Big Data and Data Mining

Image Classification

Fenerbahce University

Instructors

Assist. Prof. Vecdi Emre Levent

Office: 311

Email : emre.levent@fbu.edu.tr

Agenda

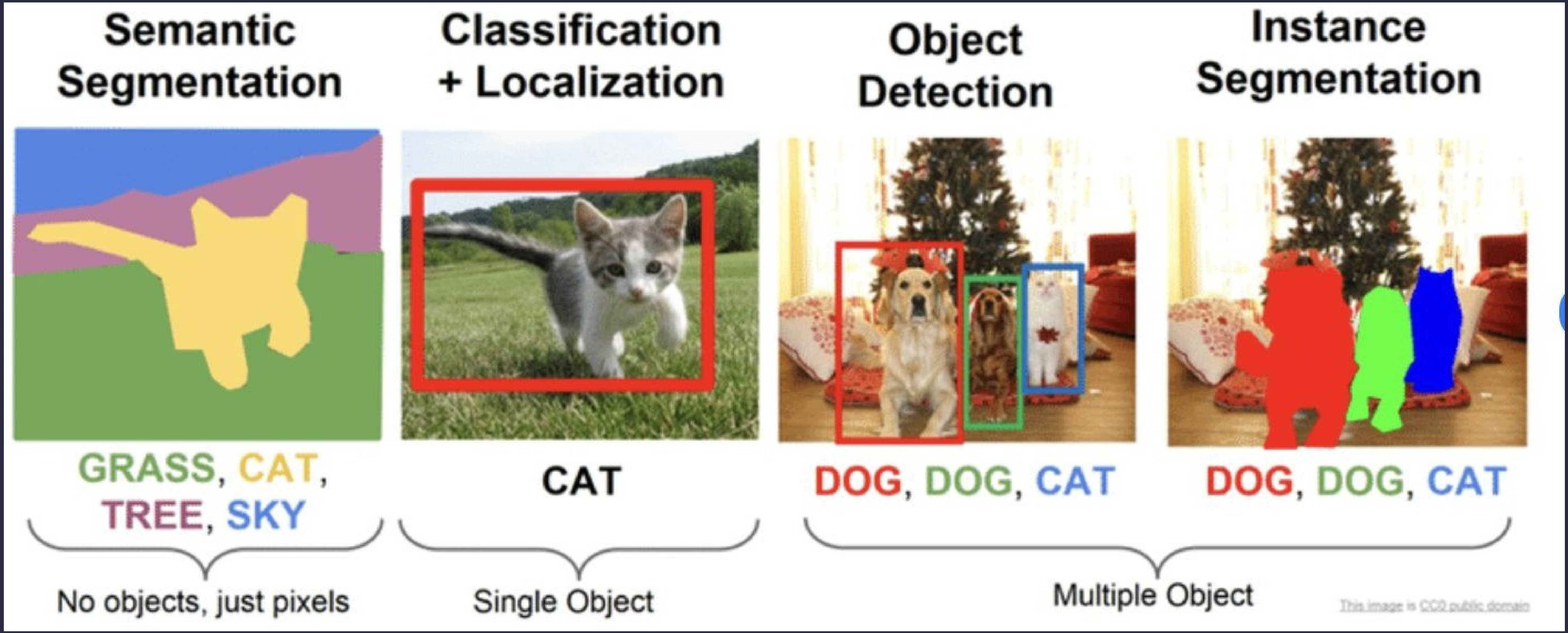
1. Introduction to Image Segmentation.
    a. Problem definition
    b. Standard Datasets
2. Why YOLO
3. Various versions of YOLO
4. Image segmentation - SOTAArchitectures

Agenda

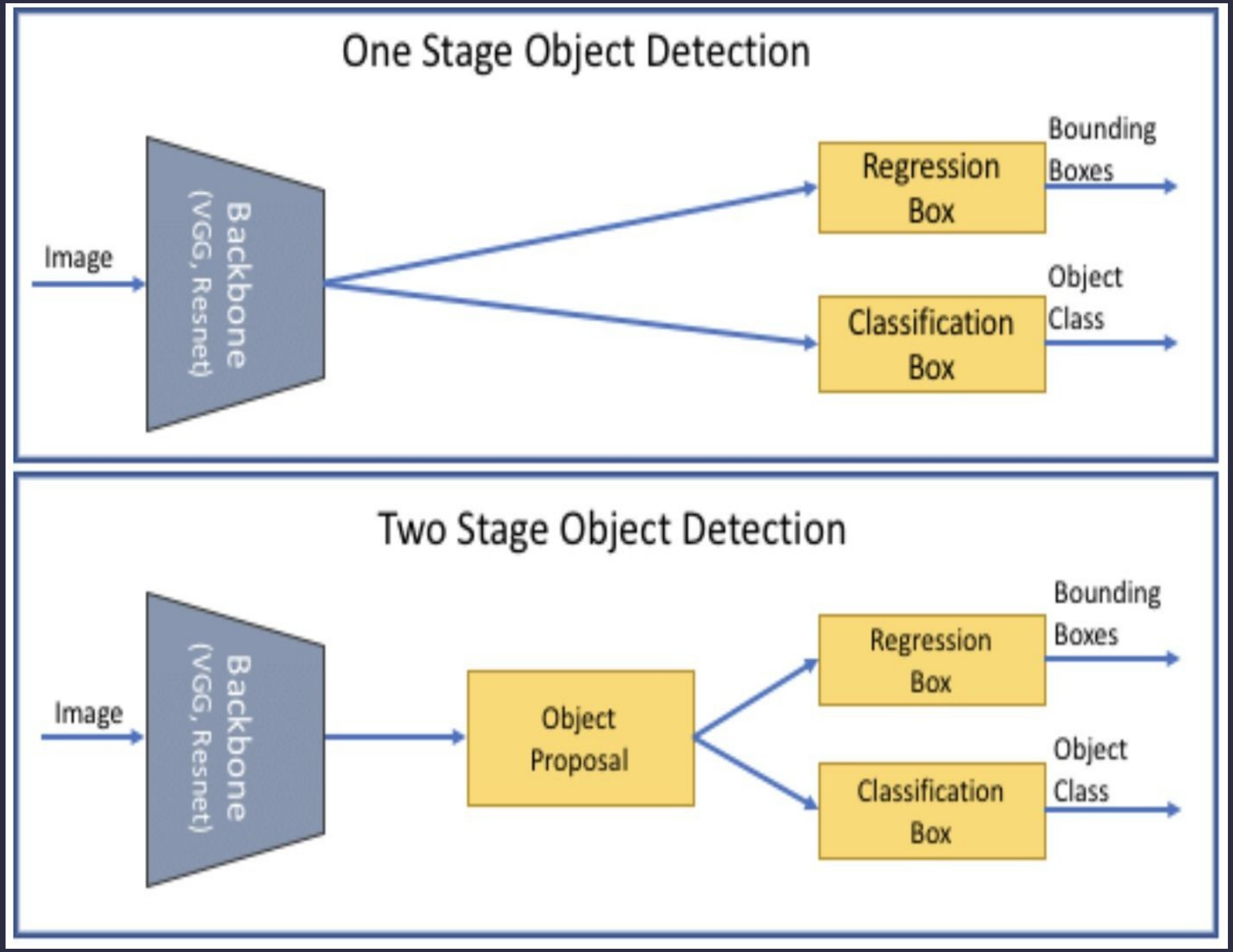
1. Introduction to Image Processing.
    a. Problem definitions -
        i. Object detection
        ii. Image Segmentation
    b. Standard Datasets
2. R-CNN Family
3. **YOLO Family**
    a. Various versions of YOLO (v1 - v7)
4. Image Processing - SOTAArchitectures

Key Idea - Understanding the journey behind refining models for better speed and/or accuracy for a particular task

Problem Definition



Object Detection



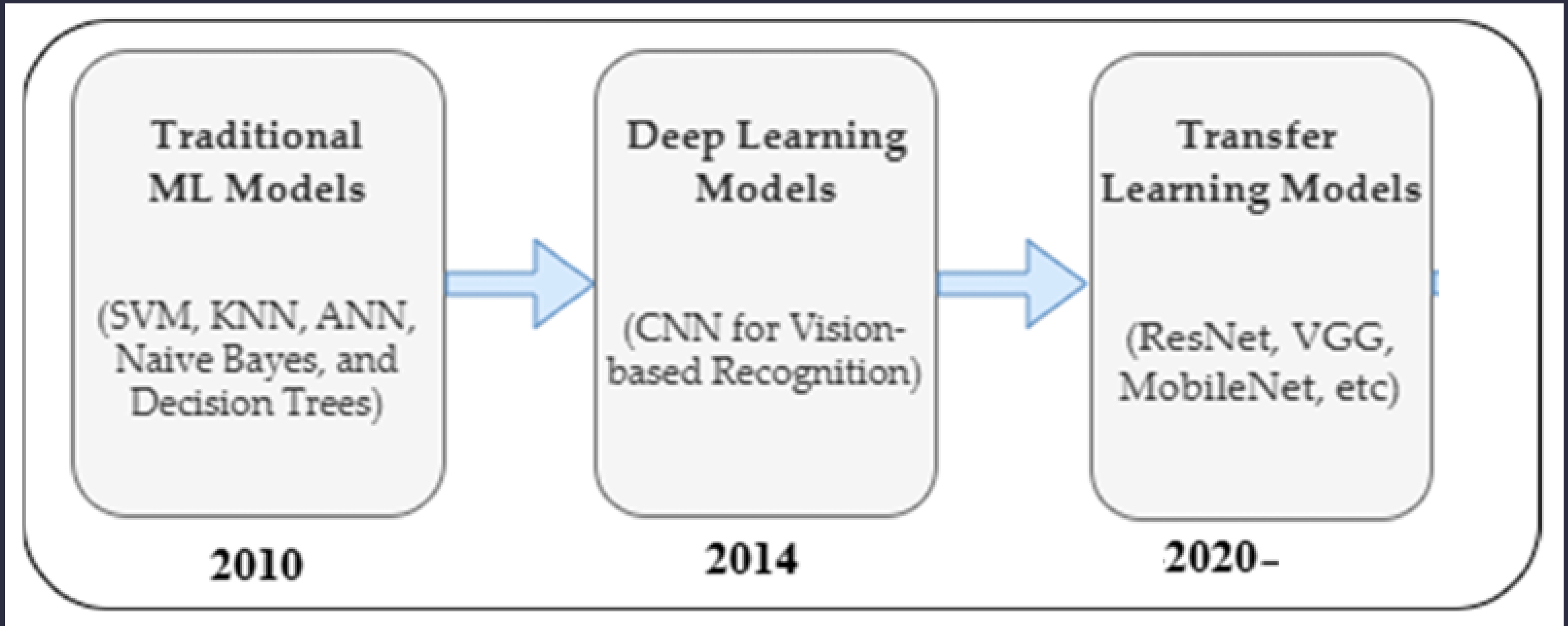
- Classifying multiple instances of objects and localizing within an image
- One-Stage Methods - Better inference speed - YOLO, SSD and RetinaNet
- Two-Stage Methods - Better Accuracy - Faster R-CNN, Mask R-CNN and Cascade R-CNN

Datasets

- ImageNet - Large dataset containing annotated images based on WordNet's hierarchical structure
- PASCAL VOC - Pattern Analysis, Statistical Modelling and Computational Learning - Visual Object Classes Dataset
- MS COCO - Microsoft Common Objects in Context
- STL-10 : Subset of ImageNet with 10 Classes. Also has unlabeled images
- CIFAR-x : x defines the number of classes

All of these can be used for object detection, segmentation, dense pose estimation, key point detection, etc.

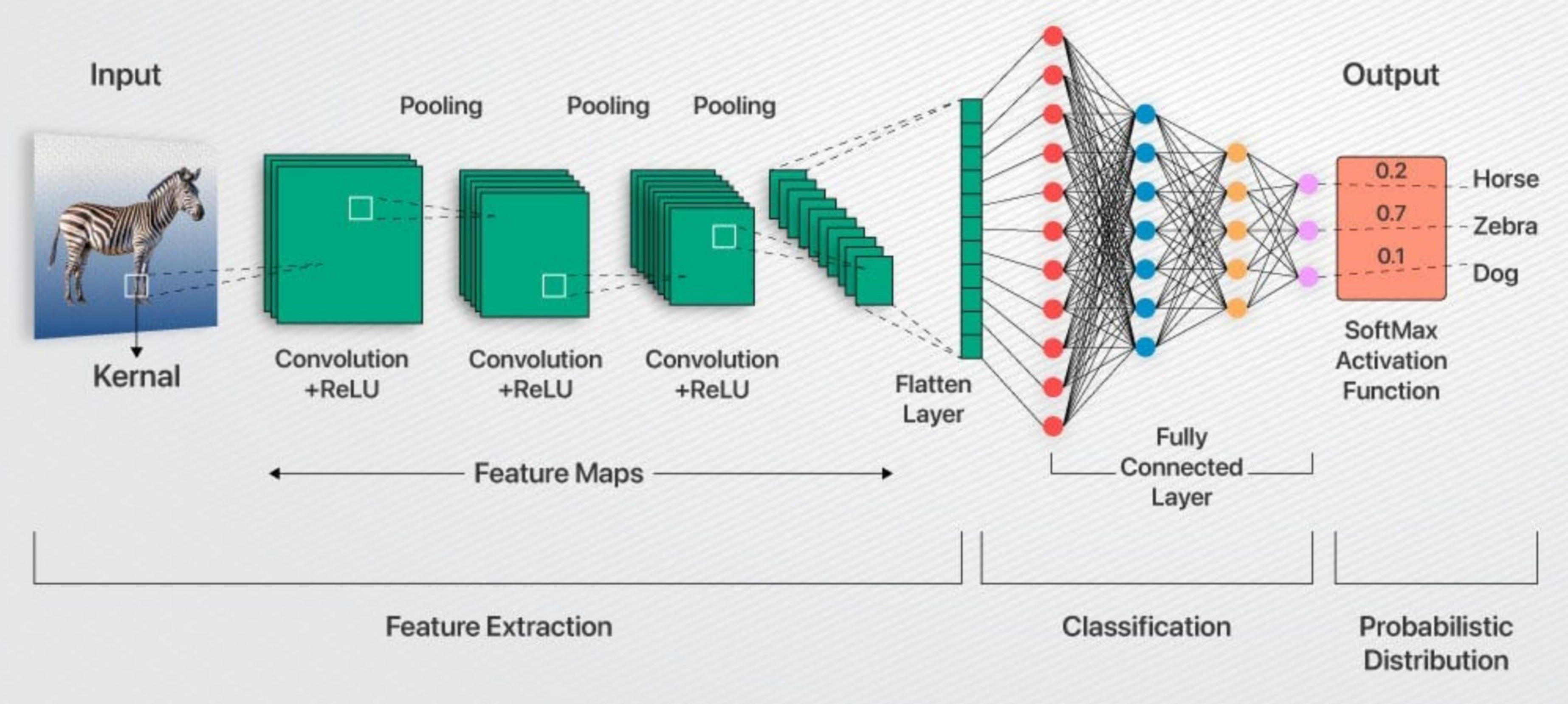
Learning Based Methods



Learning Based Methods

Convolutional Neural Network

Typically includes a series of convolutional blocks followed by a number of fully connected layers.

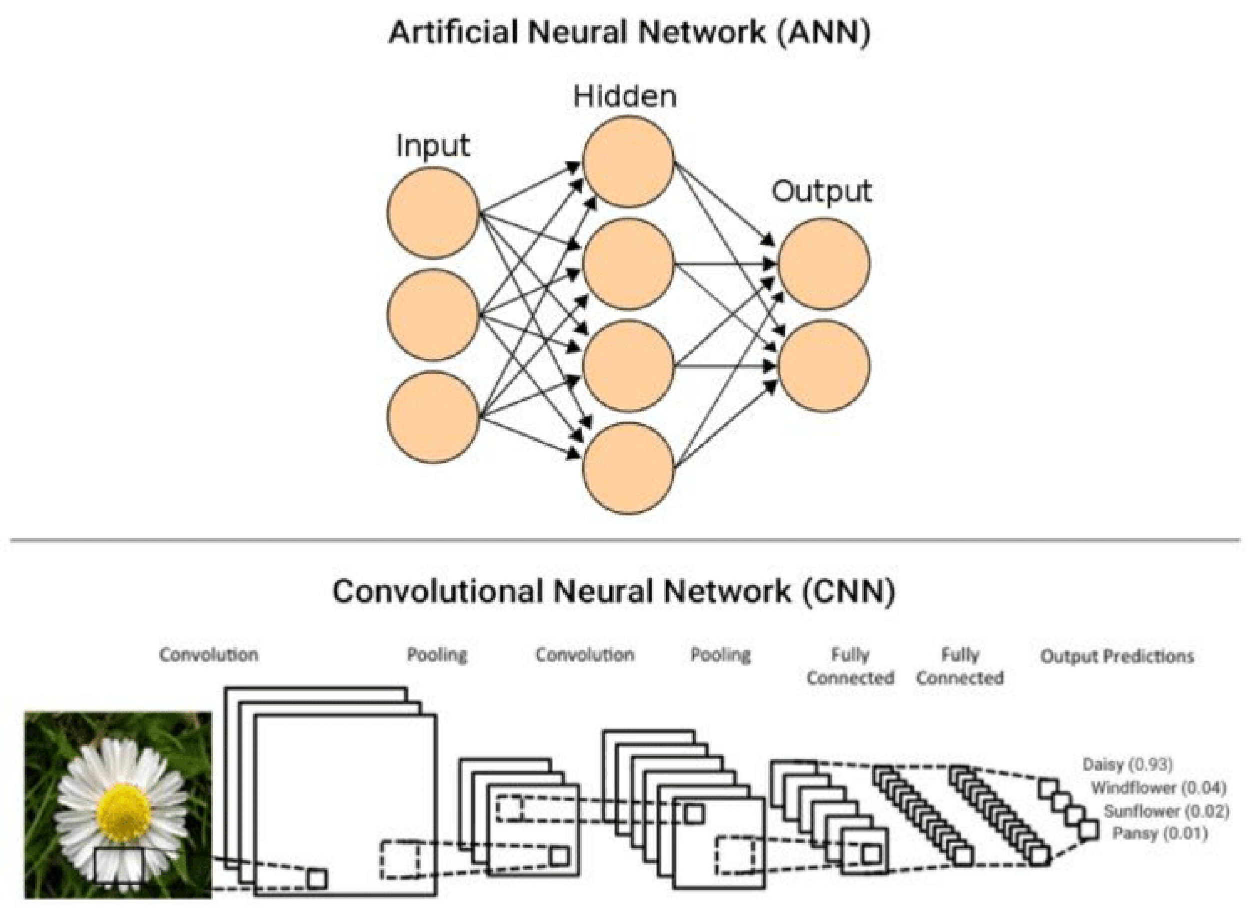


Learning Based Methods

Convolutional Neural Network

- One problem with using a fully connected MLP network for processing images is that image data is generally quite large, which leads to a substantial increase in the number of trainable parameters.

- This can make it difficult to train such networks for a number of reasons.

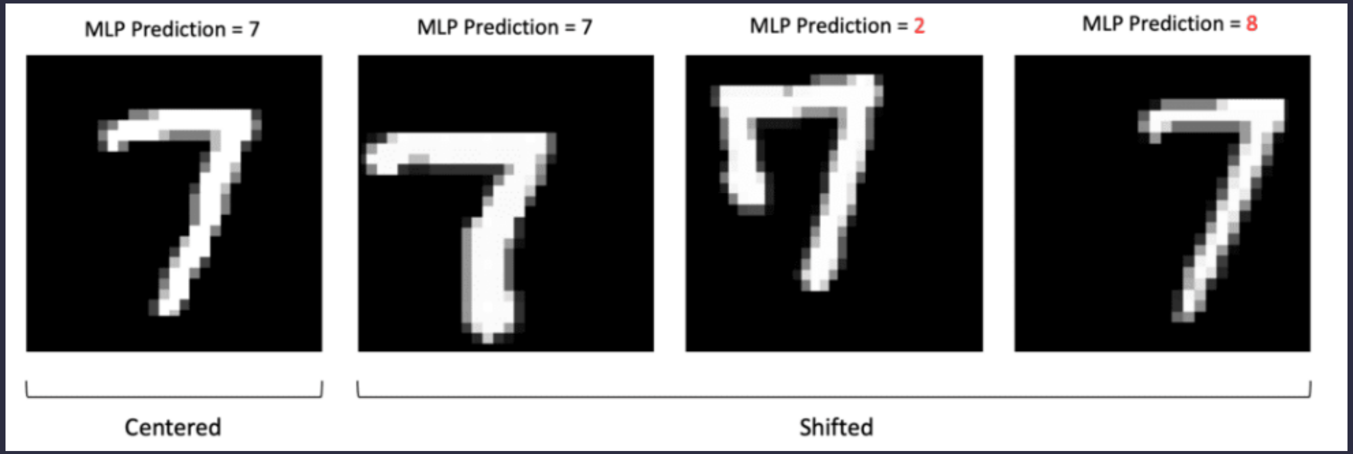


Learning Based Methods

Convolutional Neural Network

- Multilayer Perceptron (MLP) process image data is that they are not translation invariant

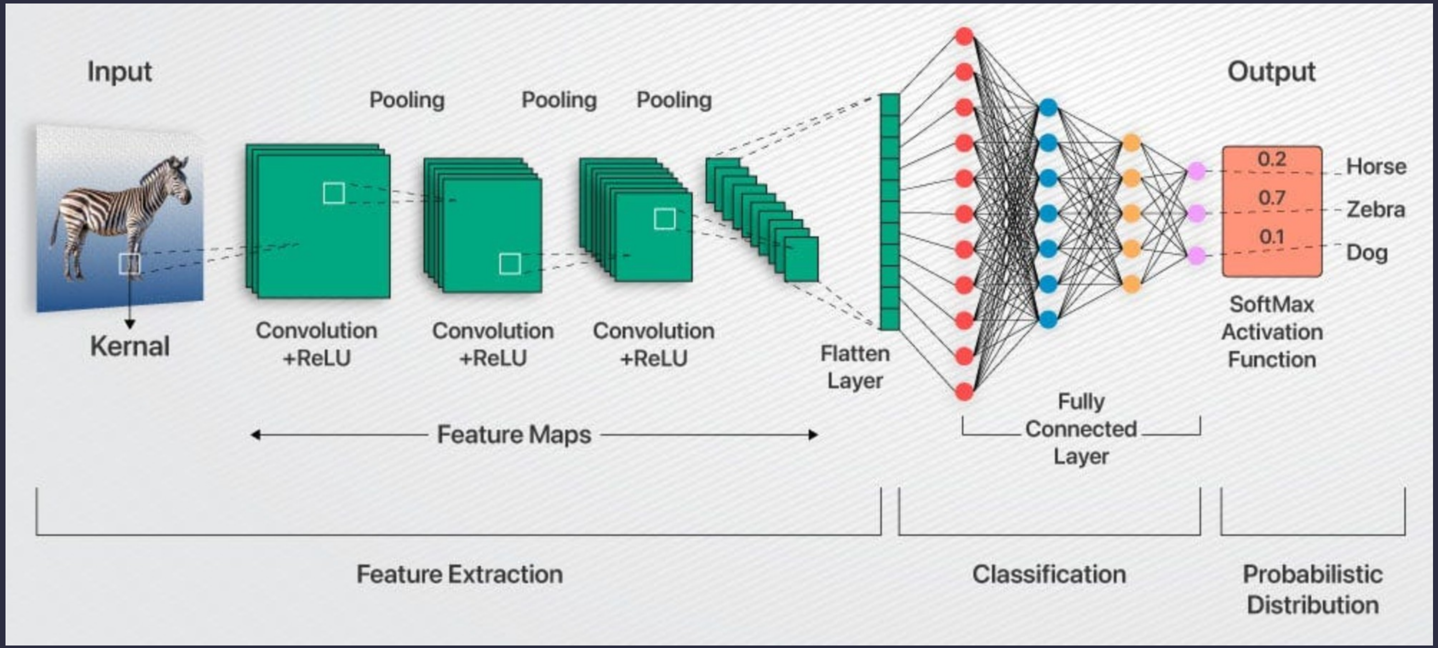
- Means that the network reacts differently if the main content of the image is shifted



Learning Based Methods

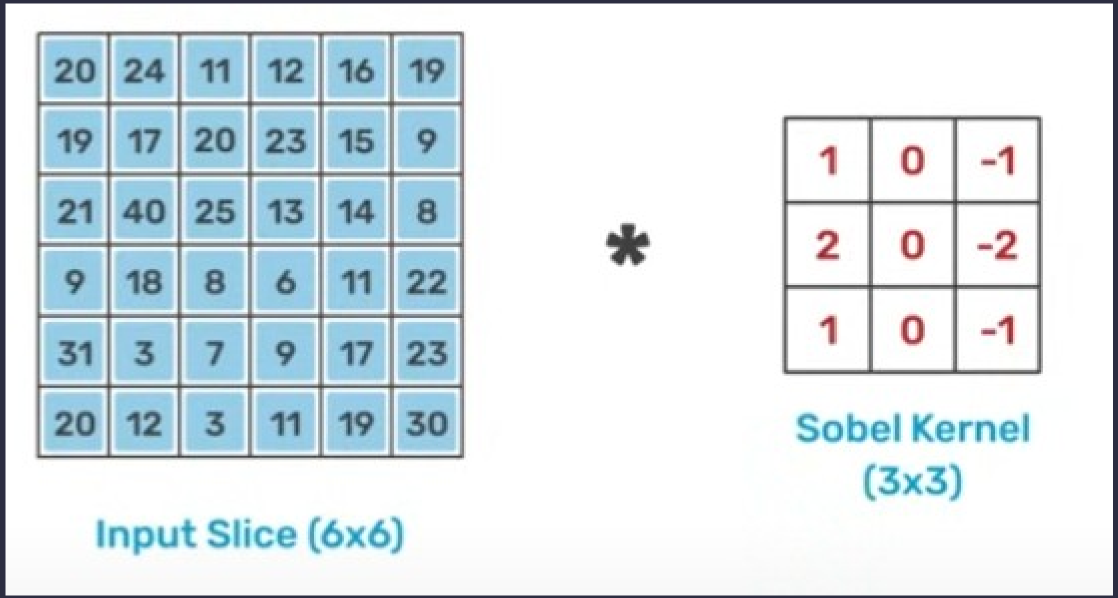
Convolutional Neural Network

- Effectively and efficiently process image data.
- Largely due to the use of **convolution** operations to extract features from images
- Key feature of convolutional layers, called parameter sharing
    - Same weights are used to process different parts of the input image
    - Detect feature patterns that are translation invariant as the kernel moves across the image



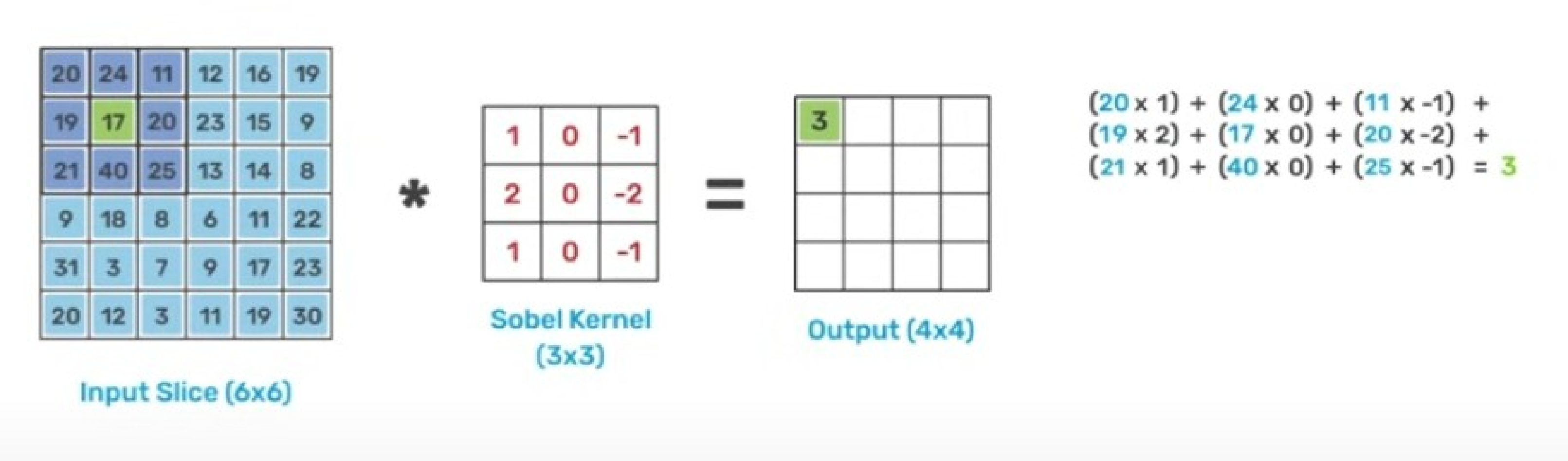
Learning Based Methods

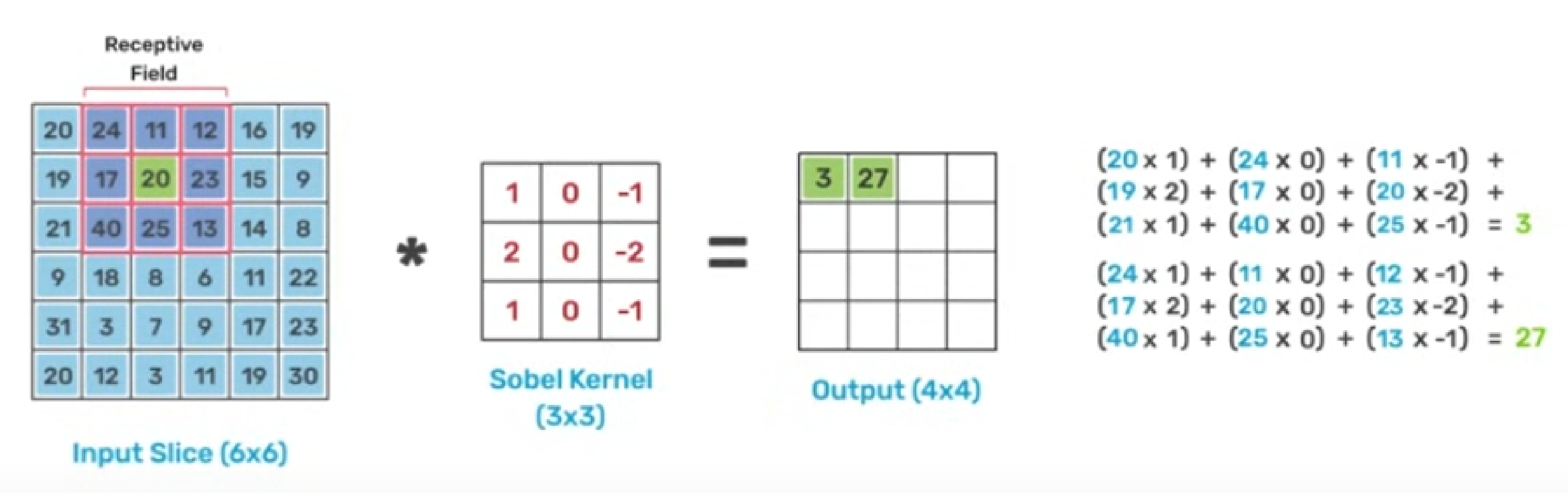
Convolutional Neural Network



Learning Based Methods

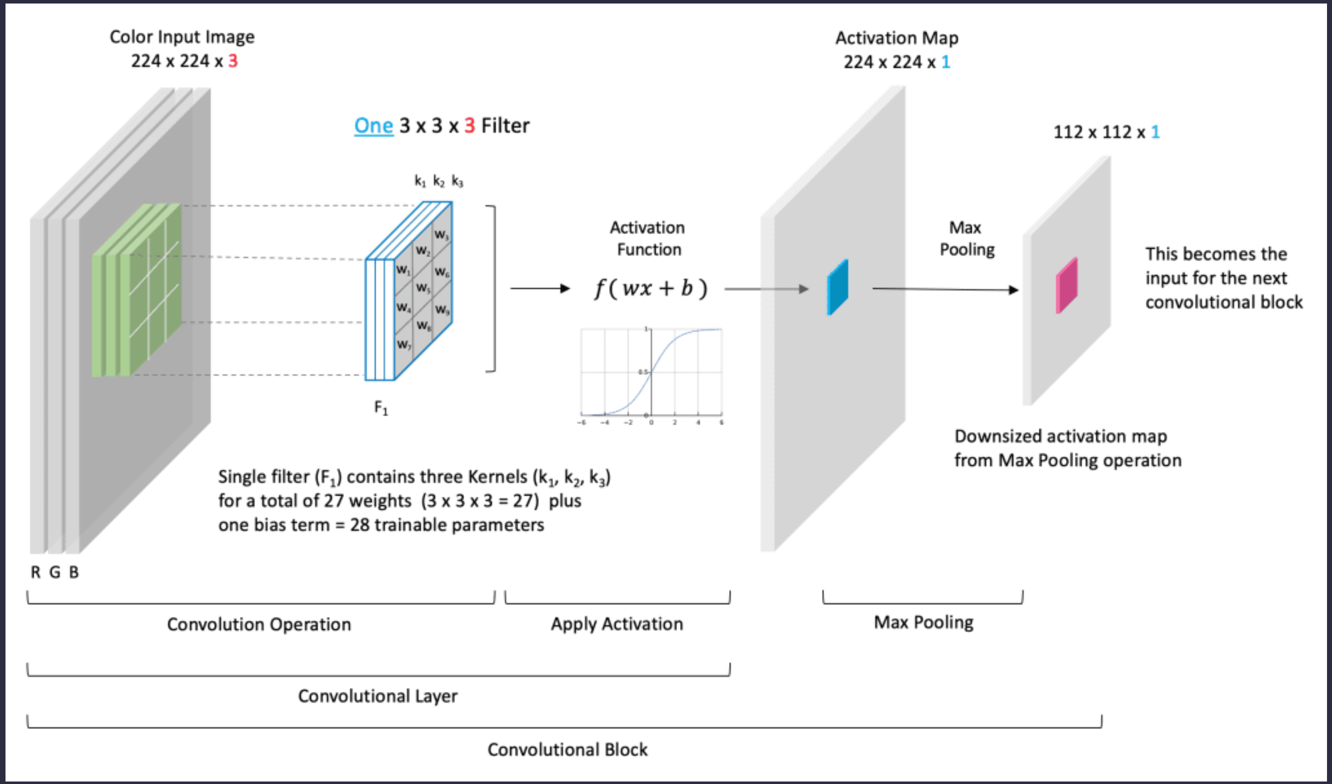
Convolutional Neural Network

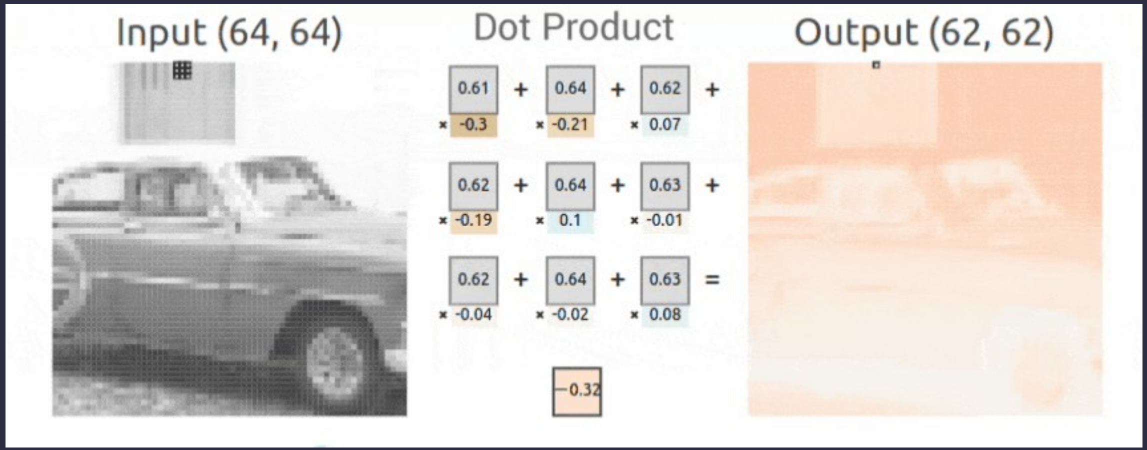




Learning Based Methods

Convolutional Neural Network





Learning Based Methods

Convolutional Neural Network

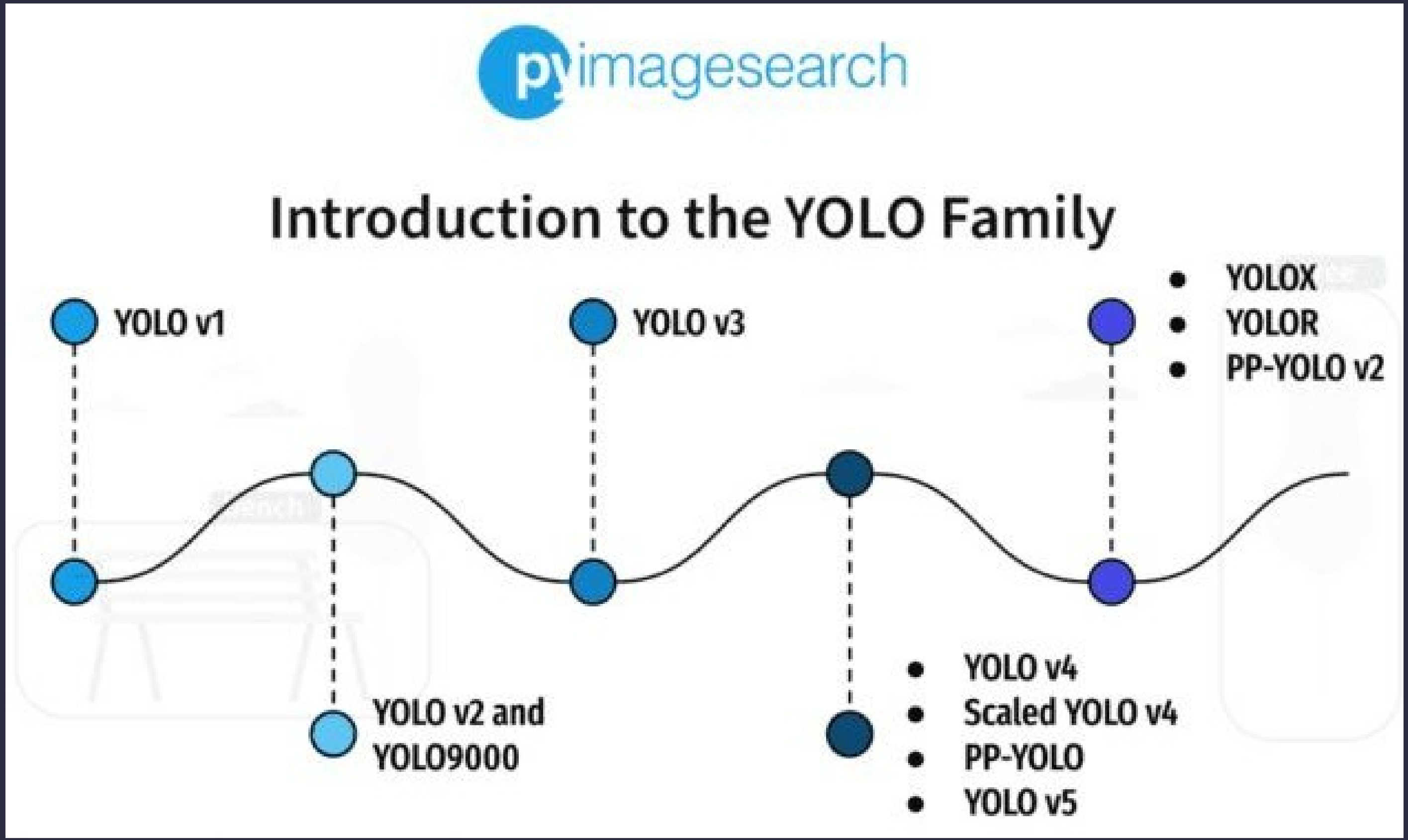
Example

https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/images/cnn.ipynb?hl=tr#scrollTo=0LvwaKhtUdOo

YOLO - You Only Look Once

- Belong to a new family of Object Detection Networks: Single Shot Detectors
    - Takes a single shot of the image to detect multiple objects,
    - R-CNN Series have a separate Region Proposal Network (RPN) and then a network for detecting objects from each proposal
    - Much faster than CNN models - Faster CNN (73.2% mAP at 7 FPS) and YOLOv1 (63.4% mAP at 45 FPS)

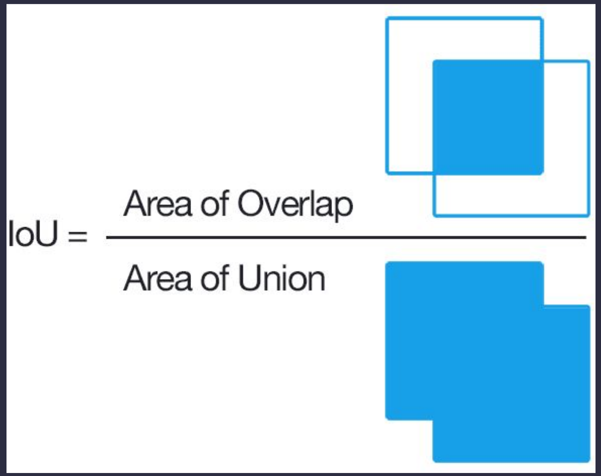
The YOLO Family



YOLOv1

Motives and Concepts:

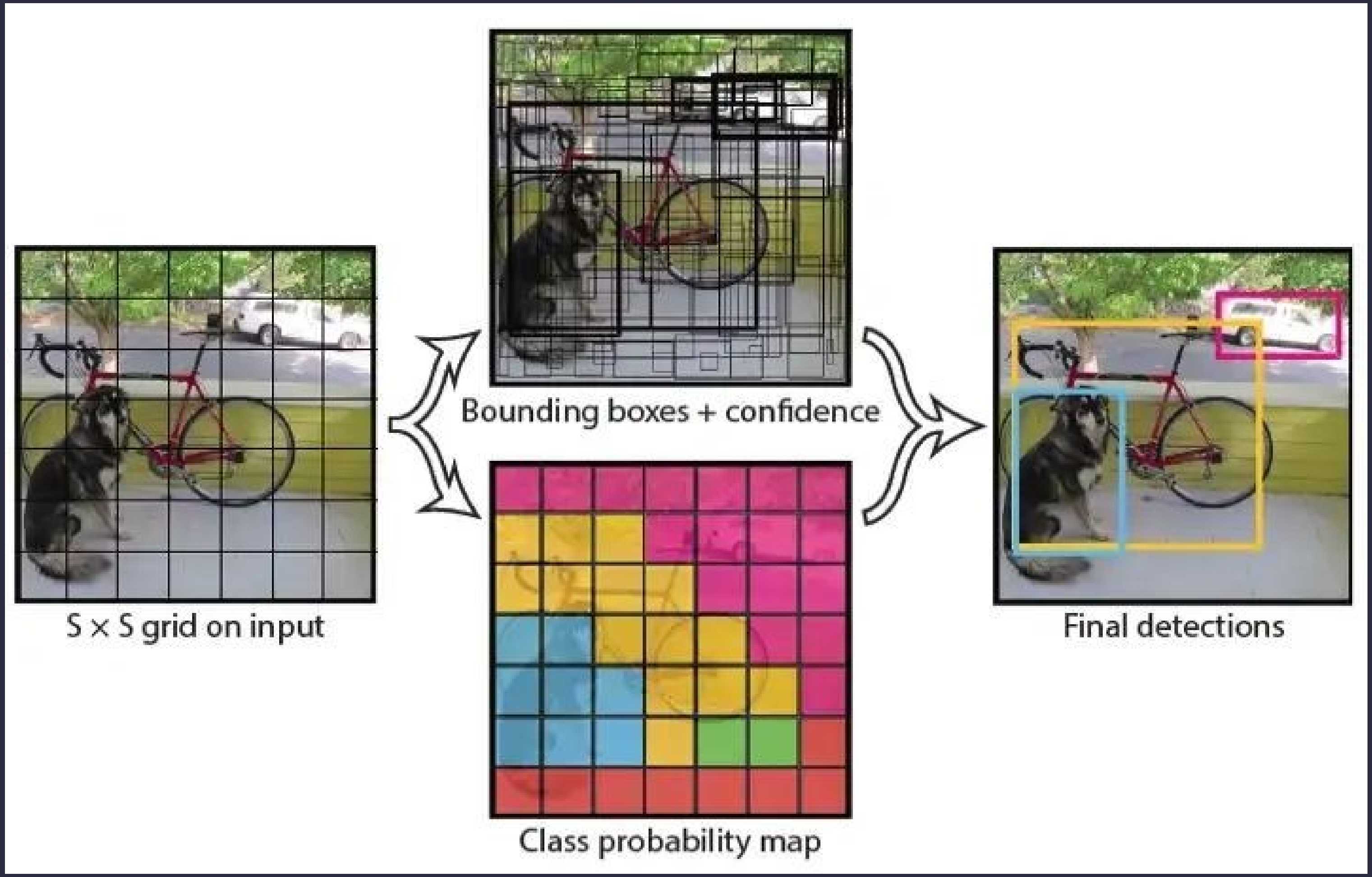
- Divides the input image into an S × S (7x7) grid.
- If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object.
- Each grid cell predicts B (2) bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model is that the box contains an object and also how accurate it thinks the box is that it predicts.
- Formally we define confidence as Pr(Object) ∗ IOU. If no object exists in that cell, the confidence score should be zero.
  Otherwise, we want the confidence score to equal the intersection over union (IOU) between the predicted box and the ground truth.



[Img ref: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/]

YOLOv1

Illustration



YOLOv1

Architecture:

- Inspired by GoogLeNet
- Alternating 1 × 1 convolutional layers reduce the features space from preceding layers
-

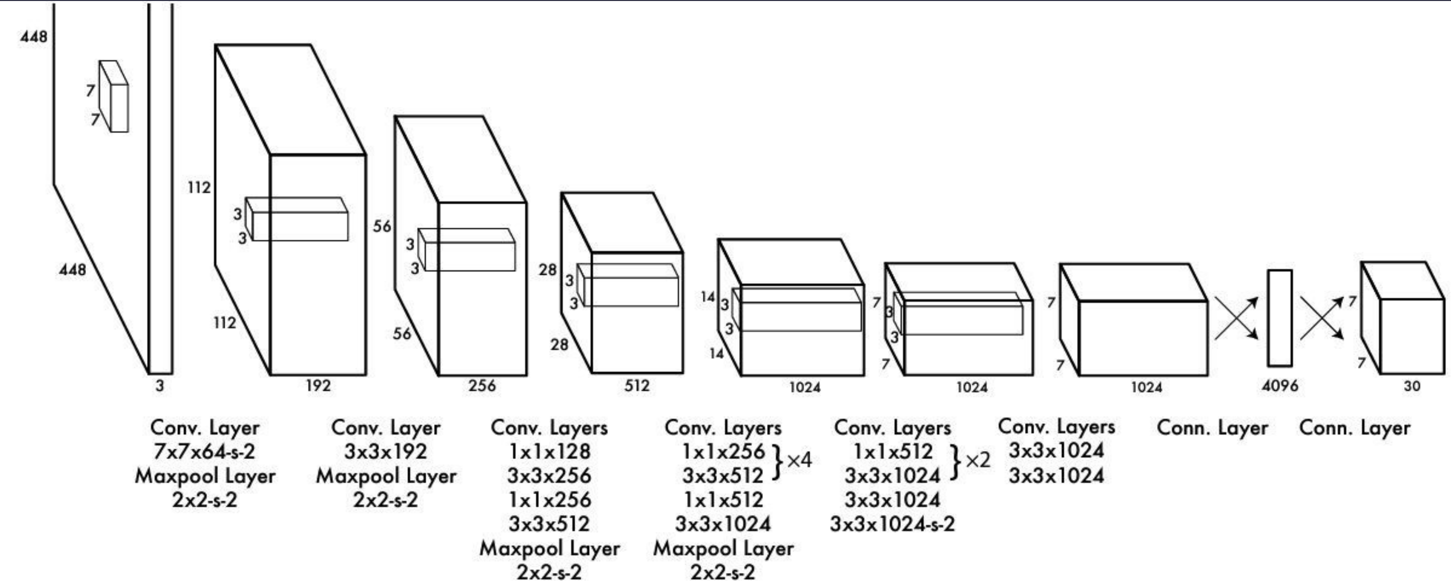
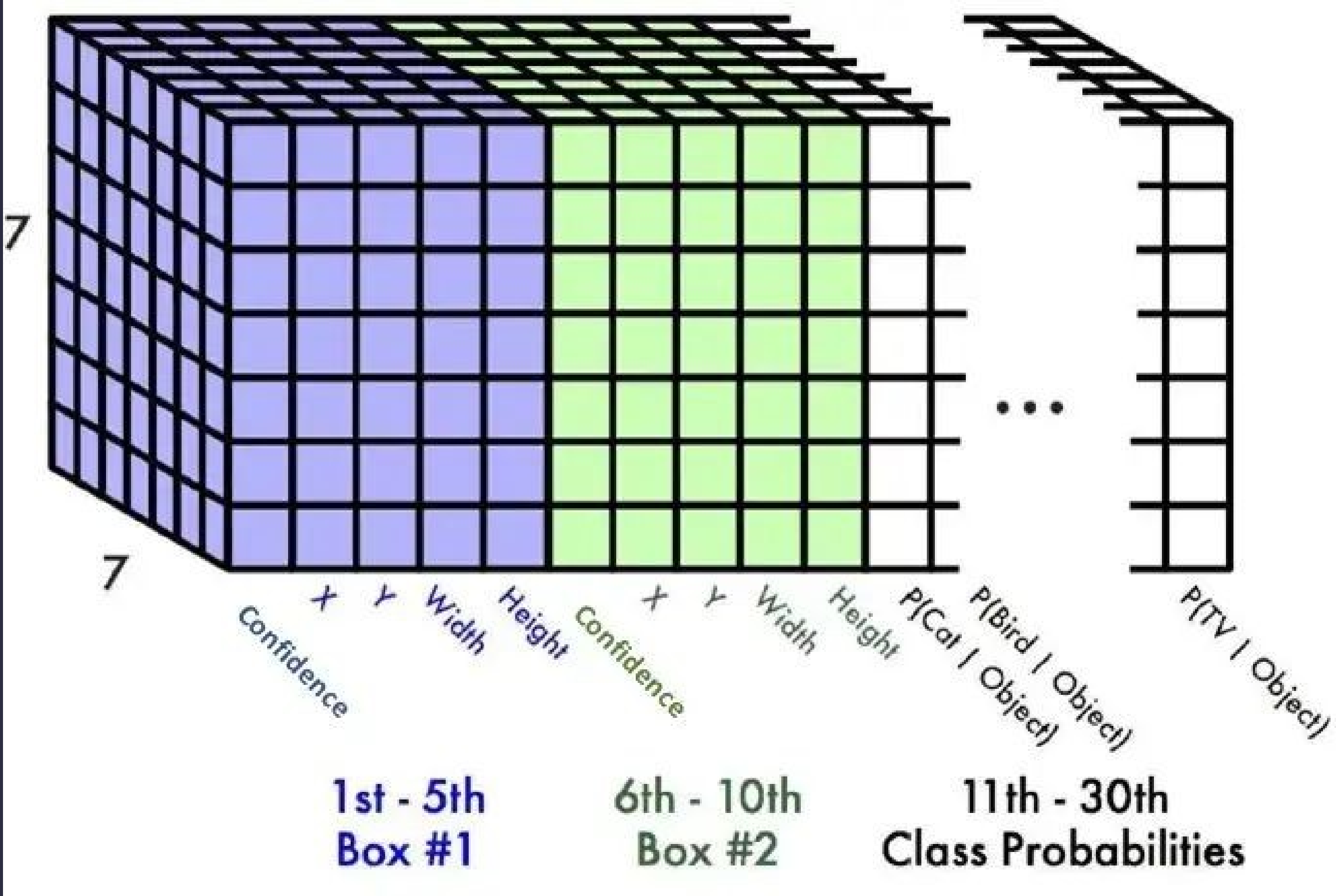


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

YOLOv1

Output and how it makes sense:



YOLOv2 or YOLO9000: Better, Faster, Stronger

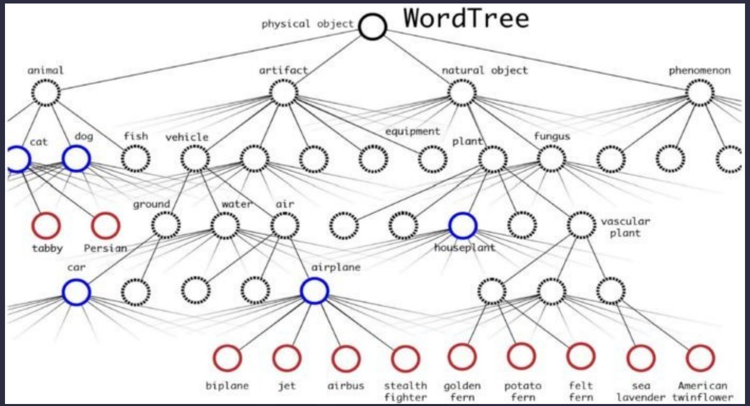
Performance:

- At 67 FPS, YOLOv2 gets 76.8% mAP on PASCAL VOC 2007.
- At 40 FPS, YOLOv2 gets 78.6% mAP which is better than Faster R-CNN using ResNet and SSD.

YOLOv2 or YOLO9000: Better, Faster, Stronger

Improvements on YOLOv1:

- Batch Normalization - 2% improvement
- Double input image resolution - 4% mAP improvement
- No Fully connected layers
    - Conv only architecture in which each Anchor Box predicts bounding boxes in its region
    - Class and objectness is calculated for each anchor box - +7% recall
    - Calculated the size of anchor boxes using K (5) means clustering
- Trained on variety of different input image dimensions (320*320 - 608*608)
- Trained for classification (MS COCO dataset) and object detection (stronger)
    - Combining classes from MS COCO and ImageNet by hierarchically clustering classes - finally 9418 classes - 19.7% mAP



YOLOv2 or YOLO9000: Better, Faster, Stronger

How to formulate Anchor Boxes: For every bounding box, we would predict the tx, ty, tw, th, and to which inform the following:

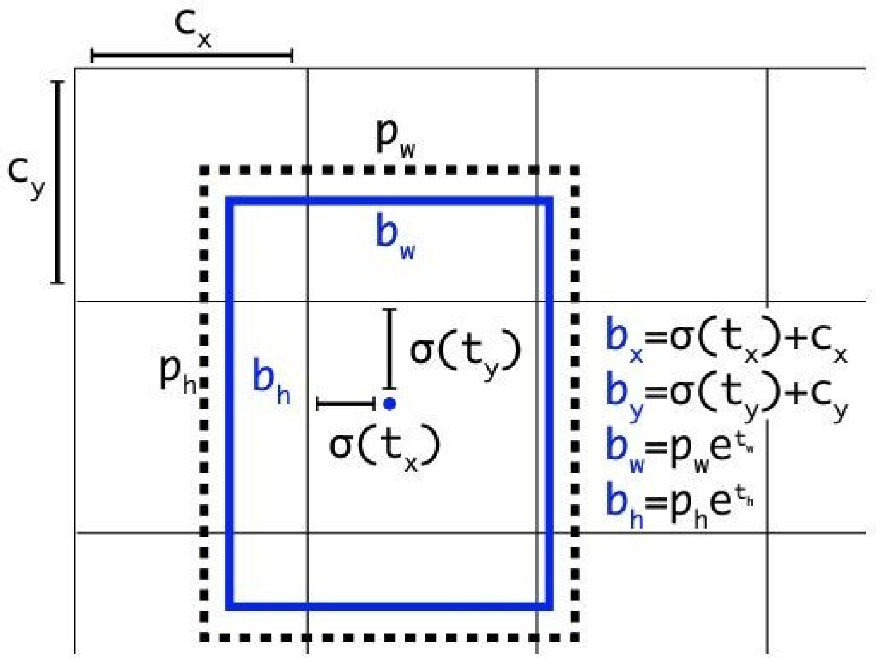


Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

YOLOv3: An Incremental Improvement

Some high level changes:

- Class predictions - Softmax is not used, independent logistic classifiers are used with binary cross entropy loss (classes are not mutually exclusive - eg. different datasets)
- 9 anchor boxes with 3 of each scale - you should calculate your own (K-Means)

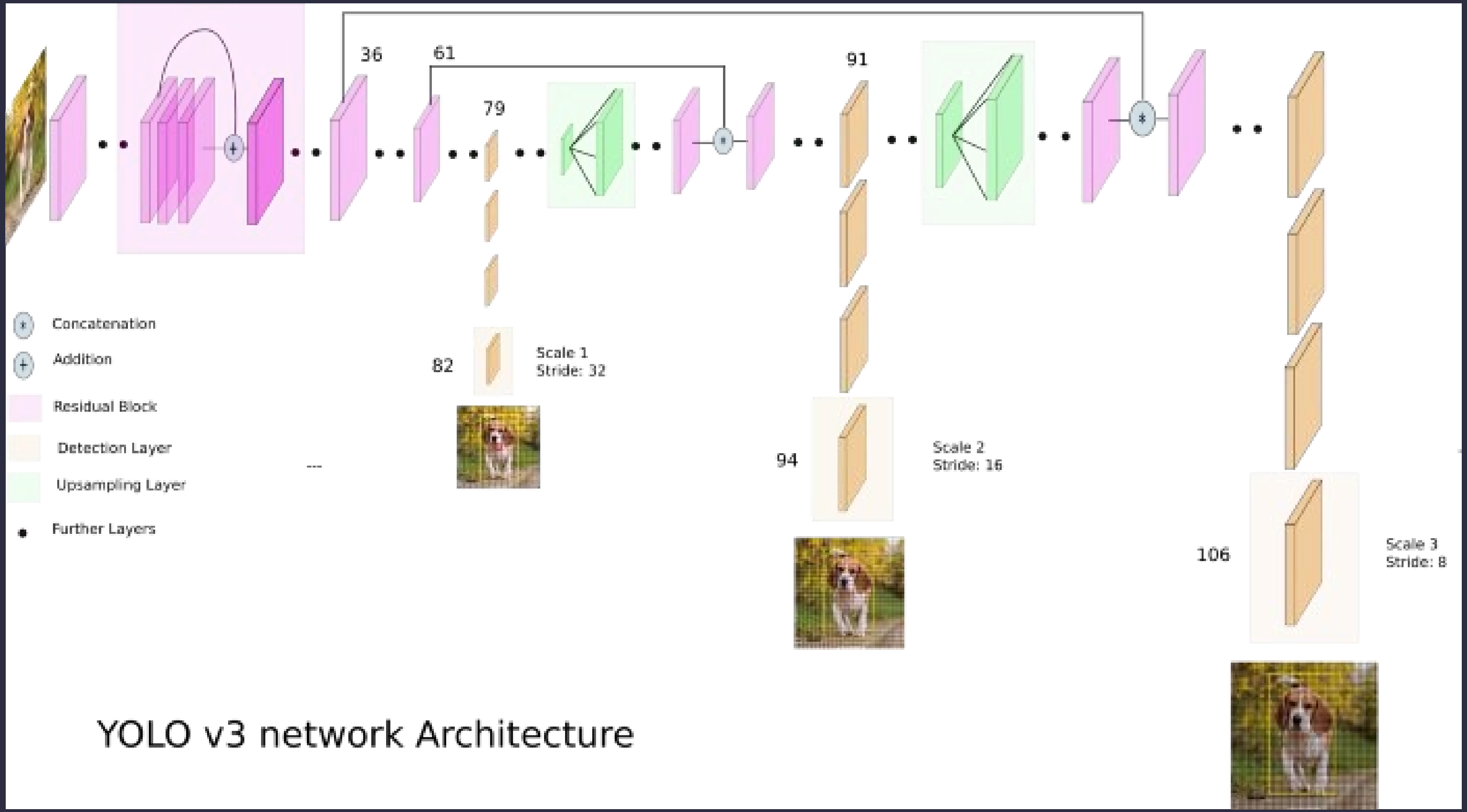
YOLOv3:An Incremental Improvement

Architecture:

- DarkNet 53 (from DN19) - better feature extractor with residual connections
- Feature map upsampling and residual connection - access to finer details at multiple level - Detection at 3 different levels (down-sampled by 32, 16, and 8)
    - YOLOv2 Struggled with small object detection - solved here

YOLOv3: An Incremental Improvement

- Architecture



YOLOv3:An Incremental Improvement

Results

- 3x faster but in terms of mAP, loses out to RetinaNet but very much comparable.
- PS: about Focal loss, They tried using focal loss, but It dropped their mAP about 2 points.

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++ [5]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [8]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [6]	Inception-ResNet-v2 [21]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [20]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2 [15]	DarkNet-19 [15]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [11, 3]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [3]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [9]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [9]	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

YOLOv4: Optimal Speed and Accuracy of Object Detection

Aims and results:

- More accurate but just as Fast.
- Improves YOLOv3's AP and FPS by 10% and 12%, respectively.
- extended as Scaled-YOLOv4

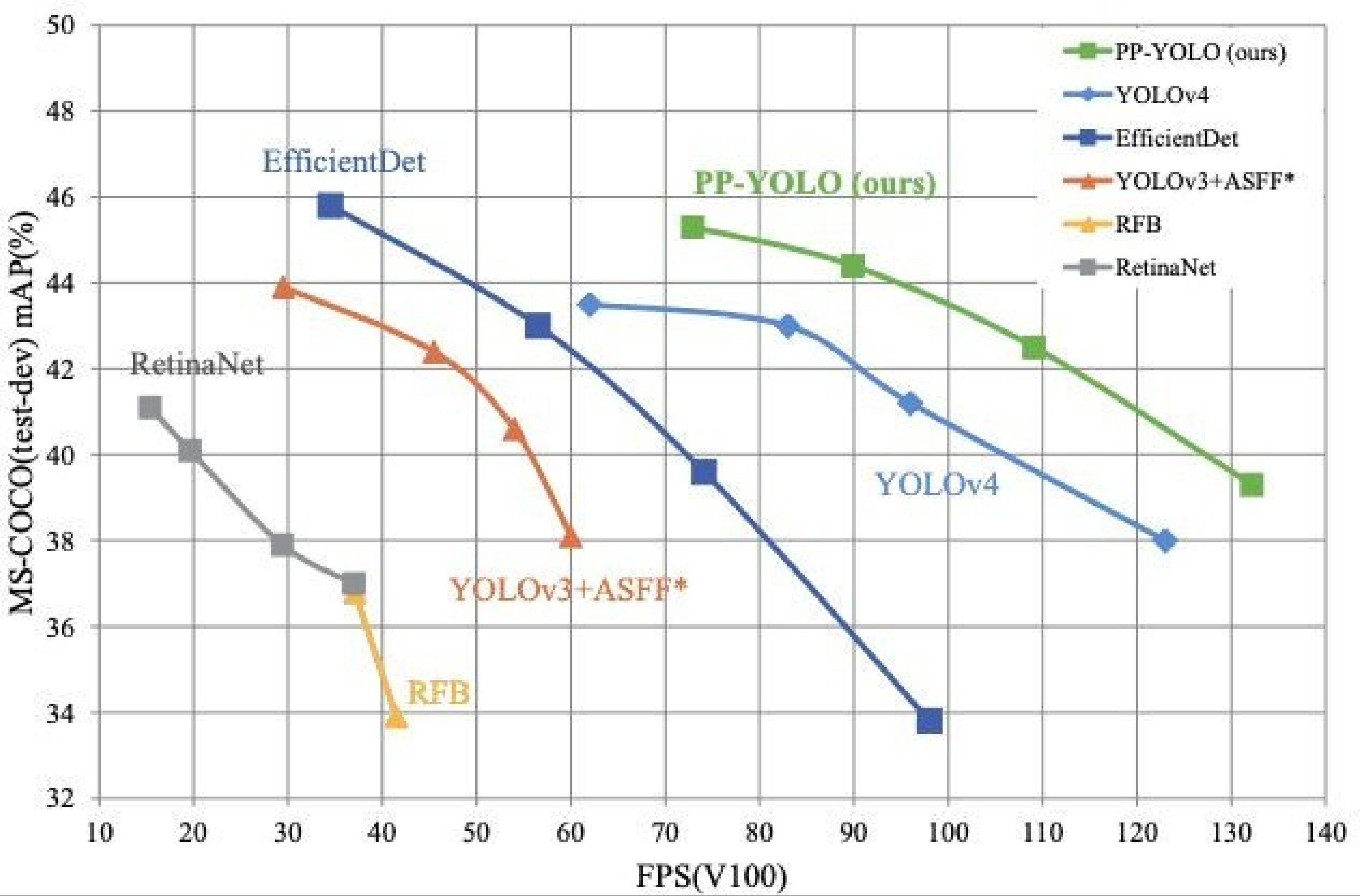
PP-YOLO

Overview:

- PaddlePaddle is a deep learning framework written by Baidu, which has a massive repository of Computer Vision and Natural Language Processing models)
- Their YOLOv3 implementations pushed further with larger batches, Exponential decay, DropBlock, SSP, CoordConv etc.

PP-YOLO

Results:



YOLOv5

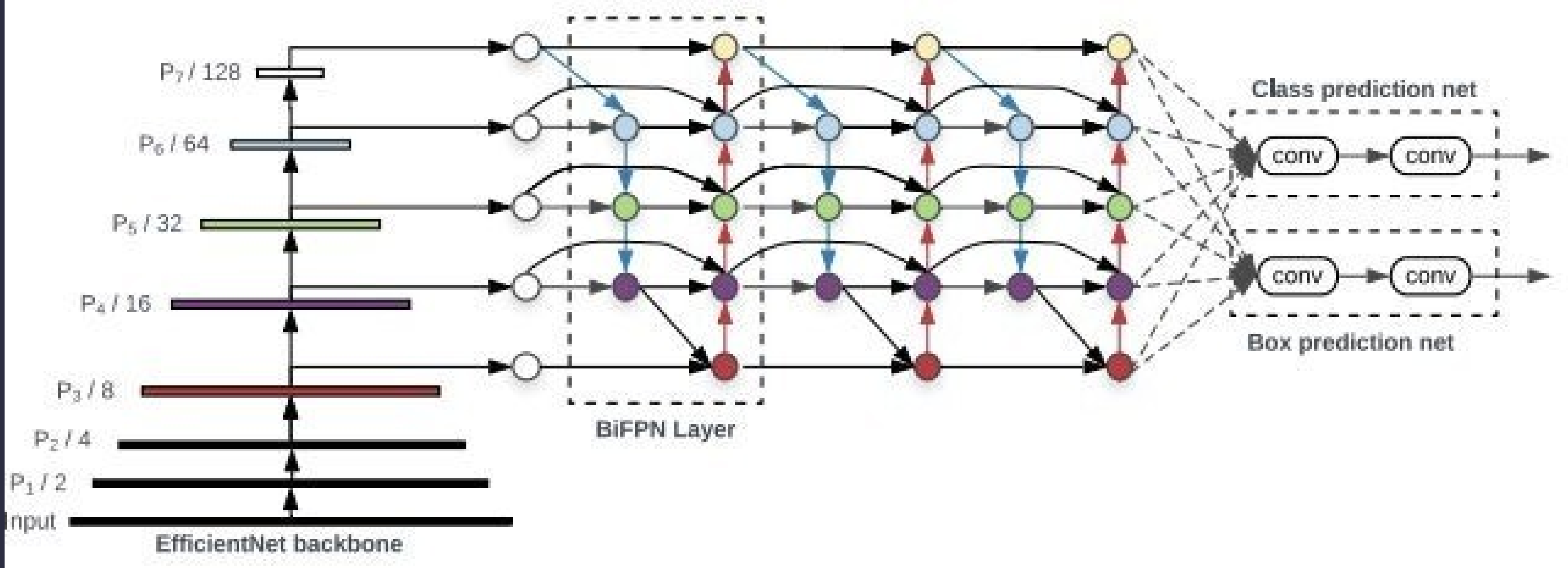
Overview:

- Native implementation in PyTorch, int16 training - helped.
- *Data Augmentation Techniques from YOLOv4*
- Architecture very close to YOLOv3 - and then improved on eventually
- Great documentation on Training custom dataset, and multi-GPU training - Test run on Google Colab Pro
- Benchmarked to be faster than YOLOv4 for the same mAP scores
- Comes in nano, small, medium, large and extra large sizes

YOLOv5

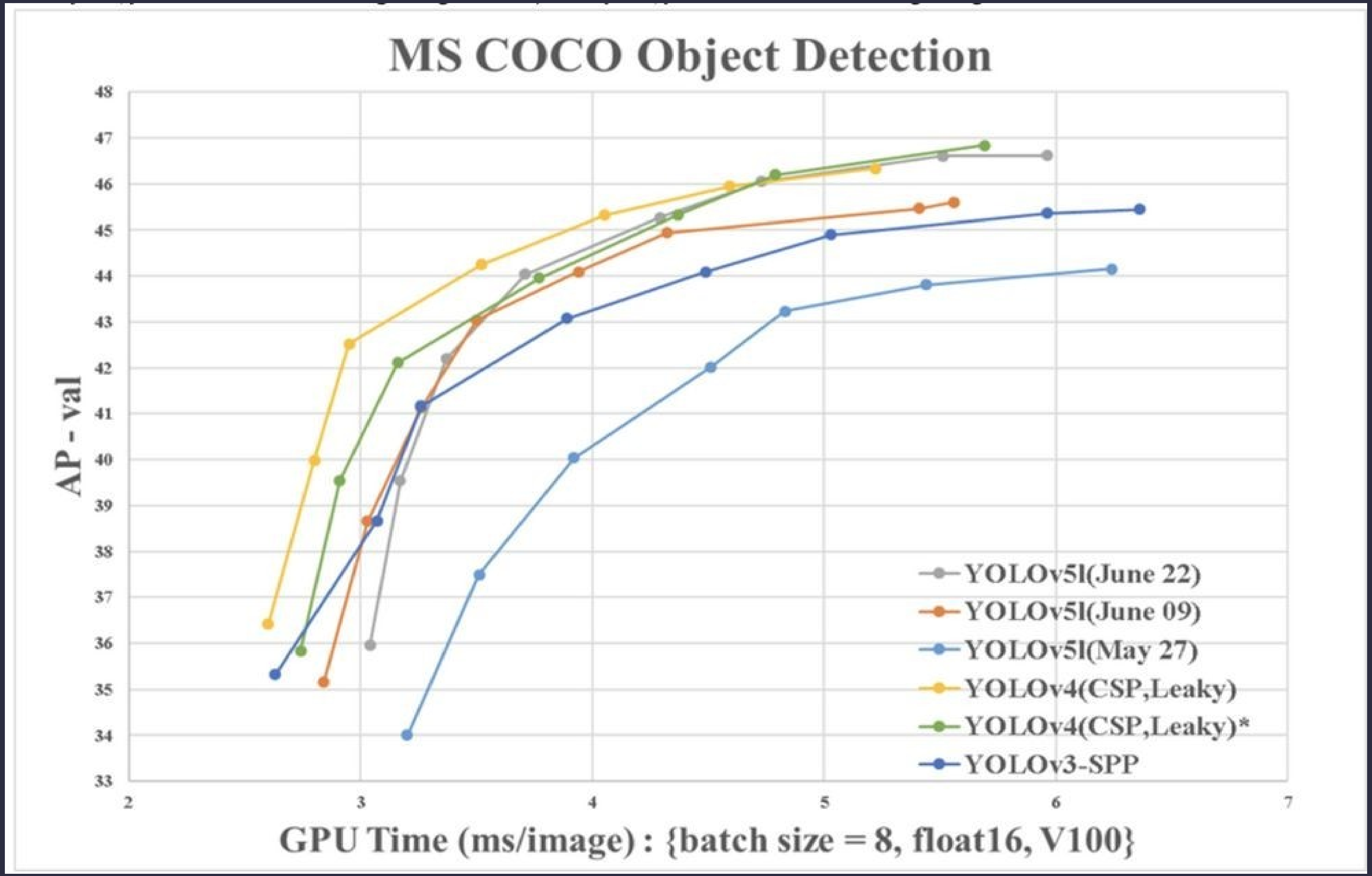
Architecture:

- Backbone: A convolutional neural network that aggregates and forms image features at different granularities. (CSP)
- Neck: A series of layers to mix and combine image features to pass them forward to prediction. (PANet)
- Head: Consumes features from the neck and takes box and class prediction steps.



YOLOv5

Performance:



YOLOX

Overview:

- Switch back to detector to an anchor-free manner.
  - Anchors were domain specific - less generalized
  - Detection heads were more complicated
    - Now the predictions were directly distance from top and left, plus height and width of the detection. - faster and better performance.
    - Sample close locals as positives - Center Sampling in FCOS
    - SimOTA (based on Optimal Transport Assignment for Object Detection)
-

YOLOX

Overview (cont.)

- A decoupled head and the leading label assignment strategy SimOTA to achieve state-of-the-art results.
- experiments indicate that the coupled detection head may harm the performance.
  - Decoupling these heads improves speed of convergence, but decreases AP,
  - They use 'little decoupled head' - -1.1ms in speed
-

YOLOX

Architecture:

- Shows how the ends are split into different modules for classification, Regression and P(obj)

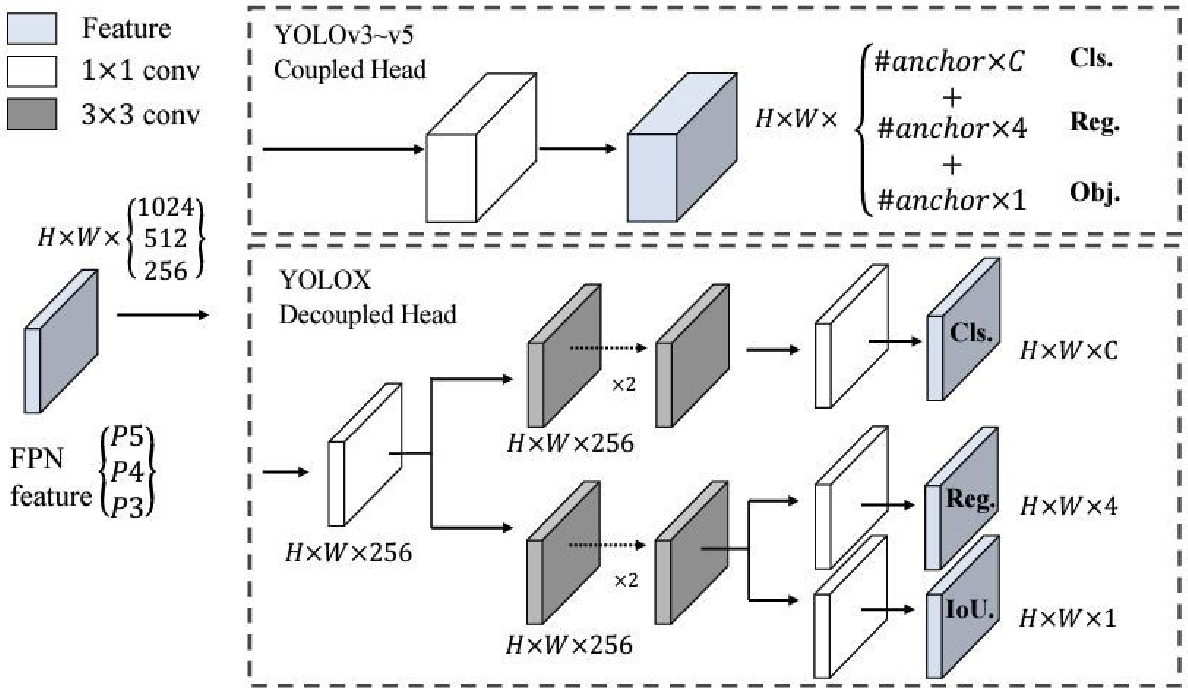
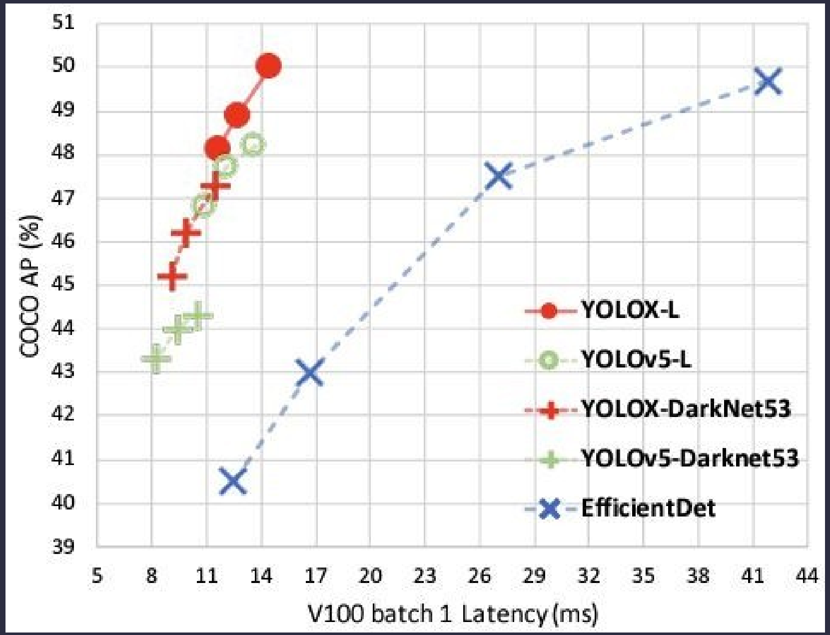


Figure 2: Illustration of the difference between YOLOv3 head and the proposed decoupled head. For each level of FPN feature, we first adopt a 1×1 conv layer to reduce the feature channel to 256 and then add two parallel branches with two 3×3 conv layers each for classification and regression tasks respectively. IoU branch is added on the regression branch.

YOLOX

Performance:



Methods	AP (%)	Parameters	GFLOPs	Latency	FPS
YOLOv3-ultralytics[2]	44.3	63.00 M	157.3	10.5 ms	95.2
YOLOv3 baseline	38.5	63.00 M	157.3	10.5 ms	95.2
+decoupled head	39.6 (+1.1)	63.86 M	186.0	11.6 ms	86.2
+strong augmentation	42.0 (+2.4)	63.86 M	186.0	11.6 ms	86.2
+anchor-free	42.9 (+0.9)	63.72 M	185.3	11.1 ms	90.1
+multi positives	45.0 (+2.1)	63.72 M	185.3	11.1 ms	90.1
+SimOTA	**47.3** (+2.3)	63.72 M	185.3	11.1 ms	90.1
+NMS free (optional)	46.5 (-0.8)	67.27 M	205.1	13.5 ms	74.1

YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors

Architecture:

- By controlling the shortest longest gradient path, a deeper network can learn and converge effectively (‘Designing network design strategies’ paper). In this paper, they propose Extended-ELAN (E-ELAN) based on ELAN.

YOLO

YOLO Example:

https://colab.research.google.com/github/ultralytics/yolov5/blob/master/tutorial.ipynb?utm_source=chatgpt.com#scrollTo=7mGmQbAO5pQb

YOLO

YOLO Example:

Custom Dataset

https://colab.research.google.com/github/roboflow-ai/notebooks/blob/main/notebooks/train-yolov7-object-detection-on-custom-data.ipynb?utm_source=chatgpt.com#scrollTo=1iqOPKjr22mL

MobileNet

- Proposed by Andrew G. Howard in 2017 Uses depth-wise separable convolutions to build light weight deep neural networks.
- Introduces two simple global hyper-parameters that efficiently trade off between latency and accuracy, allowing the model builder to choose the right sized model for their application based on the constraints of the problem

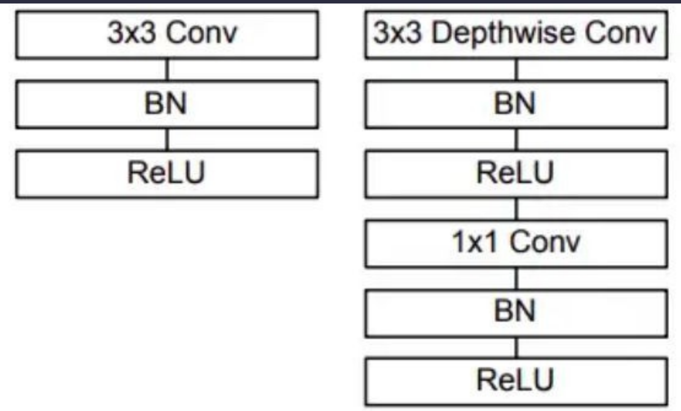
Figure 6. Example objection detection results using MobileNet SSD.

MobileNet

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

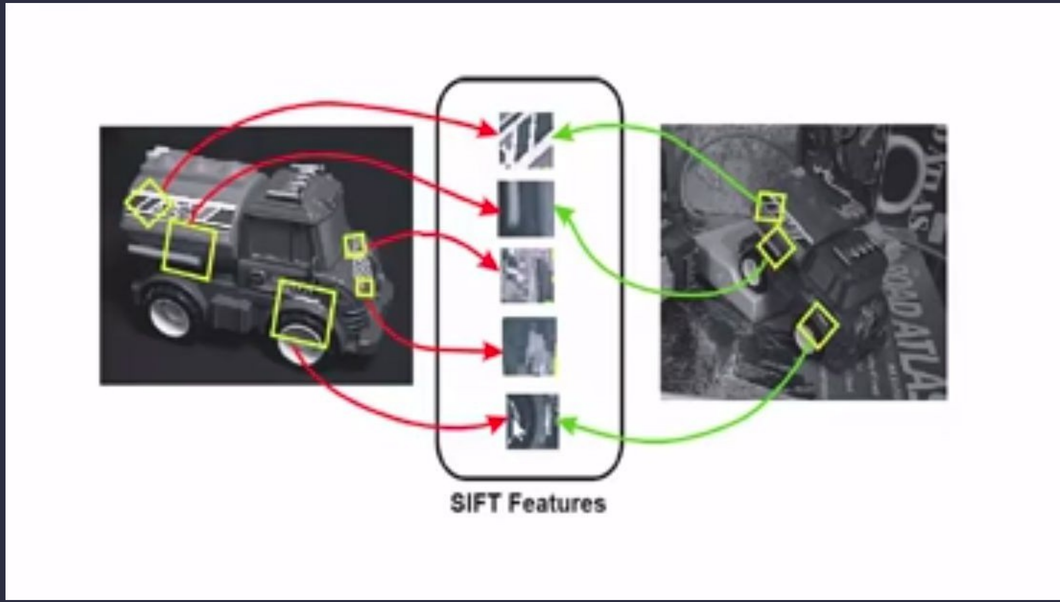
Network Architecture: MobileNet



Left: Standard Convolution followed by batch normalization and RELU. Right: Depthwise convolution layer and

SIFT

- Introduced by D. Lowe et.al. in 2004
- Non-learning based approach, Scale-Invariant Feature Transform has 4 main steps:
  - Scale-space peak selection: Potential location for finding features.
  - Keypoint Localization: Accurately locating the feature keypoints.
  - Orientation Assignment: Assigning orientation to keypoints.
  - Keypoint descriptor: Describing the keypoints as a high dimensional vector.
  - Keypoint Matching



Images in this slide from https://medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40