Practical Object Detection and Segmentation

Vincent Chen and Edward Chou

Agenda

- Why would understanding different architectures be useful?
- Modular Frameworks
- Describe Modern Frameworks
  - Detection
  - Segmentation
  - Trade-offs
  - Open Source Links
- Using Detection for Downstream Tasks

Why do I need this?

- SoTA Object Detectors are really good!
  - Used in consumer products
- Understanding trade-offs: *when should I use each framework?*
- Object detection/segmentation is a first step to many interesting problems!
  - While not perfect, you can assume you have bounding boxes for your visual tasks!
  - Examples: scene graph prediction, dense captioning, medical imaging features

Modular Frameworks

- Base network
  - Feature extraction
- Proposal Generation
  - Sliding windows, RoI, Use a network?

Modern Convolutional Detection/Segmentation

Detection

- R-FCN
- Faster R-CNN
- YOLO
- SSD

Segmentation

- Mask R-CNN
- SegNet
- U-Net, DeepLab, and more!

Modern Convolutional Object Detectors

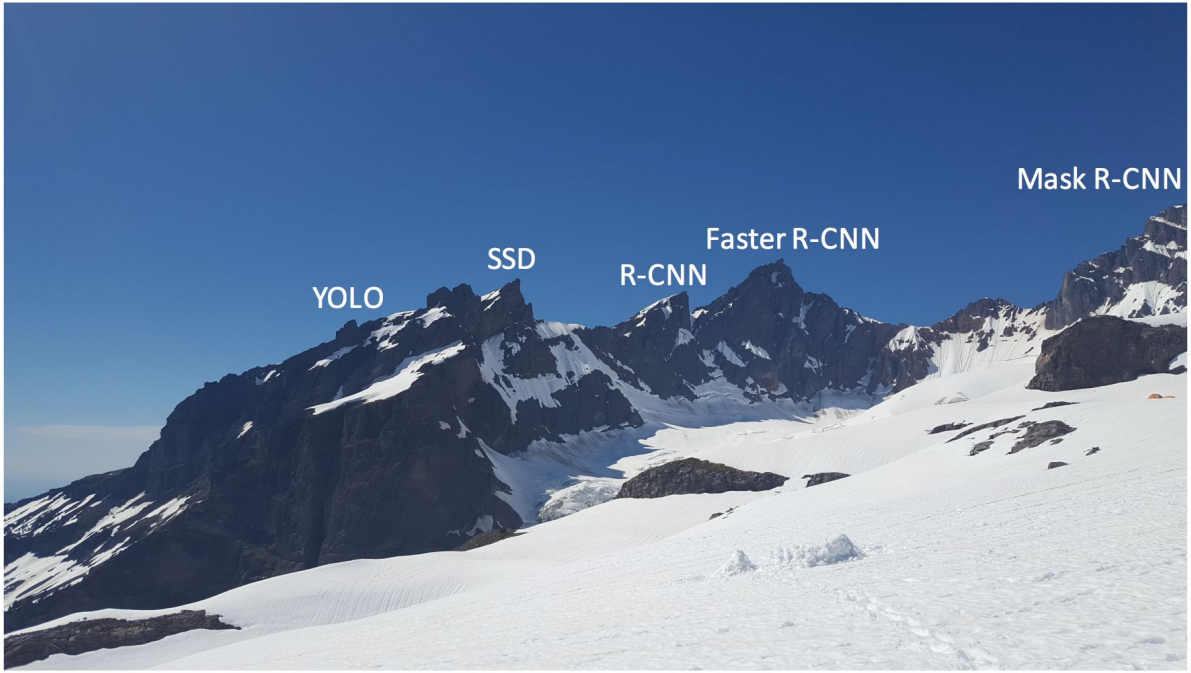


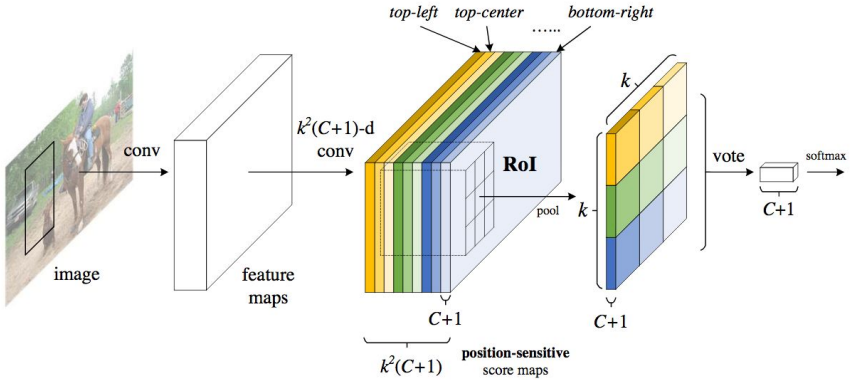
Image from: http://deeplearning.csail.mit.edu/instance_ross.pdf

Faster-R CNN

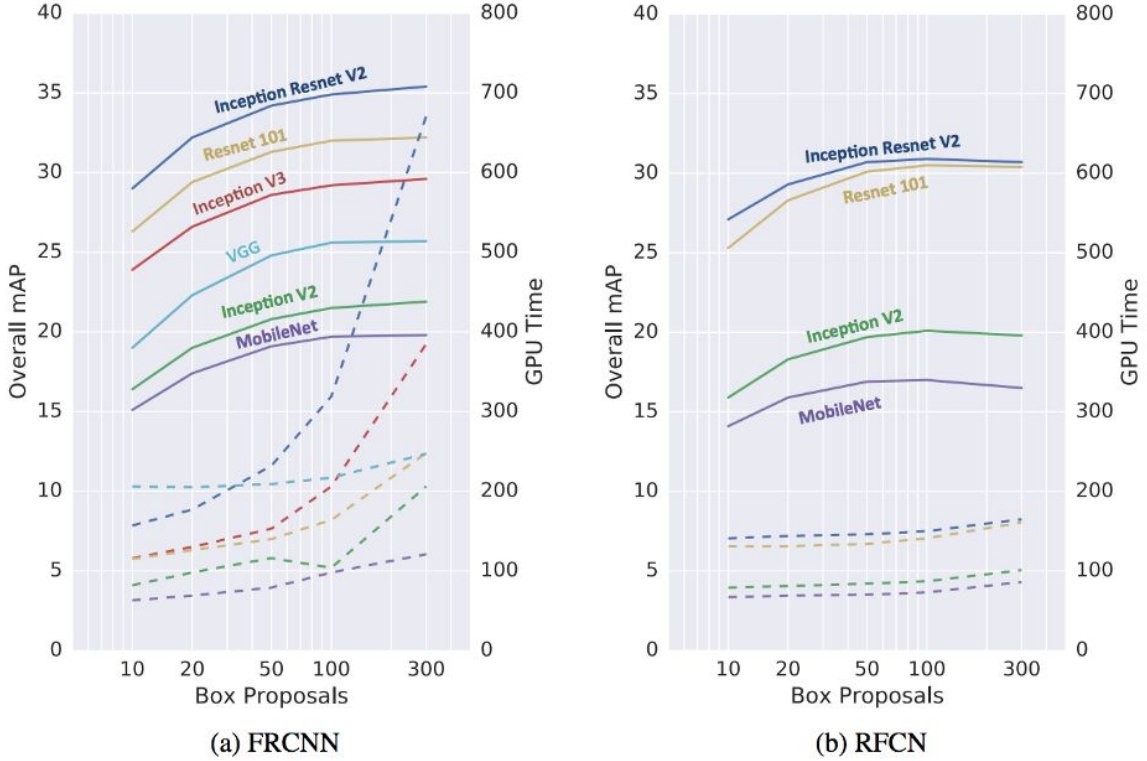
- History
  - R-CNN: Selective search → Cropped Image → CNN
  - Fast R-CNN: Selective search → Crop feature map of CNN
  - Faster R-CNN: CNN → Region-Proposal Network → Crop feature map of CNN
- Proposal Generator → Box classifier
- Best performance, but longest run-time
- End-to-end, multi-task loss
- Can use fewer proposals, but running time is dependent on proposals
- https://github.com/endernewton/tf-faster-rcnn

R-FCN

- Addresses translation-variance in detection
  - Position-sensitive ROI-pooling
- Good balance between speed & performance
  - 2.5 - 20x faster than Faster R-CNN
- https://github.com/daijifeng001/R-FCN



Tradeoff: Number of Proposals

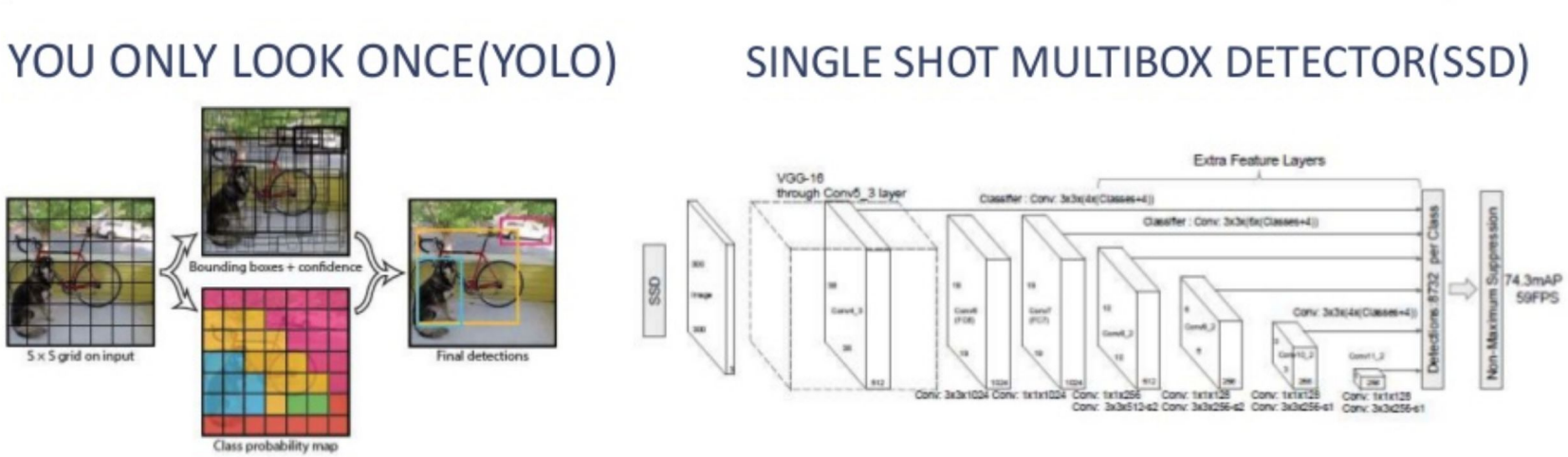


(a) FRCNN (b) RFCN

Image from: https://arxiv.org/pdf/1611.10012.pdf

Detection without proposals: YOLO/SSD

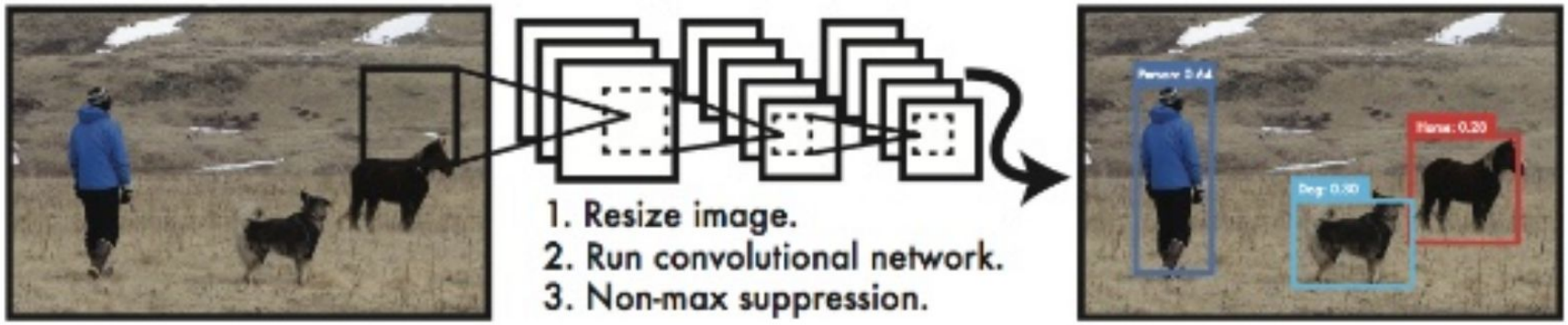
- Several techniques pose detection as a regression problem (a.k.a single shot detectors)
- Two of the most popular ones: YOLO/SSD



Images from: https://www.slideshare.net/TaegyunJeon1/pr12-you-only-look-once-yolo-unified-realtime-object-detection

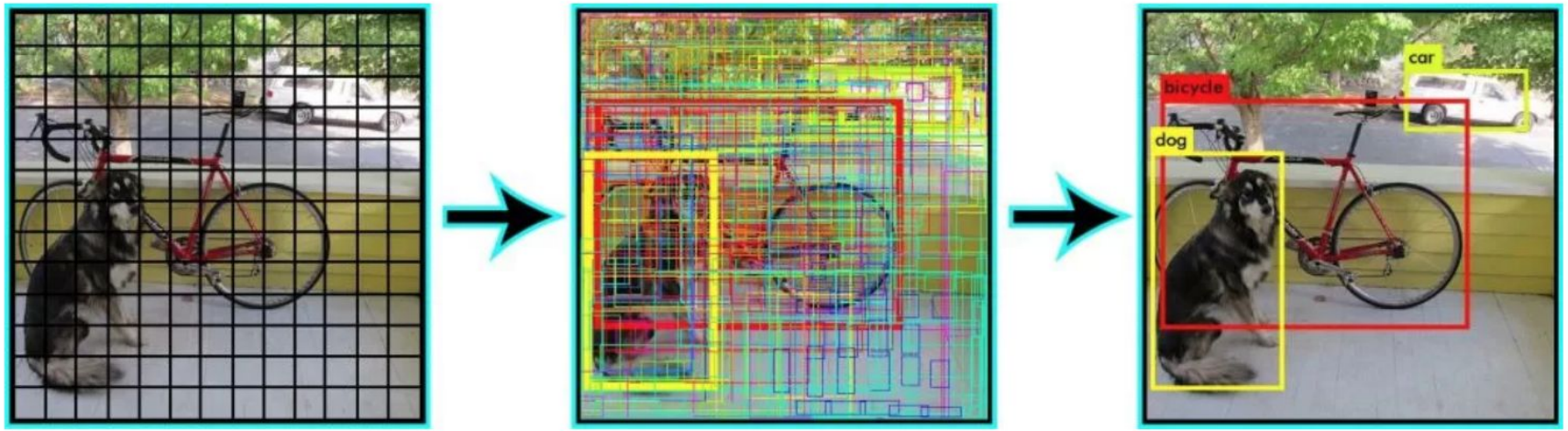
YOLO

- Super fast (21~155 fps)
- Finds objects in image grids at parallel
- Only slightly worse performance than Faster R-CNN



Images from: https://www.slideshare.net/TaegyunJeon1/pr12-you-only-look-once-yolo-unified-realtime-object-detection

YOLO



Images from: https://www.slideshare.net/TaegyunJeon1/pr12-you-only-look-once-yolo-unified-realtime-object-detection

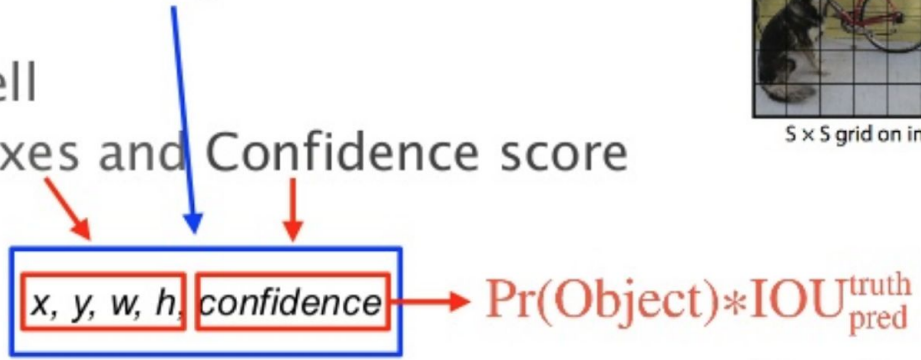
Unified Detection

- All BBox, All classes

1) Image → **S x S** grids

2) grid cell

→ **B**: BBoxes and Confidence score



→ **C**: class probabilities w.r.t #classes

$Pr(Class_i|Object)$

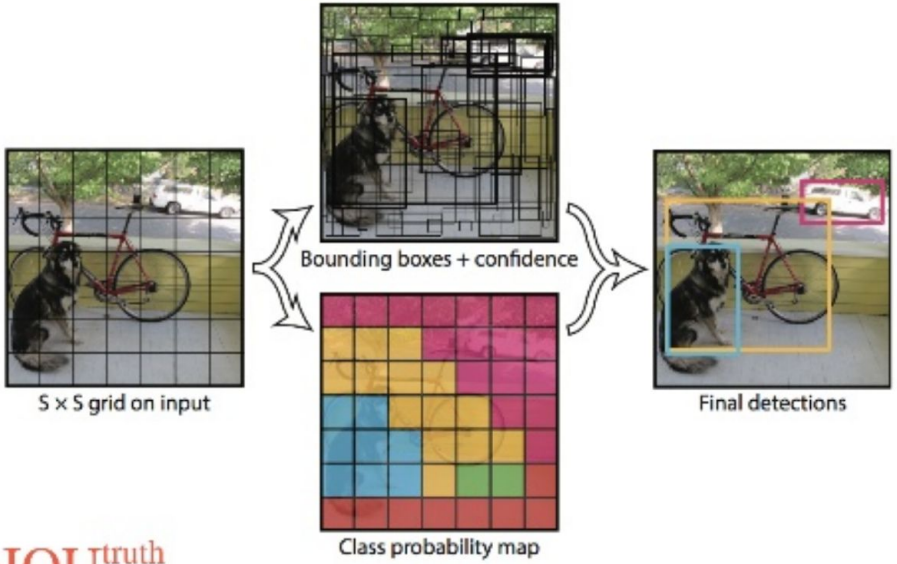


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

Slide from: https://www.slideshare.net/TaegyunJeon1/pr12-you-only-look-once-yolo-unified-realtime-object-detection

Limitations of YOLO

- Groups of small objects
- Unusual aspect ratios - struggles to generalize
- Coarse Features (Due to multiple pooling layers from input images)
- Localization error of bounding boxes - treats error the same for small vs large boxes

YOLO vs YOLO v2

- YOLO: Uses InceptionNet architecture
- YOLOv2: Custom architecture - Darknet

	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**

Table from *YOLO9000: Better, Faster, Stronger* (https://arxiv.org/abs/1612.08242)

YOLO Versions

YOLO (darknet) - https://pjreddie.com/darknet/yolov1/ (C++)

YOLO v2 (darknet) - https://pjreddie.com/darknet/yolov2/ (C++)

- Better and faster - 91 fps for 288 x 288

YOLO v3 (darknet) - https://pjreddie.com/darknet/yolo/ (C++)

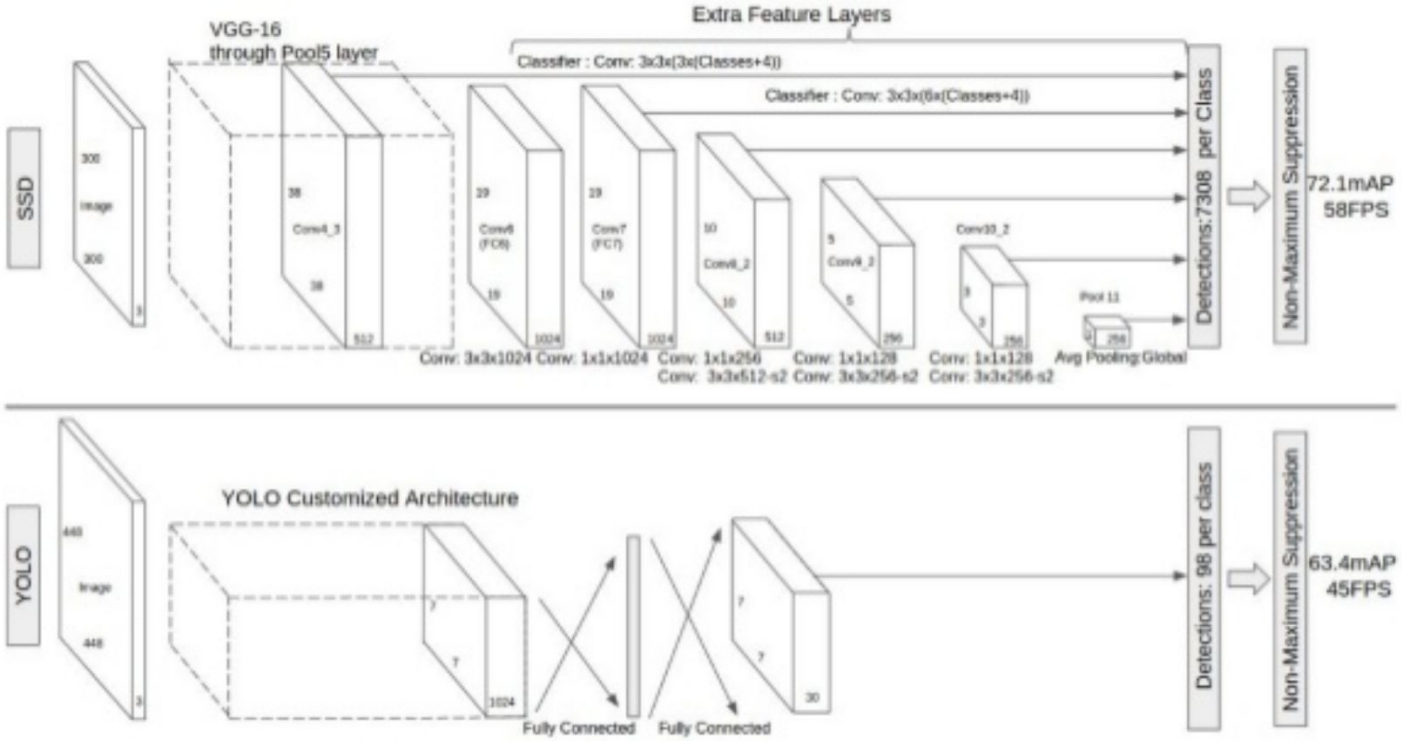
YOLO (caffe) - https://github.com/xingwangsfu/caffe-yolo

YOLO (tensorflow) - https://github.com/thtrieu/darkflow

SSD

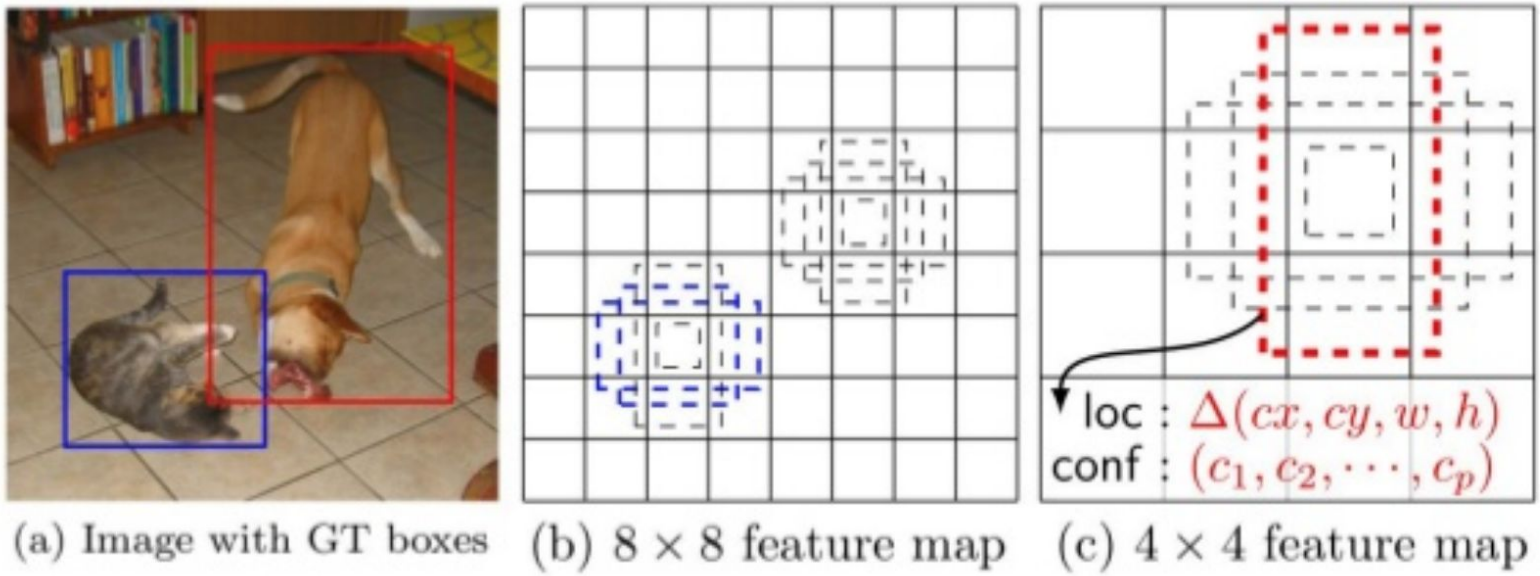
- End-to-end training (like YOLO)
  - Predicts category scores for fixed set of default bounding boxes using small convolutional filters (different from YOLO!) applied to feature maps
  - Predictions from different feature maps of different scales (different from YOLO!), separate predictors for different aspect ratio (different from YOLO!)

SSD vs YOLO



Images from: https://www.slideshare.net/xavigiro/ssd-single-shot-multibox-detector

SSD Visualization



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

Images from: https://www.slideshare.net/xavigiro/ssd-single-shot-multibox-detector

SSD Limitations

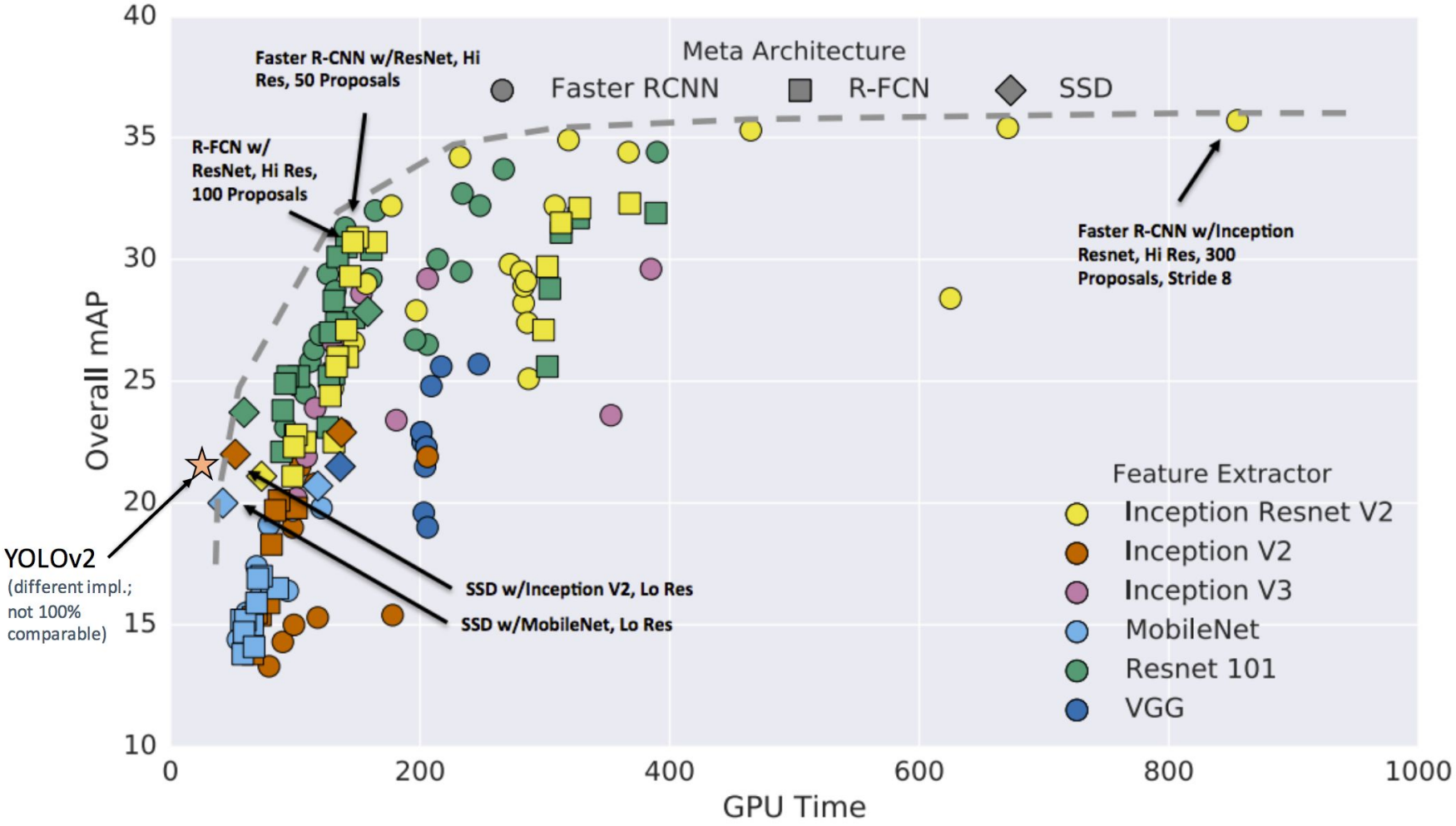
- For training, requires that ground truth data is assigned to specific outputs in the fixed set of detector outputs
- Slower but more accurate than YOLO
- Faster but less accurate than Faster R-CNN

SSD Versions

SSD (caffe) - https://github.com/weiliu89/caffe/tree/ssd

SSD (tensorflow) - https://github.com/balancap/SSD-Tensorflow

SSD (pytorch) - https://github.com/amdegroot/ssd.pytorch



Slide from Ross Girshick's CVPR 2017 Tutorial, Original Figure from Huang et al

Object Size Performance Comparisons

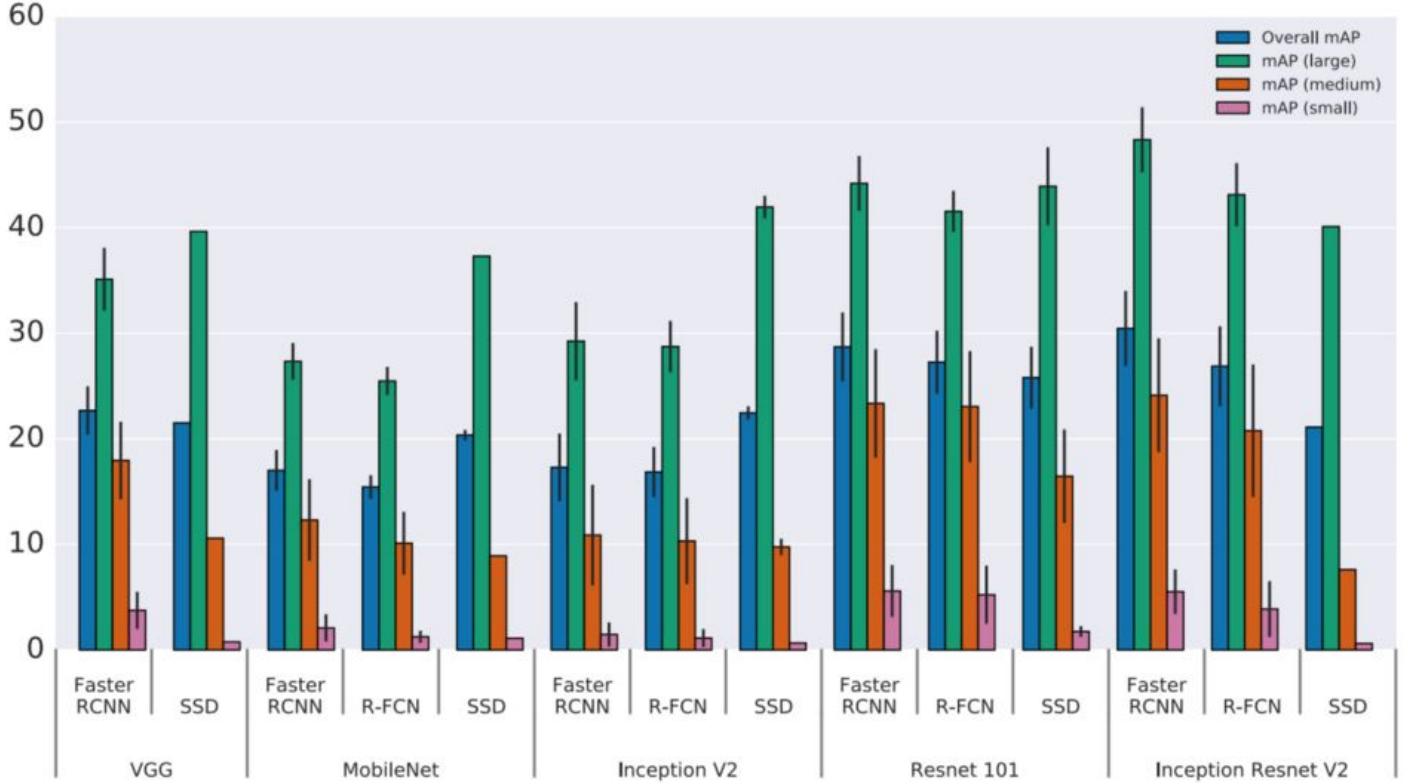


Image from: https://arxiv.org/pdf/1611.10012.pdf

Semantic/Instance-level Segmentation

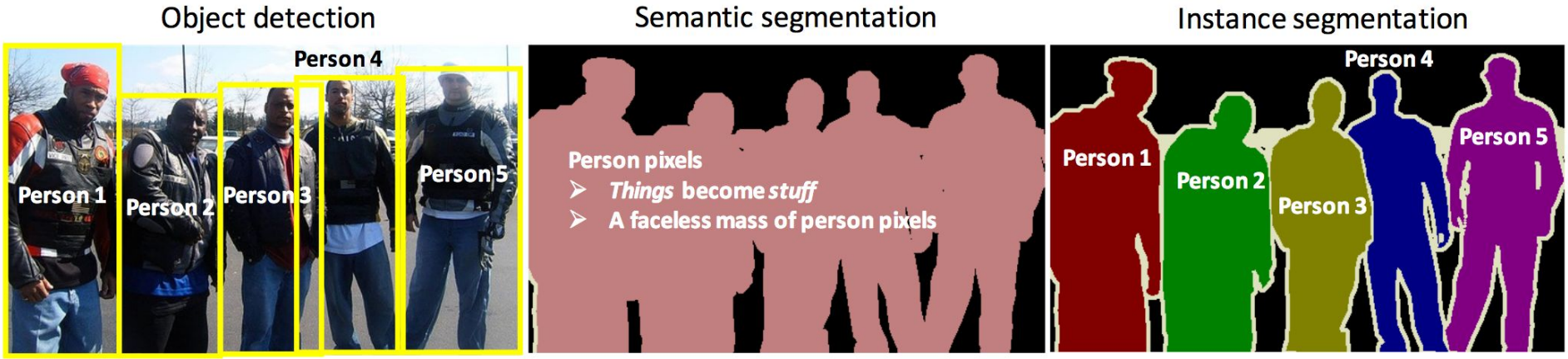
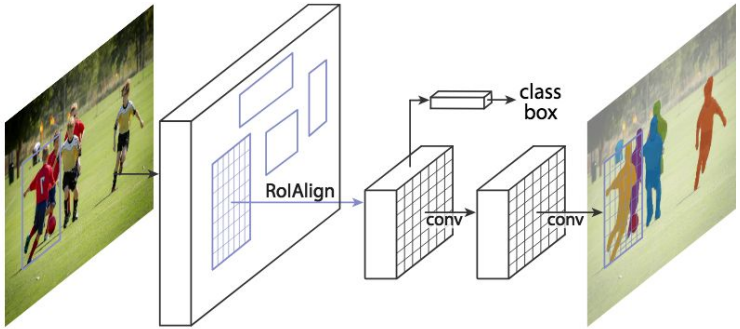


Image from PASCAL VOC

Mask R-CNN



From He et. al 2017

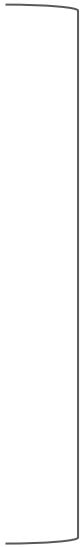
Mask R-CNN

1. Backbone Architecture
2. Scale Invariance (e.g. Feature Pyramid Network (FPN))
3. Region Proposal Network (RPN)
4. Region of interest feature alignment (RoIAlign)
5. Multi-task network head
    a. Box classifier
    b. Box regressor
    c. Mask predictor
    d. Keypoint predictor

Slide from Ross Girshick's CVPR 2017 Tutorial

Mask R-CNN

1. Backbone Architecture
2. Scale Invariance (e.g. Feature Pyramid Network (FPN))
3. Region Proposal Network (RPN)
4. Region of interest feature alignment (RoIAlign)
5. Multi-task network head
    a. Box classifier
    b. Box regressor
    c. Mask predictor
    d. Keypoint predictor



modular!

Slide from Ross Girshick's CVPR 2017 Tutorial

Seg-Net

Encoder-Decoder framework

Use dilated convolutions, a convolutional layer for dense predictions.

Propose ‘context module’ which uses dilated convolutions for multi scale aggregation.

Uses a novel technique to upsample encoder output which involves storing the max-pooling indices used in pooling layer. This gives reasonably good performance and is space efficient (versus FCN)

Segnet Architecture

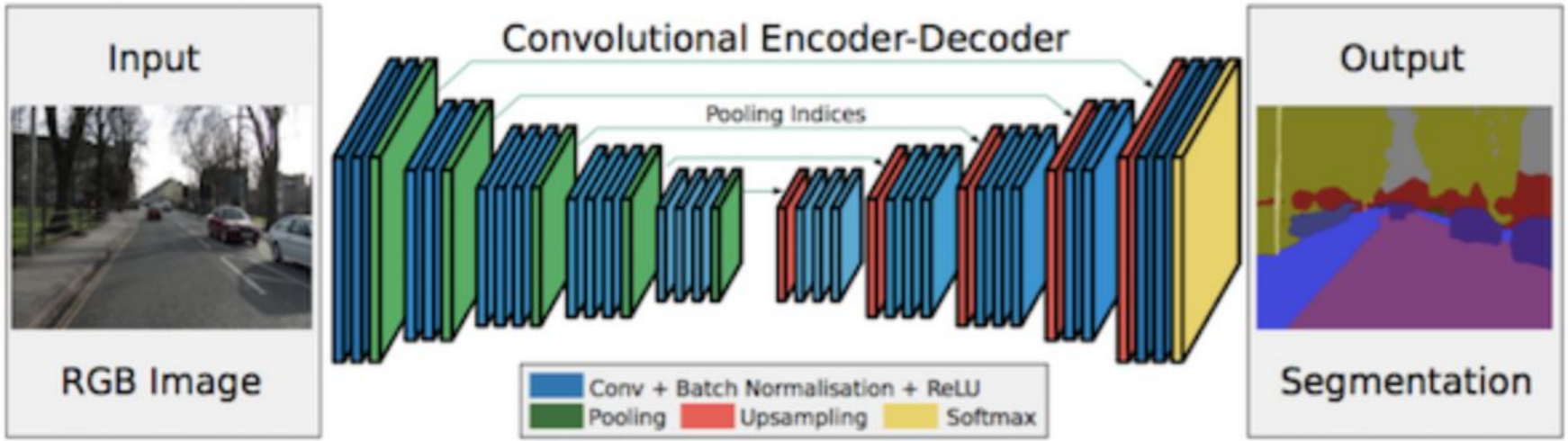


Image from: http://blog.qure.ai/notes/semantic-segmentation-deep-learning-review

Segnet Limitations

- Applications include autonomous driving, scene understanding, etc.
- Direct adoption of classification networks for pixel wise segmentation yields poor results mainly because max-pooling and subsampling reduce feature map resolution and hence output resolution is reduced.
- Even if extrapolated to original resolution, lossy image is generated.

Segnet Versions

Segnet (Caffe) - https://github.com/alexgkendall/caffe-segnet

Segnet (Tensorflow) - https://github.com/tkuanlun350/Tensorflow-SegNet

Segnet vs Mask R-CNN

Segnet

- Dilated convolutions are very expensive, even on modern GPUs.
-

Mask R-CNN

- Without tricks, Mask R-CNN outperforms all existing, single-model entries on every task, including the COCO 2016 challenge winners.
- Better for pose detection

Other Segmentation Frameworks

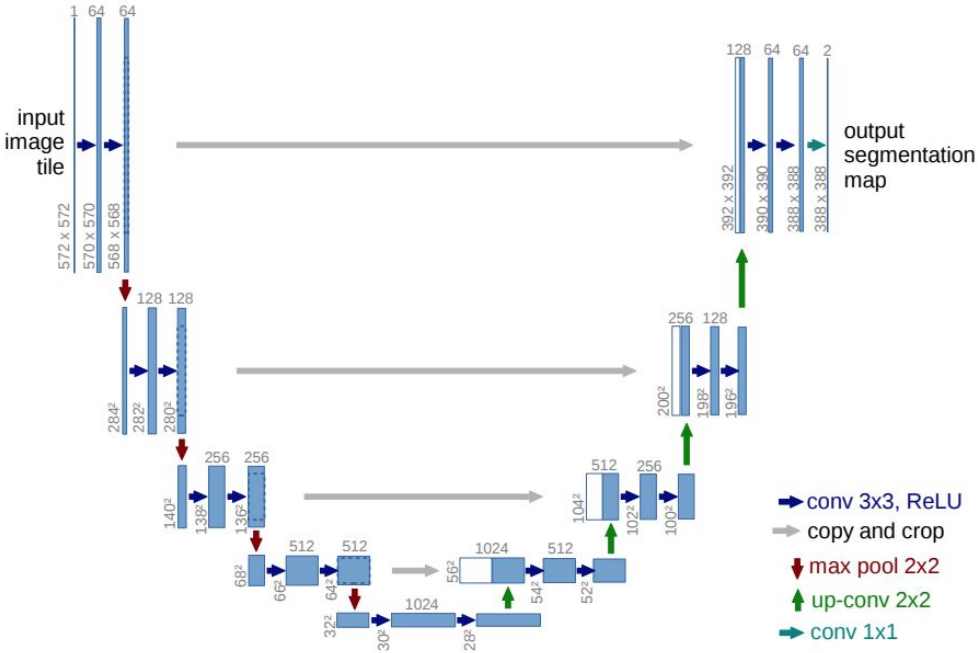
U-Net - Convolutional Networks for Biomedical Image Segmentation

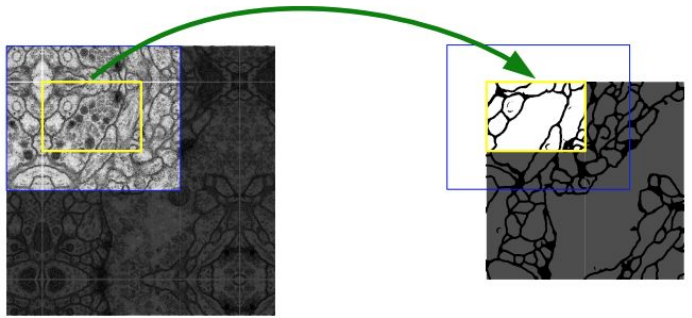
- Encoder-decoder architecture.
- When desired output should include localization, i.e., a class label is supposed to be assigned to each pixel
- Training in patches helps with lack of data

DeepLab - High Performance

- Atrous Convolution (Convolutions with upsampled filters)
- Allows user to explicitly control the resolution at which feature responses are computed

U-Net

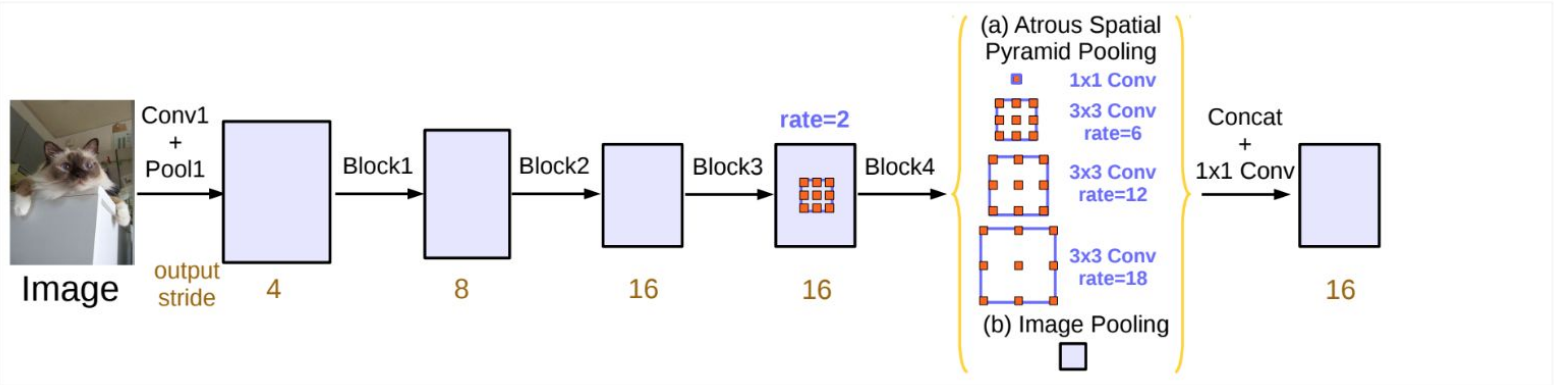




Figures from Ronneberger (2015). (https://arxiv.org/abs/1505.04597)

DeepLab

ResNet block uses atrous convolutions, uses different dilation rates to capture multi-scale context. On top of this new block, it uses Atrous Spatial Pyramid Pooling (ASPP). ASPP uses dilated convolutions with different rates as an attempt of classifying regions of an arbitrary scale.



Images from https://sthalles.github.io/deep_segmentation_network/

Other Segmentation Frameworks

U-Net (Keras) - https://github.com/zhixuhao/unet

DeepLab (Caffe) - https://github.com/Robotertechnik/Deep-Lab

DeepLabv3 (Tensorflow) - https://github.com/NanqingD/DeepLabV3-Tensorflow

Model Zoo

Model Zoo

https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md

Object Detection

https://github.com/tensorflow/models/blob/master/research/object_detection/object_detection_tutorial.ipynb

Further Reading

Speed/accuracy tradeoffs for modern convolutional object detectors (2017): https://arxiv.org/pdf/1611.10012.pdf