Machine Learning Engineer Nanodegree

Capstone Project

Rohith Nama

March 1st, 2019

I. Definition

Project Overview

Computer vision seems to be the possible solution for many existing human problems and it is set to change the user experience in day-to-day life. There is a clear growth pattern in it's adaptability. However, there are many limitations in the applications that can be developed for the day-to-day usage. It is because of the lack of infrastructure and the need for enhanced computing capabilities. Recently, this field seems to expand further with the availability of basic infrastructure such as AWS deeplens. Currently, many computer vision applications are aiming at tracking human emotions / experiences and build analytics on top of it- which is very powerful. This capstone project aims at building an end-to-end application using DeepLens to track the basic human emotions[1]. This project can be further expanded to understand human emotions at theaters, hospitals, seminars etc to understand the human emotions and integrate with alexa to tell a joke, fact, quote or any interesting information that can potentially enhance the mood of the user. The goal for this project is to focus on the core part, which is to build a model to detect human emotion.

I'm personally interested in developing the products that integrate computer vision and human experience. In the near future, I would like to pursue this path to develop any complex applications with in the healthcare and user experience domains.

Problem Statement

The core part of the project is to develop a model that is able to detect basic human emotions. This problem is based on the Kaggle Challenge- Challenges in Representation Learning: Facial Expression Recognition Challenge[2].

The goal is to detect the human emotion when the image of the person is passed to a model. Detecting human face and further classifying based on emotion involves great amount of learning by an ML model. The best model that fits for this purpose is convolutional neural network, which is a class of deep learning. CNN's are imitation of neural network in the brain and they are built in a multi layer fashion, where each layer learns the particular patterns. More number of layers helps the model to learn and perform better. However, it also increases the complexity and computational costs. Therefore, the CNN model has to be architected to balance the accuracy and computational costs (both time and resources) based on the purpose of the application.

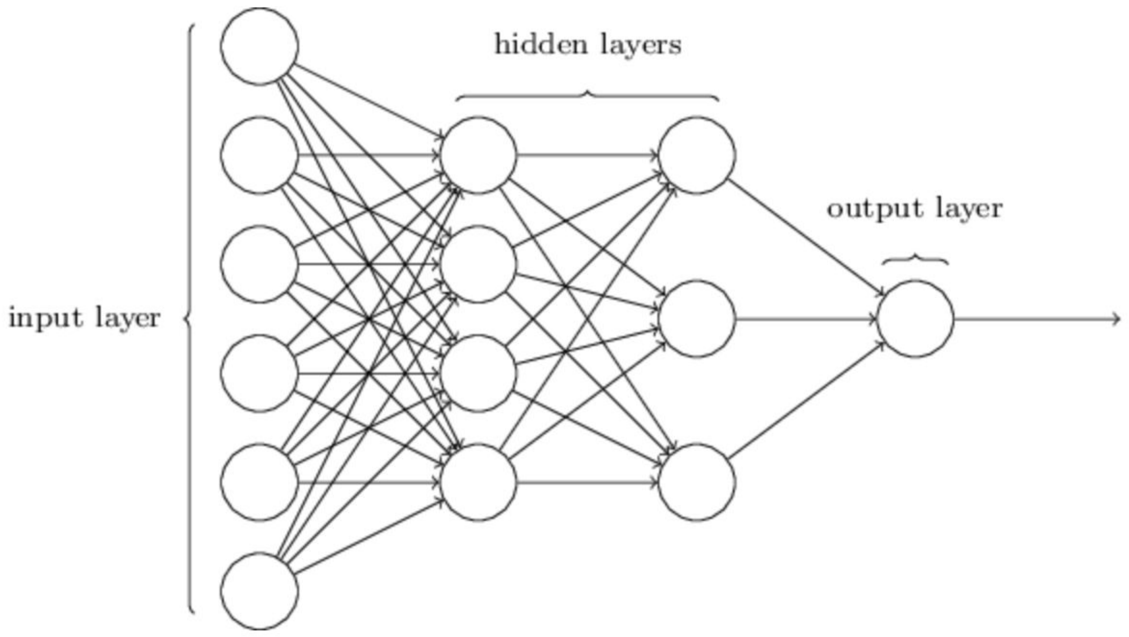


Figure 1: Demonstration of various layers in a CNN model.

Metrics

For CNN model, performance can be evaluated using the accuracy of the results. Accuracy can be obtained by analyzing the results obtained with the test data. The benchmark model used the same metric to understand the performance of the model. Accuracy is a simple metric to obtain given a dataset is well balanced as mentioned in the dataset summary.

Accuracy = (count (predicted output = expected output))/length (test data)) * 100

The project is primarily focused on the classifying human emotions. Therefore, the model used to classify the dataset can be evaluated based on whether the model can classify the emotions and how often it can do right. The metric that fits for this purpose is Accuracy, which can be calculated while training and testing the CNN model. There are many ways to evaluate CNN model. However, for this particular use case, Accuracy is the way to go as it fits the intended purpose of this project. It also helps in exploring the ways to optimize the model because we obviously need to find ways to achieve higher accuracy so it can be qualified to be used in day to day applications.

II. Analysis

Data Exploration

The dataset for this project is provided by Kaggle Challenge[2]. The dataset consists of 48X48 pixel grayscale images of faces. The dataset can be used in categorizing 7 basic human emotions.

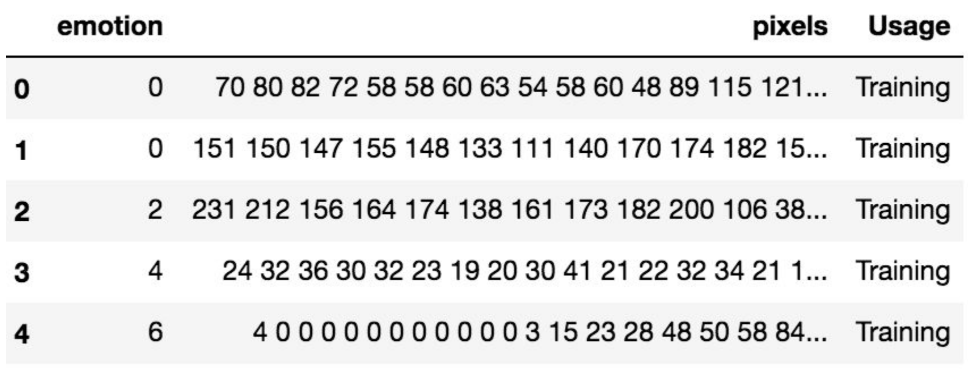
Info of the dataset mentioned on Kaggle Challenge:

The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). The training set consists of 28,709 examples. The public test set used for the leaderboard consists of 3,589 examples.

Figure 2: Images in the dataset with various emotions.

Exploratory Visualization

The sample data is shown in the figure below. It has 3 columns- emotion, pixels, and usage.



	emotion	pixels	Usage
0	0	70 80 82 72 58 58 60 63 54 58 60 48 89 115 121...	Training
1	0	151 150 147 155 148 133 111 140 170 174 182 15...	Training
2	2	231 212 156 164 174 138 161 173 182 200 106 38...	Training
3	4	24 32 36 30 32 23 19 20 30 41 21 22 32 34 21 1...	Training
4	6	4 0 0 0 0 0 0 0 0 0 0 0 0 3 15 23 28 48 50 58 84...	Training

Figure 3: Sample Dataset

Emotions and pixels are the columns used to train and test the data. As mentioned, emotion has seven categories and the distribution of the dataset based on these emotions is presented here.

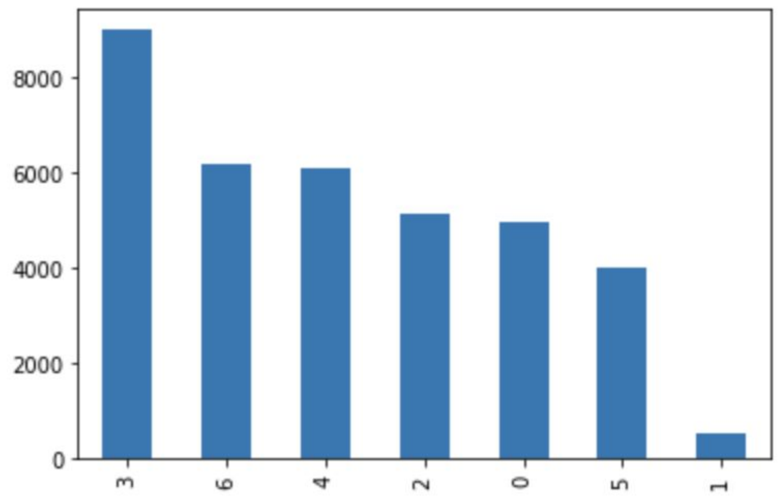


Figure 4: Distribution of the data points with respect to emotions.

The visualization depicts that the dataset is not balanced enough, particularly for the emotion 1-disgust. The can impact the accuracy of the model and hence need to be removed or merged with the similar data points, which is discussed in the data processing section.

Algorithms and Techniques

Convolution Neural Network (CNN) is the go to algorithm for object detection or any image processing techniques. As shown in the figure 5, CNN model is primarily a combination of convolution layer, pooling layer, and activation layer. As images goes through each layer, patterns are detected and mapped. Convolution layer checks for patterns in the section of image, whereas pooling layers reduces the number of params, and activation layers map the output value into a range of probabilities, thus predicting the output. Optimization of the CNN model can be done at each layer with various parameters based on the requirements and behavior of the model for that particular dataset. Training CNN model is an iterative process- as input is passed to the model, patterns are learned at each layer, and is continuously evaluated and adjusted with the back propagation by transmitting the loss function back to each layer so the learning weights are adjusted accordingly. Thus training the model with a large number of input dataset helps the model to learn the patterns effectively, thus improving the accuracy.

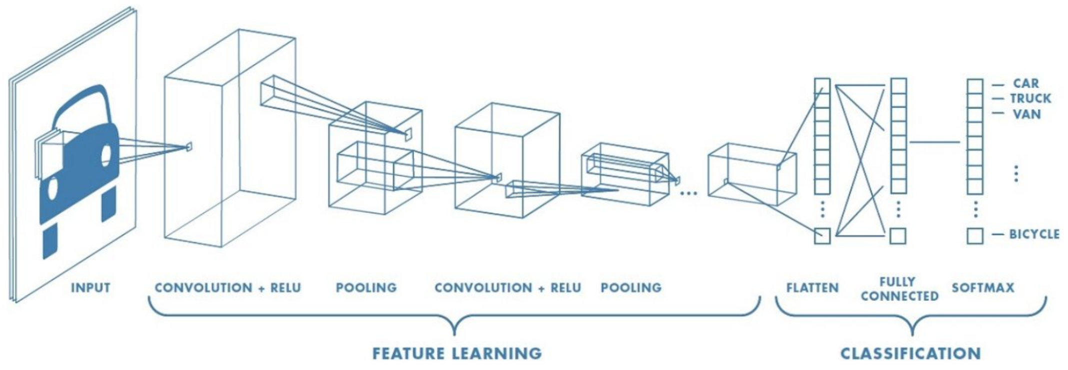


Figure 5: Typical architecture of a CNN model that presents various layers and optimizers

As detecting the patterns in image is complex process, CNN needs large number of datasets and parameters to obtain a good accuracy. If the CNN architecture is created and tuned well, it helps achieve the human level performance in many use cases. The dataset used in this project is big enough to train the model and achieve a decent level of accuracy.
Along with the dataset, tuning of the model is necessary to obtain an improved accuracy. The following parameters can be tuned for the optimization of CNN model.

Neural Network Architecture:

- ➔ Number of Layers
- ➔ Layer Parameters (such as optimization, regularization)
- ➔ Layer Types (Convolution, Fully Connected, Pooling)

Parameters to Optimize:

- ➔ Number of Epochs
- ➔ Batch Size
- ➔ Batch Normalization
- ➔ Regularization (L1&L2)
- ➔ Strides & Padding

Further, generating more data using Image Data Generator library can assist in providing more training data, which results in the improved accuracy.

In this project, good emphasis is placed on tuning these parameters to achieve the benchmark target accuracy. All of the params, mentioned in this section are tuned. Initially, the CNN model is trained with the training dataset and is used to tune the model. Later, testing data set is used to test the accuracy of the model. In fact, training and test will be taken care by providing the respect params in model evaluation.

Benchmark

The winner of the Kaggle competition has an accuracy level around 70% and this will be used as a benchmark for this project. The goal for this capstone project is to gain an accuracy level beyond 70 for the test dataset given in the challenge as well as the dataset obtained through the real-time video streaming from the DeepLens.

The results obtained by integrating the model with the DeepLens will be stored and plotted to gain a good insight. As DeepLens enables the possibility of testing in real-time, building a model to get an accuracy beyond 70 could be challenging.

III. Methodology

Data Preprocessing

As presented in the data exploration section, this dataset has two fields. The emotion field is the one that needs to be processed as the distribution of the dataset is not balanced enough. (Figure 4).

Since disgust has no enough data points, it can be merged with the Anger emotion as they both have significant correlation in expression. This will also provide a good balance to the dataset.

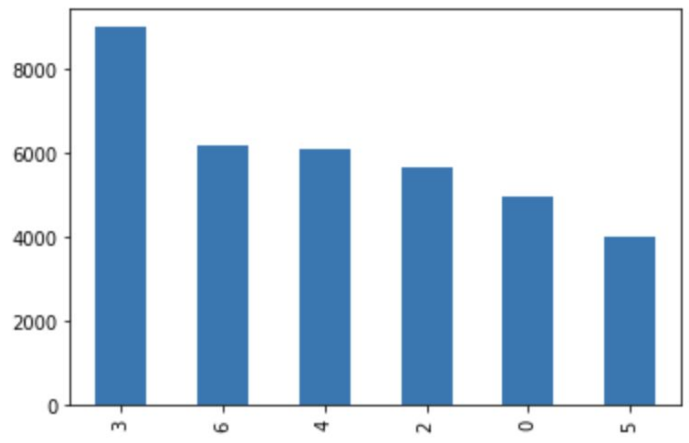


Figure 6: Processed Dataset with No Disgust Emotion.

Implementation

Implementation of CNN model typically follows a similar approach to any ML model. The process followed in this project is as follows.

- ➔ Clean the data to obtain a balanced dataset.
- ➔ Load and process the data using helper functions
  - load_fer2013 to load the data
  - Preporcess_input to process the data
- ➔ Split the training and testing data in 80-20 ratio.
- ➔ Use Image data generator to increase the training dataset.
- ➔ Define the model architecture and training parameters.
- ➔ Define activation function and optimizer.
- ➔ Obtain the model summary, train the model and log the accuracy.
- ➔ If either accuracy is not enough or the model has very high parms, tune the optimizers or re-architect the model

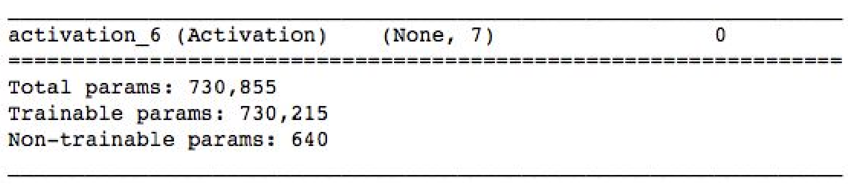
The primary challenge in approach to building this CNN model is identifying the right architecture and evaluating it. At first the traditional architecture was implemented with 32-64-128. Even though, the accuracy seemed to be fair compared to the benchmark, for the accuracy that is getting, the model seems to be unrealistic for the number of params that the model needs to train on. This also indicates the implementing the model in day to day applications may not work well. Therefore, a significant research has to be done to identify the model that fits the requirements and can be implemented or integrated with other applications easily. Other than that, optimizing the model has been very straightforward as this model makes use of standard optimizers.

In the project, the model is implemented in three phases by following the similar approach. Each time, the accuracy is observed using the logs and the model is improvised further. The three phases of the model are further discussed below.

- ➔ Phase-1: 32-64-128 with 10 Epochs
- ➔ Phase-2: 32-64-128 with 30 Epochs
- ➔ Phase-3: Mini Exception classification based on the research paper. .

Phase-1 & Phase-2:

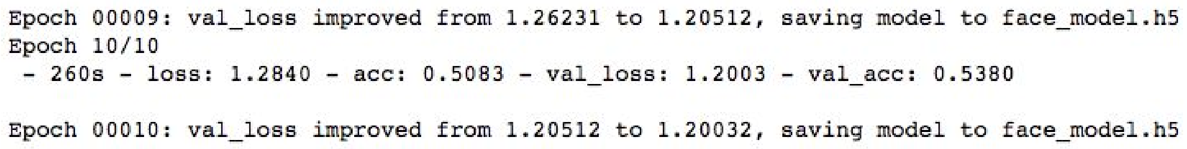
The traditional 32-64-128 CNN architecture is implemented with batch normalization, relu activation, max pooling, and dropout. The model has total params around 730k.



```
_________________________________________________________________
activation_6 (Activation)    (None, 7)                 0
=================================================================
Total params: 730,855
Trainable params: 730,215
Non-trainable params: 640
_________________________________________________________________
```

Figure 7: Number of Params For the traditional CNN for Phase-1 & Phase-2

For 10 epochs, the validation accuracy is around 53%, which is fair but not close enough to the bench mark.

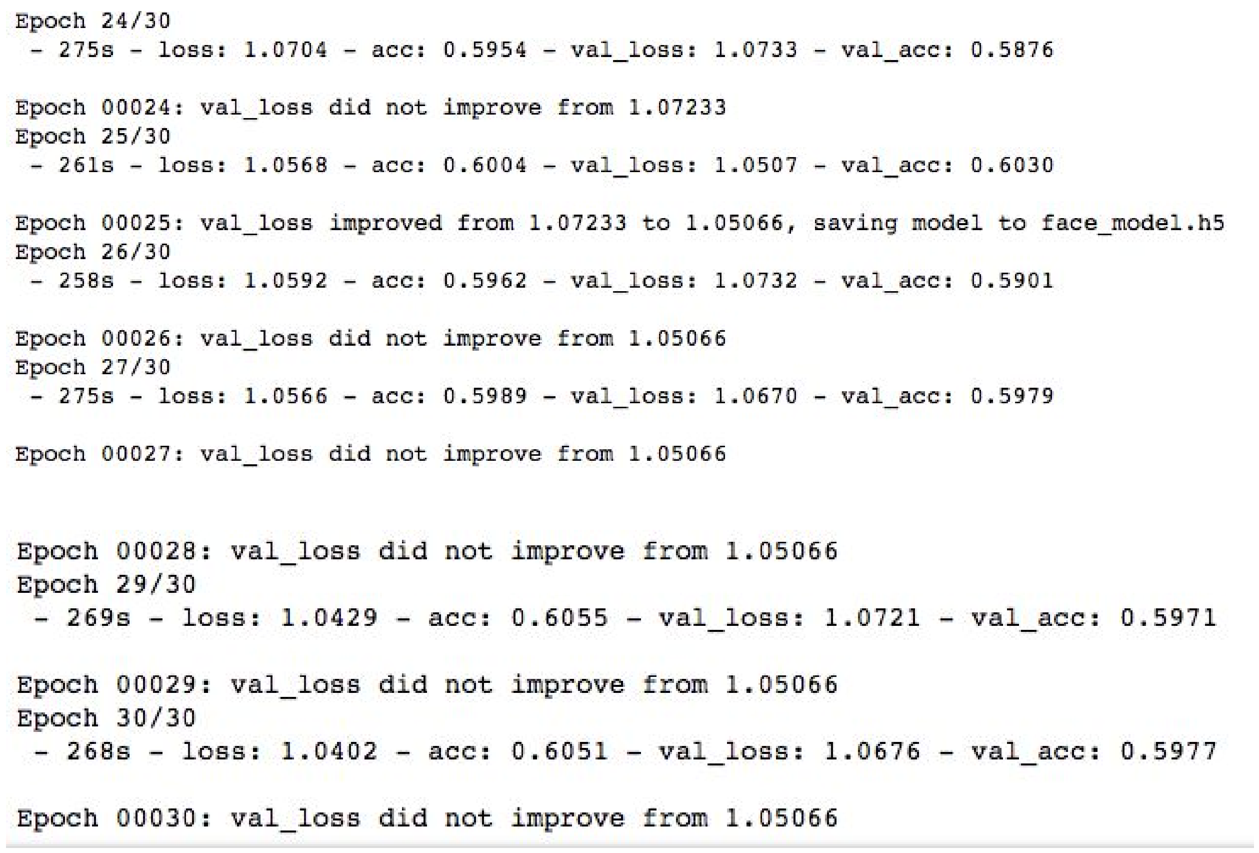


```
Epoch 00009: val_loss improved from 1.26231 to 1.20512, saving model to face_model.h5
Epoch 10/10
 - 260s - loss: 1.2840 - acc: 0.5083 - val_loss: 1.2003 - val_acc: 0.5380

Epoch 00010: val_loss improved from 1.20512 to 1.20032, saving model to face_model.h5
```

Figure 8: Accuracy of the model for Phase-1

For 30 epochs, the validation accuracy reached around 60%, which is relatively better than the previous one. But the accuracy seems to be saturated after 25th epoch and increasing the epochs doesn't improve accuracy any more.



```
Epoch 24/30
 - 275s - loss: 1.0704 - acc: 0.5954 - val_loss: 1.0733 - val_acc: 0.5876

Epoch 00024: val_loss did not improve from 1.07233
Epoch 25/30
 - 261s - loss: 1.0568 - acc: 0.6004 - val_loss: 1.0507 - val_acc: 0.6030

Epoch 00025: val_loss improved from 1.07233 to 1.05066, saving model to face_model.h5
Epoch 26/30
 - 258s - loss: 1.0592 - acc: 0.5962 - val_loss: 1.0732 - val_acc: 0.5901

Epoch 00026: val_loss did not improve from 1.05066
Epoch 27/30
 - 275s - loss: 1.0566 - acc: 0.5989 - val_loss: 1.0670 - val_acc: 0.5979

Epoch 00027: val_loss did not improve from 1.05066
```

```
Epoch 00028: val_loss did not improve from 1.05066
Epoch 29/30
 - 269s - loss: 1.0429 - acc: 0.6055 - val_loss: 1.0721 - val_acc: 0.5971

Epoch 00029: val_loss did not improve from 1.05066
Epoch 30/30
 - 268s - loss: 1.0402 - acc: 0.6051 - val_loss: 1.0676 - val_acc: 0.5977

Epoch 00030: val_loss did not improve from 1.05066
```

Figure 9: Accuracy of the model for Phase-2

Further, this architecture is not generalizable enough and it takes a lot of time to train all the params. If we add more layers, the number of params to train goes up and will take a lot of time to train and will need additional computational resources.

In order to improve the accuracy and make this more generalizable, a new architecture mini Xception is adapted from the research paper[1]. And this is further discussed in Refinement section.

Refinement

Phase-3:

The architecture is inspired by Xception architecture which eliminates the need of fully connected layers thus reducing the number of params and the computation time/cost without sacrificing the accuracy to a great extent. This architecture is a fully-convolutional neural network that contains 4 residual depth-wise separable convolutions where each convolution is followed by a batch normalization operation and a ReLU activation function. The last layer applies a global average pooling and a soft-max activation function to produce a prediction.

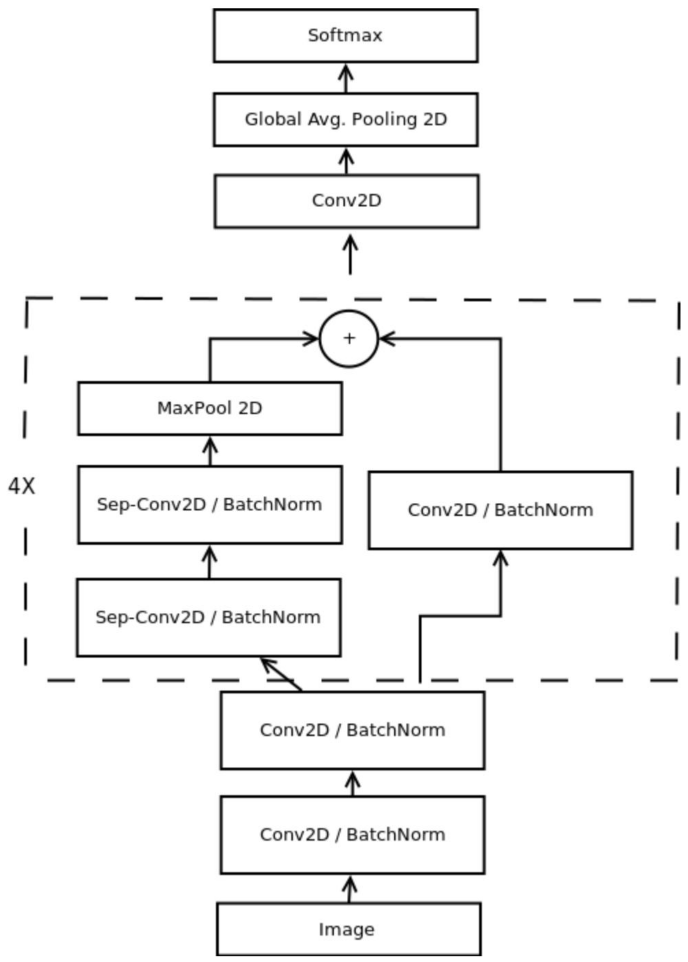
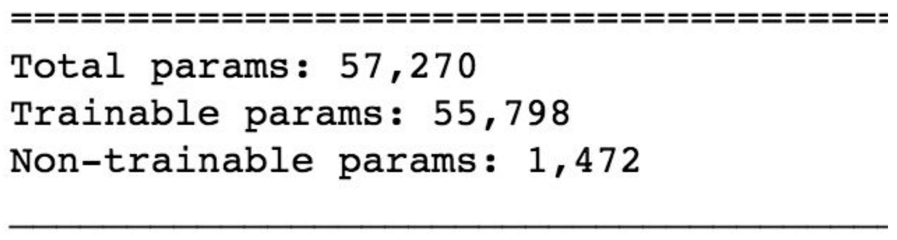


Figure 10: Mini Xception model adapted from the research paper(1).

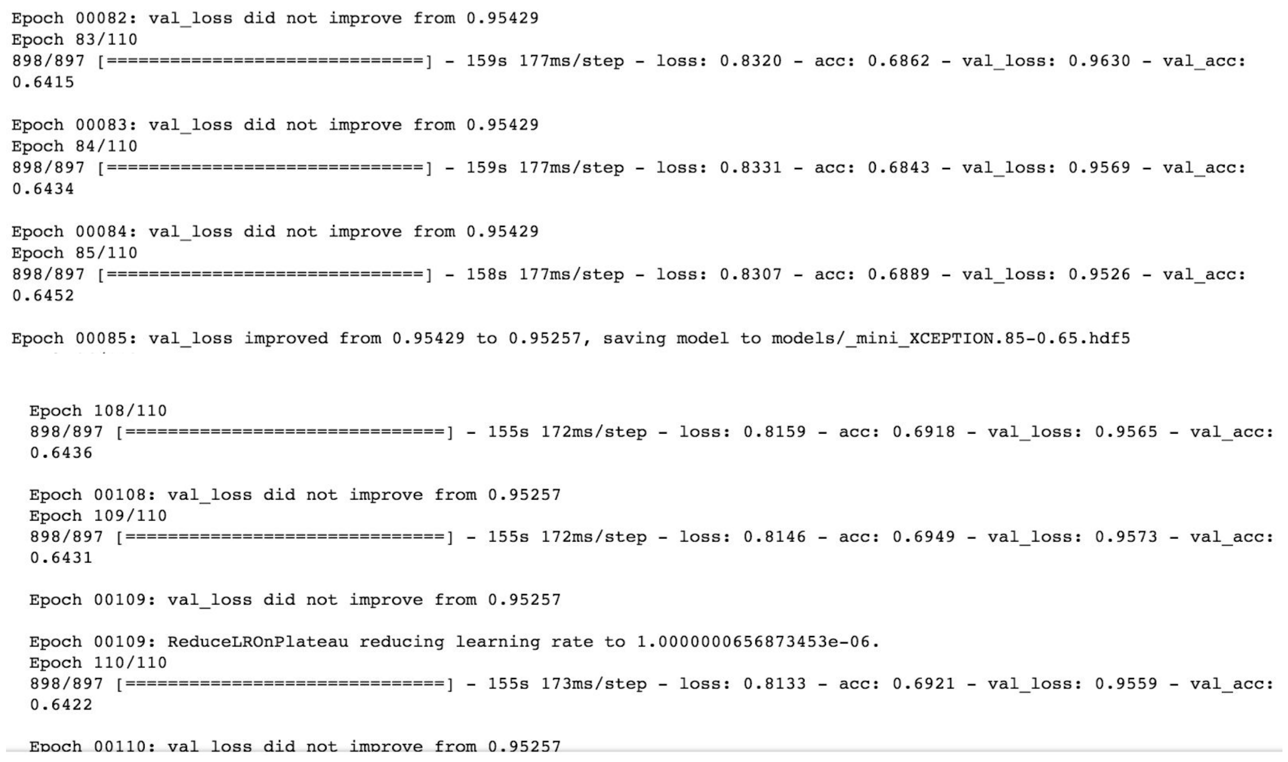
These convolutions are composed of two different layers: depth-wise convolutions and point-wise convolutions. Depth-wise separable convolutions reduces the computation with respect to the standard convolutions by reducing the number of parameters. The number of params for this model are 57k, which is 10% of the previous model.



```
==========================================
Total params: 57,270
Trainable params: 55,798
Non-trainable params: 1,472
__________________________________________
```

Figure 11: Number of Params to be trained for miniXception.

The accuracy obtained for this project is close to the accuracy reported in the research paper, which is around 65%. The model is optimized with all the params mentioned in the Algorithm and Techniques section. Further, increasing the number of epochs doesn't improve the accuracy as results seem to be saturated after 85th epoch.



```
Epoch 00082: val_loss did not improve from 0.95429
Epoch 83/110
898/897 [==============================] - 159s 177ms/step - loss: 0.8320 - acc: 0.6862 - val_loss: 0.9630 - val_acc:
0.6415

Epoch 00083: val_loss did not improve from 0.95429
Epoch 84/110
898/897 [==============================] - 159s 177ms/step - loss: 0.8331 - acc: 0.6843 - val_loss: 0.9569 - val_acc:
0.6434

Epoch 00084: val_loss did not improve from 0.95429
Epoch 85/110
898/897 [==============================] - 158s 177ms/step - loss: 0.8307 - acc: 0.6889 - val_loss: 0.9526 - val_acc:
0.6452

Epoch 00085: val_loss improved from 0.95429 to 0.95257, saving model to models/_mini_XCEPTION.85-0.65.hdf5
```

```
  Epoch 108/110
  898/897 [==============================] - 155s 172ms/step - loss: 0.8159 - acc: 0.6918 - val_loss: 0.9565 - val_acc:
  0.6436

  Epoch 00108: val_loss did not improve from 0.95257
  Epoch 109/110
  898/897 [==============================] - 155s 172ms/step - loss: 0.8146 - acc: 0.6949 - val_loss: 0.9573 - val_acc:
  0.6431

  Epoch 00109: val_loss did not improve from 0.95257

  Epoch 00109: ReduceLROnPlateau reducing learning rate to 1.0000000656873453e-06.
  Epoch 110/110
  898/897 [==============================] - 155s 173ms/step - loss: 0.8133 - acc: 0.6921 - val_loss: 0.9559 - val_acc:
  0.6422

  Epoch 00110: val_loss did not improve from 0.95257
```

Figure 12: Accuracy of miniXception implemented in Phase-3.

IV. Results

Model Evaluation and Validation

The final model is used in this project is MiniXception which is presented in figure 10. This model clearly demonstrates the superiority over the traditional CNN model in terms of architecture with the optimal computational and accuracy balance. Further, this model makes use of all the optimization params that helps to improve the accuracy. As reported in the paper, this same model obtained the accuracy of 96% for IMDB gender classification dataset as mentioned in the research paper[1]. This model is generalizable enough to work on various datasets. Above all, the accuracy obtained from this model is comparatively close to the benchmark set for this project.

Robustness:
Often, in CNN models, there is a possibility that model is overlearned, which implies that it works well only for the particular data points that the model is trained on. In order to test the robustness of the model, it is tested on the random images obtained from the google search and is presented in figure 13 and discussed in further detail in the Free-Form Visualization section. This model performed as expected on the unsee datasets with a decent accuracy.

Justification

The maximum accuracy obtained using this model is 65% compared to the benchmark of 70%. Even with higher number of epochs the accuracy doesn't vary much and it seems to be saturated at this level. This model is superior in terms of architecture, optimization, training params, the accuracy, and generalizability. The results obtained from the final model are significant enough for the facial expression detection with the given dataset.

V. Conclusion

Free-Form Visualization

The model is tested further with the images obtained from the google. The model is able to classify the images with good accuracy because these images clearly demonstrates the expression. However, for the images that couldn't detect the expression, it is basically treating it as a neutral mode (more images are presented in the file). It is also possible that more variation in input images that the model is trained on could result in better accuracy to classify the more complex expressions that humans express in daily life.

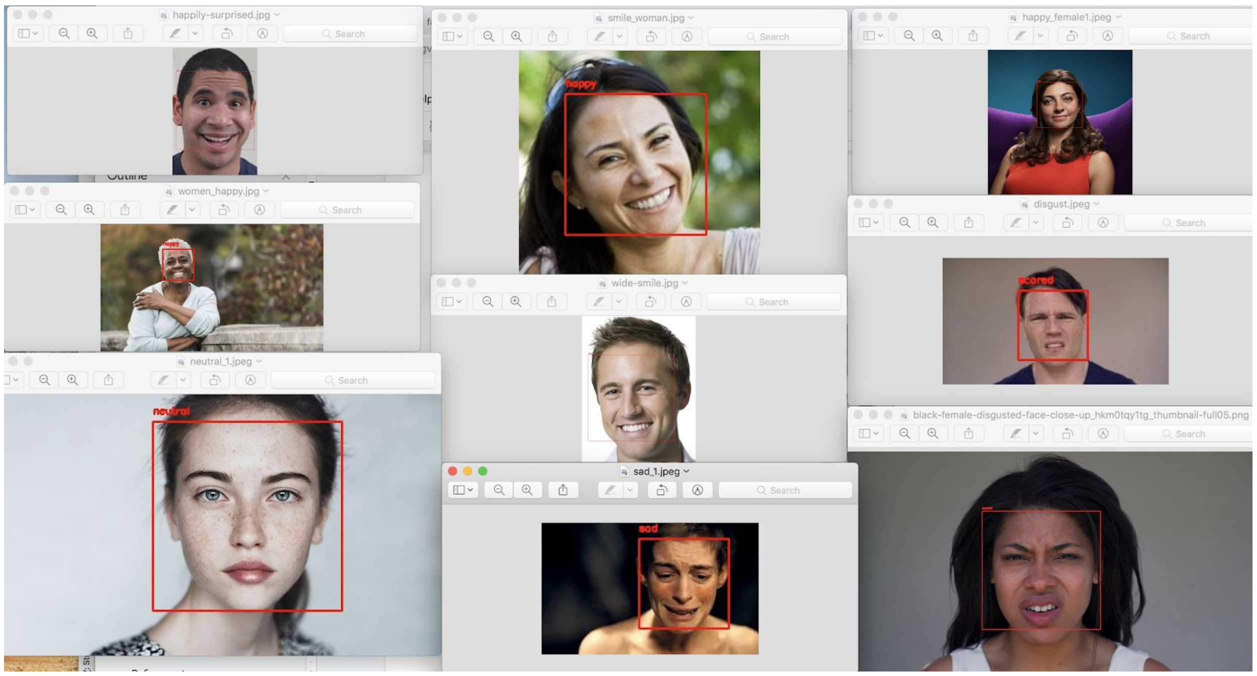


Figure 13: Visualization of the tested images obtained from Google.

Reflection

Very generic approach to build the ML model is followed in this project, where it starts with exploring and processing the dataset, building and fine tuning the model with an iterative process, and testing it.

The interesting as well as difficult part of the project is building and fine tuning the model. I had to spend significant time to research on the best architecture that fits the use case as the traditional sequential architecture always has limitations. The final architecture of the project, mini Xception, is very superior to traditional CNN and many other models as the parameters it trains and the computation time/cost required is a way lesser. This model is generic enough to be employed for any other datasets and is a good candidate to be used in any day-to-day applications as the demand for computation is low.

Improvement

The initial plan of the project is to deploy this in real-time using DeepLens and Alexa. However, because of the changes in my workplace, currently, I have no access to the required hardware. Deploying this model in real-time would have been a good learning experience.

As this model has a clean dataset, there is no room for improvement in terms of the data cleaning and processing. Similarly, the model is optimized enough to obtain the decent accuracy. However, the results of the model could be improved in multiple ways such as.

- ➔ Pre-training the model on similar datasets and running this model on top of those weights.
- ➔ Applying transfer learning using some architectures like VGG, ResNet, Xception.
- ➔ Providing more data and employing a complex architecture like ResNet would improve the accuracy. However, it will be very heavy on computation.
- ➔ Doing further research on the Mini Xception to add fully connected layer at the end but trying to minimize the params it needs to train on could result in a better model. However, I'm not sure yet on how to approach this yet.

References:

1.Arriaga, O., Valdenegro-Toro, M., & Plöger, P. (2017). Real-time convolutional neural networks for emotion and gender classification. *arXiv preprint arXiv:1710.07557*.

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