IMAGE CLASSIFICATION WITH TENSORFLOW

PDAN8412 POE

NOVEMBER 23, 2022

SEAN TEUCHERT
ST10041890

Contents

1. Introduction:

The dataset that has been selected to build an image classification model consists of 35 685 PNG images of examples of human faces expressing seven different emotions (happiness, neutral, sadness, anger, surprise, disgust, fear). Each image dimension is 48x48 pixels, grey scale, and has been categorised into subfolders of each class. To satisfy the dataset size requirement of the POE, data augmentation has been applied to blur and rotate each image by 45 and 75 degrees. This will also simulate the model receiving and unclear image during production which will ultimately improve model performance when receiving new data. After overcoming the possible ethical issues that come with recording people, this type of model can be applied to improve many different industries. A possible use case for this model is in the retail industry or any industry that consists of queues. A camera can be setup to record and analyse the emotions of the customers waiting in queue. If the model identifies that the customers are starting to become unhappy, it can alert a manager that there may be a problem. This will allow managers to perform other tasks on busy days to improve efficiency instead of having to physically walk around to determine the mood of the queue. Another possible use case for this type of model is to analyse the emotion displayed by users when reacting to certain features or marketing on a business's website or application. In both use cases, the author recommends using an image classification model over human driven feedback like surveys or interviews. Human driven feedback often has bias when it is being collected for example, it is common for customers to rush the completion of surveys and provide feedback that doesn't truly reflect their opinions and during an Interview, the interviewer may have an influence on the interviewee's reaction. By implementing an image classification model to automate this process, it will allow the model to record customers natural reactions in real time. This will reduce the influence that external factors may have on the results and provide the business with an efficient tool to monitor customer satisfaction.

Author of the Dataset: Ananthu Balakrishnan (ananthu017)

Dataset Link: https://www.kaggle.com/datasets/ananthu017/emotion-detection-fer

Please note:

A Jupyter Notebook was used to develop the initial code, but I used Visual Studio Code to write the finalized code. The file types are the same therefore the code can be run in Jupyter Notebook or Visual Studio Code.

This project was built using both TensorFlow and Keras libraries. The POE specifies that TensorFlow must be used in this project. Kera's uses TensorFlow's backend, therefore they are very similar and consists of similar algorithms. The main reason for using Keras is due to the frequent updates it receives; therefore, the data analyst has chosen to use Keras to remain up to date with the latest technology.

2. What is Image Classification?

Image classification is the process of identifying what an image represents for example, a model can be trained to recognise photos that represent three different types of fruits: apples, bananas, and peaches (TensorFlow, 2022). Image classification is a supervised machine learning algorithm. A supervised machine learning algorithm aims to map the mapping function between the input data and the desired output (IBM Cloud Education, 2020). This means that the model is provided with labelled data to train algorithms to classify data or predict outcomes. The model is provided with a defined set of target classes (objects to identify in the images) and trained to recognise them using the labelled example photos (Google Developers, 2022).

Deep Learning Algorithms can be used for image classification. Deep Learning is a subset of Machine learning that is based on Artificial Neural Networks (ANN). An ANN consists of interconnected nodes which are inspired by the human brain, and three layers: input, hidden and output layer. This project mainly focuses on using Convolutional Neural Networks (CNN) to perform image classification. A CNN is a deep learning algorithm that can take an input image, assign importance to various aspects/objects in the image, and is able to differentiate one from the other (Saha, 2018). CNNs are ideal for image classification because it can handle high dimensionality images without losing its information.

3. Why is the Dataset appropriate for Image Classification with TensorFlow?

Image classification requires a labelled dataset and images. The selected dataset is appropriate for image classification because it consists of images and class labels. The dataset has been divided into seven subfolders and each subfolder represents an emotion (happiness, neutral, sadness, anger, surprise, disgust, fear). The images can be imported using python and the correct label can be extracted with the image, Refer to notebook for this step. Another reason this is a suitable dataset is because the images are already in greyscale. An CNN can consist of many different layers therefore, it is better to train the model using greyscale images because it reduces the number of dimensions in the model which reduces model complexity and training time. The dataset also consists of a neutral emotion which provides the model will appropriate data to differentiate between the different emotions. The dataset consists of imbalanced samples, but Data Augmentation has been applied to increase the size of low sample classes. The dataset consists of images of different people, this helps combat overfitting. It reduces the chances of the model learning unwanted features of one specific person for example, if every image was taken by a person with a birth mark on their face, the model will look for the birth mark when identify what emotion the person is displaying because that face feature has been mapped to that emotion as a common trait.

4. What is the question that the analysis will answer?

The purpose of this analysis is to answer the following questions:

- Is it possible to identify human emotion from an image using Deep Learning?
- Can a Deep Learning model be used to categorize the emotions of humans into seven different classes?

5. Types of analysis performed:

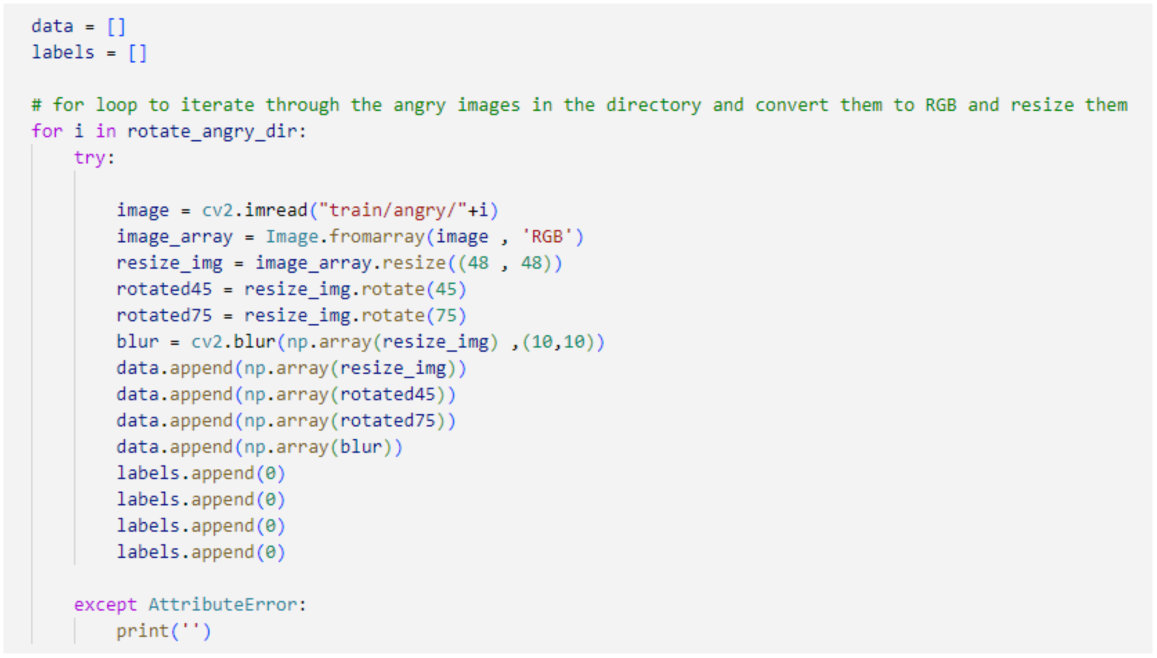
- Descriptive Analysis – Visualize the dataset
- Predictive Analysis – Convolutional Neural Network (CNN) and Transfer Learning has been used to build a model to predict human emotion.

6. Exploratory Data Analysis and Data Wrangling

To improve the data analyst's understanding of the dataset, Exploratory Data Analysis (EDA) has been performed. A figure plot was used to display a single image from each emotion, this helps the analyst determine what features it may link to each emotion. A second figure plot was used to determine if the images were correctly imported from the subfolders and if data augmentation was applied correctly. A count plot was used to determine the sample sizes of each class, this highlights the possibility of overfitting and underfitting.

After performing EDA, the next step that was applied is Feature Engineering or Data Wrangling. This step consists of converting the data to the most appropriate format before fitting the model to ensure the model performs optimally.

Figure 1 illustrates the dataset import method:

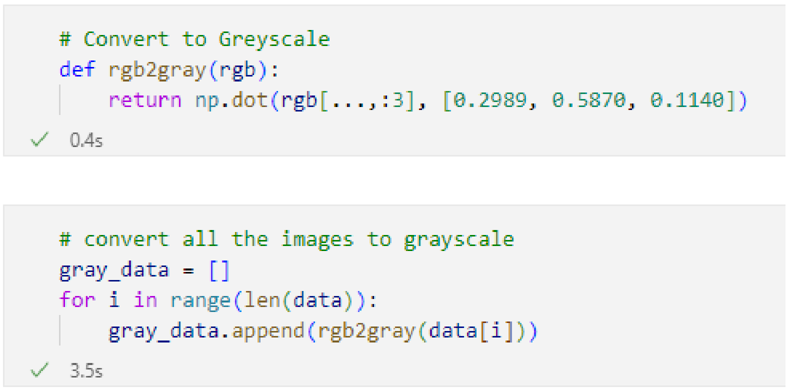


```
data = []
labels = []

# for loop to iterate through the angry images in the directory and convert them to RGB and resize them
for i in rotate_angry_dir:
    try:

        image = cv2.imread("train/angry/"+i)
        image_array = Image.fromarray(image , 'RGB')
        resize_img = image_array.resize((48 , 48))
        rotated45 = resize_img.rotate(45)
        rotated75 = resize_img.rotate(75)
        blur = cv2.blur(np.array(resize_img) ,(10,10))
        data.append(np.array(resize_img))
        data.append(np.array(rotated45))
        data.append(np.array(rotated75))
        data.append(np.array(blur))
        labels.append(0)
        labels.append(0)
        labels.append(0)
        labels.append(0)

    except AttributeError:
        print('')
```

Figure 1: Dataset Import method

This method has been used to import the images and create the dataset. It uses the directory path of the different subfolders to read the images, record each image information, resize each image, rotated each image by 45 and 75 degrees, blur each image and add the image data and image labels to the dataset. This method was repeated for the remaining classes.

Figure 2 illustrates the method to convert to greyscale:



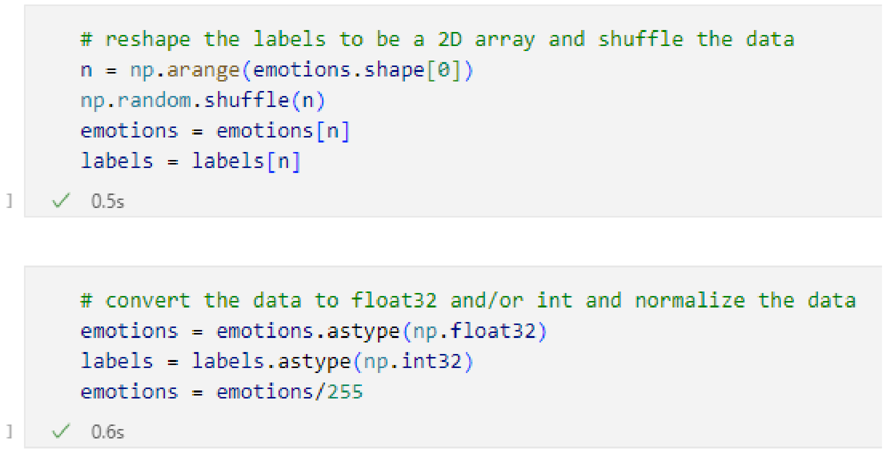
```
# Convert to Greyscale
def rgb2gray(rgb):
    return np.dot(rgb[...,:3], [0.2989, 0.5870, 0.1140])
✓ 0.4s
```

```
# convert all the images to grayscale
gray_data = []
for i in range(len(data)):
    gray_data.append(rgb2gray(data[i]))
✓ 3.5s
```

Figure 2: Greyscale method

The images were imported in RGB format, but the images are captured in greyscale, therefore the images must be converted to greyscale to increase model accuracy. This method converts the RGB metrics into greyscale.

Figure 3 illustrates the methods used to normalize the dataset:



```
# reshape the labels to be a 2D array and shuffle the data
n = np.arange(emotions.shape[0])
np.random.shuffle(n)
emotions = emotions[n]
labels = labels[n]
✓ 0.5s
```

```
# convert the data to float32 and/or int and normalize the data
emotions = emotions.astype(np.float32)
labels = labels.astype(np.int32)
emotions = emotions/255
✓ 0.6s
```

Figure 3 Normalization methods

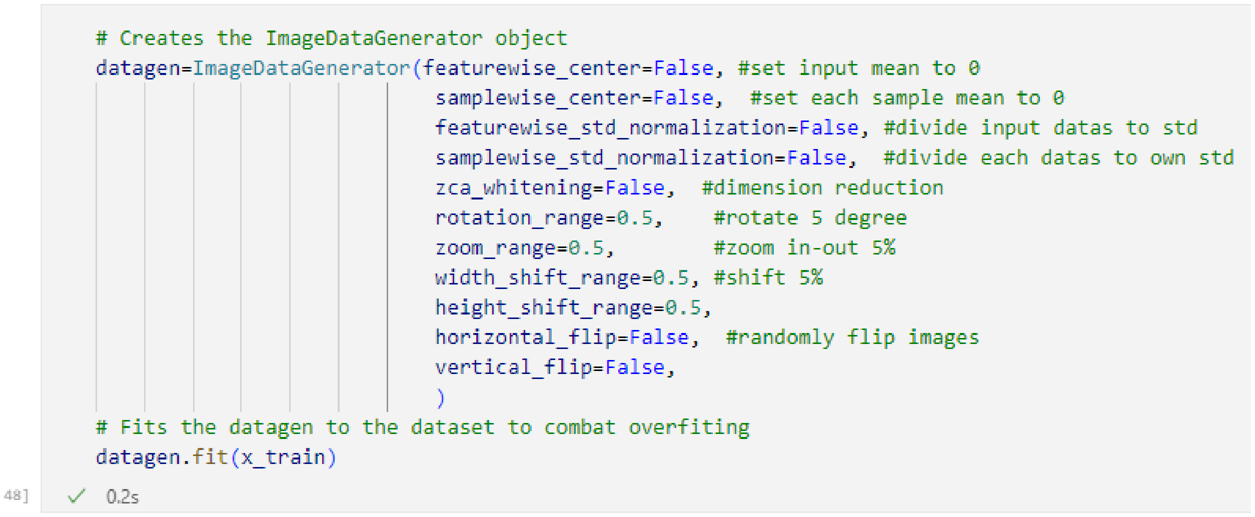
The first method is used to shuffle the dataset once it has been converted to a Numpy array. This will ensure that the model does not learn any unwanted underlying trend within the dataset order. The second method is used to convert the image data to a float32 and label data to Int32 (required model formats) and divide the image data by 255. This will normalize the dataset and ensure that the model is working with smaller numbers which will reduce training time.

7. Overfitting and Underfitting:

Overfitting refers to a model that models the training data too well or has been over trained with a large amount of data. This can become a problem because the model will not be able to correctly analyse a new dataset. Underfitting refers to a model that cannot capture the underlying trend of the data, this is a result of using the incorrect model to analyse the data or by not providing enough test data.

The first method the data analyst has applied to combat overfitting is called “Early Stopping” from the Keras Library. it is a method that allows the user to specify a large number of training epochs and stop training once the model performance stops improving on a holdout validation dataset (Brownlee, 2020). The second method the data analyst has applied to combat overfitting and underfitting is data augmentation, this increases the sample sizes within the dataset. As mentioned in Section 6 and illustrated by Figure 1, the data was augmented using the code in the screenshot. Another method to perform data augmentation is by using the built-in function made by TensorFlow called Image Data Generator.

Figure 4 illustrates the Image Data Generator method:



```
# Creates the ImageDataGenerator object
datagen=ImageDataGenerator(featurewise_center=False, #set input mean to 0
                            samplewise_center=False,  #set each sample mean to 0
                            featurewise_std_normalization=False, #divide input datas to std
                            samplewise_std_normalization=False,  #divide each datas to own std
                            zca_whitening=False,  #dimension reduction
                            rotation_range=0.5,    #rotate 5 degree
                            zoom_range=0.5,        #zoom in-out 5%
                            width_shift_range=0.5, #shift 5%
                            height_shift_range=0.5,
                            horizontal_flip=False,  #randomly flip images
                            vertical_flip=False,
                            )
# Fits the datagen to the dataset to combat overfiting
datagen.fit(x_train)
```

48] ✓ 0.2s

Figure 4: Image Data Generator Method

This method is used to perform Data Augmentation using TensorFlow. The main differences between the methods in Figure 1 and Figure 4 is that the Image Data Generator will augment the data when it is needed during each epoch. The data analyst identified two problems with this method, the dataset size is unknown when using the Image Data Augmentation method and it increased the training time. It is difficult to identify overfitting without knowing the dataset size for example, the rotate variable could be set too low therefore the method generates a large dataset of images that are very similar and becomes overfit. If the data analyst could track the number of images this error could have been spotted. Therefore, the data analyst decided to rather use the method in Figure 1.

8. Table and Graph Explanation:

Figure 5 illustrates the number of images per emotion:

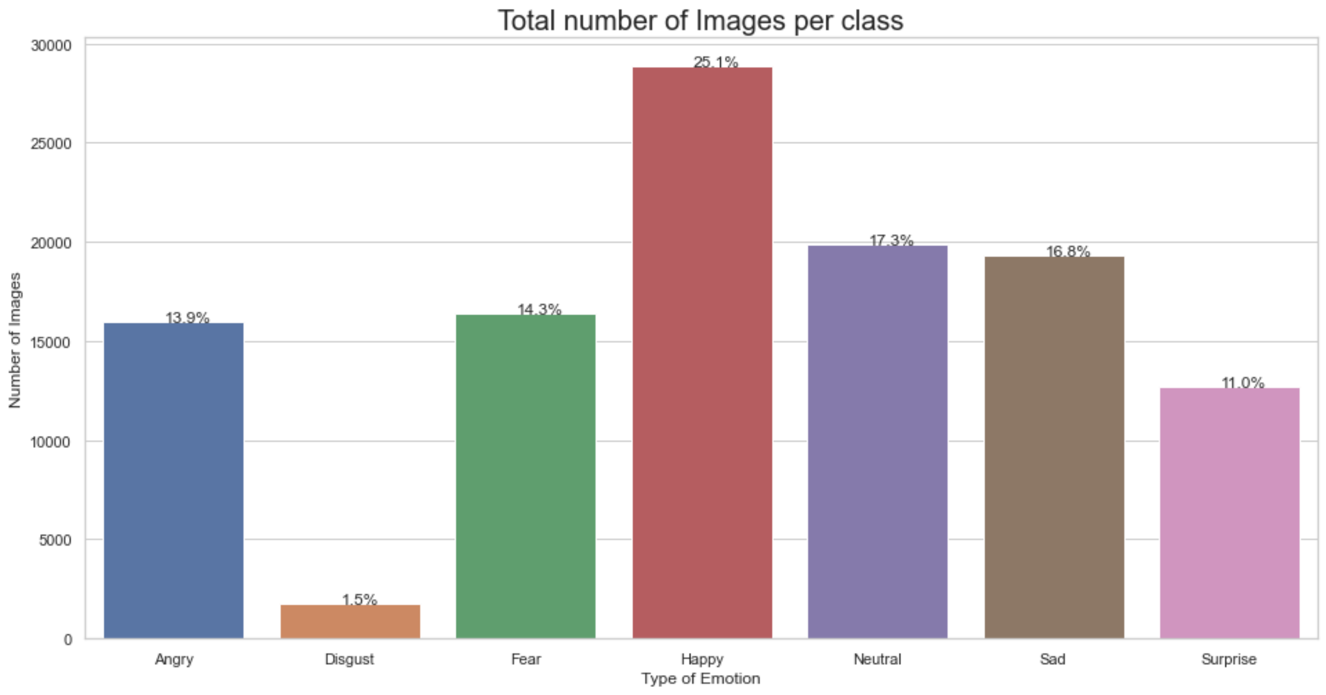


Figure 5: Number of Images per Class bar chart

This bar chart conveys the information about the spread of the dataset. The x-axis indicates the type of emotion, and the y-axis indicates the number of images per emotion. From this we can interpret what classes have the lowest and highest sample sizes. This information is important because it may affect the performance of the model at predicting the low sample classes. This graph also helps the data analyst decide whether to implement data augmentation.

Figure 6 illustrates the training and validation accuracy of the model per epoch:

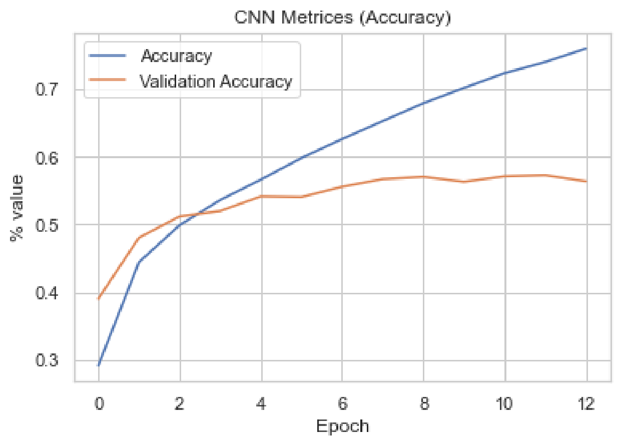


Figure 6: Accuracy vs Epoch line graph

This graph conveys information about how the training accuracy and validation accuracy of the model increases per epoch. The x-axis indicates the epoch number, and the y-axis indicates the accuracy percentage. This information is important to know because it helps the data analyst identify when the model is starting to overfit, this is indicated by when the line starts levelling out. The data analyst has applied the Early_stopping method to combat overfitting, this graph illustrates that the method worked correctly as the training stopped once the validation accuracy started oscillating.

Figure 7 illustrates the training and validation loss of the model per epoch:

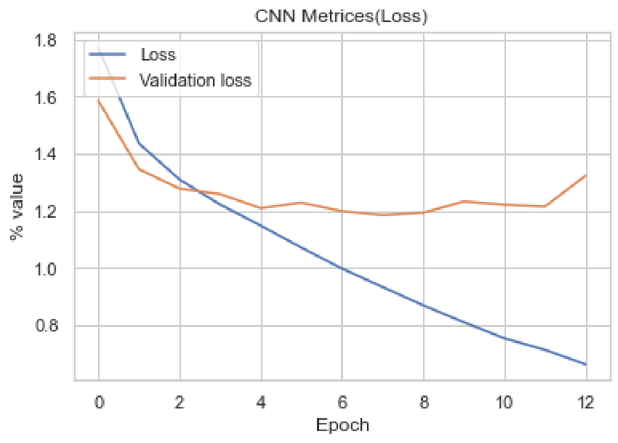


Figure 7: Loss vs Epoch line graph

This graph conveys information about how the training and validation loss of the model increases per epoch. The x-axis indicates the epoch number, and the y-axis indicates the loss percentage. This information is important to know because it helps the data analyst identify when the model is starting to overfit, this is indicated by when the line starts levelling out. The data analyst has applied the Early_stopping method to combat overfitting, this graph illustrates that the method worked correctly as the training stopped once the validation loss started oscillating.

Figure 8 illustrates CNN classification confusion matrix:

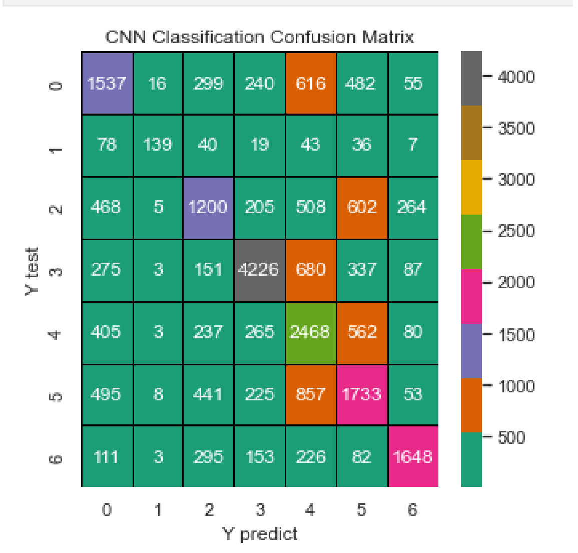


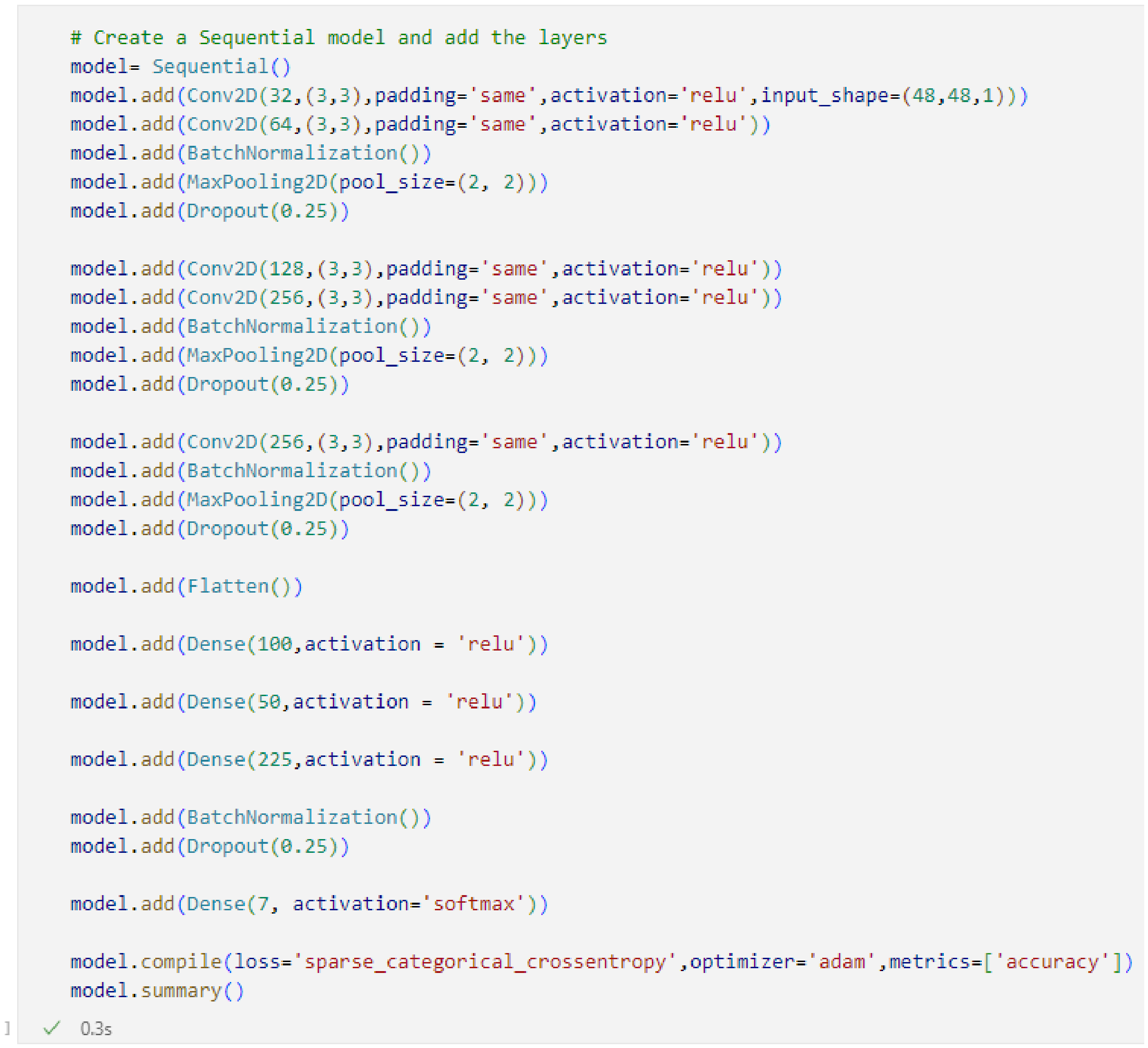
Figure 8: CNN classification confusion matrix

A confusion matrix visually conveys information about how accurate the model can make predictions. It is important to make predictions using a test dataset that was kept separate during training to ensure that the model is not overfitting. After making predictions using a test dataset, the confusion matrix outputs how many predictions were correct and how many were incorrect. The legend represents the number of predictions made which is reflected in the confusion matrix blocks. The x-axis represents the predicted outcomes for each class and the y-axis represents the actual classes from the test dataset.

9. Tuning the Model's Hyperparameters

This project has made use of a convolutional neural network to classify images into seven different classes of emotion. By nature, there is no clear best practice for tuning the hyperparameters of a CNN. The data analyst has adopted a trial-and-error method when attempting to tune the hyperparameters of the model. The data analyst has built multiple different models using different layer combinations and after testing the best model was included in the notebook.

Figure 9 illustrates the best model for this dataset:



```python
# Create a Sequential model and add the layers
model= Sequential()
model.add(Conv2D(32,(3,3),padding='same',activation='relu',input_shape=(48,48,1)))
model.add(Conv2D(64,(3,3),padding='same',activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128,(3,3),padding='same',activation='relu'))
model.add(Conv2D(256,(3,3),padding='same',activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(256,(3,3),padding='same',activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(100,activation = 'relu'))

model.add(Dense(50,activation = 'relu'))

model.add(Dense(225,activation = 'relu'))

model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Dense(7, activation='softmax'))

model.compile(loss='sparse_categorical_crossentropy',optimizer='adam',metrics=['accuracy'])
model.summary()
```

✓ 0.3s

Figure 9: CNN Model

This model consists of 5 layers and requires an input shape of 48,48,1. This means that the model is expecting an input shape of 48x48 pixels and one metric (greyscale data information). The activation function SoftMax and loss type Sparse Categorical Crossentropy was used since it's the most appropriate for the dataset's data format.

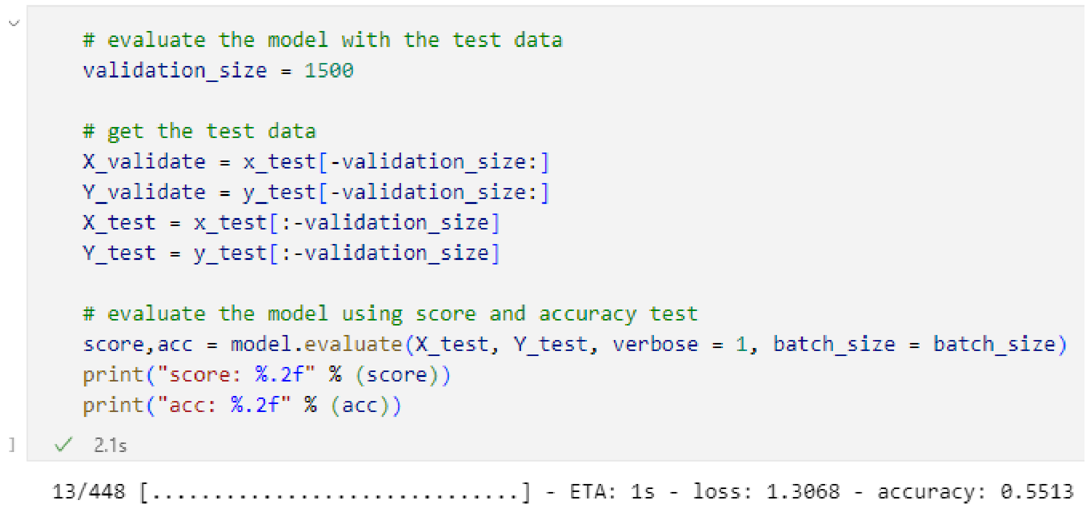
10. Evaluating the Model:

During analysis, the data analyst built three different models using three different machine learning algorithms. Each model was evaluated using the following methods:

- Accuracy Score
- Classification Report
- Confusion Matrix

Please note: this section only explains the steps for one model, but the same steps were applied to all three models. The results are discussed later in section 11.

Figure 10 illustrates the model evaluate method using accuracy and loss:



```
# evaluate the model with the test data
validation_size = 1500

# get the test data
X_validate = x_test[-validation_size:]
Y_validate = y_test[-validation_size:]
X_test = x_test[:-validation_size]
Y_test = y_test[:-validation_size]

# evaluate the model using score and accuracy test
score,acc = model.evaluate(X_test, Y_test, verbose = 1, batch_size = batch_size)
print("score: %.2f" % (score))
print("acc: %.2f" % (acc))
```

```
13/448 [..............................] - ETA: 1s - loss: 1.3068 - accuracy: 0.5513
```

Figure 10: Model Evaluation using validation data

The model has been evaluated using a validation dataset that is taken from the y_test variable which was not included in the model training. This ensures that the model is not evaluating against the data it learnt from and producing a false accuracy. Accuracy refers to how often we can expect our model to correctly predict an outcome. An accuracy score of 0.55 is a good and this means that majority of the time it will make accurate predictions. Loss refers to the loss function of the model, this measures how different predicted outputs compare to the expected outputs (Kamali, 2021). The model has a loss function of 1.3 which is a decent value for loss. The lower the loss the better the model will perform due to low number of errors.

Figure 11 illustrates the CNN classification confusion matrix:

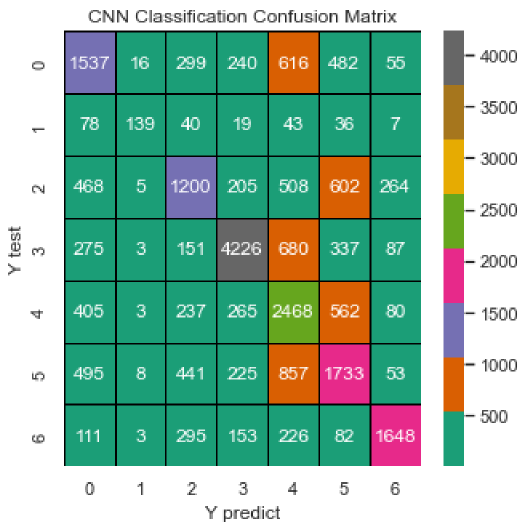
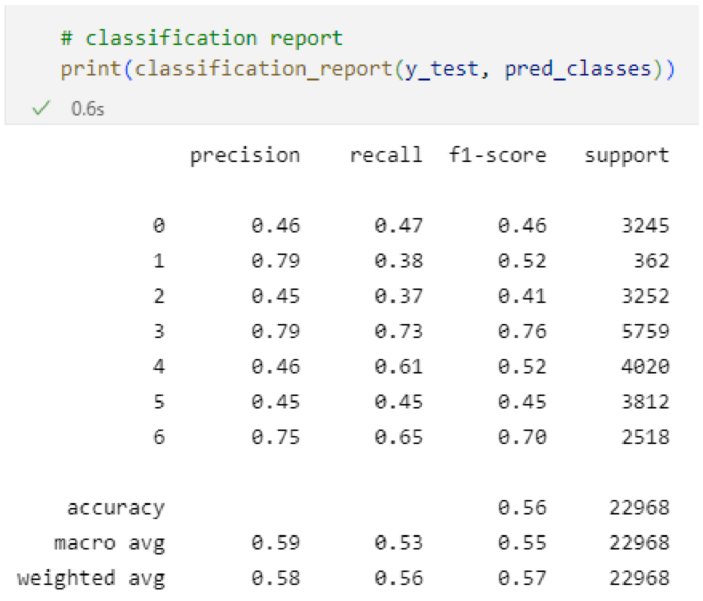


Figure 11: CNN classification confusion matrix

We can further investigate the model's prediction performance visually using a confusion matric. A confusion matrix illustrates how many predicts were incorrectly predicted. From the confusion matrix we can see that the model correctly predicted majority of the classes but the classes that performed the worst were: angry, disgusted, sad. We already know that these classes consist of small sample sizes therefore we can assume that they are under performing for this reason.

Figure 12 illustrates the Sklearn classification report:



```
# classification report
print(classification_report(y_test, pred_classes))
✓ 0.6s
              precision    recall  f1-score   support

           0       0.46      0.47      0.46      3245
           1       0.79      0.38      0.52       362
           2       0.45      0.37      0.41      3252
           3       0.79      0.73      0.76      5759
           4       0.46      0.61      0.52      4020
           5       0.45      0.45      0.45      3812
           6       0.75      0.65      0.70      2518

    accuracy                           0.56     22968
   macro avg       0.59      0.53      0.55     22968
weighted avg       0.58      0.56      0.57     22968
```

Figure 12: Sklearn Classification Report

The precision score measures the proportion of positively predicted labels that are actually correct. Recall score represents the model's ability to correctly predict the positives out of actual positives. F1_score is a performance metric that gives equal weight to both the Precision and Recall for measuring its performance in terms of accuracy (Kumar, 2022). Due to the model consisting of multiple classes, it is easier to interpret the weighted average of the model. The weighted average for precision is 0.58, this indicates that majority of the positive predicted values are correct. The weighted average for recall is 0.56, this indicates that majority of the time the model can identify the actual positives out of all positives in the dataset. A weighted average for F1-score is 0.57, this indicates that the model is somewhat optimized and can provide accurate predictions.

11. Interpreting the Results:

The data analyst has built four different models to test which method produces the best results. The models produced the following results:

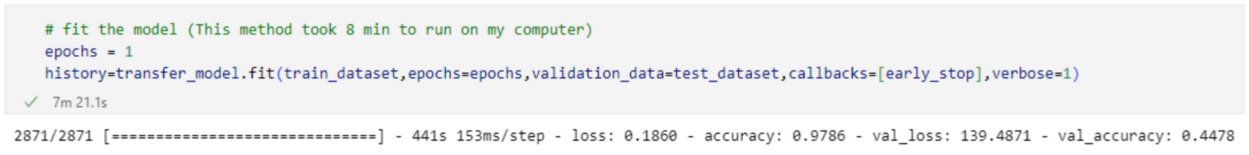
- 1st model – 75% training accuracy & 0.66 loss, 56% validation accuracy & 1.3 loss
- 2nd model (Image Data Generator) – 42% training accuracy & 1.4 loss, 51% validation accuracy & 1.2 loss
- 3rd model (TensorFlow Input Pipeline) – 76% training accuracy & 0.66 loss, 58% validation accuracy & 1.2 loss
- 4th model (Transfer Learning) - 97% training accuracy & 0.18 loss, 44% validation accuracy & 139.4 loss

The best performing model based on accuracy is the 4th model (Transfer learning), it outperformed the second-best model (3rd model) by 22%. I would not recommend using this model until it has been tested further. All three models excluding the 4th model (was not tested) were able to make correct predictions majority of the time. The validation accuracy and loss are a lot lower than the training accuracy and loss; this is due to the small size of the training and test dataset. The dataset was augmented therefore similar images have been reused multiple times, this has resulted in the model to struggle to clearly identify the underlying trends. This is not a big problem; an easy solution is to provide the model with more training data. Hundreds of thousands if not millions of images are required to successfully perform image classification and produce a good accuracy. Due to this reason, the data analyst believes that the model can be improved by providing it with more training data.

12. Possible more appropriate Methods:

A possible more appropriate method for this analysis would be to implement Transfer Learning. Transfer Learning is a machine learning technique where a model is developed for a task is reused as the starting point for a model on a second task (Brownlee, 2017). Image classification using a CNN requires large datasets of images to produce accurate results. This is due to the nature of CNNs and the high dimensionality of images, a transfer learning model can be implemented to solve this problem by accessing models that have been trained on large datasets. Another benefit of transfer learning is reducing training times due to companies like TensorFlow and Google having access to very powerful computers which specialize in processing algorithms. The last benefit of transfer learning is increasing model performance and accuracy. A model that has been trained and refined by TensorFlow is probably more likely to produce better results than a newly trained model, this is due to TensorFlow's access to vast volumes datasets and powerful computers.

Figure 13 illustrates a simple MobileNetv2 transfer learning model:



```
# fit the model (This method took 8 min to run on my computer)
epochs = 1
history=transfer_model.fit(train_dataset,epochs=epochs,validation_data=test_dataset,callbacks=[early_stop],verbose=1)
✓ 7m 21.1s
2871/2871 [==============================] - 441s 153ms/step - loss: 0.1860 - accuracy: 0.9786 - val_loss: 139.4871 - val_accuracy: 0.4478
```

Figure 13: Transfer Learning model fit

It is important to note that this transfer learning model has not been tuned and was only tested using one epoch due to long training times. The training times of this model is long due to MobileNetv2's complex 52-layer model. This was just a proof of concept and to see whether a transfer learning model could outperform the other models. From Figure 13, we can interpret that the transfer learning model outperformed the previous best model within one epoch. The transfer learning model increased the accuracy by 22%. To conclude, a transfer learning model is a more appropriate method because it provides the model with large amounts of training data, reduces the required processing power, and has the potential to increase model accuracy.

13. Summary of Code:

The model created follows the structure of Figure 14:

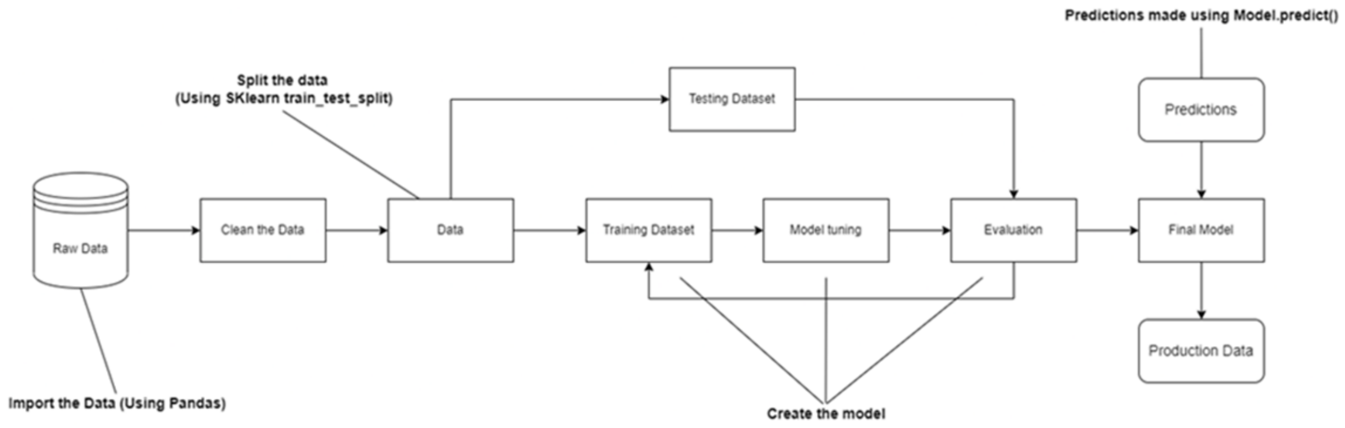


Figure 14: Summary of Code diagram

The raw data is first imported using Pandas and image reading libraries. The data is visualized using graphs to gain an understanding of any potential trends. The data is then cleaned using Featured engineering to eliminate any null values and one-hot encode. The next step is to split the dataset into training and test data (80% train and 20% test data). The training data is used to train the model and the algorithm selected to train the model is Convolutional Neural Network. After building the model, the hyperparameters are tuned by trial and error. After tuning the hyperparameters, the model is evaluated using the test data and the Sklearn libraries (classification report, accuracy score, confusion matrix).

14. Conclusion:

The dataset that has been selected for image classification consists of PNG images of the human face displaying different types of emotion. The dataset consists of over 35 000 images and was augmented to increase the dataset to over 100 000 observations and 7 classes. After performing Exploratory Data Analysis, it is clear that the images were taken in greyscale and the dataset is imbalanced. The data analyst applied data augmentation and the early stopping method to combat overfitting. The model that was created with a TensorFlow Input Pipeline (3rd model) outperformed the other two models with an accuracy of 75%. The best performing model, but it was not tested thoroughly, was the MobileNetv2 transfer learning model (4th model) which had an accuracy of 97%. A possible use case for this model is to gather real-time feedback from customers by recording their faces and predicting their emotion. For example, this model can be used to analyse customers waiting in queues and determine whether these customers are starting to become unhappy. A manager can then be alerted to try and perform damage control. Another example of a use case for this model, the model can be integrated with a mobile application and used to gather real-time feedback about certain app features or marketing. The model can successfully classify the seven different emotions and differentiate between them. This model can be used to gather real-time feedback or determine the general mood of customers, this is important information because a large amount of business rely on customer satisfaction as a business model. A satisfied customer will ultimately result in an increase in profits. The outcome achieve by this model is that deep learning can be used to identify human emotion. All the analysis questions are answered by the results produced by the model and the model was able to successfully categorise the emotions of humans into seven different categorizes.

References

Brownlee, J., 2017. *Machine Learning Mastery.* [Online]
Available at: https://machinelearningmastery.com/transfer-learning-for-deep-learning/
[Accessed 21 11 2022].

Brownlee, J., 2020. *Machine Learning Mastery.* [Online]
Available at: https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/
[Accessed 12 09 2022].

Google Developers, 2022. *ML Practicum: Image Classification.* [Online]
Available at: https://developers.google.com/machine-learning/practica/image-classification#:~:text=Image%20classification%20is%20a%20supervised,the%20input%20to%20the%20model.
[Accessed 21 11 2022].

IBM Cloud Education, 2020. *Supervised Learning.* [Online]
Available at: https://www.ibm.com/cloud/learn/supervised-learning
[Accessed 21 11 2022].

Kamali, K., 2021. *Training Galaxy Project.* [Online]
Available at: https://training.galaxyproject.org/archive/2021-06-01/topics/statistics/tutorials/RNN/tutorial.html#:~:text=A%20loss%20function%20measures%20how,used%20to%20train%20the%20model.
[Accessed 14 09 2022].

Kumar, A., 2022. *Vital Flux.* [Online]
Available at: https://vitalflux.com/accuracy-precision-recall-f1-score-python-example/
[Accessed 06 06 2022].

Saha, S., 2018. *Towards Data Science.* [Online]
Available at: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
[Accessed 17 11 2022].

TensorFlow, 2022. *Image Classification.* [Online]
Available at: https://www.tensorflow.org/lite/examples/image_classification/overview
[Accessed 21 11 2022].