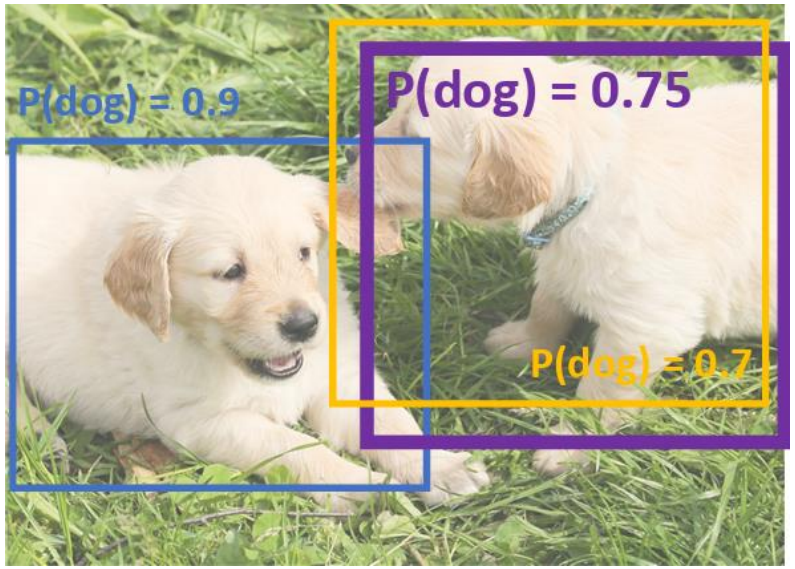


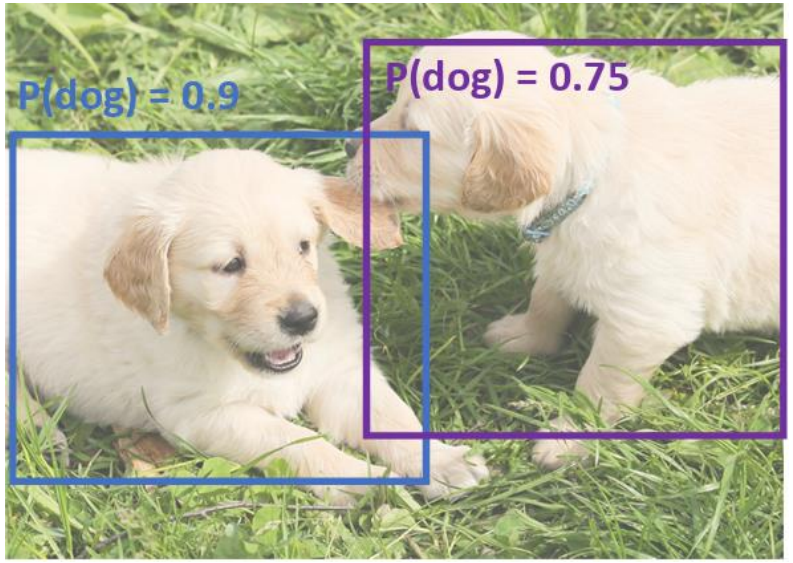
IoU(■, ■) = **0.78**
IoU(■, ■) = 0.05
IoU(■, ■) = 0.07

or consider threshold

Non-Max Suppression (NMS)

- **Problem**: Object detectors often output many overlapping detections:
- **Solution**: Post-process raw detections using Non-Max Suppression(NMS)





IoU(■, ■) = **0.74**

another evaluate metric for object detection

Mean Average Precision (mAP)

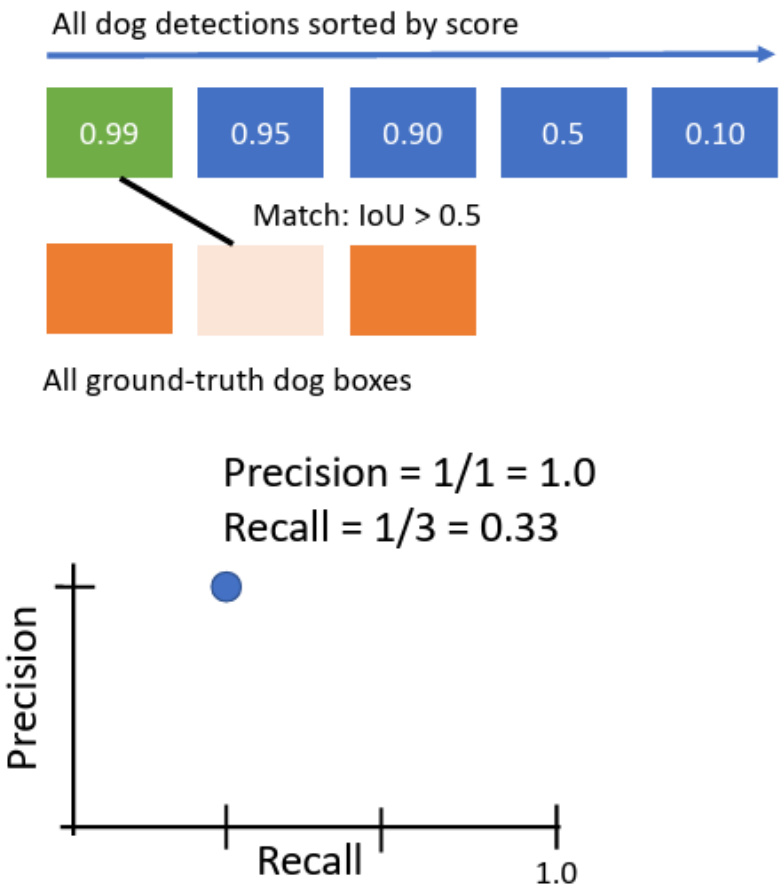
- Run object detector on all test images (with NMS)
- For each category, compute Average Precision (AP) = area under Precision vs Recall Curve

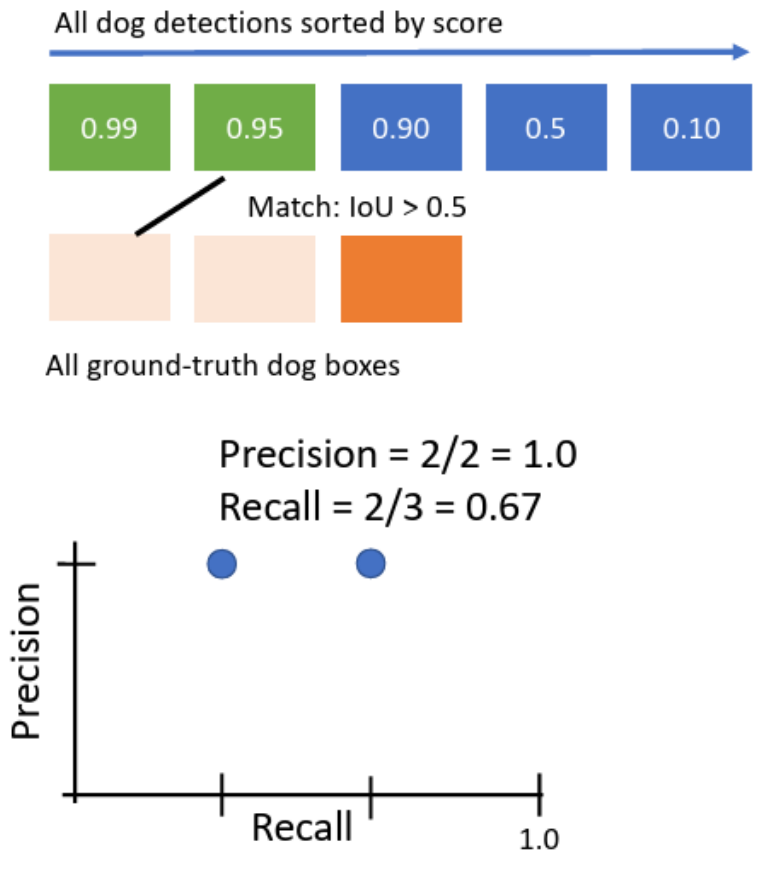
1. For each detection (highest score to lowest score)
   - If it matches some GT box with IoU > 0.5, mark it as positive and eliminate the GT
   - Otherwise mark it as negative
   - Plot a point on PR Curve
2. Average Precision (AP) = area under PR curve

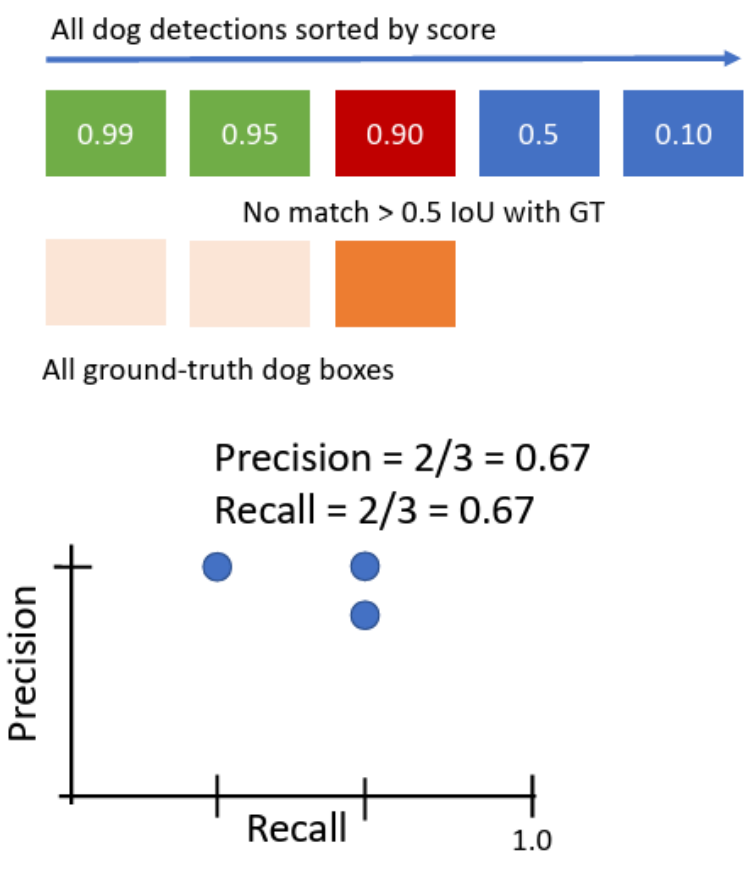
ex - mAP50, mAP70, mAP95 (IoU number)

Mean Average Precision (mAP)

5 bounding box

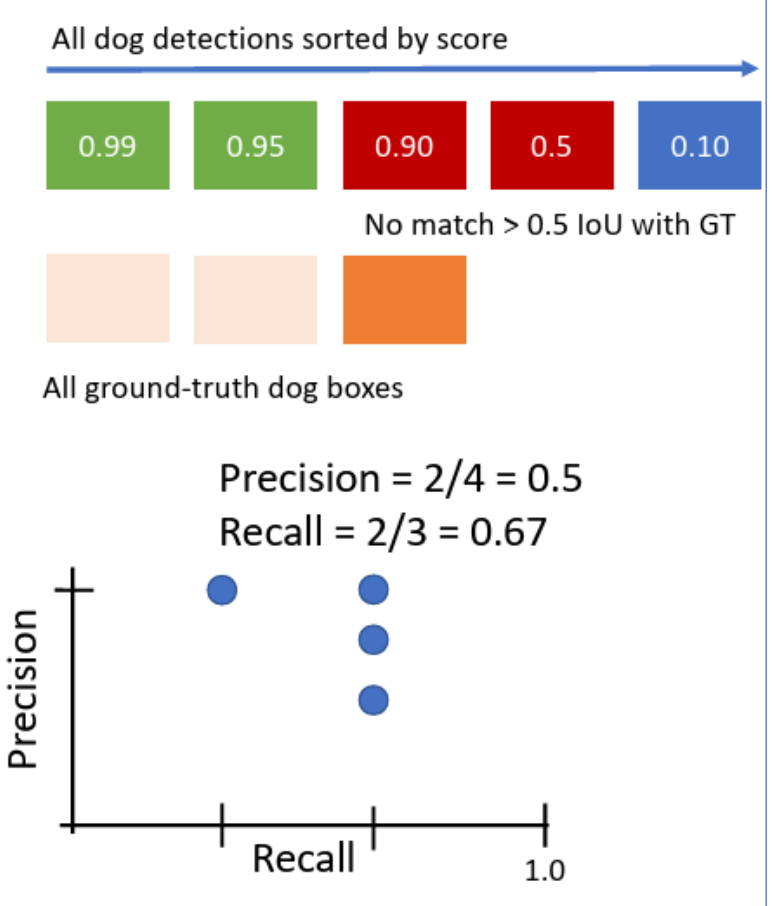


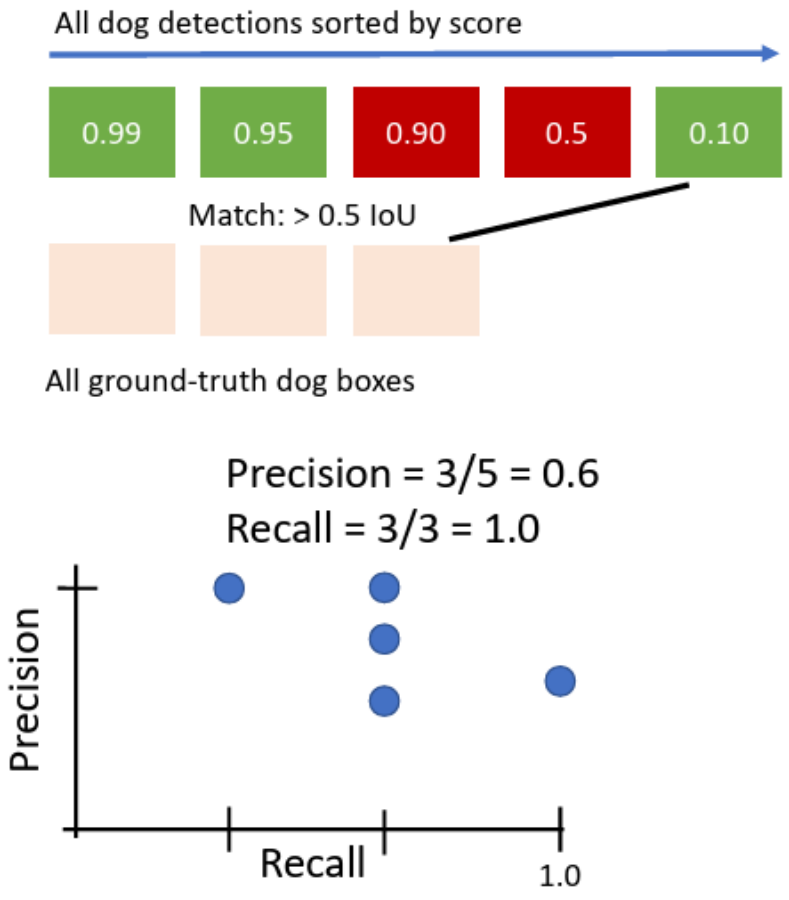


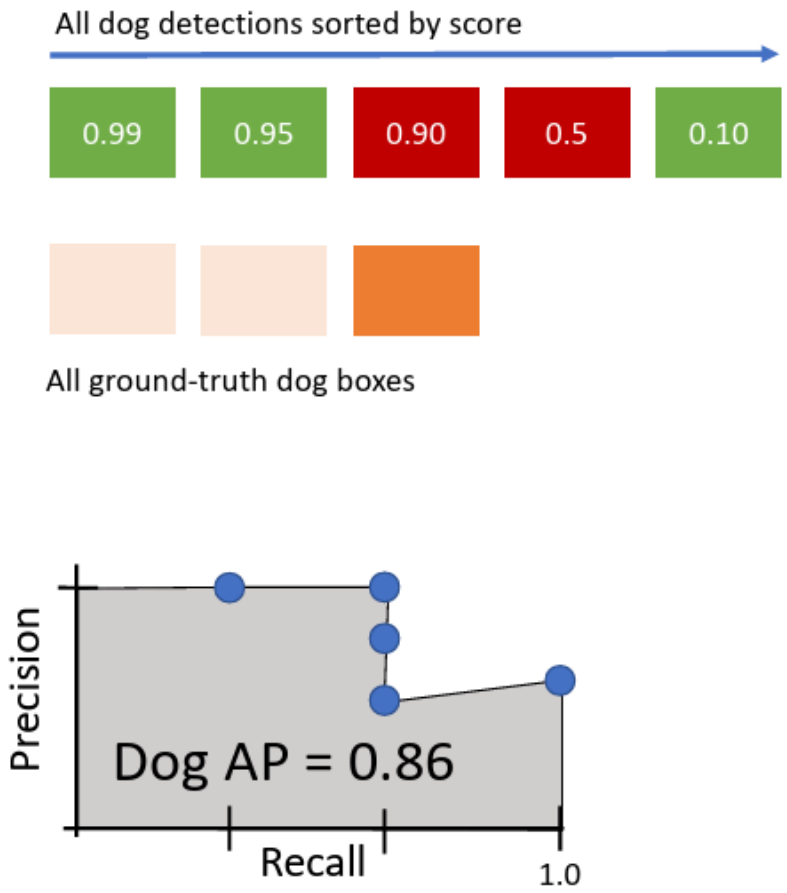


prediction score with IoU > 0.5?

Mean Average Precision (mAP)







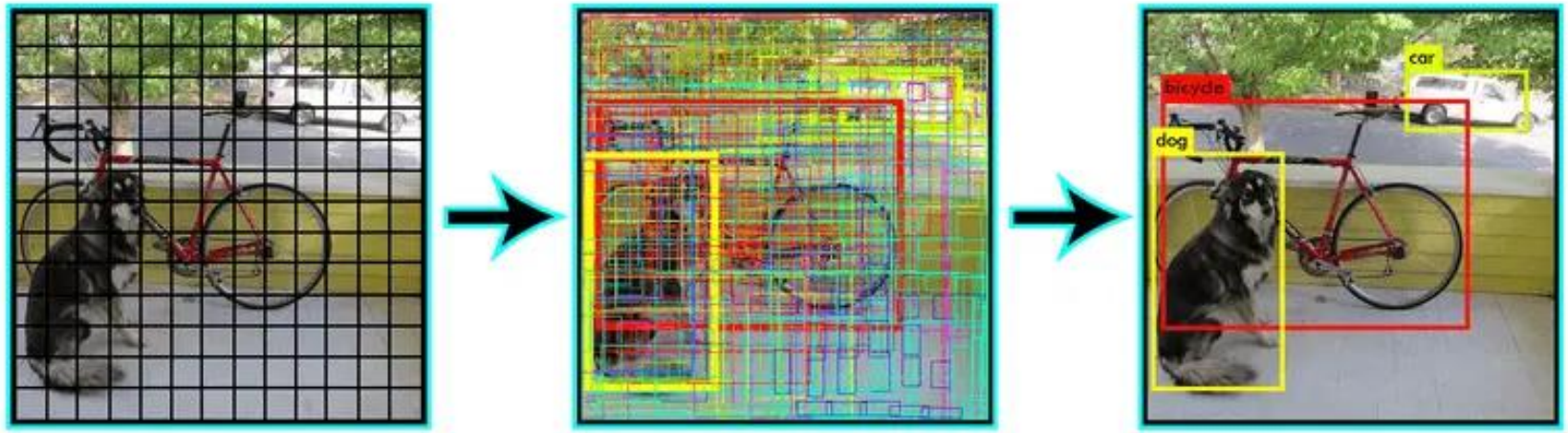
Mean Average Precision (mAP)

- **How to get AP = 1.0?**
- Hit all GT boxes with IoU > 0.5, and have no "false positive" detections ranked above any "true positives"
- **Mean Average Precision (mAP)** = average of AP for each category 4. For "COCO mAP": Compute mAP@threshfor each IoU threshold (0.5, 0.55, 0.6, …, 0.95) and take average

man, animal, things

YOLO Object Detection

- YOLO = You Only Look Once (YOLO)
- Goal of YOLO algorithm is to do object detection both fast **AND** with high accuracy



Key Insight

Previous Approaches

- A separate model for generating bounding boxes and for classification (more complicated model pipeline)
- Need to run classification many times (expensive computation)
- Looks at limited part of the image (lacks contextual information for detection)

YOLO Algorithm

- A single neural network for localization and for classification (less complicated pipeline)
- Need to inference only once (efficient computation)
- Looks at the entire image each time leading to less false positives (has contextual information for detection)

SOTA(State of the art)

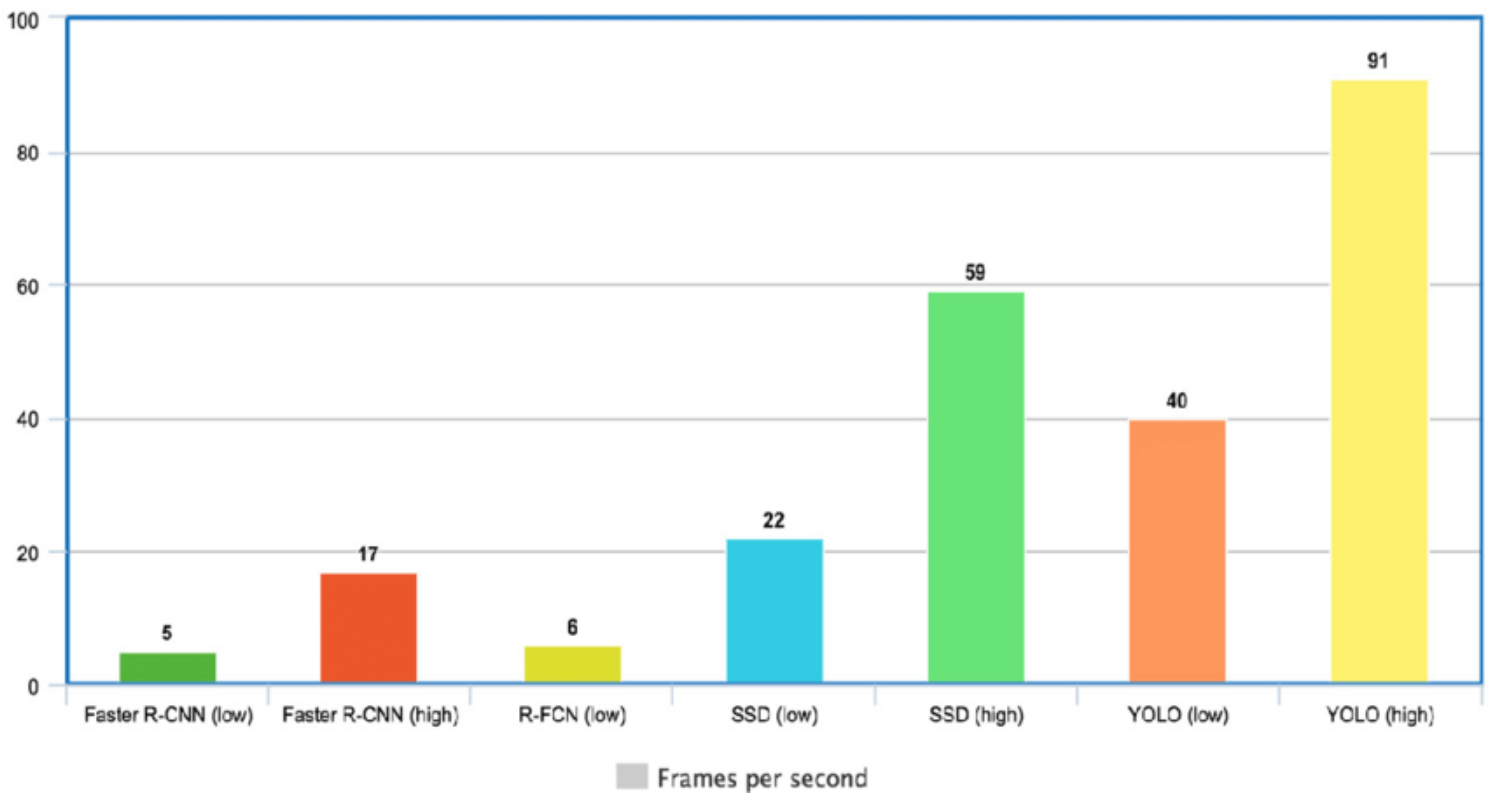
mask R-CNN

double stage detector
more time consumption
more accuracy
*for instance segmentation(not detection)

YOLO

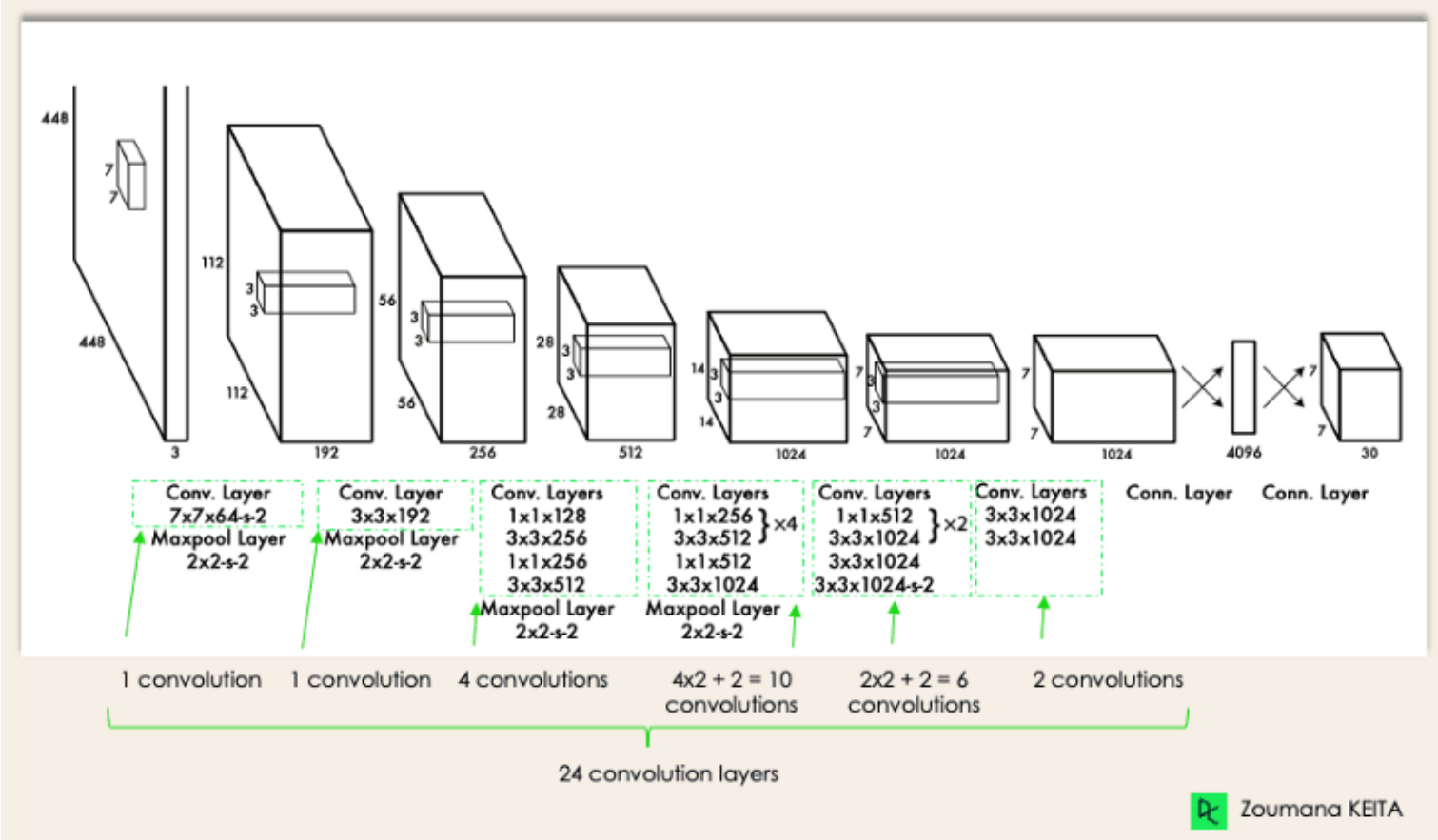
single state detector. faster but less acc -> real time object detection

- Comparison of FPS between YOLO and each Neural Network Based Object Detection approaches.



https://www.datacamp.com/blog/yolo-object-detection-explained

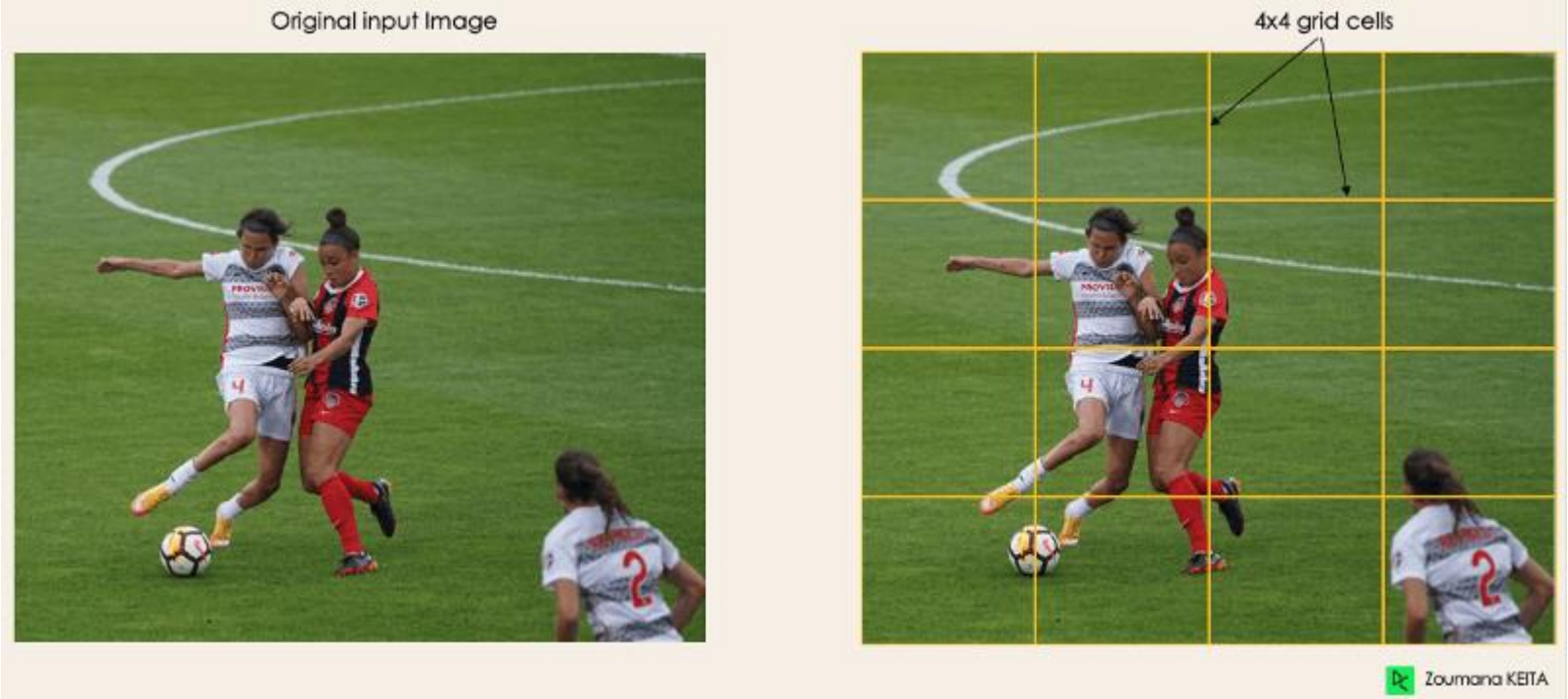
YOLO Architecture



YOLO Architecture (Joseph Redmon et al. , 2016)

How does YOLO work?

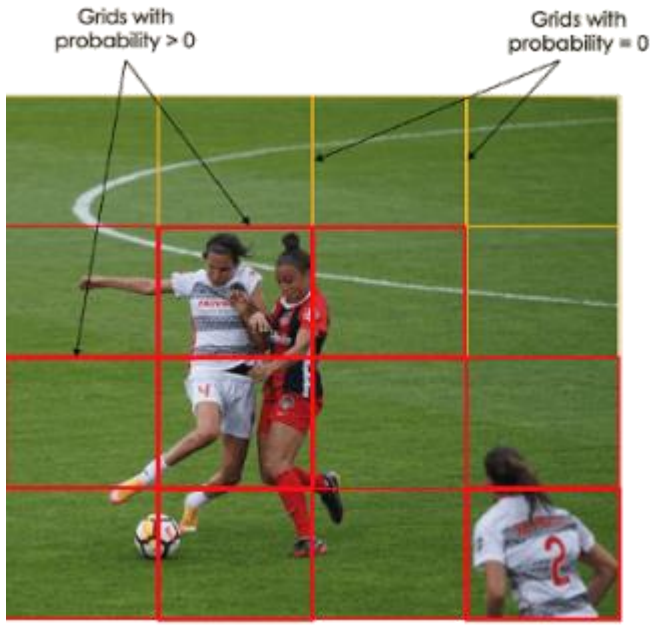
- **Residual Blocks** : YOLO will divide the image into NxN grid cells and each cell is responsible for localizing and predicting the class of object with the probability or confidence value (Pr (object) * IoU)

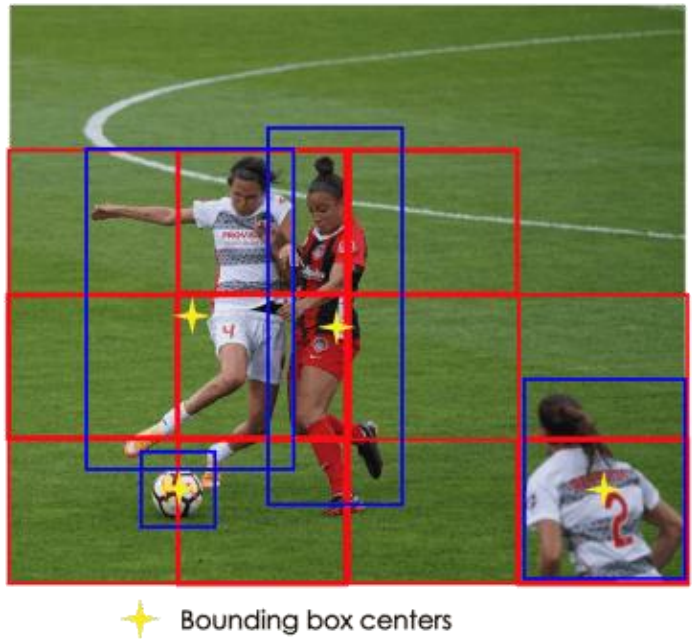


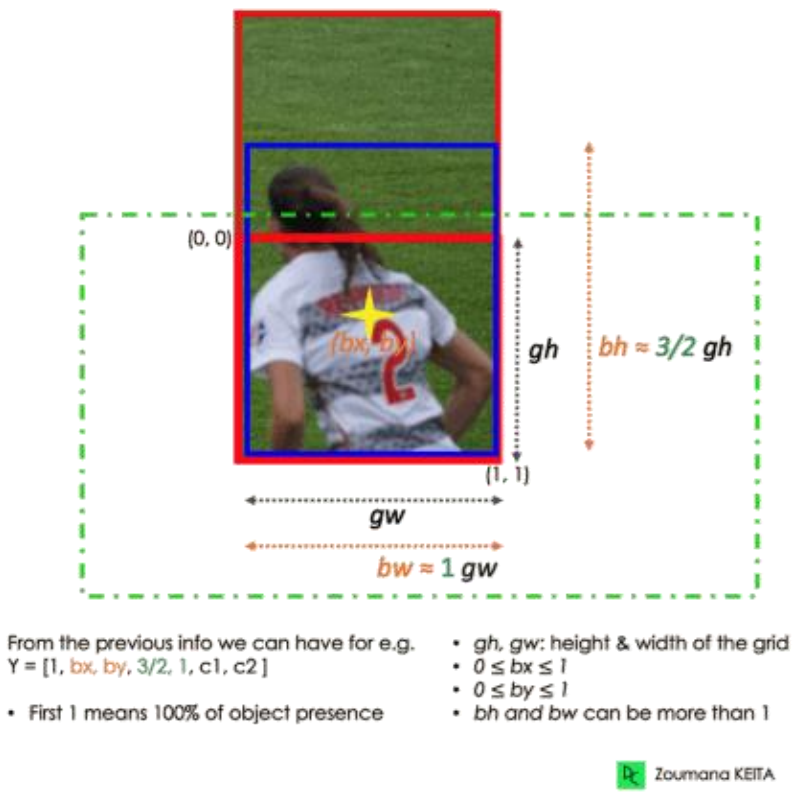
https://www.datacamp.com/blog/yolo-object-detection-explained

How does YOLO work?

- **Bounding box regression** : YOLO uses a single bounding box regression to predict the height, width, center, and class of objects

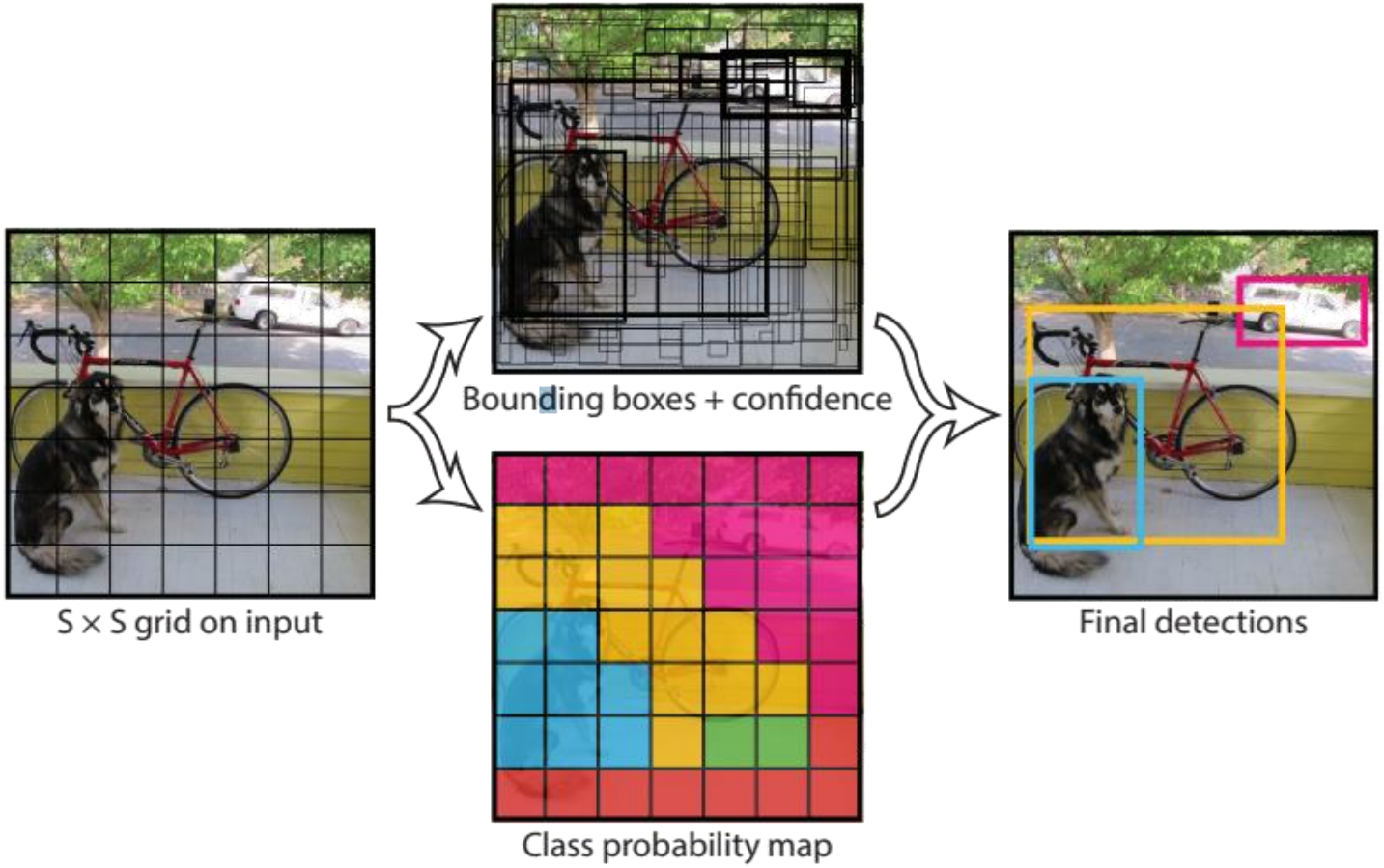






https://www.datacamp.com/blog/yolo-object-detection-explained

How does YOLO work?



The Model (Joseph Redmon et al. , 2016)

YOLO Greatest Hits

Jun 8, 2015 — **YOLO is introduced**
Joseph Redmon et al / UW
You Only Look Once: Unified, Real-Time Object Detection

Dec 25, 2016 — **YOLOv2 aka YOLO9000**
Joseph Redmon et al / UW
YOLO9000: Better, Faster, Stronger

Apr 8, 2018 — **YOLOv3**
Joseph Redmon et al / UW
YOLOv3: An Incremental Improvement

Apr 23, 2020 — **YOLOv4**
Alexey Bochkovskiy et al
YOLOv4: Optimal Speed and Accuracy of Object Detection

Jun 9, 2020† — **YOLOv5**
Glenn Jocher et al / Ultralytics
YOLOv5 GitHub

Jul 23, 2020* — **PP-YOLO**
X Long et al / Baidu
An Effective and Efficient Implementation of Object Detector

Nov 16, 2020* — **Scaled-YOLOv4**
Chuyi Li, Alexey Bochkovskiy et al
Scaled-YOLOv4: Scaling Cross Stage Partial Network

May 10, 2021 — **YOLOR**
Chien-Yao Wang et al
You Only Learn One Representation: Unified Network for Multiple Tasks

Jun 1, 2021 — **YOLOS**
Yuxin Fang et al / HUST
You Only Look at One Sequence: Rethinking Transformer in Vision through Object Detection

Jul 18, 2021* — **YOLOX**
Zheng Ge et al / Megvii
YOLOX: Exceeding YOLO Series in 2021

Jun 2022† — **YOLOv6**
Chuyi Li et al / Meituan
YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications

Jul 6, 2022 — **YOLOv7**
Chien-Yao Wang, Alexey Bochkovskiy et al
YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors

Jan 10, 2023† — **YOLOv8**
Glenn Jocher et al / Ultralytics
YOLOv8 GitHub

Jan 13, 2023 — **YOLOv6 3.0**
Chuyi Li et al / Meituan
YOLOv6 v3.0: A Full-Scale Reloading

May 2, 2023† — **YOLO-NAS**
Shay Aharon et al / Deci
YOLO-NAS: A Next-Generation, Object Detection Foundational Model generated by Deci's Neural Architecture Search Technology

Jan 30, 2024* — **YOLO-World**
Tianheng Cheng et al / Tencent
YOLO-World: Real-Time Open-Vocabulary Object Detection

Feb 21, 2024* — **YOLOv9**
Chien-Yao Wang et al
YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information

May 23, 2024 — **YOLOv10**
Ao Wang et al / Tsinghua Univ
YOLOv10: Real-Time End-to-End Object Detection

Sep 30, 2024† — **YOLO 11**
Glenn Jocher et al / Ultralytics
YOLOv11 GitHub

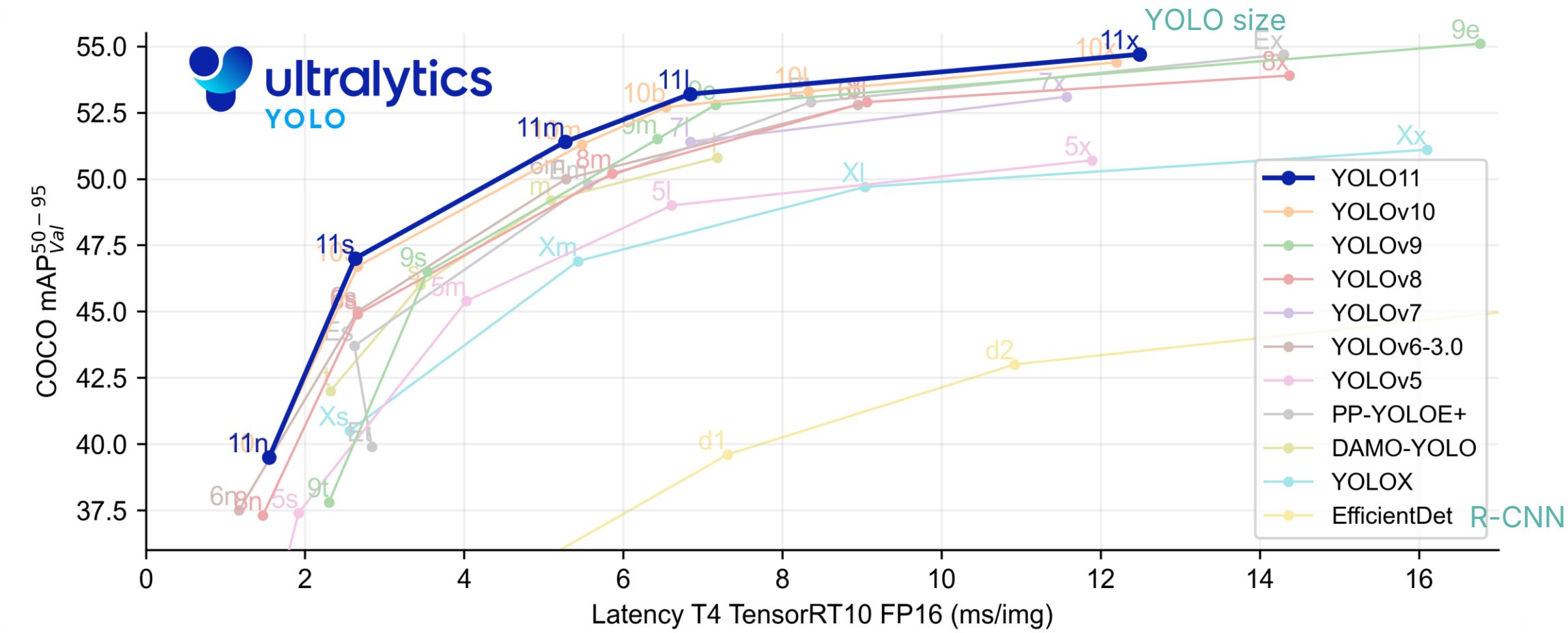
*Denotes paper updated after first publication date
†Denotes repository predates paper publication date

V11 detect+segment perf comparable to R-CNN

Yolo Performance

n nano
s small
m medium
l large
x xlarge

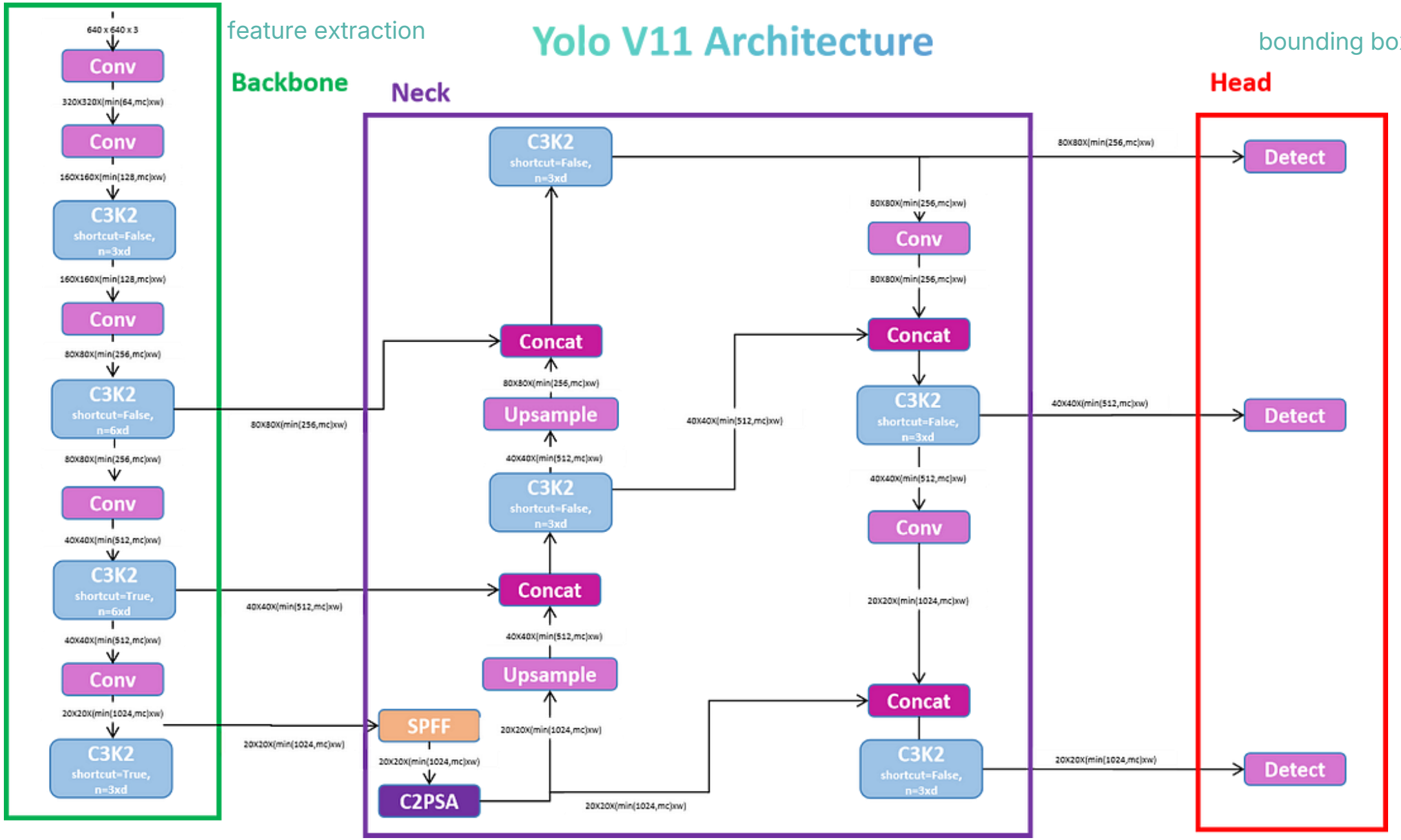
consider cycle time and flow down to process time



Yolo Performance

Method		Input Size	Backbone	Parameters (Millions)	FPS	G-FLOPs (G)	mAP (%)
SSD		300 × 300	VGG16	24.54	29	61.45	0.361
		300 × 300	Mobilenetv2	4.47	35	1.53	0.328
YOLOv3		416 × 416	Darknet-53	61.56	27	65.65	0.422
		416 × 416	Efficient-B0	7.02	21	3.84	0.435
YOLOv4		416 × 416	CSPDark-53	63.98	19	60.01	0.517
		416 × 416	Mobilenetv2	39.06	28	29.74	0.461
YOLOv5	S	640 × 640	CSPDarknet53 + SPP	7.08	36	16.54	0.352
	M	640 × 640	CSPDarknet53 + SPP	21.09	23	50.69	0.397
	L	640 × 640	CSPDarknet53 + SPP	46.67	15	114.68	0.416
YOLOX	Tiny	640 × 640	Modified CSP	5.03	31	15.24	0.390
	S	640 × 640	Modified CSP	8.94	31	26.77	0.404
	M	640 × 640	Modified CSP	25.29	20	73.75	0.492
	L	640 × 640	Modified CSP	54.15	14	155.70	0.539
Faster-RCNN		600 × 600	VGG16	136.83	8	369.89	0.475
		600 × 600	ResNet50	28.35	7	941.01	0.499
EfficientDet	D0	512 × 512	Efficient-B0	3.83	13	4.78	0.328
	D1	640 × 640	Efficient-B1	6.56	10	11.59	0.404
	D2	768 × 768	Efficient-B2	8.01	9	20.71	0.487
	D3	896 × 896	Efficient-B3	11.91	7	47.23	0.503
	D4	1024 × 1024	Efficient-B4	20.56	4	105.55	0.552
Our Approach		416 × 416	Ghost module	11.04	31	6.63	0.569

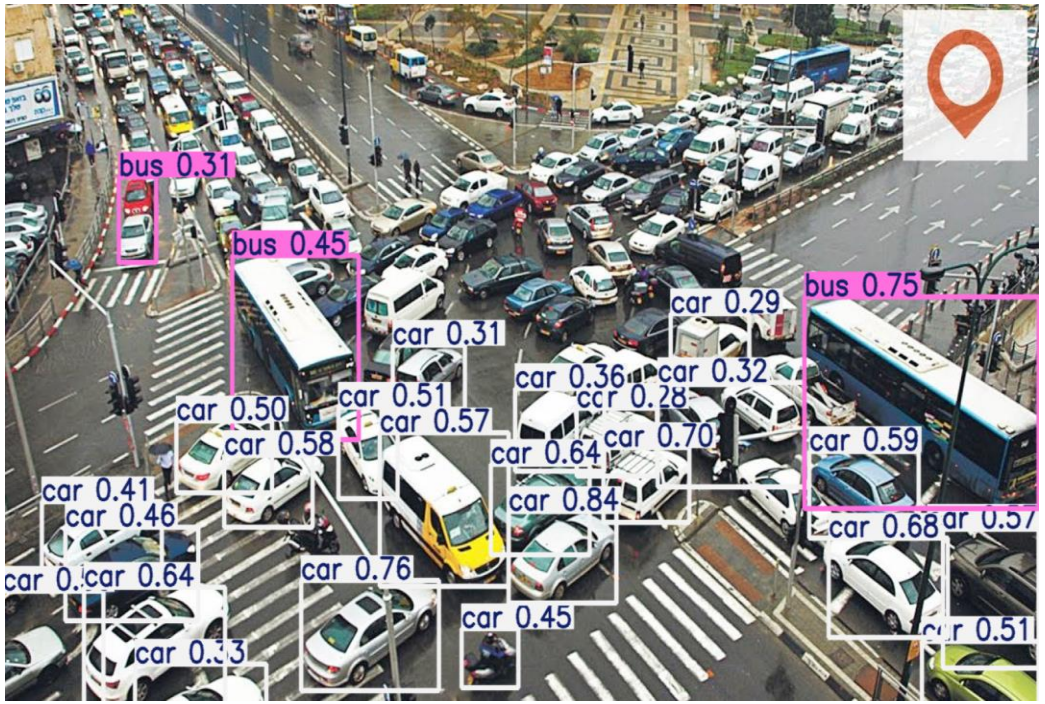
Yolo V11 Architecture



YoloV11 Testing

- Install: pip install ultralytics
- Test with sample code

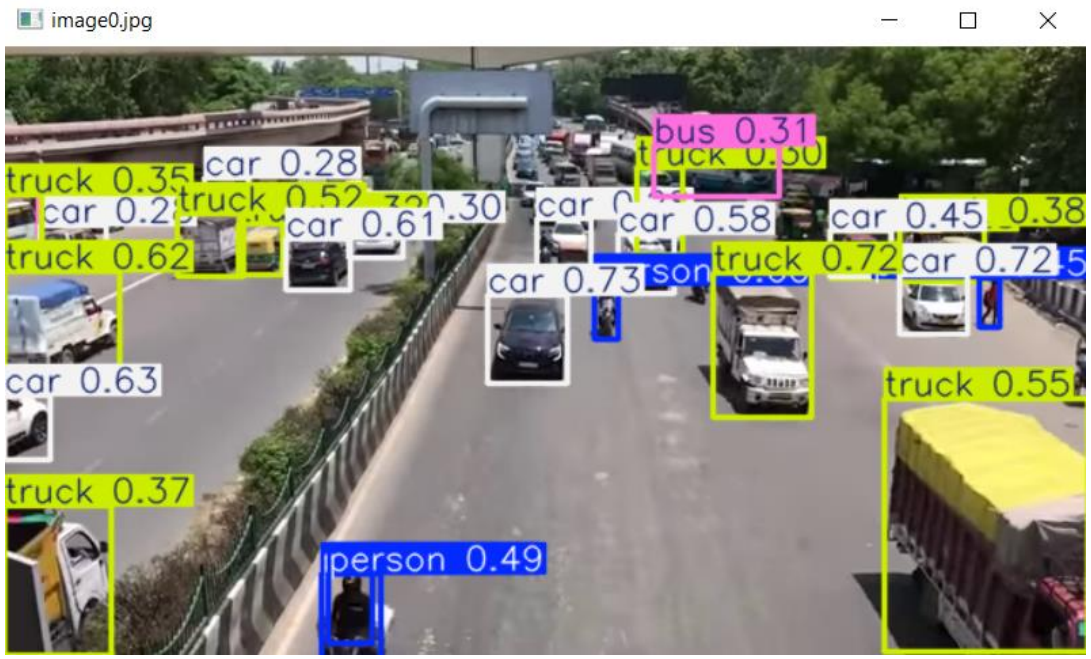
```python
from ultralytics import YOLO
import cv2

model = YOLO("yolo11n.pt")

results = model("traffic.jpg", show = True)
cv2.waitKey(0)
```



COCO - Common Objects in Context

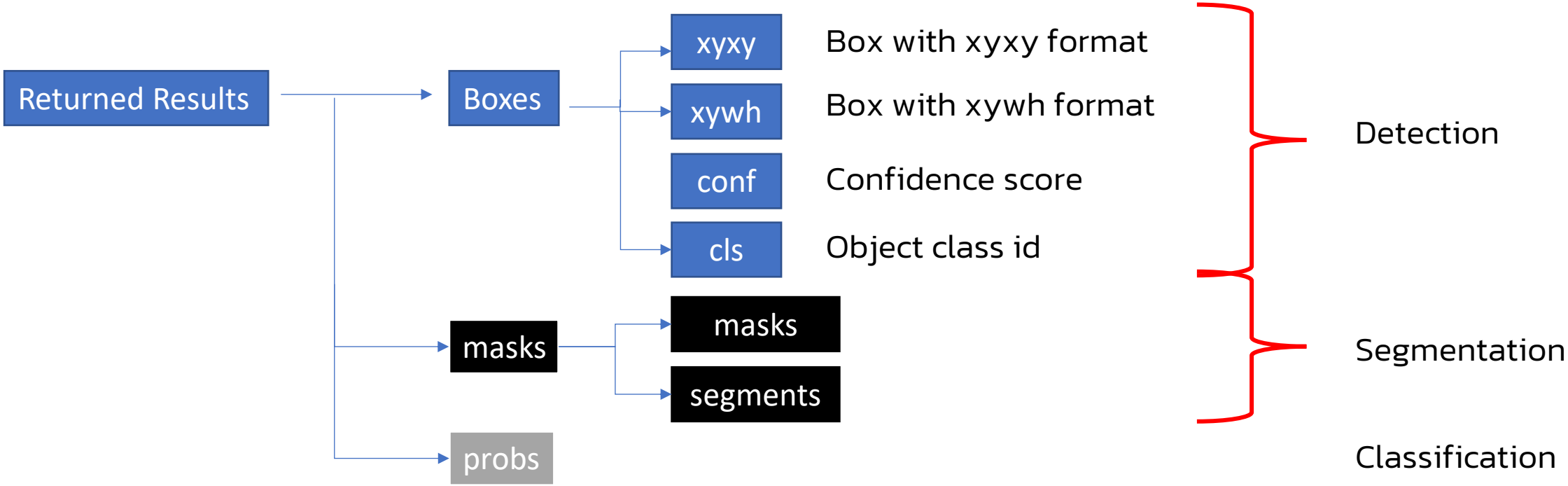
Video Object Detection

```python
from ultralytics import YOLO
import cv2

vid = cv2.VideoCapture("car_video.mp4")
model = YOLO("yolo11n.pt")

while True:
    ret,frame = vid.read()
    results = model(frame, show = True)
    cv2.waitKey(1)
```



YOLO result structure

- Write a code to print result from YOLO detector ana explore more details.



EX10: Finding the centroid of object.

ex10_intro_yolo.py > ...

```python
from ultralytics import YOLO
import cv2

vid = cv2.VideoCapture("car_video.mp4")
model = YOLO("yolo11n.pt")

while True:
    ret,frame = vid.read()
    results = model(frame, show = False)
    for result in results:
        boxes = result.boxes.xyxy.cpu().numpy()
        for box in boxes:
            b = box.tolist()
            x1,y1,x2,y2 = b
            cx = int((x1+x2)/2)
            cy = int((y1+y2)/2)
            cv2.circle(frame, (cx, cy), 5, (0, 0, 255), -1)
    cv2.waitKey(1)
    cv2.imshow("Results", frame)
```

EX11: Count the object with line.

- Count the number of the bicycle that pass the line.

ex11_count_object.py > ...

```python
from ultralytics import YOLO
import cv2

vid = cv2.VideoCapture("car_video.mp4")
model = YOLO("yolo11n.pt")

line_y = 300
object_count = 0

while True:
    ret,frame = vid.read()
    results = model(frame, show = False)

    for result in results:
        boxes = result.boxes.xyxy.cpu().numpy()
        classes = result.boxes.cls.cpu().numpy()

        for box, clsid in zip(boxes, classes):
            b = box.tolist()
            x1,y1,x2,y2 = b

            if clsid == 3:
                cx = int((x1+x2)/2)
                cy = int((y1+y2)/2)
                cv2.circle(frame, (cx, cy), 5, (0, 0, 255), -1)

                if cy > line_y - 5 and cy < line_y + 5:
                    object_count += 1
    cv2.line(frame, (0, line_y), (frame.shape[1], line_y),
                (255, 0, 0), 2)
    cv2.putText(frame, f"Count: {object_count}", (50, 50),
                cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 255), 2)
    cv2.waitKey(1)
    cv2.imshow("Results", frame)
```

EX11: Count the object with line.

- Count the number of the bicycle that pass the line.

SUT

SURANAREE UNIVERSITY OF TECHNOLOGY

INSTITUTE OF

ENGINEERING

Mechatronics

Modern Automotive