Single Shot Scale-Invariant Face Detector and Recognition of Faces
Final Report

Team Members: Jasmeet Narang (A20438656)
Shirish Mecheri Vogga (A20456808)

1. ABSTRACT

Face detection is the most essential and first step to recognition, and it is used to detect faces in the images. It is a computer vision application used to identify human faces in digital images. Face detection has been used in multiple applications like, detect faces in real-time for surveillance and tracking of people, identifying people on social media platforms, validate identity at ATMs etc. The **paper [1]** presents a real-time face detector, named Single shot Scale-invariant Face detector, that performs superiorly on various scales of faces with a single deep neural network. Additionally, faces can be recognized using VGG Face model like VGG Face or VGG Face2. These are ResNet-50 or SqueezNet-ResNet-50 neural network architecture that are trained on the dataset.

2. PROBLEM STATEMENT

Face detection is the key step of many subsequent face related applications, such as face alignment, face recognition, face verification and face tracking. The main idea of the project is to be able to detect small and large faces in an image and use a deep learning model to perform face recognition on celebrity's dataset. To do so we build a single shot scale invariant face detection model with a wide range of anchor-associated layers and a series of reasonable anchor scales so as to handle different scales of faces well. Once we have the detected faces of celebrities, VGG Face a deep learning model is used to recognize the celebrity.

3. PROPOSED SOLUTION

Face Detector:

The model is based on the anchor-based detection framework, the main drawback of the framework is that the performance drops dramatically as the face becomes smaller. To improve this, we develop a network architecture with a wide range of anchor associated layers whose stride size gradually doble from 4 to 128 pixels, this ensures that the different scales if faces have adequate features for detection at corresponding anchor associated layers.

The architecture is based on the VGG16 network, below is the details architecture

- Base Convolution layers: we keep the VGG16 from conv1 to pool5
- Extra Convolutional Layers: we convert the fully connected layer 6 and 7 to convolutional layers by subsampling their parameters and adding an extra convolutional layer behind that to create a multi scale feature map.

- Detection Convolution layers: the conv3_3, conv4_3, conv5_3, conv_fc7, conv6_2 and conv7_2 layers are associated with different scales of anchors to predict detections.
- Normalization Layers: Comparing to other detection layers, conv3 3, conv4 3 and conv5 3 have different feature scales. Hence, we use L2 normalization to rescale their norm to 10, 8 and 5 respectively.
- Predicted Convolutional Layers: Each detection layer is followed by a p×3×3×q convolutional layer, where p and q are the channel number of input and output, and 3×3 is the kernel size. For each anchor, we predict 4 offsets relative to its coordinates and Ns scores for classification, where Ns = Nm + 1 (Nm is the maxout background label) for conv3 3 detection layer and Ns = 2 for other detection layers.
- Multi-task loss layer: We use softmax for classification and L1 loss for regression

Face Recognition Model:
Once we have the bounding box detecting faces in an image, we used the keras-vggface library that provides the pre-trained VGG Model to create a face recognition model. The first time the model is created, the library will download the model weights and save then in the ./keras/models/vggface/ directory in home directory. The size of the weights for the resnet50 model is about 158 megabytes. The model then makes predictions for each face in the image.

4. IMPLEMENTATION DETAILS

Face Detector:
Our solution is based on the modified version of a Single Shot decoder or SSD.
Our model consists of a base VGG16 network connected to a set of convolution layers to further process the image.

In this architecture in order to make it scale invariant we extract the results from the 3 layers of the VGG16 namely ["block3_conv3", "block4_conv3", "block5_conv3"] Which acts as the detection layers for the small and medium faces since these layers have a high resolution but lesser information , while results of the layers in the later convolution layers are extracted which contains information about larger faces as they are lower resolution but contain very high spatial resolution which indicates the amount of information stored in that layer.

These layers extracted are known as the detected Layers, the results extracted from the VGG16 model are first subjected to L2 normalization since the results generated in this layer compared to later convoluted layers are higher in magnitude therefore in order to

reduce it to a same scale we perform this operation. These detected layers are then convoluted with a layers having the number of units as (no_classes * no_boxes) where no_classes indicates the face class and the background class while the no_boxes indicates how many boxes for each detection layer. Similarly, to detect the bounding box coordinates we again convolve the detection layers with convolutional layers having units as (no_boxes * 4) which replaces the dense layer there by producing output of that shape. Hence this way we make the model fully convolutional and also efficient since dense layers are a performance overhead to the model

Our total predicted layers are 6 and hence each of these layers predict the box and the class label for each of the box . All of these layers that predict the class are concatenated fed to a softmax activation function to generate the probabilities and lastly, we concatenate all of them to generate the final output for the model as shown below

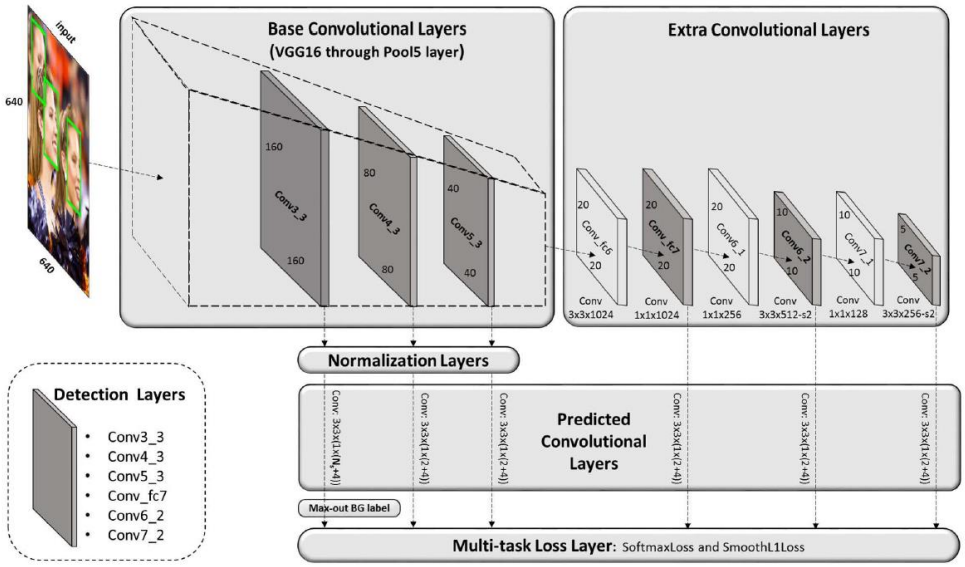


Figure 2. Architecture of Single Shot Scale-invariant Face Detector (S^3FD). It consists of **Base Convolutional Layers**, **Extra Convolutional Layers**, **Detection Convolutional Layers**, **Normalization Layers**, **Predicted Convolutional Layers** and **Multi-task Loss Layer**.

Just like an SSD we also use the anchor matching strategy in our model in order to detect multiple faces in a box or a layer if they were to exist. However, in SSDFace model we found that this strategy failed to recognize faces of smaller or medium in size, hence in this model we generate anchors for each layer of variable scales thereby detecting images of all sizes.
The scales used here are from 16 to 256 based solely on the receptive field of each of the predicted layers as shown below

Detection Layer	Stride	Anchor	RF
conv3_3	4	16	48
conv4_3	8	32	108
conv5_3	16	64	228
conv_fc7	32	128	340
conv6_2	64	256	468
conv7_2	128	512	724

Table 1. The stride size, anchor scale and receptive field (RF) of the six detection layers. The receptive field here is related to 3×3 units on the detection layer, since it is followed by a 3×3 predicted convolutional layer to predict detections.

We implement two new methods to make it a scale invariant model as shown below

1.Scale Compensation Anchor matching

Since the face scales are continuous while the anchor scales are fixed or discrete , we propose a new anchor matching strategy wherein we adjust the threshold from 0.5 to 0.35 as the IOU threshold and match those anchors that are greater than the above mentioned threshold, and instead of discarding the remaining anchor boxes we then consider those anchor boxes whose IOU is greater than 0.1 with the ground truth boxes and from these boxes we only select the top N boxes from this list where N here is the average number of boxes selected which satisfied the first threshold. In doing so we capture more faces if any in the image and thereby also capturing small or background images

2.Max Out Background Label

Since the boxes generated will be biased towards the background class hence considering all the boxes which correspond to the background label would make the model more inclined to predict boxes for the background class instead of foreground class, and from the below table we find that the number of boxes generated for the background class is more for the earlier layers in VGG16 model since it does not contain enough information to detect face images

Position	Scale	Number	Percentage (%)
conv3_3	16	25600	**75.02**
conv4_3	32	6400	18.76
conv5_3	64	1600	4.69
conv_fc7	128	400	1.17
conv6_2	256	100	0.29
conv7_2	512	25	0.07

Table 2. Detailed information about anchors in a 640×640 image.

Hence in order to fix this imbalance problem , we choose to consider a certain amount or ration of the negative class or background boxes by first determining the total amount of background boxes and sorting them according to their softmax scores and only choosing top k boxes from them way, this way we train the model to differentiate between a background or foreground class by training on a subset of the background class boxes. This concept is very similar to Hard negative mining which is implemented in a basic SSD model as well wherein we train only on a certain subset of the background bounding boxes. The ratio used in this model is 3:1 ratio, where for every x positive box we detect we consider 3x negative boxes to train the model

Loss Function:

For the above model we use a softmax classification loss and smooth l1 loss as the regression loss for the box coordinates. Since the classes are biased, we balance the softmax loss by multiplying it by a constant term lambda in our model which we have chosen as 4. While for the regression loss we only take into account for a positive label class i.e. (Face class) if not this loss is zero for background classes as we don't need to predict bounding boxes for background classes

The loss function mathematically is as shown below

$$L(\{p_i\},\{t_i\}) = \frac{\lambda}{N_{cls}}\sum_i L_{cls}(p_i,p_i^*) + \frac{1}{N_{reg}}\sum_i p_i^* L_{reg}(t_i,t_i^*),$$

Where pi and ti are the class probabilities and the box coordinates.
While pi and pi* indicates the class probabilities for background and foreground class

Face Recognition:
Once the faces are detected and we receive the bounding boxes, these bounding boxes can be of higher width and height than the size of the image. This can cause problems while performing recognition. Therefore, we only choose the ones that are lower than the dimensions of the image. Now, we used the keras-vggface library that provides the pre-trained VGG Model to create a face recognition model. The pre-trained VGG Face model consists of 22 layers and 37 deep units. The structure of the model is demonstrated below.

VGG Face model extracts the activation vector of the fully connected layer in the CNN architecture. It uses the two subspaces learning methods, namely, linear discriminate analysis (LDA) and principal component analysis (PCA) to learn the subspace of the activation vectors for face recognition, this helps in obtaining compact representation and improves performance.

To the model we pass in the faces detected from face detector model as an input. To this we perform predictions using the *'model.predict()'* method. This gives me the probability of a given face belonging to one or more known celebrities. Once a prediction is made, the class integer can be mapped to the names of the celebrities and the top five names with the highest probability can be retrieved.

5. DATASET AND DATA PREPROCESSING

To detect the faces, we use **WIDER FACE [2]**. WIDER FACE has 32,293 images and labels 393,703 faces with high degree of variability in scale, pose and occlusion. The database is split into training 40%, validation 10%, and testing 50% set. The images are divided into easy, medium, and Hard subsets according to the difficulties of the detection. The images and annotations are available online. Our Single Shot Scale-invariant Face detector is trained on the training set and tested on validation set. We were not able to work on the testing set because of the time constraints.

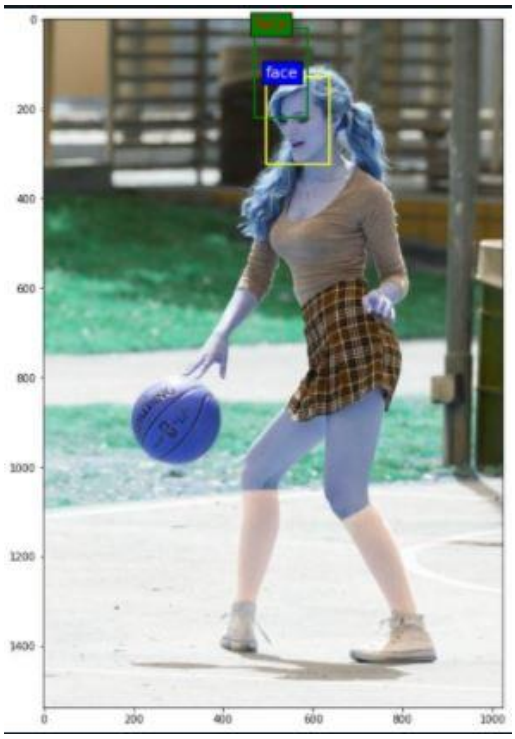
In the WIDER FACE dataset true bounding boxes, class labels and file names are provided in a MATLAB file. Therefore, we wrote a function to separate out the true bonding boxes, class labels and file names in separate variables for train, validation, and test data. The data generator requires the file to be in csv format in the following order: class labels, bounding boxes and file names, therefore, write the variable to csv in this order.

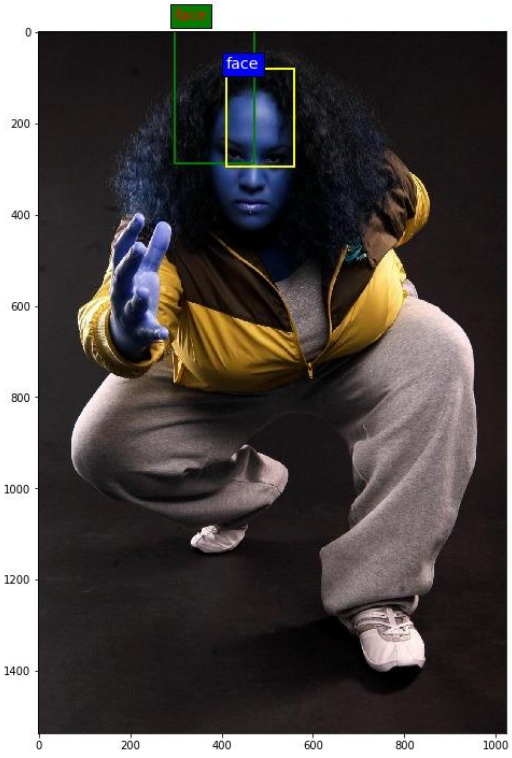
Once the WIDER FACE dataset is used to train the face detector, we use the **IMDB WIKI [3]** dataset to recognize the face of famous celebrities using our pre trained deep learning model. The IMDB-WIKI consists of 469,723 face images from 20,284 celebrities from IMDB and 62,328 from Wikipedia, thus 523,051 in total.

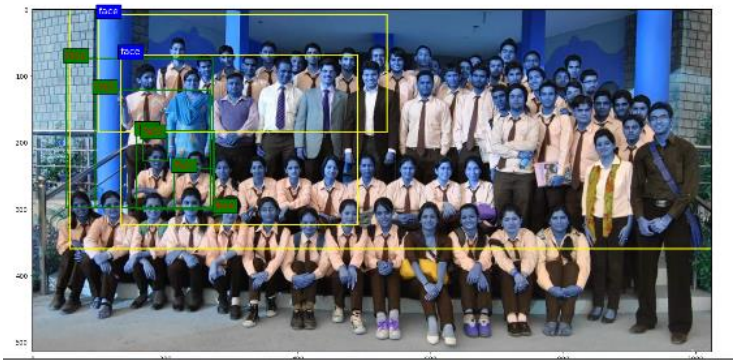
6. MODEL EVALUATIONS

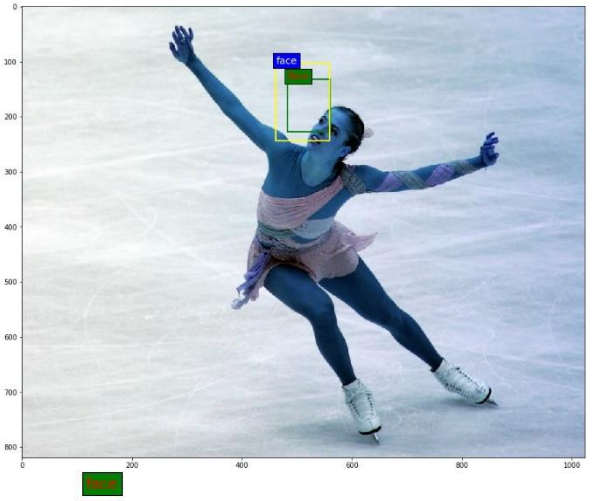
Face Detector:

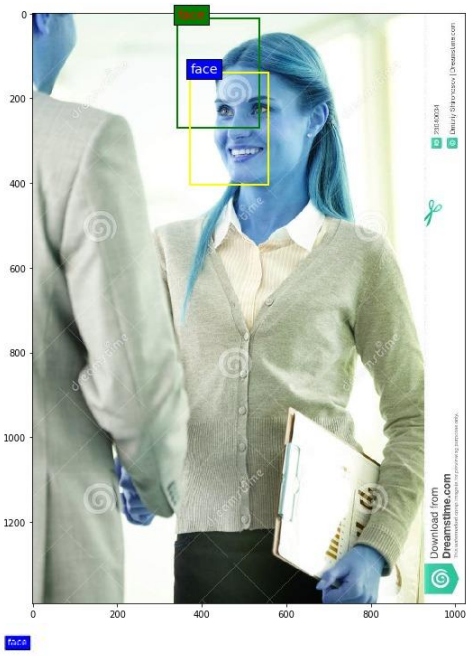
We perform predictions on the validation dataset of WIDER FACE dataset. Below are few of the images with their predicted bounding boxes



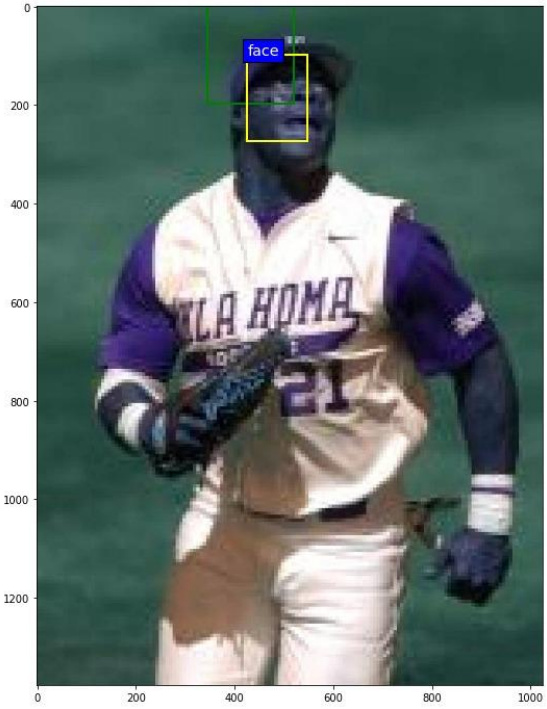



face
face


face
face


face
face
Download from
Dreamstime.com

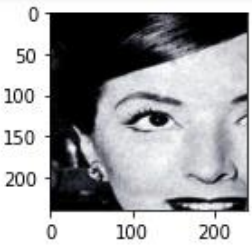
face


face


face

Here, green boxes are true bounding boxes and yellow are the predicted bounding boxes. Looking at these images we can say that the model predicts faces with pose, occlusion, expression etc. However, in some cases it creates bounding boxes larger than the image size and doesn't work at its best on images with multiple faces. The model was trained for 155 epochs with steps per epoch of 403.

Face Recognition:
We perform predictions on the IMDB-WIKI dataset. Below we show few of the faces extracted from the images using the face detector and the top five celebrity names with the highest probability for the face



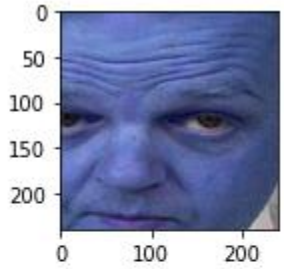
"b' Agustina_Cherri': 0.911%"

"b' Coraima_Torres': 0.865%"

"b' Carman_Lee': 0.789%"

"b' Samantha_Eggar': 0.764%"

"b' Rika_Zara\\xc3\\xaf': 0.755%"



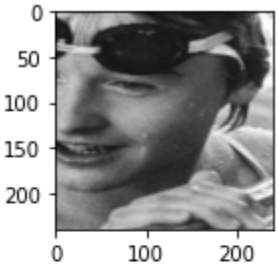
"b' Jack_Bauer': 11.328%"

"b' John_C._Reilly': 10.744%"

"b' Gigi_DAlessio': 5.332%"

"b' Adam_Housley': 5.235%"

"b' Ellie_Simmonds': 4.921%"



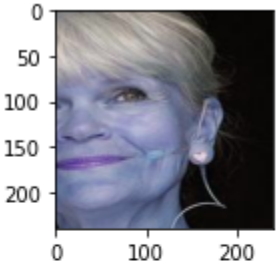
"b' Ekaterina_Makarova': 12.794%"

"b' Emma_Moffatt': 3.405%"

"b' Cara_Dillon': 2.299%"

"b' Fionnuala_Britton': 1.760%"

"b' Allison_Mack': 1.703%"



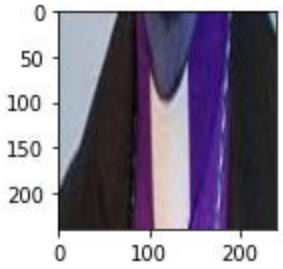
"b' Zo\\xc3\\xab_Ball': 13.671%"

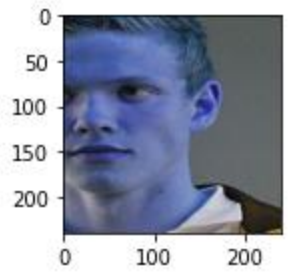
"b' Martha_Coakley': 8.274%"

"b' Ma\\xc5\\x82gorzata_Braunek': 4.157%"

"b' Sherri_Coale': 3.161%"

"b' Tjitske_Reidinga': 2.287%"





"b' Natsumi_Matsubara': 0.435%"

"b' Bob_Beckel': 0.368%"

"b' Venke_Knutson': 0.281%"

"b' Nurul_Izzah_Anwar': 0.275%"

"b' Peter_Altmaier': 0.222%"

"b' Fabian_Hamb\\xc3\\xbcchen': 20.083%"

"b' Jonas_Jerebko': 4.521%"

"b' Marcell_Jansen': 4.460%"

"b' Koen_Verweij': 2.663%"

"b' Max_Riemelt': 2.051%"

Form these predictions we can see that overall, the model does a decent job in recognizing faces. Although, sometimes the model doesn't predict the face correctly, however, still tries to recognize the celebrity.

7. ISSUES FACED

Wider face dataset has a lot of variance and training the model for convergence takes a lot of resources and time. Additionally, the anchor masking algorithm is very promising however, the number of boxes to consider after the first stage is also a hyper parameter to be found in order to make it scale invariant

The model however does predict multiple faces even when its occluded however extremely small images are ignored such as a gathering or a parade. Which can be optimized given enough time by tweaking anchor masking algorithm.

Unfortunately, we were not able to deploy our model on the DeepLens. We believe it is due to some network issues. We were not able to connect our laptop to DeepLens Wi-Fi network.

8. FUTURE IMPROVEMENTS

We believe training out face detector model on some of the IMDB-WIKI dataset can have a drastic change in the predictions. Additionally, training for more epochs and using high quality GPU can reduce the training time and give better predictions.

9. REFERENCES

1. https://openaccess.thecvf.com/content_ICCV_2017/papers/Zhang_S3FD_Single_Shot_ICCV_2017_paper.pdf
2. https://www.tensorflow.org/datasets/catalog/wider_face
3. https://data.vision.ee.ethz.ch/cvl/rrothe/imdb-wiki/
4. https://keras.io/examples/vision/retinanet/
5. https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab
6. https://github.com/pierluigiferrari/ssd_keras