REPORT

CECS 551 – Assignment 8

Name: Aishwarya Bhavsar

CSULB ID: 029371509

Date – 11th April 2022

I certify that this submission is my original work – AVB

- **GOAL:** **To Develop face recognition software using pre-trained YOLO V3 and Facenet model.**
- **STEPS/PROCEDURE:**

STEP 1: Created a dataset of 1200 celebrity images from the original dataset containing 202,599 images. (30 images for each of the 40 celebrities)

STEP 2: Applied YOLOv3 Algorithm.

STEP 3: Applied Facenet detection.

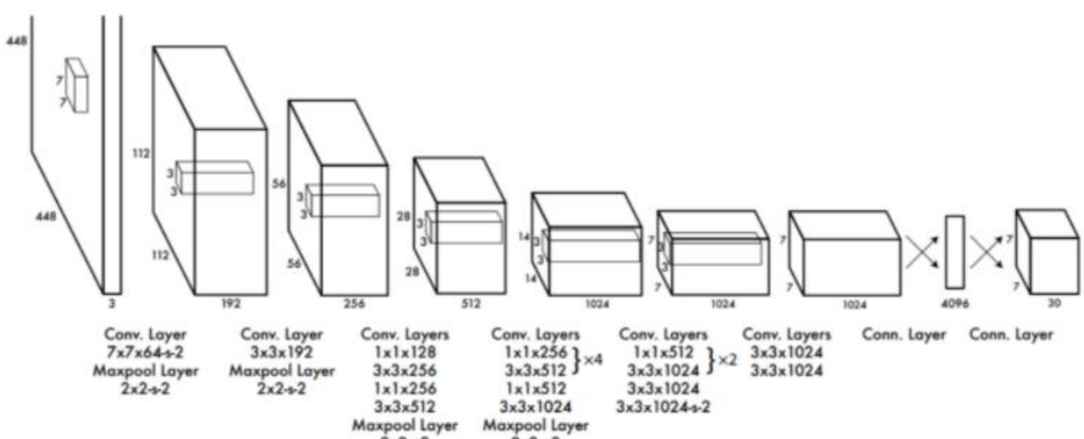
STEP 4: Convert the image to embedding vector.

STEP 5: Calculate and plot Precision and Recall.

- **RESULTS:**

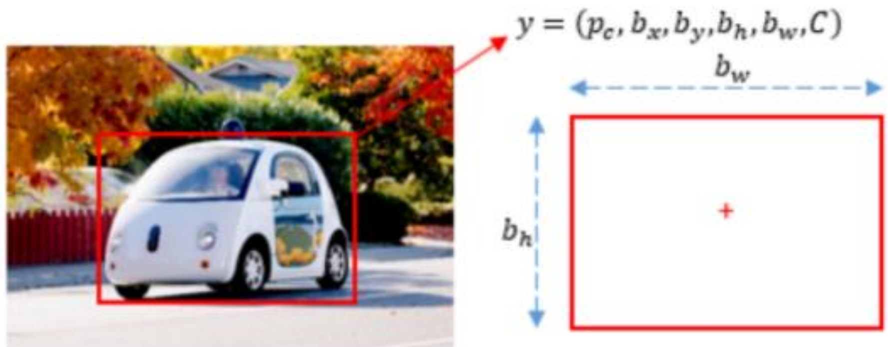
YOLO Algorithm:

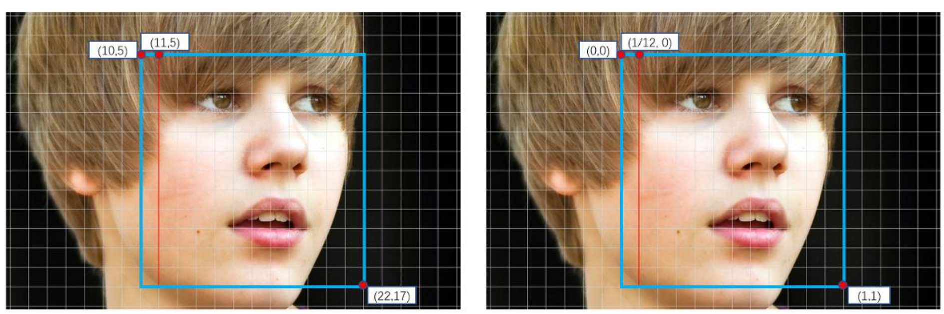
You only look once (YOLO), a single convolutional neural network different from prior detectors, frames object as a regression problem to spatially separated bounding boxes and associated class probabilities directly from full images in one evaluation [1]. The full network YOLOv1 architecture with 24 convolutional layers and 2 fully connected layers is shown below in figure.



How YOLO works:

The YOLO algorithm divides any given image into the S×S grid. Each grid cell on the input image predicts a fixed number of boundary boxes (anchor boxes) for an object. As for each boundary box the network outputs offset 4 element values (bx, by, bh, bw), one confidence pc, and C conditional class probabilities.





The coordinates (bx, by) represent the bounding box's center relative to the bounds of the grid cell in the input image. The bw and bh are box's width, and height respectively. The confidence pc is the probability hat a box contains an object and how accurate the boundary box is.

YOLOv3 has 3 different scales, at each scale predict 3 anchor boxes.

YOLOv3 applies the k-means cluster to determine the priors (anchors).

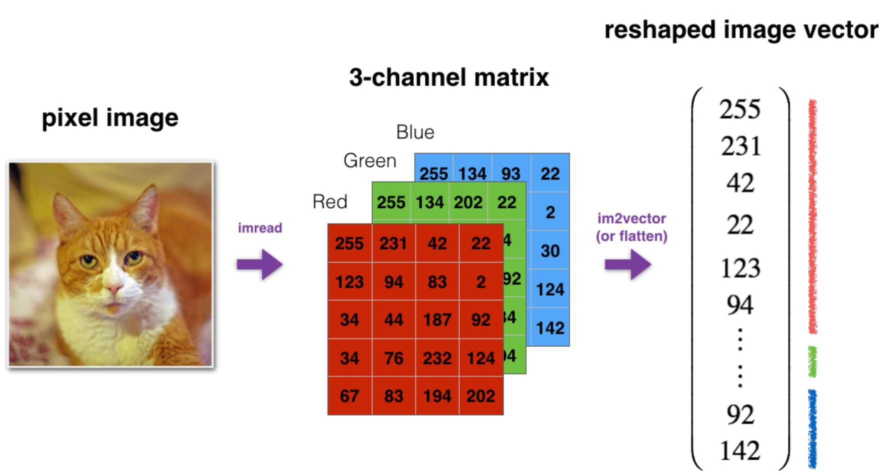
YOLOv3 uses a new network called Darknet-53 for performing feature extraction

The network uses successive 3 × 3 and 1 × 1 convolutional layer.

	Type	Filters	Size	Output
	Convolutional	32	3 x 3	256 x 256
	Convolutional	64	3 x 3 / 2	128 x 128
	Convolutional	32	1 x 1	
1x	Convolutional	64	3 x 3	
	Residual			128 x 128
	Convolutional	128	3 x 3 / 2	64 x 64
	Convolutional	64	1 x 1	
2x	Convolutional	128	3 x 3	
	Residual			64 x 64
	Convolutional	256	3 x 3 / 2	32 x 32
	Convolutional	128	1 x 1	
8x	Convolutional	256	3 x 3	
	Residual			32 x 32
	Convolutional	512	3 x 3 / 2	16 x 16
	Convolutional	256	1 x 1	
8x	Convolutional	512	3 x 3	
	Residual			16 x 16
	Convolutional	1024	3 x 3 / 2	8 x 8
	Convolutional	512	1 x 1	
4x	Convolutional	1024	3 x 3	
	Residual			8 x 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Table 2. Darknet-53 [3]

Embedding vectors: Converting a face image into numerical data



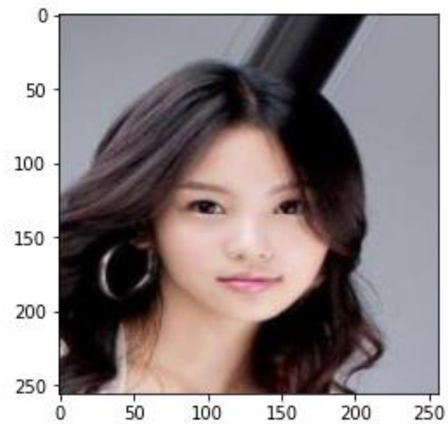
$f($  $) =$

$$\begin{pmatrix} 0.112 \\ 0.067 \\ 0.091 \\ 0.129 \\ 0.002 \\ 0.012 \\ 0.175 \\ \vdots \\ 0.023 \end{pmatrix}$$

We get 1200 embedding vectors for 1200 images.

Conversion Of Images into Embedding Vector and applied flattening and normalization.

```
Image:
(256, 256, 3)
```




```
[0.32974282 0.85666019 0.05894995 0.74907148 0.24750969 0.53491944
 0.27259308 0.94578403 0.34352449 0.48639581]
```

Euclidean Distance:

$$d(\mathbf{p},\mathbf{q}) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

p, q = two points in Euclidean n-space

q_i, p_i = Euclidean vectors, starting from the origin of the space (initial point)

n = n-space

```
# using for loop
a = list_embeddings

# printing the list using loop
for x in range(len(a)):
    print(distance.euclidean(a[x],a[x-1]))
```

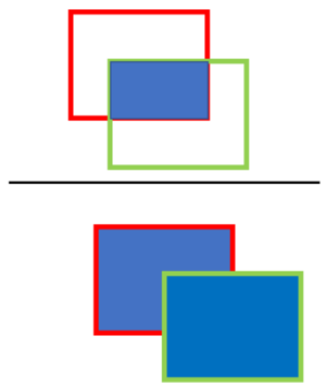
```
15.255477973140366
13.308284858550515
10.944386163754464
11.9162972071459
12.242808540214265
14.519597915340547
15.064556109993775
10.491491781976704
11.902779072662932
11.381331722599525
15.50803935474357
13.180986546993502
15.013013015970973
14.770182753502827
11.030871686454432
12.715869067136753
12.072002500922158
12.023612720033855
```

- **DISCUSSIONS:**

$$Pr = \frac{\sum_{n=1}^{S} \mathrm{TP}_n}{\sum_{n=1}^{S} \mathrm{TP}_n + \sum_{n=1}^{N-S} \mathrm{FP}_n} = \frac{\sum_{n=1}^{S} \mathrm{TP}_n}{\text{all detections}},$$

$$Rc = \frac{\sum_{n=1}^{S} \mathrm{TP}_n}{\sum_{n=1}^{S} \mathrm{TP}_n + \sum_{n=1}^{G-S} \mathrm{FN}_n} = \frac{\sum_{n=1}^{S} \mathrm{TP}_n}{\text{all ground truths}}.$$

- Intersection over Union=



- Red is ground truth bounding box and green is predicted bounding box
- **Precision** measures the model trustiness in classifying positive samples.
- **Recall** measures how many positive samples were correctly classified by the model.
- The precision considers both the positive and negative samples were classified.
- Recall only considers the positive samples in its calculations.
- In other words, the precision is dependent on both the negative and positive samples.
- Recall is dependent only on the positive samples (and independent of the negative samples).
- The precision considers when a sample is classified as *Positive*, but it does not care about correctly classifying *all* positive samples. The recall cares about correctly classifying *all* positive samples, but it does not care if a negative sample is classified as positive.
- When a model has high recall but low precision, then the model classifies most of the positive samples correctly, but it has many false positives (i.e. classifies many *Negative* samples as *Positive*). When a model has high precision but low recall, then the model is accurate when it classifies a sample as *Positive,* but it can only classify a few positive samples.

Inference: Non-maximal suppression

To remove duplications.

Conclusion:

1. When IoU > 0.5 we get a good confidence and accuracy.
2. By keeping the tau values in between 0 and 1 we get a good precision.
3. Euclidean distance is a good measure when the activation function is softmax.
4. YOLOv3 detects the face with accuracy
5. YOLOv3 predicts an objectness score for each bounding box using logistic regression.
6. YOLO's prediction has a shape = S * S *(B * 5 + C)
7. $$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{t_w}$$
$$b_h = p_h e^{t_h}$$

Confidence(τ)	IOU	IOU > 0.5?	$\sum$TP(τ)	$\sum$FP(τ)	$Pr(\tau)$	$Rc(\tau)$
99%	0.91	Yes	1	0	1.0000	0.0833
98%	0.70	Yes	2	0	1.0000	0.1667
95%	0.86	Yes	3	0	1.0000	0.2500
95%	0.72	Yes	4	0	1.0000	0.3333
94%	0.91	Yes	5	0	1.0000	0.4167
92%	0.86	Yes	6	0	1.0000	0.5000
89%	0.92	Yes	7	0	1.0000	0.5833
86%	0.87	Yes	8	0	1.0000	0.6667
85%	-	No	8	1	0.8889	0.6667
82%	0.84	Yes	9	1	0.9000	0.7500

criteria	Precision		Recall		F1 score	
Index	Baseline	Baseline + WN	Baseline	Baseline + WN	Baseline	Baseline + WN
1	0.774	0.765	0.561	0.666	0.650	**0.712**
2	0.719	0.722	0.628	0.686	0.670	**0.704**
3	0.802	0.811	0.848	0.840	0.824	**0.826**
4	0.667	0.649	0.449	0.545	0.537	**0.593**
5	0.804	0.781	0.660	0.735	0.725	**0.758**
6	0.887	0.882	0.841	0.861	0.864	**0.871**
7	0.676	0.632	0.220	0.284	0.332	**0.392**
8	0.637	0.651	0.516	0.560	0.570	**0.602**
9	0.829	0.829	0.772	0.791	0.800	**0.810**
10	0.849	0.866	0.835	0.838	0.842	**0.852**

Threshold	Accuracy	Precision	Recall	F1 Score
0.00	0.37	0.37	1.00	0.54
0.05	0.41	0.39	1.00	0.56
0.11	0.55	0.45	0.99	0.62
0.16	0.78	0.63	0.94	0.76
0.21	0.86	0.81	0.83	0.82
0.26	0.86	0.96	0.67	0.79
0.32	0.83	0.99	0.56	0.71
0.37	0.78	1.00	0.42	0.59
0.42	0.75	1.00	0.32	0.49
0.47	0.70	1.00	0.19	0.32
0.53	0.66	1.00	0.10	0.18
0.58	0.65	1.00	0.07	0.12
0.63	0.65	1.00	0.06	0.11
0.68	0.64	1.00	0.03	0.06
0.74	0.63	1.00	0.02	0.04
0.79	0.63	1.00	0.02	0.04
0.84	0.63	1.00	0.01	0.03
0.89	0.63	1.00	0.01	0.02
0.95	0.63	1.00	0.01	0.02
1.00	0.63	1.00	0.00	0.01

- **References:**

1.https://github.com/DrManishSharma/YOLO_Obj_Detection/blob/master/Yolo_Obj_Detection_Using_Colab.ipynb

2. https://github.com/sthanhng/yoloface/blob/master/yoloface.py

3. https://alexeyab84.medium.com/yolov4-the-most-accurate-real-time-neural-network-on-ms-coco-dataset-73adfd3602fe

4. https://github.com/AlexeyAB/darknet

5. https://github.com/chinmaykumar06/face-detection-yolov3-keras/blob/main/face-detection.ipynb

6. https://towardsdatascience.com/object-detection-using-yolov3-and-opencv-19ee0792a420

7. https://machinelearningmastery.com/how-to-perform-object-detection-with-yolov3-in-keras/

8. https://machinelearningmastery.com/roc-curves-and-precision-recall-curves-for-imbalanced-classification/#:~:text=A%20precision%2Drecall%20curve%20(or,)%20vs%20Precision%20(y).

9. https://github.com/davidsandberg/facenet

10. https://github.com/timesler/facenet-pytorch

11. https://github.com/akshaybahadur21/Facial-Recognition-using-Facenet/blob/master/requirements.txt

12. https://github.com/joonson/face_trainer

13. https://github.com/axinc-ai/yolov3-face

14. https://github.com/ipazc/mtcnn

15. https://towardsdatascience.com/face-detection-using-mtcnn-a-guide-for-face-extraction-with-a-focus-on-speed-c6d59f82d49