Real-Time Face Recognition

| Gaurav Bajaj | Shubham Jain |
| --- | --- |
| *School of Engineering and Applied Science* | *School of Engineering and Applied Science* |
| *Ahmedabad University* | *Ahmedabad University* |
| [gaurav.b@ahduni.edu.in](mailto:Gaurav.b@ahduni.edu.in) | [shubham.j@ahduni.edu.in](mailto:shubham.j@ahduni.edu.in) |

| Raj Chauhan | Raj Gariwala |
| --- | --- |
| *School of Engineering and Applied Science* | *School of Engineering and Applied Science* |
| *Ahmedabad University* | *Ahmedabad University* |
| raj.c1[@ahduni.edu.in](mailto:Gaurav.b@ahduni.edu.in) | raj.g@ahduni.edu.in |

***Abstract* - Face recognition is a biometric technology used for recognizing identity through the human faces. In this paper, a Real-time Face Recognition Model is developed using MobileNetV2 and SqueezeNet Convolution Neural Network. A comparison in terms of time taken to recognize facial images and accuracy is also performed. Data of 105 celebrities from Kaggle was used to train both models to classify images. Embedding vectors for each human face are generated (while training) in both models and are used to identify and recognize human faces in a cross-platform camera like a laptop webcam or mobile camera. Emphasis is laid on developing a fast and accurate face recognition system.**

***Keywords -* MobileNet\V2, SqueezeNet, Real-time Face Recognition System, Deep Learning, Convolutional Neural Networks**

I. INTRODUCTION

Face recognition is a method of recognizing an individual using their face. Face recognition systems can be used to identify individuals in photos, videos, or in real-time. The main motivation behind building a real-time face recognition system is to enable faster processing and seamless integration in various real-world applications such as public safety systems, authentication systems, retail-store systems, etc.

During the initial days, the speed of recognition of human faces in machines was slow and the accuracy seemed to be lower than manual recognition. However, with the development of deep learning and the increased application of Convolution Neural Network (CNN), the accuracy and speed of human-face recognition by machines are greatly improved.

We have tried to build a real-time face recognition system using the MobileNet and SqueezeNet models which are capable of detecting and recognizing images from cameras (mobile or laptop) accurately. In real-time, based on these models, we have developed and compared the face recognition systems in order to make an accurate facial recognition system.

II. LITERATURE SURVEY

In this paper, we have tried to develop an accurate real-time face recognition system. We have developed face recognition systems using both MobileNetV2 and SqueezeNet Model. The main reason for choosing MobileNetV2 and SqueezeNet was that both the models are developed to provide low latency as they both use less no of parameters in the network. The architecture and usage of fewer parameters in order to provide high latency for both the models are explained in the section below.

MobileNet-v2: It is a 53-layered deep convoluted neural network that can categorize images into about 1000 different objects, including keyboards, mice, and various animals. As a result, this network has learned a variety of rich feature representations for a range of photos. The input size of an image for this network is 224x224 pixels. MobileNet models contain depthwise separable convolution blocks, each block in MobileNet V2 has a 1x1 expansion layer, as well as depthwise and pointwise convolution layers. Each block’s output is a bottleneck in the bottleneck residual block. Before going into depthwise convolution, a 1x1 expansion convolutional layer extends the number of channels based on the expansion factor in the data. The residual connection exists to help the flow of gradients through the network. Batch normalization and the ReLU6 activation function are included in each layer of MobileNet V2. the projection layer’s output, on the other hand, lacks an activation function.

MobileNet-v2 Architecture: There are two main types of blocks: bottleneck blocks with stride 1 and stride 2 blocks. As previously mentioned, each of the blocks has three levels. The bottleneck block will not have a residual connection if stride=2 is utilized for depthwise convolution.

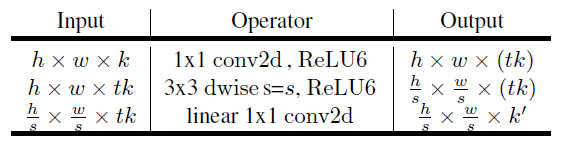


Fig-1. Bottleneck residual block

The expansion factor used in most of the layers is 6. Therefore, the output has 64\*6 = 384 channels if the input has 64 channels.

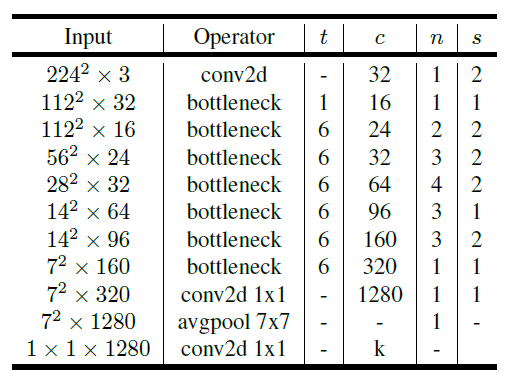


Fig-2.MobileNet V2 Overall Architecture

SqueezeNet: It has AlexNet level accuracy and 50x fewer parameters and the model size is less than 0.5MB. Three architectural design strategies were adapted to attain AlexNet accuracy with fewer parameters. (1) Replaced 3x3 convolution filters with 1x1 filters as 1x1 filters have 9x fewer parameters than 3x3 convolution filters. (2) Decreasing the number of input channels to 3x3. Consider a convolution layer that only has 3x3 filters. The total number of parameters (in this layer) = (number of input channels) \*(number of filters)\*(3\*3). So, in order to decrease the number of parameters, input channels are also decreased to 3x3 filters using a squeeze layer (described below). (3) Giving convolution layers large activation maps by downsampling late in the network. The intuition behind this idea was that large convolution maps can lead to higher classification accuracy.

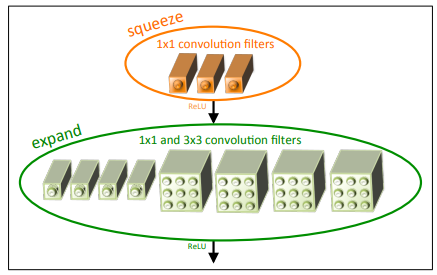


Fig-3 Fire Module in SqueezeNet

Fire Module in SqueezeNet: It consists of a squeeze convolution layer that has only 1x1 filters. This layer feeds into a ‘expand layer’ having a mix of 1x1 and 3x3 convolution filters (illustrated in fig-3). There are three hyperparameters (tunable dimensions) in this Fire module: s1x1, e1x1, and e3x3. s1x1 is the number of 1x1 filters in the squeeze layer. e1x1 and e3x3 are a number of 1x1 and 3x3 filters in the expand layer. The use of 1x1 filters in Fire modules is based on strategy-1. s1x1 is set to be less than (e1x1 + e3x3) in order to limit the number of input channels to 3x3 layers as discussed in strategy-2.

SqueezeNet Architecture:

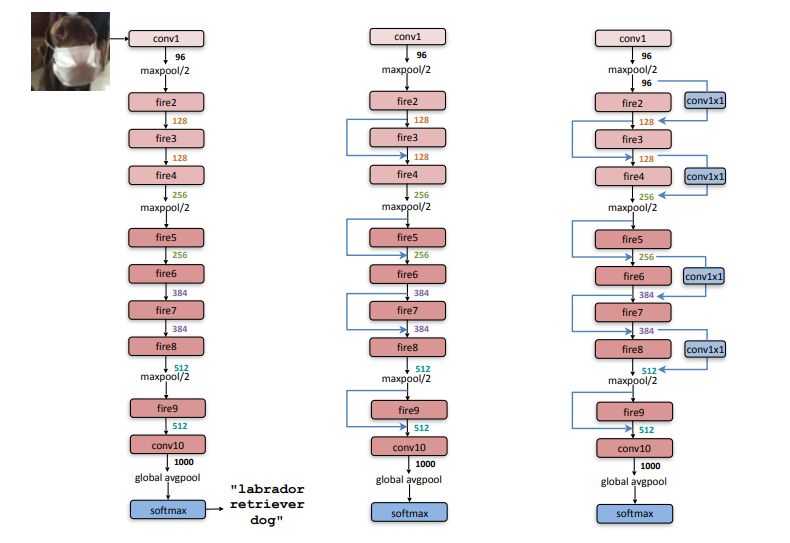


Fig-4 Macroarchitectural view of SqueezeNet

At the start of the SqueezeNet architecture, there is a single convolution layer (conv1). It is followed by 8 fire modules (fire 2-9) and it ends with a final convolution layer (conv10). The number of filters is increased gradually from the start to the end of the network. In SqueezeNet, max-polling with a stride of 2 is performed after layers conv1, fire4, fire8, and conv10. These late placements of max-polling are because of strategy-3 (which was discussed above).

III. IMPLEMENTATION

In order to decide upon the model to serve our metrics, i.e. speed and accuracy, we looked at 2 different models developed for face classification - MobileNetV2 and Squeezenet. We attempted to implement the feature of real-time face recognition on both models and compared their results by introducing a slight twitch in the last layer of the model.

For training both models, we have used the 105\_classes\_pins dataset available over Kaggle. It is a dataset that contains about 17534 images of about 105 different celebrities. The best thing about this dataset is that it is well cropped and labeled which does not call for any large pre-processing. The dataset was divided into training and testing sets in the ratio of 4:1 (80% and 20%). We performed a bit of Exploratory Data Analysis to get an overview of the dataset, for understanding model input size and for Data Augmentation. Data Augmentation would help in learning facial features by the model efficiently. During Data Augmentation, we introduced augmentations like shear, zoom, horizontal flip, rotation by a range of angle, width shift, and height shift.



Fig-5. Sample Faces from the dataset

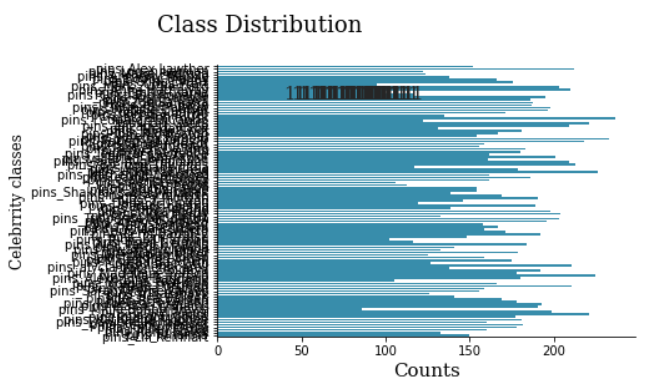


Fig-6. Face Class Distribution Analysis

Both the models were trained using the splitted training-testing sets and their training and validation accuracy and loss were monitored over 60 epochs. Extra layers such as the Global average pooling layer and Dropout layers were added to improve the accuracy. Both were trained using categorical cross-entropy as loss function and Adam optimizer with a learning rate of 3\*10-4 for the Squeezenet model and 1\*10-4 for the MobileNetV2 model. We have also used ReduceLROnPlateau as a callback API to reduce the learning rate when validation loss stagnates during training. This is done in order to prevent the overfitting of the models. The weights of both the models are initialized with their image net weights and trained using transfer learning. The results of accuracy and loss over epochs for both models are shown in the results section.

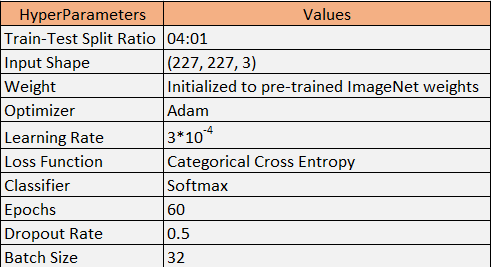
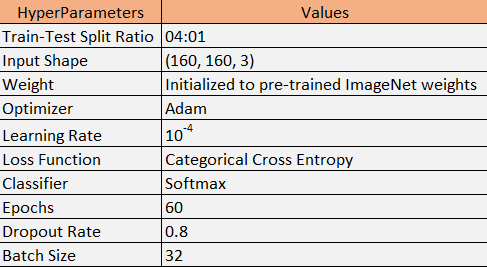


Fig-7. Squeezenet Model Hyperparameters

  
Fig-8. MobileNetV2 Model Hyperparameters

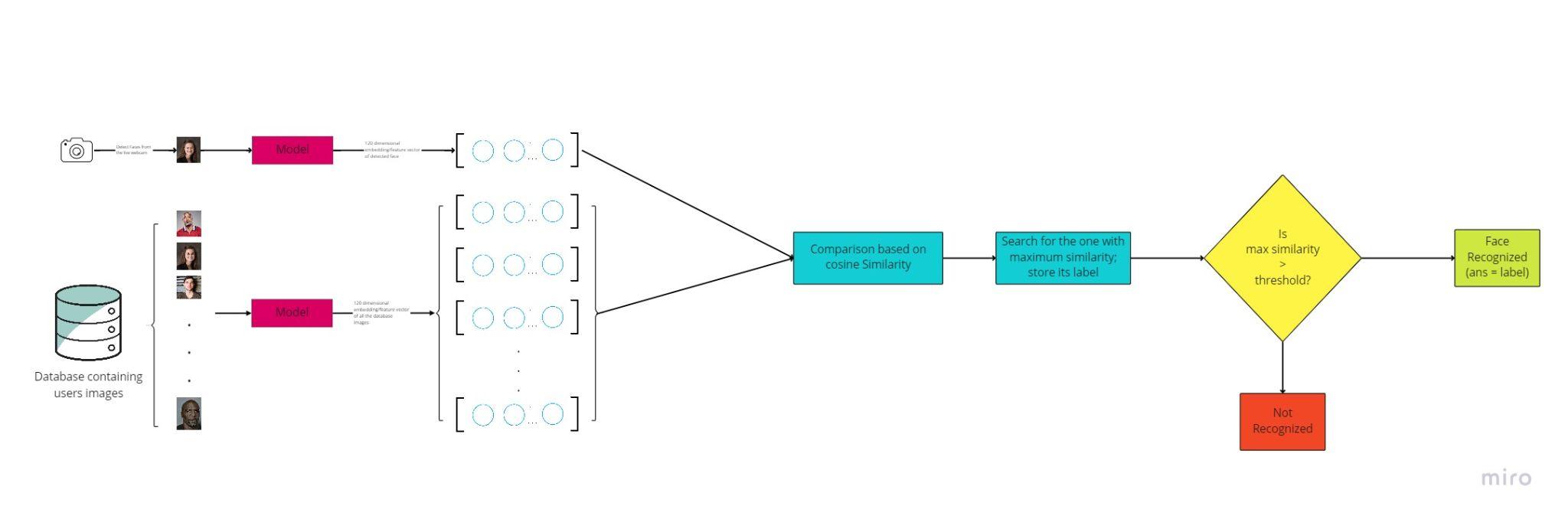


Fig-9. Face Recognition Process Block Diagram

Once the models are successfully trained, we would remove the last layer of the model with a 120-dimensional Dense Layer. This is done in order to obtain the 120-dimensional feature vector for the faces inputted into the model. These 120-dimensional vectors would contain facial features like nose, eyebrows, eyes, iris, lips, nostrils, etc. Once both the models are trained successfully, we saved their weights for future use. The above block diagram shows the flow in the process of Face Recognition in real-time through cameras. For different people/users in the database for whom we would like to recognize faces in real-time, we would feed their cropped faces into the trained models and obtain their 120-dimensional embedding/feature vectors. All these embedding vectors are saved for similarity comparison in the future.

Now comes the part of detecting faces from the frames obtained through the camera. This is done using Haar Cascade Algorithm. Haar Cascade face detection technique is really useful in detecting faces, eyes, upper and lower body. The faces detected in the frame are inputted to the model to obtain their own 120-dimensional feature vector. We now do a similarity comparison between each of the stored vectors of the database images and the newly obtained vector of the detected face. We measure this similarity using cosine similarity which aims at finding the closeness between 2 vectors based on the angle between them. More is the cosine similarity, more is the closeness between 2 vectors and smaller is the angle between them. We find the vector amongst the vectors of the database images with maximum similarity with the vector of the face detected in the camera. The maximum similarity is then compared with a threshold value to evaluate if its a proper recognition or not, i.e., if the value is less than the threshold, we can say that it is a “Not Recognized” face, otherwise, it’s a recognized face and the person whose face is recognized is the one corresponding to the vector with maximum similarity in the database. We’ve currently set the threshold at 0.8.

IV. RESULTS

1. SqueezeNet:

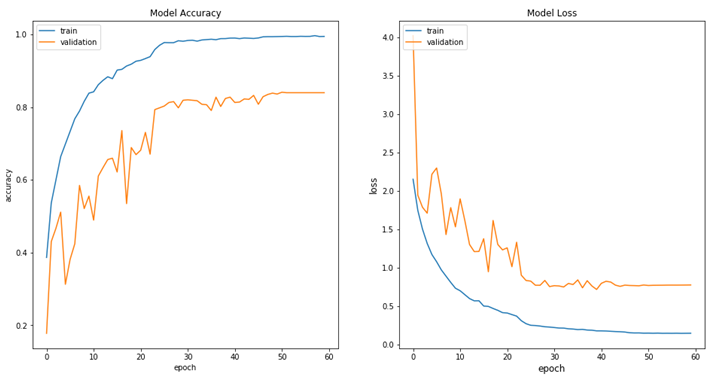
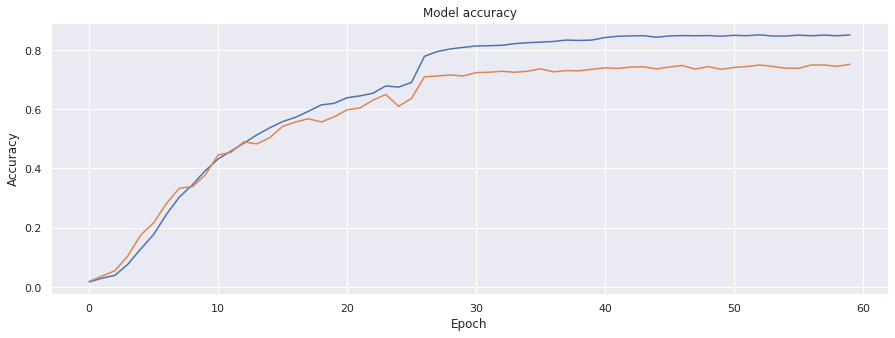


Fig-10: SqueezeNet - Accuracy vs Epoch

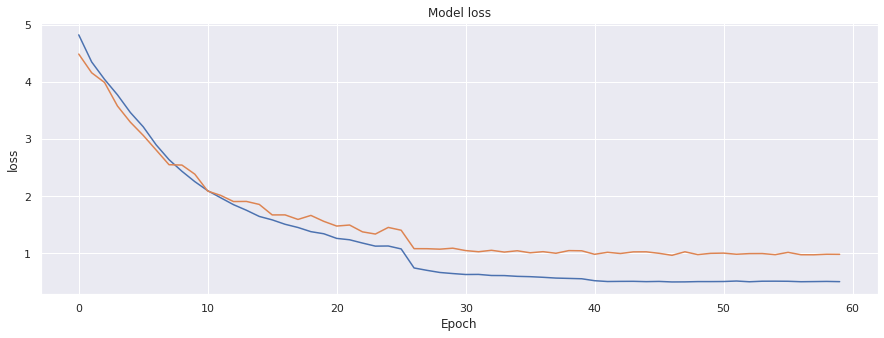
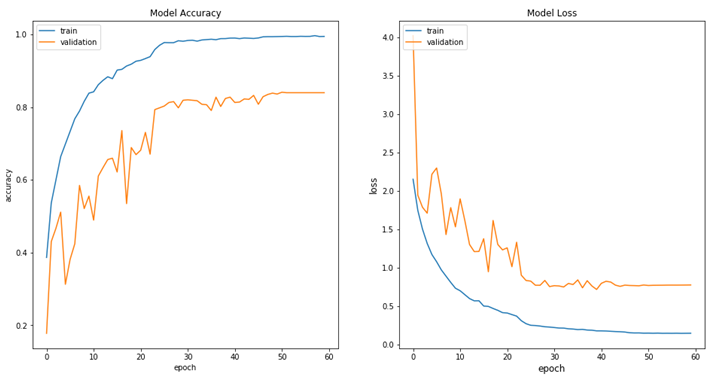
****

Fig-11: SqueezeNet - Loss vs Epoch

1. MobileNetV2:

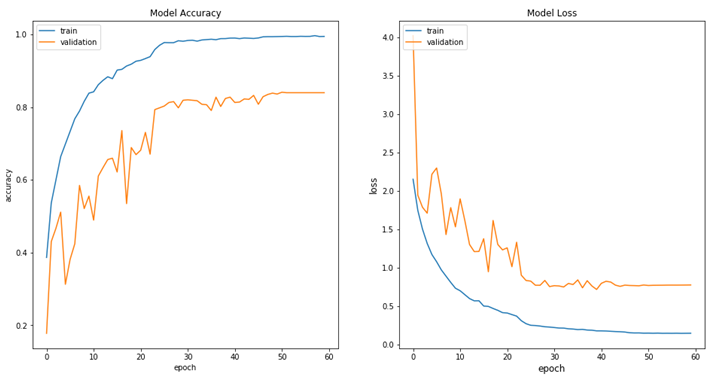
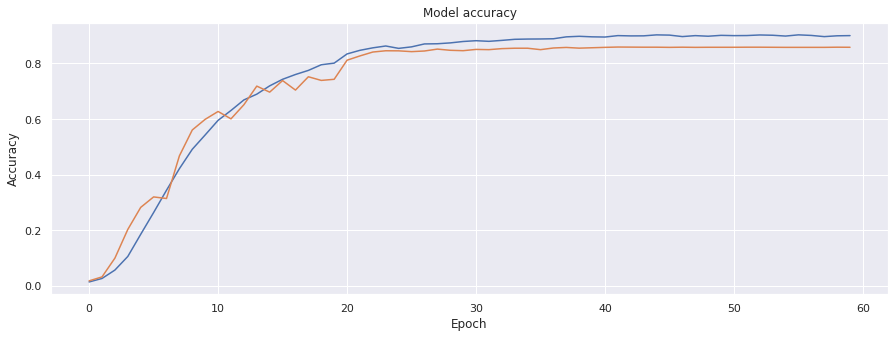


Fig-12: MobiletNetV2 - Accuracy vs Epoch

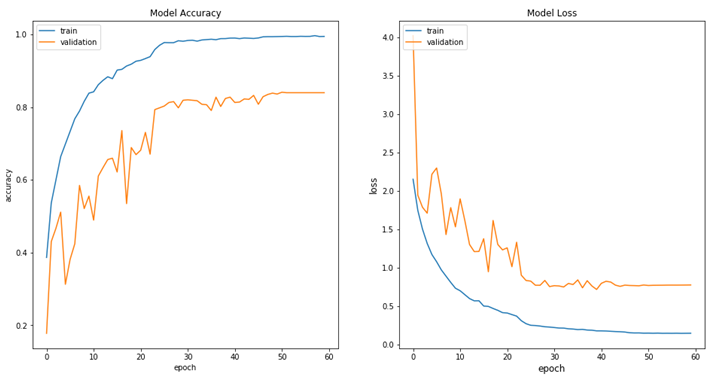
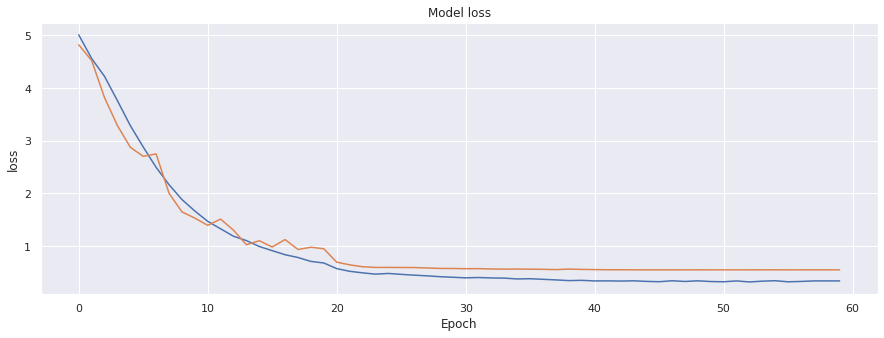


Fig-13 : MobiletNetV2 - Loss vs Epoch

On fitting both models, we observed that Training Accuracy and Training Loss rate in MobileNetV2 were about 90% and 3% respectively till the last epoch, while having 85% and 5% respectively validation accuracy and validation accuracy. Whereas in SqueezeNet, Training accuracy and training loss was about 85% and 3% respectively, while validation accuracy and loss were 75% and 10% with major differences. We can observe in both graphs that there is some overfitting but this will not affect the results as we reduced overfitting by a drop-out layer and improved Data Augmentation. Moreover, since we are using the model for extracting facial features, classification accuracy and loss won’t affect the model’s usage. So compared to the Mid-semester’s result the amount of overfitting is reduced by a great extent in MobileNetV2. The model size of SqueezeNet is smaller than MobileNet having fewer parameters and fewer weights to train, which gets results faster compared to MobileNet.

Further, we used both models to recognize faces to conclude which model is better and faster in terms of performance in recognizing images of people. In figure-14, we can see that the model was able to recognize “Raj Chauhan” but was unable to recognize another person, but on the flip side in fig-15, we can see that both people “Raj Chauhan” and “krutin” were recognized.

Despite that, both of the models were fluctuating results and were not continuously showing accurate results but we observed that Images with SqueezeNet had better and faster accuracy. Also, the SqueezeNet model, being smaller in size, showed fast face recognition (less latency) compared to MobileNet model.

The input images in the MobileNetV2 are a matrix of 160x160, while they are of size 227x227 in SqueezeNet. This could be another reason for better functioning of SqueezeNet as there is less loss of spatial information. Hence, we get more precision and better accuracy in more frames per second.

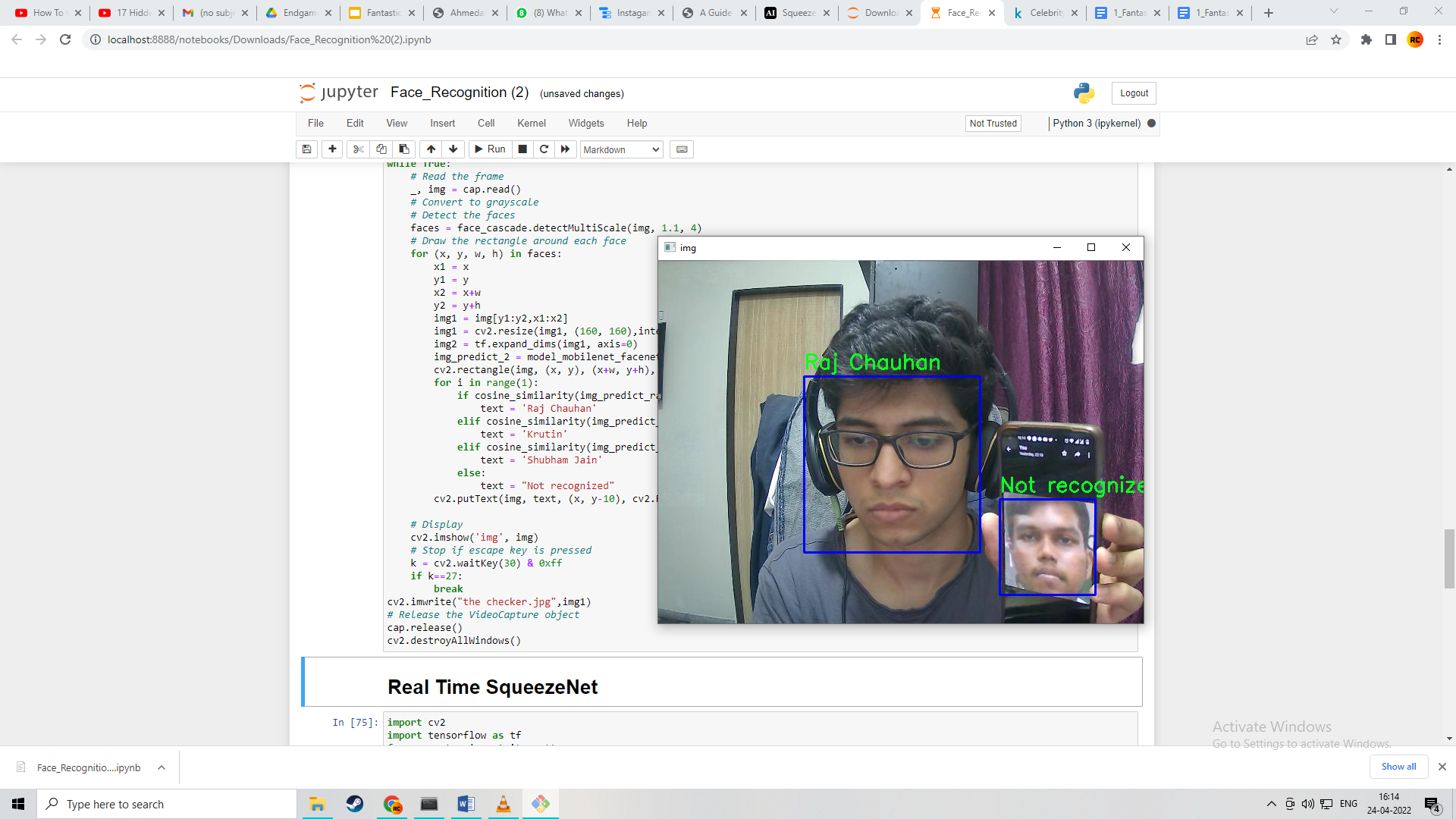


Fig-14: MobileNetV2 Face Real Time Face Recognition

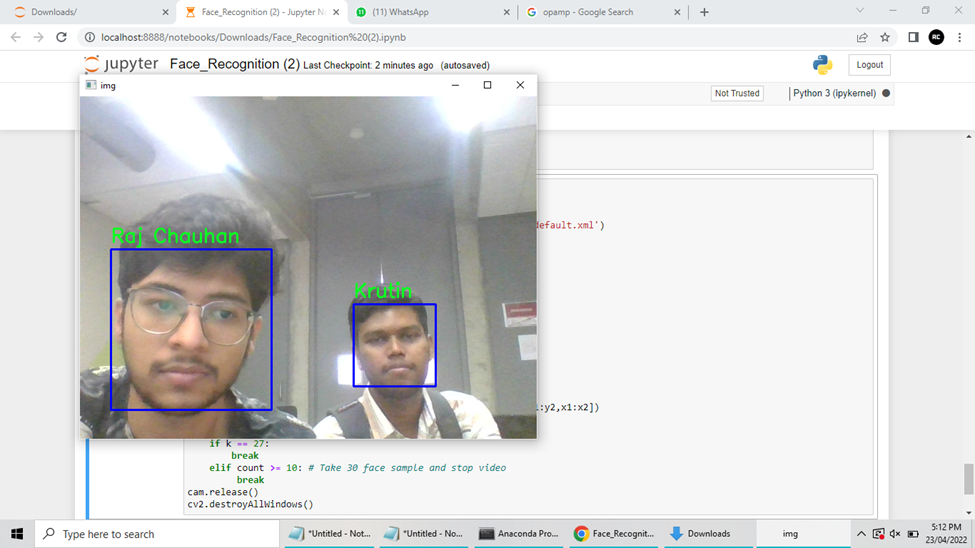


Fig-15: SqueezeNet Face Real Time Face Recognition

V. CONCLUSION

As observed in the above results, we can conclude that even though both the models were trained on same parameters such as epochs and datasets, SqueezeNet gives better performance in terms of accuracy, efficiency, speed and recognition with multiple people as the input size of the image is bigger than MobileNetV2. However, in spite of high training and validation accuracy, we still faced false negatives and some errors in recognition when multiple people are in one frame together. Due to time constraints, we were not able to implement triplet loss in the models, which may also be a leading factor in less accuracy and fluctuating recognitions. Also, we could further improve the system by setting proper thresholding after doing tons of experimentation with various kinds of images and different conditions. On the contrary, we were focussing on two real-life problems which were Light Illumination and the Distance of the image from the camera. We could not implement the illumination problem but we managed to resolve the distance problem. No matter how far the person is from the camera, only the face image will be resized and collected, and processed to recognize the face.

Finally, all members of the Fantastic Four team think that they learned and experienced many new things whilst doing this project, there were many aspects that were new to us, and we were not aware of as all of us were not from machine learning background, still, we tried our best to focus and work on the project, as we worked more on the project we got more interested and encouraged to know and explore more about the Deep learning and Face recognition for the start. We may not have deployed this project as a full-fledged and 100% accurate project, but as beginners, we think that this is the best we could have done and we are very much satisfied with what we have achieved.

VI. REFERENCES

1. Simon Low, “SqueezeNet and MobileNet: Deep learning models for mobile phones, 03-May-2018, AI in Practice. Available: https://aiinpractice.com/squeezenet-mobilenet/#:~:text=Furthermore%2C%20MobileNet%20achieves%20really%20good,using%20transfer%20learning%20or%20distillation
2. A. Michele, V. Colin, and D. D. Santika, “MobileNet convolutional neural networks and support vector machines for Palmprint recognition,” *Procedia Computer Science*, 01-Oct-2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919310658. [Accessed: 24-Apr-2022].
3. S.-H. Tsang, “Review: MOBILENETV2 - light weight model (image classification),” *Medium*, 01-Aug-2019. [Online]. Available: https://towardsdatascience.com/review-mobilenetv2-light-weight-model-image-classification-8febb490e61c#:~:text=MobileNetV2%20Overall%20Architecture&text=In%20typical%2C%20the%20primary%20network,and%20uses%203.4%20million%20parameters. [Accessed: 24-Apr-2022].
4. F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “Squeezenet: Alexnet-level accuracy with 50X ... - arxiv.org,” SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and &lt;0.5MB model size, 2017. [Online]. Available: https://arxiv.org/pdf/1602.07360v3.pdf. [Accessed: 24-Apr-2022].
5. S.-H. Tsang, “Review: SqueezeNet (image classification),” Medium, 22-Apr-2019. [Online]. Available: https://towardsdatascience.com/review-squeezenet-image-classification-e7414825581a. [Accessed: 24-Apr-2022].
6. H. Gao, “Notes on squeezenet,” Medium, 25-Jan-2018. [Online]. Available: https://medium.com/@smallfishbigsea/notes-of-squeezenet-4137d51feef4. [Accessed: 24-Apr-2022].
7. Avidrishik, “A guide to squeezenet architecture: Compressed neural network,” Medium, 07-Sep-2021. [Online]. Available: https://medium.com/@avidrishik/squeezenets-architecture-compressed-neural-network-7741d24ca56f. [Accessed: 24-Apr-2022].
8. Github Repository: <https://github.com/Jimmy290901/CSE541-Computer-Vision-2022-FantasticFour>