Deep Learning for Computer Vision

CNNs for Object Detection: Single-Stage Methods

Vineeth N Balasubramanian

Department of Computer Science and Engineering
Indian Institute of Technology, Hyderabad

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

Exercise: Smooth L1 Loss

Why is Smooth L1 loss less sensitive to outliers than L2 loss?

If the deviation of predicted output from ground truth is very high, squaring the difference explodes the gradient. This can happen in L2 loss for outliers, and is mitigated in the Smooth L1 loss.

Recall: Contemporary Object Detection Methods

Region Proposal-based:

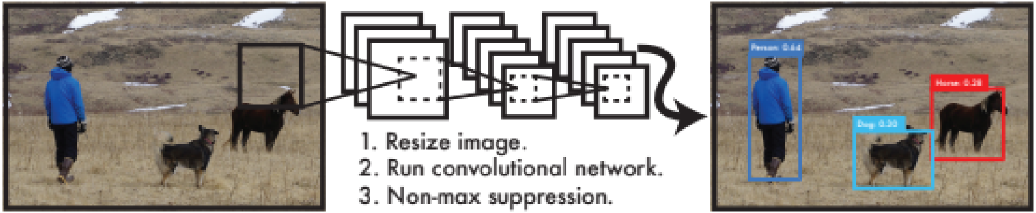
- Two-stage detection framework
- In the first stage, potential object regions are proposed (through methods such as Selective Search or Region Proposal Network, which we will see soon)
- In the second stage, a classifier processes the candidate regions
- More robust in performance but slower

Dense Sampling-based:

- One-stage detection framework
- Integrates region proposals and detection by acting on a dense sampling of possible locations
- Simple and fast but performance not as good as Region Proposal-based methods

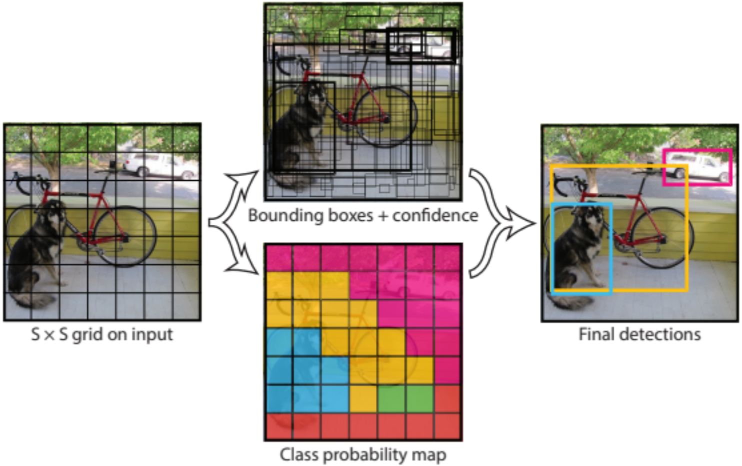
You Only Look Once (YOLO): v1[1]

- Single-stage detector based on Overfeat
- Speed with good performance the main aim



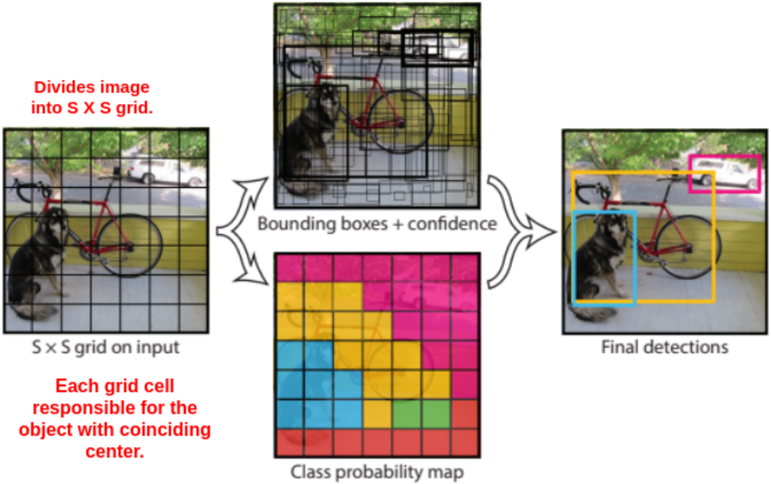
[1] Redmon et al, You Only Look Once: Unified, Real-Time Object Detection, CVPR 2016

YOLO v1: Unified Detection



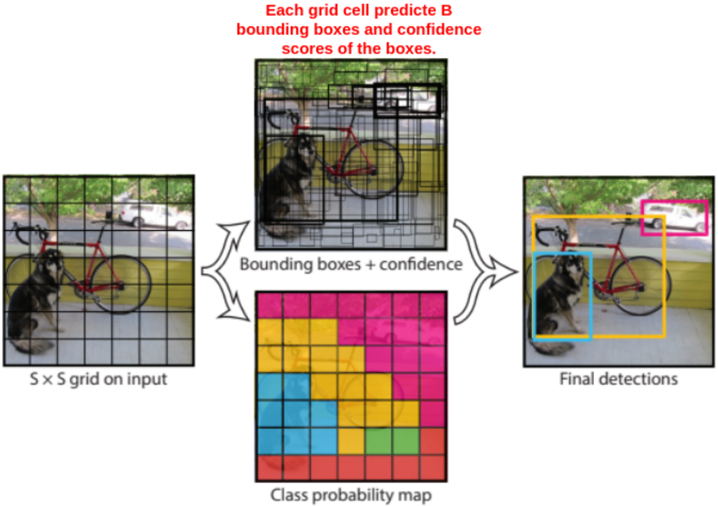
YOLO v1: Unified Detection

Divides image into S X S grid.

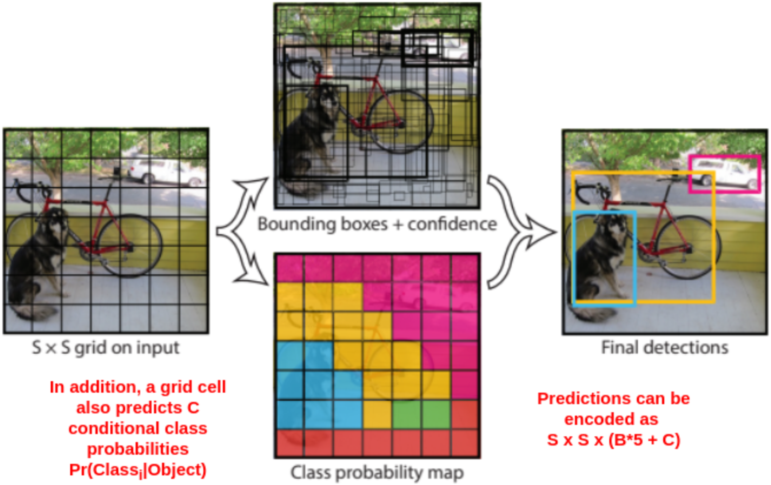


Each grid cell responsible for the object with coinciding center.

YOLO v1: Unified Detection



YOLO v1: Unified Detection



In addition, a grid cell also predicts C conditional class probabilities $Pr(Class_i|Object)$

Predictions can be encoded as $S \times S \times (B*5 + C)$

YOLO v1: Bounding Boxes and Confidence Scores

Bounding Boxes:

- Each bounding box gives 4 coordinates x, y, w, h
- (x, y) = Coordinates representing center of the object **relative to the grid cell**
- (w, h) = Width and height of the object **relative to the whole image**

YOLO v1: Bounding Boxes and Confidence Scores

Bounding Boxes:

- Each bounding box gives 4 coordinates x, y, w, h
- (x, y) = Coordinates representing center of the object **relative to the grid cell**
- (w, h) = Width and height of the object **relative to the whole image**

Confidence Score:

- Reflects how confident the model is that the box contains an object and also how accurate it thinks the box is
- Formally,

$$\text{Confidence} \; = Pr(\text{Object}) * IOU_{pred}^{truth}$$

YOLO v1: Conditional Class Probabilities

- Regardless of number of boxes B, we only predict one set of class probabilities per grid cell
- At test time, class-specific confidence scores for each box are given by:

$$Pr(Class_i|Object) * Pr(Object) * IOU_{pred}^{truth} = Pr(Class_i) * IOU_{pred}^{truth}$$

YOLO v1: Loss Function

Loss function used to train YOLO v1 given by:

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i}\right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i}\right)^2 \right] + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 + \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

where:

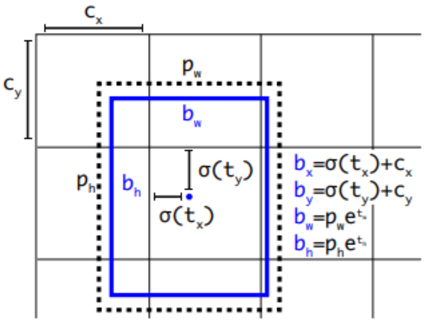
- $\mathbb{1}_{i}^{\text{obj}}$ denotes if object appears in cell i
- $\mathbb{1}_{ij}^{\text{obj}}$ denotes that j^{th} bounding box predictor in cell i is 'responsible' for that prediction

YOLO v1: Limitations

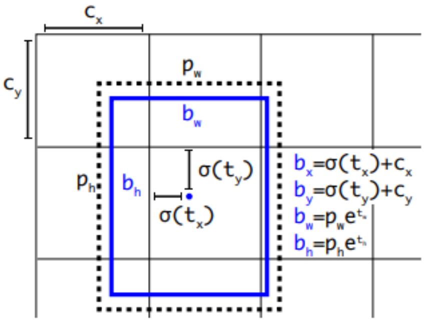
YOLO v1: Limitations

- Detects only a small number of objects
- Cannot detect objects that are small or close due to strong spatial constraints
- High localization error
- Relatively low recall

YOLO v2: Anchor Boxes

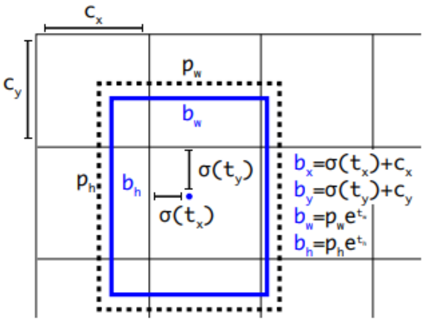


YOLO v2: Anchor Boxes



- Predicts 5 coordinates per anchor box: t_x, t_y, t_h, t_w, t_o

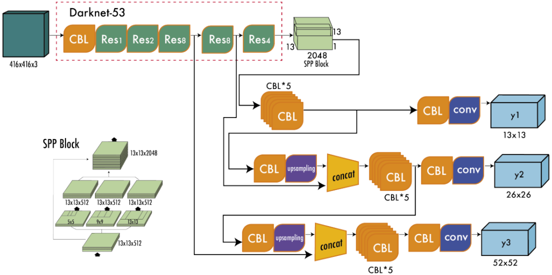
YOLO v2: Anchor Boxes



- Predicts 5 coordinates per anchor box: t_x, t_y, t_h, t_w, t_o
- If cell is offset from top left corner of image by (c_x, c_y) and bounding box has width and height p_w, p_h, then predictions correspond to:

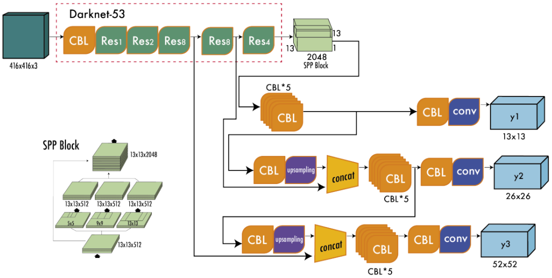
$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$
$$Pr(\text{object}) * IOU(b, \text{object}) = \sigma(t_o)$$

YOLOv3: Logistic Classifiers, Multi-scale



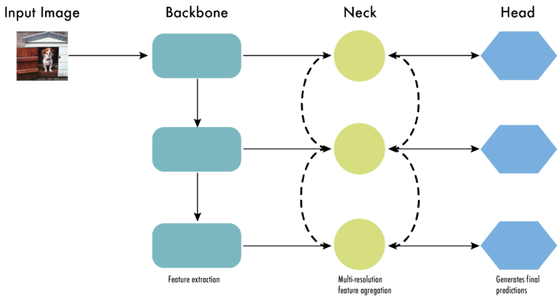
- **Bounding Box Prediction:**
  - Uses logistic regression to predict objectness score for each bounding box.
  - Assigns only one anchor box to each ground truth object.
- **Class Prediction:**
  - Utilizes binary cross-entropy for independent logistic classifiers.
  - Allows multiple labels for the same box in complex datasets.
- **New Backbone:** Darknet-53 with 53 convolutional layers and residual connections.

YOLOv3



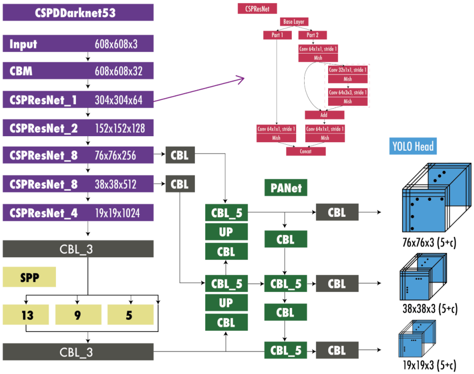
- **Multi-Scale Predictions:** Predicts three boxes at three different scales to improve small object detection.
- **Spatial Pyramid Pooling (SPP):**Adds a modified SPP block to the backbone, concatenating multiple max-pooling outputs with different kernel sizes.
- **Anchor Boxes:** YOLOv2 utilizes five anchor boxes per grid cell, whereas YOLOv3 employs three anchor boxes for each of the three different scales.

YOLO Architecture: Backbone, Neck, and Head



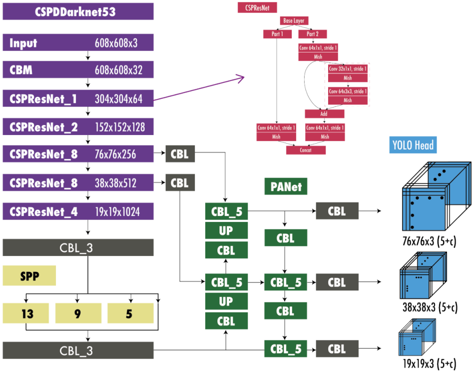
- **Backbone:**
  - Extracts features from the input image.
  - Typically a CNN trained on large-scale image classification tasks.
- **Neck:**
  - Aggregates and refines features.
  - Enhances spatial and semantic information across scales.
- **Head:**
  - Performs final predictions (e.g., classification, localization).
  - Includes post-processing like non-maximum suppression (NMS).

YOLOv4



- **Enhanced Architecture:**
  - Utilizes CSPDarknet53 with cross-stage partial connections (CSPNet) for improved computational efficiency
  - Employs Mish activation function $f(x) = x\tanh(\text{softplus}(x))$
- **Bag-of-Specials (BoS) Integration:**
  - Incorporates spatial pyramid pooling (SPP) and path aggregation network (PANet) to combine features across scales
  - Adds a modified spatial attention module (SAM) for better feature refinement

YOLOv4



- **Bag-of-Freebies (BoF) for Advanced Training:**
  - Mosaic augmentation combines four images into one, improving the model's ability to detect objects outside their usual context
  - DropBlock regularization replaces Dropout, enhancing regularization
- **Self-Adversarial Training (SAT):**
  - Introduces perturbations to deceive the model into thinking the ground truth object isn't present, improving robustness
- **Hyperparameter Optimization:**
  - Uses genetic algorithms to search for optimal hyperparameters during the first 10% of the training process and a cosine annealing scheduler to alter the learning rate during training

YOLO Versions (Part 1)[2]

YOLO Version	Year	Main Developments
YOLOv1	2016	Introduced as a single-stage detector; divides the image into a grid and predicts bounding boxes and class probabilities.
YOLOv2	2017	Added anchor boxes, improved training with high-resolution images, and introduced passthrough layers for small object detection. Introduced multiscale training.
YOLOv3	2018	Predicts classes for each bounding box instead of grids, uses multi-scale predictions at three different scales for improved accuracy.
YOLOv4	2020	Enhanced using techniques like CSPNet, PANet, and mish activation; focused on improving both speed and accuracy.
YOLOv5	2020	No paper was published; included improvements like auto-anchor and scaled YOLO versions.

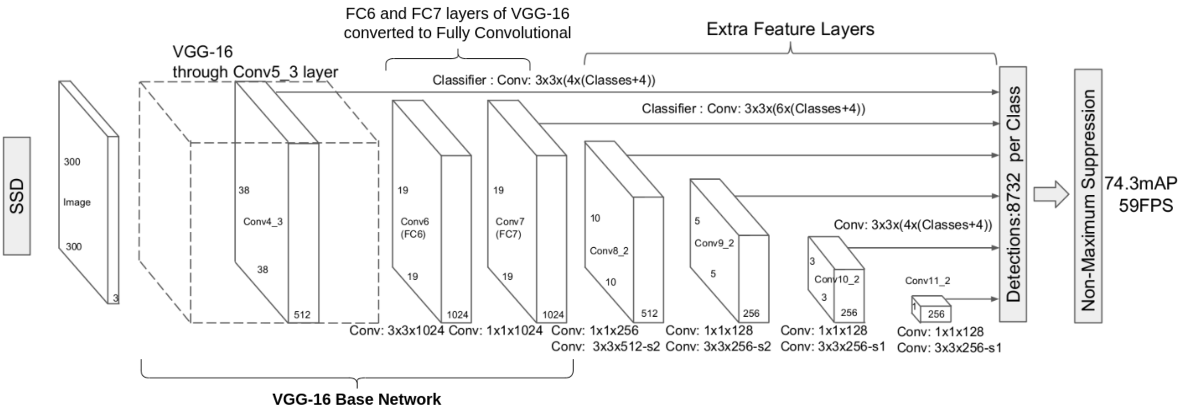
[2] Terven & Cordova-Esparza, A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS, arXiv 2024

YOLO Versions (Part 2)[3]

YOLO Version	Year	Main Developments
YOLOv6	2022	Introduced RepVGG-based backbone for better performance and used VariFocal loss for classification and SIoU/GIoU for regression
YOLOv7	2022	Introduced Extended Efficient Layer Aggregation Network and Model Scaling for concatenation-based models. In the bag-of-freebies, they included planned re-parameterized convolution and coarse label assignment for auxiliary head and fine label assignment for the lead head
YOLOv8	2023	Used Anchor-free Split Ultralytics Head and supports variety of pretrained models
YOLO-NAS	2023	Algorithm-generated model with quantization blocks that made the model faster and more accurate.
YOLOv9	2024	Introduced Information Bottleneck Principle, Reversible Functions, Programmable Gradient Information and Generalized Efficient Layer Aggregation Network
YOLOv10	2024	Introduced consistent dual label assignments for NMS-free training, a holistic efficiency-accuracy driven model design, which outperformed previous models with lower latency and fewer parameters.

[3] Terven & Cordova-Esparza, A Comprehensive Review of YOLO Architectures in Computer Vision: From YOLOv1 to YOLOv8 and YOLO-NAS, arXiv 2024

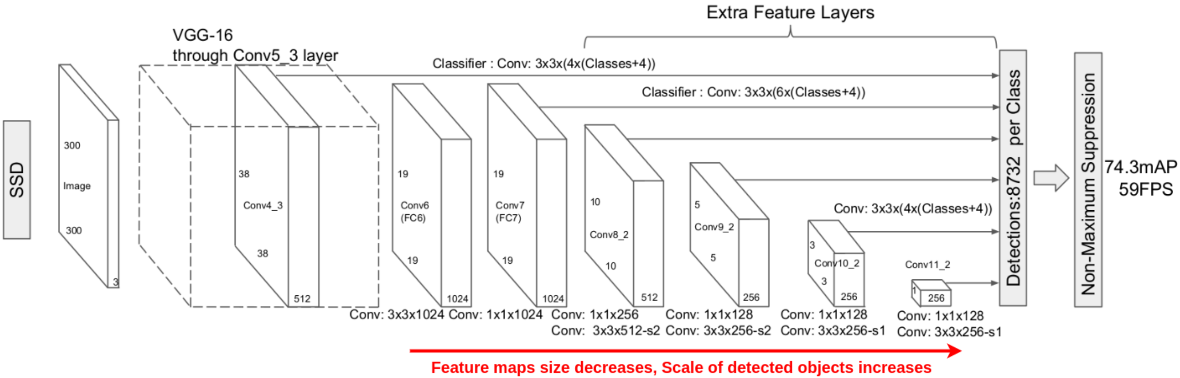
Single Shot MultiBox Detector (SSD)[4]



[4] Liu et al, SSD: Single Shot MultiBox Detector, ECCV 2016

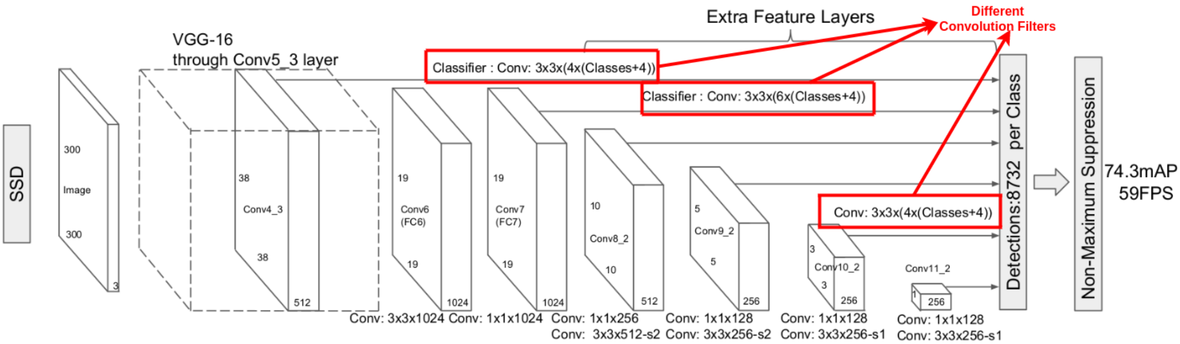
SSD: Model

Multi-scale Feature maps for Detection



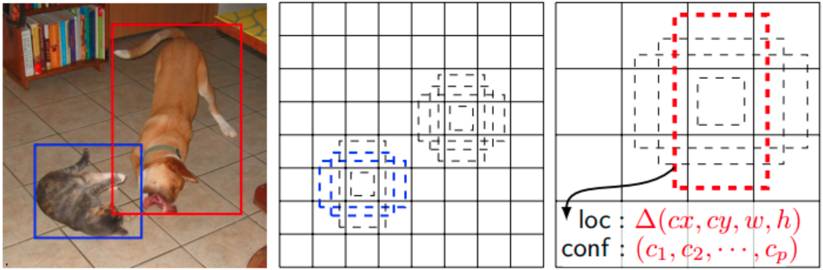
SSD: Model

Exclusive convolutional predictors for each feature map



A feature layer of size $m \times n$ with c channels gives $m \times n$ locations (grid cells); bounding box offsets output values relative to grid cell location (like in Faster R-CNN)

SSD: Anchor Boxes and Aspect Ratios



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

- For each of k default boxes with different aspect ratios, SSD predicts c class-specific scores and 4 anchor box offsets
- For an $m \times n$ feature map, there are $(c+4)kmn$ outputs

SSD: Loss Function

Weighted sum of localization loss (`loc`) and confidence loss (`conf`):

$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

where:

- x_{ij}^p is $\{1,0\}$ if i^{th} default box matches j^{th} ground truth box of category p
- $c =$ class probabilities
- $l, g =$ predicted and ground truth box parameters

SSD: Loss Function

- **Localization loss** L_{loc} given by:

$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^w = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^h = \log\left(\frac{g_j^h}{d_i^h}\right)$$

Regress offsets for center (cx, cy) of anchor box (d) and its width (w) and height (h)

SSD: Loss Function

- **Localization loss** L_{loc} given by:

$$L_{loc}(x,l,g) = \sum_{i \in Pos}^{N} \sum_{m \in \{cx,cy,w,h\}} x_{ij}^{k} \text{smooth}_{\text{L1}}(l_i^m - \hat{g}_j^m)$$

$$\hat{g}_j^{cx} = (g_j^{cx} - d_i^{cx})/d_i^w \qquad \hat{g}_j^{cy} = (g_j^{cy} - d_i^{cy})/d_i^h$$

$$\hat{g}_j^{w} = \log\left(\frac{g_j^w}{d_i^w}\right) \qquad \hat{g}_j^{h} = \log\left(\frac{g_j^h}{d_i^h}\right)$$

  Regress offsets for center (cx, cy) of anchor box (d) and its width (w) and height (h)

- **Confidence loss** L_{conf} is softmax loss of class confidences (probabilities):

$$L_{conf}(x,c) = -\sum_{i \in Pos}^{N} x_{ij}^{p} log(\hat{c}_i^p) - \sum_{i \in Neg} log(\hat{c}_i^0) \quad \text{where} \quad \hat{c}_i^p = \frac{\exp(c_i^p)}{\sum_p \exp(c_i^p)}$$

SSD: Practical Implementation

Hard Negative Mining:

- Similar to Faster R-CNN, most anchor boxes are negative
- To counter it, select negative examples that have highest confidence loss such that ratio between negatives and positives is $\sim 3:1$

SSD: Practical Implementation

Hard Negative Mining:

- Similar to Faster R-CNN, most anchor boxes are negative
- To counter it, select negative examples that have highest confidence loss such that ratio between negatives and positives is $\sim 3:1$

Data Augmentation: Training samples are obtained as below:

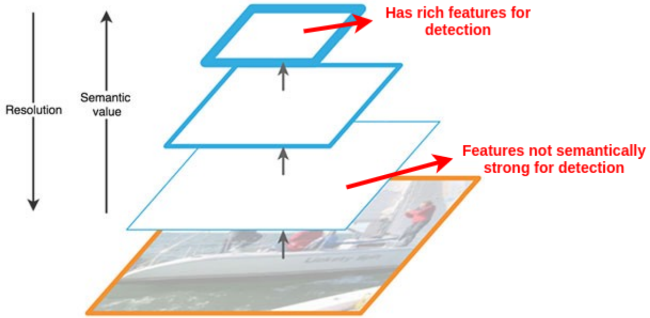
- Use original image
- Randomly sample a patch, such that minimum IoU with objects is in $\{0.1, 0.3, 0.5, 0.7, 0.9\}$

SSD: Performance

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

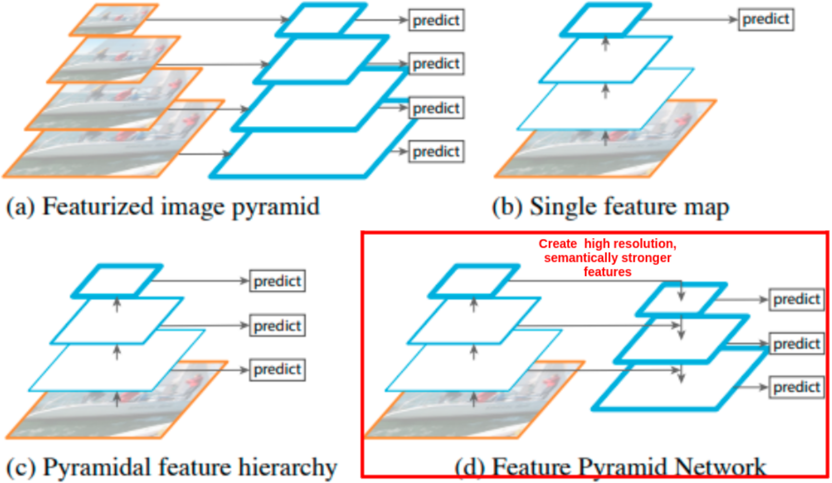
Feature Pyramid Network (FPN)[5]

- Feature maps from initial layers (which are high resolution) cannot be used for detection
- FPN provides a top-down pathway to construct higher resolution layers from a semantically rich layer

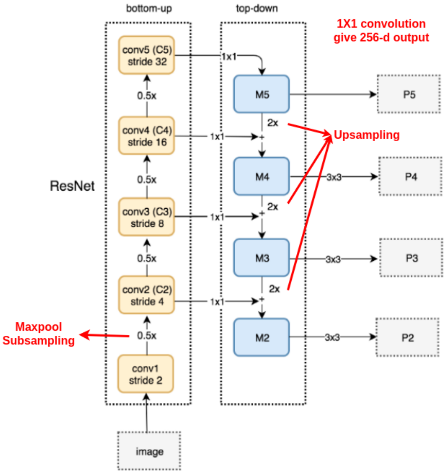


[5]Lin et al, Feature Pyramid Networks for Object Detection, CVPR 2017

Feature Pyramid Network

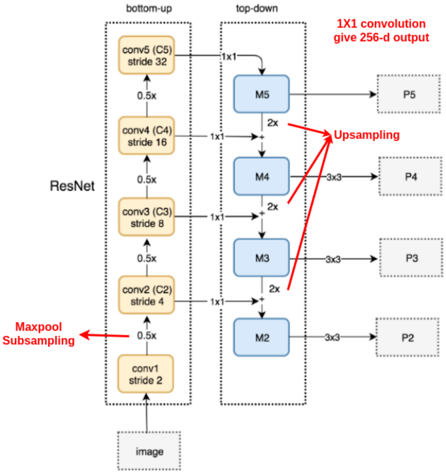


Feature Pyramid Network: Methodology



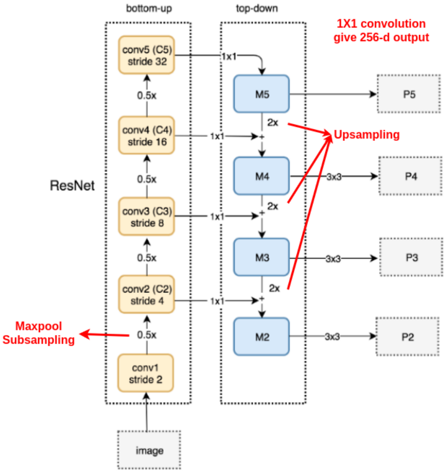
- All convolutional feature maps are treated with a 1×1 convolution with 256 channels

Feature Pyramid Network: Methodology



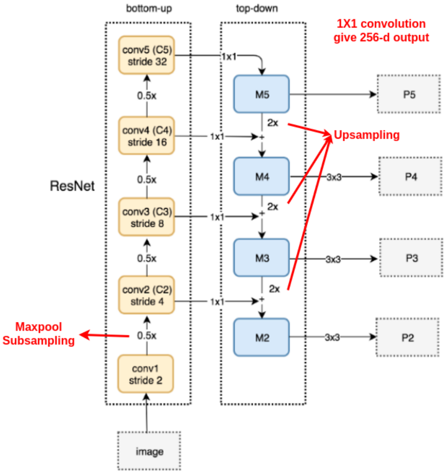
- All convolutional feature maps are treated with a 1×1 convolution with 256 channels
- M5 is upsampled by a factor of 2 (Maxpool at $1/2 \times 1/2$)

Feature Pyramid Network: Methodology



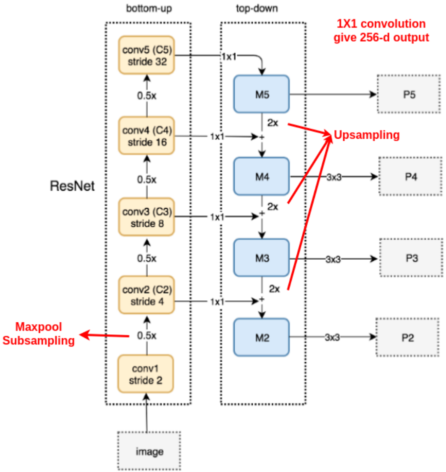
- All convolutional feature maps are treated with a 1×1 convolution with 256 channels
- M5 is upsampled by a factor of 2 (Maxpool at $1/2 \times 1/2$)
- M5 and C4 signals are element-wise added to give M4; similarly followed in the downward direction

Feature Pyramid Network: Methodology



- All convolutional feature maps are treated with a 1×1 convolution with 256 channels
- M5 is upsampled by a factor of 2 (Maxpool at $1/2 \times 1/2$)
- M5 and C4 signals are element-wise added to give M4; similarly followed in the downward direction
- 3×3 convolution is used to reduce aliasing effect of Ms

Feature Pyramid Network: Methodology

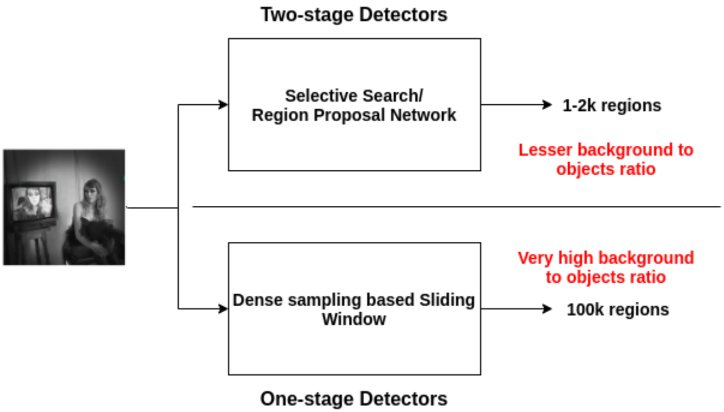


Credit: Jonathan Hui, Medium.com

- All convolutional feature maps are treated with a 1×1 convolution with 256 channels
- M5 is upsampled by a factor of 2 (Maxpool at $1/2 \times 1/2$)
- M5 and C4 signals are element-wise added to give M4; similarly followed in the downward direction
- 3×3 convolution is used to reduce aliasing effect of Ms
- Finally, Ps are individually fed into exclusive object detectors

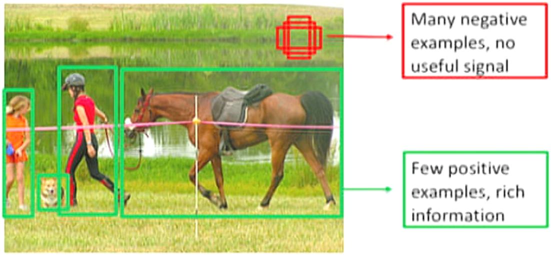
RetinaNet[6]

Intuition: Two-stage detectors are more accurate than one-stage detectors due to lesser class imbalance between background (negative) and object-containing (positive) proposals

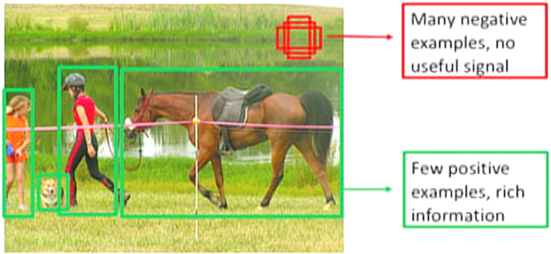


[6] Lin et al, Focal Loss for Dense Object Detection, ICCV 2017

Class Imbalance in Object Detection: The Problems

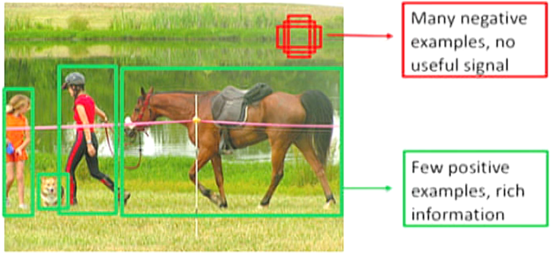


Class Imbalance in Object Detection: The Problems



- Training is inefficient as easy negatives contribute no useful signal

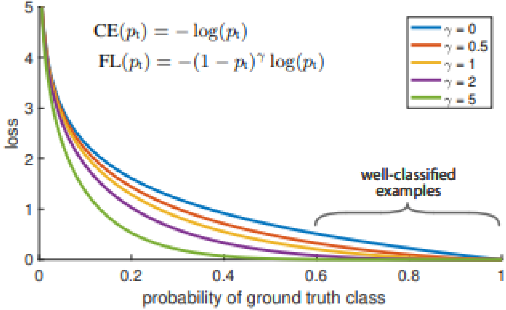
Class Imbalance in Object Detection: The Problems



- Training is inefficient as easy negatives contribute no useful signal
- Loss due to easy negatives overwhelms loss due to positives and thereby, training process can lead to degenerate models; hard negative training alleviates it to some extent

Credit: Sik-Ho Tsang, TowardsDataScience.com

CE Loss is Bad[7]



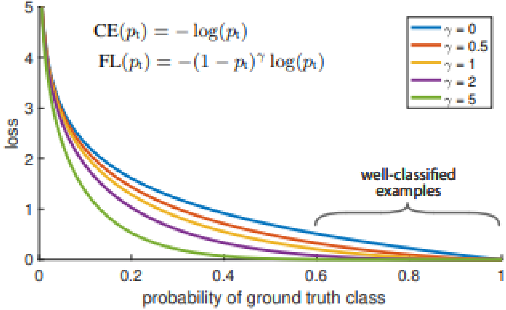
- CE Loss for binary classification is typically implemented as:

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

- Can be rewritten as
$CE(p, y) = CE(p) = -\log(p_t)$

[7]Lin et al, Focal Loss for Dense Object Detection, ICCV 2017

CE Loss is Bad[7]



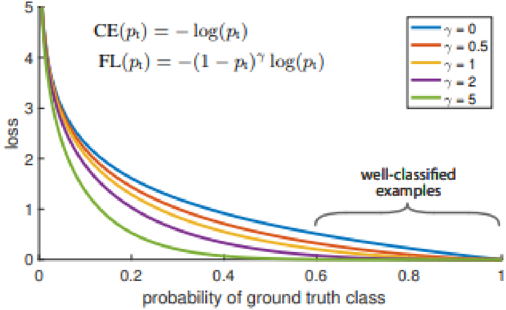
- CE Loss for binary classification is typically implemented as:

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

- Can be rewritten as $CE(p, y) = CE(p) = -\log(p_t)$
- Can be observed in graph ($\gamma = 0$ curve) that easily classifiable examples ($p >> .5$) incur loss of non-trivial magnitude

[7]Lin et al, Focal Loss for Dense Object Detection, ICCV 2017

RetinaNet: Balanced Cross Entropy and Focal Loss

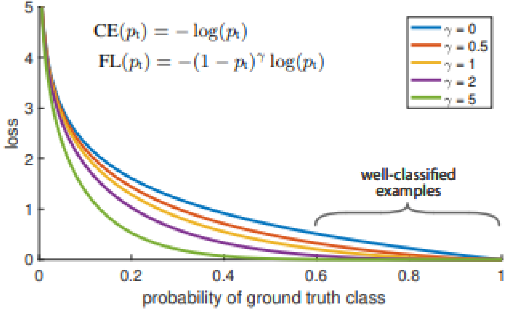


Balanced Cross Entropy:

- Introduces a weighting factor $\alpha \in [0, 1]$ which can be inverse class frequency or a hyperparameter
- α-balanced CE loss is given by:

$$CE(p_t) = -\alpha \log(p_t)$$

RetinaNet: Balanced Cross Entropy and Focal Loss



Balanced Cross Entropy:

- Introduces a weighting factor $\alpha \in [0, 1]$ which can be inverse class frequency or a hyperparameter
- α-balanced CE loss is given by:

$$CE(p_t) = -\alpha \log(p_t)$$

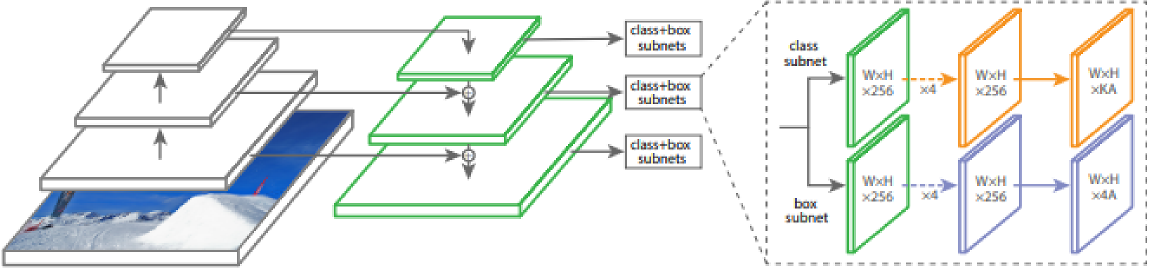
Focal Loss:

- Adds a modulating factor $(1 - p_t)^\gamma$ to CE loss, with tunable focusing parameter $\gamma \geq 0$.
- Focal loss is given by:

$$FL(p_t) = -(1 - p_t)^\gamma \log(p_t)$$

RetinaNet: Architecture

FPN + Focal Loss



(a) ResNet (b) feature pyramid net (c) class subnet (top) (d) box subnet (bottom)

Detectron

- Framework developed by Facebook AI Research (FAIR) to implement state-of-the-art object detection algorithms
- Highly flexible providing support across various algorithms and backbone networks
- See Detectron and Detectron2 for details

Homework

Readings

- Object Detection for Dummies, Part 4
- YOLO Family: All you want to know
- Understanding SSD
- Understanding FPN
- Understanding RetinaNet

Homework

Exercises

- YOLO9000 and YOLOv3 were follow-ups of YOLOv2. What was different in these extensions? Find out! (*Hint:* see the YOLO Family: All you want to know link)
- Given two bounding boxes in an image: an upper-left box which is 2×2, and a lower-right box which is 2×3 and an overlapping region of 1×1, what is the IoU between the two boxes?
- Consider using YOLO object detector on a 19×19 grid, on a detection problem with 20 classes, and with 5 anchor boxes. During training, for each image, you will need to construct an output volume y as the target value for the neural network; this corresponds to the last layer of the neural network. (y may include background). What is the dimension of this output volume?

References

[1] Wei Liu et al. "Ssd: Single shot multibox detector". In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[2] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[3] Tsung-Yi Lin et al. "Feature pyramid networks for object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 2117–2125.

[4] Tsung-Yi Lin et al. "Focal loss for dense object detection". In: *Proceedings of the IEEE international conference on computer vision*. 2017, pp. 2980–2988.

[5] Joseph Redmon and Ali Farhadi. "YOLO9000: better, faster, stronger". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 7263–7271.