Detect human emotion through the images

**Adelene Ng, Wong Cheng Kwan, Perumal Srinivas**

# Introduction

The objective of this project is to develop a deep learning model to detect the human facial emotion through images. Given an image or video footage of people, we want to develop a deep learning model to predict the 7 different facial emotion expressions: angry, disgust, fear, happy, neutral, sad and surprise. We use the deep learning approach because it can automatically determine the important features unlike in classical machine learning, facial features such as the eye, nose and mouth have to be first defined before the system can identify which features are more important in detecting a particular emotion.

# Task Distribution

Given this is a three-man project, this is how the different tasks were distributed amongst the team members (Table 1):

|  |  |  |
| --- | --- | --- |
| No | Tasks | Team Members |
| 1 | Understanding of Business Case | All |
| 2 | Exploratory Data Analysis | All |
| 3 | Data pre-processing and preparation | All |
| 4 | * Modeling VGGNet16 (Wong Cheng Kwan) * MobileNetV1 (Perumal Srinivas) * DenseNet121 (Adelene Ng) | Individual |
| 5 | Evaluation | Individual |
| 6 | Deployment | All |
| 7 | Project Presentation | All |

Table 1 – Work Breakdown and Task Allocation

# Datasets

For this project, we will use the Facial Expression Recognition - 2013 dataset from [Kaggle](https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data) [1].

The data consists of **48x48** grayscale facial images. The task is to categorize each face based on the emotion shown in the facial expression into one of seven categories (see Table 2)

| Emotion Identifier | Emotion |
| --- | --- |
| 0 | Angry |
| 1 | Disgust |
| 2 | Fear |
| 3 | Happy |
| 4 | Sad |
| 5 | Surprise |
| 6 | Neutral |

Table 2 – Facial Emotion Categories

The dataset is stored in the ‘fer2013.csv’ file. It contains 3 columns, "emotion", "pixels" and ‘Usage’. The type associated with each of the columns are listed as follows:

|  |  |
| --- | --- |
| Column Name | Type |
| Emotion | Int64 |
| Pixels | object |
| Usage | object |

Table 3 – Type associated with Column

The "emotion" column contains a numeric code ranging from 0 to 6, inclusive, for the sentiment that is present in the image. The "pixels" column contains a string surrounded in quotes for each image. The contents of this string a space-separated pixel values in row major order. The ‘Usage’ column categorizes the data row being either training or test.

The training set consists of **28,709 labelled images**, and **7178 images** in the **test set**.

# Exploratory Data Analysis

We first plot some sample faces from the dataset. This is shown in Figure 1.



Figure 1 – Plot Sample Faces

The frequency distribution of the different facial emotions is shown in Table 4**.**

| Frequency Distribution of Dataset | Facial Emotion Count Sorted |
| --- | --- |
|  |  |

Table 4 – Frequency Distribution of Facial Emotions in Dataset

From Table 4, we observe that ‘Happiness’ has the highest count (7215) and ‘Disgust’ has the lowest count (436).

# Machine Learning Design and Development Approaches

We plan to use the following deep learning architectures for this project:

* VGGNet16 (Wong Cheng Kwan)
* MobileNetV1 (Perumal Srinivas)
* DenseNet121 (Adelene Ng)

For each of the different architectures, we first build the baseline model. Next, we train the model and examine some of the resulting evaluation metrics. We then run different experiments on each of the different architectures, fine tuning the models in the process. The best performing model will then be used for final deployment.

We will not be using transfer learning in this project as this does not allow us to fine tune the earlier layers.

We will be using the following parameters to fine tune our model performance:

* Vary the batch size
* Learning rate
* Dropout layer
* Use different optimizers

Since too many epochs can lead to overfitting of the training dataset, and too few may result in an underfit model, we apply early stopping as a form of regularization to avoid the overfitting problem.

# Model Tuning & Experimental Logs

## DenseNet121 (Adelene Ng)

DenseNets [2] are densely connected convolutional neural networks. They are used widely for image classification tasks, segmentation, image reconstruction tasks, for example.

In a typical CNN, layer 3 receives input from layer 2 and layer 2 receives input from layer 1. In a dense block, which is a building block for DenseNets, layer 3 receives inputs from all previous layers. This is illustrated in Figure 2.



Figure 2 - Regular CNN (Left) and the Dense Block (Right)

The baseline DenseNet121 architecture is shown in Figure 3. The ‘visualkeras’ package is used to produce the visualization for this baseline architecture.



Figure 3 – DenseNet121 Architecture

The DenseNet architecture was chosen because it offered several advantages and these are:

1. reduces the vanishing-gradient problem
2. strengthens feature propagation
3. encourages feature reuse
4. substantially reduces the number of parameters

Feature propagation and reuse are reinforced because for each layer, the feature-maps of all preceding layers are used as inputs, and its own feature-maps of the current layer are used as inputs into all subsequent layers.

The downside to DenseNet is that it is memory hungry because of the need for multiple concatenation operations because data is copied multiple times.

.

### Experimental Run 1 (Baseline)

The first step was to create a baseline model. In run 1, the following parameters were used:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Batch size | # Epochs | Learning rate | Optimizer | Dropout | ES - Patience | Loss | Accuracy |
| 128 | 108/200 | 0.0001 | Adam | 0.4 | 20 | 1.4980 | 59.1% |

Table 5 – Hyperparameters for Run 1

A batch size of 128 was adopted. The number of epochs was set to 200. Early stopping (ES) was used, with a patience set to 20. It is usually set to a value between 10 and 100. This is the number of epochs to wait after which training will be stopped if no improvement is observed. Adam with a learning rate of 0.0001 was used as the optimizer. A dropout of 0.4 was used.

The default DenseNet121 parameter settings used in the baseline model is shown in the table below:

| Parameters | Value |
| --- | --- |
| num\_blocks | 3 |
| num\_layers\_per\_block | 4 |
| growth\_rate | 16 |
| dropout\_rate | 0.4 |
| compress\_factor | 0.5 |
| eps | 1.1e-5 |
| num\_filters | 16 |

Table 6 – DenseNet121 Baseline Model Default Parameters

The DenseNet121 baseline model had 92 layers with a total of 2,377,958 parameters.

This baseline run produced a loss of **1.498** and an accuracy of **59.1%** was achieved. The loss and accuracy graphs are shown below:

| Training Accuracy | Training Loss |
| --- | --- |
|  |  |

Table 7 – Model Training Accuracy and Loss

The confusion matrix for this run is shown in Figure 4.

| Confusion Matrix | Comments |
| --- | --- |
|  | We can see that happiness and surprise has many correct classifications, anger, disgust and neutrality have some correct classifications whereas fear and sadness had very few correct classifications. These results imply that our model is very good at detecting smiles and surprised expressions, but is unable to pick up on more varied emotions such as sadness or neutrality, which can look different on different people. |

Figure 4 – Confusion Matrix for Run 1

### Experimental Run 2 (ReduceLROnPlateau)

The objective of “ReduceLROnPlateau” is to reduce the learning rate when a metric has stopped improving. If no improvement is seen for a “patience” number of epochs, the learning rate is reduced by factor value (new\_lr = lr \* factor), where factor = 0.2. The resulting accuracy is less than the baseline model. This may be due to the training stopping earlier compared to the baseline model (91 vs. 108).

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Batch size | # Epochs | Learning rate | Optimizer | Dropout | ES - Patience | Loss | Accuracy |
| 128 | 91/200 | 0.0001 | Adam | 0.4 | 20 | 1.5204 | 57.4% |

Table 8 - Hyperparameters for Run 2

The loss and accuracy graphs are shown Table 9.

|  |  |
| --- | --- |
| Training Accuracy | Training Loss |
|  |  |

Table 9 - Model Training Accuracy and Loss

The confusion matrix for this run is shown Figure 5.

| Confusion Matrix | Comments |
| --- | --- |
|  | We see again that happiness, disgust and surprise have many correct classifications, anger, sadness and neutrality have some correct classifications whereas fear had very few correct classifications. These results imply that our model is very good at detecting smiles disgust and surprised expressions, but is unable to pick up on more varied emotions such as sadness or neutrality, which can look different on different people. |

Figure 5 - Confusion Matrix for Run 2

### Experimental Run 3 (Vary Dropout Rates)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Batch size | # Epochs | Learning rate | Optimizer | Dropout | ES - Patience | Loss | Accuracy |
| 128 | 59/200 | 0.0001 | Adam | 0.8 | 20 | 2.3336 | 25.4% |
| 128 | 42/200 | 0.0001 | Adam | 0.2 | 20 | 1.1601 | 61.6% |
| 128 | 40/200 | 0.0001 | Adam | 0.1 | 20 | 1.3121 | 60.1% |

Table 10 - Hyperparameters for Run 6

| Dropout | Training Accuracy | Training Loss |
| --- | --- | --- |
| 0.8 |  |  |
| 0.2 |  |  |
| 0.1 |  |  |

Table 11 - Model Training Accuracy and Loss for different dropout values (0.8, 0.2 and 0.1)

| Dropout | Confusion Matrix | Comments |
| --- | --- | --- |
| 0.8 |  | In this case, we see that everything gets classified as a ‘happy’ face. The ‘happy’ emotion has the highest number in the dataset used for training, so when trained, the model would easily identify ‘happy’ faces. The poor results from the other categories could be explained when using high dropout, many correlated and useful features from previous layers are discarded resulting in drastic drop in accuracy. |
| 0.2 |  | Again, ‘happiness’ has many correct classifications. This category has the highest counts in the dataset, so when the model is trained with many such examples, it would be able to correctly classify this emotion. ‘Surprise’ also has many correct classifications. The model finds it hard to discern between ‘sad’ and ‘neutral’ expressions, also between ‘disgust’ and ‘fear’, also between ‘anger’ and ‘disgust’ which can look dissimilar on different people. |
| 0.1 |  | Here, we see again that ‘happiness’ has many correct classifications. It has the highest counts in the dataset, and the resulting model can correctly classify this emotion. Disgust, sadness and surprise show fewer correct classifications. Anger, fear and neutrality show the least numer of correct classifications. |

Table 12 – Confusion Matrix for Dropout: 0.8, 0.2 and 0.1

The results show that a high dropout rate resulted in low accuracy. In a DenseNet architecture, information from all the previous layers is being passed down the network. Setting a high dropout rate to 0.8 could mean that many of the useful features and correlations previously learned was discarded and the resulting loss in information resulted in a drastic drop in accuracy. It is observed that reducing the dropout increases the model accuracy, which is most likely due to less information from previous layers being discarded.

### Applying Grid Search

Next, grid search was applied to the following experimental runs to determine:

* Best optimizer to use
* Best Learning Rate
* Best Batch Size

Grid search made it easier to keep track of hyperparameters tested. However, having said that, since I used the default value of “None” for CV (cross validation), this defaults to using the 5-fold cross validation.

Although grid search could have been applied using a search space as follows:

params = {

‘optimizers’: [‘SGD’, ‘Adam’],

‘learning\_rate’: [0.001, 0.0001, 0.00001],

‘batch\_size’: [32, 64, 128]

}

which results in 2x3x3=18 different parameter combinations. Since a default cross-validation of 5-fold was used, this would mean the grid search code would train the model 18 x 5 = 90 times. I chose not to do it this way because of the long computational times.

#### Experimental Run 4 (Vary Learning Rates)

In this run, we apply grid search to determine the best learning rate. 3 different values of learning rates were tried and these are: **0.001, 0.0001, 0.00001**.

| Batch size | Learning rate | Optimizer | Dropout | ES - Patience | Best score | Best Parameters |
| --- | --- | --- | --- | --- | --- | --- |
| 128 | 0.001, 0.0001, 0.00001 | Adam | 0.4 | 20 | 59.7% | 0.001 |

Table 13 – Grid Search to determine best learning rate

#### Experimental Run 5 (Vary Optimizers)

In this run, we apply grid search to determine the best optimizer. 2 different types of optimizers were tried. These were **Adam** and **SGD**.

| Batch size | Learning rate | Optimizer | Dropout | ES - Patience | Best score | Best Parameters |
| --- | --- | --- | --- | --- | --- | --- |
| 128 | 0.0001 | SGD, Adam | 0.4 | 20 | 58.5% | Adam |

Table 14 – Grid Search to determine best optimizer

#### Experimental Run 6 (Vary Batch Sizes)

In this run, we apply grid search to determine the best batch size. 3 different batch sizes were tried. These were **32**, **64** and **128**. A batch size of 256 was not used because my system ran out of memory.

| Batch size | Learning rate | Optimizer | Dropout | ES - Patience | Best score | Best Parameters |
| --- | --- | --- | --- | --- | --- | --- |
| 32, 64 and 128 | 0.0001 | Adam | 0.4 | 20 | 56.5% | 64 |

Table 15 – Grid Search to determine best batch size

### Experimental Run 7 (Use best parameters from runs 1 - 6)

In this run, we use the best hyperparameters from previous experimental runs:

| Batch size | # Epochs | Learning rate | Optimizer | Dropout | ES - Patience | Loss | Accuracy |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 64 | 43/200 | 0.001 | Adam | 0.2 | 20 | 1.5301 | 60.8% |

Table 16 - Hyperparameters for Run 7

The loss and accuracy graphs are shown in Table 17.

|  |  |
| --- | --- |
| Training Accuracy | Training Loss |
|  |  |

Table 17 - Model Training Accuracy and Loss

The confusion matrix for this run is shown in Figure 6.

|  |  |
| --- | --- |
| Confusion Matrix | Comments |
|  | Again, we see that happiness and surprise has many correct classifications, anger, disgust and fear have some correct classifications and sadness, and neutrality had very few correct classifications. These results imply that our model is very good at detecting smiles and surprised expressions, but is unable to pick up on more varied emotions such as sadness or neutrality, which can look different on different people. When run against the test images, the model was unable to predict fear, neutral and sadness correctly. |

Figure 6 – Confusion Matrix for Run 7

### Experimental Run 8 (Use Best parameters with Data Augmentation)

In this run, we use the best hyperparameters resulting from runs 1 – 6 with data augmentation to see if we can squeeze even better performance from our model.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Batch size | # Epochs | Learning rate | Optimizer | Dropout | ES - Patience | Loss | Accuracy |
| 64 | 81/200 | 0.001 | Adam | 0.2 | 20 | 1.0264 | 67.2% |

Table 18 – Hyperparameters for Run 8

Table 19 shows the data augmentation parameters that were applied to the training dataset for this run:

| Parameter | Description | Value |
| --- | --- | --- |
| rescale | Rescales the pixel values of the image to be between [0, 1] | 1./255 |
| rotation\_range | Randomly rotates the image clockwise by 150 | 15 |
| width\_shift\_range | Horizontal shift of the image. Shift image width by 15% | 0.15 |
| height\_shift\_range | Vertical shift of the image. Shift image height by 15% | 0.15 |
| shear\_range | Shear the image by 15% | 0.15 |
| zoom\_range | Randomly zooms in or out of the image in the range of [1 - 0.15, 1 + 0.15] | 0.15 |
| horizontal\_flip | Flips along the horizontal axis. | True |
| fill\_mode | Points outside the boundaries of the input are filled according to the given mode, in this case it will fill the area with the nearest pixel. | 'nearest' |

Table 19 – Data Augmentation Parameters used in Run 8

The loss and accuracy graphs are shown in Table 20.

|  |  |
| --- | --- |
| Training Accuracy | Training Loss |
|  |  |

Table 20 – Model Training Accuracy and Loss

The confusion matrix for this run is shown in Figure 7.

|  |  |
| --- | --- |
| Confusion Matrix | Comments |
|  | After applying data augmentation, we see that happiness, disgust and surprise have many correct classifications, sadness, and neutrality have some correct classifications, and anger and fear have few correct classifications. These results imply that our model is very good at detecting smiles, disgust and surprise, but is unable to pick up on more varied emotions such as anger or fear, which can look different on different people. In the dataset, we observe there were very few disgust images. This explains the poor performance before augmentation. Data augmentation seems to have produced a more ‘even’ spread of the different classifications unlike in previous runs. |

Figure 7 – Confusion Matrix for Run 8

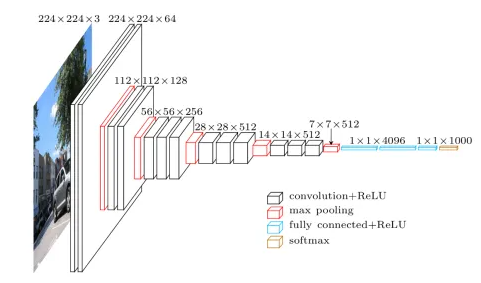
Next, I wanted to determine the number of misclassified images from the validation set. Table 21 shows the number of misclassified images. Happy and Surprise show the least misclassifications, the former emotion most likely due to having the highest number of training samples in the dataset. Sadness and Neutral emotions have similar number of misclassifications and their distribution is comparable. These emotions can look different on different persons and are subjective. Anger and fear show similar misclassifications too and they have similar distributions too. Disgust surprisingly showed quite a low number of misclassifications, in spite having the fewest number of training images.

| Emotion | Misclassified | Percentage Misclassified |
| --- | --- | --- |
| Angry | Total 491 miss labels out of 958 for emotion Angry | 51.3% |
| Disgust | Total 31 miss labels out of 111 for emotion Disgust | 27.9% |
| Fear | Total 552 miss labels out of 1024 for emotion Fear | 53.9% |
| Happy | Total 202 miss labels out of 1774 for emotion Happy | 11.4% |
| Sad | Total 438 miss labels out of 1247 for emotion Sad | 35.1% |
| Surprise | Total 176 miss labels out of 831 for emotion Surprise | 21.2% |
| Neural | Total 462 miss labels out of 1233 for emotion Neutral | 37.5% |

*Table 21 – Misclassified Images*

## VGGNet16 (Wong Cheng Kwan)

VGGNet is a classical convolutional neural network architecture used in large-scale image processing and pattern recognition. The VGG network introduced the concept of grouping multiple convolution layers with smaller kernel sizes instead of having one Conv layer with a large kernel size. This caused the number of features at the output to reduce and including multiple Relu layers instead of one, thus increasing learning instances.



### Experimental Run with VGG16 vs VGG19

VGG19 will have another three Convolutional layers with learnable weights, which also meant with deeper depth than VGG16.

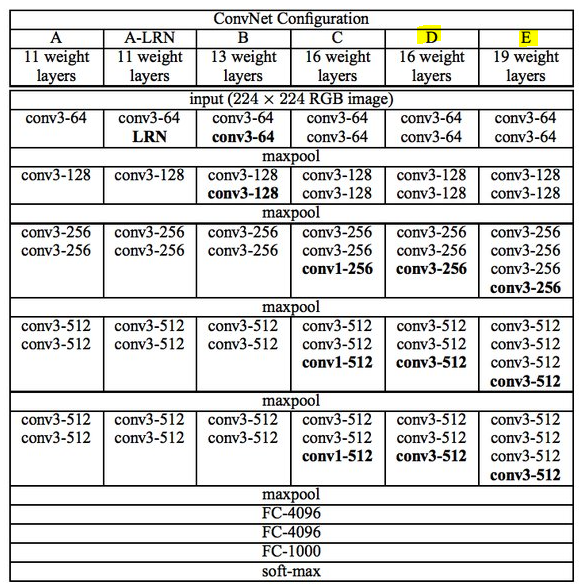


Table 22 – VGG 16 and 19 Comparison (Wong Cheng Kwan)

Despite of this deep layers, where VGG19 is having 15.8% more learnable weights for computation, it only yields very slight accuracy improvement of 0.71% over VGG16. That is why VGG16 is chosen as the basic model.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Model | Batch Size | Dropout | Epochs with Early Stopping | Learning Rate | Optimizers | Validation accuracy |
| 1 | VGG16(base) | 64 | - | 25 / 200 | 1e-4 | Adam | 58.65% |
| 2 | VGG19 | 64 | - | 26 / 200 | 1e-4 | Adam | 59.07% |

Table 23 – VGG 16 and 19 Experiments (Wong Cheng Kwan)

### Experimental Run for Baseline – with different dropout rate

A table created as below for recording the validation accuracy, with adjustment made to the parameters one at the time.

Although VGG16-1 is having the highest accuracy, VGG16-1a is a better fit, as it is showing the following qualities.

* The plot of training loss decreases to a point of stability.
* The plot of validation loss decreases to a point of stability and has a small gap with the training loss.

Thus VGG16-1a will be used as the baseline for the subsequence experiments for improvement.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | Model | Batch Size | Dropout | Epochs with Early Stopping | Learning Rate | Optimizers | Validation accuracy |
| 1 | VGG16(base) | 64 | - | 25 / 200 | 1e-4 | Adam | 58.65% |
| 2 | VGG16-1 | 64 | 0.5 | 37 / 200 | 1e-4 | Adam | 62.11% |
| 3 | VGG16-1a | 64 | 0.8 | 116 / 200 | 1e-4 | Adam | 60.69% |
| 4 | VGG16-1b | 64 | 0.2 | 27 / 200 | 1e-4 | Adam | 61.86% |

Table 24 – VGGNet16 Baseline Model Parameters (Wong Cheng Kwan)

|  |  |  |  |
| --- | --- | --- | --- |
| VGG16(Base) | VGG16-1 | VGG16-1a | VGG16-1b |
|  |  |  |  |

Table 25 – Accuracy and Loss Graphs Comparison (Wong Cheng Kwan)

### Experimental Run with the rest of parameters

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Run | Model | Batch Size | Epochs with Early Stopping | Patience by ‘val\_loss’ | Learning Rate | Optimizers | Val\_accuracy |
| 5 | VGG19 | 64 | 26 / 200 | 10 | 1e-4 | Adam | 59.07% |
| 6 | VGG16-2 (fm vgg16-1a) | 128 | 86/200 | 10 | 1e-4 | Adam | 58.00% |
| 7 | VGG16-3 (fm vgg16-2) | 128 | 14 / 200 | 10 | 1e-3 | Adam | 24.94% |
| 8 | VGG16-4 (fm vgg16-3) | 128 | 45 / 200 | 10 | 1e-5 | Adam | 25.0% |
| 10 | VGG16-6 (fm vgg16-2) | 128 | 200 / 200 | 10 | 1e-4 | SGD | 24.94% |
| 11 | VGG16-1C (from vgg16-1a) | 128 | 115/200 | 20 | 1e-4 | Adam | 57.54% |
| 12 | VGG16-2a (augmentation) (fm vgg16-2) | 128 | 133 / 200 | 10 | 1e-4 | Adam | 52.00% |

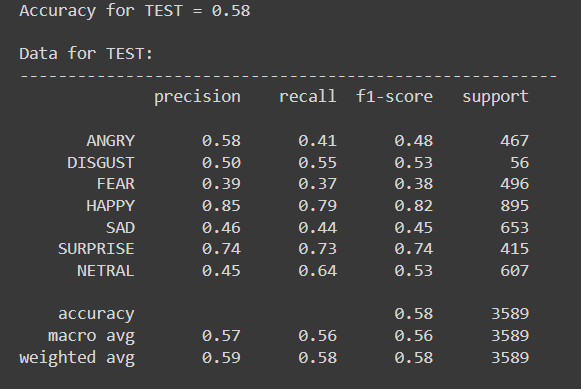
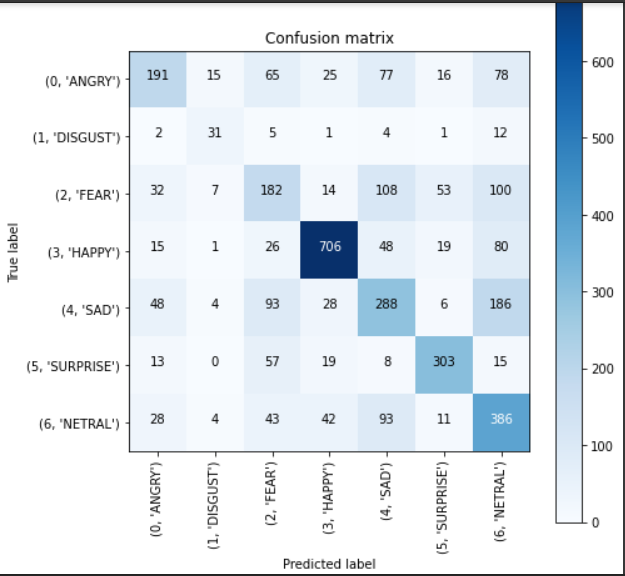
Table 26 – Experimental Run (Wong Cheng Kwan)

Run 1 is a base model of VGG16 without any dropout rate.

Run 2 to 4 is using different dropout rate to get the best possible learning curves graphs.

Run 5, where model is VGG19, is to experiment whether it yield better result over VGG16. From the result, it showed the depth of the model is not the main contributor to the overall accuracy.

Run 6 is derive from VGG16-1a, but with batch size of 128. Batch size is not creating a big impact on the model accuracy, as they are more related to memory space needed. ‘Happy’ had the hight f1-score while ‘Fear’ is having the lowest f1-score of 38%, which meant the model is not predicting ‘Fear’ well.



Run 7 and 8 were for learning rate experimenting. The higher learning rate took almost 24 hr just to run 14 epochs.

Run 10 tried to use SGD as optimizer, but the result is bad.

Run 11 increase the patience to 20 in early stopping, hoping the increase of epochs can give raise to a better result.

Run 12 refer to next section.

### Experimental Run the best accuracy with augmentation

From the dataset, “Disgust” is having 436 records, while “Happy” is 7215. Data categories were thus not well distributed. Data augmentations were introduced to narrowing down such gap. Data were augmented based on random flip in horizontal and vertical, random rotation by 20%, random contrast by 20%, 80%, and random zoom 20%.

## MobileNetV1 (Perumal Srinivas)

The MobileNet architecture was chosen as it offered the following advantages and are as follows:

1. MobileNet (Efficient Convolutional Neural Networks for Mobile Vision Applications) is an architecture that focuses on making the deep learning networks very small and having low latency.
2. The MobileNet models can be easily deployed on mobile and embedded edge devices.
3. MobileNets are based on a streamlined architecture that uses depth-wise separable convolutions to build light weight deep neural networks.
4. The MobileNet uses a Depthwise separable convolution instead of the usual convolution to reduce the computation time and parameters.

It works by applying convolution to each channel of the image instead of as a block n times, and then using 1x1 convolution to get n filters.

Depthwise separable convolution is a depthwise convolution followed by a pointwise convolution as follows:

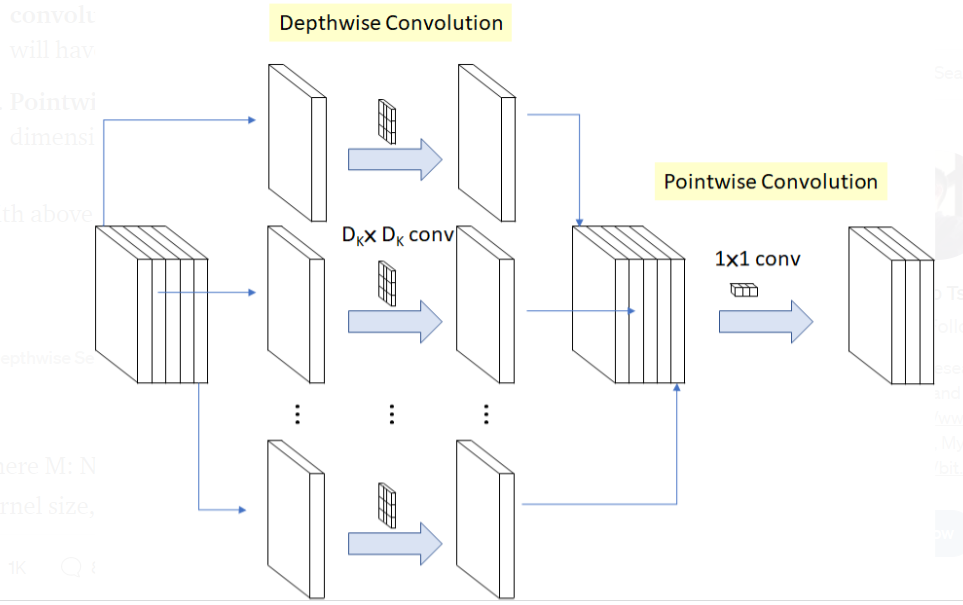


Table 27 – Depthwise convolution followed by pointwise convolution

In the above, we have 5 channels, then we will have 5 DKXDK spatial convolution, Pointwise convolution actually is the 1X1 convolution to change the dimension.With above there is less computation about 8 to 9 times.

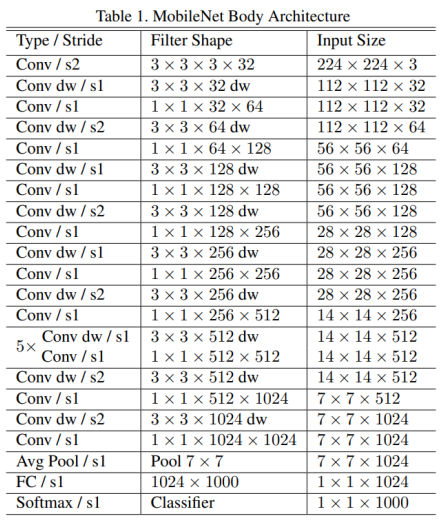


Table 28 - MobileNetV1 Architecture

Batch Normalization (BN) and ReLU are applied after each convolution:

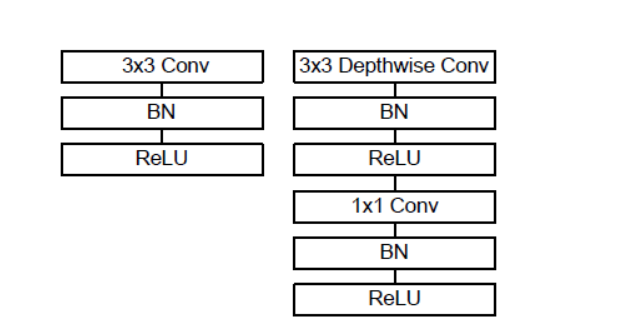


Table 29 - Batch Normalization (BN) and ReLU

### Experiment Depthwise Separable Convolution vs Standard Convolution for ImageNet

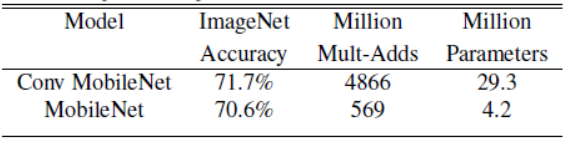


Table 30 - Depthwise Separable Convolution vs Standard Convolution

MobileNet only got 1% loss in accuracy, but the Mult-Adds and parameters are reduced tremendously.

### Experimental Run for Baseline – with a scale down on the original parameters of MobileNetV1

The first step was to create a baseline model. In run 1, the following parameters were used:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Batch size | # Epochs | Learning rate | Optimizer | Parameters | ES - Patience | Loss | Accuracy |
| 32 | 83/200 | 0.001 | Nadam | 44,263 | 20 | 1.68 | 50.10% |

Table 31 – Hyperparameters for Run 1

A batch size of 32 was adopted. The number of epochs was set to 200. NAdam with a learning rate of 0.001 was used as the optimizer.

The MobileNetV1 baseline model had a total of 44,263 parameters.

This baseline run produced a loss of **1.68** and an accuracy of 50.10% was achieved. The loss and accuracy graphs are shown below and is a very high over fitting and need to tune the hyper parameters and data augmentation.

| Training Accuracy | Training Loss |
| --- | --- |
|  |  |

Table 32 – Model Training Accuracy and Loss

The confusion matrix as shown below

| Confusion Matrix | Comments |
| --- | --- |
|  | We can see that happiness and surprise has many correct classifications, anger, sad, disgust and neutrality have less. The model needs to be improved and further trained. |

Table 33 - Confusion Matrix

When increased the parameters the model results are not very encouraging and when further reduced and hyper parameters tuned, and with data augmentation the accuracy has increased.

### Experimental Run -Varying Optimizer, Batch size & Layers with ReduceLROnPlateau and Data Augmentation

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Run | Batch size | # Epochs | Learning rate | Parameters | Optimizer | ES – Patience | Loss | Accuracy |
| 2 | 128 | 194/200 | 0.001 | 76,327 | Nadam | 20 | 1.41 | 46.92% |
| 3 | 64 | 197/200 | 0.0001 | 76,327 | Adam | 20 | 1.66 | 34.02% |
| 4 | 128 | 105/200 | 0.007 | 44, 263 | Nadam | 20 | 1.24 | 53.39% |
| 5 | 128 | 195/200 | 0.001 | 205063 | Nadam | 20 | 1.57 | 38.95% |

Table 34 - Varying Hyper Parameters

| Parameter | Model 3 | All other Models Value (2, 4, 5) |
| --- | --- | --- |
| Rotation\_range | 3 | 15 |
| Width\_shift\_range | 0.1 | 0.15 |
| Height\_shift\_range | 0.3 | 0.15 |
| Shear\_range | 0.1 | 0.15 |
| Zoom\_range | 1.0 | 1.0 |
| Horizontal\_flip | True | True |
| Fill\_mode | ‘nearest’ | 'nearest' |

Table 35 - Data Augmentation Parameters

| Run | Training Accuracy | Training Loss |
| --- | --- | --- |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |

Table 36 - Training Accuracy and Loss

| Run | Confusion Matrix | Comments |
| --- | --- | --- |
| 2 |  | In this case, ‘happy’ face classification is high, followed by ‘surprise’. The ‘Disgust’ face is very less The ‘sad’, ‘neutral’, angry’ are relatively less results and may need a better modelling to have it improved. |
| 3 |  | In this case, we see that ‘happy face’ and ‘surprise face’ emotion has the highest number in the dataset used for training. The disgust is not properly recognised as need further training. The categories like ‘neutral’, ‘sad’, ‘fear’ and ‘angry’ has a lower correct classification. |
| 4 |  | The following model has a better performance compared to the previous. The Disgust,Sad and fear is very low. |
| 5 |  | Only ‘happy’ and ‘Surprise’ have a higher number , Disgust has very low and other emotions have the least. Though this model has the higher parameters but still does not improve predictability for this dataset. |

Table 37- Confusion Matrix of all Run

# Best Model

Here is a summary showing the best results from the 3 different models the team tried:

| Model | Accuracy |
| --- | --- |
| VGGNet16 | 58% |
| MobileNetV1 | 53% |
| DenseNet121 | 67.2% |

Table 38 – Accuracy Results from 3 Models

From Table 38, we observe that the best model is DenseNet. We will use this model in our deployment.

# Deployment

Once we have determined what the best model is, this saved model can then be deployed to do inference.

Both still images as well real time emotion detection on a video stream are supported in our deployment.

This is installed as a desktop application. Images are sent to the application which then uses our trained model to do inference on the images. Results are displayed with bounding boxes drawn over the detected faces showing the emotion that was identified.

Our desktop application can also do real time detection of emotions on a video stream.

# Future Directions

Having built our models, some of the other approaches we can try are:

* Applying model averaging ensemble techniques for facial emotion classification
* Use a pre-trained model as a feature extractor. This can be achieved by loading the model and then simply adding new layers, through adding new convolutional and pooling layers to expand upon the feature extraction capabilities of the model or adding new fully connected classifier type layers to learn how to interpret the extracted features on a new dataset, or some combination of the two.

# Reference

1. [Face expression recognition dataset | Kaggle](https://www.kaggle.com/jonathanoheix/face-expression-recognition-dataset)

1. [Densely Connected Convolutional Networks](https://arxiv.org/pdf/1608.06993v5.pdf) CVPR 2017 Gao Huang Zhuang Liu Laurens van der Maaten Kilian Q. Weinberger
2. Improving Facial Emotion Recognition with Image processing and Deep Learning Ksheeraj Sai Vepuri San Jose State University
3. Improving Facial Emotion Recognition Systems Using Gradient and Laplacian Images Ram Krishna Pandey, Member IEEE, Souvik Karmakar, A G Ramakrishnan, Senior Member IEEE and N Saha