Optimizing and Creating CNN’s for Facial Expression Recognition

Group-7-CP468-Artificial Intelligence

Ajay Sodhi, Justin Chiu

2024-08-02

Table Of Contents

Abstract

An individual's facial expression provides vital insight into what they are feeling and experiencing. Being able to recognize an emotion has many practical applications in various fields such as whether distress or fatigue is perceived in a driver's face, aiding psychologists better to understand a patient's emotions and mental state, or being able to evaluate customer satisfaction when using a product or service.

The objective of this project is to create and optimize a CNN from scratch as well as fine-tune three pre-trained models to perform well on our dataset. Various techniques were applied with the intent of maximizing the model's validation accuracy without overfitting the training data. For each model, we experimented with different configurations of data augmentation, optimizers such as Rmsprop and Adam, varying layers and complexities, unfreezing various layers within the pre-trained models, as well as testing techniques to mitigate overfitting such as Dropout, early stopping, L2 Regularization and elastic net regression. These models are trained to classify images from a dataset found on Kaggle created by Pierre-Luc Carrier and Aaron Courville which features 28,079 files classified into 6 different classes each representing a facial expression such as Anger, Fear, Happy, Neutral, Sad, and Surprise. Each file is grayscale 48x48 jpg which features a facial expression. It is worth noting that each face occupies about the same amount of space in each image as it is centred on the individual's face. The dataset is also split into two folders titled "Testing" and "Training". Training consists of about 80% of the files and is meant to be trained on a model while testing consists of about 20% of the files and is the validation set. The success of each model is evaluated against the following metrics: accuracy, loss, validation accuracy, validation loss, and loss. To better understand model performance, a confusion matrix and ROC curve are displayed for each model. Pre-trained models include ResNet50, MobileNetV2 and InceptionV3

1 Introduction to the Problem

With the rise of artificial intelligence (AI) and machine learning (ML), our project aims to redefine approaches to developing AI systems, specifically focusing on facial expression identification. Having the ability to recognize and identify facial expressions is a key factor for interpersonal communication and can also have various applications in many fields such as psychology.

To achieve our AI model we will use the implementation using TensorFlow, an open-source machine learning framework by Google to enhance the machine learning and artificial intelligence of facial expression identification. More specifically we would be using the Keras API as well, a high-level API for TensorFlow that makes code readable and provides clear error messages to keep development as smooth as possible. Keras API will allow us to design and train neural networks to make our development more efficient and user-friendly by providing clear and readable code.

Looking at past efforts, challenges that will be faced are if the facial expression identification detection is live feed, it will be challenging to detect and identify moving facial features. On top of that, the AI would need to keep updating the live information. Another challenge is to detect accurate facial expressions regardless of age differences and gender. We aim to overcome the limitations of previous artificial intelligence and machine learning by leveraging the use of datasets and APIs to eventually enhance our future set.

The primary problem that the artificial intelligence of facial expression identification plans to address is the unpredictable nature of human emotions. Being able to predicate human emotions is essential in the market industry. For example, having facial expression identification could help companies know if customers are enjoying their product or not. This is just one example of the need for facial expression identification. By being able to identify challenges from past works, our group will be able to operate efficiently and tackle the challenges that matter most. This also leads to opportunities for improvement as leveraging past used accurate datasets and samples would bring the artificial intelligence up to date with other facial expression identification.

1.1 Instructions

Firstly, download the dataset that contains the facial expressions from the following link: https://www.kaggle.com/datasets/apollo2506/facial-recognition-dataset. The first block of code in the Python notebook will prompt the user to upload a file, please upload the zip file downloaded which should be named “archive.zip”. From here, run all subsequent blocks of code excluding the fully commented out blocks, and the blocks of code at the end which contain models that yielded worse results. These models are found underneath a header that appears after all models have been trained.

2. Solutions

Some of the solutions discussed were following Keras documentation to start off creating the base of our CNN. As our task was to create one Keras CNN from scratch and adjust three different pre-trained CNNs. Our Keras CNN needed to implement solutions to increase accuracy and decrease loss. Some of the proposed solutions were L2 regularization, dropout and elastic net regression. We attempted to incorporate all of these methods but we found that elastic net regression reduced our validation accuracy by about 3-4%. Different values for L2 regularization and dropout were experimented with until we settled on a solution that yields a relatively high validation accuracy without overfitting. Various methods of data augmentation were also attempted such as flipping the images horizontally, zooming in, altering contrast, rotating images, and even adding a “salt and pepper” layer which randomly makes about 30% of the pixels either white or black while still maintaining the original features of the image. For the model we built from scratch, any sort of data augmentation yielded worse results so we decided to not augment the data in any way. Attempts were also made to increase our validation accuracy by applying the “salt and pepper” layer to every image in the dataset and then appending it to the end which essentially doubled the amount of files. We figured that the unsatisfactory results were because the dataset is relatively small, however after increasing the size of the dataset with augmented images the validation accuracy decreased while the training accuracy increased. Slight data augmentation was later used for the pre-trained models since they yielded better results. We also tried models of varying complexities as we experimented with models that had fewer layers and more layers but we found that 11 layers gave the best results for the from scratch model. For our pre-trained models, we found that unfreezing some layers yielded better results as it allowed these models to better pick up the features of the dataset and classify images with higher accuracy. Layers were also added on top of these pre-trained models to specifically output for the 6 classes given in the dataset.

3. Results

3.1 From Scratch Model

This model consists of 11 layers and is designed to classify an image of a face into 1 of 6 categories being Angry, Fear, Happy, Neutral, Sad, and Surprise. The first rescaling layer is meant to normalize the images by scaling the pixel values to be between 0 and 1 which improves the performance of the model. The next Conv2D layer is meant to extract basic features such as edges while the MaxPooling2D layer reduces the dimensions of the input for the next layer, which helps extract dominant features while decreasing computation. More Conv2D and MaxPooling2D layers are added before a Dropout layer is added. This Dropout layer sets 40% of the input units to zero to prevent overfitting. Next, a Flatten layer converts the feature maps into a single vector that can be interpreted by the Dense layer. The Dense layer is responsible for interpreting the features that have been extracted by the Conv2D and MaxPooling2D layers. L2 regularization is applied with a value of 0.001 which helps to combat overfitting before the final output layer which outputs the final predictions. When compiling the model, we decided to use the RMSprop optimizer because it yielded better results than the Adam optimizer. The loss function used is sparse_categorical_cross entropy which interprets the labels as integers and applies a softmax function which outputs the probability of the input belonging to a certain class. This model is trained on 30 epochs using the entire “Training” folder as the train dataset, and the “Testing” folder as the validation dataset which results in an 80% - 20% split. After training for 30 epochs, the metrics were measured as: Accuracy = 0.6148, loss = 1.1917, val_accuracy = 0.5776 and val_loss = 1.275.

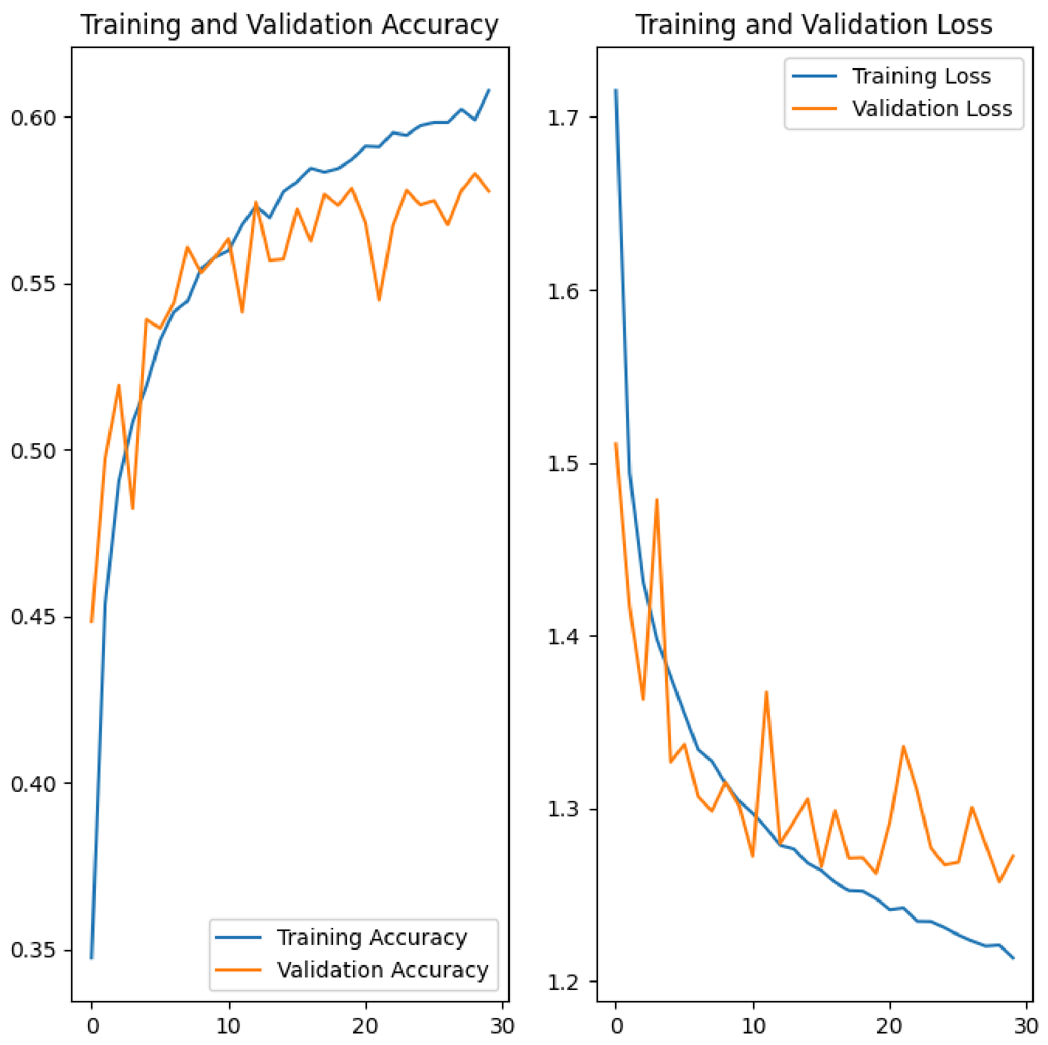
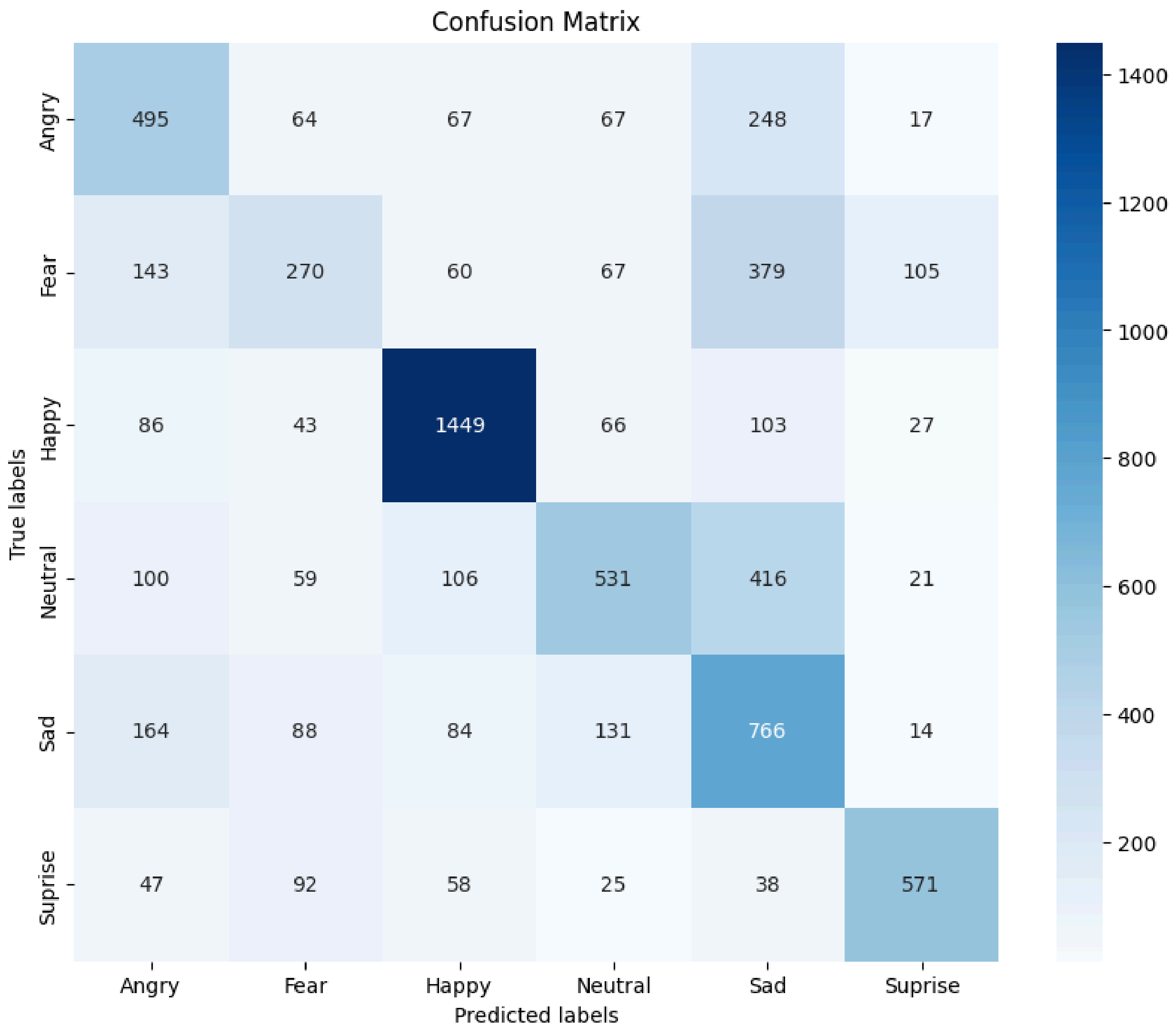


Figure 3.1.1: Depicts training and validation accuracy and loss as more epochs are trained

Overall, Figure 3.1.1 shows a steady increase in performance but the validation loss and accuracy is shown to be more volatile which suggests that the model has issues with generalization.



True \ Predicted	Angry	Fear	Happy	Neutral	Sad	Suprise
Angry	495	64	67	67	248	17
Fear	143	270	60	67	379	105
Happy	86	43	1449	66	103	27
Neutral	100	59	106	531	416	21
Sad	164	88	84	131	766	14
Suprise	47	92	58	25	38	571

Figure 3.1.2: Confusion Matrix for the from scratch model

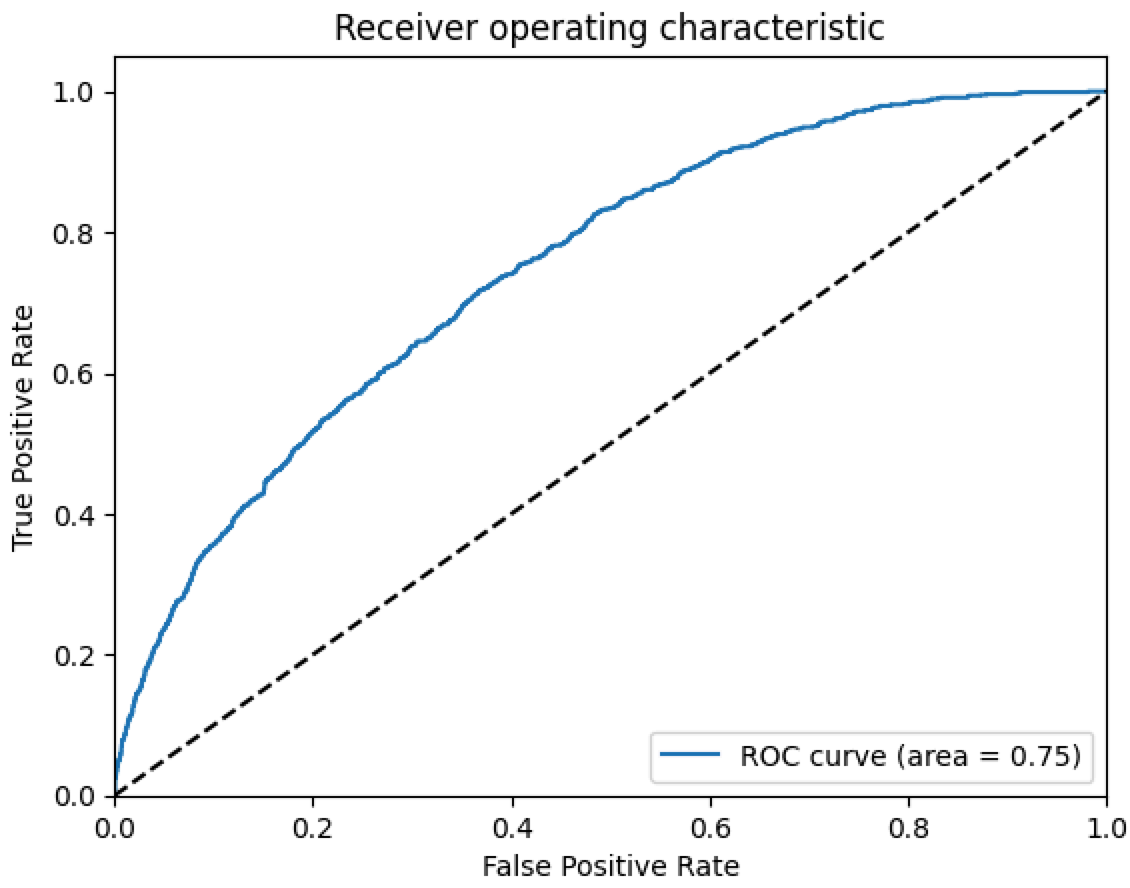


Figure 3.1.3: AUROC curve for the from scratch model

Figure 3.1.2 is a confusion matrix that displays false positives, false negatives, true negatives, and true positives for the from scratch model. Predictions are made against the validation set. This model excels at detecting happy and sad faces while it struggles to recognize fear and neutral expressions. Figure 3.1.3 is an AUROC curve and represents the model's ability to distinguish between positive and negative classes. The area under the ROC curve being equal to 0.75 conveys that the model is able to effectively distinguish positive and negative classes as it performs significantly better than a random classifier.

3.2 ResNet50 Model

ResNet50 is a pre-trained CNN model that is excellent at image classification. This dataset was initially trained on the ImageNet dataset which suggests that it should excel at facial expression recognition since it has the ability to extract meaningful features from images. Firstly a ResNet50 model is loaded with pre-trained weights from ImageNet which we found to significantly increase our validation accuracy. The top layer is not included since a Dense layer and output layer are added to specifically classify images in the dataset. Before these layers are added to the loaded ResNet50 model, the images are resized from 48x48 to 224x224 since ResNet50 is trained to accept larger images. Most of the initial layers of the model are frozen to retain their learned features while the last 35 layers of the model are trainable which allows the model to better learn high level features from the input. Added to this pre-trained model are a GlobalAveragePooling2D layer which reduces dimensions to a single vector, a Dense layer to learn features, A BatchNormalization layer and a Dropout layer to provide regularization and reduce overfitting, and lastly an output layer. Before the data is provided as input to the model, data augmentation is applied. Images from the dataset are augmented by rotating images 15 degrees, shifting the width and height of the images by 15%, flipping the images horizontally and zooming into the image by 15%. Early stopping is also implemented to reduce overfitting before the model is compiled using Rmsprop as the optimizer and "sparse_categorical_crossentropy" as the loss function. The fine-tuned ResNet50 model is trained for 10 epochs and the results at the tenth epoch are as follows: accuracy = 0.6822, loss: 0.875, val_accuracy = 0.6484, and val_loss = 0.9714

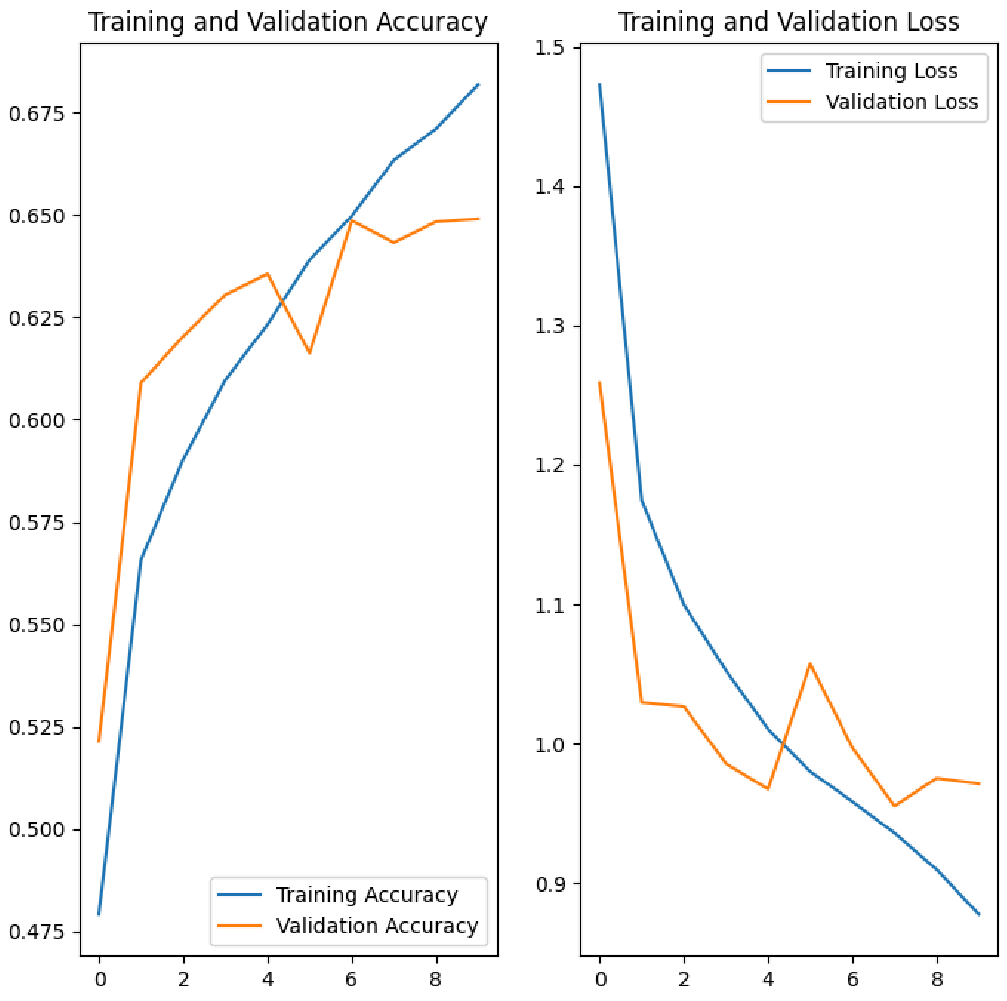


Figure 3.2.1: Depicts training and validation accuracy and loss as more epochs are trained

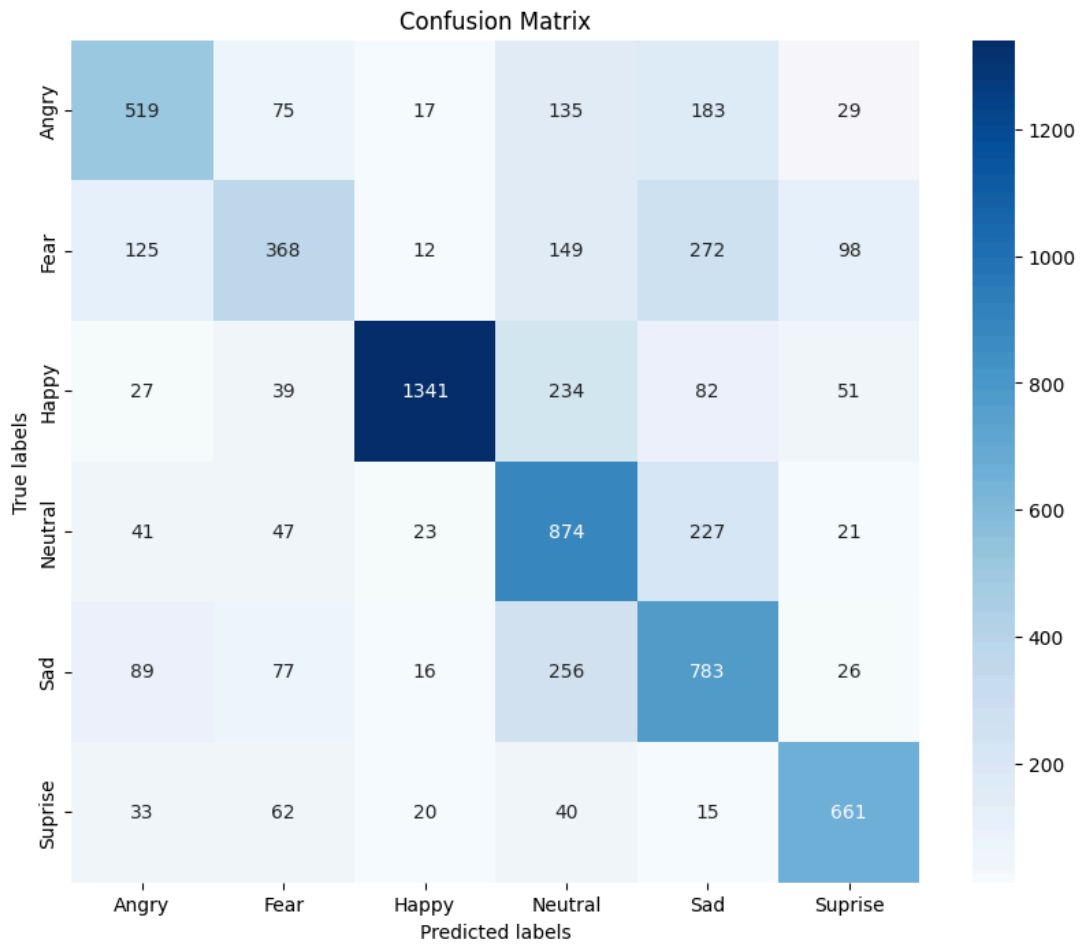


Figure 3.2.2: Confusion Matrix for the ResNet50 model

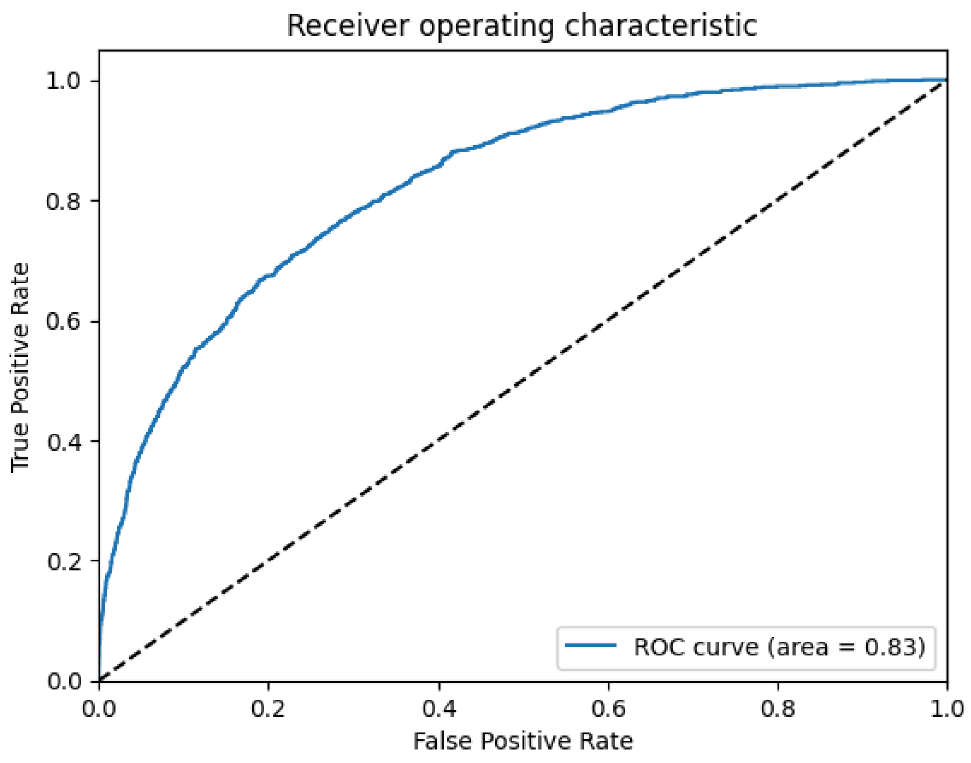


Figure 3.2.3: AUROC curve for the ResNet50 model

Overall, This ResNet50 model performs better than the from scratch model as it has a higher validation accuracy, a larger area underneath the ROC curve, and is better able to distinguish between classes which can be seen in the confusion matrix.

3.3 MobileNetV2 Model

Similar to ResNet50, MobileNetV2 is a powerful pretrained model for image classification which makes it suitable to recognize and distinguish between facial expressions. Similar to the methodology used in the ResNet50 model, the MobileNetV2 model is loaded with imageNet weights and the top layer is not included as we plan to add on more layers. Inputs are resized to 224x224 before a GlobalAveragePooling2D and dense layer are added. The fine-tuned MobileNetV2 model is compiled using the Adam optimizer and the loss function is set to categorical_crossentropy. For this model, the first 30 layers are frozen while all layers after are unfrozen to better learn and extract features from our images. The learning rate is also adjusted to provide more stability and mitigate overfitting. The same data augmentation is applied however we adjust the 15% down to 10% as it yields better results. The MobileNetV2 model is trained for 10 epochs and the results at the tenth epoch are as follows: accuracy = 0.6535, loss = 0.9133, val_accuracy = 0.6290, val_loss = 0.9866.

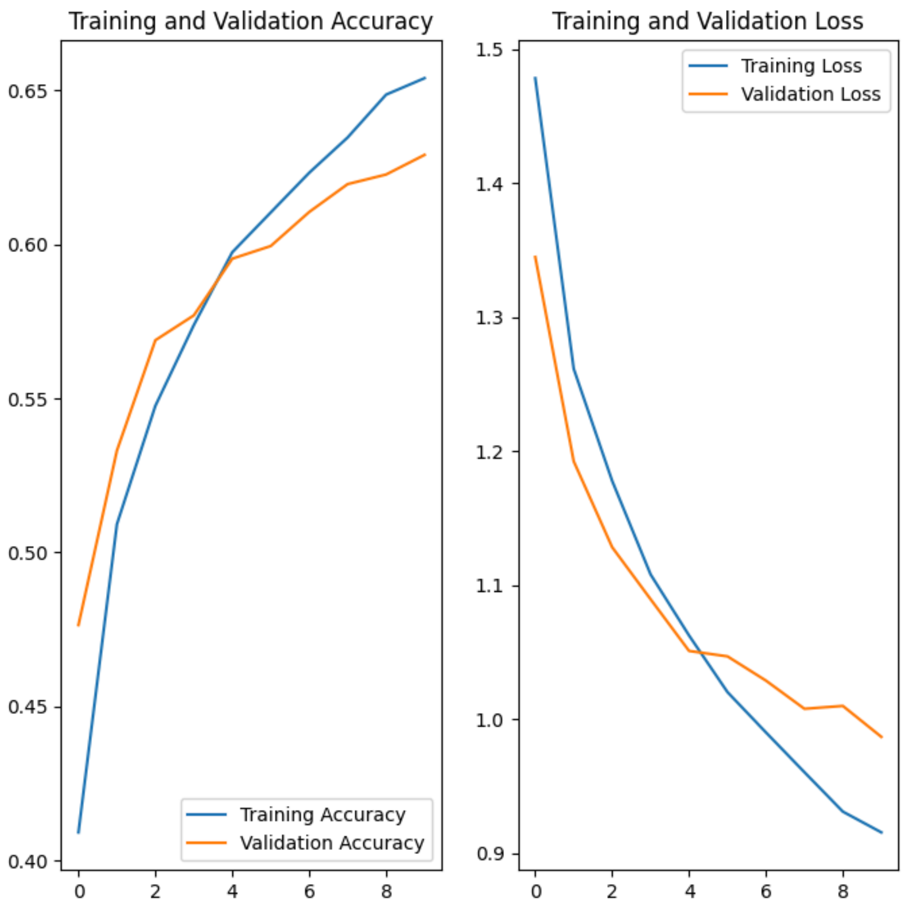
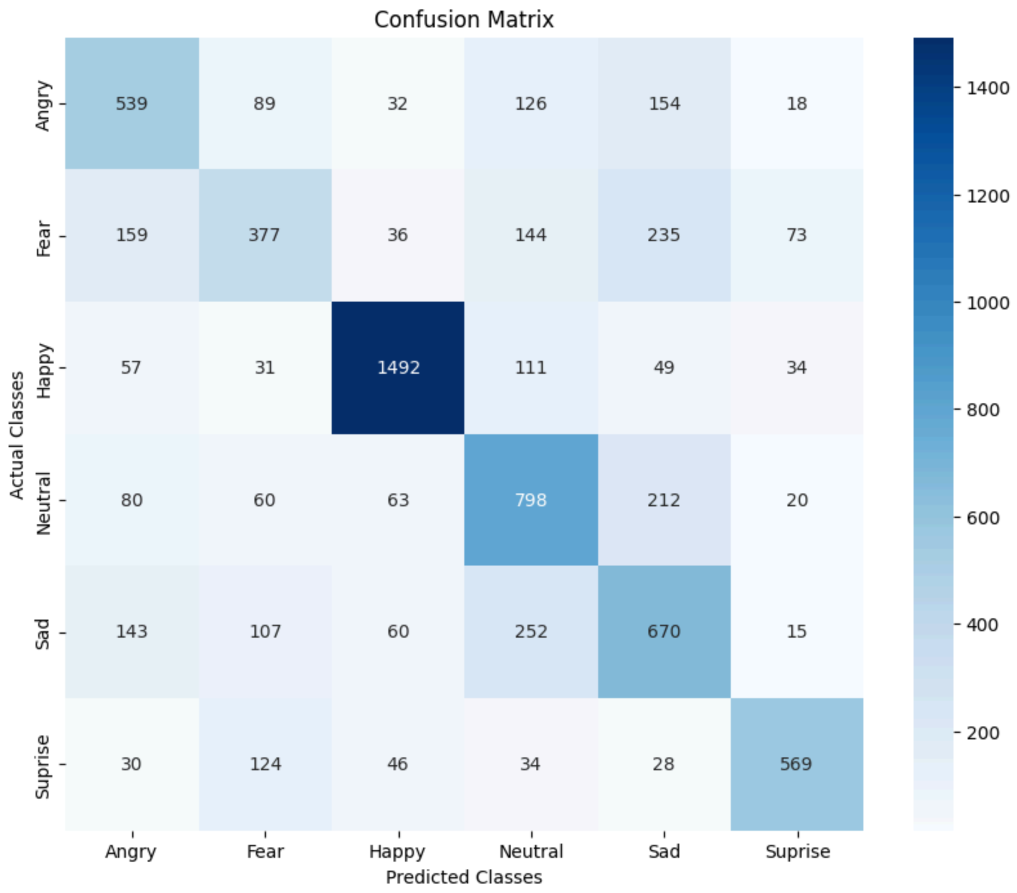


Figure 3.3.1: Depicts training and validation accuracy and loss as more epochs are trained



Actual \ Predicted	Angry	Fear	Happy	Neutral	Sad	Suprise
Angry	539	89	32	126	154	18
Fear	159	377	36	144	235	73
Happy	57	31	1492	111	49	34
Neutral	80	60	63	798	212	20
Sad	143	107	60	252	670	15
Suprise	30	124	46	34	28	569

Figure 3.3.2: Confusion Matrix for the MobileNetV2 model

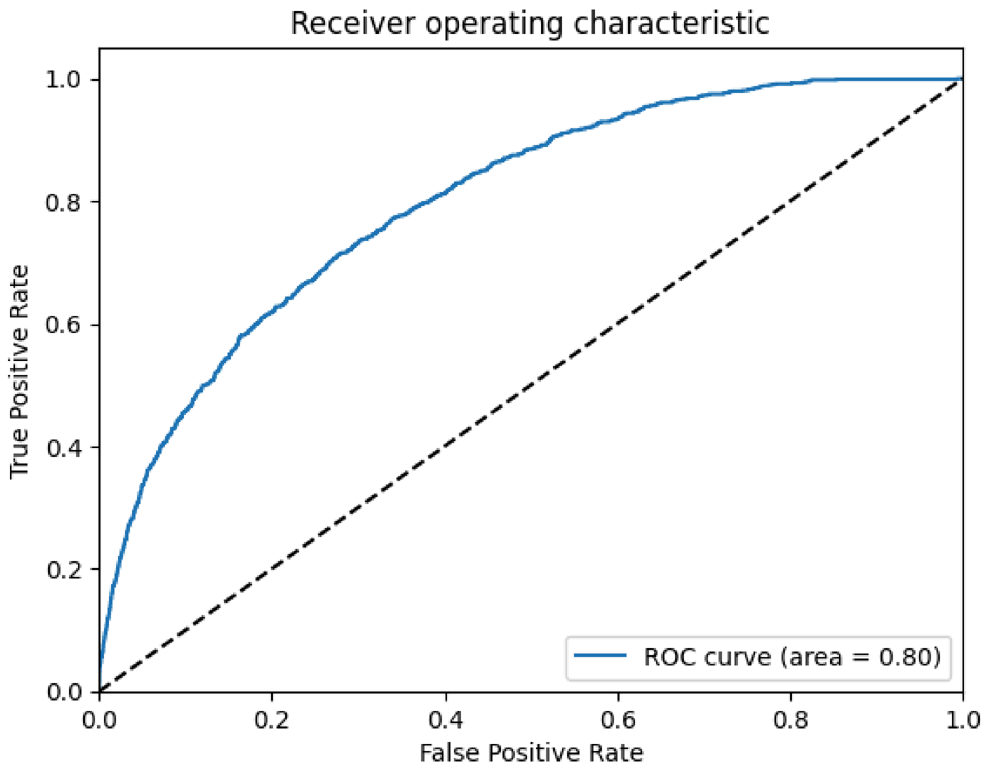


Figure 3.2.3: AUROC curve for the MobileNetV2 model

Overall, this model performs slightly worse than the ResNet50 model but better than the from scratch model. The validation accuracy and validation loss also appear to be much more stable as epochs are trained.

3.4 InceptionV3 Model

InceptionV3 is a powerful and computationally efficient image classification model that is trained on the ImageNet database. We chose this model to detect and distinguish between facial expressions since it excels at image classification. Similar to the previous pre-trained models, InceptionV3 is loaded with the imagenet weights and the top layer is not included. Instead of resizing images to 224x224, InceptionV3 requires images to be at least 299x299. The previous methodology is followed as we add a GlobalAveragePooling2D layer and a dense layer. Specifically for this model, we also added two Dropout functions with a value of 0.5 because we experienced overfitting. The model is compiled using the Adam optimizer and the loss function is set to categorical_crossentropy. Oddly enough for this model, training the earlier layers and freezing the top layers provided better results. The first 90 layers are trainable while we freeze every layer afterwards. The same data augmentation techniques are implemented to the training data before it is interpreted by the model; however, each augmentation is adjusted to be 0.2 or 20%. Early stopping is used to prevent overfitting before the model is trained on 10 epochs. The results at the tenth epoch are as follows: accuracy = 0.6516, loss = 0.9284, val_accuracy = 0.6593, and val_loss = 0.9183.

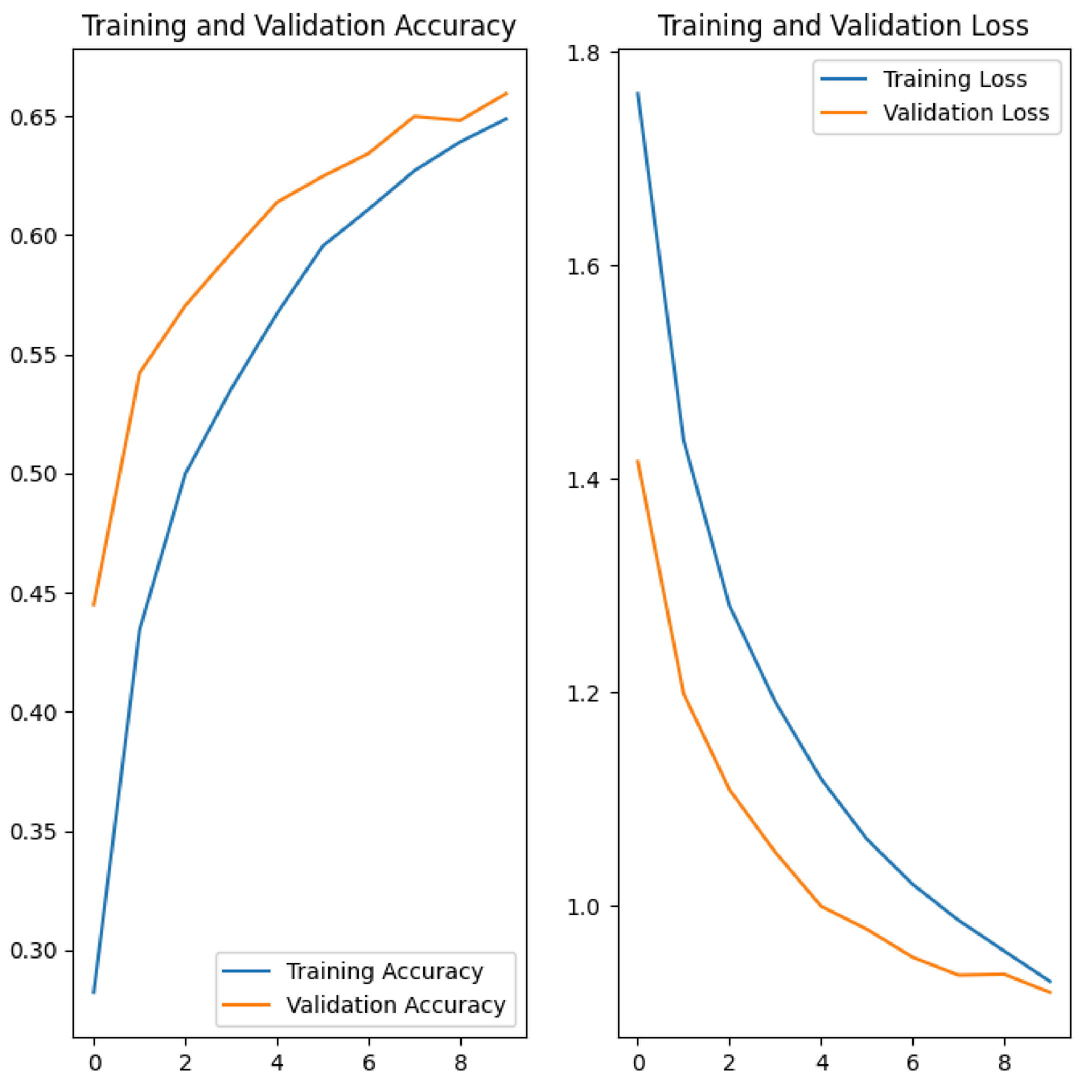
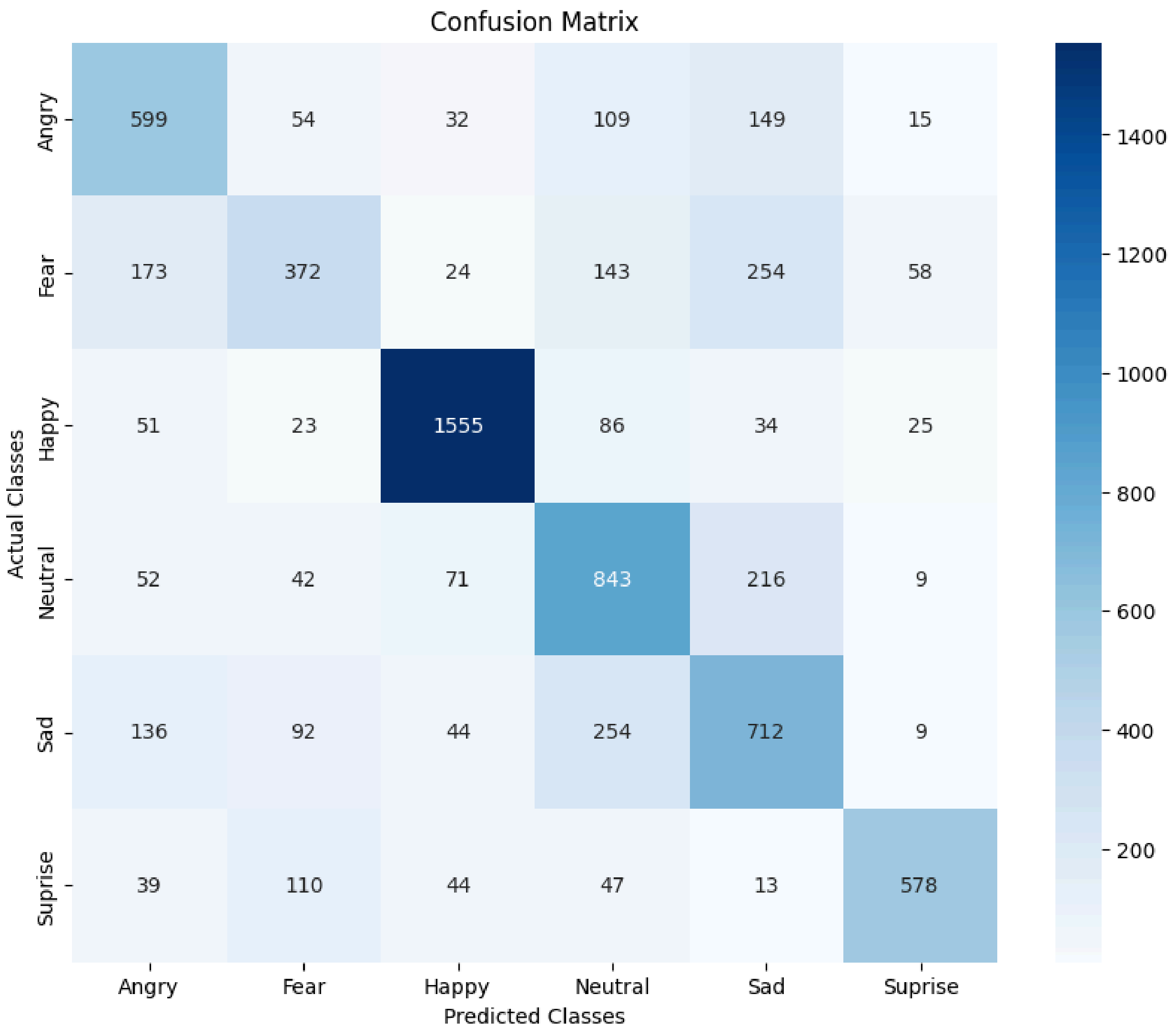


Figure 3.4.1: Depicts training and validation accuracy and loss as more epochs are trained



	Angry	Fear	Happy	Neutral	Sad	Suprise
Angry	599	54	32	109	149	15
Fear	173	372	24	143	254	58
Happy	51	23	1555	86	34	25
Neutral	52	42	71	843	216	9
Sad	136	92	44	254	712	9
Suprise	39	110	44	47	13	578

Figure 3.4.2: Confusion Matrix for the InceptionV3 model

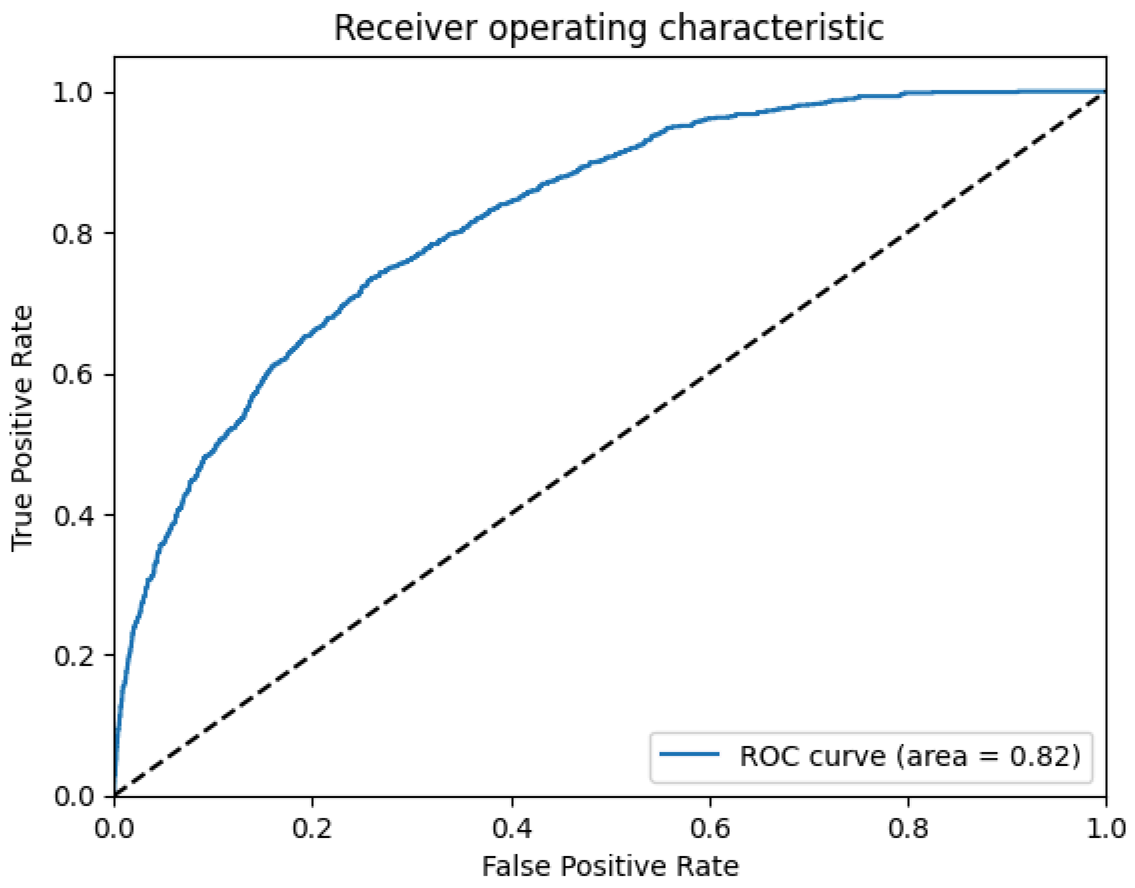


Figure 3.2.3: AUROC curve for the InceptionV3 model

Overall, this is our best performing model as it has the highest numbers for evaluations such as validation accuracy and the area underneath the ROC curve.

3.5 Ensemble Model

Lastly, the Ensemble model was created by combining the fine-tuned ResNet50, MobileNetV2, and InceptionV3 models. Predictions from each of these models are combined to better classify images. All 3 pre-trained models are loaded into a class. Using the call function, input is passed to these 3 models and each of them individually outputs a value that is concatenated into a single vector. This vector is inherently more descriptive as it is the result of combining information learned from each of the 3 models. A dense layer is then added which classifies the output. The ensemble model is compiled with the optimizer set to Adam and the loss function set to sparse_categorical_crossentropy. Each of these 3 pre-trained models are frozen meaning that they cannot learn new information. Early stopping is initialized with a patience of 3 before the model is trained on 10 epochs. Since the model is not able to learn, the training ends on the 4th epoch which results in the following evaluation: accuracy = 72.07, val_accuracy = 0.6672

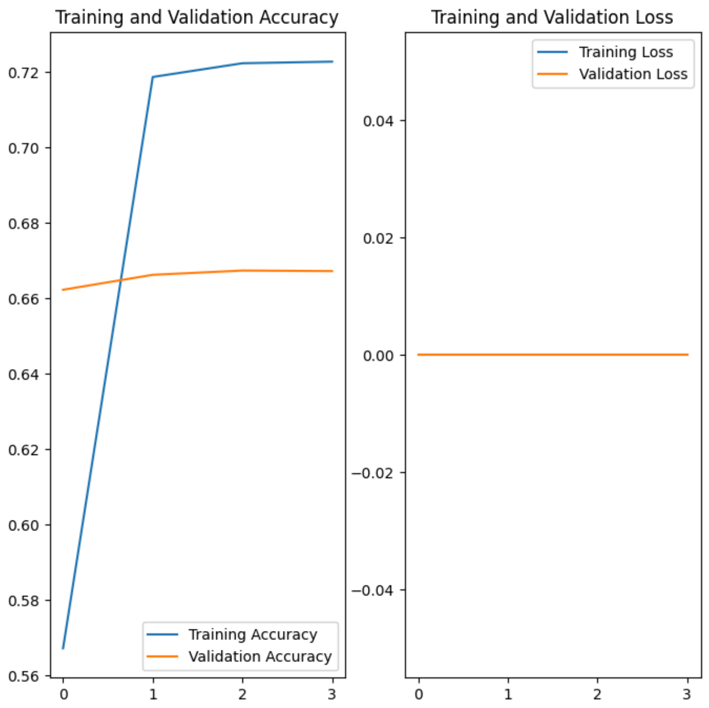


Figure 3.5.1

3.6 Gradio

Gradio was used at the end of training for all of our CNN’s. Each model used .save() so we were able to load our models into our Gradio segment. Gradio takes the input image and runs the CNN to determine which type of facial expression is being shown in the image. In Gradio the image is first resized and reshaped to correctly pass through our models and then we create a classify function for each model to determine which prediction is being made for each one. Gradio allowed us to have one image and run each model separately in different tabs to visualize the difference in outcomes and predictions for each of our models.

4 Conclusion

After fine-tuning and training these models, the ensemble model performed the best. The pre-trained models outperformed the model created from scratch which is expected as these models are more complex and trained on robust datasets which makes them more powerful. The results for the best performing model are somewhat satisfactory given that the validation accuracy is at 66.72%, which leaves room for improvement. Training and fine-tuning these models revealed some issues with the dataset that were unforeseen. Firstly, there is quite a discrepancy between the number of files within each class. For example, the training directory contains 7215 files belonging to the “Happy” class and only 3171 files in the “Surprise” class. This may explain why all of the models were able to recognize true happy expressions much more consistently than other expressions such as surprise or fear. Additionally some files are incorrectly labeled or have ambiguous classifications. For example a neutral face was found to be classified as fear which is incorrect. Another example is two different files, each containing an image of a unique individual with their eyebrows furrowed, and mouth wide open; however, one was classified as Angry while the other classified as Fear.

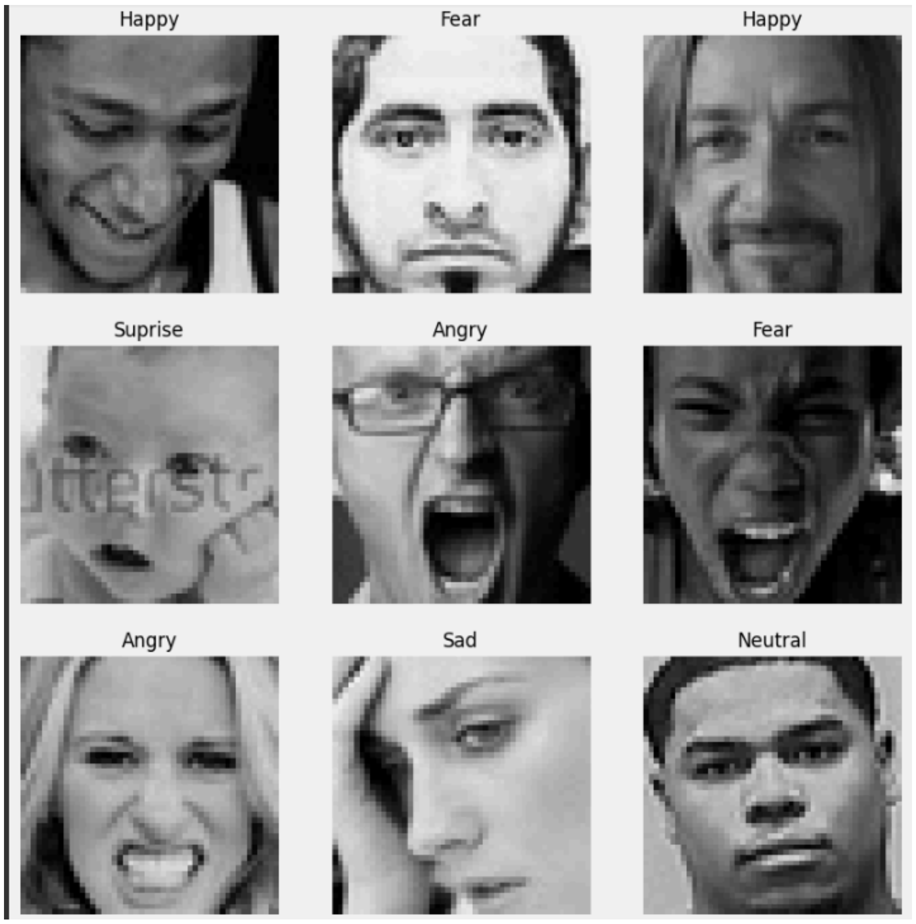


Figure 4.1 Same facial expression can be seen as two different classifications as well as a misclassified image

The last unforeseen issue is that since the images in the dataset are centered on the individual's face, our models express difficulty in classifying a facial expression when most of the actual image is not composed of a face. For example, if the face takes up about 60% of the space in an image while the other 40% is taken up by a background or body parts such as the neck and shoulders, the model will most likely misclassify the image. Moving forward, improving the dataset by ensuring files are labeled correctly and evening out the size disparity between classes seem to be the most promising solutions to yield better results.

References

Dataset:

Dahiya, G. (2020, June 30). *Facial recognition dataset*. Kaggle. https://www.kaggle.com/datasets/apollo2506/facial-recognition-dataset

Pre-trained Models:

Team, K. (n.d.-a). *Keras Documentation: Inceptionv3*. https://keras.io/api/applications/inceptionv3/

Team, K. (n.d.-b). *Keras Documentation: Mobilenet, mobilenetv2, and MobileNetV3*. https://keras.io/api/applications/mobilenet/

Team, K. (n.d.-c). *Keras Documentation: Resnet and RESNETV2*. https://keras.io/api/applications/resnet/

Libraries Used:

fchollet. (2020, April 27). *Keras documentation: Image Classification From Scratch*. Keras. https://keras.io/examples/vision/image_classification_from_scratch/

fchollet. (2023, July 10). *Keras documentation: Introduction to keras for engineers*. Keras. https://keras.io/getting_started/intro_to_keras_for_engineers/

Team, G. (n.d.). *Gradio docs*. Gradio Docs. https://www.gradio.app/docs/gradio/interface

Other References:

krishnaik06. (n.d.). *Gradio/Gradio Examples.ipynb at main · Krishnaik06/gradio*. GitHub. https://github.com/krishnaik06/Gradio/blob/main/Gradio%20Examples.ipynb