Recitation 13

Intro to Deep Learning, Fall 2022

Samruddhi Pai, Yashash Gaurav, Talha Faiz

Some of the content is inspired by CMU's 16-824: Visual Learning and Recognition course

Agenda

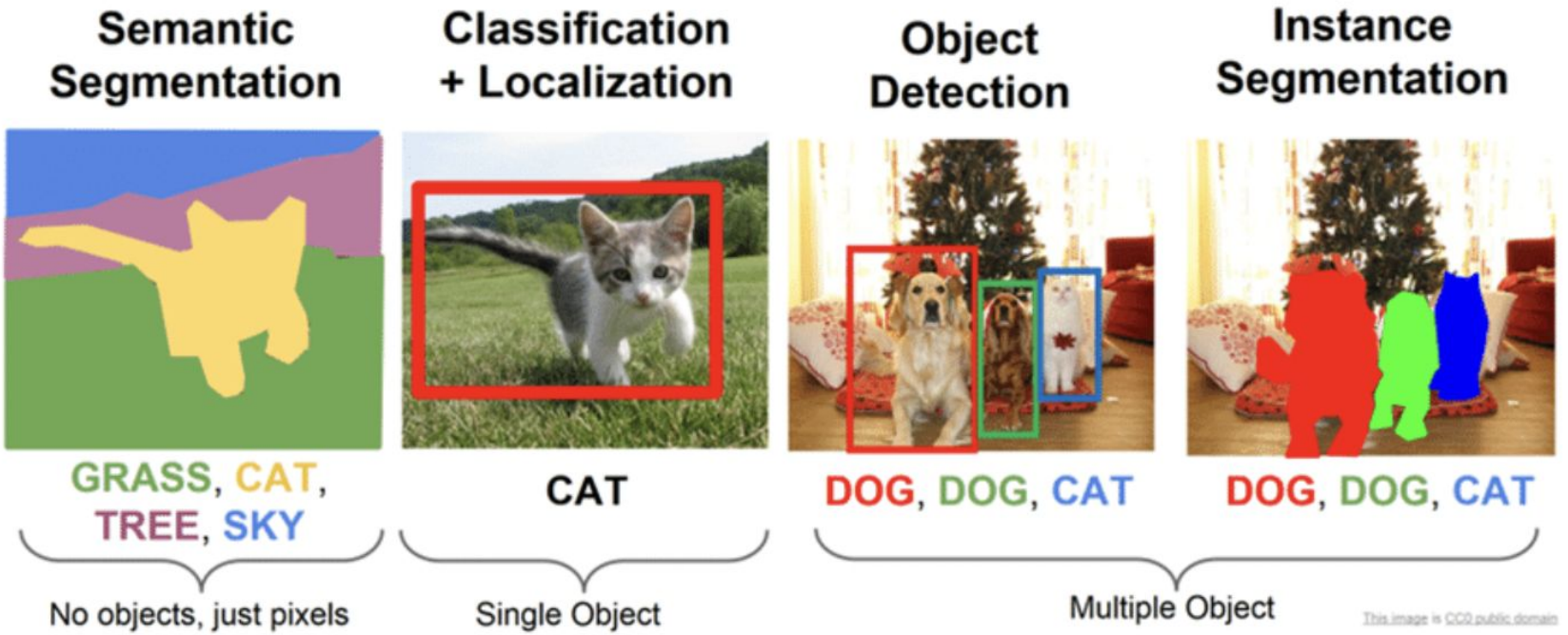
1. Introduction to Image Segmentation.
    a. Problem definition
    b. Standard Datasets
2. Why YOLO
3. Various versions of YOLO
4. Image segmentation - SOTA Architectures

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 - SOTA Architectures

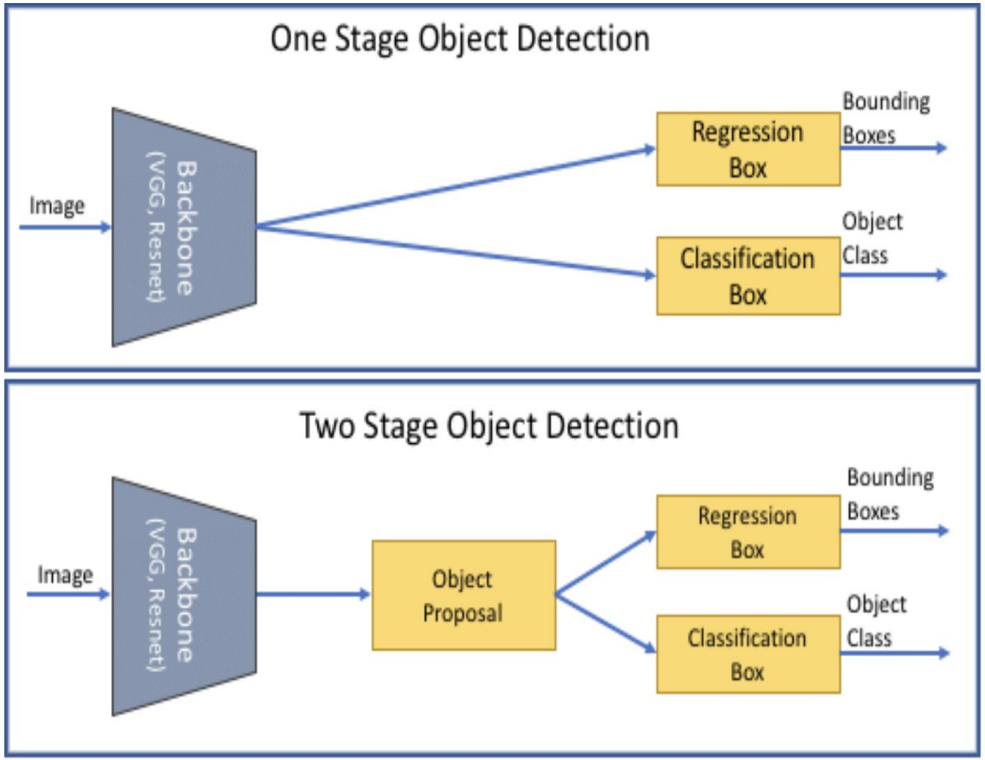
Key Idea - Understanding the journey behind refining models for better speed and/or accuracy for a particular task

Problem Definition



[Img Ref: https://www.researchgate.net/figure/Comparison-of-semantic-segmentation-classification-and-localization-object-detection_fig1_334363440]

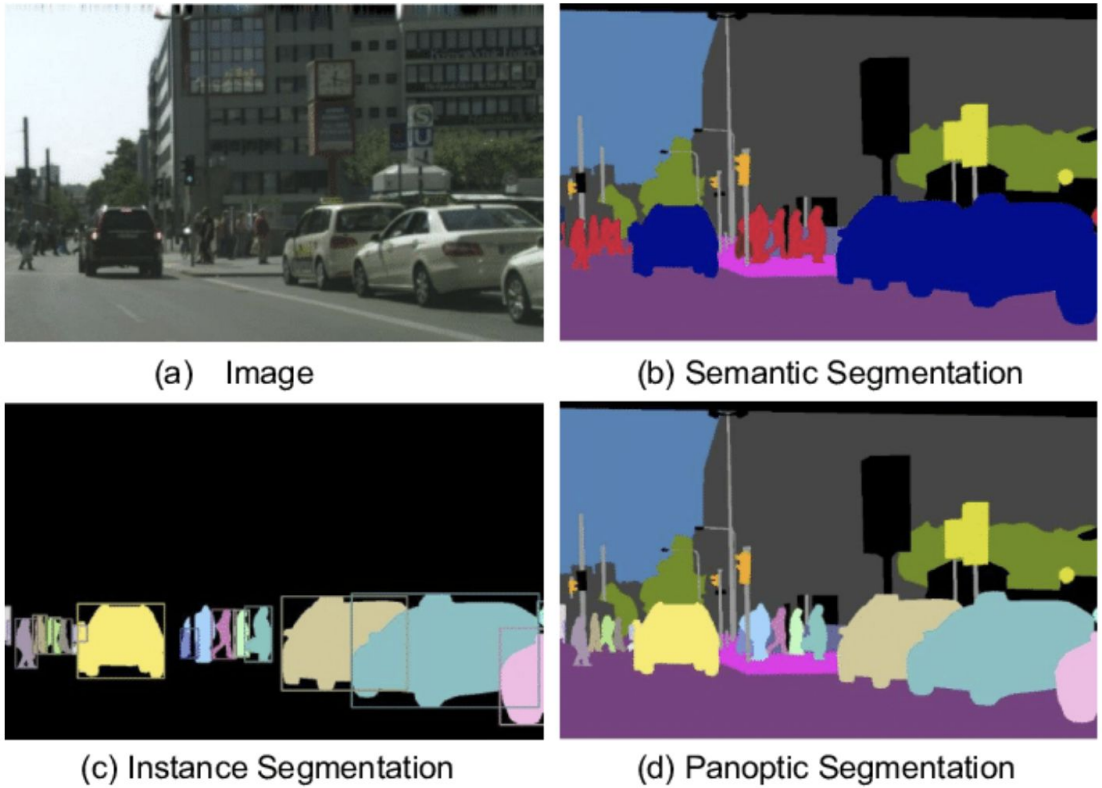
Object Detection



- 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

[Img Ref: https://www.researchgate.net/figure/Two-stage-vs-one-stage-object-detection-models_fig3_353284602]

Image Segmentation



[Img Ref: https://www.researchgate.net/figure/b-semantic-segmentation-c-instance-segmentation-and-d-panoptic-segmentation-for_fig5_342409316]

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

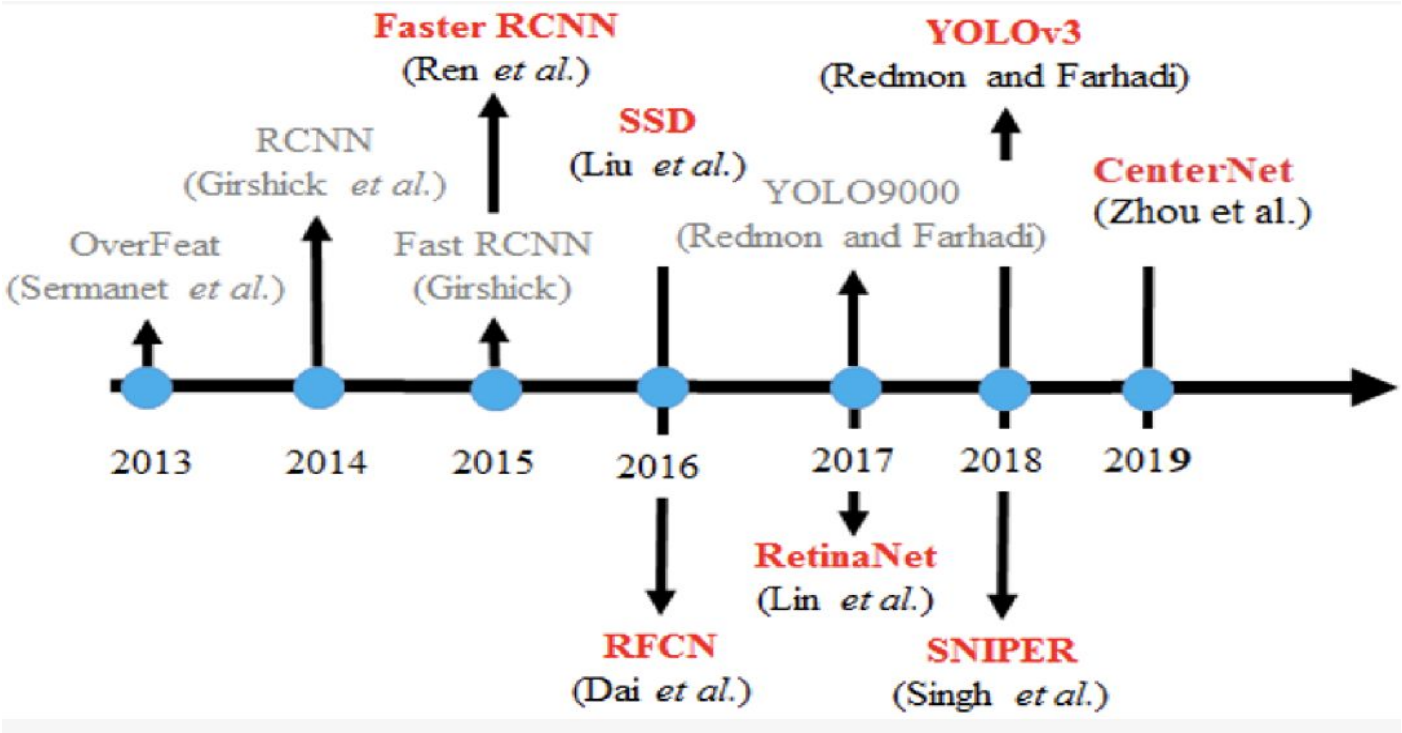
All of these can be used for object detection, segmentation, dense pose estimation, key point detection, etc.

Other task specific datasets - MNIST, KITTI, etc.

Different Methods for Solving these tasks:

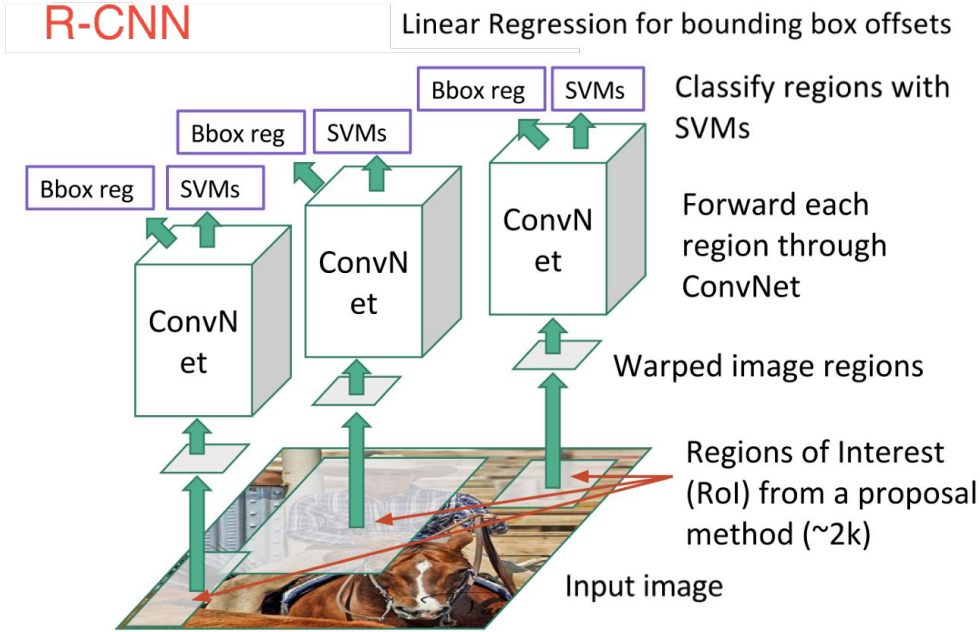
- Edge templates + nearest neighbor
- Haar Wavelets + SVM
- Other wavelet + adaBoost etc.
- Rectangular differential features + adaBoost
- Parts based binary orientation position histograms + adaBoost
- Dynamic programming
- Learning based methods

Learning Based Methods



[Ref: https://www.mdpi.com/2079-9292/9/4/583]

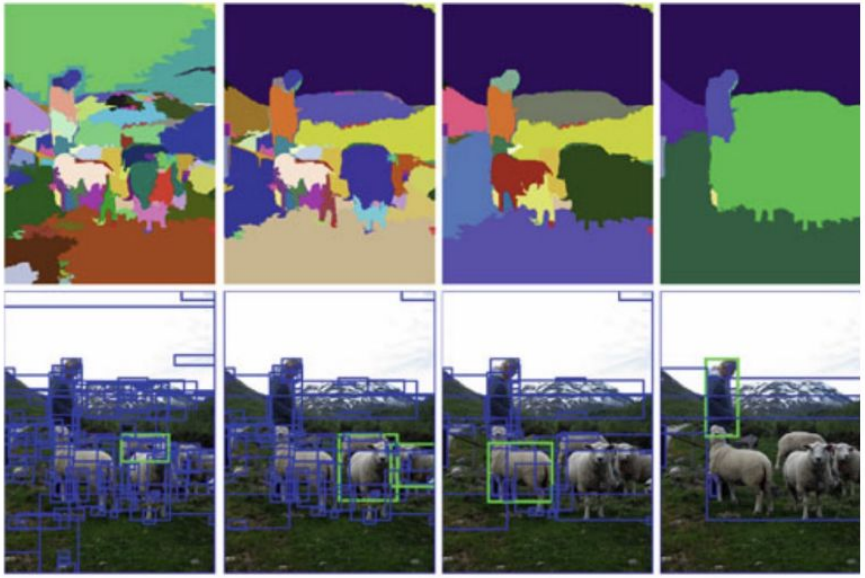
RCNN - Region-based Convolutional Neural Network [2]



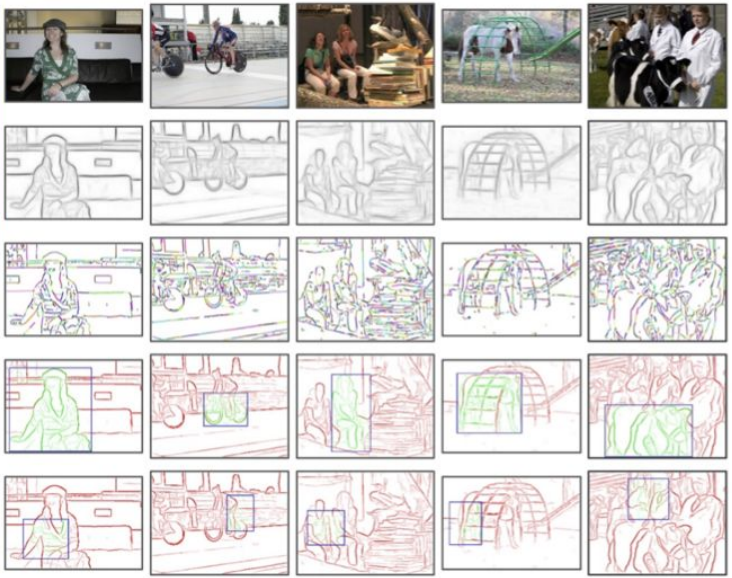
- 2K regions are proposed and reshapes
- These are passed through CNN architecture for extracting features (AlexNet, VGG 16, ResNet 50)
- Classification and localization is performed by SVMs and fully connected layers respectively

[Ref: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e]

Proposing the Regions



Selective Search [Uijlings et al.,]



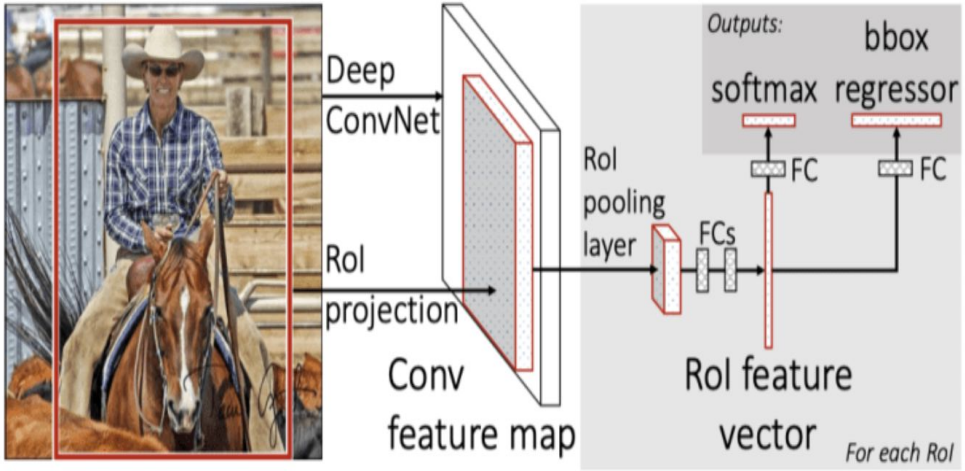
Edge boxes [Zitnick and Dollar]

Problems with RCNN

- Multistage training is expensive and time consuming - Pretrained CNN network is fine-tune on proposed regions and then final layer is replaced by SVM classifier

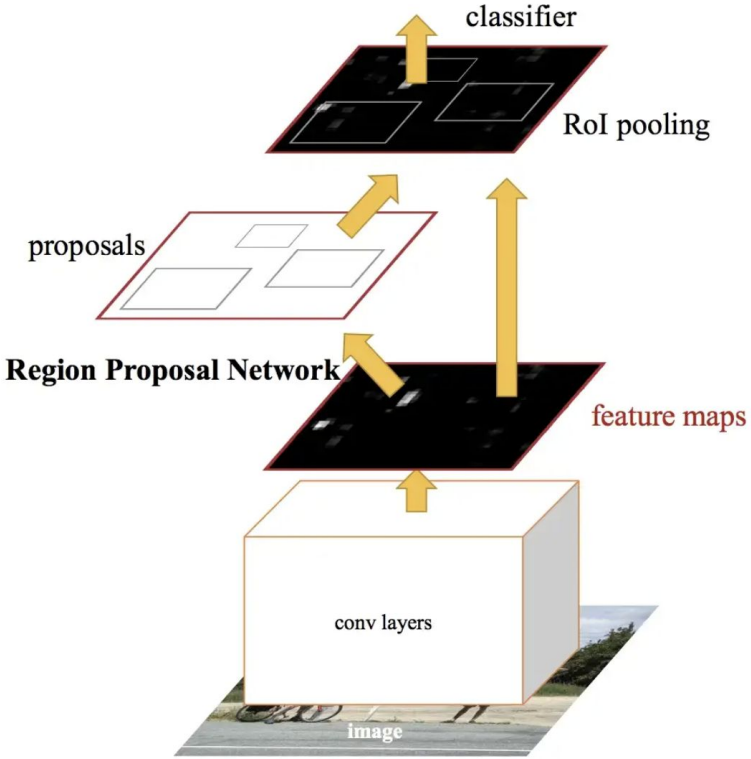
- Objects are detected over each of the regions which is time consuming. Positive and negative samples based on IOU are generated after this stage

Fast RCNN [3]



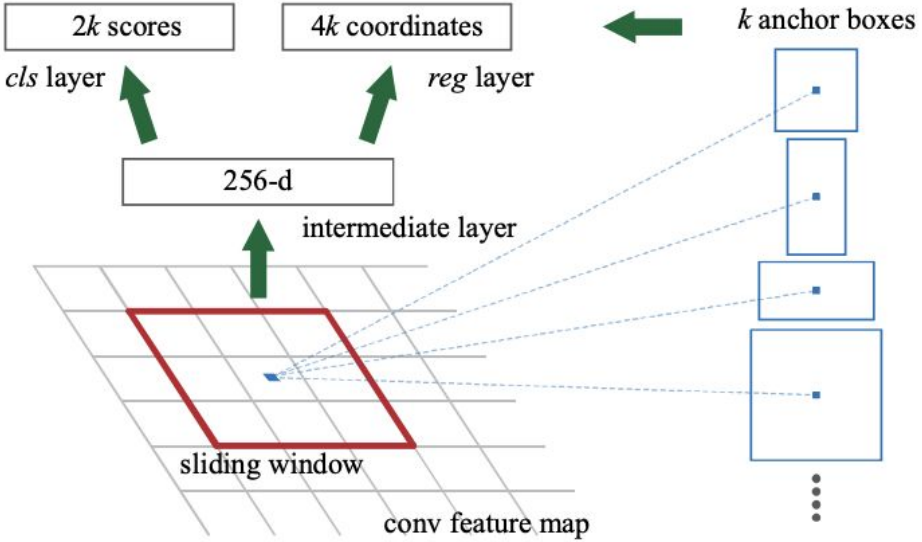
- Inspired by SPPNet, Fast RCNN uses ROI pooling where CNN is applied only once over the resized image and features corresponding to the proposed regions are selected
- Reduced processing time as features are not extracted for each selected region
- Since the image is resized, a fixed size vector is obtained at the output which is passed to fully connected layers for predicting classes and bounding box parameters

Faster RCNN [4]



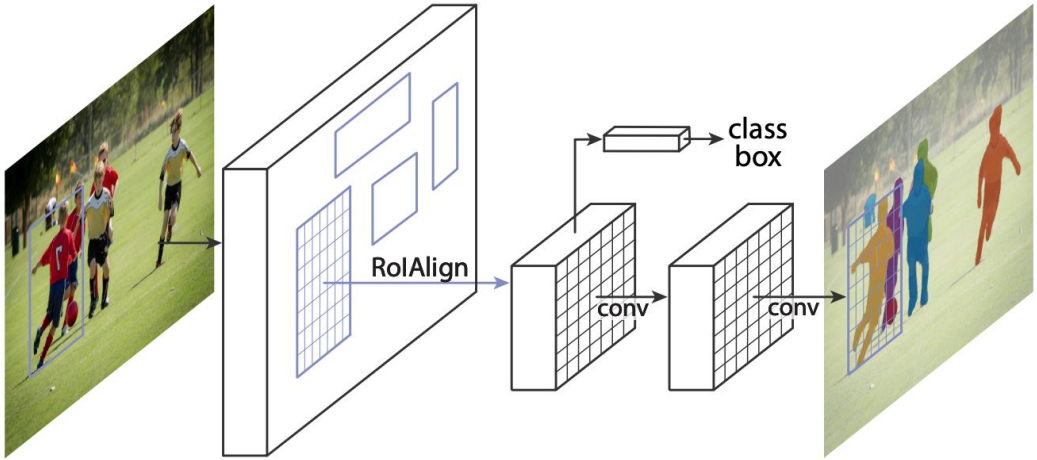
- Selective search algorithm is replaced by learning based method for proposing regions → faster
- These regions are then reshaped using ROI pooling to extract features
- Fully connected layers predict the objects and bounding boxed for them
- Mask RCNN [5] uses the same architecture but has addition head to generate pixel-by-pixel binary mask for each class

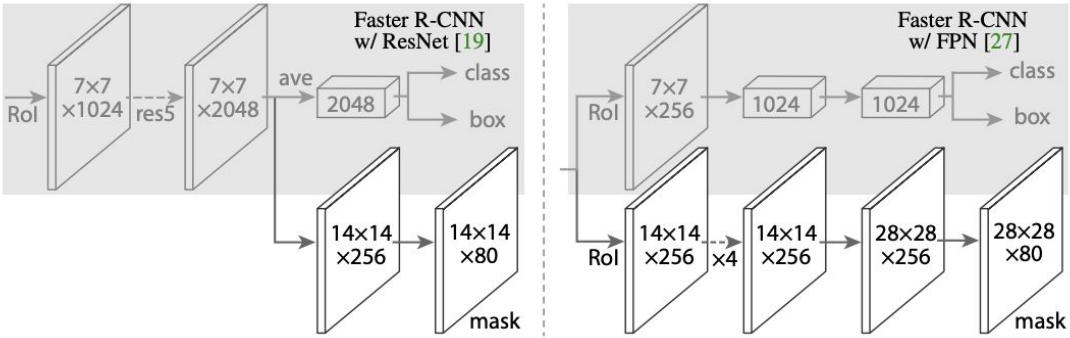
Region Proposal Network [4]



- Proposes k boxes for each sliding window
- Classification layer for objectness score (0 or 1)
- Regression layer to predict 4 coordinates for each box

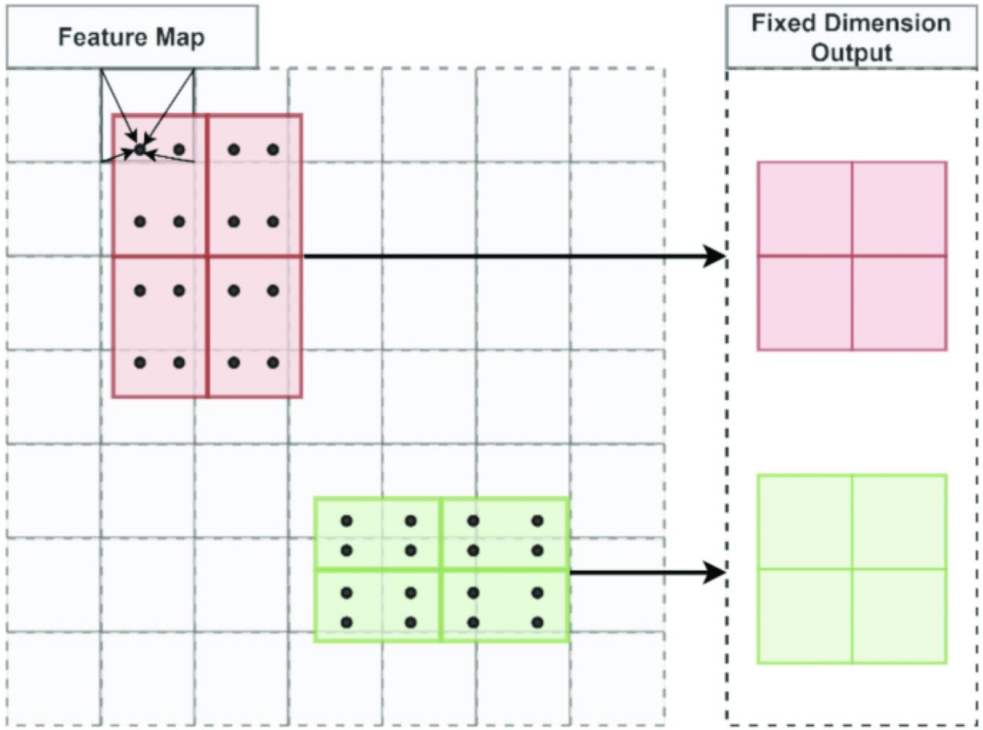
Mask R-CNN[5]





- Extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition
- Is easy to generalize to other tasks, e.g., allowing us to estimate human poses in the same framework - using multiple Feature Pyramid Network (FPN)

ROIAlign



- ROIPool - Quantize floating point values → Subdivide into spatial bins → aggregate the values
- Quantization error was acceptable for bounding boxes but but cannot be used for generating sharp masks
- ROIAlign - Use binary interpolations based on 4 corners → aggregate the features

Results on COCO dataset [3]

method	classifier	**S**	**M**	**L**
R-CNN [9, 10]	SVM	**58.5**	**60.2**	66.0
FRCN [ours]	SVM	56.3	58.7	66.8
FRCN [ours]	softmax	57.1	59.2	**66.9**

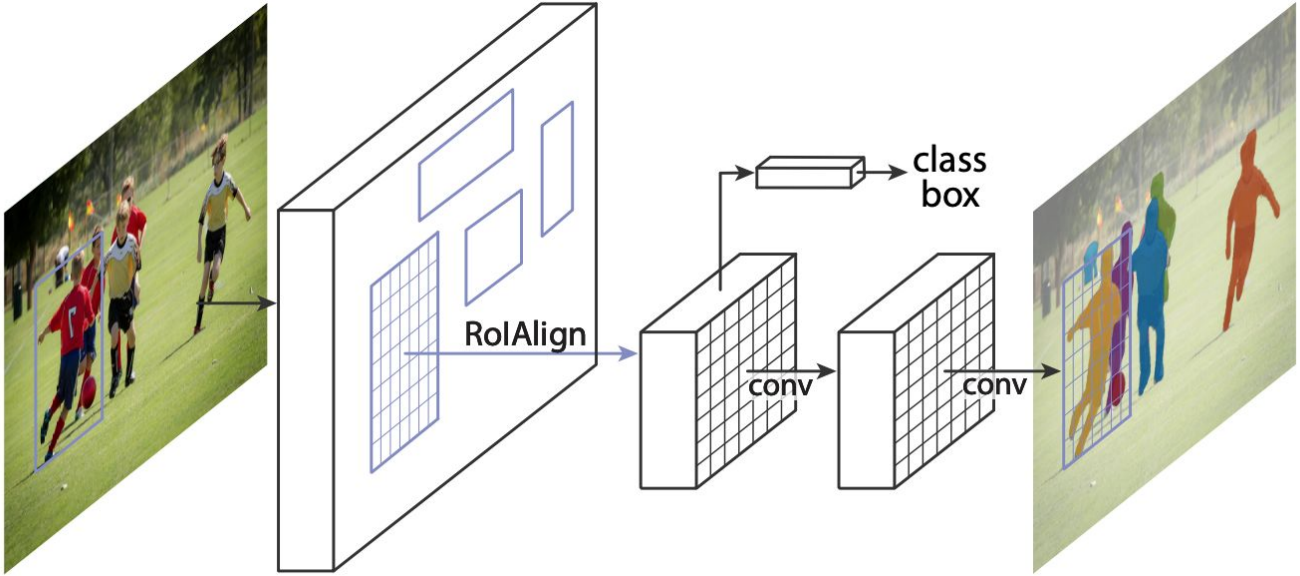
Results on COCO dataset [5]

	backbone	AP^{bb}	AP^{bb}_{50}	AP^{bb}_{75}
Faster R-CNN+++ [19]	ResNet-101-C4	34.9	55.7	37.4
Faster R-CNN w FPN [27]	ResNet-101-FPN	36.2	59.1	39.0
Faster R-CNN by G-RMI [21]	Inception-ResNet-v2 [41]	34.7	55.5	36.7
Faster R-CNN w TDM [39]	Inception-ResNet-v2-TDM	36.8	57.7	39.2
Faster R-CNN, RoIAlign	ResNet-101-FPN	37.3	59.6	40.3
Mask R-CNN	ResNet-101-FPN	38.2	60.3	41.7
Mask R-CNN	ResNeXt-101-FPN	**39.8**	**62.3**	**43.4**

Mask R-CNN

- Proposed by Kaiming He et.al. in 2017 (https://arxiv.org/pdf/1703.06870.pdf)
- Extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition
- Is easy to generalize to other tasks, e.g., allowing us to estimate human poses in the same framework

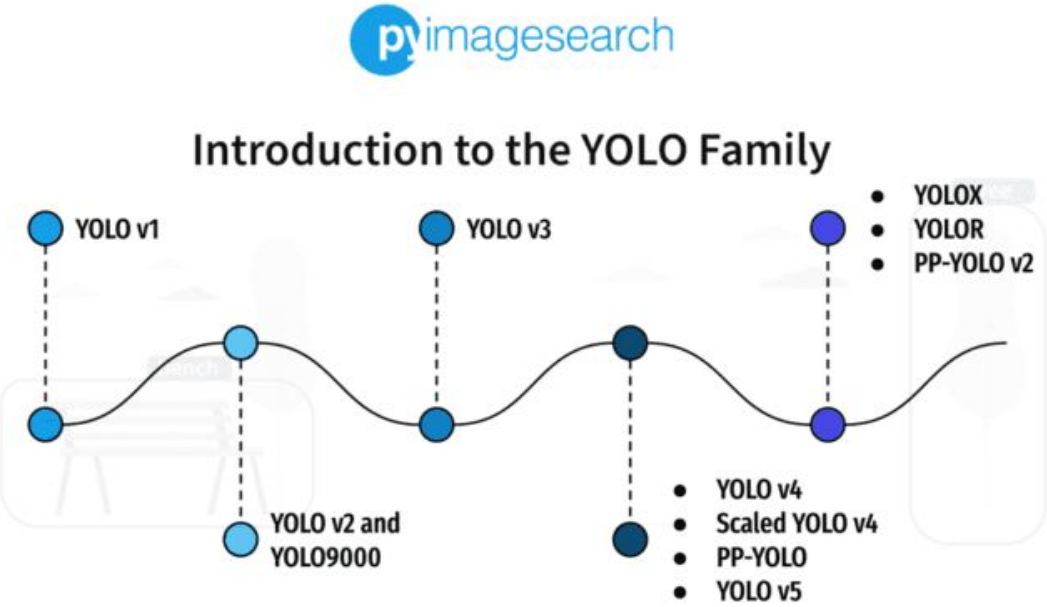
Mask R-CNN



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 R-CNN models - Faster R-CNN (73.2% mAP at 7 FPS) and YOLOv1 (63.4% mAP at 45 FPS)

The YOLO Family

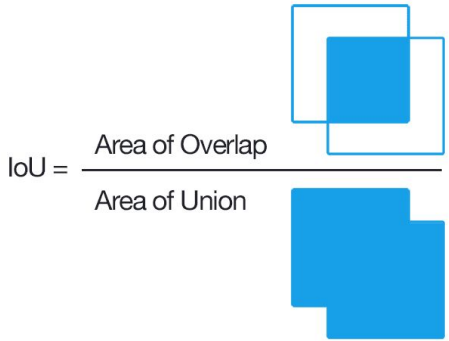


[img ref: https://pyimagesearch.com/2022/04/04/introduction-to-the-yolo-family/]

YOLOv1

Refs:

- https://arxiv.org/pdf/1506.02640.pdf
- https://towardsdatascience.com/yolov1-you-only-look-once-object-detection-e1f3ffec8a89



Motives and Concepts:

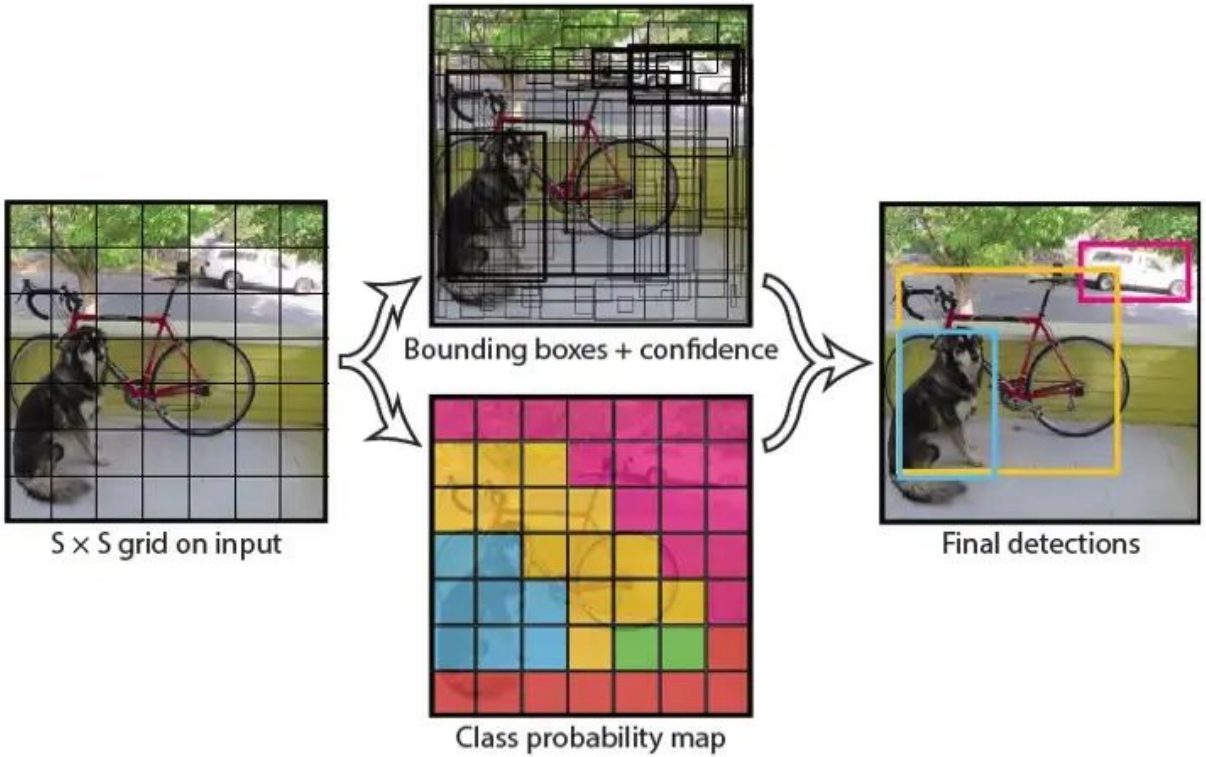
- Divides the input image into an S × S (7x7) grid.
- If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object.
- Each grid cell predicts B (2) bounding boxes and confidence scores for those boxes. These confidence scores reflect how confident the model 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
-

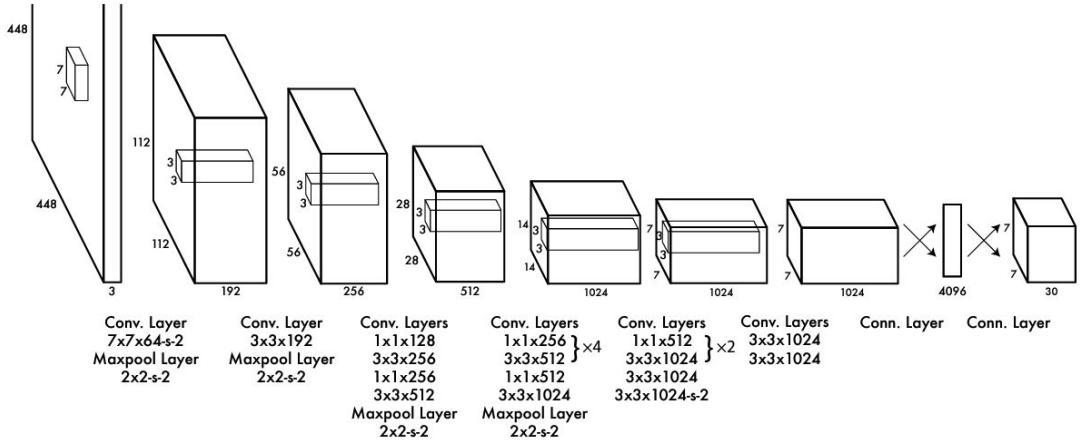
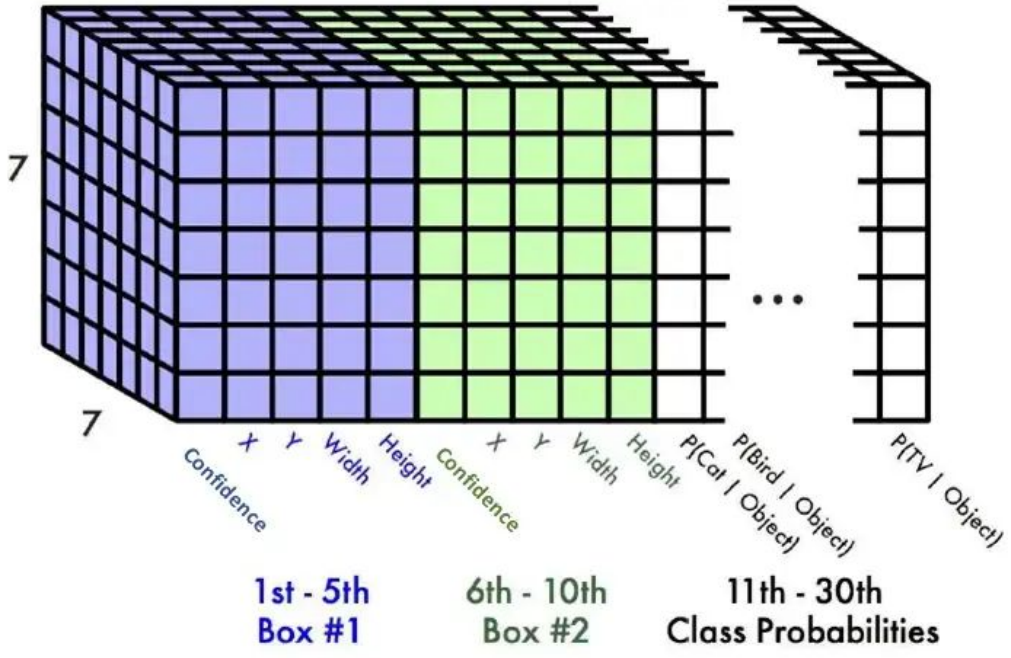


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

YOLOv1

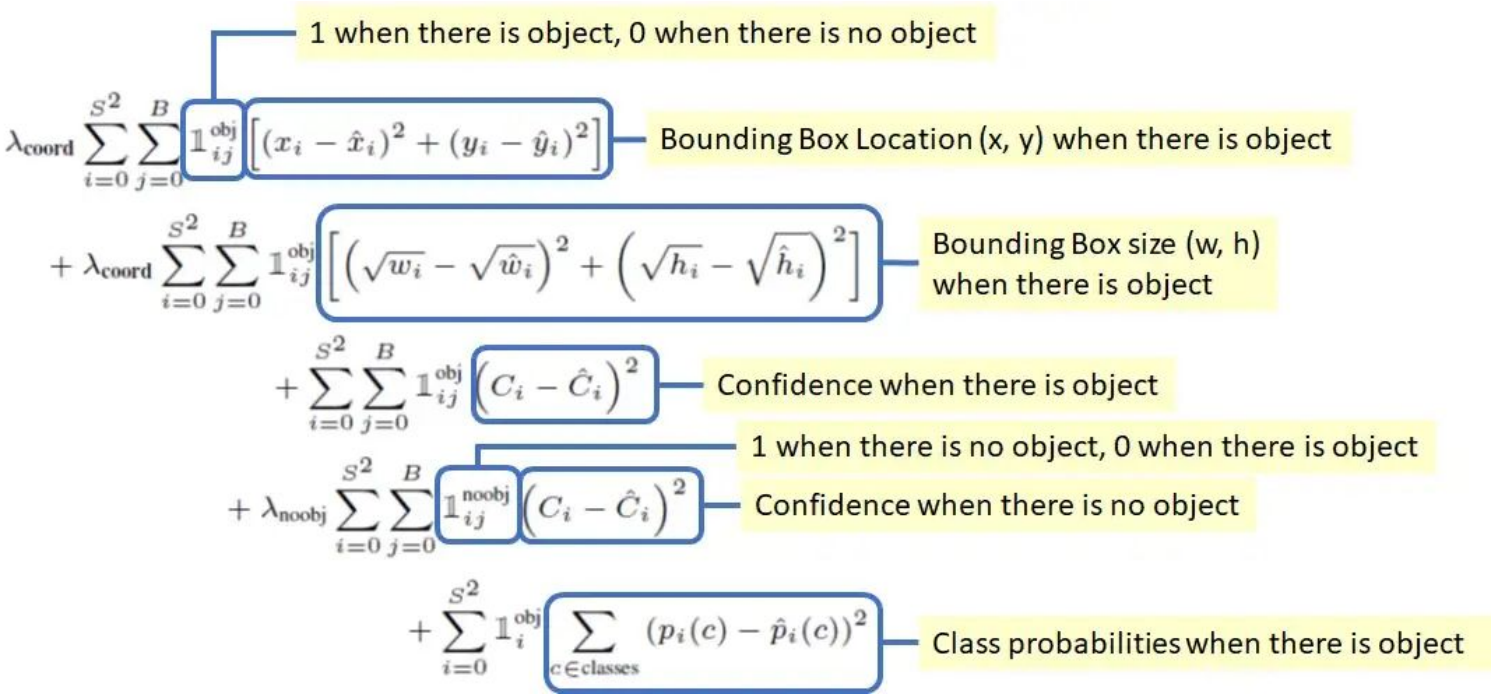
Output and how it makes sense:



Img ref: https://towardsdatascience.com/yolov1-you-only-look-once-object-detection-e1f3ffec8a89

YOLOv1

Loss:


1 when there is object, 0 when there is no object
Bounding Box Location (x, y) when there is object
Bounding Box size (w, h) when there is object
Confidence when there is object
1 when there is no object, 0 when there is object
Confidence when there is no object
Class probabilities when there is object

YOLOv1

Performance

On PASCAL VOC:

Pattern Analysis, Statistical Modelling and Computational Learning - Visual Object Classes Dataset

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

Table 1: Real-Time Systems on PASCAL VOC 2007. Comparing the performance and speed of fast detectors. Fast YOLO is the fastest detector on record for PASCAL VOC detection and is still twice as accurate as any other real-time detector. YOLO is 10 mAP more accurate than the fast version while still well above real-time in speed.

YOLOv2 or YOLO9000: Better, Faster, Stronger

Refs:

- https://towardsdatascience.com/review-yolov2-yolo9000-you-only-look-once-object-detection-7883d2b02a65
- https://arxiv.org/pdf/1612.08242.pdf

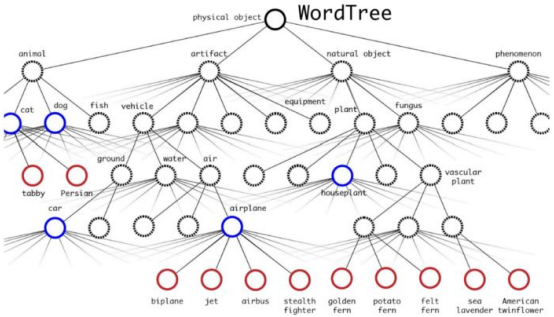
Performance:

- 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:

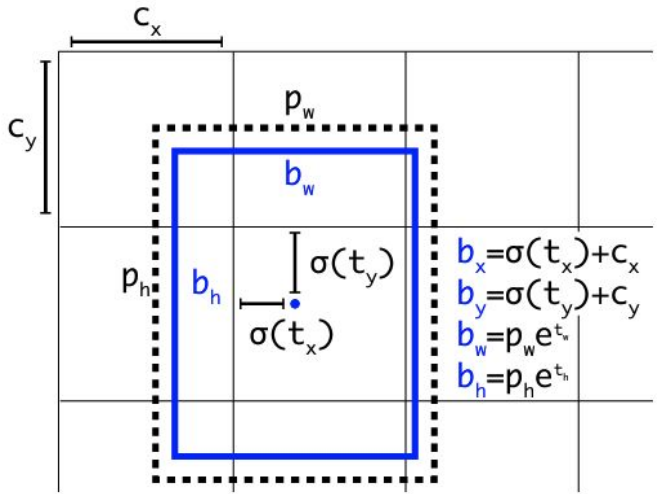


Figure 3: Bounding boxes with dimension priors and location prediction. We predict the width and height of the box as offsets from cluster centroids. We predict the center coordinates of the box relative to the location of filter application using a sigmoid function.

YOLOv2 or YOLO9000: Better, Faster, Stronger

- **Ablation results:**

	YOLO							YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓
anchor boxes?				✓	✓			
new network?					✓	✓	✓	✓
dimension priors?						✓	✓	✓
location prediction?						✓	✓	✓
passthrough?							✓	✓
multi-scale?							✓	✓
hi-res detector?								✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	**78.6**

YOLOv3: An Incremental Improvement

Refs:

- https://arxiv.org/pdf/1804.02767.pdf - Fun Read :D
- https://towardsdatascience.com/review-yolov3-you-only-look-once-object-detection-eab75d7a1ba6

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)

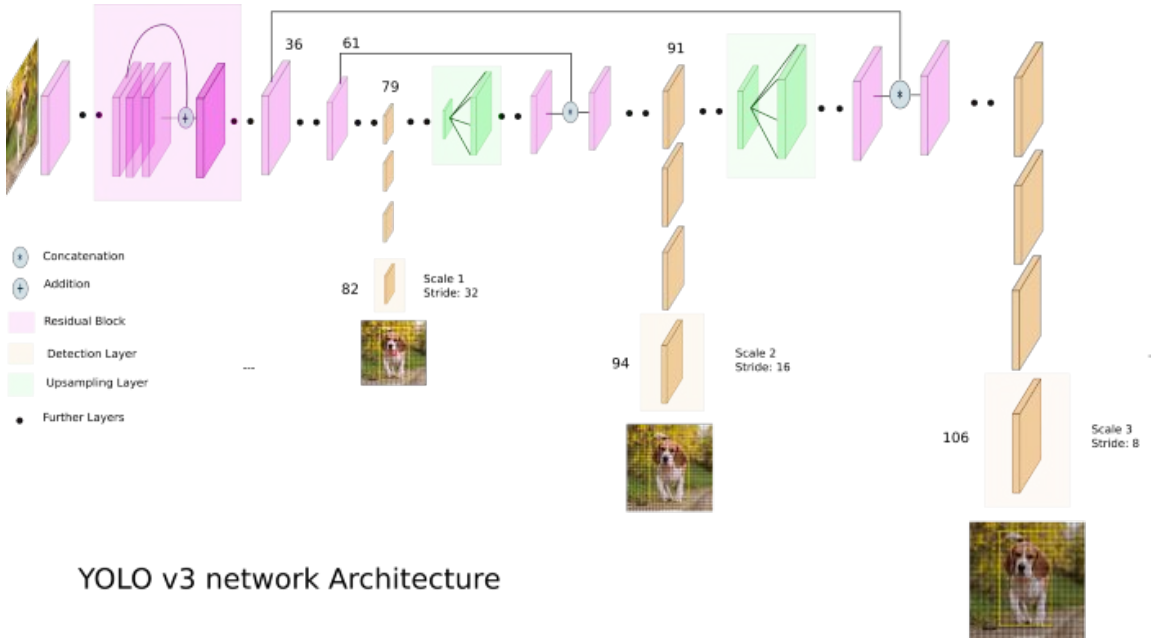
YOLOv3:An Incremental Improvement

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



YOLO v3 network Architecture

[ref: https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b]

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

Refs:

- https://arxiv.org/pdf/2004.10934.pdf
- https://sh-tsang.medium.com/review-yolov4-optimal-speed-and-accuracy-of-object-detection-8198e5b37883

Aims and results:

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

YOLOv4: Optimal Speed and Accuracy of Object Detection

Introduces some definitions which are colloquially used in Object Detection:

- Bag of freebies
  - Methods that only change the training strategy or only increase the training cost - Data Augmentation Strategies
  - Random erase, CutOut, Hide-and-seek, grid-mask, DropOut, DropConnect, DropBlock, MixUp, Using style transfer GAN, label smoothing, IoU losses.
- Bag of specials
  - Methods that only increase the inference cost by a small amount but can significantly improve the accuracy of object detection - architectural changes
  - SPP, ASPP, RFB, Spatial Pyramid Matching (SPM), Skip connections

YOLOv4: Optimal Speed and Accuracy of Object Detection

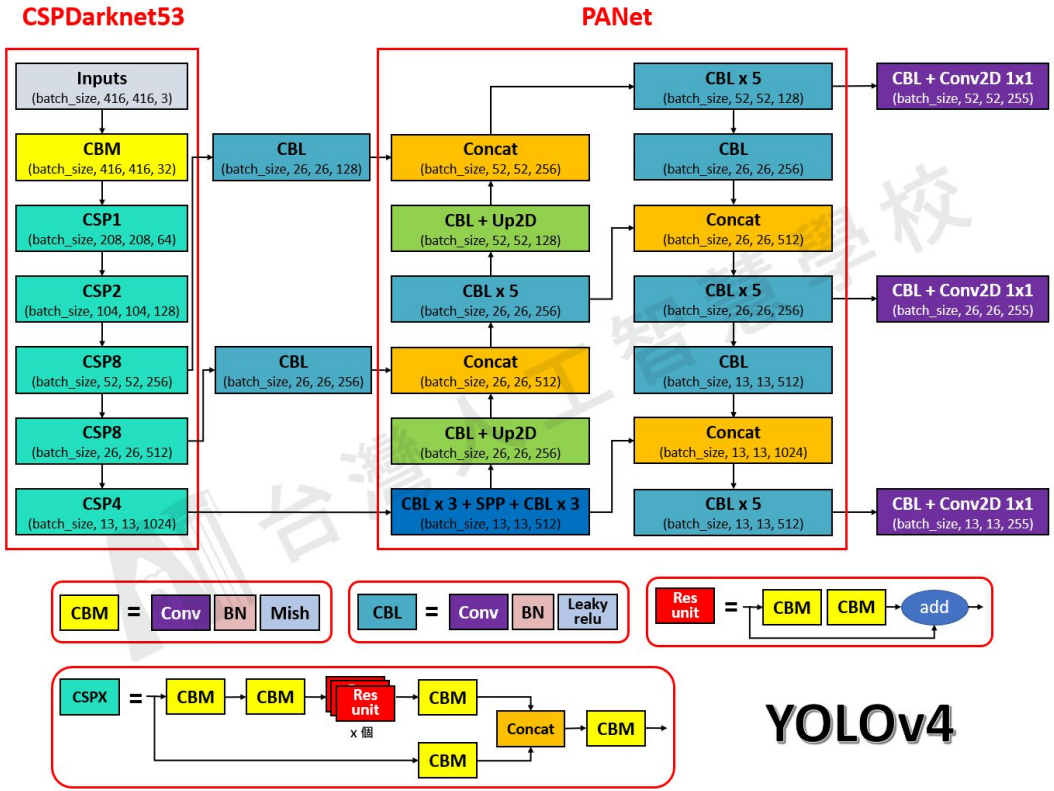
Architecture:

YOLOv4 uses CSPDarknet53 as the backbone, SPP and PANet as the neck, and YOLOv3 as the head.

- Cross Stage Partial Network (CSPNet): https://sh-tsang.medium.com/review-cspnet-a-new-backbone-that-can-enhance-learning-capability-of-cnn-da7ca51524bf
- Spatial Pyramid Pooling (SSP): https://medium.com/coinmonks/review-sppnet-1st-runner-up-object-detection-2nd-runner-up-image-classification-in-ilsvrc-906da3753679
- Path Aggregation Network (PANet): https://becominghuman.ai/reading-panet-path-aggregation-network-1st-place-in-coco-2017-challenge-instance-segmentation-fe4c985cad1b

YOLOv4: Optimal Speed and Accuracy of Object Detection

Architecture:



[image ref: https://aiacademy.tw/yolo-v4-intro/]

YOLOv4: Optimal Speed and Accuracy of Object Detection

- Additional Improvements:
  - Using a Mosaics of images (augmentation technique) to train. Helps with context mixing. Reduced need of large mini-batches.
  - Self Adversarial Training (SAT) - P1: altering the image instead of the weights, P2: detect object in this image.
  - Cross mini-Batch Normalization (CmBN) - Collects statistics only between smaller portions of a mini-batch

Figure 3: Mosaic represents a new method of data augmentation.

[img ref: https://arxiv.org/pdf/2004.10934.pdf]

YOLOv4: Optimal Speed and Accuracy of Object Detection

Some Bag of Freebies and Bag of Specials ablation results:

MixUp	CutMix	Mosaic	Bluring	Label Smoothing	Swish	Mish	Top-1	Top-5
							77.2%	93.6%
	✓	✓		✓			**77.8%**	**94.4%**
	✓	✓		✓		✓	**78.7%**	**94.8%**

Influence of BoF and Mish on the CSPDarknet-53 classifier accuracy

YOLOv4: Optimal Speed and Accuracy of Object Detection

Performance:

- YOLOv4 outperforms EfficientDet, ASFF, NAS-FPN, CenterNet, CornerNet, etc.

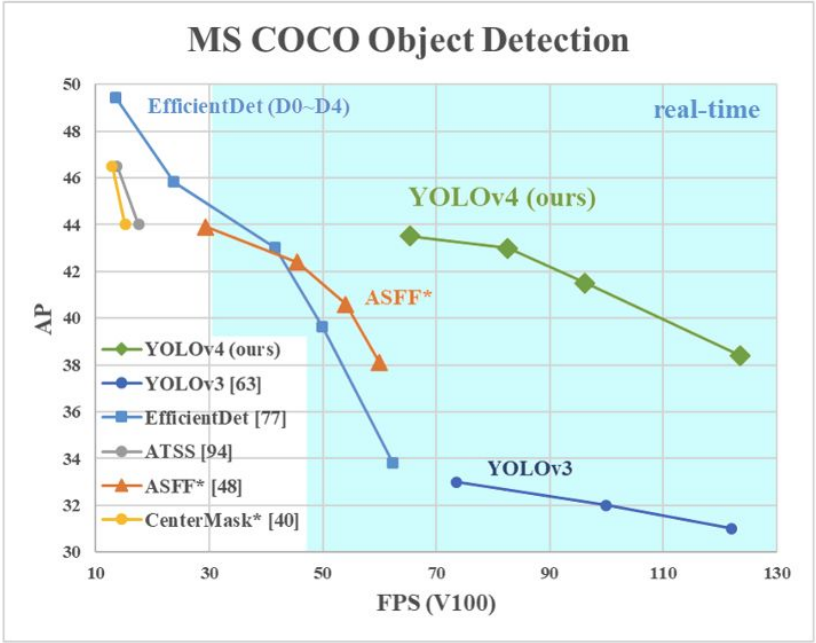


Figure 1: Comparison of the proposed YOLOv4 and other state-of-the-art object detectors. YOLOv4 runs twice faster than EfficientDet with comparable performance. Improves YOLOv3's AP and FPS by 10% and 12%, respectively.

Scaled YOLOv4/YOLOv4-CSP

Refs:

- https://arxiv.org/pdf/2011.08036.pdf
- https://sh-tsang.medium.com/review-scaled-yolov4-scaling-cross-stage-partial-network-51e3c515b0a7

Overview:

- Employs scaling of size, depth, and width of the network layers with Cross Stage Partial Network (CSPNet) blocks.
- CSP to every section of YOLOv4 network - Backbone, Neck, and Head.
- Essentially faster inference for larger resolution capacities.
- A fully CSP-ized model YOLOv4-P5 is designed and can be scaled up to YOLOv4-P6 and YOLOv4-P7.

If interested, please refer to their paper for ablation studies.

Scaled YOLOv4/YOLOv4-CSP

Results:

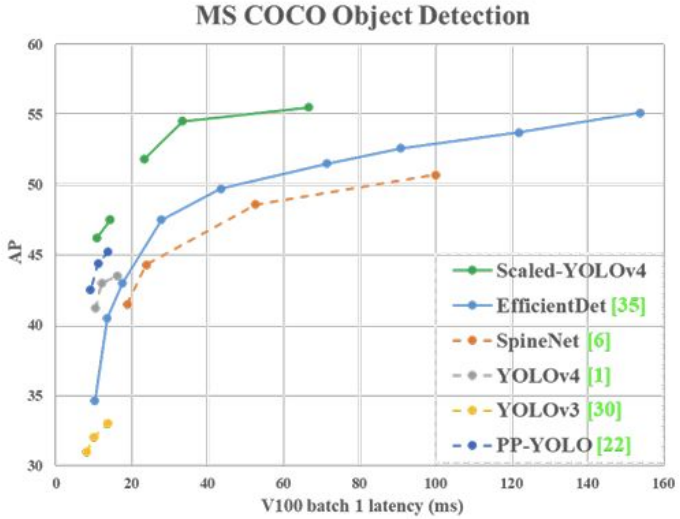


Figure 1: Comparison of the proposed YOLOv4 and other state-of-the-art object detectors. The dashed line means only latency of model inference, while the solid line include model inference and post-processing.

PP-YOLO

Refs:

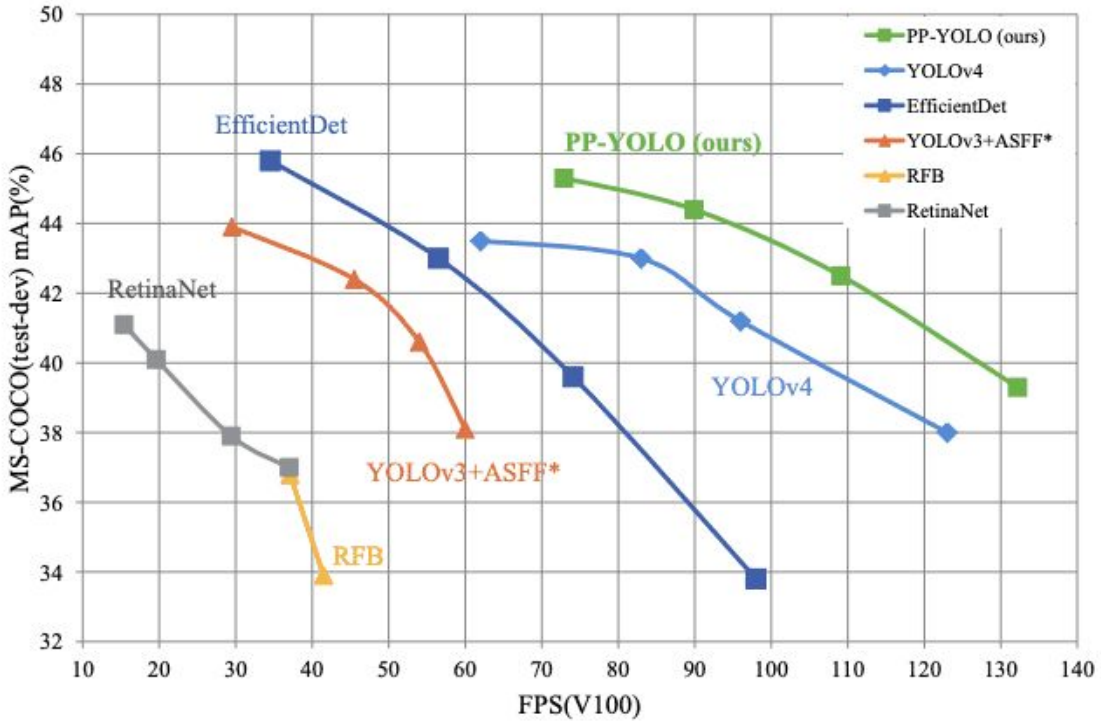
- https://arxiv.org/pdf/2007.12099.pdf
- https://pyimagesearch.com/2022/04/04/introduction-to-the-yolo-family/

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

Refs:

- https://github.com/ultralytics/yolov5 - June 2020
- Only YOLO without a paper - Still patched by Ultralytics

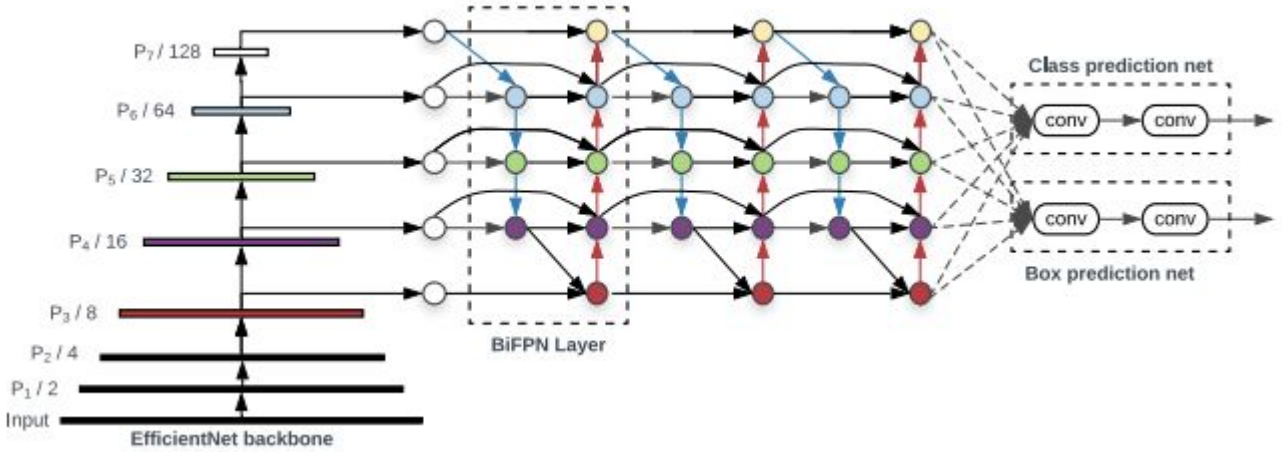
Overview:

- 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:

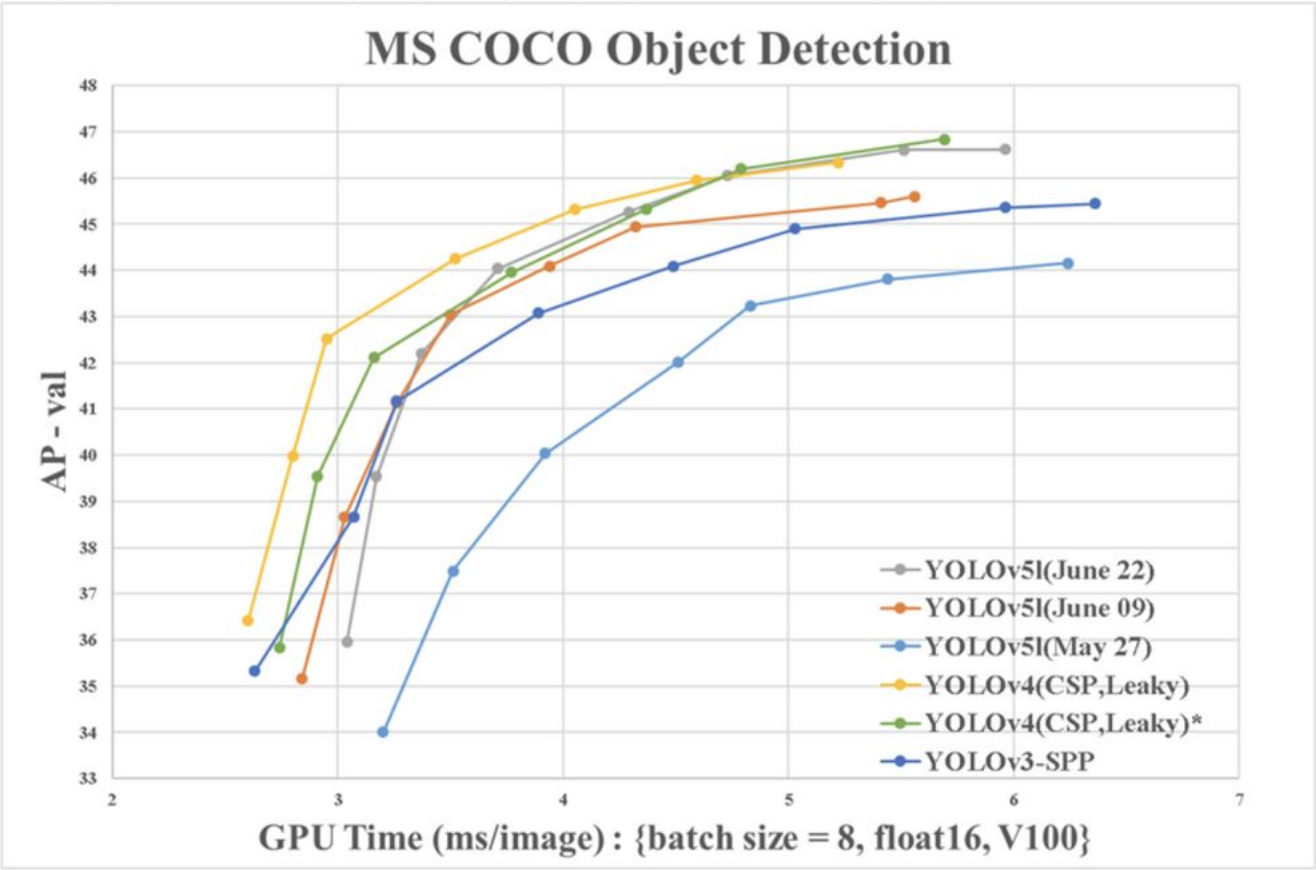
- Backbone: A convolutional neural network that aggregates and forms image features at different granularities. (CSP)
- Neck: A series of layers to mix and combine image features to pass them forward to prediction. (PANet)
- Head: Consumes features from the neck and takes box and class prediction steps.



[ref: https://blog.roboflow.com/yolov5-improvements-and-evaluation/]

YOLOv5

Performance:



YOLOX

Refs:

- https://arxiv.org/pdf/2107.08430.pdf
- https://github.com/Megvii-BaseDetection/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)

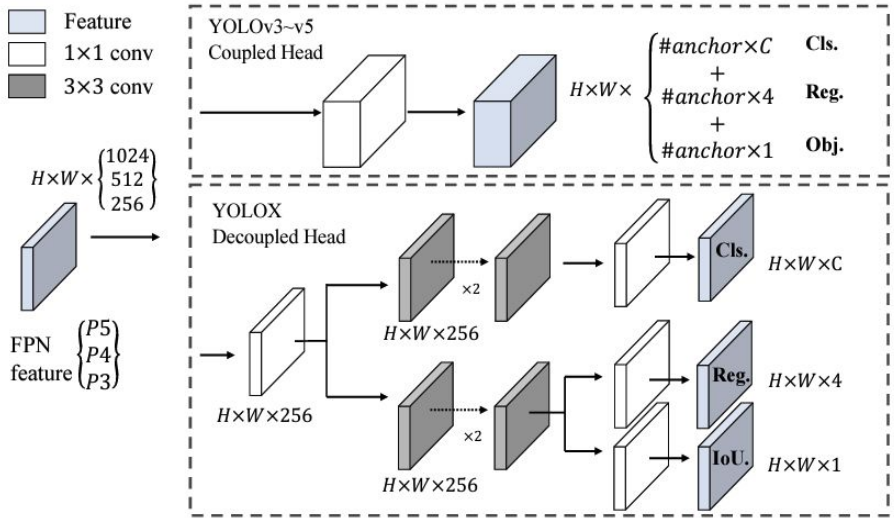
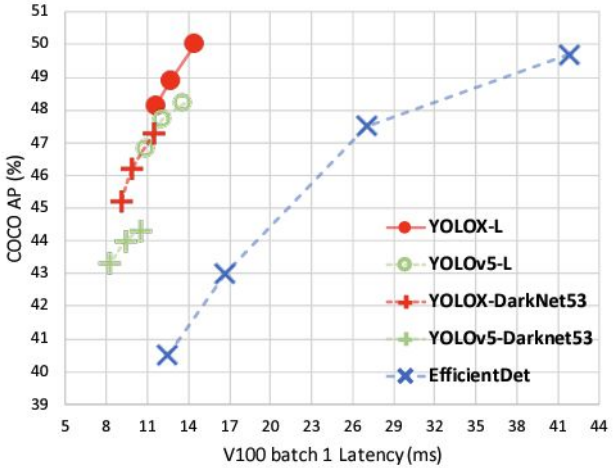


Figure 2: Illustration of the difference between YOLOv3 head and the proposed decoupled head. For each level of FPN feature, we first adopt a 1×1 conv layer to reduce the feature channel to 256 and then add two parallel branches with two 3×3 conv layers each for classification and regression tasks respectively.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

Refs:

- https://arxiv.org/pdf/2207.02696.pdf
- https://sh-tsang.medium.com/review-yolov7-trainable-bag-of-freebies-sets-new-state-of-the-art-for-real-time-object-detectors-b29f33c041a8

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.

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

Architecture (cont.)

- a convolutional layer with residual or concatenation is replaced by **re-parameterized convolution**, there should be no identity connection.
- **Auxiliary Heads** for different resolution of detection
- *Deep supervision* needs to be trained on the target objectives regardless of the circumstances of auxiliary head or lead head. - unexplored and hence;
- **New Label assigner** for each of the heads (Lead head guided label assigner and Coarse-to-fine lead head guided label assigner)

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

Performance:

- highest accuracy 56.8% AP among all known real-time object detectors with 30 FPS or higher on GPU V100.

- YOLOv7-E6 object detector (56 FPS V100, 55.9% AP) outperforms both transformer-based detector SWINL Cascade-Mask R-CNN (9.2 FPS A100, 53.9% AP) by 509% in speed and 2% in accuracy.

- YOLOv7-E6 also beats convolutional based detector ConvNeXt-XL Cascade-Mask R-CNN (8.6 FPS A100, 55.2% AP) by 551% in speed and 0.7% AP in accuracy.
-

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

Performance(cont.)

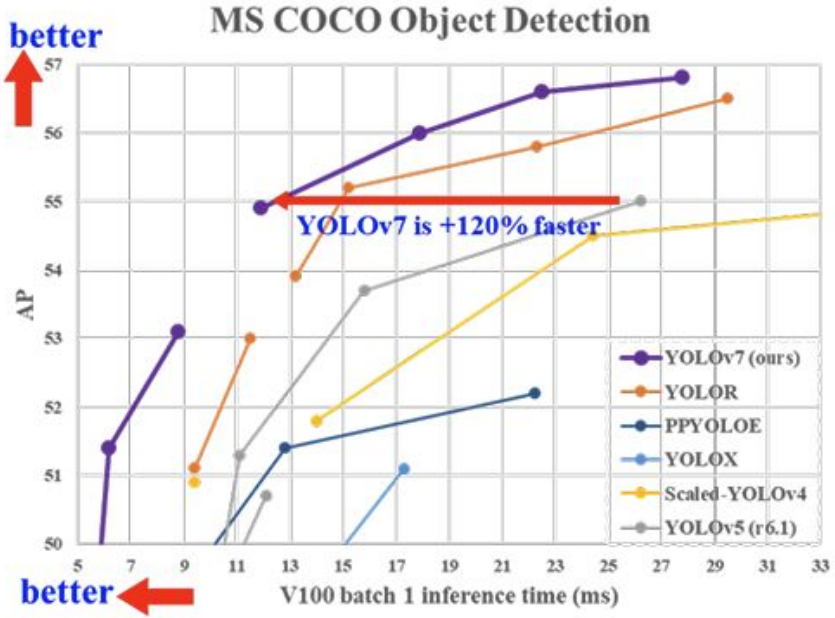


Figure 1: Comparison with other real-time object detectors, our proposed methods achieve state-of-the-arts performance.

YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications

Refs:

- https://arxiv.org/pdf/2209.02976.pdf - June 2022
- https://github.com/meituan/YOLOv6
- https://learnopencv.com/yolov6-object-detection/

Motives:

- It if for use in industrial applications, trained for more epochs (400)

YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications

Architecture and Labels:

- Experiments show **SimOTA** and **TAL** (Task Aligned Assignment) are the best label assignment strategies for anchor-free object detection models
- Revised **Reparameterized backbone and neck** - In reparameterization, the network structure changes during training and inference
- The entire backbone of the YOLOv6 architecture is called **EfficientRep** - EfficientRep Torch Implementation on HuggingFace
- Medium and Large models, the YOLOv6 architecture uses reparameterized versions of the CSP backbone. They call it the **CSPStackRep**.

YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications

Architecture:

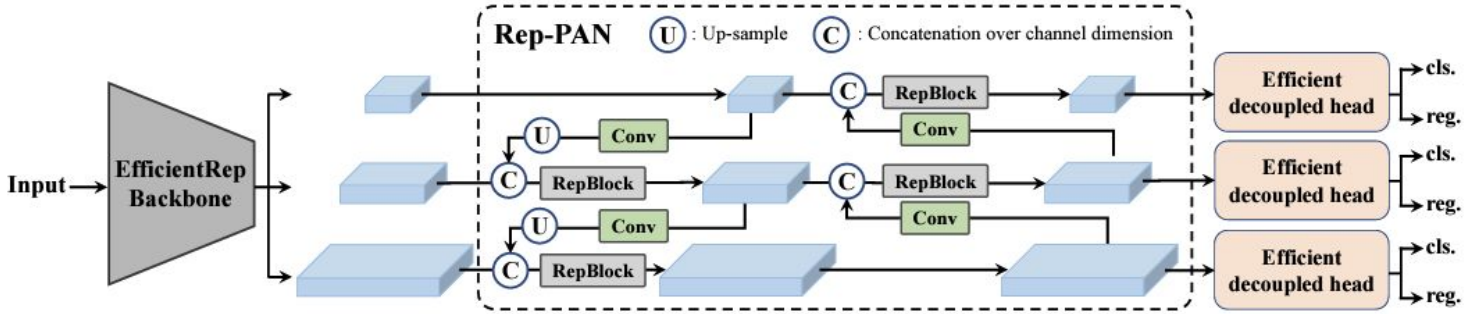


Figure 2: The YOLOv6 framework (N and S are shown). Note for M/L, RepBlocks is replaced with CSPStackRep.

YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications

Loss:

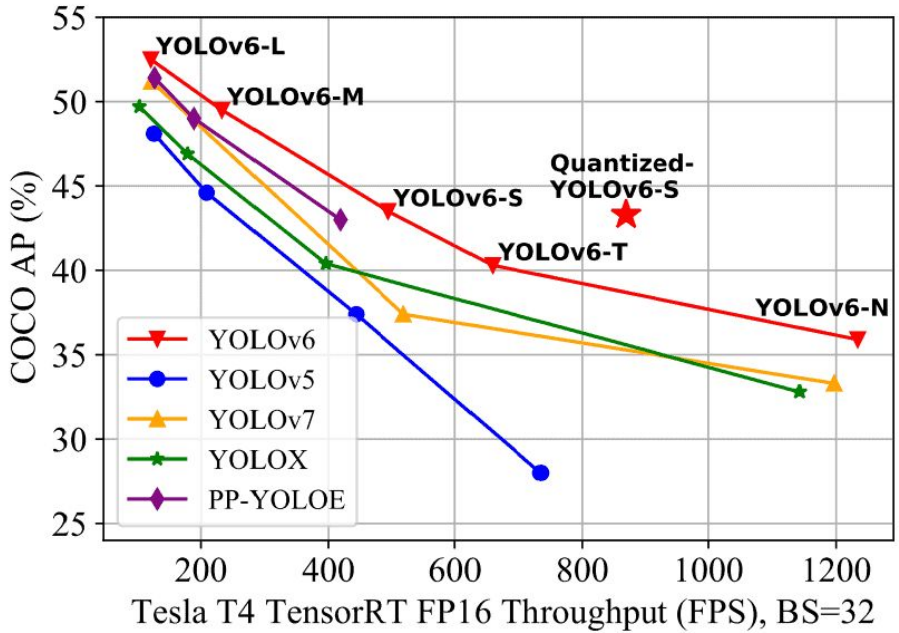
- VeriFocal Loss (VFL) - as classification loss
- Distribution Focal Loss (DFL) - along with SIoU or GIoU as box regression loss

Performance:

- YOLOv6 Nano model has achieved an mAP of **35.6%** on the COCO dataset. Also, it runs at more than **1200 FPS** on an NVIDIA Tesla T4 GPU with a batch size of 32.
- YOLOv6-L model gives an mAP of **52.5%** on the COCO validation dataset while still being able to maintain an **FPS of 121.**

YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications

Performance (cont):



(other) SOTA Architectures

- Detection:
  - MobileNet
  - SqueezeNet
  - SIFT
- Segmentation
  - U-Net
  - Fast FCN
  - Gated SCNN
  - DeepLab
  - Loss Functions

MobileNet

- Proposed by Andrew G. Howard in 2017 (https://arxiv.org/abs/1704.04861)
- 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

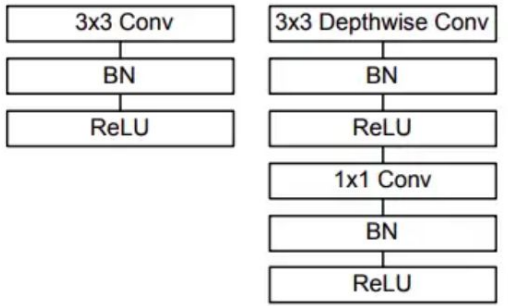
Figure 6. Example objection detection results using MobileNet SSD.

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

SqueezeNet

- Introduced by Forrest N. Iandola et.al. in 2016 (https://arxiv.org/abs/1602.07360)
- Smaller DNNs require less communication across servers during distributed training.
- Smaller DNNs require less bandwidth to export a new model from the cloud to an autonomous car.
- Smaller DNNs are more feasible to deploy on FPGAs and other hardware with limited memory
- Derivative of AlexNet (and in a way, LeNet)

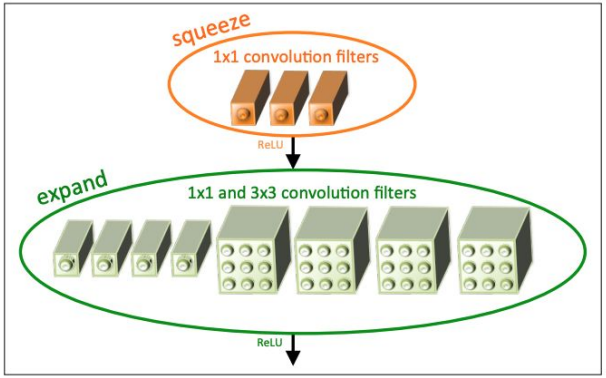


Figure 1: Microarchitectural view: Organization of convolution filters in the **Fire module**. In this example, $s_{1x1} = 3$, $e_{1x1} = 4$, and $e_{3x3} = 4$. We illustrate the convolution filters but not the activations.

SqueezeNet

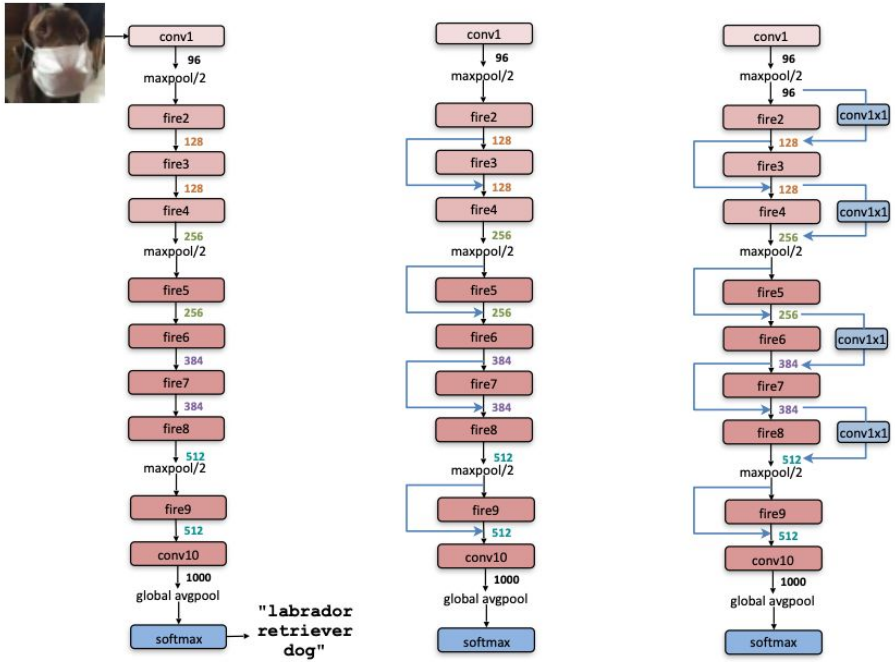
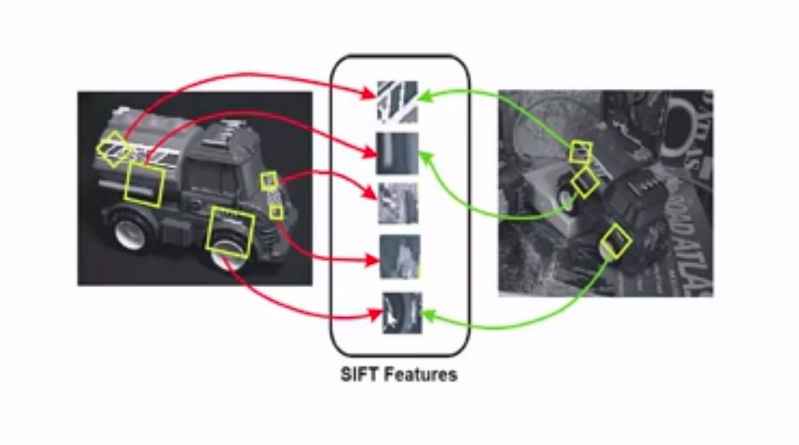


Figure 2: Macroarchitectural view of our SqueezeNet architecture. Left: SqueezeNet (Section 3.3); Middle: SqueezeNet with simple bypass (Section 6); Right: SqueezeNet with complex bypass (Section 6).

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

U-Net

- Proposed by Olaf Ronneberger et al. in 2015 (https://arxiv.org/abs/1505.04597)
- Consists of a contracting path to capture context and a symmetric expanding path that enables precise localization
- Key elements:
  - Encoder Network
  - Skip Connections
  - Bridge
  - Decoder Network

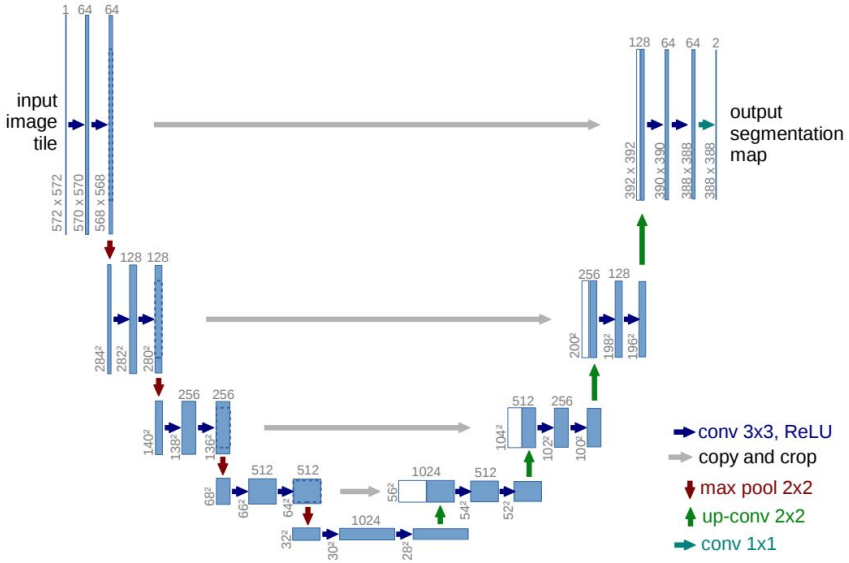


Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Fast FCN

- Introduced by Huikai Wu et al. in 2019 (https://arxiv.org/abs/1903.11816)
- Proposes a novel joint upsampling module named Joint Pyramid Upsampling (JPU) by formulating the task of extracting high-resolution feature maps into a joint upsampling problem
- Reduces the computation complexity by more than three times without performance loss

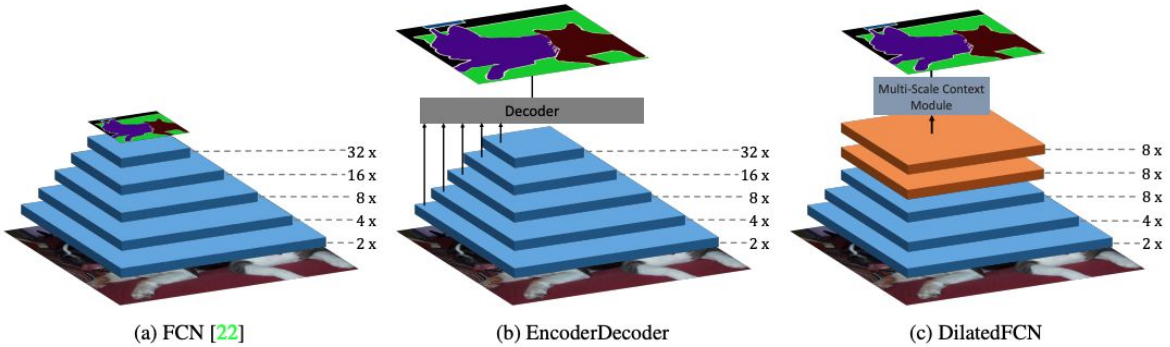
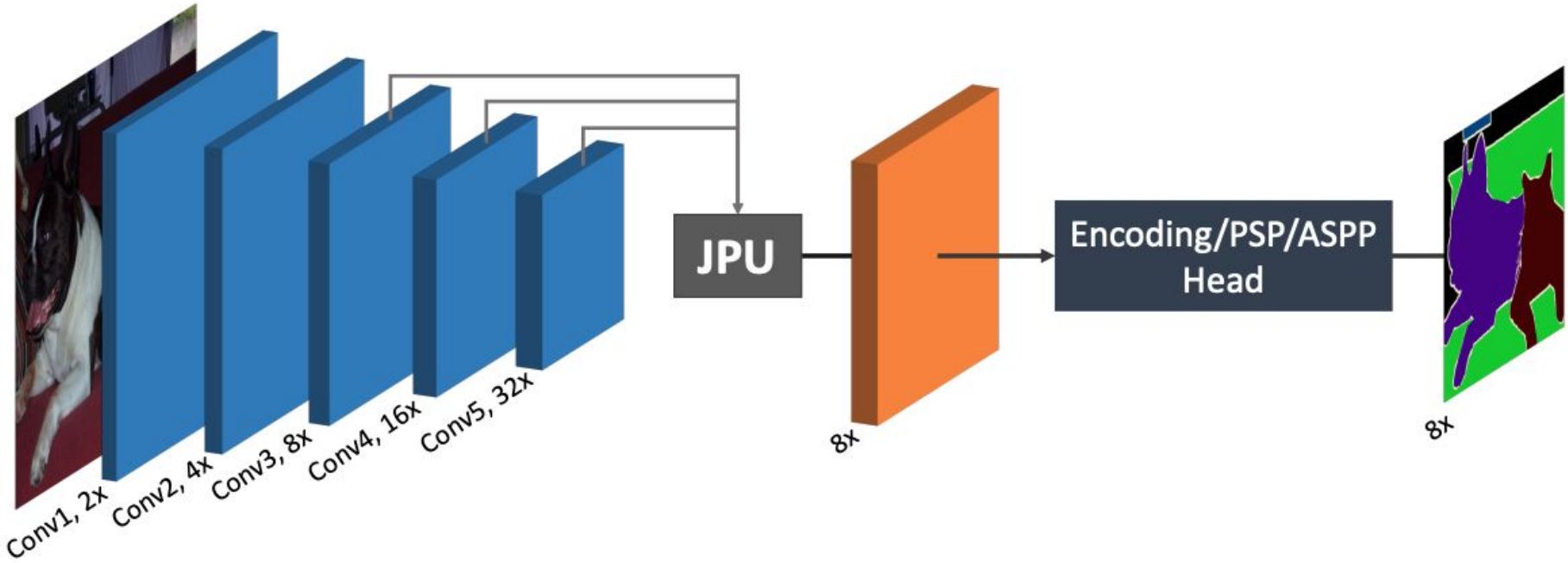


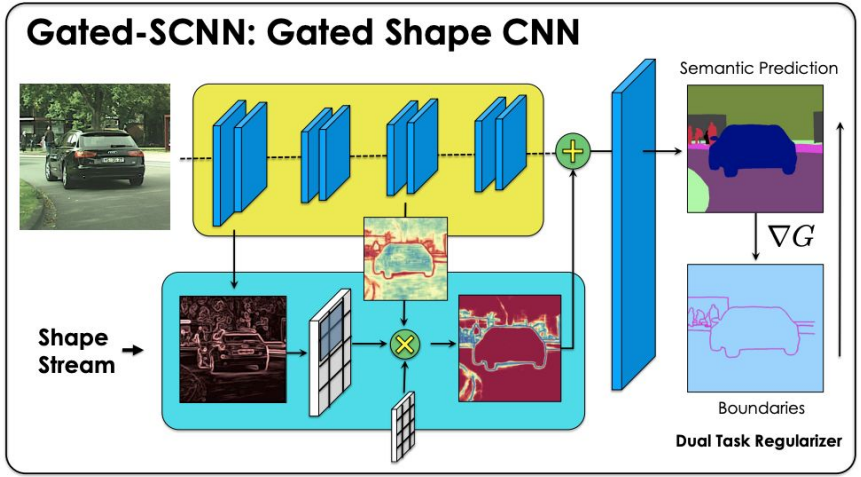
Figure 1: **Different types of networks for semantic segmentation.** (a) is the original FCN, (b) follows the encoder-decoder style, and (c) employs dilated convolutions to obtain high-resolution final feature maps. Best viewed in color.

Fast FCN

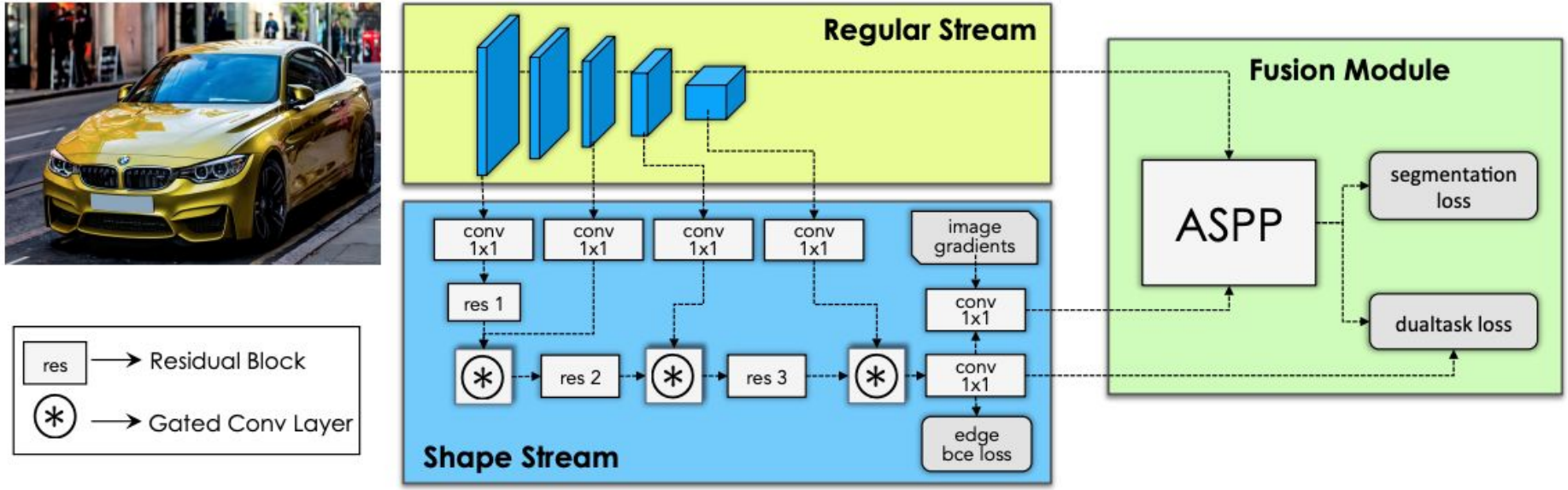

Conv1, 2x
Conv2, 4x
Conv3, 8x
Conv4, 16x
Conv5, 32x
JPU
8x
Encoding/PSP/ASPP Head
8x

Gated-SCNN

- Introduced by Towaki Takikawa et al. in 2019 in collaboration with NVIDIA (https://arxiv.org/pdf/1907.05740.pdf)
- Proposes a new two-stream CNN architecture for semantic segmentation that explicitly wires shape information as a separate processing branch

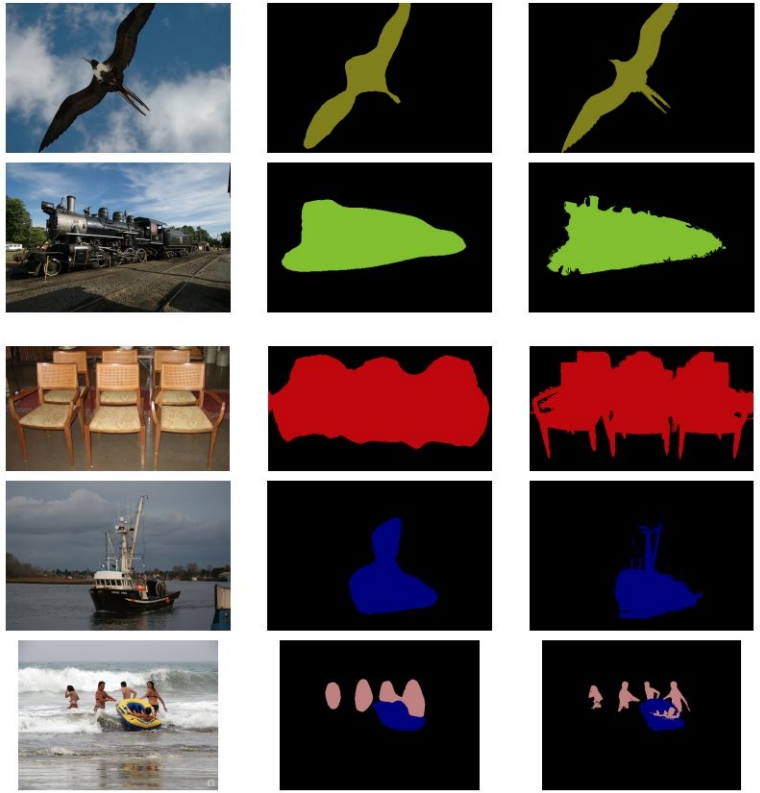


Gated-SCNN

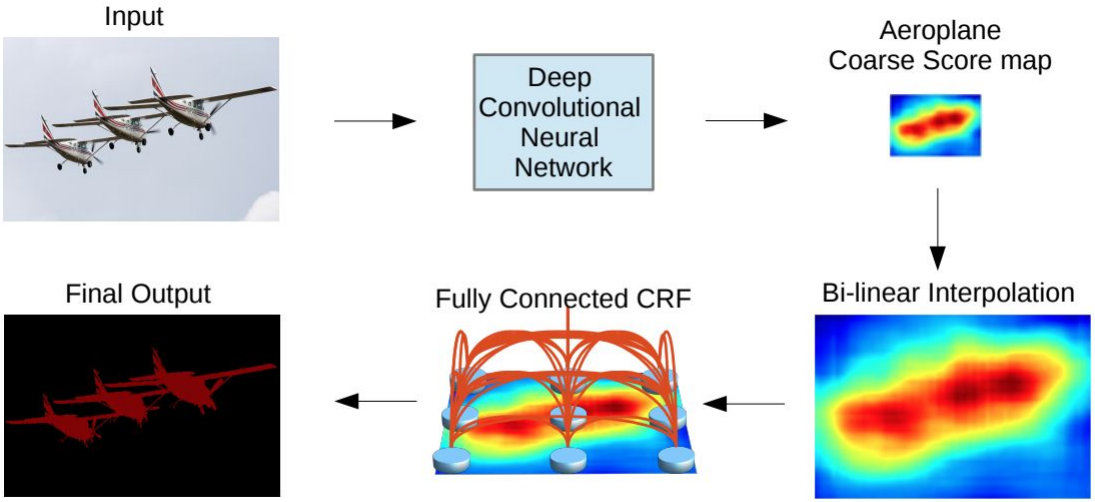

Regular Stream
Fusion Module
ASPP
segmentation loss
dualtask loss
conv 1x1
conv 1x1
conv 1x1
conv 1x1
image gradients
res 1
conv 1x1
res 2
res 3
conv 1x1
edge bce loss
Shape Stream
res
Residual Block
Gated Conv Layer

DeepLab

- Introduced by Liang-Chieh CHen et al. in 2014 (https://arxiv.org/abs/1412.7062v4)
- First, the input image goes through the network with the use of dilated convolutions. Then the output from the network is bilinearly interpolated and goes through the fully connected CRF to fine tune the result to obtain the final predictions.



DeepLab



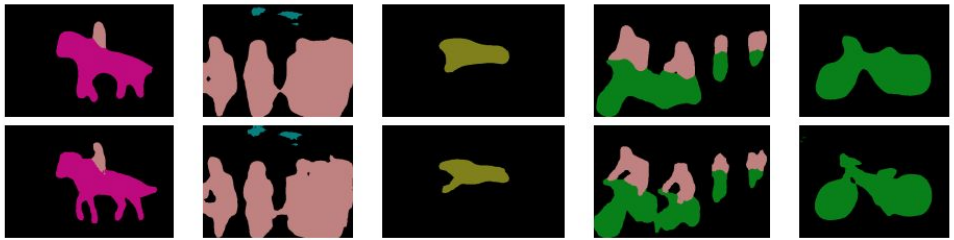


Figure 4: Incorporating multi-scale features improves the boundary segmentation. We show the results obtained by DeepLab and DeepLab-MSc in the first and second row, respectively. Best viewed in color.

Segmentation Loss Functions

Semantic segmentation models usually use a simple cross-categorical entropy loss function during training. However, if you are interested in getting the granular information of an image, then you have to revert to slightly more advanced loss functions.

Segmentation Loss Functions

- Focal Loss: $\mathrm{FL}(p_t) = -(1-p_t)^{\gamma}\log(p_t).$
- Dice Loss: $DSC = \frac{2|X \cap Y|}{|X| + |Y|}$
- Intersection over Union (IoU)-balanced Loss: $IoU = TP/(TP + FP + FN)$
- Weighted cross-entropy: $\mathrm{WCE}\left(p, \hat{p}\right) = -\left(\beta p \log(\hat{p}) + (1-p)\log(1-\hat{p})\right)$
- Boundary Loss: $\mathrm{Dist}(\partial G, \partial S) = \int_{\partial G} \|y_{\partial S}(p) - p\|^2 dp$
- Lovasz-Softmax Loss: $\mathrm{loss}(\boldsymbol{f}) = \frac{1}{|\mathcal{C}|}\sum_{c\in\mathcal{C}} \overline{\Delta_{J_c}}(\boldsymbol{m}(c))$

Segmentation Loss Functions

- TopK loss:
  - Ensure that networks concentrate on hard samples during the training process.
- Distance penalized CE loss:
  - Directs the network to boundary regions that are hard to segment.
- Sensitivity-Specificity (SS) loss:
  - Computes the weighted sum of the mean squared difference of specificity and sensitivity.
- Hausdorff distance(HD) loss:
  - Estimates the Hausdorff distance from a convolutional neural network.

Other References:

1. Image Segmentation Using Deep Learning: A Survey by Shervin et. al - https://arxiv.org/pdf/2001.05566.pdf
2. Rich feature hierarchies for accurate object detection and semantic segmentation - https://arxiv.org/abs/1311.2524v5
3. Fast R-CNN - https://arxiv.org/abs/1504.08083
4. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks - https://arxiv.org/abs/1506.01497
5. Mask R-CNN - https://arxiv.org/abs/1703.06870