

Session 16 & Assignment EVA2/3

[Submit Assignment](#)

Due Friday by 22:00 **Points** 1,000 **Submitting** a website url
Available 31 Jan at 23:00 - 7 Feb at 22:00 7 days

Object Localization - YOLO

Internet Access policy restricts access to this web page at 1



www.youtube.com/embed/NM6lrxy0bxs?start=67
the Category **YouTube - Film** is restricted.

If the access is for business requirement then please
log an AHD request for exception access.

Please refer [Infosys Internet Access Policy](#) for mo

Copyright © 2017 Infosys Limited.

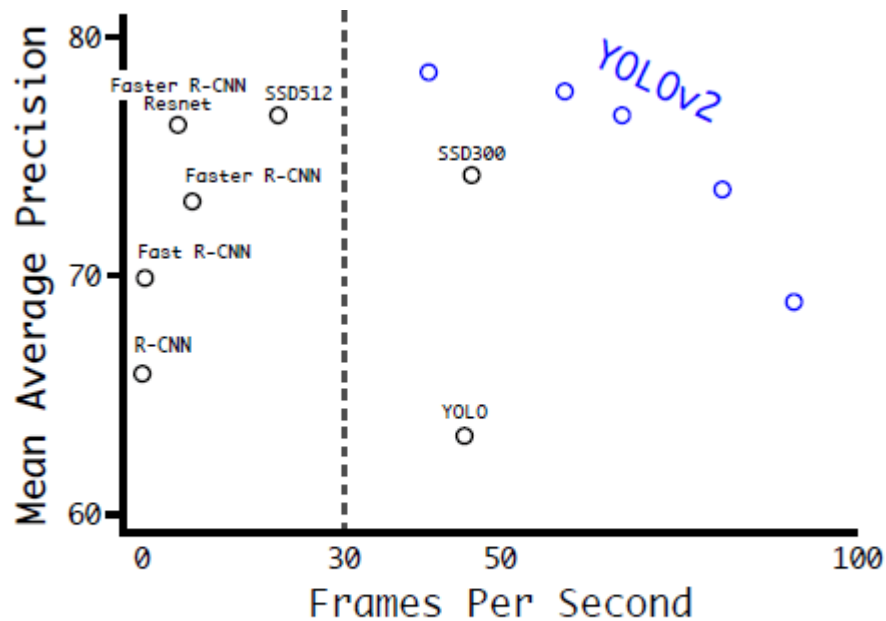
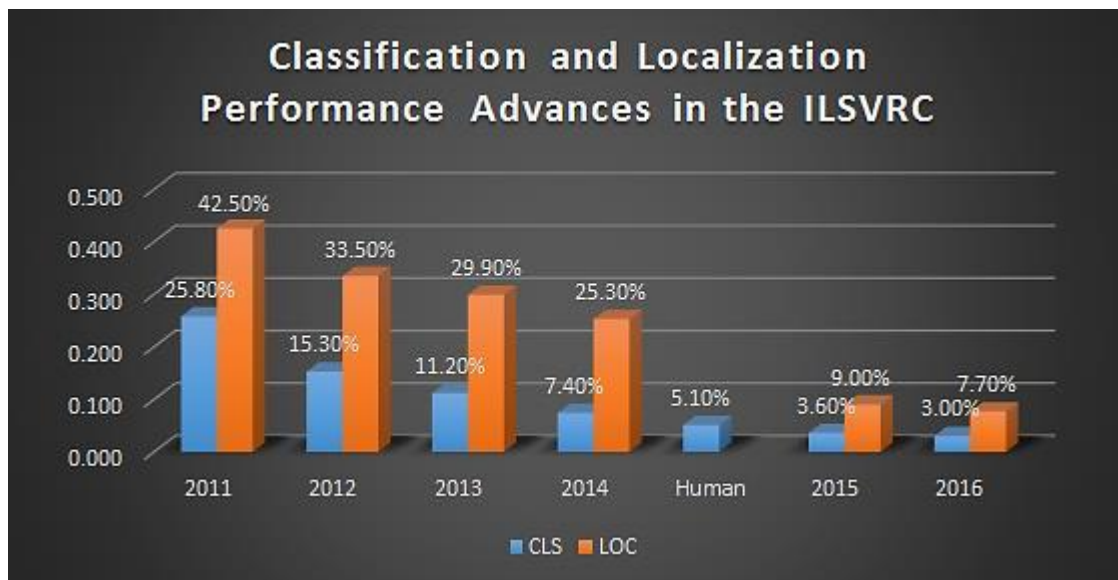


Figure 4: Accuracy and speed on VOC 2007.

Where are we today?



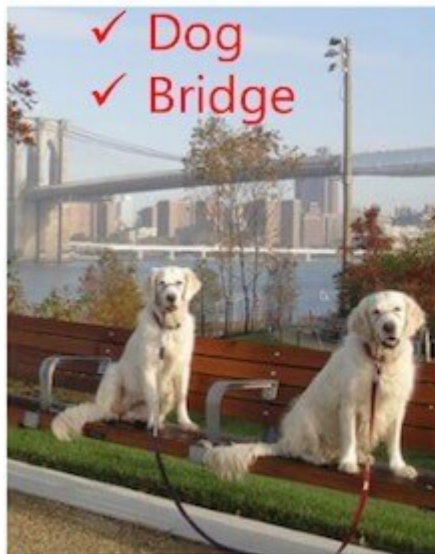
Let us observe how good ML has become is to see the ILSRVC results:

Year	Team	Layers	Contribution	Position
2010	NEC	Shallow	Fast Feature Extraction, Compression, SVM	First
2011	XRCE	Shallow	High dimensional image, SVM	First
2012	SuperVision 8	GPU based	DCNN, Dropout	First

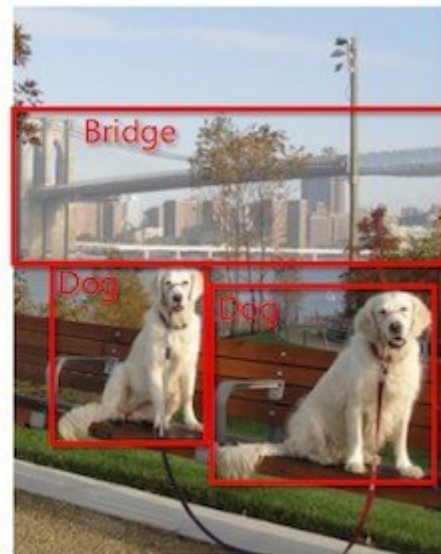
Year	Team	Layers	Contribution	Position
2013	Clarifai	8	Deconvolution visualization	First
2014	GoogLeNet	22	DCNN, 1x1 conv	First
2014	VGG	19	All 3x3 convs	Second
2015	MSRA	152	Ultra Deep, Residuals	First
2016	CUIImage	269	Ensamble, Gated Bi-Directional CNN	First
2017	Momenta	152	Squeeze & excitation, feature recalibration	First
2018*	Facebook	264	Direct connection between any two layers	SOA*

As we can see networks are getting deeper and more sophisticated. Every year there is there is an addition of new technology which is making everything till then obsolete.

Recognition vs Detection



Classification, easy these days



Object detection, still a lot harder

In object recognition, our aim is to recognize what all is there in the image, for e.g. Dog and Bridge. In object detection however, we need to specify where exactly the dog(s) and bridge are. Recognition is pretty easy these days, while detection is still a work in development.

Detection Approaches

Simplify Object Detection problem by:

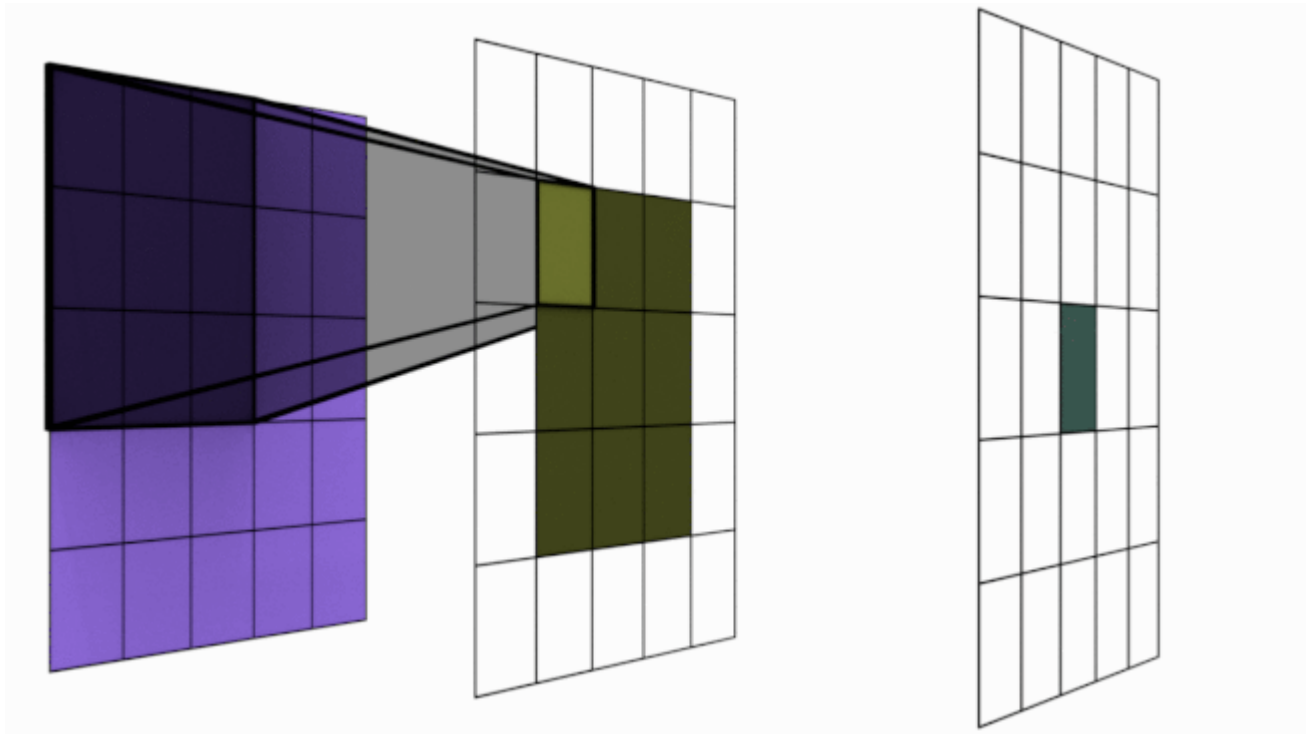
1. ignoring lower prediction values
2. predicting bounding boxes instead of exact object mask
3. mixing different receptive field layers
4. but this is easier said than done.

There are two main approaches driving detection algorithms, namely:

1. YOLO-like approach, where k-means extracted anchor boxes are used, and
2. SSD-like approach, where fixed number of predefined bounding boxes are used.

Let's recap

We know this image:

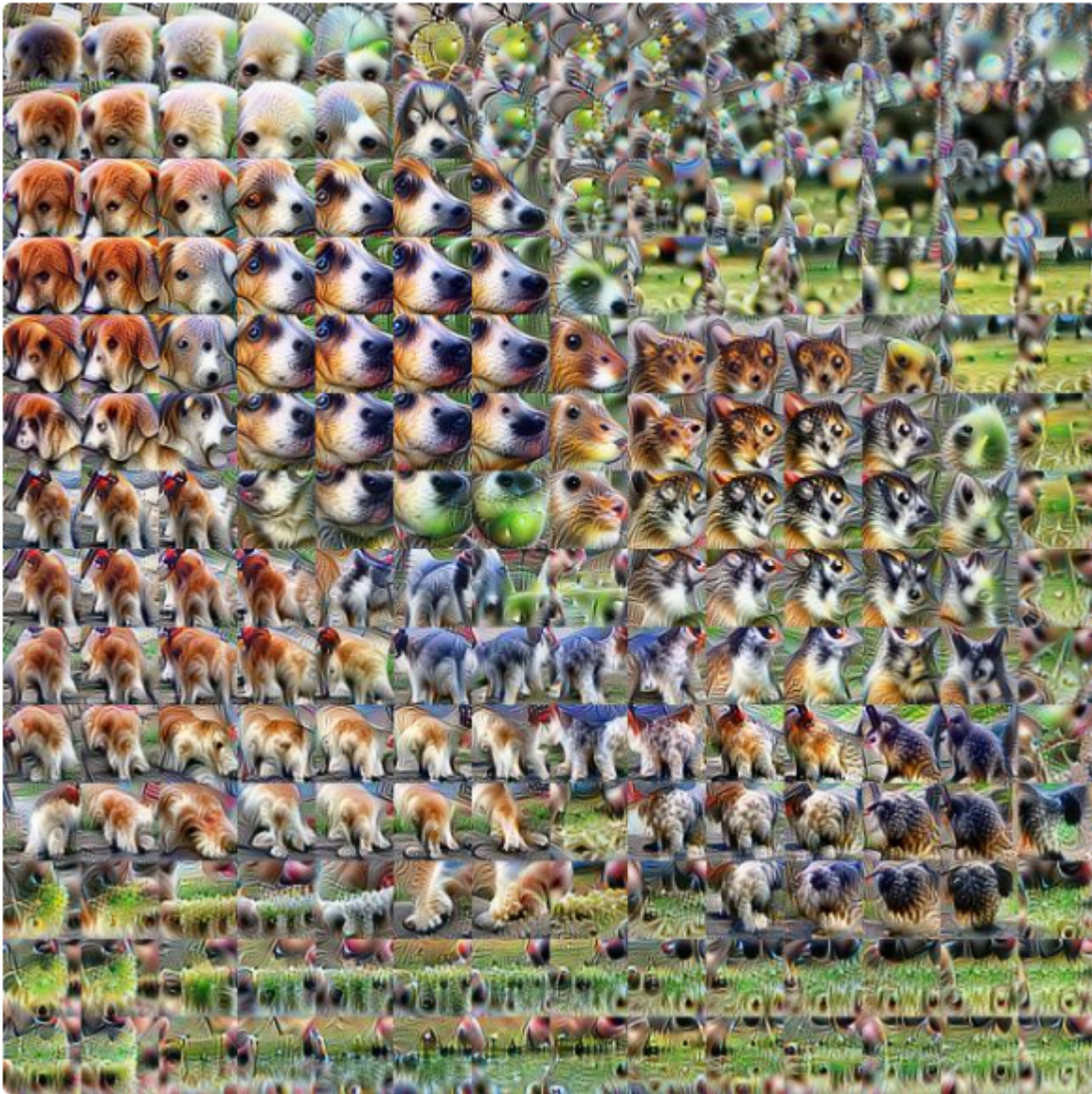


Later on

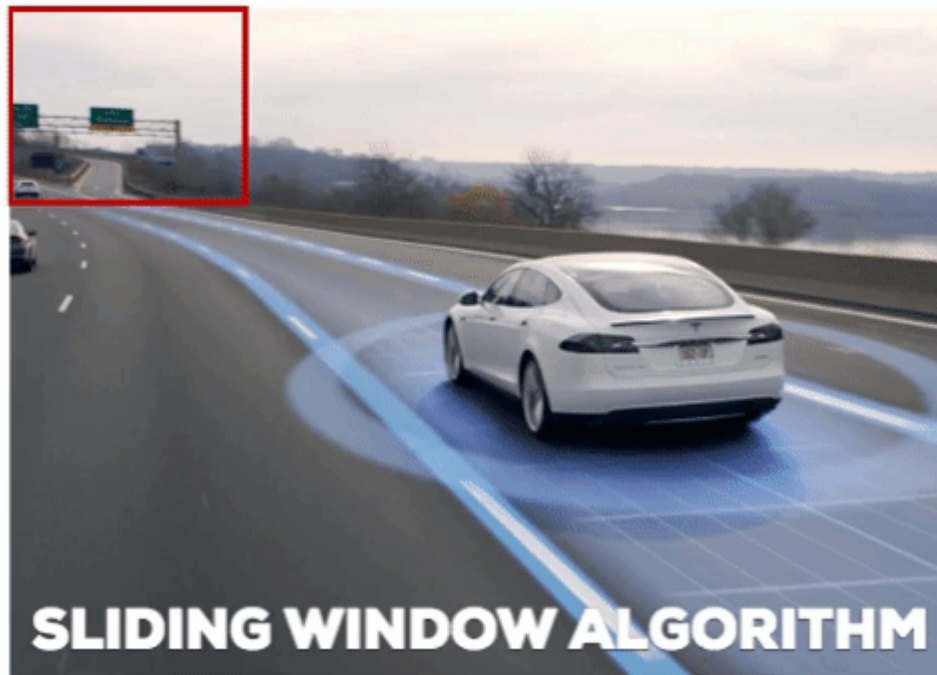


This is how prediction cells look like. You can relate the cells with the image.

THEN



How we used to find BBoxes



Why do you think this is bad? Because we need to answer few hard questions.

1. What should be the size of this window? How do we handle big and small objects at the same time?
2. Do we need to slide it at each pixel, or we can make a jump? What is the perfect jump size?

Sliding Window as a concept is simple, but extremely compute intensive.

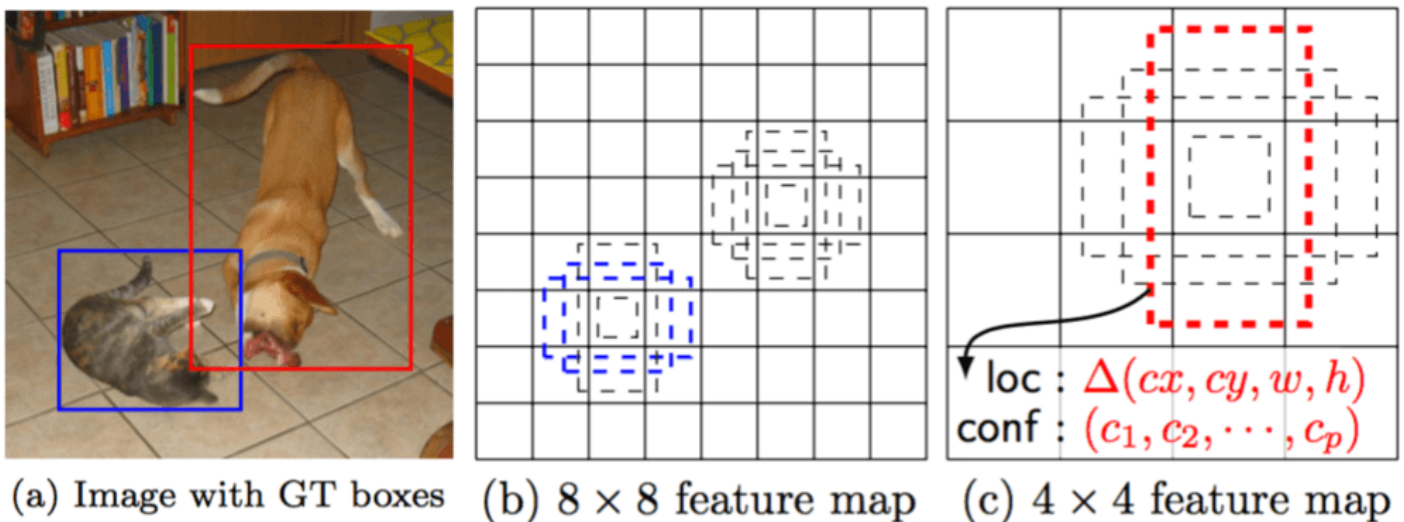
Region Proposal



RCNN uses this, and it was very bad, so bad that RCNN took 20 seconds to go through 1 image (it was running AlexNet 2000 times for 2000 proposals)

Welcome Anchor Boxes

Faster RCNN, SSD and YOLOv2, all use Anchor Boxes.

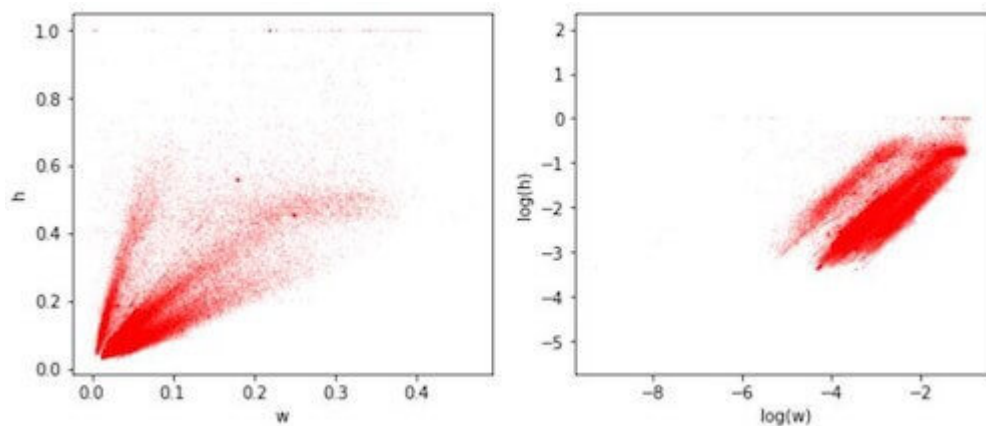


Intuitively, we know that objects in an image should fit certain common aspect ratios and sizes. For instance, we know that we want some rectangular boxes that resemble the shapes of humans. Likewise, we know we won't see many boxes that are very very thin. In such a way, we create k such common aspect ratios we call anchor boxes. For each such anchor box, we output one bounding box and score per position in the image.

How to calculate Anchor Boxes?

Let us see how to do it!

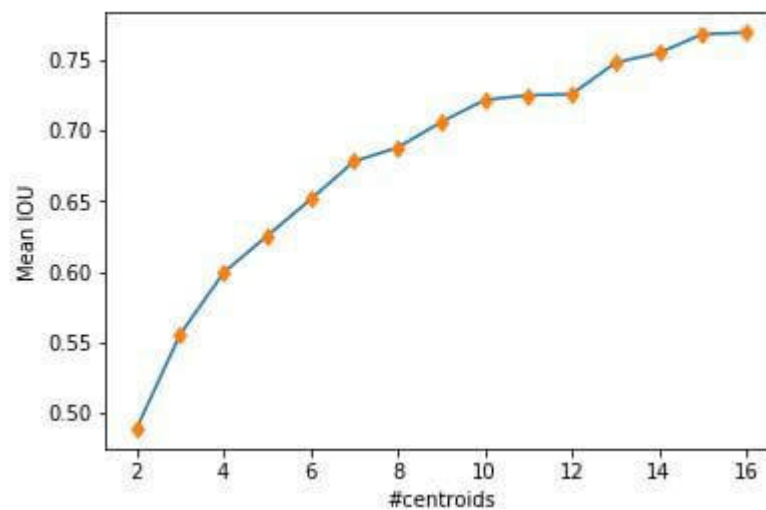
- Compute the center location, width and height of each bounding box, and normalize it by image dimensions (in the dataset)
- Plot the h and w for each box as shown in the right "h vs w graph"
- Use K-means clustering to compute cluster centers (centroids).



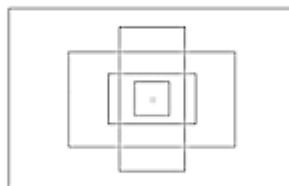
StatQuest: K-means clustering



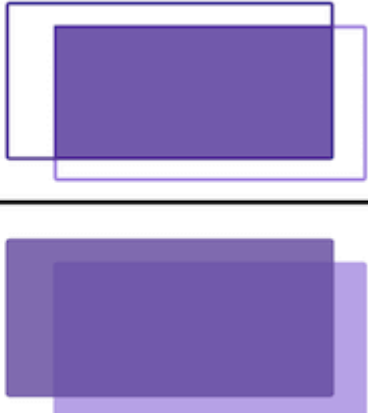
- Compute different number of clusters and compute the mean of maximum IOU between bounding boxes and individual anchors.
- Plot centroids vs mean IOU.



- Pick the top 5 anchor boxes (5 for YOLOv2 where IOU was above 65%)



Intersection Over Union

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$


Intersection over Union is an evaluation metric used to measure the accuracy of an object detector on a particular dataset.

The Scary loss function

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

We have 5 anchor boxes. For each anchor box we need Objectness-Confidence Score (where there is an object found?), 4 Coordinates (t_x , t_y , t_w and t_h), and 20 top classes. This can crudely be seen as 20 coordinates, 5 confidence scores, and 100 class probabilities as shown in the image on the right, so in total 125 filter of 1x1 size would be needed.

So we have few things to worry about:

- x_i , y_i , which is the location of the centroid of the anchor box
- w_i , h_i , which is the width and height of the anchor box
- C_i , which is the Objectness, i.e. confidence score of whether there is any object or not, and
- $p_i(c)$, which is the classification loss.

All losses are mean squared errors, except classification loss, which uses cross entropy function.

Understanding YoloV2 Loss Function



Let's look at that loss function again.

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

Now, let's break the code in the image.

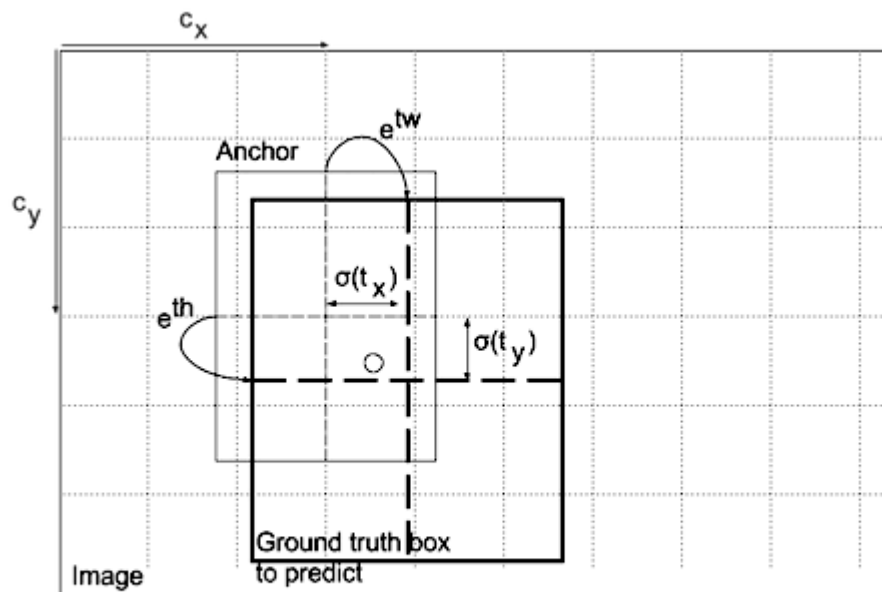
- We need to compute losses for each Anchor Box (5 in total)
 - \sum^B represents this part.
- We need to do this for each of the 13x13 cells where $S = 13$
 - \sum^{S^2} represents this part.
 - $\mathbb{1}_{ij}^{\text{obj}}$ is 1 when there is an object in the cell i , else 0.
- $\mathbb{1}_{ij}^{\text{noobj}}$ is 1 when there is no object in the cell i , else 0. We need to do this to make sure we reduce the confidence when there is no object as well.
- $\mathbb{1}_i^{\text{obj}}$ is 1 when there is a particular class is predicted, else 0.
- λ s are constants. λ is highest for coordinates in order to focus more on detection (remember, we have already trained the network for recognition!)
- We can also notice that w_i, h_i are under square-root. This is done to penalize the smaller bounding boxes as we need to adjust them more.

Check out this table:

var1	var2	(var1-var2)^2	(sqrtvar1-sqrtvar2)^2
0.0300	0.020	9.99e-05	0.001
0.0330	0.022	0.00012	0.0011
0.0693	0.046	0.000533	0.00233

var1	var2	(var1-var2)^2	(sqrtvar1-sqrtvar2)^2
0.2148	0.143	0.00512	0.00723
0.8808	0.587	0.0862	0.0296
4.4920	2.994	2.2421	0.1512

Understanding YoloV2 Paper



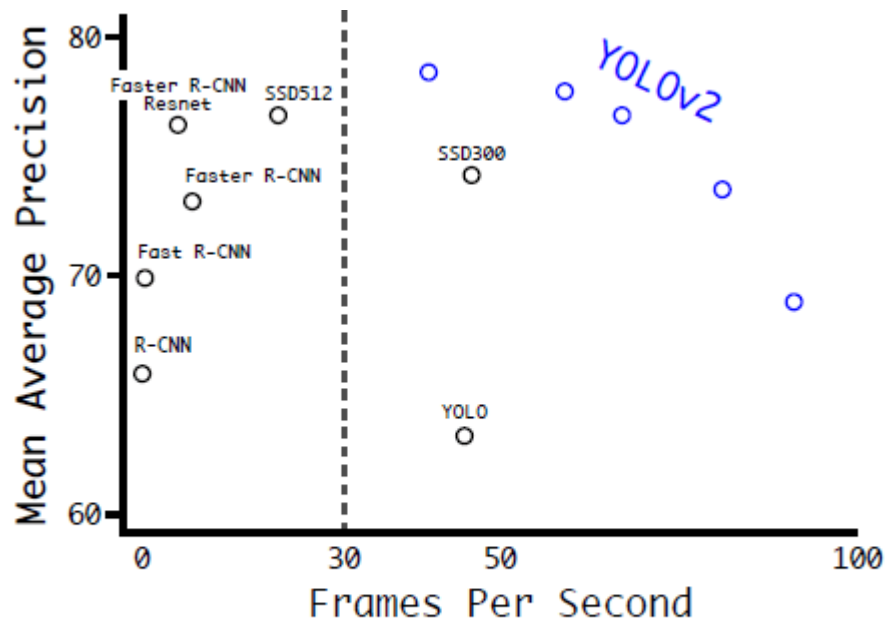
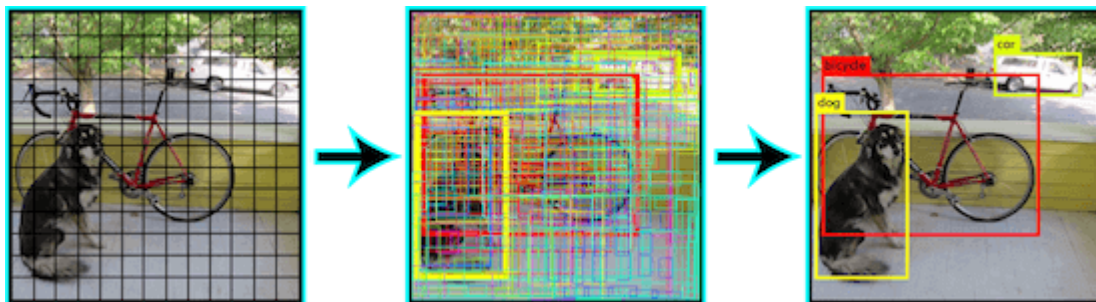
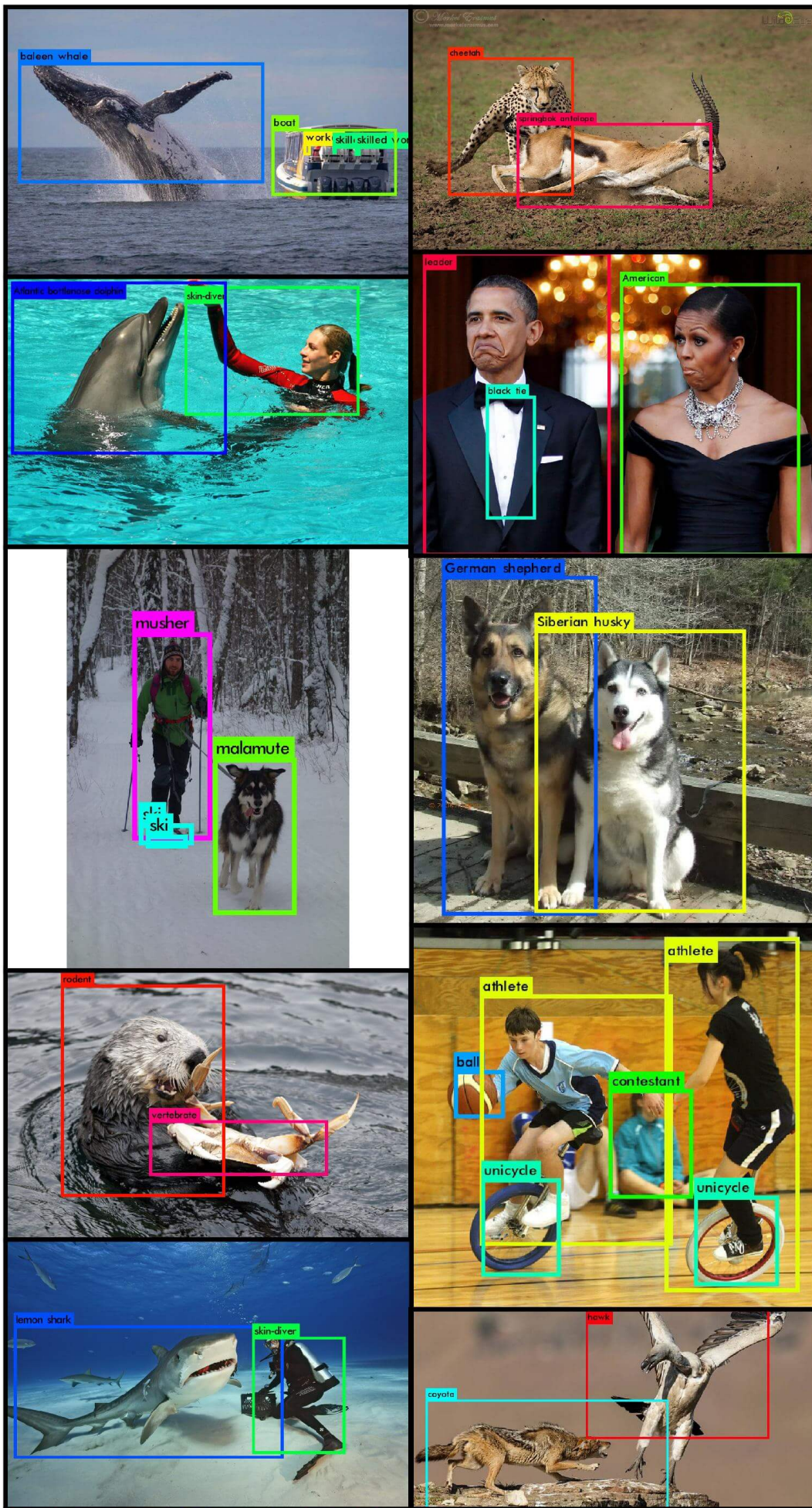


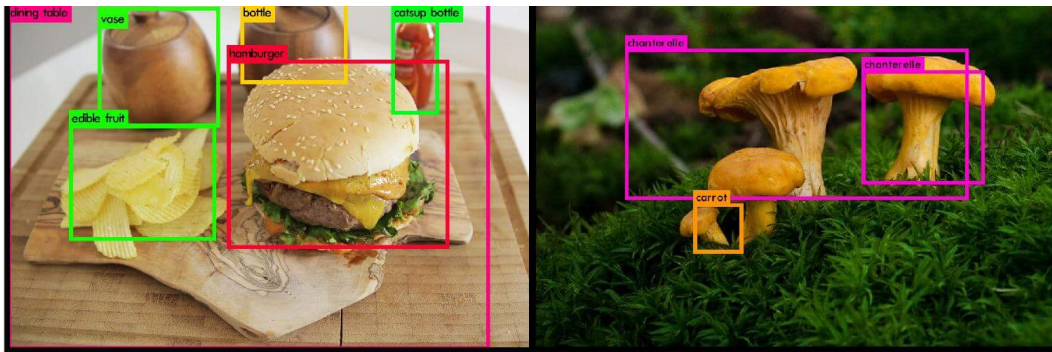
Figure 4: Accuracy and speed on VOC 2007.



- YOLOv2 fixed v1's recall and localization problems
- By adding BN to **all the layers** it gets 2% improvement in mAP
- YOLOv2 is first fine tuned on higher resolution images. This gives the network time to adjust it's filters to work better on higher resolution network. Then it is fine tuned on detection. This gives 4% mAP improvement.
- YOLOv2 fully removes FC layers and use **anchor boxes** instead to predict bounding boxes.
- YOLOv2 predicts **class** as well as **objectness** for every anchor box.
- Anchor dimensions are picked using k-means clustering on the dimensions of original bounding boxes. Final anchor boxes are: (0.57273, 0.677385), (1.87446, 2.06253), (3.33843, 5.47434), (7.88282, 3.52778), (9.77052, 9.16828)
- The network predicts 5 bounding boxes and 5 coordinates for each bounding box: (t_x , t_y , t_w , t_h and t_o)
- If the cell is offset from the top left corner of the image by c_x , c_y and the bounding box ground-truth/prior has width and height g_w , g_h then the predictions corresponds to:
 - $bx = \sigma(t_x) + c_x$, where σ is sigmoid
 - $by = \sigma(t_y) + c_y$

- $bw = g_w e^{tw}$
 - $bh = g_h e^{th}$
- YOLO predicts detection on a 13x13 feature map. This may not be sufficient to detect smaller objects. To fix this, they simply add a pass through layer that brings features from an earlier layer at 26x26 resolution. The pass through layer concatenates the higher resolution features with the low resolution features by stacking adjacent features into different channels.
- Every 10 batches, network chooses random new image dimension size (multiples of 32) from 320x320 to 608x608.
- Final model, called Darknet-19 has 19 convolution layers and 5 maxpooling layers. 1x1 convolutions are used to compress the feature representations between 3x3.
- Network is first trained on classification for 160 epochs.
- After classification training, last convolution layer is removed, and three 3x3 convolution layers with 1024 filters each followed by final 1x1 convolution layer is added. Network is again trained for 160 epochs.
- During training both, detection and classification datasets are mixed. When network sees image with detection label, full back-propagation is performed, else only classification part is back-propagated.





Annotations

Let's look at VGG Annotator: http://www.robots.ox.ac.uk/~vgg/software/via/via_demo.html
http://www.robots.ox.ac.uk/~vgg/software/via/via_demo.html)

ASSIGNMENT - EVA3

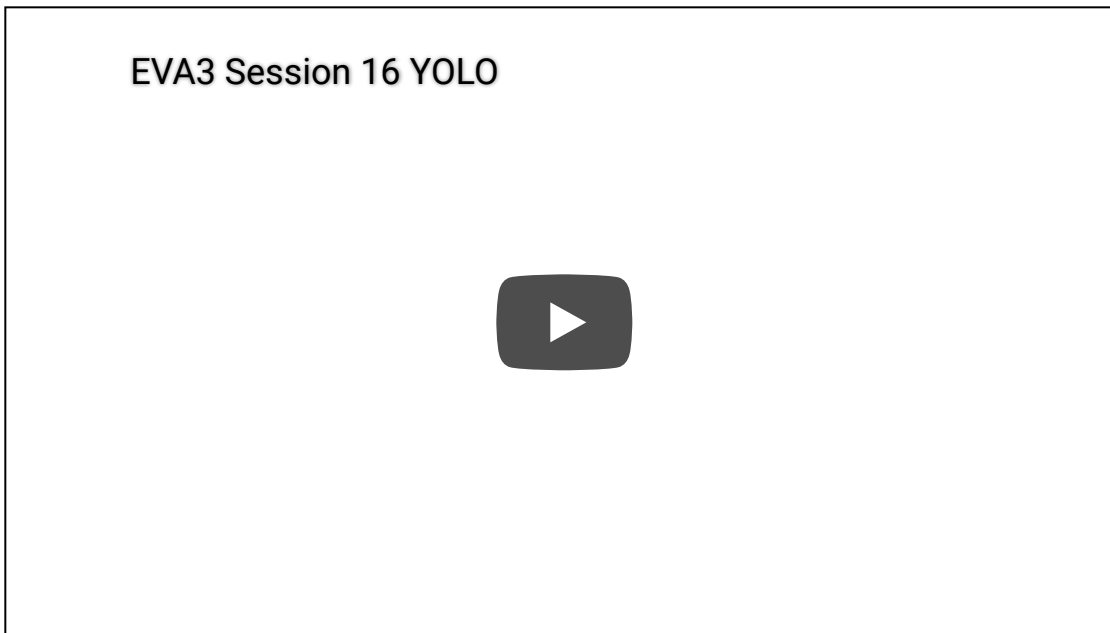
1. Refer to this [TSAI COLAB FILE](https://colab.research.google.com/drive/1iZdzl0VEG8ieRgHXKT7tNEE7iN3gw4tN)
<https://colab.research.google.com/drive/1iZdzl0VEG8ieRgHXKT7tNEE7iN3gw4tN>)
2. We have stitched multiple projects into 1, so you can train tiny-YOLO on COLAB!
3. Refer to this blog: [LINK](https://medium.com/@today.rafi/train-your-own-tiny-yolo-v3-on-google-colaboratory-with-the-custom-dataset-2e35db02bf8f) <https://medium.com/@today.rafi/train-your-own-tiny-yolo-v3-on-google-colaboratory-with-the-custom-dataset-2e35db02bf8f>) and [LINK](https://github.com/rafiuddinkhan/Yolo-Training-GoogleColab/blob/master/helmet.ipynb)
<https://github.com/rafiuddinkhan/Yolo-Training-GoogleColab/blob/master/helmet.ipynb>). This is the main source for our project.
4. Refer to the "main.py" in Step 1.
5. Here is what you need to do:
 1. create your own dataset using main.py
 2. collect 200 images (more the better) for any 1 class of your choice. e.g. this project is for the helmet. You cannot use the same dataset as in this project.
 3. you should be able to find a short youtube video for this class as well (example classes you can pick: traffic_light, dog, car, bird, flag, etc)
 4. annotate 200 images as explained in the links above
 5. replace the data_for_colab folder with your custom folder
 6. train YOLO for 1000 epochs (more the better)
 7. download 1 youtube video which has your class, you can use this:
<https://www.y2mate.com/en4> <https://www.y2mate.com/en4>)
 8. run this command

1.

```
!./darknet detector demo data_for_colab/obj.data data_for_colab/yolov3-tiny-obj.cfg backup/yolov3-tiny-obj_1000.weights -dont_show youtube_video.mp4 -i 0 -out_filename veout.avi
```

9. upload your video on YouTube.
10. Share the link with us (and your LinkedIn as well if you want!!)
11. Fixed deadline and try to do this in a group (at least share your datasets).

EVA3 Session Video:



The assignment - OLD

1. Collect 100 images of faces from online sources (you can use any existing database as well, but we need multiple faces)
2. Please make sure that there are not too many faces in the image
3. Classes are: Front, Left, Right, Up, Down, UpLeft, UpRight, DownLeft, DownRight, Top, Back.
Please make sure you have these kind of faces in your collection. Also please make sure that your LEFT is the LEFT of your screen.
4. resize your images to 400x400
5. Rename your images as img_001 to img_100.

6. Annotate these objects using **VGG Annotator** [\(http://www.robots.ox.ac.uk/~vgg/software/via/\)](http://www.robots.ox.ac.uk/~vgg/software/via/)
(using a local copy)
7. Use K-means clustering to find out the top 4 anchor boxes
8. Upload to github
 - images in a zipped folder
 - your annotation file (json)
 - k-means code
9. Add a readme file and show:
 - few screenshots of your annotations
 - your 4 bounding box dimensions

You will be using these images for your Face Recognition Session, so make sure your annotations are good.

GIGO

EVA2 Session 16 Video:



Archived Videos Below:

Session Video - Open to all Version 2

EVA 1 Session 19 Version 2



Session Video - Version 1

EVA 1 Session 19

