Deep Learning Applications
Module 2

Ganapathy Krishnamurthi

Object Detection

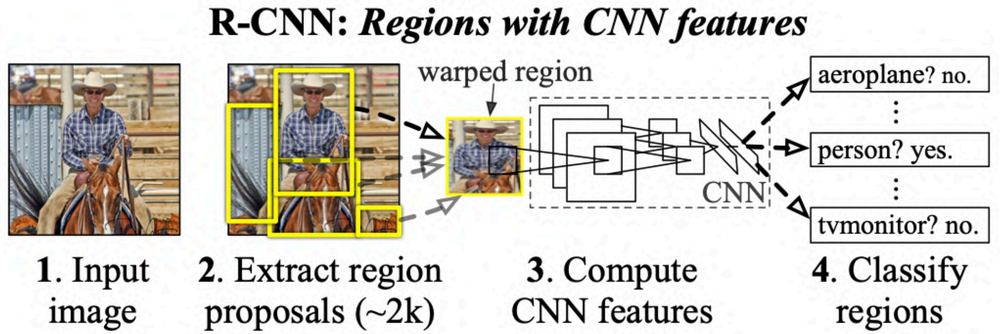
- Not all applications are image recognition, Image understanding i.e describing an image is the closest computer vision can get to Human like understanding of images
- Object detection - as the name implies- Determines the location of objects of interest in a scene
- Deep Learning for Image Recognition can be exploited for these tasks- Need appropriate data and ground truth annotations

Object Detection

- Object Localization/Detection is done by sliding a window across the object, extracting the window and passing it through an image recognition CNN.
- Infinite number of combinations of windows- not feasible
- Most object detection frameworks try to figure out a way to extract so called region proposals either using conventional computer vision algorithms or using a deep neural network.
- Proposals are passed it though a classification and regression network for determining class and bounding box.
- There are networks that extract proposals and classify in one pass through the network. They are much faster

R-CNN

- It can detect upto 80 different types of objects in images
- First module generates 2000 region proposals using an algorithm called selective search algorithm
- The regions are resized to a predetermined size and the second module which is a CNN extracts features (4096 dim)
- The last module uses SVM to classify the region proposal into background or one of the object classes



Rich feature hierarchies for accurate object detection and semantic segmentation
Tech report (v5)

Ross Girshick Jeff Donahue Trevor Darrell Jitendra Malik
UC Berkeley
{rbg,jdonahue,trevor,malik}@eecs.berkeley.edu

R-CNN

- Multi stage model - independent modules
- Requires huge storage for training
- Depends on a separate algorithm for generating region proposals
- Feature extraction is done independently for every region proposal - realtime execution is not possible

Fast R-CNN

- Similar to R-CNN
- A new layer called ROI pooling extracts equal-length feature vectors from all proposals in one image
- Faster R-CNN has only a single stage
- Computations are shared across all proposals using the ROI pooling layer which makes it faster than R-CNN
- Image is provided as input and the class probabilities and bounding box coordinates of the detected objects are output
- Still relies on selective search algorithm for region proposal generation

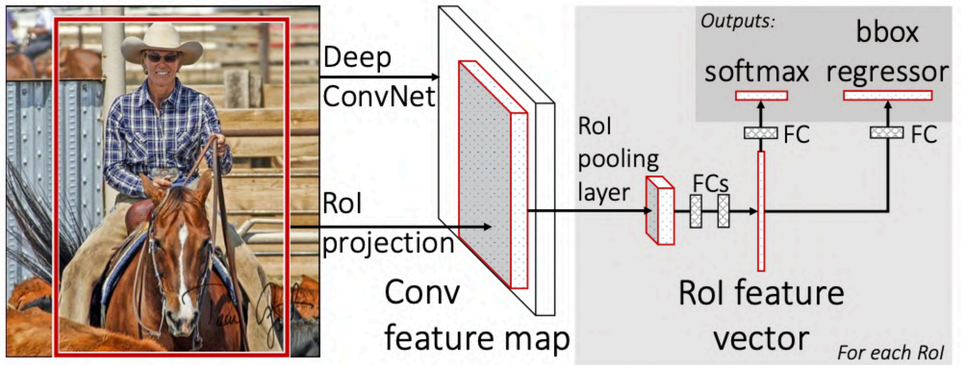


Figure 1. Fast R-CNN architecture. An input image and multiple regions of interest (RoIs) are input into a fully convolutional network. Each RoI is pooled into a fixed-size feature map and then mapped to a feature vector by fully connected layers (FCs). The network has two output vectors per RoI: softmax probabilities and per-class bounding-box regression offsets. The architecture is trained end-to-end with a multi-task loss.

https://arxiv.org/pdf/1504.08083.pdf

Fast R-CNN
RoI Pooling

- A fixed size feature map is obtained from deep CNN- the last convolutional layer for instance
- For every region proposal from the input, the appropriate section of the feature map is identified and scaled to a predefined size (7x7)
  - Divide region into cells
  - Find largest value in cell
- Allows one to use same feature map for analysing all region proposals

Faster R-CNN

- Region proposal network , a fully convolutional network that generates proposals using anchor boxes and different aspect ratios
- Convolutional computations are shared across the RPN and the Fast R-CNN
- Architecture of Faster R-CNN
  - RPN- Generates region proposals
  - Fast R-CNN- For object detection

Faster R-CNN

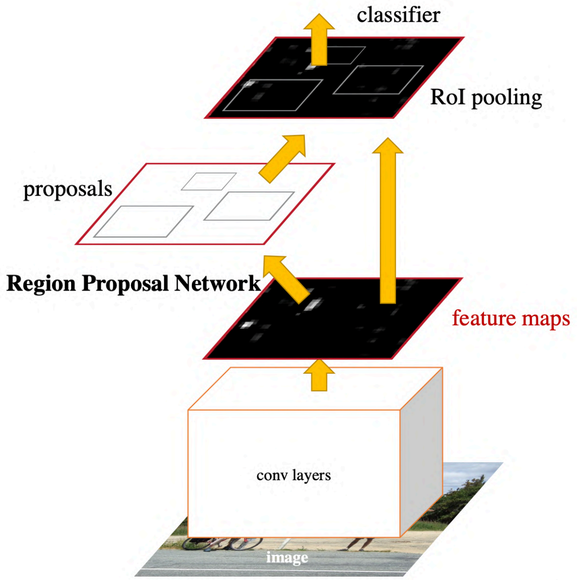


Figure 2: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network.

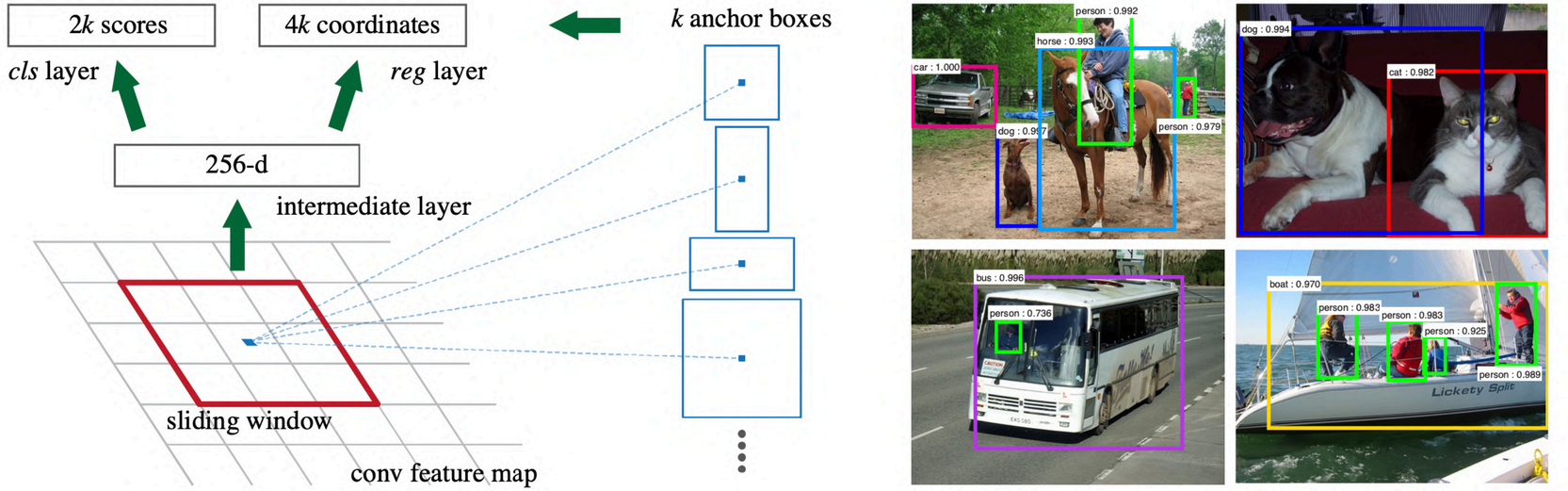


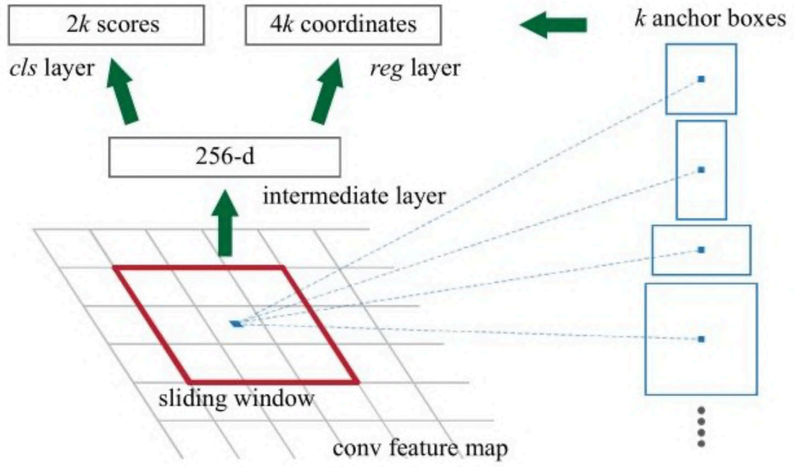
Figure 3: **Left**: Region Proposal Network (RPN). **Right**: Example detections using RPN proposals on PASCAL VOC 2007 test. Our method detects objects in a wide range of scales and aspect ratios.

Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun

Anchor boxes

- The last convolutional layer feature map is used for features
- A nxn window is slid across the feature map generating K region proposals
- Proposal is parametrised by a reference box - anchor box
- Scale and aspect ratio are the parameters
- Each nxn region proposal, a feature vector is extracted and input to 2 fully connected layers
- Cls layer for objectness score for each region proposal
- Reg layer which returns a 4D vector defining the bounding box of the region



Object Detection/Instance Segmentation - Mask R-CNN

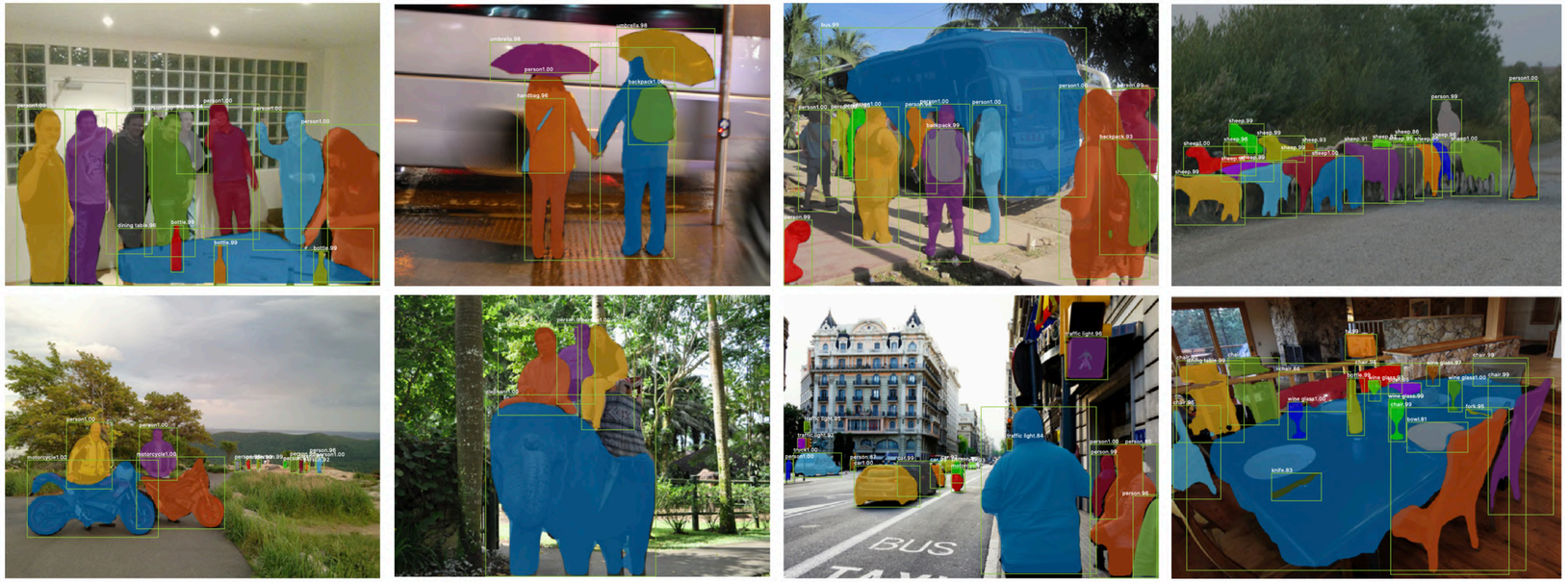


Figure 2. **Mask R-CNN** results on the COCO test set. These results are based on ResNet-101 [19], achieving a *mask* AP of 35.7 and running at 5 fps. Masks are shown in color, and bounding box, category, and confidences are also shown.

Mask R-CNN

Kaiming He Georgia Gkioxari Piotr Dollár Ross Girshick

Facebook AI Research (FAIR)

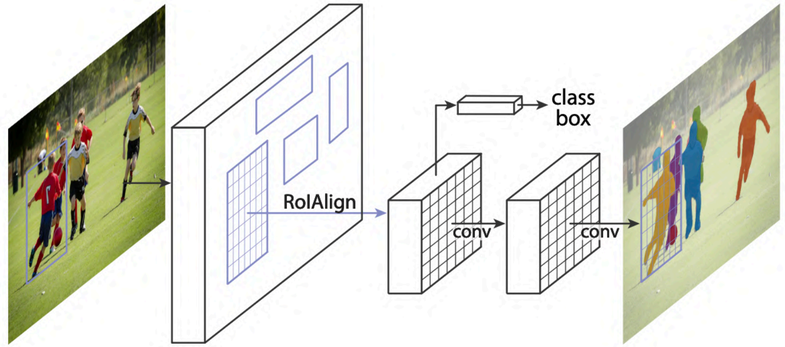


Figure 1. The **Mask R-CNN** framework for instance segmentation.

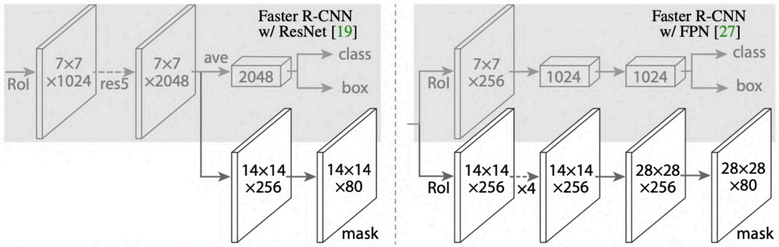


Figure 4. **Head Architecture**: We extend two existing Faster R-CNN heads [19, 27]. Left/Right panels show the heads for the ResNet C4 and FPN backbones, from [19] and [27], respectively, to which a mask branch is added. Numbers denote spatial resolution and channels. Arrows denote either conv, deconv, or *fc* layers as can be inferred from context (conv preserves spatial dimension while deconv increases it). All convs are 3×3, except the output conv which is 1×1, deconvs are 2×2 with stride 2, and we use ReLU [31] in hidden layers. *Left*: 'res5' denotes ResNet's fifth stage, which for simplicity we altered so that the first conv operates on a 7×7 RoI with stride 1 (instead of 14×14 / stride 2 as in [19]). *Right*: '×4' denotes a stack of four consecutive convs.

Mask R-CNN
RoI Align

- In RoI Pooling layers, we quantise the size of the boxes after mapping to the last convolutional layer as well as when covering the box into a fixed size feature map, for e.g. 3x3

- In RoI Align, no quantisation is done. The coordinates of the boxes are used as it is and interpolation is done to estimate the final feature map.

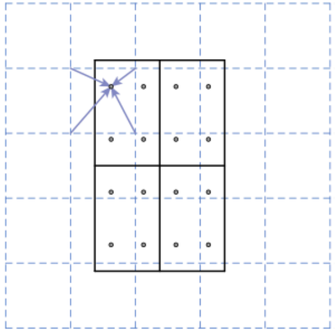


Figure 3. **RoIAlign:** The dashed grid represents a feature map, the solid lines an RoI (with 2×2 bins in this example), and the dots the 4 sampling points in each bin. RoIAlign computes the value of each sampling point by bilinear interpolation from the nearby grid points on the feature map. No quantization is performed on any coordinates involved in the RoI, its bins, or the sampling points.

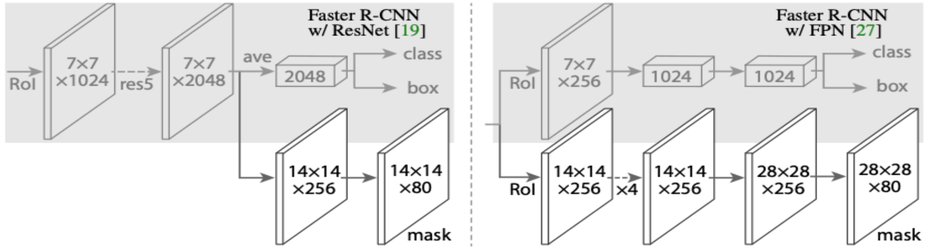
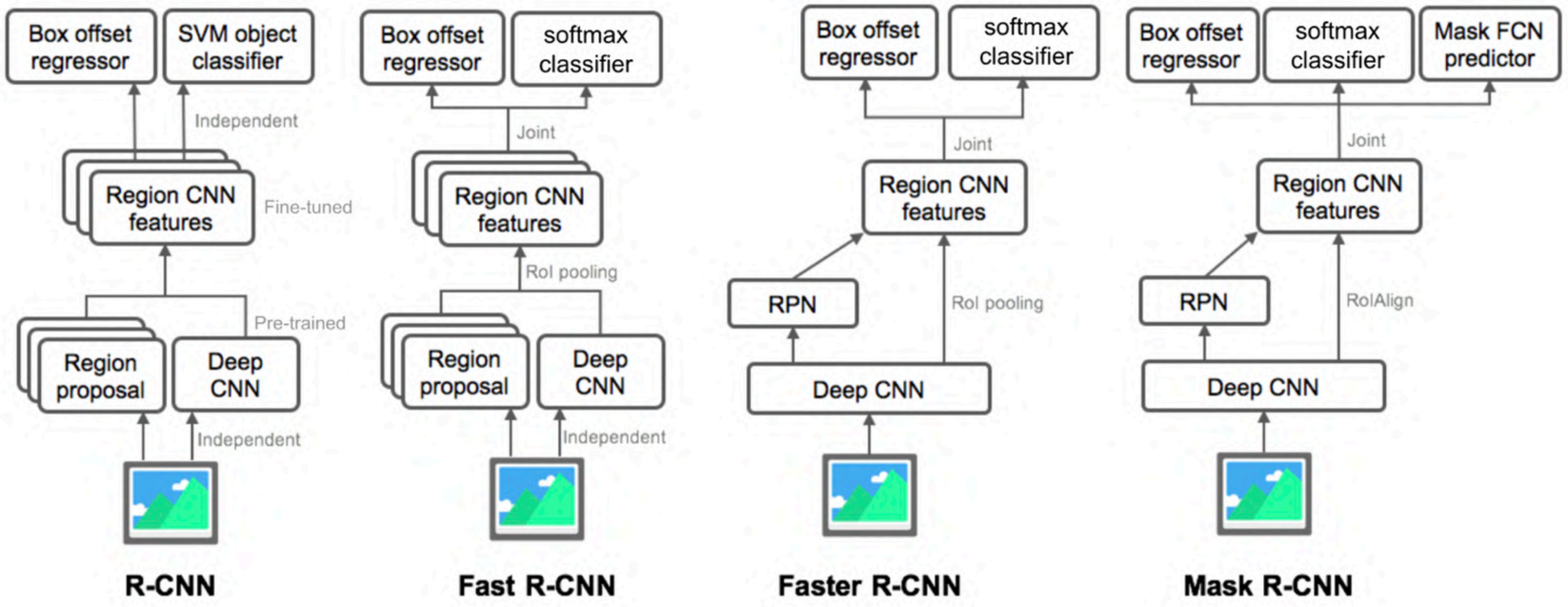


Figure 4. **Head Architecture**: We extend two existing Faster R-CNN heads [19, 27]. Left/Right panels show the heads for the ResNet C4 and FPN backbones, from [19] and [27], respectively, to which a mask branch is added. Numbers denote spatial resolution and channels. Arrows denote either conv, deconv, or *fc* layers as can be inferred from context (conv preserves spatial dimension while deconv increases it). All convs are 3×3, except the output conv which is 1×1, deconvs are 2×2 with stride 2, and we use ReLU [31] in hidden layers. *Left*: 'res5' denotes ResNet's fifth stage, which for simplicity we altered so that the first conv operates on a 7×7 RoI with stride 1 (instead of 14×14 / stride 2 as in [19]). *Right*: '×4' denotes a stack of four consecutive convs.

Model Comparison



Object Detection- RetinaNet

$$\mathrm{CE}(p, y) = \begin{cases} -\log(p) & \text{if } y = 1 \\ -\log(1-p) & \text{otherwise.} \end{cases}$$

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1-p & \text{otherwise,} \end{cases}$$

and rewrite $\mathrm{CE}(p, y) = \mathrm{CE}(p_t) = -\log(p_t)$.

$$\mathrm{CE}(p_t) = -\alpha_t \log(p_t).$$

$$\mathrm{FL}(p_t) = -\alpha_t (1-p_t)^\gamma \log(p_t).$$

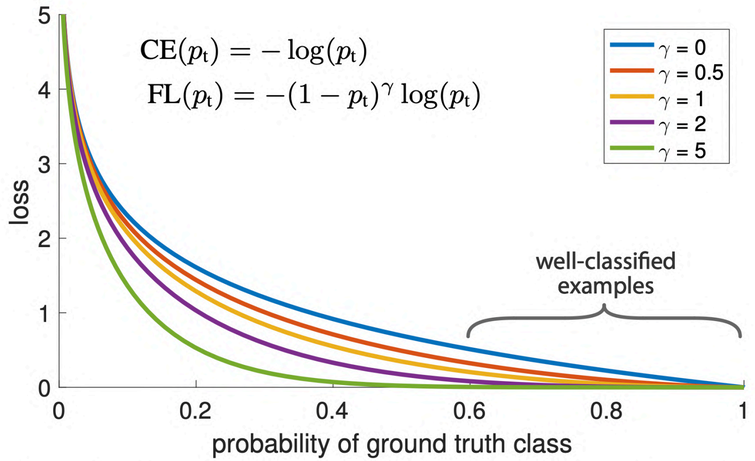
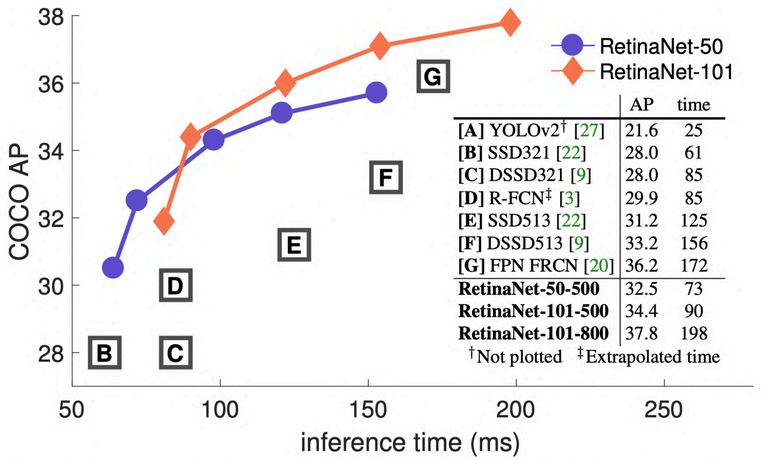


Figure 1. We propose a novel loss we term the *Focal Loss* that adds a factor $(1-p_t)^\gamma$ to the standard cross entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > .5$), putting more focus on hard, misclassified examples. As our experiments will demonstrate, the proposed focal loss enables training highly accurate dense object detectors in the presence of vast numbers of easy background examples.



	AP	time
[A] YOLOv2† [27]	21.6	25
[B] SSD321 [22]	28.0	61
[C] DSSD321 [9]	28.0	85
[D] R-FCN‡ [3]	29.9	85
[E] SSD513 [22]	31.2	125
[F] DSSD513 [9]	33.2	156
[G] FPN FRCN [20]	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	37.8	198

†Not plotted ‡Extrapolated time

Figure 2. Speed (ms) versus accuracy (AP) on COCO `test-dev`. Enabled by the focal loss, our simple one-stage *RetinaNet* detector outperforms all previous one-stage and two-stage detectors, including the best reported Faster R-CNN [28] system from [20]. We show variants of RetinaNet with ResNet-50-FPN (blue circles) and ResNet-101-FPN (orange diamonds) at five scales (400-800 pixels). Ignoring the low-accuracy regime (AP<25), RetinaNet forms an upper envelope of all current detectors, and an improved variant (not shown) achieves 40.8 AP. Details are given in §5.

Object Detection RetinaNet

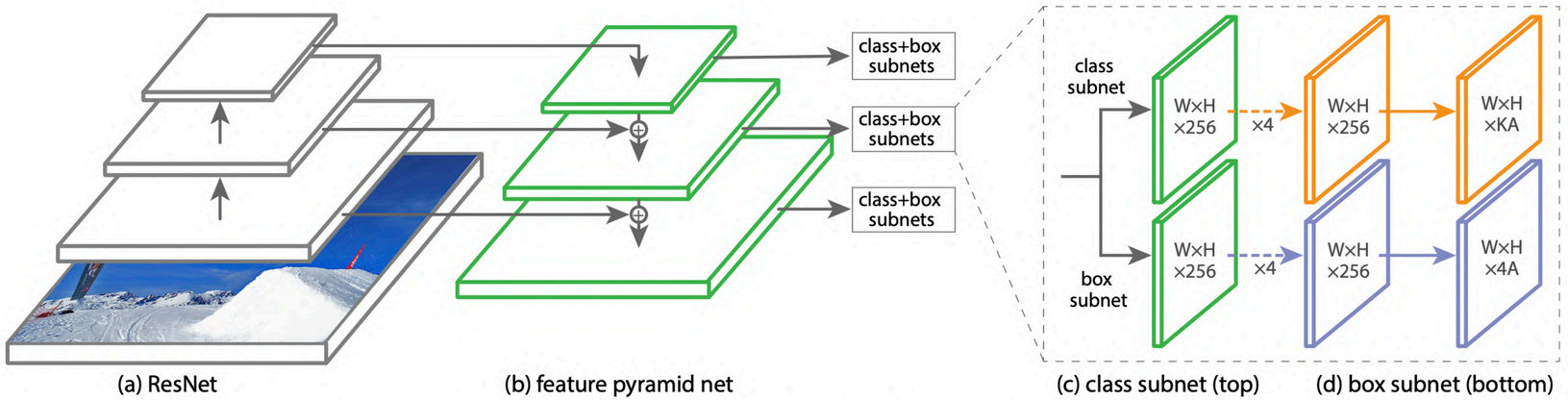


Figure 3. The one-stage **RetinaNet** network architecture uses a Feature Pyramid Network (FPN) [20] backbone on top of a feedforward ResNet architecture [16] (a) to generate a rich, multi-scale convolutional feature pyramid (b). To this backbone RetinaNet attaches two subnetworks, one for classifying anchor boxes (c) and one for regressing from anchor boxes to ground-truth object boxes (d). The network design is intentionally simple, which enables this work to focus on a novel focal loss function that eliminates the accuracy gap between our one-stage detector and state-of-the-art two-stage detectors like Faster R-CNN with FPN [20] while running at faster speeds.

Object Detection-Yolo

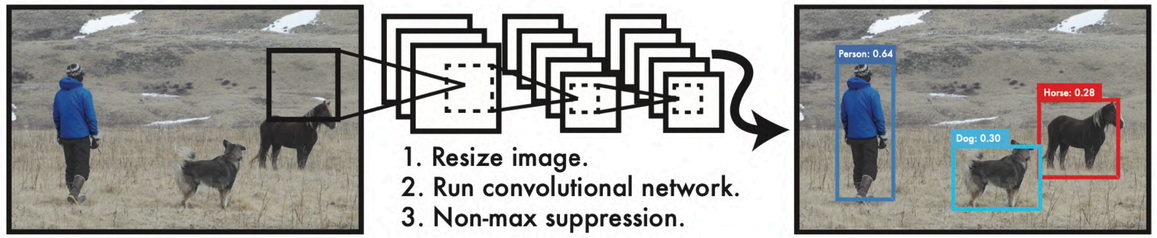


Figure 1: The YOLO Detection System. Processing images with YOLO is simple and straightforward. Our system (1) resizes the input image to 448 × 448, (2) runs a single convolutional network on the image, and (3) thresholds the resulting detections by the model's confidence.

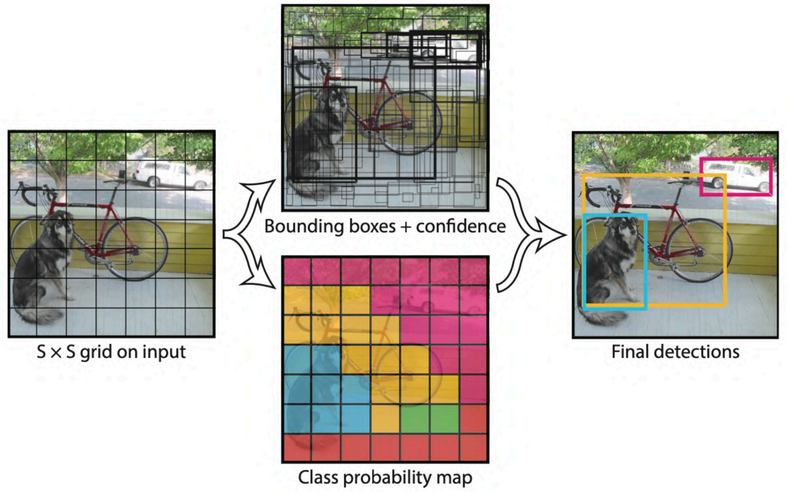
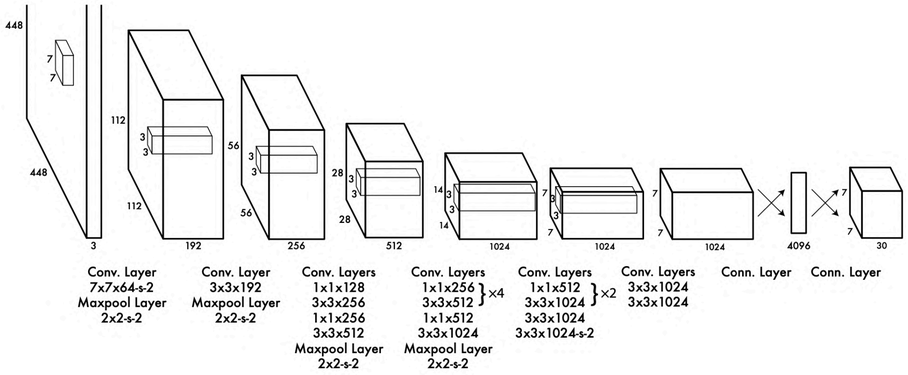


Figure 2: The Model. Our system models detection as a regression problem. It divides the image into an $S \times S$ grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B * 5 + C)$ tensor.

You Only Look Once:
Unified, Real-Time Object Detection

Joseph Redmon*, Santosh Divvala*†, Ross Girshick¶, Ali Farhadi*†
University of Washington*, Allen Institute for AI†, Facebook AI Research¶
http://pjreddie.com/yolo/

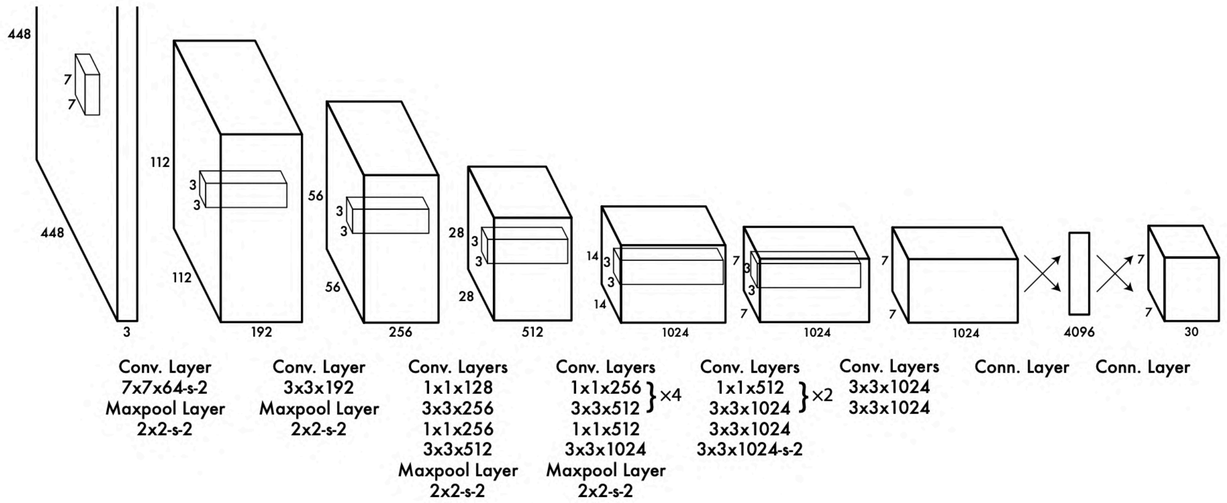


YOLO

- Use pertained network for image classification
- Split an image in to SxS cells.
- If object centre falls in a cell, that cell is “responsible” for detecting that instance
- Each cell predicts location of B bounding boxes, confidence score and probability of object class (given object in bounding box
- Network predicts probability of object belonging to every class. Only one set of probabilities per cell
- SXSXB bounding boxes. Four locations per box, 1 confidence score, and K conditional probabilities for object recognition=> SxSx(5B+K)

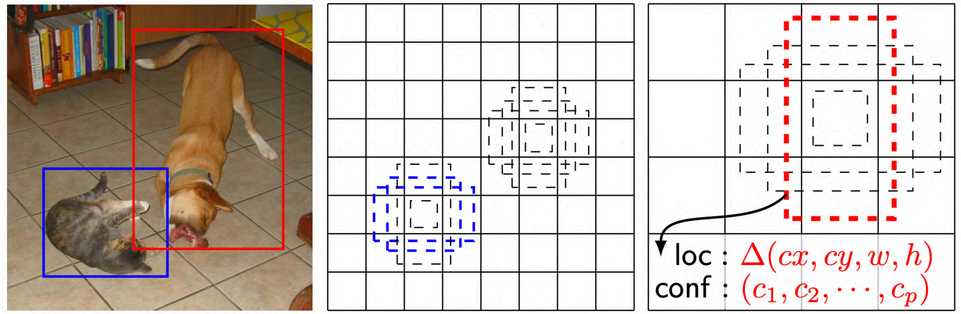
YOLO

- Output of pertained network is modified to shape SxSx(5B+K)
- Loss function has localisation loss and classification loss



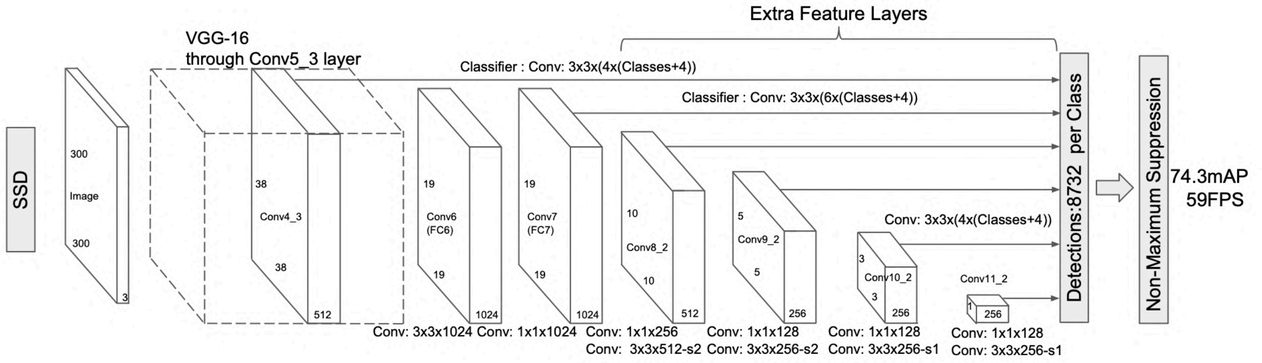
For evaluating YOLO on PASCAL VOC, we use $S = 7$, $B = 2$. PASCAL VOC has 20 labelled classes so $C = 20$. Our final prediction is a $7 \times 7 \times 30$ tensor.

SSD
Single Shot MultiBox Detector



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

Fig. 1: **SSD framework.** (a) SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, we evaluate a small set (e.g. 4) of default boxes of different aspect ratios at each location in several feature maps with different scales (e.g. 8×8 and 4×4 in (b) and (c)). For each default box, we predict both the shape offsets and the confidences for all object categories ($(c_1, c_2, \cdots, c_p)$). At training time, we first match these default boxes to the ground truth boxes. For example, we have matched two default boxes with the cat and one with the dog, which are treated as positives and the rest as negatives. The model loss is a weighted sum between localization loss (e.g. Smooth L1 [6]) and confidence loss (e.g. Softmax).

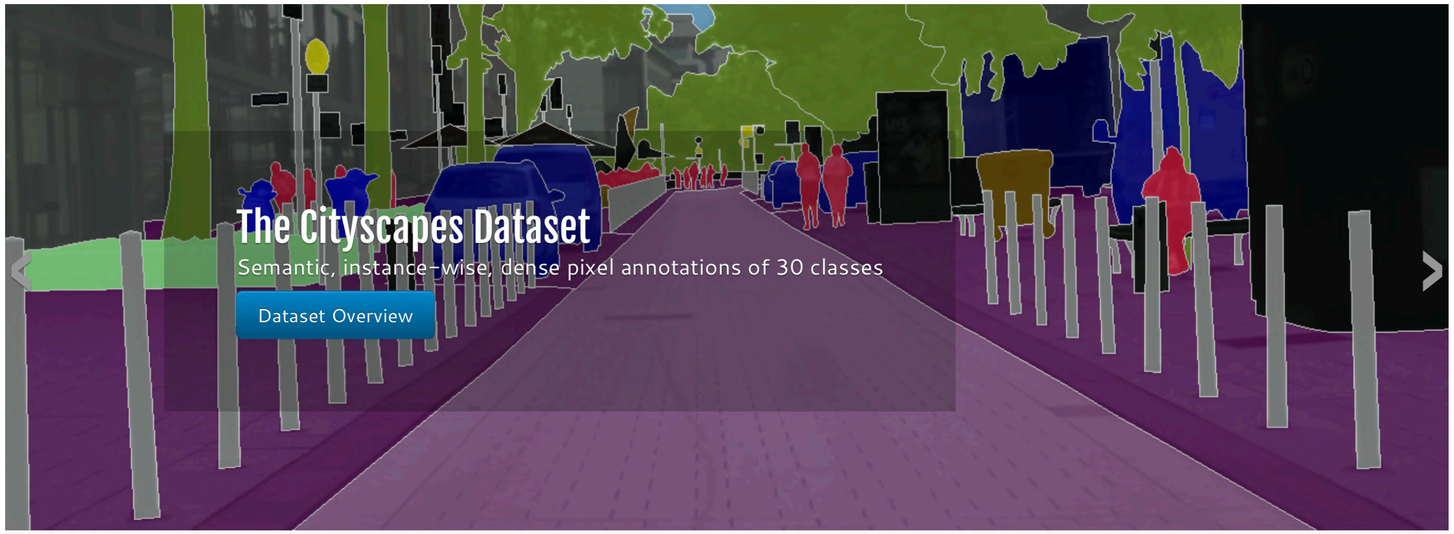


Metrics for Segmentation and Detection

- Dice Score or Intersection over union for object localisation
- For detection/recognition top-5 or top-1 accuracy

Datasets for Detection/localization

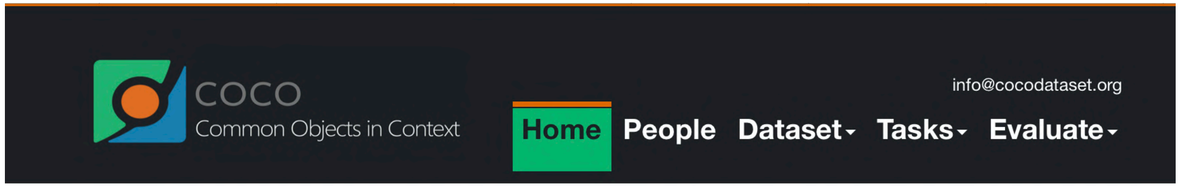
- ImageNet
- Cityscapes
- Common Objects in Context



What is COCO?

COCO is a large-scale object detection, segmentation, and captioning dataset. COCO has several features:

- Object segmentation
- Recognition in context
- Superpixel stuff segmentation
- 330K images (>200K labeled)
- 1.5 million object instances
- 80 object categories
- 91 stuff categories
- 5 captions per image
- 250,000 people with keypoints



YOLO Loss Function

Our final layer predicts both class probabilities and bounding box coordinates. We normalize the bounding box width and height by the image width and height so that they fall between 0 and 1. We parametrize the bounding box x and y coordinates to be offsets of a particular grid cell location so they are also bounded between 0 and 1.

$$
\begin{aligned}
&\lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\textbf{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \qquad (3)
\end{aligned}
$$

R-CNN loss function

$$\hat{G}_x = P_w d_x(P) + P_x \quad (1)$$
$$\hat{G}_y = P_h d_y(P) + P_y \quad (2)$$
$$\hat{G}_w = P_w \exp(d_w(P)) \quad (3)$$
$$\hat{G}_h = P_h \exp(d_h(P)). \quad (4)$$

Each function $d_\star(P)$ (where $\star$ is one of x, y, h, w) is modeled as a linear function of the pool_5 features of proposal P, denoted by $\boldsymbol{\phi}_5(P)$. (The dependence of $\boldsymbol{\phi}_5(P)$ on the image data is implicitly assumed.) Thus we have $d_\star(P) = \mathbf{w}_\star^{\mathrm{T}} \boldsymbol{\phi}_5(P)$, where $\mathbf{w}_\star$ is a vector of learnable model parameters. We learn $\mathbf{w}_\star$ by optimizing the regularized least squares objective (ridge regression):

$$\mathbf{w}_\star = \underset{\hat{\mathbf{w}}_\star}{\operatorname{argmin}} \sum_i^N (t_\star^i - \hat{\mathbf{w}}_\star^{\mathrm{T}} \boldsymbol{\phi}_5(P^i))^2 + \lambda \left\| \hat{\mathbf{w}}_\star \right\|^2$$

$$t_x = (G_x - P_x)/P_w$$
$$t_y = (G_y - P_y)/P_h$$
$$t_w = \log(G_w/P_w)$$
$$t_h = \log(G_h/P_h).$$

Hard Negative Mining

- Get Positive training samples
- A random subset of negative training samples
- Train network
- Test on unseen data i.e. images
- Determine false positives
- Take some of the false positives from the previous step and add to training set
- Repeat training

Non Maximal suppression

- Object detection networks have a large number of proposals per image.
- Leads to a lot of boxes per image and per object
- Have to choose one while suppressing the rest
- Usually a confidence score is output per proposal
  - Algorithm
    - Select proposal with highest confidence score
    - Remove all other proposals with IoU with current proposal greater than a threshold
    - Iterate