**Name : Panneerselvam N**
**Role : Research Intern – Inspect**
**Date : 02/06/2021**

# You Only Look Once V2

Yolo v2 is an improved version of Yolo v1, Which optimizes the limitations of Yolo v1.

## Optimizations:

### Batch Normalization :

- Batch Normalization is one of the Regularization technique which is used for avoid over fitting in deep neural networks.

- In Yolo v2 , Batch Normalization is added instead of Dropout.Which gives +2% improvements in mAP.

### Higher Resolution Classifier :

- Firstly, the CNN layers are fine-tuned with 448x448 size in ImageNet dataset for first 10 epochs. After ,the all CNN layers are fine-tuned with object detection datasets.

- This operation give +4% mAP .

### Convolutions with Anchor Boxes:

- In Yolo v1 , there is an limitation that it predicts only one objects in one grid cell. It doesn't predicts multiple objects with respect one grid cell .

- Yolo v2 Comes with Anchor boxes for overcome above problem. Anchor box means it is a predefined bounding boxes with different height and width ratios.

- In Yolo v2 architecture , Convolutional layers used instead of fully connected layers.

### Dimension Clusters :

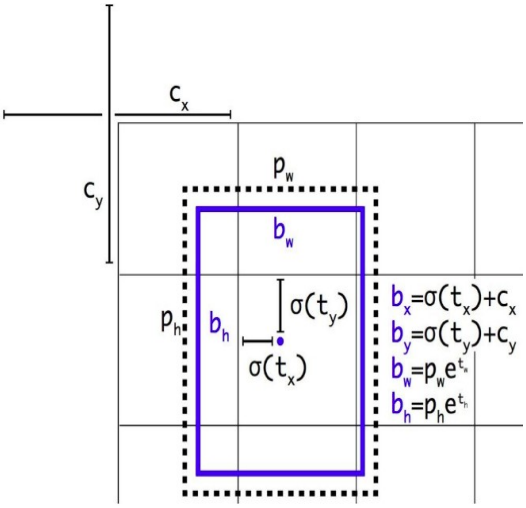
- IOU based K-means clusters are used to predict good number of bounding boxes and sizes for datasets.

- Initially the datasets are trained with k-means clusters , by based on this cluster output the number and size Anchor boxes are predicted . In research paper , Authors tried many K values , but K=5 gives good mAP.

| | Box Generation | # | Avg IOU |
|---|---|---|---|
| Euclidean Distance based | Cluster SSE | 5 | 58.7 |
| IOU based | Cluster IOU | 5 | 61.0 |
| Faster R-CNN | Anchor Boxes [15] | 9 | 60.9 |
| IOU based | Cluster IOU | 9 | 67.2 |

**Direct Location Prediction:**
In Yolo v1, initial bounding box prediction is unstable because of there is no constraints , after some iterations only it will go to start convergence. And also bounding boxes are randomly predicted.

Yolo v2 comes with Direct bounding Location prediction for overcome above problem.




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

where

$t_x, t_y, t_w, t_h$ are predictions made by YOLO.

$c_x, c_y$ is the top left corner of the grid cell of the anchor.

$p_w$, $p_h$ are the width and height of the anchor.

$c_x, c_y, p_w, p_h$ are normalized by the image width and height.

$b_x, b_y, b_w, b_h$ are the predicted boundary box.

$\sigma(t_o)$ is the box confidence score.

**Steps for Good Bounding Box Prediction :**

- Form thousands of candidate anchor boxes around the image
- For each anchor box predict some offset from that box as a candidate box
- Calculate a loss function based on the ground truth example
- Calculate a probability that a given offset box overlaps with a real object
- If that probability is greater than 0.5, factor the prediction into the loss function
- By rewarding and penalizing predicted boxes slowly pull the model towards only localizing true objects

**Fine-Grained Features :**
To detect small objects well, the 26×26×512 feature maps from earlier layer is mapped into 13×13×2048 feature map, then concatenated with the original 13×13 feature maps for detection.

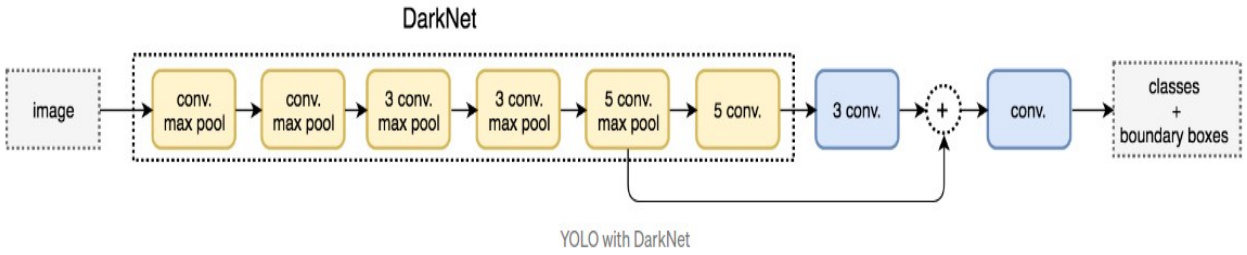
**Multi-scale training :**
Multi scale method is very useful for extract very small and large object features.

Yolo v2 rescale the training image for every 10 epochs randomly.

**Summary of Optimization Improvements :**

| | 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** |

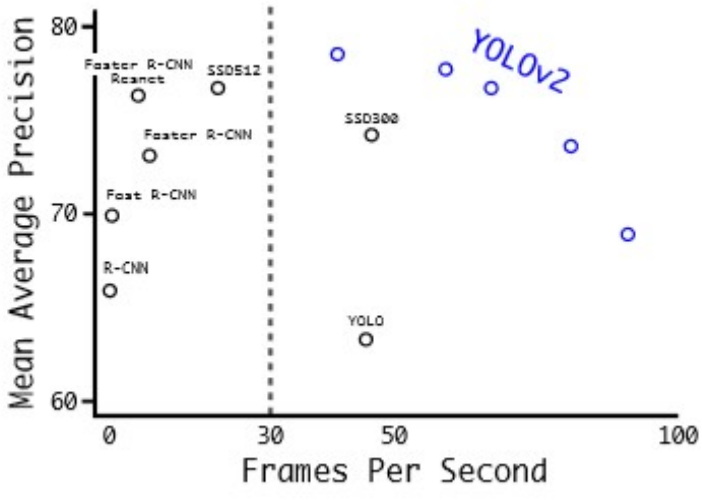
**Yolo v2 Architecture :**




Yolo v2 uses Darknet-19 as it's CNN architecture , which contains 19 Convolutional layers for feature extraction.

Output part contains 11 convolutinal layers ,First Three is 3x3 filters with 1024 filters followed by one 1x1 filters with 125 output channels(5 box predictions each with 25 parameters).

**Performance :**



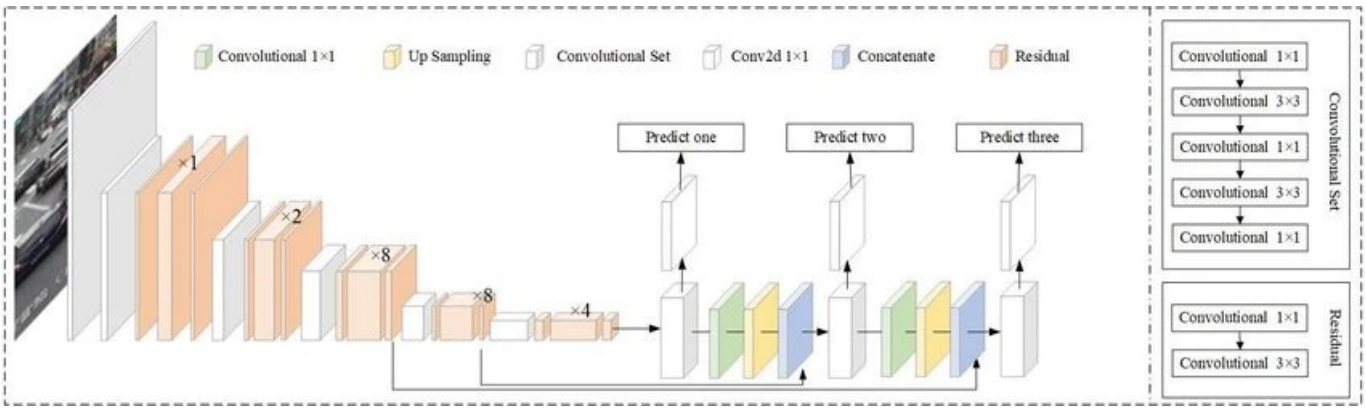
# YOLO V3

Yolo v3 is improved version of yolo v2 with high accuracy.

**Main Optimizations Of Yolo v3:**

- Three scaled outputs
- Feature Pyramid Network
- ResNet

**Architecture :**




**Explanation :**

- Yolo v3 uses Darknet-53 as CNN for extracting important features. This Darknet contains 53 layers.

- In Yolo v3 , Detection layer also contains 53 Convolutional layers. Totally ,106 layers.

- Yolov3 uses Residual Networks (1x1 and 3x3) to avoid Gradient exploding and vanishing problem.

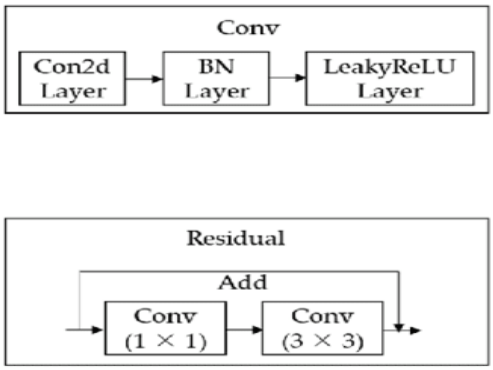
- In Yolo v3, the detection is done by applying 1 x 1 detection kernels on feature maps of three different sizes at three different places in the network.

- The shape of detection kernel is 1 x 1 x (B x (5 + C)). Here B is the number of bounding boxes a cell on the feature map can predict, '5' is for the 4 bounding box attributes and one object confidence and C is the no. of classes.

- YOLO v3 uses binary cross-entropy for calculating the classification loss for each label while object confidence and class predictions are predicted through logistic regression.

- No Pooling layer is used in Yolo v3.
- Features Pyramid Network(FPN) is used to Up-sampling image.

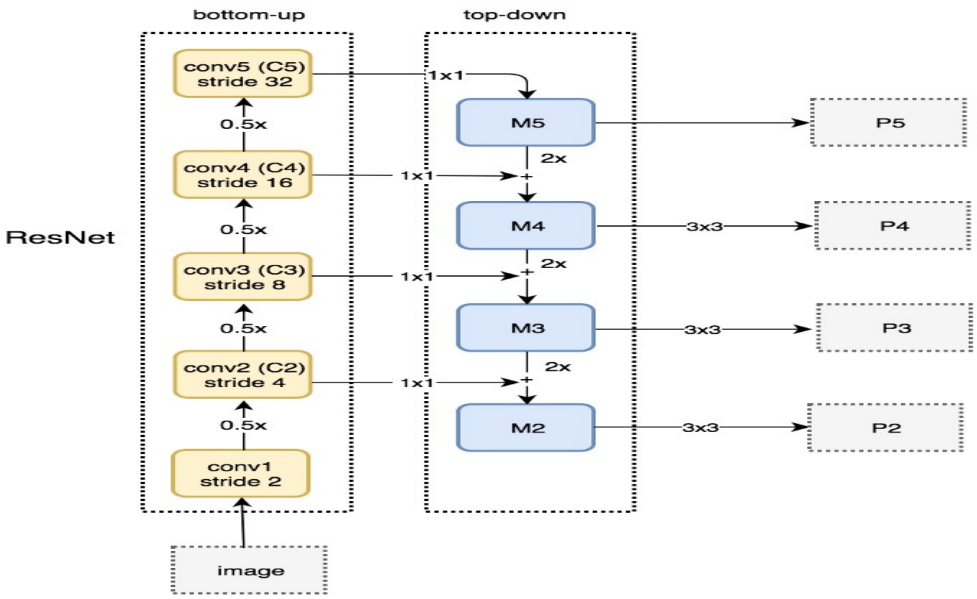
| Layer | Filters size | Repeat | Output size |
|---|---|---|---|
| Image | | | 416 × 416 |
| Conv | 32 3 × 3/1 | 1 | 416 × 416 |
| Conv | 64 3 × 3/2 | 1 | 208 × 208 |
| Conv<br>Conv<br>Residual | 32 1 × 1/1<br>64 3 × 3/1 | Conv<br>Conv<br>Residual × 1 | 208 × 208<br>208 × 208<br>208 × 208 |
| Conv | 128 3 × 3/2 | 1 | 104 × 104 |
| Conv<br>Conv<br>Residual | 64 1 × 1/1<br>128 3 × 3/1 | Conv<br>Conv<br>Residual × 2 | 104 × 104<br>104 × 104<br>104 × 104 |
| Conv | 256 3 × 3/2 | 1 | 52 × 52 |
| Conv<br>Conv<br>Residual | 128 1 × 1/1<br>256 3 × 3/1 | Conv<br>Conv<br>Residual × 8 | 52 × 52<br>52 × 52<br>52 × 52 |
| Conv | 512 3 × 3/2 | 1 | 26 × 26 |
| Conv<br>Conv<br>Residual | 256 1 × 1/1<br>512 3 × 3/1 | Conv<br>Conv<br>Residual × 8 | 26 × 26<br>26 × 26<br>26 × 26 |
| Conv | 1024 3 × 3/2 | 1 | 13 × 13 |
| Conv<br>Conv<br>Residual | 512 1 × 1/1<br>1024 3 × 3/1 | Conv<br>Conv<br>Residual × 4 | 13 × 13<br>13 × 13<br>13 × 13 |




Above Figure shows Darknet-53 Architecture

**Up-sampling :**

After 82th and 94th covolutional layers the image is Up-sample to predict medium an low level objects.




Above image shows the Architecture of FPN(Feature Pyramid Network)

**Working :**

- First input image is given to bottom up network, each blocks is Residual CNN block . Input image is feed forwarded through thid Network. The every block downn samples the input image by 2.

- This Bottom Up networks ouputs are shared to top down block by convolutional layer 1x1.

- In Top-Down block, each shared layer is added with previous layer of Bottom-up block's output. It increase the size by 2.

- Output of Top-Down blocks are perform Convolution operation with 3x3 kernels. Finally different sizes of Image pyramid is produced(P2,P3,P4,P5).

**Improvements:**
The average precision for small objects improved, it is now better than Faster RCNN but Retinanet is still better in this.
1.As MAP increased localization errors decreased.
2.Predictions at different scales or aspect ratios for same object improved because of the addition of feature pyramid like method.

**Performance :**
YOLOv3 gives a MAP of 57.9 on COCO dataset for IOU 0.5 and the table below shows the comparisons:

| | backbone | AP | $AP_{50}$ | $AP_{75}$ | $AP_S$ | $AP_M$ | $AP_L$ |
|---|---|---|---|---|---|---|---|
| *Two-stage methods* | | | | | | | |
| Faster R-CNN+++ [3] | ResNet-101-C4 | 34.9 | 55.7 | 37.4 | 15.6 | 38.7 | 50.9 |
| Faster R-CNN w FPN [6] | ResNet-101-FPN | 36.2 | 59.1 | 39.0 | 18.2 | 39.0 | 48.2 |
| Faster R-CNN by G-RMI [4] | Inception-ResNet-v2 [19] | 34.7 | 55.5 | 36.7 | 13.5 | 38.1 | 52.0 |
| Faster R-CNN w TDM [18] | Inception-ResNet-v2-TDM | 36.8 | 57.7 | 39.2 | 16.2 | 39.8 | **52.1** |
| *One-stage methods* | | | | | | | |
| YOLOv2 [13] | DarkNet-19 [13] | 21.6 | 44.0 | 19.2 | 5.0 | 22.4 | 35.5 |
| SSD513 [9, 3] | ResNet-101-SSD | 31.2 | 50.4 | 33.3 | 10.2 | 34.5 | 49.8 |
| DSSD513 [2] | ResNet-101-DSSD | 33.2 | 53.3 | 35.2 | 13.0 | 35.4 | 51.1 |
| RetinaNet [7] | ResNet-101-FPN | 39.1 | 59.1 | 42.3 | 21.8 | 42.7 | 50.2 |
| RetinaNet [7] | ResNeXt-101-FPN | **40.8** | **61.1** | **44.1** | **24.1** | **44.2** | 51.2 |
| YOLOv3 608 × 608 | Darknet-53 | 33.0 | 57.9 | 34.4 | 18.3 | 35.4 | 41.9 |