FER-M: Facial Expression Recognition on Mobile Devices

Sanket Naik[1], Sahil Nair[2], Manish Chavan[3], Sachin Barahate[4]

[1-3]Department of Computer Engineering, Vasantdada Patil Pratishthan's College of Engineering, Mumbai, Maharashtra, India
[4]Professor, Department of Computer Engineering, Vasantdada Patil Pratishthan's College of Engineering, Mumbai, Maharashtra, India

---***---

Abstract – *The task of facial expression recognition has been studied extensively. However, not a lot of research focuses on the deployment of the various deep-learning based approaches used for this task. We aim to compare different Neural Networks and the factors that should be considered when deploying these models on mobile devices.*

***Key Words*: Deep Learning, Facial Expression Recognition, CNN, ResNet, Computer Vision, PyTorch, OpenCV, Android.**

1. INTRODUCTION

We as humans convey a lot of messages through our expressions. The facial expression of a person can often give us a brief idea about their emotional state and their mood. Humans can easily understand facial expressions of another person with minimum delay and a high accuracy. However, achieving the same level of accuracy and speed has been a challenge for machines. Recent advances in the field of Deep Learning have made the machines more accurate and has significantly lowered the prediction latency.

There has been a lot of research focussed on the task of machine learning using modern techniques such as Convolutional Neural Networks, Residual Networks and other Deep Learning based approaches. However, these approaches have mainly focussed on achieving the highest accuracy and have not considered the factors that affect the deployment of such deep-learning based models on mobile devices.

In this paper, we compare various approaches such as Convolutional Neural Networks (CNNs) and Residual Neural Network (ResNet) and try to find the approach which is best suited for the task of facial expression recognition on mobile devices.

2. RELATED WORK

Researchers have extensively studied various approaches for achieving the best performance for the task of facial expression recognition. Approaches such as Convolutional Neural Networks [1], Attentional Convolutional Networks [2], and 2-Channel Convolutional Networks [3] have shown that Convolutional Neural Networks are the best performers for this task. However, due to the complexity of these models, they may not be the best for the task of performing facial expression recognition on mobile devices. Other approaches such as Residual Neural Networks [4] also have similar performance as that of Convolutional Neural Networks

Moreover, the findings of various researchers [5][6] after studying the performance of Convolutional Neural Networks and Residual Neural Networks suggests that models should have a lower number of trainable parameters in order to minimize the size of the final model and have low latency during the prediction process.

3. METHODOLOGY

We developed three different models to get a comprehensive understanding of the factors affecting the deployment of deep learning models on mobile devices. The three models that were developed were – a Shallow Convolutional Neural Network, and a Deep Neural Network, a Residual Neural Network. The Residual Neural Network was based on the ResNet-9 architecture [7]. All models were trained on the same hyperparameters to eliminate the variance that would result from different hyperparameters. We used the one cycle learning rate method [8] where the learning rate of the optimizer starts from a lower value and slowly rises till it reaches the defined maximum value. The learning rate is then slowly reduced till we reach the lower value a few epochs before the end. Finally, in the remaining iterations the learning rate is annihilated way below the lower learning rate value. The hyperparameters used for the training process were as follow:

Table -1: Hyperparameters used for the training process

Parameter	**Value**
Epochs	20
Batch Size	256
Max. Learning Rate	0.008
Optimizer	Adam

The three models were trained using the hyperparameters shown in Table 1 and their performance was compared. The parameters that we compared were validation loss, validation accuracy, training loss and model size. Later, we built an Android app using Flutter to compare the real-world accuracy, size of the app and the prediction latency on mobile devices.

4. DATASET & DATA PREPARATION

The model was trained on the FER-2013 Dataset [9]. This dataset consists of grayscale images of faces of size 48x48 pixel. The faces have already been centered in the images and the faces generally occupy the same volume of space in each image. The dataset consists of a training test and a test set. The training set consist of 28,709 images and the test set consists of 3,589 images. All images in the dataset are labelled and have been categorized into one of the seven given categories where 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise and 6=Neutral.

The images from the dataset were loaded using the DataLoader class of PyTorch. For both sets, the Grayscale transformation was applied which loads all images as grayscale images. This was followed by the ToTensor transformation which converts the images into PyTorch tensors. Then, the Normalize transformation was applied. This transformation normalizes all the images so that all the tensor values are between 0 and 1. Additionally, the RandomHorizontalFlip and RandomCrop transformations were applied to the training set to augment the data. These transformations help to extract more information from the dataset.

5. ANALYSIS

5.1 Model Training & Testing

All models shared some common features such as the Conv Block which was simply a 2D Convolution followed by Batch Normalization and a ReLU activation layer. The Shallow Convolutional Neural Network was built of 3 Conv Blocks, each followed by a 2D Max Pooling Layer. The output was then Flattened and passed through 2 Linear layers, each followed by a ReLU activation. A Linear layer was added as the output layer. The Deep Convolutional Neural Network had a total of 6 Conv Blocks in pairs of two, each pair was followed by a 2D Max Pooling Layer. Then, just like the Shallow CNN, the output was Flattened and passed through 2 Linear layers, each followed by a ReLU activation. A Linear layer was added as the output layer. The ResNet-9 model was based on the standard definition of the ResNet-9 architecture [7].

After the three models were trained for 20 epochs with the hyperparameters as shown in Table 1, their accuracy and loss values were compared. The Training Loss was the lowest for the ResNet-9model and the highest for the Shallow CNN as shown below in Chart 1. Even though isn't an important metric, it does show that the ResNet-9 model might be overfitting to the data.

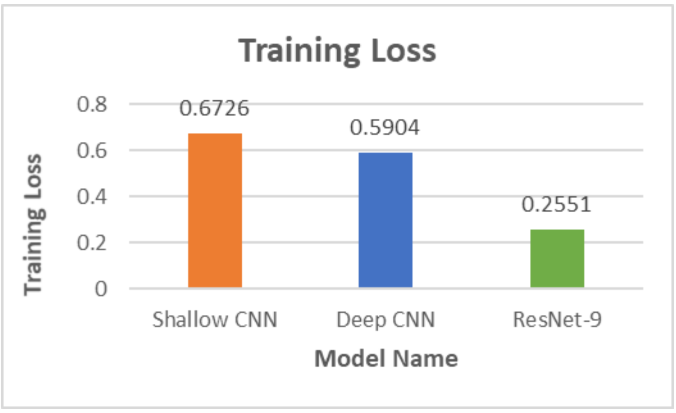

Chart -1: A comparison of the Training Loss

Another loss metric that we compared was the Validation Loss which is the Loss when the model is run on the test set. Here, the Shallow CNN actually performed the best while the ResNet-9 model had the highest loss as shown below in Chart 2. This is further proof of the ResNet-9 model overfitting to the training set.

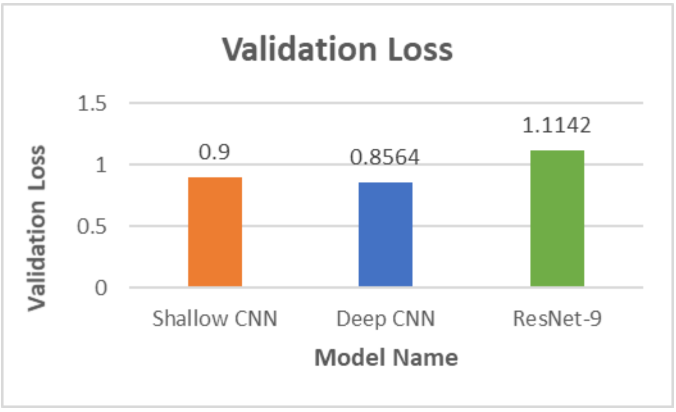

Chart -2: A comparison of the Validation Loss

The Validation Accuracy which is the accuracy of the models on the test set was then compared. We found that the models performed quite similarly with the ResNet-9 model having the highest accuracy and the Shallow CNN having the lowest accuracy as shown below in Chart 3.

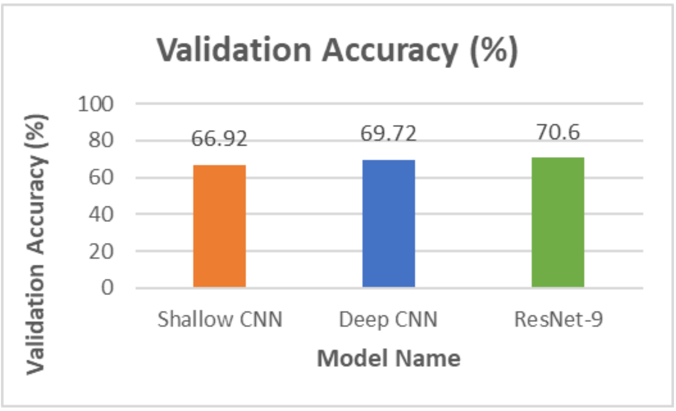

Chart -3: A Comparison of the Validation Accuracy

The final metric that we considered after model training was the model size. This would directly affect the size of the app that the model is deployed to and having a large model would make the app size large. We found that the Shallow CNN and the Deep CNN models were a lot larger than the ResNet-9 model as shown below in Chart 4.

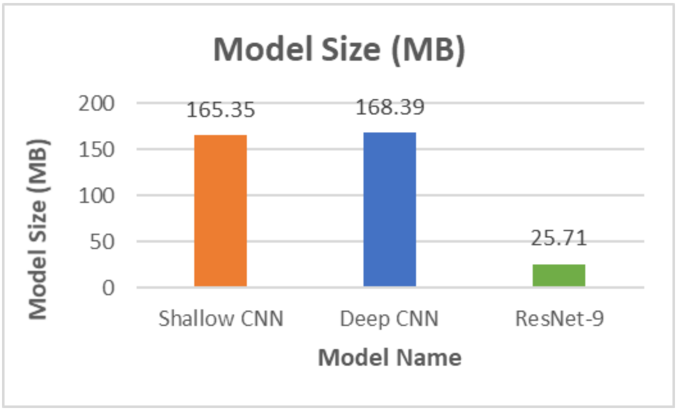

Chart -4: A Comparison of the Model Size

5.2 Deployment on Mobile

After the three models were created, we deployed them to an Android app built using Flutter [10]. The app uses a camera to capture an image (Figure 1). This image is then converted to a grayscale image. We use a Haar Cascade Classifier [11] provided by the OpenCV Android SDK to detect faces in the image. The detected faces are then cropped and resized to get an image of size 48x48 pixels. The resized image is then converted to a tensor using the helper functions provided by the PyTorch Mobile SDK [12]. Finally, we predict the expression on the face by passing the tensor to the model. The predicted expression is then displayed to the user on a new page along with the time taken as shown below in Figure 1.

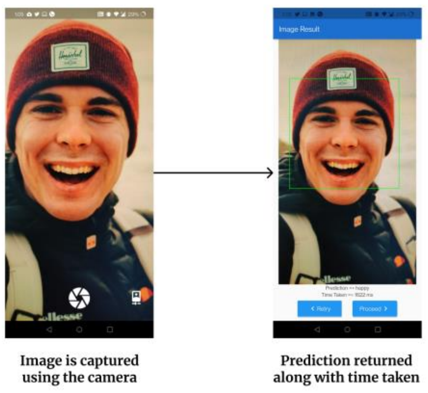
Image Result
Retry
Proceed
Image is captured using the camera
Prediction returned along with time taken

Figure -1: The Prediction process on the Android App

5.3 Results

As a part of the final testing, we deployed the three models through the Android app and tested them. The first point of comparison was the app size. The app size is directly related to the model size and so the results were similar to the results seen in the model size comparison. The ResNet-9 model had the smallest app size and the Deep CNN had the largest app size as shown below in Chart 5. This proves that the complexity of the model has a direct correlation with the app size after model deployment.

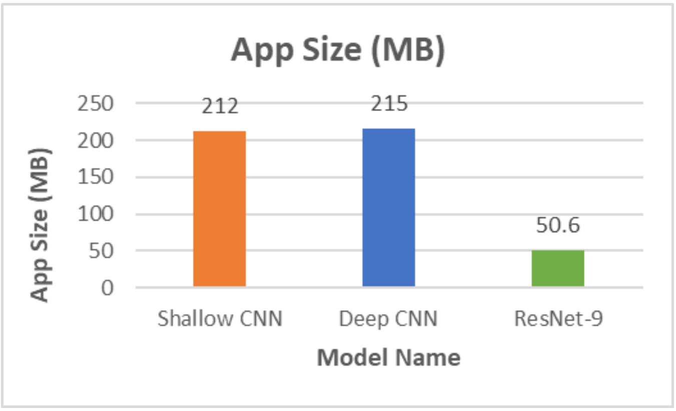

Chart -5: A comparison of the App Size in MB

The next metric that was compared was the average latency. This is the time taken for getting a prediction from the model. We noted the latency for five runs of each model and calculated the average latency for the model. The Deep CNN was the worst performer here. The Shallow CNN and ResNet-9 model had similar latencies and only varied by a small amount as shown below in Chart 6. This proves that a complex model which has more trainable parameters takes more time to get the prediction as compared to a smaller and less complex model.

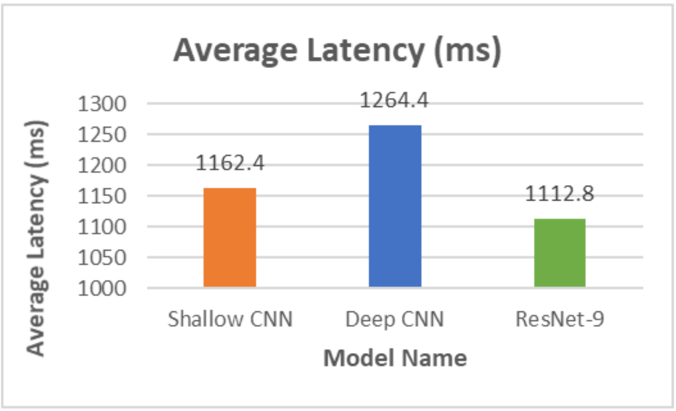

Chart -6: A Comparison the Average Latency (in ms)

Finally, we tested the models on 50 real world images to get the real-world accuracy of the models. To get the accuracy, we checked how many images were correctly predicted by the models by using the simple percentage formula. We observed that the Shallow CNN actually performed the best and the Deep CNN had the worst performance as shown below in Chart 7. This proves that the most complicated model may not be the most accurate one in real-world scenarios.

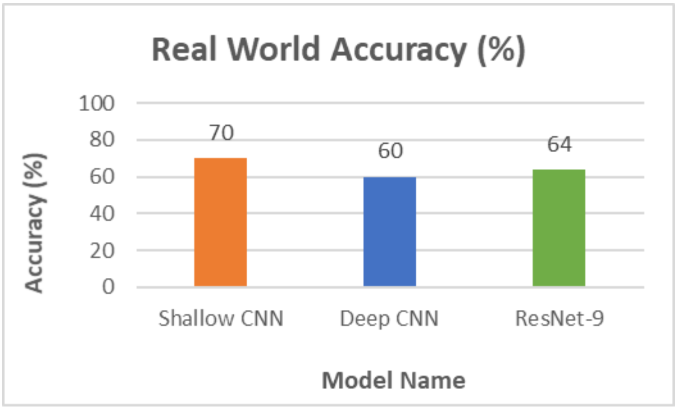

Chart -7: A Comparison of the real-world accuracy

6. CONCLUSION

We developed three different models for the task of facial expression recognition and compared various metrics to determine the most important things to consider when deploying deep-learning based models on mobile devices. Our results have shown that complex and deeper models aren't a good fit for mobile deployment and may actually perform worse than simpler models. Also, the app size directly depends on the model size which means that a smaller model is better suited for deployment on mobile devices. Additionally, a more complex model also increases the latency which is the total time taken to get the prediction from the model.

REFERENCES

[1] Alizadeh, Shima, and Azar Fazel. 'Convolutional Neural Networks for Facial Expression Recognition'. ArXiv:1704.06756 [Cs], Apr. 2017. arXiv.org

[2] Minaee, Shervin, and Amirali Abdolrashidi. 'Deep-Emotion: Facial Expression Recognition Using Attentional Convolutional Network'. ArXiv:1902.01019 [Cs], Feb. 2019. arXiv.org

[3] Hamester, Dennis & Barros, Pablo & Wermter, Stefan. (2015). Face expression recognition with a 2-channel Convolutional Neural Network. 1-8. 10.1109/IJCNN.2015.7280539.

[4] He, Kaiming, et al. 'Deep Residual Learning for Image Recognition'. ArXiv:1512.03385 [Cs], Dec. 2015. arXiv.org

[5] Howard, Andrew G., et al. 'MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications'. ArXiv:1704.04861 [Cs], Apr. 2017. arXiv.org

[6] Luo, Chunjie, et al. 'Comparison and Benchmarking of AI Models and Frameworks on Mobile Devices'. ArXiv:2005.05085 [Cs, Eess], May 2020. arXiv.org

[7] https://myrtle.ai/learn/how-to-train-your-resnet-4-architecture/

[8] Smith, Leslie N. 'A Disciplined Approach to Neural Network Hyper-Parameters: Part 1 -- Learning Rate, Batch Size, Momentum, and Weight Decay'. ArXiv:1803.09820 [Cs, Stat], Apr. 2018. arXiv.org

[9] https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge

[10] https://flutter.dev/

[11] https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

[12] https://pytorch.org/mobile/android/