ELEG 5491: Introduction to Deep Learning

Object Detection in Computer Vision

Prof. LI Hongsheng

Office: SHB 428
e-mail: hsli@ee.cuhk.edu.hk
web: https://dl.ee.cuhk.edu.hk

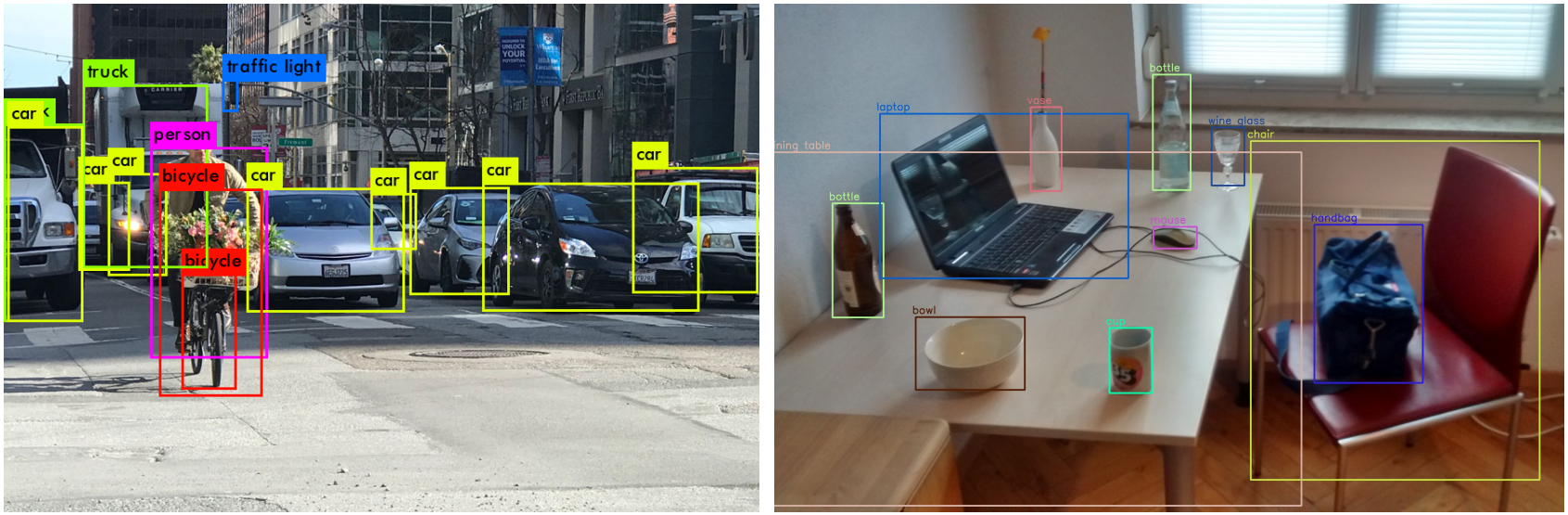
Department of Electronic Engineering
The Chinese University of Hong Kong

March 2023

Outline

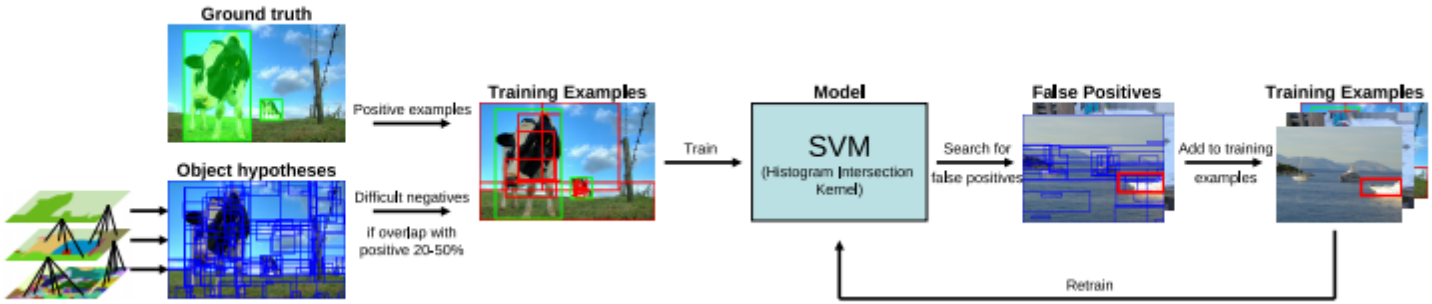
The Problem of Object Detection

- Given an input image, **object detection** aims at localizing objects of inerest with rectangle boxes and class labels (usually called bounding boxes)



Selective Search for Object Detection

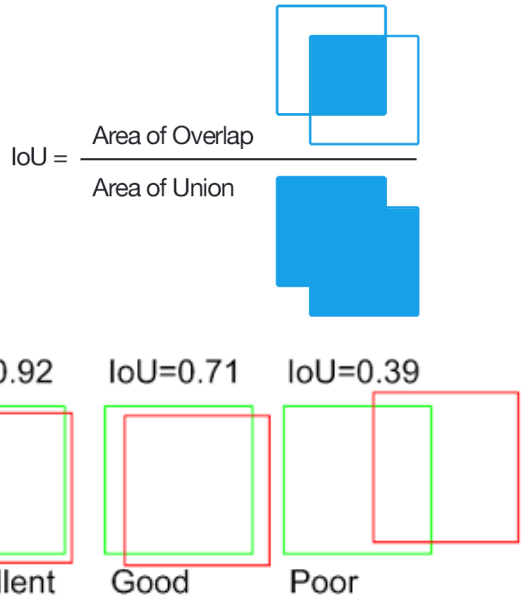
- Selective search is a method for generating a series of object proposals or object hypotheses (candidate bounding boxes that are likely to enclose objects of inerest) from the input image
- One of the first attempts of general object detection with deep learning is R-CNN



- It uses hierarchical clustering on pixels to obtain clusters and then obtains object proposals
- An image patch is cropped from each object proposal. A multi-class classifier (SVM in the original paper) is then used to classify each each patch into the `background` or one of the foreground (object of interest) classes

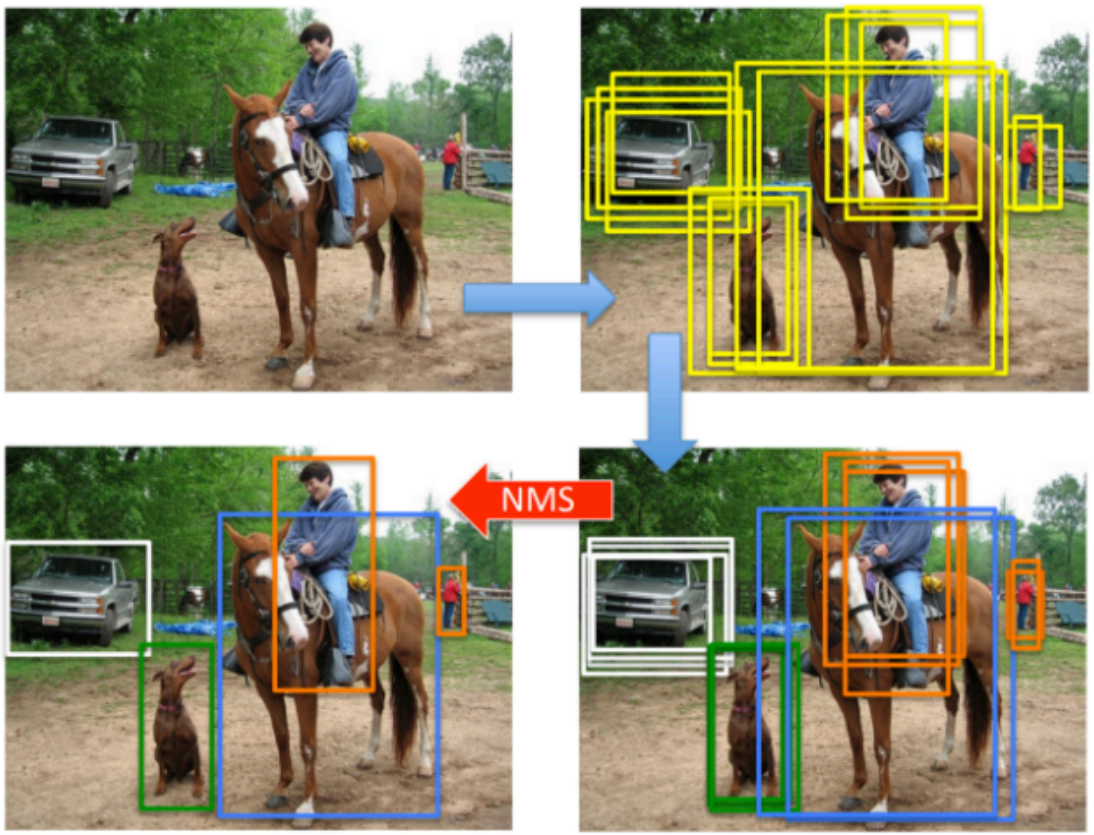
Box Overlapping

- During the training, we would like to determine positives and negatives from the object hypotheses with ground-truth bounding boxes
- **Intersection-over-Union** is used to determine the overlapping between two bounding boxes
- IoU $= 1$ means 100% overlapping between two boxes, and IoU $= 0$ means 0% overlapping between two boxes
- The selective search paper uses proposals of IoU$\in [0.2, 0.5]$ with ground-truths as difficult (hard) negatives



Non-maximum Suppression

- During the testing stage, a ground-truth object might be covered by multiple bounding box



- **Non-Maximum Suppression (NMS)** is introduced to let higher-scoring boxes to "kill" lower-scoring boxes

Non-maximum Suppression

- **Non-Maximum Suppression (NMS)** is introduced to let higher-scoring boxes to "kill" lower-scoring boxes. Greedy NMS is of most frequently used
- The following algorithm is executed for each class separately
- For instance, when handling the "person" class boxes, only the person detection boxes form the $\mathcal{B}$ set

Input : $\mathcal{B} = \{b_1, .., b_N\}$, $\mathcal{S} = \{s_1, .., s_N\}$, N_t
$\mathcal{B}$ is the list of initial detection boxes
$\mathcal{S}$ contains corresponding detection scores
N_t is the NMS threshold

begin
  $\mathcal{D} \leftarrow \{\}$
  while $\mathcal{B} \neq$ *empty* **do**
    $m \leftarrow$ argmax $\mathcal{S}$
    $\mathcal{M} \leftarrow b_m$
    $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{M}$; $\mathcal{B} \leftarrow \mathcal{B} - \mathcal{M}$
    for b_i *in* $\mathcal{B}$ **do**
      if $iou(\mathcal{M}, b_i) \geq N_t$ **then**
        $\mathcal{B} \leftarrow \mathcal{B} - b_i$; $\mathcal{S} \leftarrow \mathcal{S} - s_i$
      end
      NMS
      $s_i \leftarrow s_i f(iou(\mathcal{M}, b_i))$
      Soft-NMS
    end
  end
  return $\mathcal{D}, \mathcal{S}$

Detection Datasets

- PASCAL Visual Object Classes (VOC) 2012 dataset contains 20 object categories. It is split into three subsets: 1,464 images for training, 1,449 images for validation and a private testing set
- COCO is a large-scale detection benchmark has 80 object categories with `trainval35k` split (115k images) for training, `minival` split (5k images), and `test_dev` split (20k iamges) without any annotation
- ImageNet has 200 classes with 456k training images, 20k validation images, and 40 test images
- OpenImage is released by Google. Its latest v6 version has 1.6M boxes of 600 categories on 1.9M images, making it the largest existing dataset with object location annotations. However, currently only some large companies "play" it as it requires a huge amount of computational resources

Evaluation Metric

- For a given object detection box, it is considered correct if its IoU with a ground-truth box is over 0.5 (sometimes other thresholds are also used). If the IoU> 0.5, the detection box is considered true positive (TP), otherwise it is considered as false positive (FP)
- Precision and recall are defined as

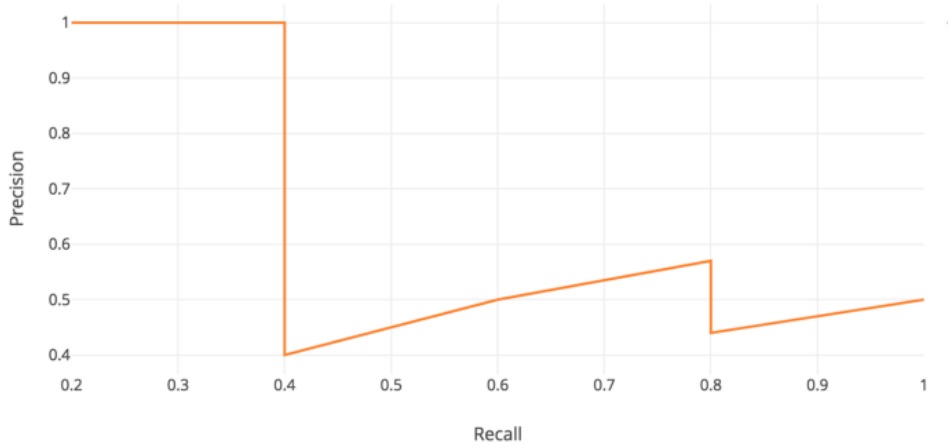
$$\text{Preciesion} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- Assume that there are 5 objects in an image and a model generates 10 boxes
- The boxes are first ranked according to their predicted confidence scores

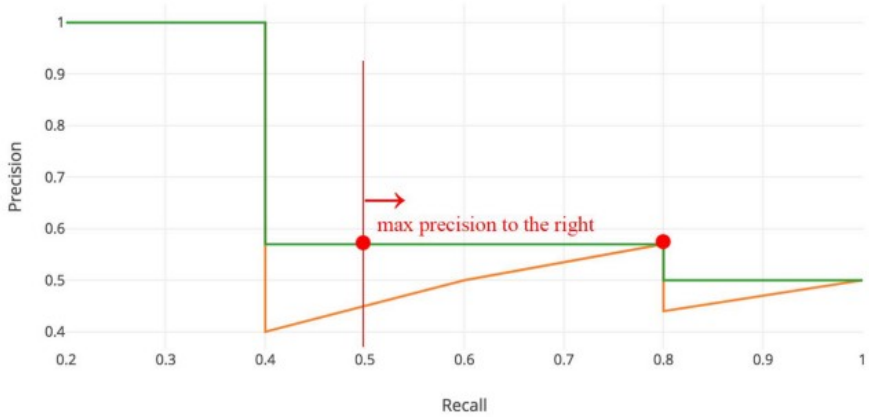
Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

Evaluation Metric

- If we count increasing numbers of detection boxes as the final results, the recall increases and the precision gradually decreases

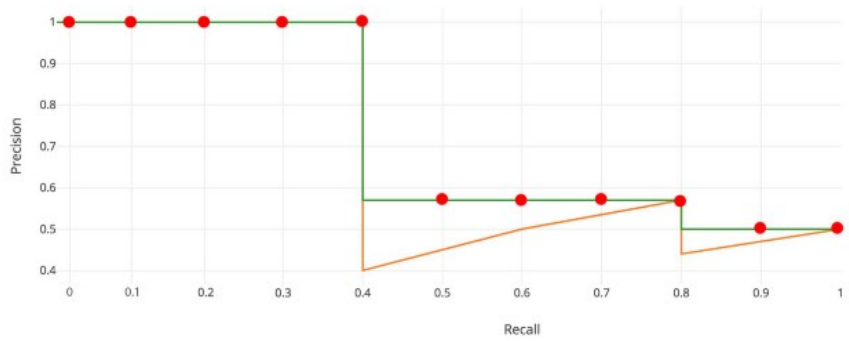


- We smooth out the P-R curve



Evaluation Metric

- **Interpolated AP:** we can divide the recall value into multiple points and compute the average of maximum precision value of them

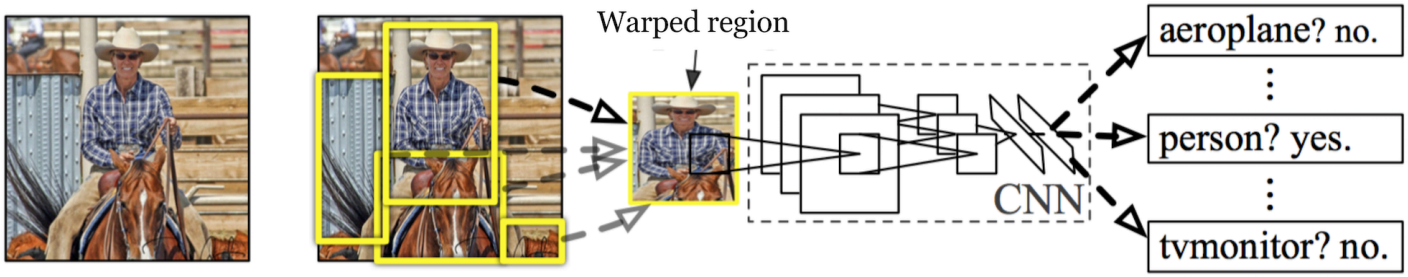


- **COCO mAP:** For COCO, AP$[.5 : .95]$ corresponds to the average AP for IoU from 0.5 to 0.95 with a step size of 0.05. AP is the average over 10 IoU levels on 80 categories

```
Average Precision (AP):
   AP                  % AP at IoU=.50:.05:.95 (primary challenge metric)
   AP^IoU=.50          % AP at IoU=.50 (PASCAL VOC metric)
   AP^IoU=.75          % AP at IoU=.75 (strict metric)
AP Across Scales:
   AP^small            % AP for small objects: area < 32^2
   AP^medium           % AP for medium objects: 32^2 < area < 96^2
   AP^large            % AP for large objects: area > 96^2
Average Recall (AR):
   AR^max=1            % AR given 1 detection per image
   AR^max=10           % AR given 10 detections per image
   AR^max=100          % AR given 100 detections per image
AR Across Scales:
   AR^small            % AR for small objects: area < 32^2
   AR^medium           % AR for medium objects: 32^2 < area < 96^2
   AR^large            % AR for large objects: area > 96^2
```

R-CNN

- R-CNN is short for "Region-based Convolutional Neural Networks". It follows the pipeline of the previous conventional method but replaces the SVM classifier with the CNN classifier
- Pipeline



- Category-independent region proposals are obtained via selective search
- Region proposals are resized to a fixed size as required by a CNN
- Train the CNN for $(K+1)$-class classification (background + K foreground classes). Abandon the last FC (classification) layer (in their original paper)
- Given every image region, one forward propagation through the CNN generates a feature vector, which is then fed into a binary SVM trained for each class independently

Bounding Box Regression in R-CNN

- The positive samples are proposed regions with IoU≥ 0.3, and negative samples are irrelevant others.
- The positions of the candidate boxes generated by selective search might not be accurate
- A regression head is trained to correct the predicted detection box by estimating correction offsets using CNN features
- Given a region proposal's box coordinate $\mathbf{p} = (p_x, p_y, p_w, p_h)$ (center coordinate, width, height) and its corresponding ground truth box coordinates $\mathbf{g} = (g_x, g_y, g_w, g_h)$, the regressor is trained to predict the following targets

$$
\begin{aligned}
t_x &= (g_x - p_x)/p_w \\
t_y &= (g_y - p_y)/p_h \\
t_w &= \log(g_w/p_w) \\
t_h &= \log(g_h/p_h)
\end{aligned}
$$

- Given the predicted $\hat{t}_x, \hat{t}_y, \hat{t}_w, \hat{t}_h$, the modified box $(\hat{g}_x, \hat{g}_y, \hat{g}_w, \hat{g}_h)$ becomes

$$
\begin{aligned}
\hat{g}_x &= p_w \hat{t}_x + p_x \\
\hat{g}_y &= p_h \hat{t}_y + p_y \\
\hat{g}_w &= p_w \exp(\hat{t}_w) \\
\hat{g}_h &= p_h \exp(\hat{t}_h)
\end{aligned}
$$

The regression loss and the overall loss

- The regression loss is modeled as MSE/L2 loss

$$\mathcal{L}_{\text{reg}} = \sum_{i \in \{x,y,w,h\}} (t_i - \hat{t}_i)^2 + \lambda \|\theta\|_2^2,$$

  where $\|\theta\|_2^2$ is the weight regularization term

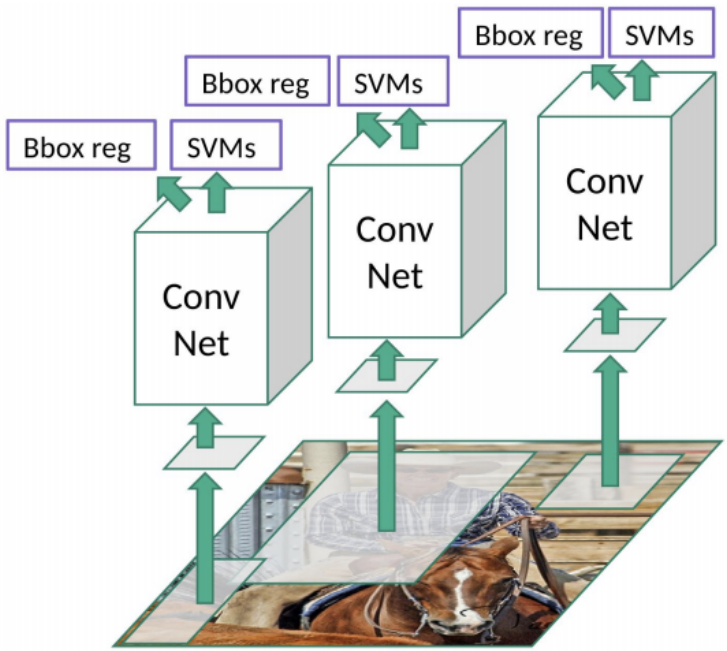
- The classification loss $\mathcal{L}_{\text{cls}}$ is modeled a conventional multi-class cross entropy loss, and the overall loss is a weighted combination of the two loss terms

$$\mathcal{L}_{\text{overall}} = \mathcal{L}_{\text{cls}} + \gamma \mathcal{L}_{\text{reg}}$$

  where γ is a weighting hyper-parameter between the two sub-tasks

Problems with R-CNN

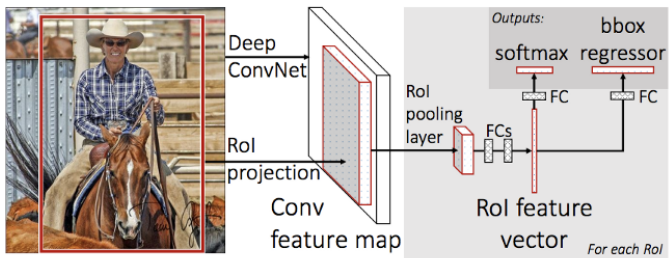
- It still takes a huge amount of time to train the network as you would have to classify 2,000 region proposals per image
- It cannot be implemented in real time as it takes around 47 seconds for each test image
- The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals

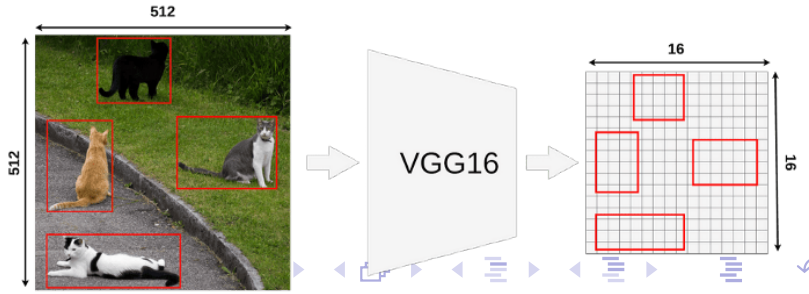


Fast R-CNN

- Fast R-CNN is proposed to solve some issues of the R-CNN
- Instead of cropping and resizing image patches from the input image, the input image is first processed by a CNN to obtain its feature maps, whose spatial size is generally smaller than the original image (e.g., 1/32 of the input image size)
- Given a region proposal, it is resized to match the size of the feature map
- The RoI pooling operation is introduced to crop the feature maps with the resized object proposal, and the cropped feature maps are then further resized to a fixed spatial size ($7 \times 7 \times 512$ in the original paper)
- **Positive samples:** IoU≥ 0.5; **Negative samples:** IoU$\in [0.1, 0.5)$. Each mini-batch consists of 1:3 positive and negative samples
- Smooth L1 loss function for bounding box regression

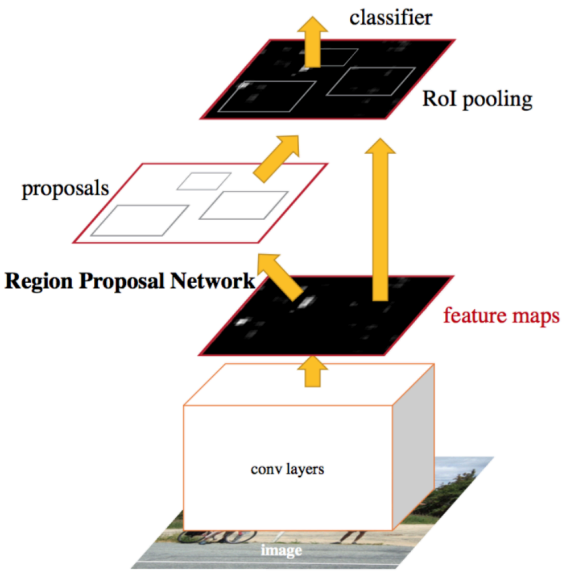
$$\mathcal{L}_1^{\text{smooth}}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

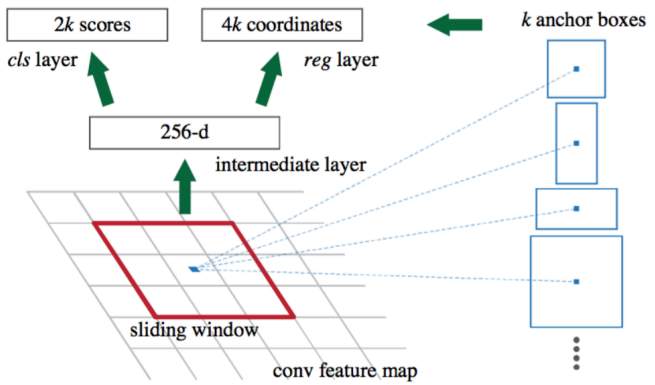




Faster R-CNN

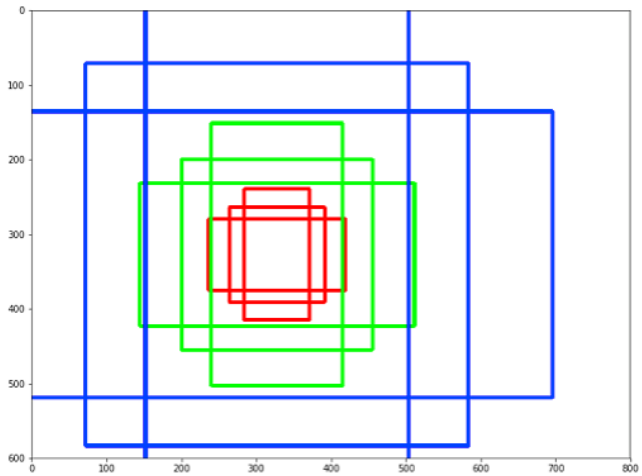
- **Problems with Fast R-CNN:**
  - RoI pooling (we skip details here) quantizes RoI coordinates into image coordinates. The pooled features are not accurate
  - Region proposals are not learned from training data
- Faster R-CNN was proposed to make the region proposals generated from the CNN as well
- The first part of the CNN is called Region Proposal Network (RPN) to generate object proposals. The RPN first generates an $n \times n$ feature maps, which are then processed by $3 \times 3 + 1 \times 1$ conv to generate

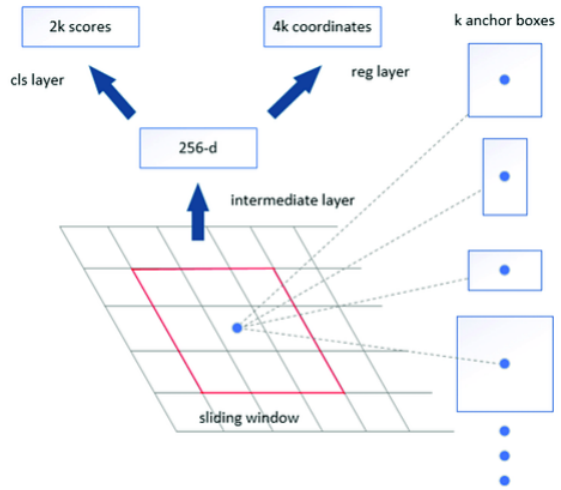




Faster R-CNN

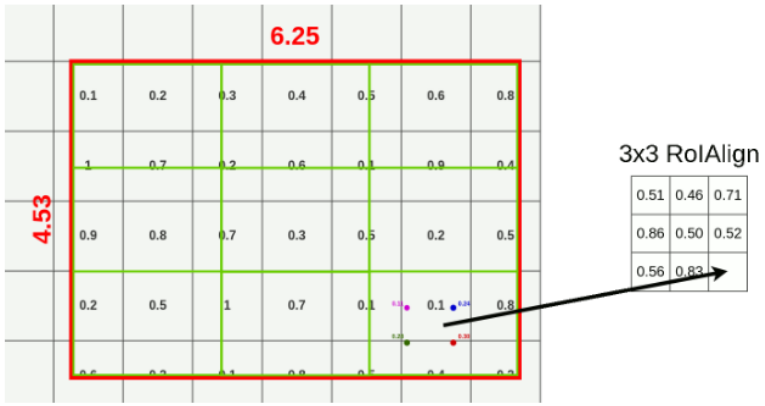
- At each of the $n \times n$ spatial location of the topmost feature maps from the RPN, $k = 9$ **anchor boxes** are used as initial boxes for estimating object proposals, i.e., offsets are predicted based on the initial boxes to predict the object proposals
- 3 scales and 3 aspect ratios ($3 \times 3 = 9$ anchor boxes) are considered at each spatial location
- For each spatial location, 3×3 conv $+ 1 \times 1$ conv layers are built up on the RPN topmost feature maps, to predict $2k$ proposal (yes/no) confidence scores and $4k$ proposal regression (center, size, width, height)
- After the region proposals are generated, NMS (IoU threshold 0.7) is performed for each class independently to remove duplicate region proposals with the predicted proposal confidence scores





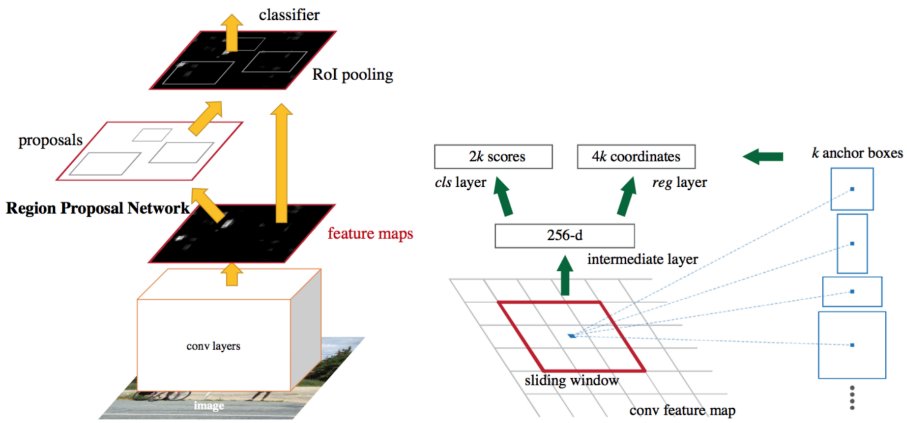
Region Features Cropping with ROI Align

- ROI Align is now the standard operator for cropping features from a feature map according to a region proposal
- Given a feature map and a bounding box, the feature patch in the bounding box can be cropped out by ROI Align
- 7×7 and 14×14 are the two most commonly choices of the ROI feature patch size
- Given each location of inerest, bilinear interpolation followed by 2×2 max pooling is conducted for each feature map channel independently.
- If the input feature map is of size $h \times w \times c$, the ROI Aligned feature (sometimes called the ROI features or regional features) can be of size $7 \times 7 \times c$, which has the same channel number as the input feature map



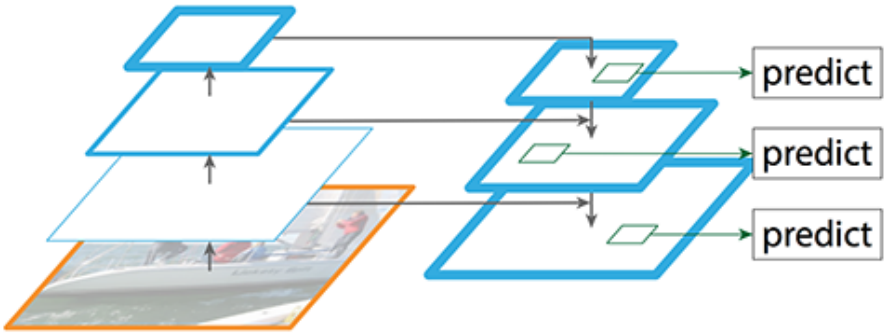
Faster R-CNN

- For region proposal, two types of positive labels: 1) the anchor(s) with the highest IoU overlap with a ground-truth box; 2) an anchor that has an IoU overlap higher than 0.7. Negative labels: an anchor has an IoU lower than 0.3 for any ground-truth box
- As there are generally more negative samples than positive samples in each image. Each image contains 1:3 to 1:1 positive and negative samples. If not enough positive samples, pad the mini-batch with the negative ones
- After the RPN is trained, it can generate region proposals, which RoI pools features from the proposals for training a Fast R-CNN model with a classification head and a regression head
- Faster R-CNN shares parameters for both the RPN and the Fast R-CNN model, and jointly traines both RPN and Fast R-CNN



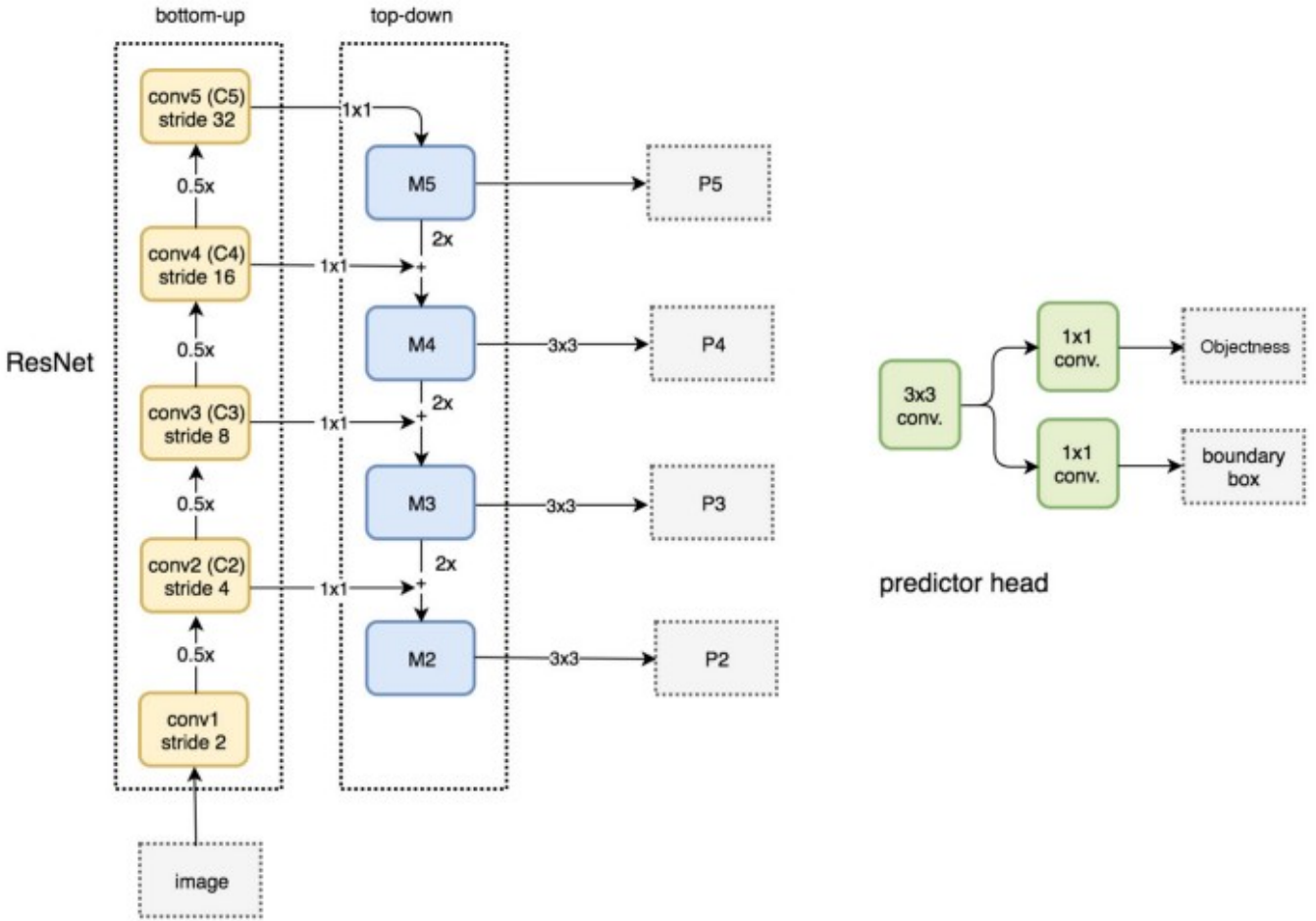
Feature Pyramid Network

- The above Faster R-CNN RoI pools features from only the topmost feature maps. However, their spatial sizes might be too small for localizing small-scale objects
- For instance, for an 16×16 tiny object in the original image. Its corresponding spatial size in the $1/32$ topmost feature maps might be smaller than one feature pixel
- Feature Pyramid is a series of features with different scales, so that large objects can be detected from low-res feature maps while small objects can be detected from high-res feature maps
- Feature Pyramid network (FPN) gradually upsamples feature maps with bilinear interpolation and add skip connections to additively fuse encoder and decoder feature maps



Feature Pyramid Network

- The FPN can generate region proposals from the multi-scale feature maps
- Anchor boxes of different scales are used in P_2, P_3, P_4, P_5

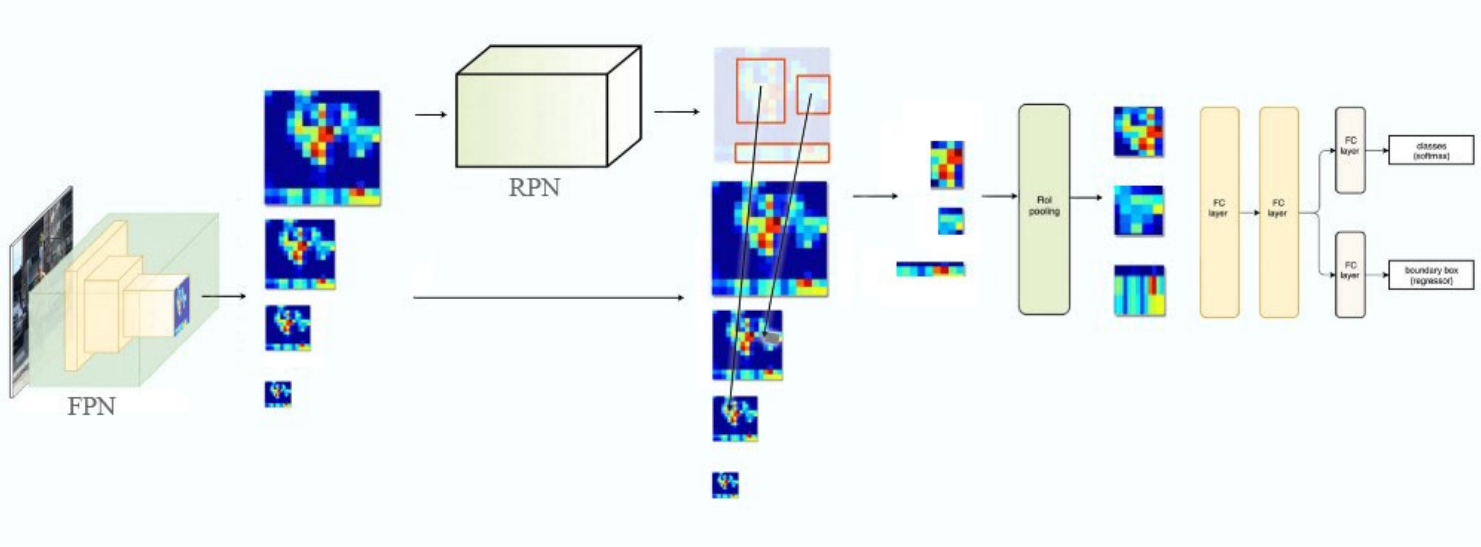


Feature Pyramid Network with Faster R-CNN

- Based on the size of the ROI, we select the feature map layer in the most proper scale to extract the feature patches
- The formula to pick the feature maps is based on the width w and height h of the ground-truth box

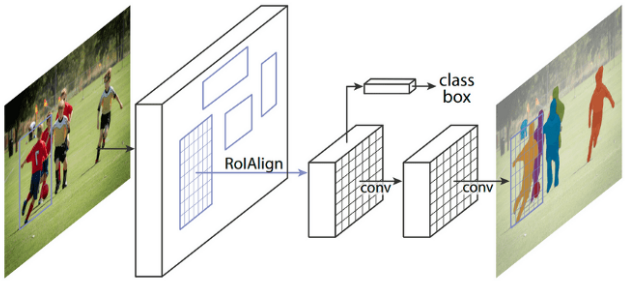
$$k = \left\lfloor k_0 + \log_2(\sqrt{wh}/224) \right\rfloor$$

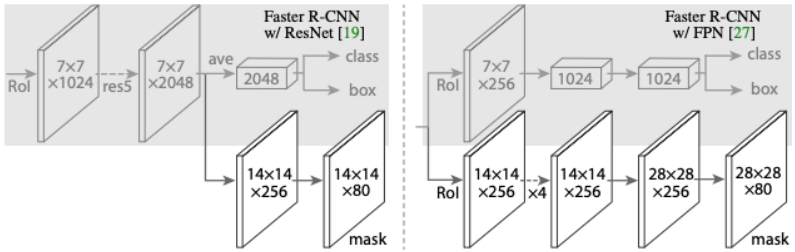
where k_0 denotes the scale of the topmost feature maps and k denotes the P_k layer of the FPN to generate the feature patch

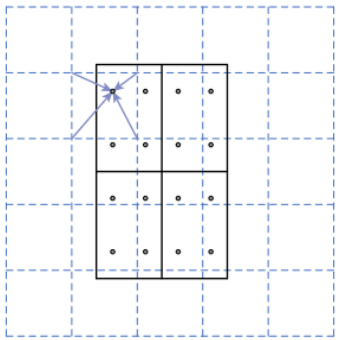


Mask R-CNN

- Mask R-CNN is proposed to not just tackle the object detection but also the instance segmentation problem, where each instance requires to obtain a separate mask
- It has the existing classification and regression head. Its major change is an instance segmentation head for segmenting instance-level mask
- It also replaces the RoI pool operation with the RoI Align operation, which uses bilinear interpolation to extract the feature patches from the topmost feature maps





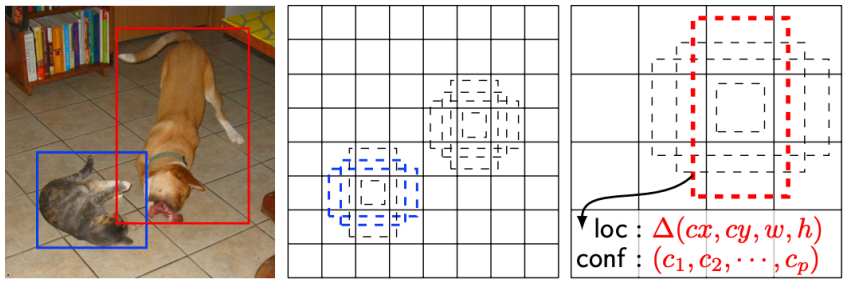


Online Hard Negative Mining (OHEM)

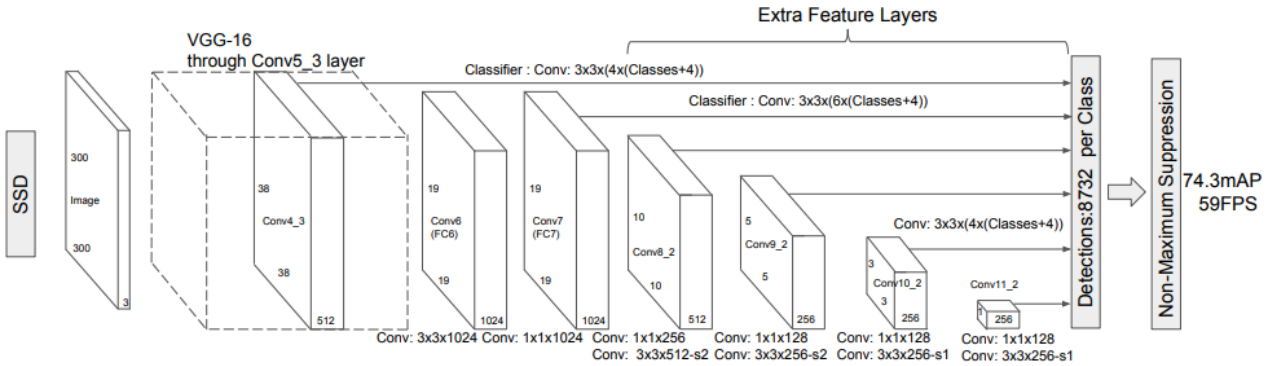
- In Faster R-CNN or Fast R-CNN, each mini-batch consists of positive samples from an image and a random subset of negative samples. It forms a mini-batch of 1:3 positive and negative samples
- However, most negative samples might be too simple and don't provide enough supervisions on training the object detector
- Given an input image and a series of region proposals, instead of randomly selecting a subset of negative samples. OHEM forward passes all region proposals and obtains their loss values for each RoI
- Each of their loss value represents how well the current network performs on each RoI. Hard examples are selected by sorting the input RoIs by loss and taking the B/N examples for which the current network performs worst
- Co-located RoIs with high overlap are likely to have correlated losses. Given a list of RoIs and their losses, NMS with IoU threshold 0.7 is used by iteratively selecting the RoI with the highest loss, and then removing all lower loss RoIs that have high overlap with the selected region
- The procedure described above does not need a foreground-background ratio for data balancing

Single Shot MultiBox Detector (SSD)

- The previous methods are called two-stage methods, as they generate region proposals first, and use an additional classification-and-regression head for refining the proposals
- Single-shot Detector (SSD) is an one-stage detector. It can be viewed as just using the RPN part of Faster R-CNN to generate object detection boxes
- It doesn't adopt FPN but still uses feature maps of different scales to generate object detection boxes corresponding to different scales

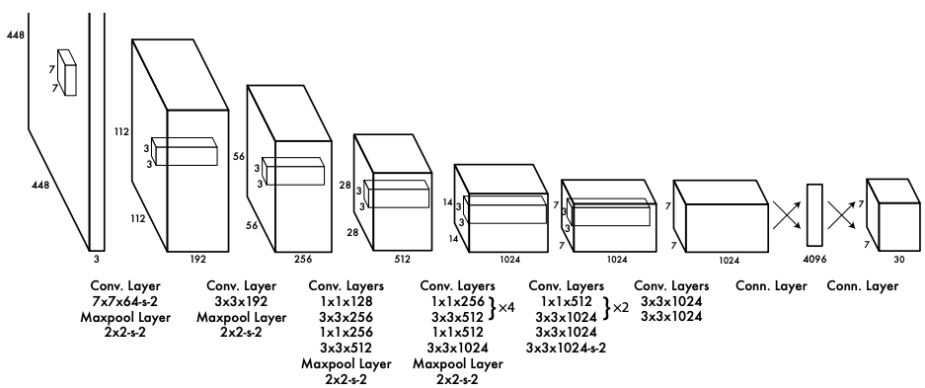


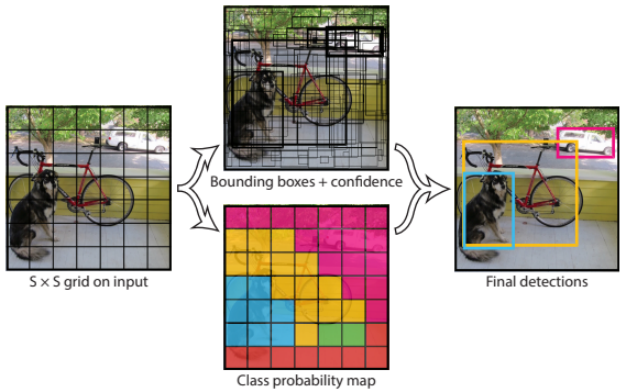
(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map



YOLO (You Only Look Once): A Real-time One-stage Object Detector

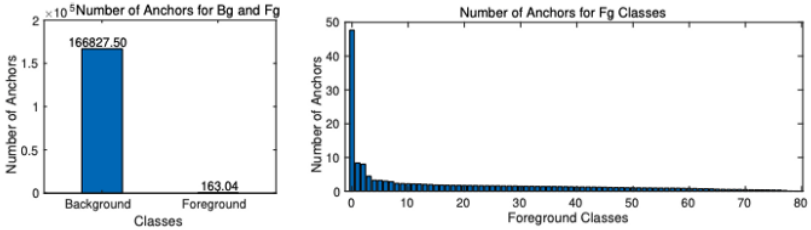
- The input image is converted into $7 \times 7 \times 1024$ feature maps. $S \times S$ cells (7×7 in the original paper) are used to estimate object detection boxes
- Each cell predicts multiple B ($B = 2$) bounding boxes and their confidence scores. If a cell covers a portion of an object, it will be potentially considered to be responsible for detecting the object during training
- At training time we only want one bounding box predictor to be responsible for each object. We assign one predictor to be "responsible" for predicting an object based on which prediction has the highest current IOU with the ground truth
- There is a series of YOLOv2, v3, v4 ... that further improve both accuracy and efficiency





Focal Loss

- Even with OHEM, training becomes inefficient as most of the samples are easy negatives which contribute no useful signal
- This problem is especially apparent for one-stage object detectors (SSD and YOLO)



- Below is the conventional binary cross entropy loss for detection

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

where p is the predicted foreground probability

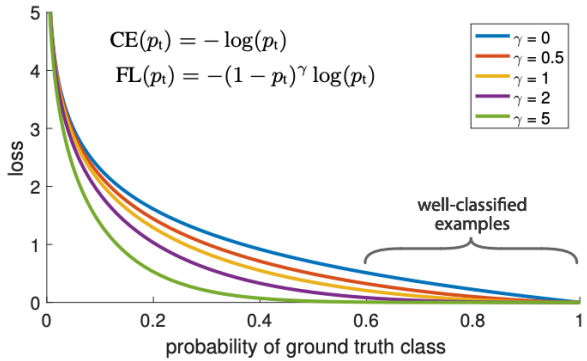
- The focal loss is modified to

$$FL(p) = \begin{cases} -\alpha(1-p)^{\gamma}\log(p), & y = 1 \\ -(1-\alpha)p^{\gamma}\log(1-p), & \text{otherwise} \end{cases}$$

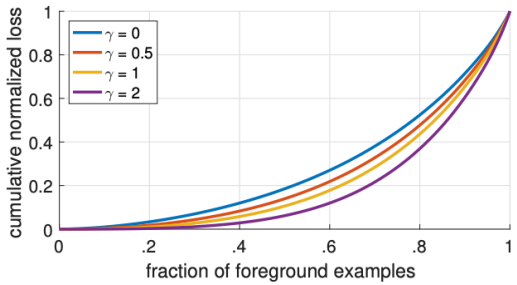
where $\alpha \in [0.25, 0.75]$ changes the detector's focus to more on positive or negative samples. $\alpha = 0.25$ is used in the original paper

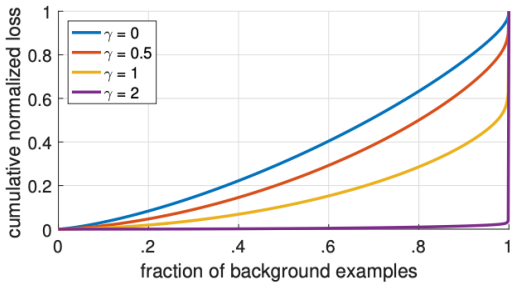
Focal Loss

- $\gamma > 1$ makes the loss penalizes more on the badly-classified samples



- Cumulative distribution functions for positive and negative samples are shown
- Approximately 20% of the hardest positive samples account for roughly half of the positive loss. As γ increases, more of the loss gets concentrated in the top 20% of examples, but the effect is minor
- As γ increases, substantially more weights start to concentrate on the hard negative examples

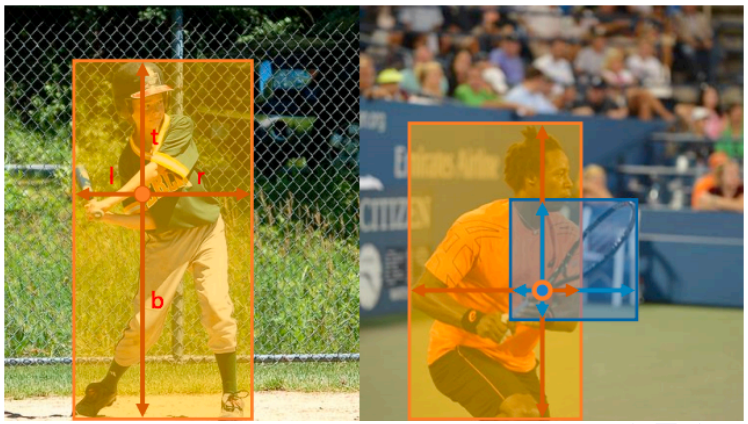




FCOS: Fully Convolutional One-Stage Object Detection

- Using anchors in modern detectors involves a lot of hyper-parameters. For instance, the number of anchors per section of the image, the ratio of dimensions of the boxes, the number of sections an image should be divided into, the IoU threshold for determine positives/negatives, etc.
- FCOS also utilizes FPN to generate multi-scale feature maps. Without using anchors, each pixel in the feature maps is trained to estimate the object box if it is within a ground-truth box
- For a point residing inside a ground-truth box, it predicts a 4D vector (l^*, t^*, r^*, b^*) encoding its location within the bounding box

$$l^* = x - x_0^{(i)}, \quad t^* = y - y_0^{(i)}$$
$$r^* = x_1^{(i)} - x, \quad b^* = y_1^{(i)} - y$$



FCOS: Fully Convolutional One-Stage Object Detection

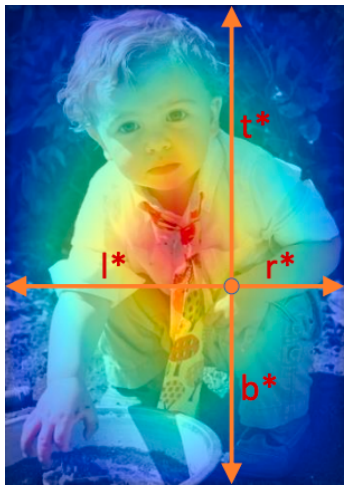
- FCOS also utilizes FPN backbone to generate multi-scale feature maps of five scales $P_3 - P_7$ (having strides of 8, 16, 32, 64, 128)
- Unlike anchor-based detectors, which assign anchor boxes with different sizes to different feature levels, it directly limits the range of bounding box regression for each level
- First compute the regression targets l^*, t^*, r^*, b^* for each location on all feature levels. If a location satisfies $\max(l^*, t^*, r^*, b^*) > m_i$ or $\max(l^*, t^*, r^*, b^*) < m_{i-1}$, it is set as a negative sample
- m_i is the maximum distance that feature level i needs to regress. $m_2, m_3, m_4, m_5, m_6, m_7$ are set as 0, 64, 128, 256, 512 and ∞, respectively. In this way, objects with different sizes are assigned to different feature levels
- If a location, even with multi-level prediction used, is still assigned to more than one ground-truth boxes, we simply choose the ground truth box with minimal area as its target

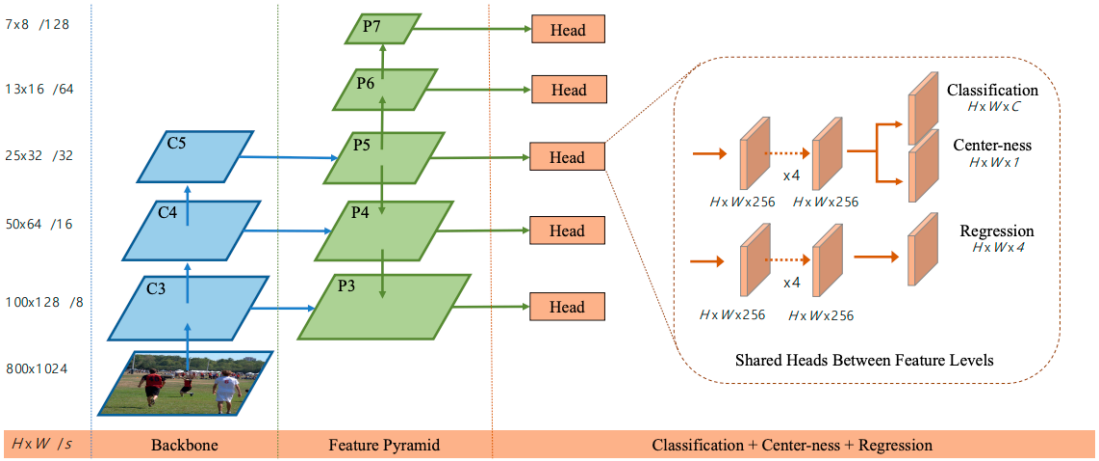
FCOS: Fully Convolutional One-Stage Object Detection

- It also has a head for predicting "centerness" of each location

$$\text{centerness}^* = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}}$$

- The FCOS system has three prediction heads





- **Inference.** Given an input image, we forward it through the network and obtain the classification scores and the regression prediction for each location on the feature maps. The original paper chooses location with confidence score > 0.05 as positive samples

Generalized Intersection over Union Loss for Box Regression

- IoU can be used to determine the overlap between a predicted box and a ground truth box
- However, if we directly treat the negative IoU as a loss, it has issues as well

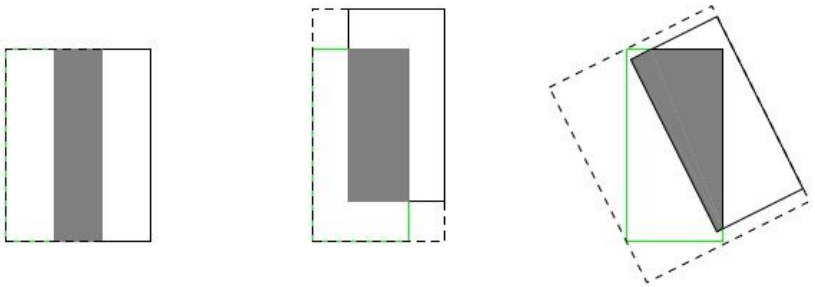


Figure: Left to right: IoU $= 0.33$ for all cases

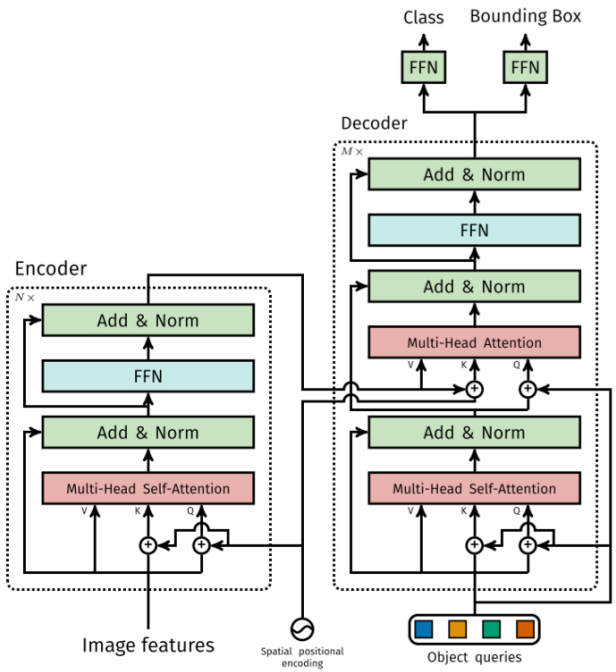
- Generalized IoU (GIoU) is defined as for two boxes A and B. Also define the smallest box C that encloses both A and B

$$\mathrm{GIoU} = \mathrm{IoU} - \frac{|C \backslash (A \cup B)|}{|C|}$$

- The GIoU loss is defined as $\mathcal{L}_{\mathrm{GIoU}} = 1 - \mathrm{GIoU}$ and the loss can be back-propagated to update box coordinates
- It works better on YOLOv3 than Faster R-CNN and Mask R-CNN, as the latter two have much denser anchor boxes and their regressions are much easier

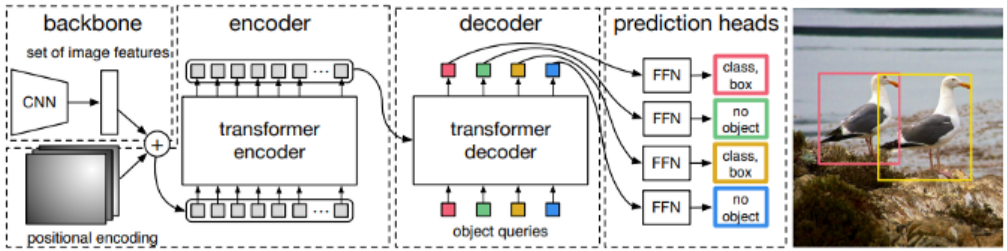
DETR: End-to-End Object Detection with Transformers

- The DEtection TRansformer (DETR) predicts all objects at once, and is trained end-to-end with a set loss function and abandons the manually specified positive sample assignment during training and non-maximum suppression during testing
- DETR predicts a fixed-size set of N predictions in a single pass. N is chosen to be significantly larger than the typical number of objects
- During training, the predictions are online assigned to different ground truth boxes with a bipartite matching



DETR: End-to-End Object Detection with Transformers

- **Backbone.** Given an input image $x_{\text{img}} \in \mathbb{R}^{H_0 \times W_0 \times 3}$, an CNN backbone obtains a visual feature map $z_0 \in \mathbb{R}^{H \times W \times C}$ ($H = H_0/32, W = W_0/32$ in the original paper), which is reshaped to $HW \times C$ feature sequence
- **Transformer encoder.** The $HW \times C$ features with additive positional encoding are fed into the Transformer Encoder to conduct self-attention for multiple times
- **Transformer decoder.** It takes N object query vectors as inputs and decodes them **in parallel**. The query vectors are initialized randomly and learned by back-propagation
- **Prediction FNN.** Given decoder's N feature vectors, the final predictions are computed by a 3-layer perceptron with ReLU activation function, and a linear projection layer. The head predicts the normalized center coordinates, height and width of the box w.r.t. the input image, and the class confidences
- As N is greater than the actual number of objects, an additional `background` ($\varnothing$) class is also introduced



DETR: End-to-End Object Detection with Transformers

- Let y be the ground truth set of objects, and $\hat{y} = \{\hat{y}_i\}_{i=1}^{N}$ the set of N predictions
- As N is generally larger than $|y|$, we pad y with $\varnothing$ to have size N
- We find a bipartite matching between the two sets by searching for a permutation $\sigma \in \mathfrak{S}_N$ of N elements with the lowest cost $\hat{\sigma} = \underset{\sigma \in \widetilde{S}_N}{\arg\min} \sum_i^N \mathcal{L}_{\text{match}}\left(y_i, \hat{y}_{\sigma(i)}\right)$ where $\mathcal{L}_{\text{match}}\left(y_i, \hat{y}_{\sigma(i)}\right)$is a pair-wise matching cost between ground truth y_i and a prediction with index $\sigma(i)$
- Each ground truth $y_i = (c_i, b_i)$ has a class label c_i and box coordinates $b_i \in [0, 1]^4$. For the prediction with index $\sigma(i)$, we define its class c_i probability as $\hat{p}_{\sigma(i)}\left(c_i\right)$
- The matching loss is defined as

$$\mathcal{L}_{\text{match}} = -\mathbf{1}_{\{c_i \neq \varnothing\}} \hat{p}_{\sigma(i)}\left(c_i\right) + \mathbf{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}\left(b_i, \hat{b}_{\sigma(i)}\right)$$

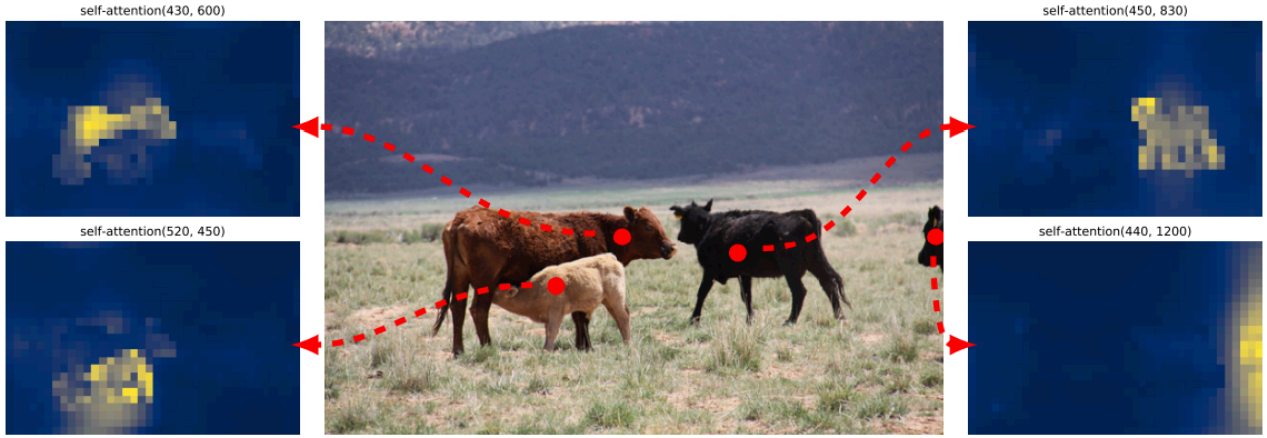
- The box loss is defined as

$$\lambda_{\text{iou}} \mathcal{L}_{\text{iou}}\left(b_i, \hat{b}_{\sigma(i)}\right) + \lambda_{\text{L1}} \left\| b_i - \hat{b}_{\sigma(i)} \right\|_1$$

  where $\mathcal{L}_{\text{iou}}$ is the generalized IoU loss
- This optimal assignment is computed efficiently with the Hungarian algorithm

Visualization of What did DETR Learn

- Visualization of self-attention in the Transformer encoder



- Visualization of box predictions from 20 prediction slots (object queries) in the DETR decoder

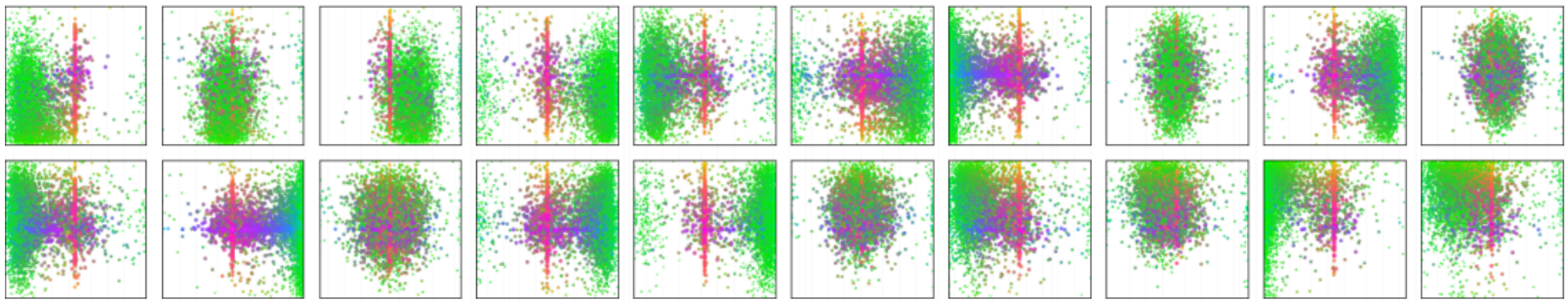


Figure: All box predictions in the COCO val set. Each box center is represented by one dot. Red, blue, and green dots represent large, medium, and small-scale object boxes.

References

- https://lilianweng.github.io/lil-log/2017/12/31/object-recognition-for-dummies-part-3.html
- Uijlings, J. R., Van De Sande, K. E., Gevers, T. and Smeulders, A. W., 2013. Selective search for object recognition. International journal of computer vision, 104(2), pp.154-171.
- Girshick, Ross, Jeff Donahue, Trevor Darrell, and Jitendra Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 580-587. 2014.
- Girshick, Ross. "Fast R-CNN." In Proceedings of the IEEE international conference on computer vision, pp. 1440-1448. 2015.
- Ren, Shaoqing, Kaiming He, Ross Girshick, and Jian Sun. "Faster R-CNN: Towards real-time object detection with region proposal networks." arXiv preprint arXiv:1506.01497 (2015).
- Shrivastava, Abhinav, Abhinav Gupta, and Ross Girshick. "Training region-based object detectors with online hard example mining." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 761-769. 2016.
- Liu, Wei, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. "SSD: Single shot multibox detector." In European conference on computer vision, pp. 21-37.

References

- Lin, Tsung-Yi, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. "Focal loss for dense object detection." In Proceedings of the IEEE international conference on computer vision, pp. 2980-2988. 2017.
- Redmon, Joseph, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You only look once: Unified, real-time object detection." In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779-788. 2016.
- Tian, Zhi, Chunhua Shen, Hao Chen, and Tong He. "FCOS: Fully convolutional one-stage object detection." In Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9627-9636. 2019.
- He, Kaiming, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mark R-CNN" In Proceedings of the IEEE international conference on computer vision, pp. 2961-2969. 2017.