**STUDY ON YOLOv2 ARCHITECTURE:**

**YOLO v2:**

1. YOLO has higher localization errors and the recall (measure how good to locate all objects) is lower. YOLOv2 is the second version of the YOLO with the objective of improving the accuracy significantly while making it faster.
2. YOLOv2, or YOLO9000, is a single-stage real-time object detection model. It improves upon YOLOv1 in several ways, including the use of Darknet-19 as a backbone, batch normalization, use of a high-resolution classifier, and the use of anchor boxes to predict bounding boxes, and more.
3. YOLOv2, is state-of-the-art on standard detection tasks like PASCAL VOC and COCO. Using a novel, multi-scale training method the same YOLOv2 model can run at varying sizes, offering an easy tradeoff between speed and accuracy.
4. It predicts detections for more than 9000 different object categories.

Ref.: Paper: <https://arxiv.org/pdf/1612.08242.pdf>

**YOLOv2 can detect wide variety of objects:**

1. Current object detection datasets are limited compared to datasets for other tasks like classification and tagging. The most common detection datasets contain thousands to hundreds of thousands of images with dozens to hundreds of tags. Classification datasets have millions of images with tens or hundreds of thousands of categories. However, labelling images for detection is far more expensive than labelling for classification or tagging.
2. To harness the large amount of classification data and use it to expand the scope of current detection systems, this method uses a hierarchical view of object classification that allows to combine distinct datasets together.
3. YOLOv2 proposes a joint training algorithm that allows us to train object detectors on both detection and classification data. Our method leverages labelled detection images to learn to precisely localize objects while it uses classification images to increase its vocabulary and robustness.
4. YOLOv2 uses dataset combination method and joint training algorithm to train a model on more than 9000 classes from ImageNet as well as detection data from COCO.

**YOLOv2 is better:**

1. YOLO suffers from a variety of shortcomings relative to state-of-the-art detection systems. Error analysis of YOLO compared to Fast R-CNN shows that YOLO makes a significant number of localization errors. Furthermore, YOLO has relatively low recall compared to region proposal-based methods. Thus we focus mainly on improving recall and localization while maintaining classification accuracy.
2. With YOLOv2, we want a more accurate detector that is still fast. Instead of scaling up the network, they have simplified the network to make the representation easier to learn.

**Batch Normalization:**

1. By adding batch normalization on all of the convolutional layers in YOLO we get more than 2% improvement in mAP.
2. Batch normalization also helps regularize the model.
3. With batch normalization we can remove dropout from the model without overfitting.
4. Batch Normalization is a technique that mitigates the effect of unstable gradients within deep neural networks. BN introduces an additional layer to the neural network that performs operations on the inputs from the previous layer.
5. The operation standardizes and normalizes the input values. The input values are then transformed through scaling and shifting operations.
6. Benefits of Batch Normalization:
   1. Inclusion of Batch Normalization technique in deep neural networks improves training time
   2. BN enables the utilization of larger learning rates, this shortness the time of convergence when training neural networks
   3. Reduces the common problem of vanishing gradients
   4. Covariate shift within neural network is reduced

Ref.: <https://www.kdnuggets.com/2020/08/batch-normalization-deep-neural-networks.html>

**High Resolution Classifier:**

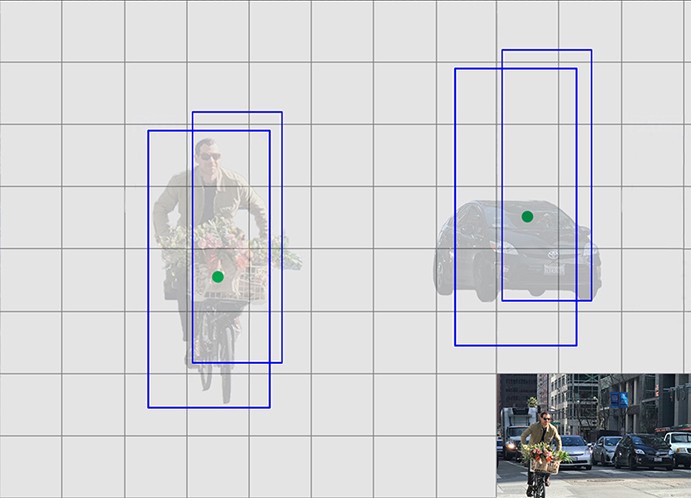
1. YOLO trains the classifier network at 224 × 224 and increases the resolution to 448 for detection. This means the network has to simultaneously switch to learning object detection and adjust to the new input resolution.
2. For YOLOv2 they first fine tune the classification network at the full 448 × 448 resolution for 10 epochs on ImageNet. This gives the network time to adjust its filters to work better on higher resolution input. Then fine tuning on the resulting network for detection is performed. This high resolution classification network gives us an increase of almost 4% mAP.

**Convolutional with Anchor boxes:**

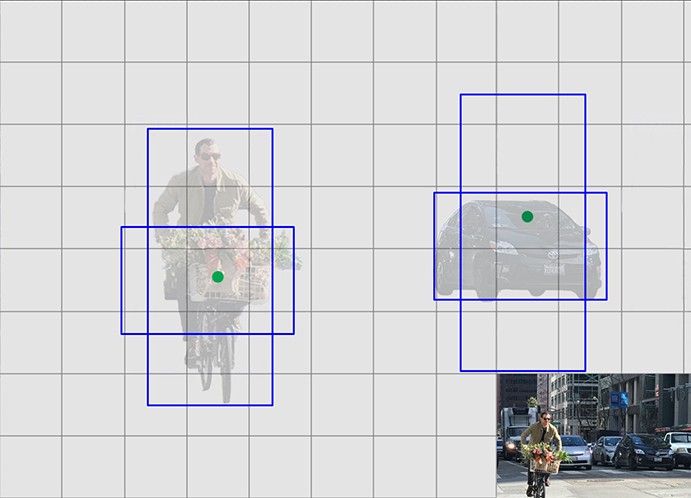
1. YOLO predicts the coordinates of bounding boxes directly using fully connected layers on top of the convolutional feature extractor. Instead of predicting coordinates directly Faster R- CNN predicts bounding boxes using hand-picked priors.
2. We remove the fully connected layers from YOLO and use anchor boxes to predict bounding boxes. First we eliminate one pooling layer to make the output of the network’s convolutional layers higher resolution. We also shrink the network to operate on 416 input images instead of 448×448.
3. We do this because we want an odd number of locations in our feature map so there is a single center cell. Objects, especially large objects, tend to occupy the center of the image so it’s good to have a single location right at the center to predict these objects instead of four locations that are all nearby.
4. YOLO’s convolutional layers downsample the image by a factor of 32 so by using an input image of 416 we get an output feature map of 13 × 13.
5. When we move to anchor boxes we also decouple the class prediction mechanism from the spatial location and instead predict class and objectness for every anchor box.
6. The objectness prediction still predicts the IOU of the ground truth and the proposed box and the class predictions predict the conditional probability of that class given that there is an object.
7. Using anchor boxes we get a small decrease in accuracy. YOLO only predicts 98 boxes per image but with anchor boxes our model predicts more than a thousand.
8. Without anchor boxes our intermediate model gets 69.5 mAP with a recall of 81%.
9. With anchor boxes our model gets 69.2 mAP with a recall of 88% .

Ref.: <https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e208>

**YOLO predictions:**



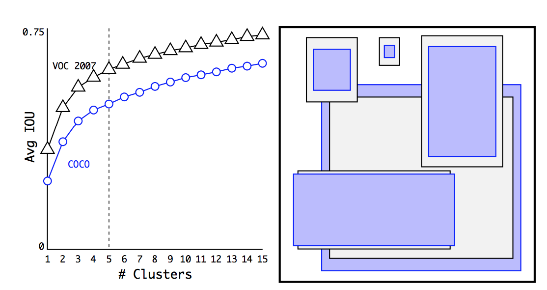
**YOLOv2 predictions:**



1. We predict offsets to each of the anchor boxes above. If we constrain the offset values, we can maintain the diversity of the predictions and have each prediction focuses on a specific shape. So the initial training will be more stable.
2. Anchors are also called priors.

**Dimension Clusters:**

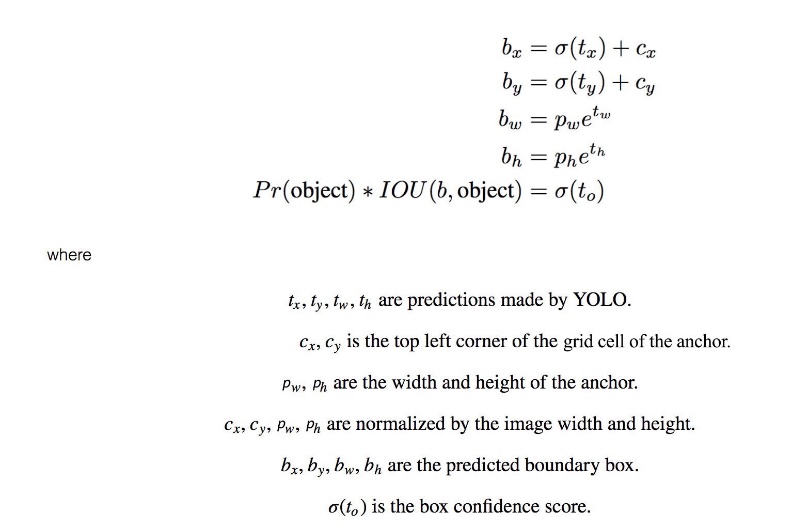
1. We encounter two issues with anchor boxes when using them with YOLO. The first is that the box dimensions are handpicked. The network can learn to adjust the boxes appropriately but if we pick better priors for the network to start with we can make it easier for the network to learn to predict good detections.
2. Instead of choosing priors by hand, we run k-means clustering on the training set bounding boxes to automatically find good priors. If we use standard k-means with Euclidean distance larger boxes generate more error than smaller boxes. However, what we really want are priors that lead to good IOU scores, which is independent of the size of the box.



1. On the left, we plot the average IoU between the anchors and the ground truth boxes using different numbers of clusters (anchors). As the number of anchors increases, the accuracy improvement plateaus. For the best return, YOLOv2 settles down with 5 anchors. On the right, it displays the 5 anchors’ shapes. The purplish-blue rectangles are selected from the COCO dataset while the black border rectangles are selected from the VOC2007. In both cases, we have more thin and tall anchors indicating that real-life boundary boxes are not arbitrary.

**Direct location Prediction:**

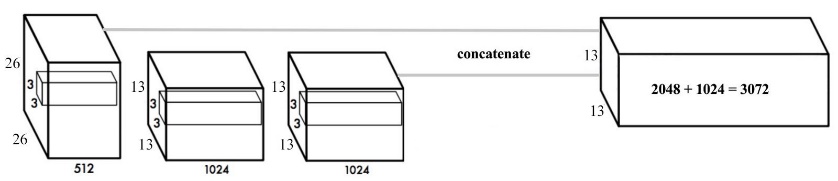
1. We make predictions on the offsets to the anchors. Nevertheless, if it is unconstrained, our guesses will be randomized again.
2. YOLOv2 predicts 5 parameters (tx, ty, tw, th, and to) and applies the sigma function to constraint its possible offset range.



1. With the use of k-means clustering (dimension clusters) and the improvement mentioned in this section, mAP increases 5%.

**Fine-Grained Features:**

1. Convolution layers decrease the spatial dimension gradually. As the corresponding resolution decreases, it is harder to detect small objects.
2. Other object detectors like SSD locate objects from different layers of feature maps. So each layer specializes at a different scale.



**Multi-Scale Training:**

1. After removing the fully connected layers, YOLO can take images of different sizes.
2. If the width and height are doubled, we are just making 4x output grid cells and therefore 4x predictions.
3. Since the YOLO network downsamples the input by 32, we just need to make sure the width and height is a multiple of 32. During training, YOLO takes images of size 320×320, 352×352, … and 608×608 (with a step of 32).
4. For every 10 batches, YOLOv2 randomly selects another image size to train the model. This acts as data augmentation and forces the network to predict well for different input image dimension and scale.
5. In additional, we can use lower resolution images for object detection at the cost of accuracy. This can be a good tradeoff for speed on low GPU power devices.
6. At 288 × 288 YOLO runs at more than 90 FPS with mAP almost as good as Fast R-CNN. At high-resolution YOLO achieves 78.6 mAP on VOC 2007.