SSD Training Tutorial

J. Cory – Embedded Vision Specialist
1.11.19

Credit to: Louie Valena for detection metrics slides

Introduction

ML Task: Detection

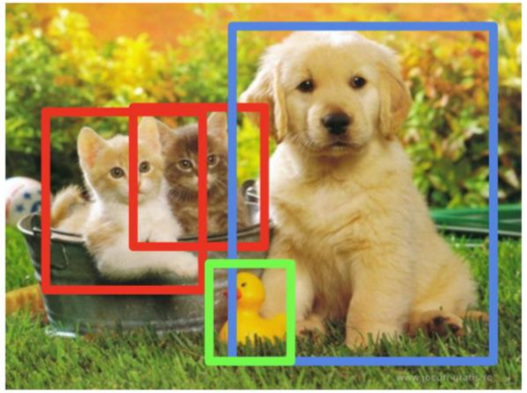
Deep CNN Architecture

Backbone

Object Classification

Bounding Box Regression

- VGG
- Inception
- ResNet



- **SSD、YOLO、Faster RCNN**

- **Task:**
  - Input: Image
  - Output: Class label and location of objects
  - Evaluation metric:
    - mAP, mean average precision
    - Precision at fixed recall
    - Recall at fixed precision

- **Applications:**
  - Face detection
  - License plate detection
  - Pedestrian, cyclist and car detection in surveillance/automotive.

Detection Metrics

Metrics: Precision and Recall

> **Precision: how many selected items are relevant?**

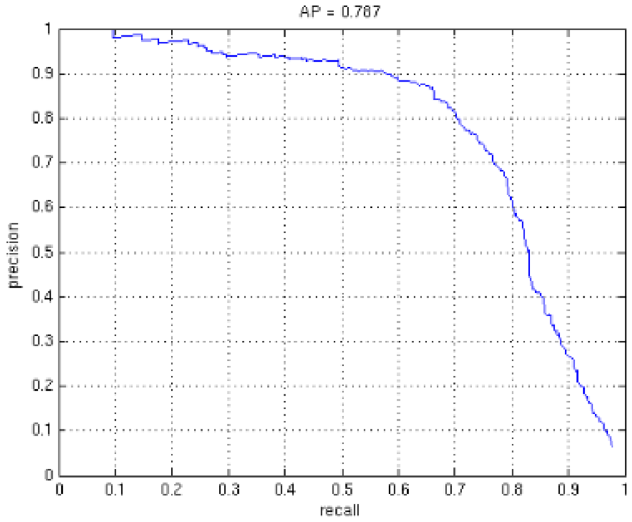
> **Recall: how many relevant items are selected?**

> **Ex.: 12 dogs and several cats are in an image**

>> The classifier predicted the presence of 8 dogs; it properly identified 5 dogs as dogs (true positive), but it incorrectly identified 3 cats as dogs (false positive)

>> Precision = TP/(FP + TP) = 5/(3 + 5) = 0.625

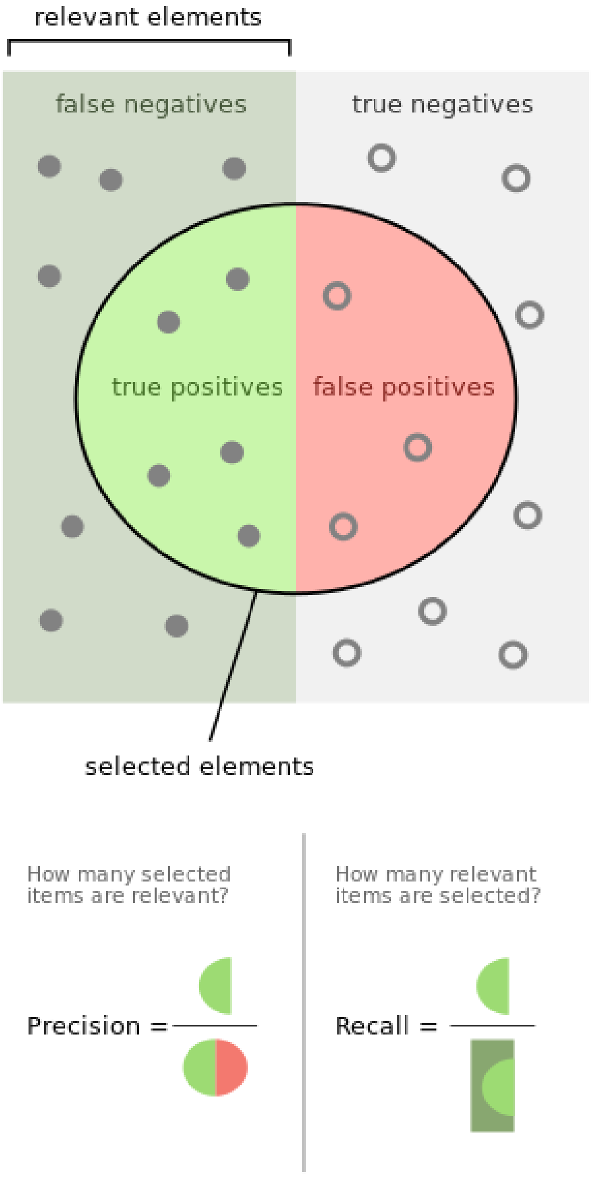
>> Recall = 5/12 = 0.4167



https://www.aiukraine.com/wp-content/uploads/2016/09/Pashchenko_AIUkraine_2016.pdf

Average Precision (AP) is the area under the precision-recall curve

mean Average Precision (AP) is the average of the AP of all classes. It provides an indication of how well objects are localized and identified.

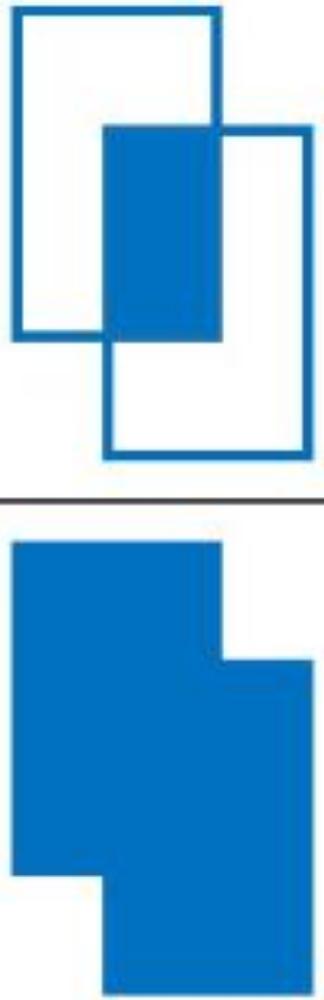


https://en.wikipedia.org/wiki/Precision_and_recall

XILINX

Intersection over Union (1/2)

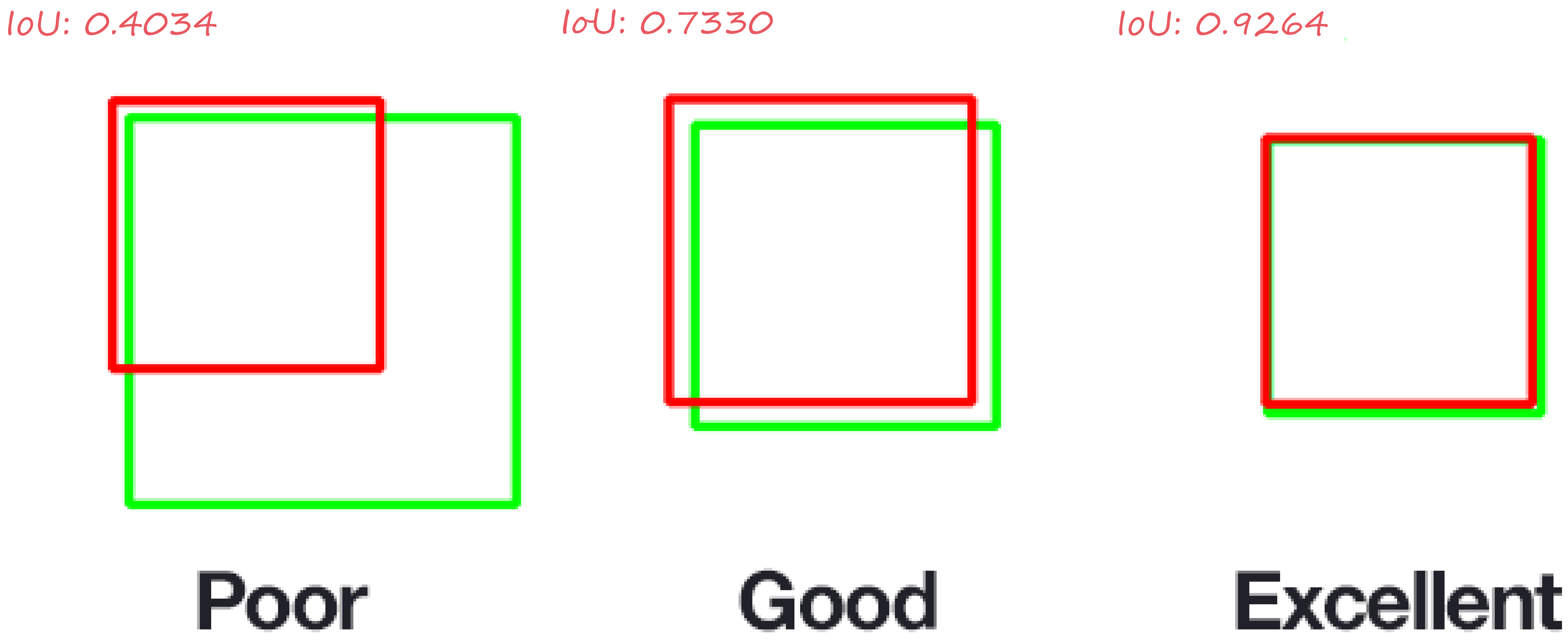
$$\text{Score} = \frac{\text{Area of overlap}}{\text{Area of union}}$$



When the score exceeds a threshold (e.g. 0.5), and the object is correctly identified, a true positive is declared

https://www.aiukraine.com/wp-content/uploads/2016/09/Pashchenko_AIUkraine_2016.pdf

Intersection over Union (2/2)



https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

mAP and IoU

- **The “goodness” of object recognition engines is evaluated using the following criteria**
  - Was the *position* of the objects correctly identified => IoU
  - Were the objects correctly identified => mAP
- **When only the mAP value is given, the IoU is implicit in the “rulebook”**
  - The PASCAL VOC challenge recognizes a true positive when IoU > 0.5
- **Engines using the COCO dataset usually show the IoU value used in the mAP calculation (e.g. mAP_{IoU}, mAP@IoU)**

XILINX

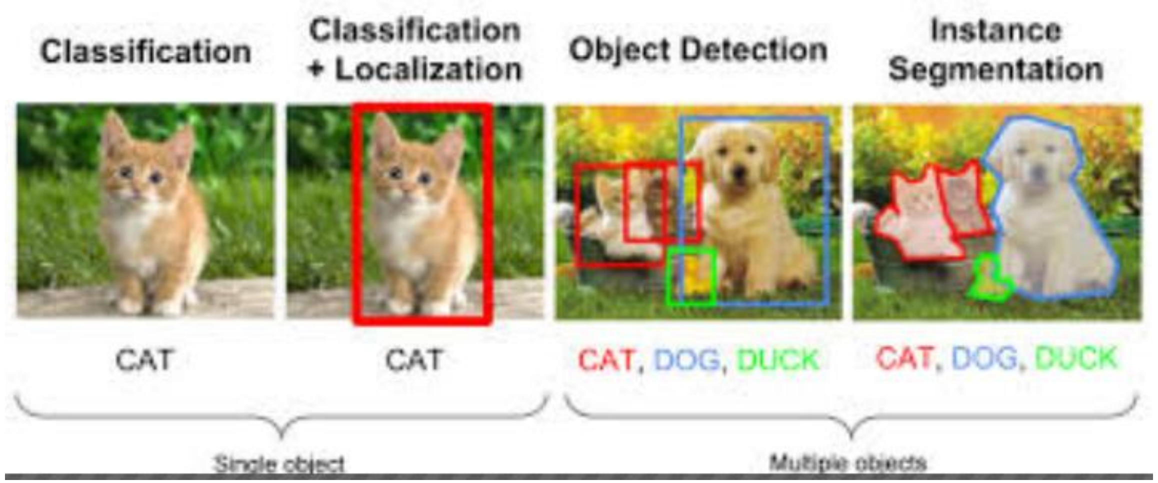
Detection Background Overview

Detection Background Information (1)

> **A great overview to start with for Detection networks (which doesn't include SSD) is available here: https://youtu.be/2xtx-gk3PqY**
>> The following background information is a high level summary and the reference above does a much better job giving a detailed overview of the task at hand

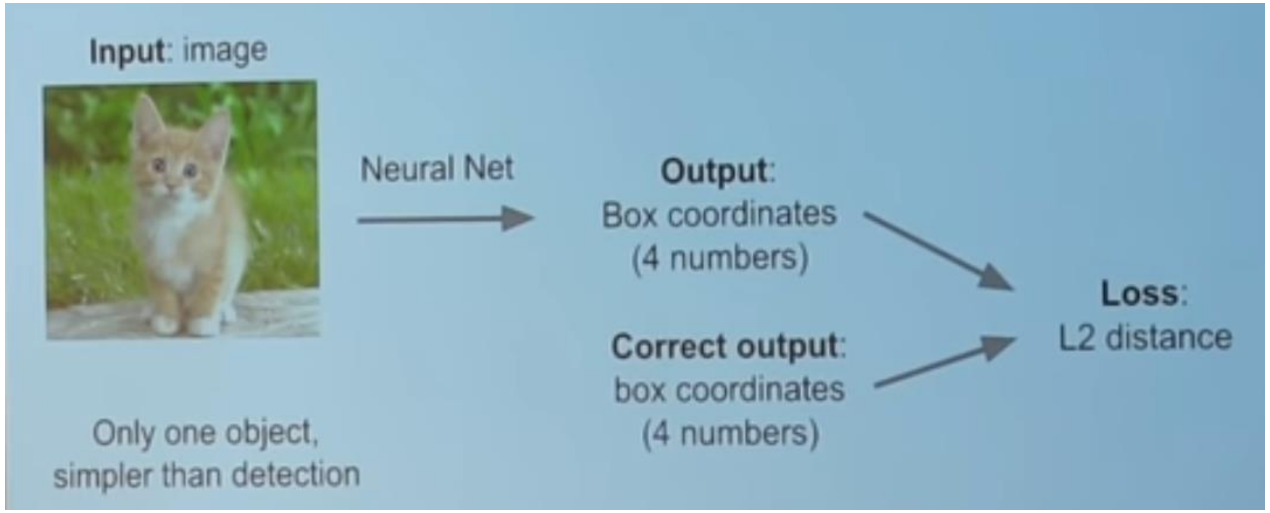
> **Assumption: we have already reviewed classification which produces a class output based on an input image**
– The next step toward detection is localization

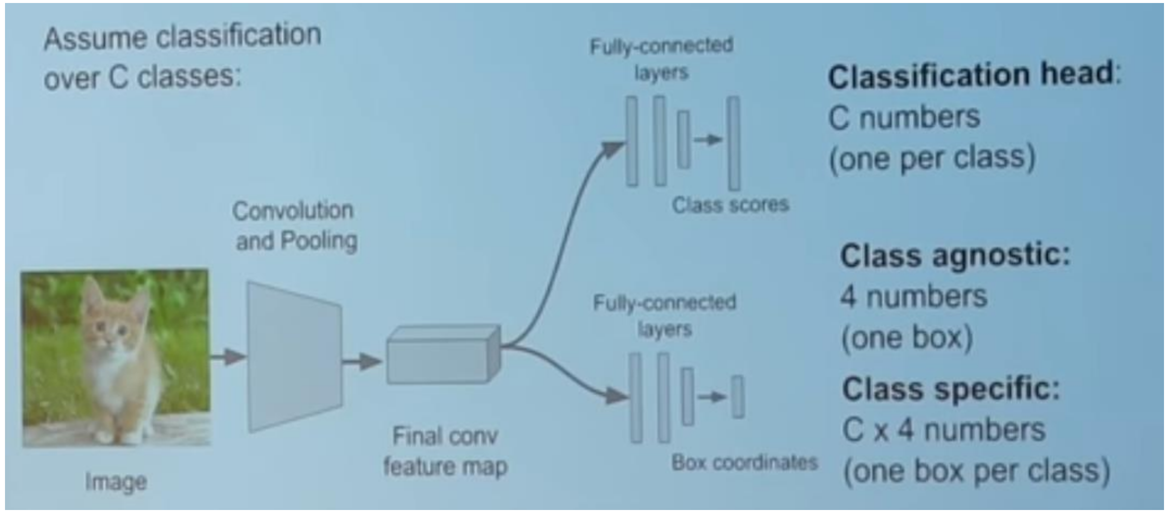
> **Localization not only classifies the image but locates the object's instance within that image**



Detection Background Information (2)

- **Localization alone can be modeled as a regression problem which outputs (and can be trained on) a series of values (i.e. box coordinates) rather than only the distinct class**
- **In addition to using the fully connected layers which are used in classification networks, a regression head is also used which will produce the box coordinates**
  - The image comes into the network and produces the class numbers as well as the box coordinates which come from the regression head
  - The correct output is also a 4 number value which can be trained against
    - The loss is then computed to train the network against these values
  - The Regression head can be attached after the convolution layers, or could also be attached after the last FC layer





Detection Background Information (3)

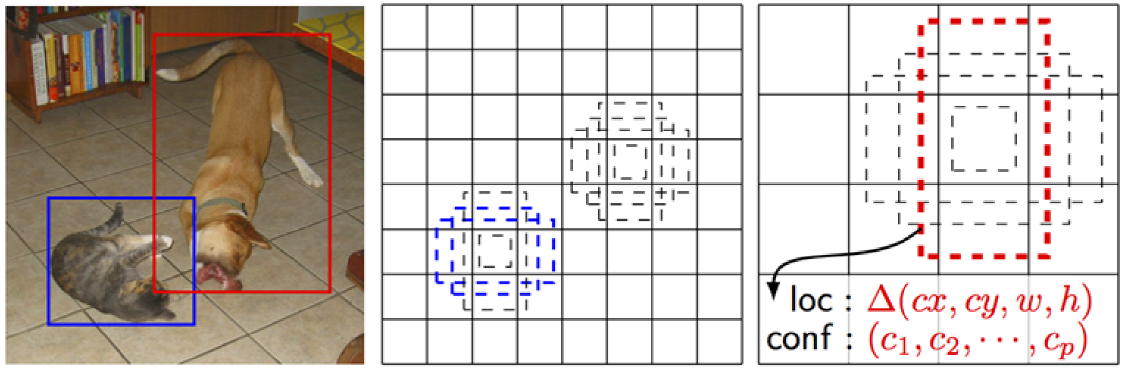
- **The localization technique can be extended for localizing more than one object**
  - Sliding window approaches can be used
    - Use a sliding window to identify classes/bounding boxes for sub-regions of an image, then combine them into a higher level score
- **HOWEVER – for a detection problem, the outputs are variable**
  - need to detect an unknown number objects of different classes, varying scales, and potential overlapping regions in a single image
- **One initial approach to address this issue was to model the detection problem as classification rather than regression**
  - Problem – need to test many different positions and scales
    - OLD Solution – if your classifier is fast enough you can just do it multiple times with sliding windows
    - R-CNN type Solution – region proposals could be used to act as a class agnostic object detector to look for blob-like regions
      - These types of networks are called “Region Proposal Networks”
      - This is complicated for training and slower – improvements were made in Fast R-CNN, and Faster R-CNN, but still lacks real time performance
        - Part of the reason is that two completely separate networks are needed – one for generating the region proposals, and one for the classification

Detection Background Information (4)

- **Even after many improvements to Region Proposal Networks (Fast R-CNN, and Faster R-CNN) the performance of these is still not “real time”**
  - ~5-7 FPS for Faster R-CNN on an x86+GPU machine (results are good, however in terms of mAP%)
- **YOLO then introduced the idea of solving the detection problem as a regression problem**
  - This eliminates the need for the region proposal stage and greatly decreases processing time
  - This is done by discretizing the output space of bounding boxes into a set of default boxes over different aspect rations and scales per feature map location
- **YOLO uses a specific set of convolutional feature layers (i.e. backbone) called DarkNet**
  - Also trained in the darknet framework
  - SSD is similar to YOLO in that it is a single shot detector, but differs in a couple major ways
    - SSD can use feature extraction layers from many various classification networks (VGG, Inception, ResNet, etc.)
      - Feature layers extract edges, shapes, etc. of interest from the input data
    - SSD also uses a different strategy for the regression head which identifies the bounding box coordinates

SSD Introduction

SSD is an unified framework for object detection with a single network. You can use the code to train/evaluate a network for object detection task. For more details, please refer to our arXiv paper and our slide.



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

System	VOC2007 test *mAP*	FPS (Titan X)	Number of Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	~6000	~1000 x 600
YOLO (customized)	63.4	45	98	448 x 448
SSD300* (VGG16)	77.2	46	8732	300 x 300
SSD512* (VGG16)	**79.8**	19	24564	512 x 512

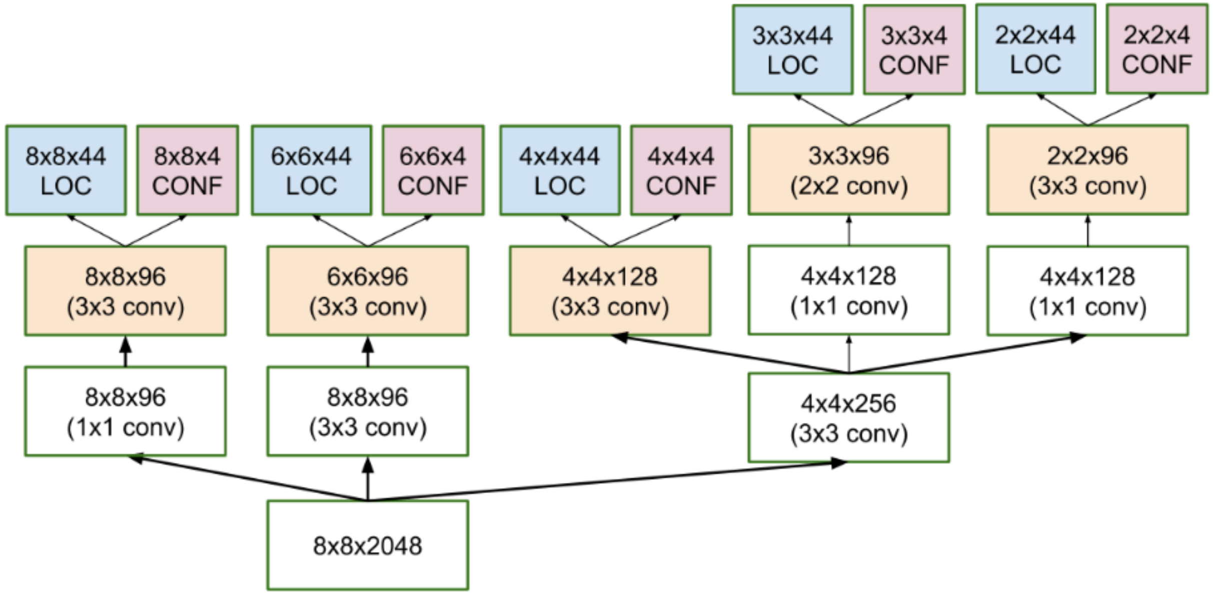
Method	VOC2007 test		VOC2012 test		COCO test-dev2015		
	07+12	07+12+COCO	07++12	07++12+COCO	trainval35k		
	0.5	0.5	0.5	0.5	0.5:0.95	0.5	0.75
SSD300*	77.2	81.2	75.8	79.3	25.1	43.1	25.8
SSD512*	**79.8**	**83.2**	**78.5**	**82.2**	**28.8**	**48.5**	**30.3**

Note: SSD300 and SSD512* are the latest models. Current code should reproduce these results.*

Citation: https://github.com/weiliu89/caffe/tree/ssd

MultiBox Background Information (1)

- The Single Shot Detector (SSD) is based on the MultiBox work done by Szegedy
- In MultiBox, the fundamental idea was to train a CNN that outputs coordinates of object boxes correctly and rank the proposals by their likelihood of being an accurate box for an object of interest
- The primary outputs of the MultiBox Network are:
  - Confidence (CONF): logistic loss on the estimates of proposal corresponding to an object of interest
  - Location (LOC): loss corresponding to some similarity measurement between objects and closest matching object box prediction (uses L2 Norm)



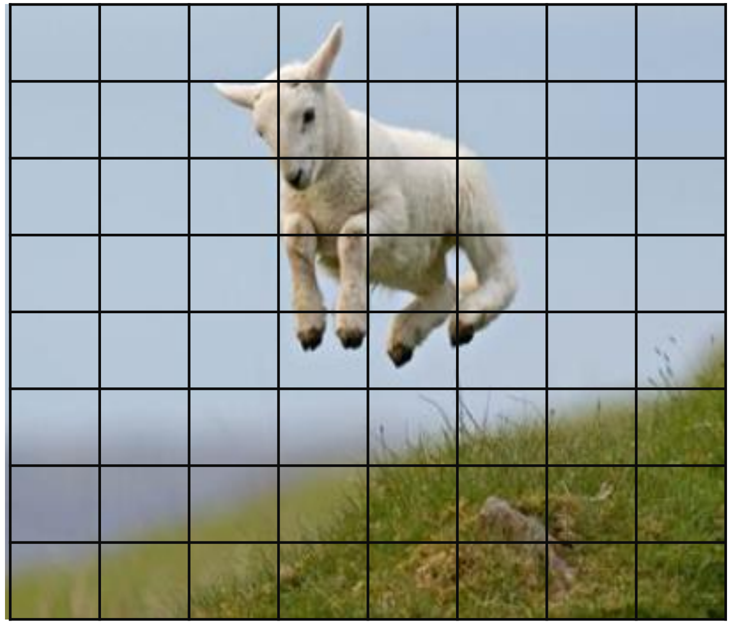
Architecture of multi-scale convolutional prediction of the location and confidences of multibox

Source: https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab

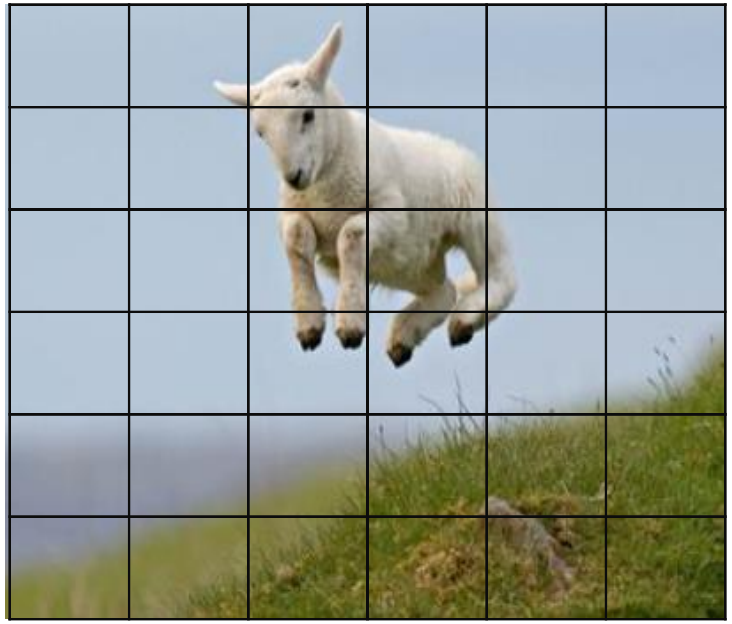
MultiBox Background Information (2)

- **MultiBox is the portion of the network used as a reference for the bounding box regressor**
  - MultiBox starts with priors (or anchors – as described in R-CNN) which are pre-computed, fixed size bounding boxes
  - The initial predictions will then start with these prior boxes and training will attempt to regress them closer to the true bounding box locations
  - The images are divided into sub regions including 8x8, 6x6, 4x4, 3x3, 2x2 and the 1x1 is used to predict the single largest prior
    - Each region contains 11 potential boxes of differing aspect ratios/sizes

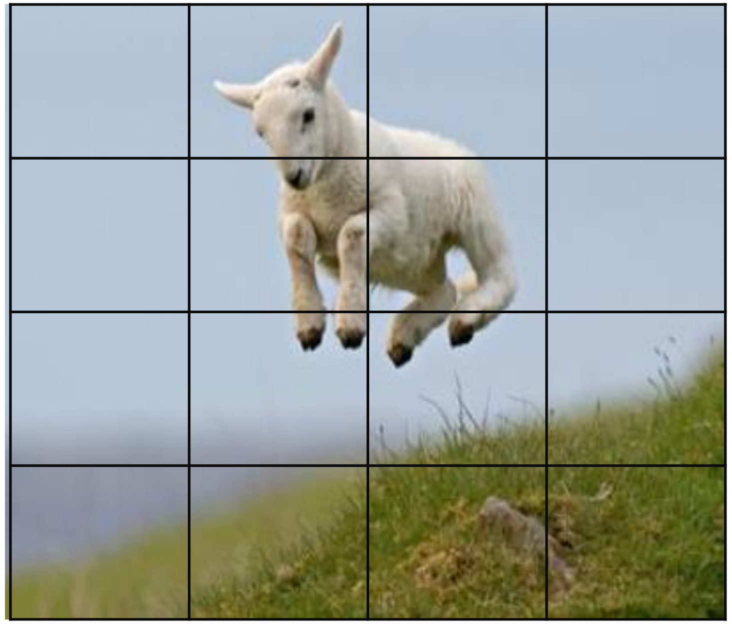
8x8 Grid



6x6 Grid

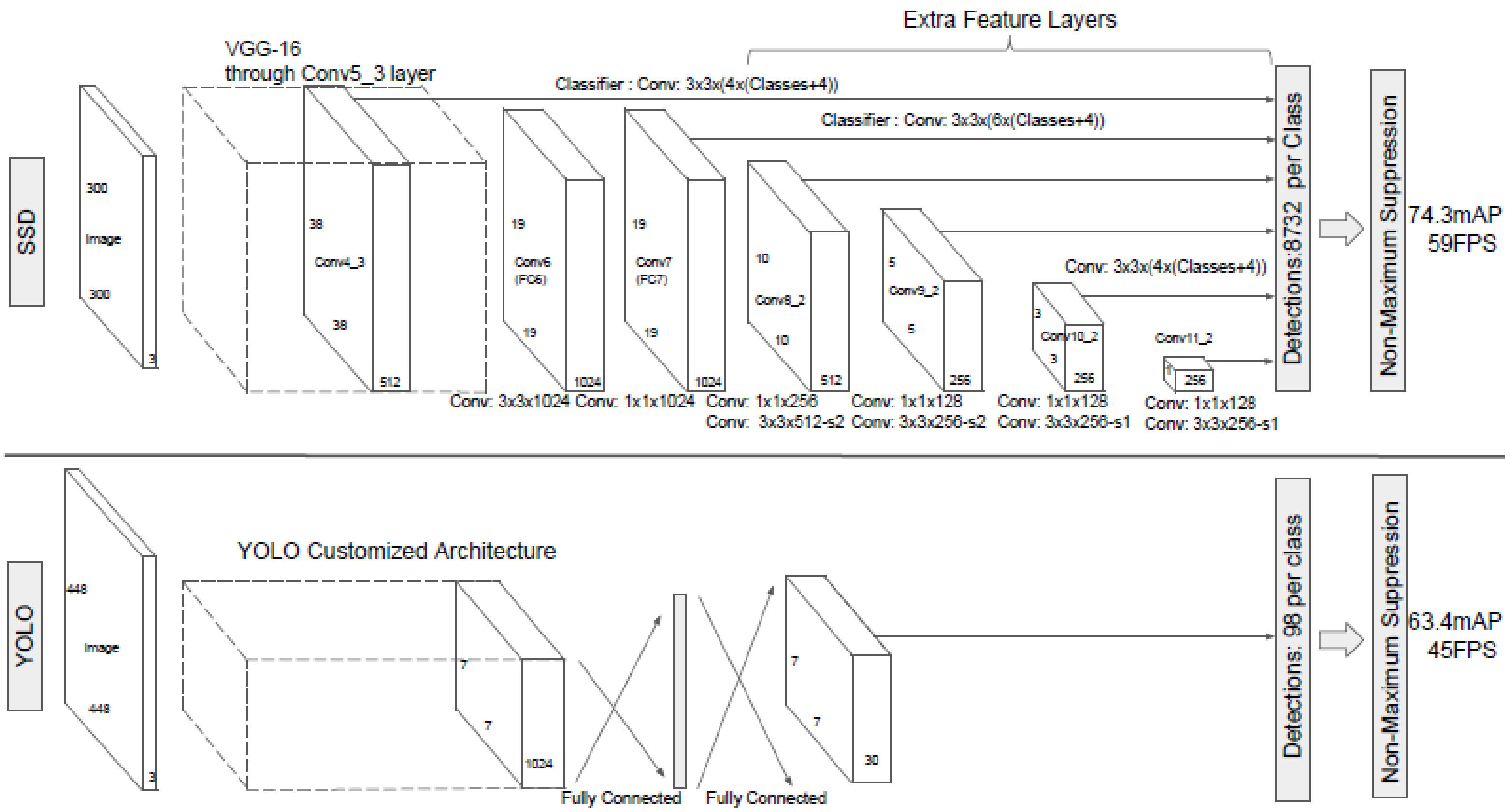


4x4 Grid



SSD Background Information

- SSD bounding box regressor is based on MultiBox and somewhat similar to YOLO
- One of the key differences is that SSD uses convolutional layers to provide different image scales (stride=2 for downsampling)

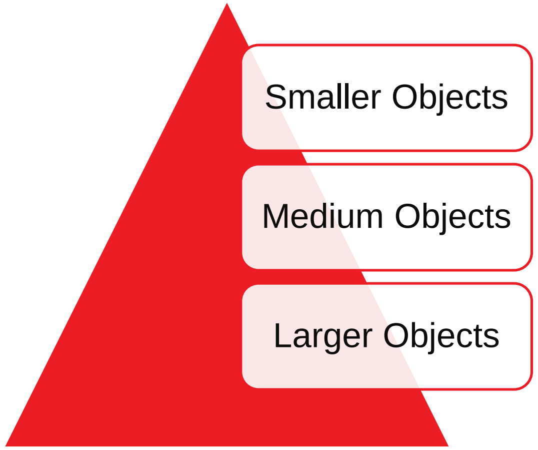


Citation: https://arxiv.org/abs/1512.02325

XILINX

Regression Outputs from SSD

- **Rather than implement the exact Prior Boxes etc. from MultiBox, SSD implements the following:**
  - 38x38 grid with 4 boxes per cell = 5776 boxes
  - 19x19 grid with 6 boxes per cell = 2166 boxes
  - 10x10 grid with 6 boxes per cell = 600 boxes
  - 5x5 grid with 6 boxes per cell = 150 boxes
  - 3x3 grid with 4 boxes per cell = 36 boxes
  - 1x1 grid with 4 boxes per cell = 4 boxes
  - TOTAL 8732 boxes

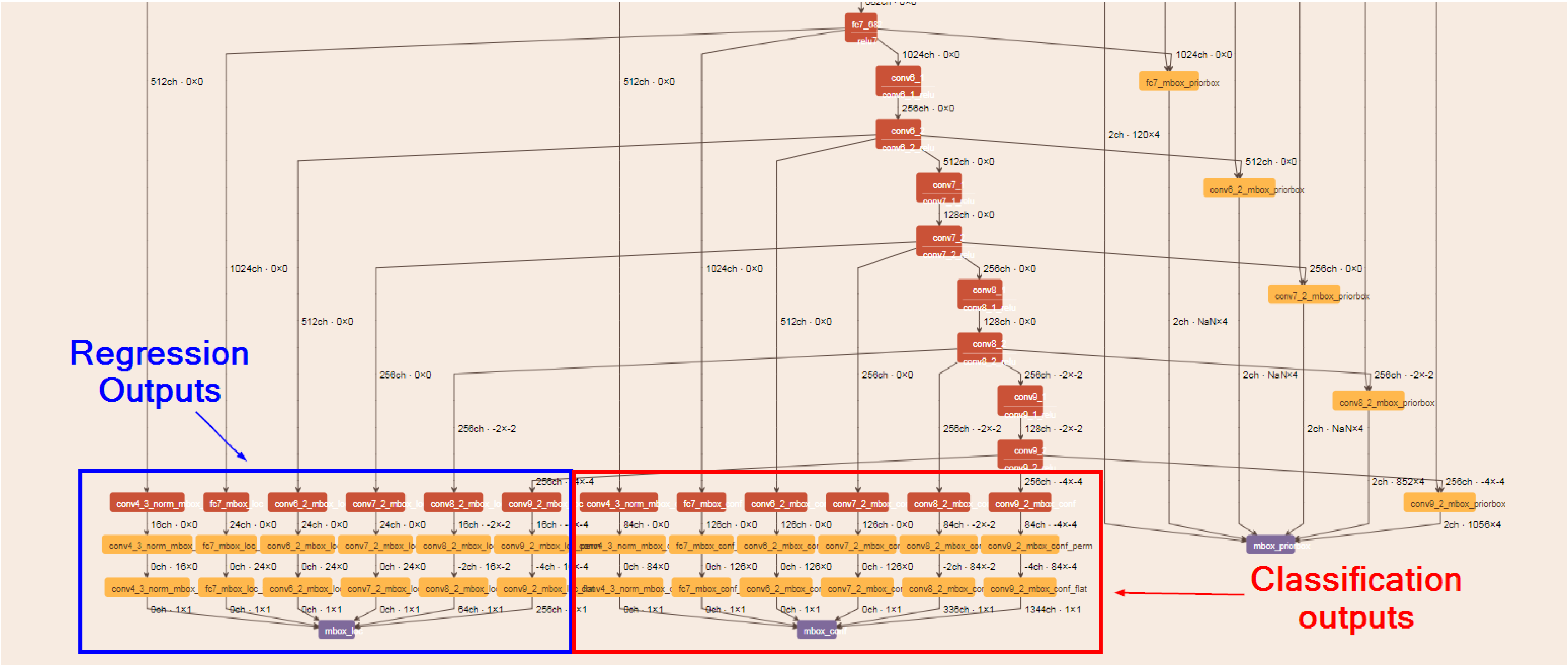


- **Convolution operations are used to create these outputs**
  - For Example:
    - 38x38 grid corresponds to a 38x38 feature map size which is the input to the convolution layer
      - 3x3 kernel size is used
    - Number of output channels (feature maps) from these layers depends on the number of boxes desired for that grid size
      - For 6 boxes, 24 output channels would be needed (4 values for each box)
      - For 4 boxes, 16 output channels would be needed (4 values for each box)

Classification Outputs from SSD

- For the classification outputs each of the grid sizes also will have a classification result
- The number of output channels depends on the number of boxes per cell, grid size, and number of output classes
- For PASCAL VOC, there are 21 output classes (20 objects and 1 background)
  - 38x38 grid -> 4 boxes/cell * 21 classes = 84 output channels @ 38x38
  - 19x19 grid -> 6 boxes/cell * 21 classes = 126 output channels @19x19
  - 10x10 grid -> 6 boxes/cell * 21 classes = 126 output channels @ 10x10
  - 5x5 grid -> 6 boxes/cell * 21 classes = 126 output channels @ 5x5
  - 3x3 grid -> 4 boxes/cell * 21 classes = 84 output channels @ 3x3
  - 1x1 grid -> 4 boxes/cell * 21 classes = 84 output channels @ 1x1
- The output feature map size then relates back to the grid location and the channel relates to the class and specific priorbox for that class
- These are finally run through softmax function

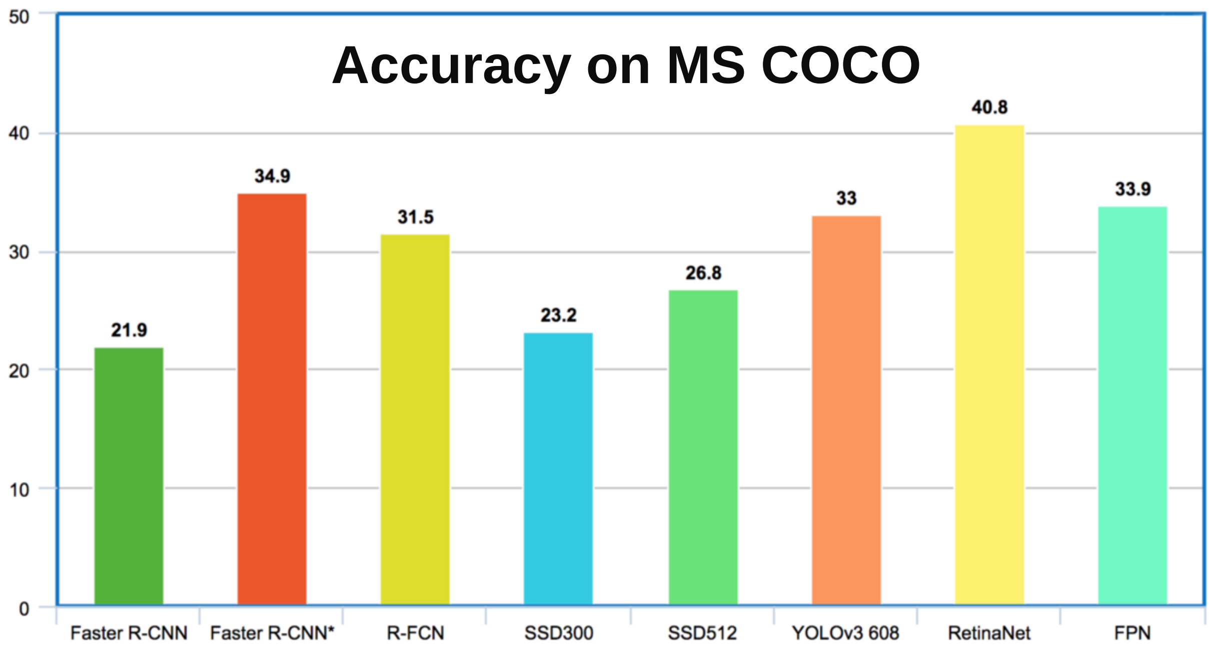
SSD Classification/Regression Subset Netscope Diagram

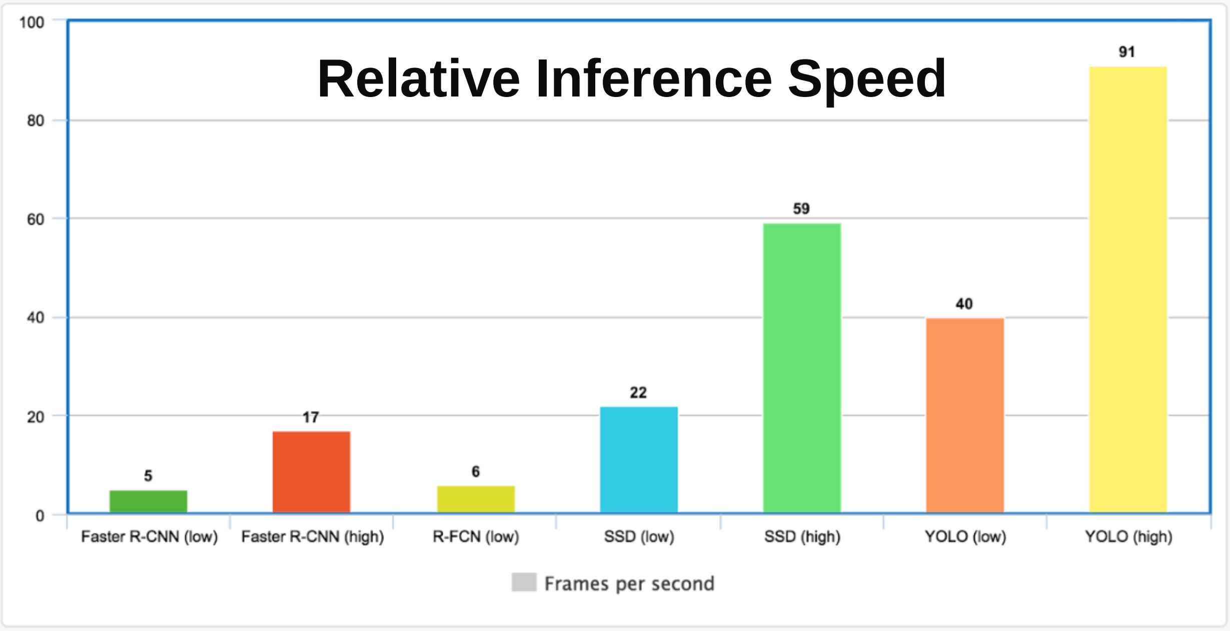


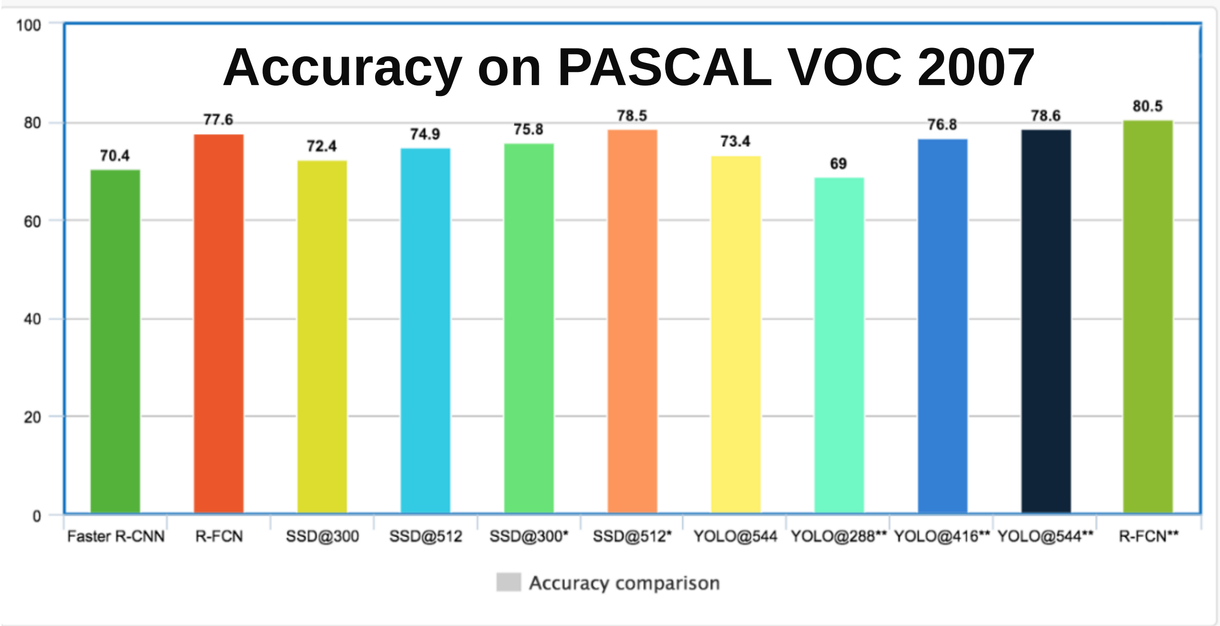
Detection Network Comparison

Comparison of Detection Network Accuracy and Speed

> **Warning** – these numbers shouldn't be taken as hard and fast comparison point as the experiments are done in different settings

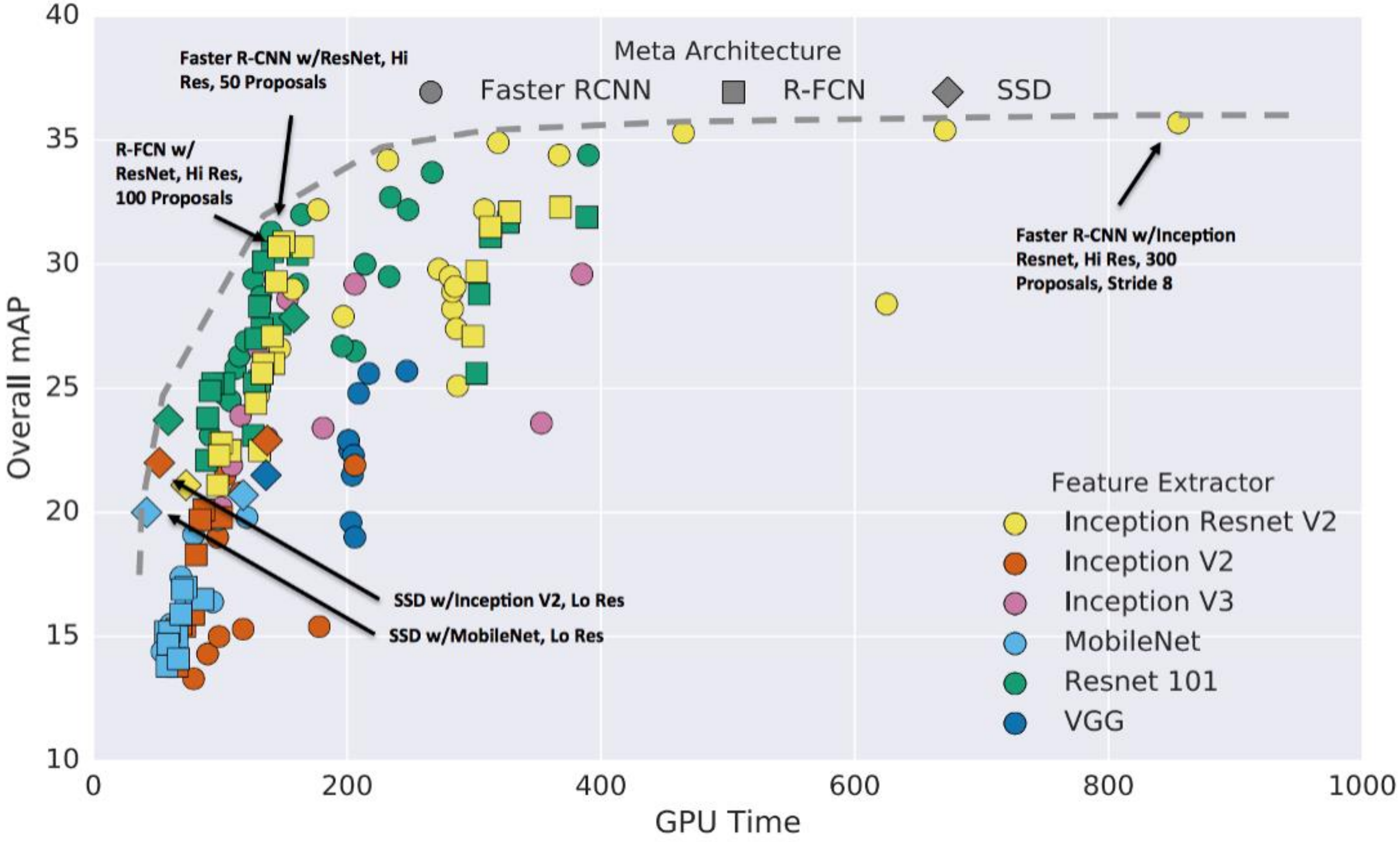






Source: https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359

Performance Variations for Detection Framework and Feature Extractor



Source: *https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359*

Detection Networks Comparisons

	SSD (300x300)	SSD (512x512)	YOLOv1	YOLOv2	Faster R-CNN
mAP % (VOC)	72.4	74.9	57.9	73.4	70.4
FPS (K40 GPU batch=1)	46	19	45	67	7
#Boxes	8732	24564	98		~6000
#Classes	20	20	20	Can be modified for Up to 9000	
Input Resolution	300x300	512x512	448x448	416x416	~1000x600

	SSD321	SSD513	YOLOv2	YOLOv3	R-FCN	DSSD321	DSSD513
mAP% (50 IoU - COCO)	45.4	50.4	44.0	55.3	51.9	46.1	53.3
FPS	16.3	8	45	34.5	11.7	11.7	6.4

Source: https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359

XILINX

Detection Networks Summary

- RCNN/Fast R-CNN/Faster R-CNN is generally the highest overall for detection accuracies, but much slower in terms of performance
- SSD is particularly good at detecting large objects, but not as good with smaller objects or objects that are close together
  - Different variants of SSD perform this task better than others
- YOLO has variants that can detect up to 9000 object classes and YOLOv3 claims to be faster and more accurate than SSD (of course likely feature layer dependent)
  - YOLO encapsulates its own feature extraction layers (Darknet) whereas SSD and Faster R-CNN etc. are designed to work with other feature extractors such as VGG, Resnet, etc.
  - YOLOv3 makes improvements to YOLOv2 specifically for smaller objects

Video Comparison of YOLOv2, YOLO9000, SSD, and Faster R-CNN

Yolov3 Video Demo

Source: https://medium.com/@jonathan_hui/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359

Adaptable.

Intelligent.