Facial Expression Recognition using Deep Learning

Adarsh Ramesh Kumar
Dane Ukken

Abstract

Facial Expression Recognition (FER) is a challenging task due to the complex and subtle nature of facial expressions. In this project, we have explored various deep learning techniques and data preprocessing strategies to improve the performance of FER systems. We utilized popular pre-trained models such as VGG16, ResNet50, MobileNetV2, and Xception for transfer learning and fine-tuning to adapt them to our specific task. Initially, we employed transfer learning by freezing the layers of pre-trained models and adding custom fully connected layers for classification. This allowed us to leverage the powerful feature extraction capabilities of these models. We then fine-tuned the models by training the last few layers, enabling them to learn more specialized features relevant to facial expressions. To further improve the model performance, we introduced data augmentation techniques such as random rotations, shifts, and flips, which enriched the training data and made the models more robust to variations in facial expressions. We also incorporated face detection and alignment using OpenCV libraries to preprocess the input images, focusing on the relevant facial regions and ensuring a consistent orientation. In addition, we experimented with the inclusion of Gaussian noise and batch normalization layers to enhance the model's generalization capabilities and reduce overfitting. These additional layers were added in various orders, such as batch normalization first, followed by Gaussian noise, and then the pre-trained model layers, to investigate their impact on the overall performance. Throughout the project, we compared the performance of each model and preprocessing strategy based on validation accuracy and other evaluation metrics such as true positive rate and false positive rate. This comprehensive analysis allowed us to identify the most effective techniques and configurations for FER, contributing valuable insights for future research in this domain.

1. Introduction

Facial expression recognition is an essential aspect of human-computer interaction, robotics, and emotion analysis. Deep learning techniques, particularly CNNs, have demonstrated remarkable success in computer vision tasks, including facial expression recognition. This paper aims to develop a deep learning model capable of accurately recognizing facial expressions from images. We discuss related work, dataset description, preprocessing, model architecture, training, hyperparameter tuning, and evaluation.

2. Related Work

Facial Expression Recognition (FER) has been an area for cutting edge research because of its real world applications in emotion analysis and human computer interaction. Deep Learning, especially CNN, has shown great potential in the field of FER. The winning approach of the FER challenge by Goodfellow et al. (2013) employed a deep CNN model. Kim et al. (2015) used a hierarchical committee of deep CNNs. Their model used multiple deep CNNs and combined their predictions using an exponentially-weighted decision fusion mechanism. Lopes et al. (2017) demonstrated the use of multiple strategies like data augmentation and transfer learning in improving the performance of CNN models in FER. In this paper, we explore multiple pre-trained models for facial expression recognition.

3. Methodology

3.1. Dataset

In this report, we present our approach to emotion recognition using the FER2013 dataset. The FER2013 dataset is a large collection of facial images in grayscale, annotated with seven emotion labels: anger, disgust, fear, happiness, sadness, surprise, and neutral. The dataset consists of 48x48 pixel grayscale images of faces, which have been manually labeled by humans, with the emotions they represent. The dataset contains a total of 35,887 facial images, ensuring that our model has ample data to learn from.

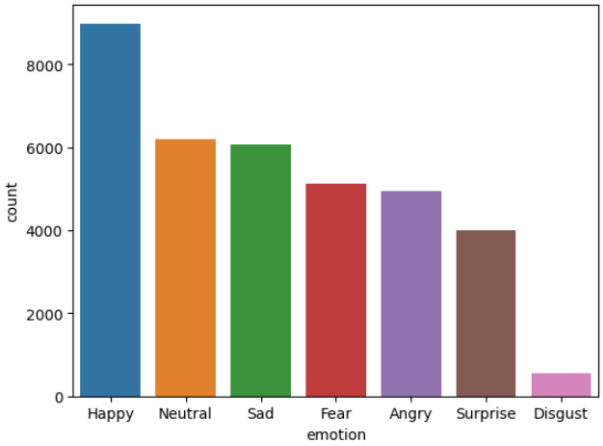

Figure 1: *Data distribution by label*

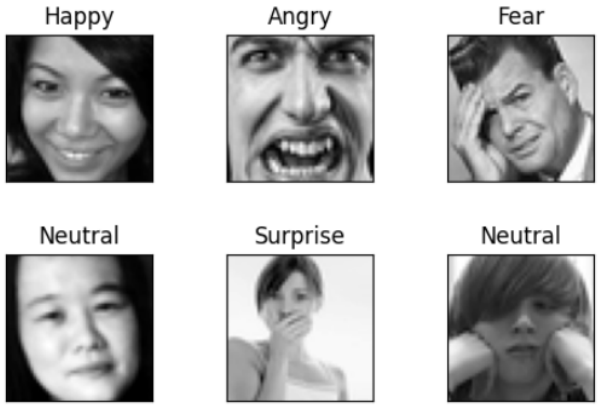

Figure 2: *Sample pixel data plotted*

The dataset is divided into three subsets based on their usage: training, private testing (validation), and public testing (test). We have utilized the images tagged with 'Training' for training our models, which consists of 28,709 images. To tune our model's hyperparameters and select the best model architecture, we used the 'Private Testing' set containing 3,589 images as our validation dataset. Finally, we assessed our model's performance on unseen data using the 'Public Testing' set, comprising 3,589 images.

3.2. Data Preprocessing

The data preprocessing steps included resizing images, normalizing pixel values, and splitting the data into training (80%), validation (10%), and testing (10%) sets. The preprocess_input function for the corresponding models are used to get the data in the required format. Finally, we obtain the emotion labels for each set by applying one-hot encoding to the original labels using the pd.get_dummies function from the Pandas library

3.3. Model Architecture

A CNN-based models are known to have great results with image classification. In this study, we focused on fine-tuning existing pre-trained models for facial expression recognition, rather than developing custom CNN architectures. Fine-tuning involves using a pre-trained model as a starting point and retraining it on the target dataset with minor adjustments to adapt the model to the specific task. We selected four popular pre-trained models to explore their performance in facial expression recognition: VGG16, ResNet50, MobileNetV2, and Xception. These models have achieved state-of-the-art results in various computer vision tasks and provide a solid foundation for our task.

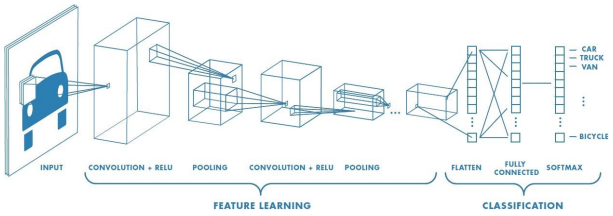

Figure 3: *CNN model*

VGG16 is a deep CNN model proposed by Simonyan and Zisserman from the Visual Geometry Group at Oxford University. It consists of 16 weight layers, including 13 convolutional layers and 3 fully connected layers. VGG16 was trained on the ImageNet dataset and has demonstrated excellent performance in image classification tasks. In our experiments, we removed the top fully connected layers and added a global average pooling layer followed by a fully connected layer with softmax activation to output probabilities for the seven emotion classes.

ResNet50, a residual learning model which addresses the vanishing gradient problem by using skip connections, which allow gradients to bypass layers during backpropagation. ResNet50 has 50 layers and has been trained on the ImageNet dataset. For our task, we removed the top fully connected layer and added a global average pooling layer, followed by a fully connected layer with softmax activation to output probabilities for the seven emotion classes.

MobileNetV2 is a lightweight and efficient deep learning model designed for mobile and embedded vision applications. It employs depthwise separable convolutions and an inverted residual structure to reduce computation and memory requirements. MobileNetV2 has been pre-trained on the ImageNet dataset. In our experiments, we replaced the top fully connected layers with a global average pooling layer and a fully connected layer with softmax activation for the seven emotion classes.

Xception is a deep learning model that extends the idea of depthwise separable convolutions in MobileNets by incorporating a modified version of the inception module. Xception has been pre-trained on the ImageNet dataset and has shown competitive performance with other state-of-the-art models. For our task, we removed the top fully connected layers and added a global average pooling layer, followed by a fully connected layer with softmax activation to output probabilities for the seven emotion classes.

In our experiments, we fine-tuned these four pre-trained models using the Facial Expression Recognition Challenge dataset. The following sections discuss the training process, hyperparameter tuning, and model evaluation.

3.4. Training the Model

In this study, we experimented with different variations of the VGG16, ResNet50, MobileNetV2, and Xception models to find the best performing model for facial expression recognition. The training process consisted of several steps,

including training with frozen layers, fine-tuning specific layers, and incorporating additional techniques such as data augmentation, batch normalization, and Gaussian noise. We also experimented with preprocessing the data using face detection.

3.4.1 Initial Training with Frozen Layers We began by training the VGG16, ResNet50, MobileNetV2, and Xception models with all layers frozen, meaning the weights in the pre-trained models remained unchanged. We trained only the added fully connected layer to adapt the models to our facial expression recognition task. Among the initial models, MobileNetV2 and VGG16 showed better and similar performance.

3.4.2 Fine-tuning VGG16 and MobileNetV2 Based on the initial results, we decided to further explore VGG16 and MobileNetV2 models by incorporating three major changes:

Data Augmentation: We applied data augmentation techniques, such as random rotations, horizontal flips, and zooming, to increase the diversity of the training data and reduce overfitting.

Batch Normalization: We added batch normalization layers to the models, which helped improve training speed and model performance by normalizing the input to each layer and reducing internal covariate shift.

Unfreezing the Last Six Layers: We unfroze the last six layers of the VGG16 and MobileNetV2 models to allow their weights to be fine-tuned during training. This enabled the models to better adapt to the specific characteristics of our dataset. Since VGG16 had better metrics the further variations are done on this to try and improve the accuracy

3.4.3 Adding Gaussian Noise As a third variation, we introduced Gaussian noise to the input data to improve the model's robustness against small perturbations in the input images. This helped the model generalize better to real-world scenarios where images might contain noise due to varying lighting conditions or other factors.

3.4.4 Face Detection Preprocessing In our fourth variation, we preprocessed the dataset by incorporating face detection. This involved detecting and cropping faces in the images before inputting them into the model. By focusing on the facial region, the model could potentially learn more relevant features for facial expression recognition and improve its overall performance.

Figure 4: *Picture before vs after face detection*

The models were trained and evaluated using the different variations described above.

3.5. Hyperparameter Tuning

Hyperparameters, such as learning rate and early stopping were tuned. The final hyperparameter values were chosen based on their performance on the validation set.

In the following section we discuss the different methods of training and their results

4. Experiments and Results

We conducted several experiments to assess the performance of the different model variations and fine-tuning strategies implemented in our study. These experiments involved training and evaluating the VGG16, ResNet50, MobileNetV2, and Xception models with all layers frozen, fine-tuning specific layers, and incorporating additional techniques such as data augmentation, batch normalization, and Gaussian noise. We also experimented with preprocessing the data using face detection.

4.1 Initial Training and Model Selection

Our initial experiments involved training the four selected models with all layers frozen. In this stage, we trained only the newly added fully connected layer with softmax activation to adapt the models to our facial expression recognition task. After the initial training, we observed that the MobileNetV2 had the best performance. Other models had similar performance. VGG16 performs comparably to these models. VGG16 has a simpler architecture, which might be less prone to overfitting when the amount of training data is limited. This could explain why it doesn't fall far behind the more sophisticated architectures of Xception and ResNet50.

Model	Accuracy	ROC AUC	Avg TPR	Avg FPR	Macro F1
MobileNet_freezed	0.5352	0.8488	0.5351	0.0836	0.5267
VGG16_freezed	0.4784	0.795	0.4786	0.0743	0.4753
Xception_freezed	0.4873	0.8187	0.4871	0.083	0.4686
ResNet50_freezed	0.4823	0.7984	0.4821	0.0853	0.458

Figure 5: *Model performances on different parameters*

Model	Angry-F1	Disgust-F1	Fear-F1	Happy-F1	Sad-F1	Surprise-F1	Neutral-F1
MobileNet_	0.4531	0.4138	0.4131	0.701	0.4627	0.7456	0.4972
VGG16_fre	0.3325	0.5263	0.376	0.6032	0.404	0.669	0.4164
Xception_fr	0.3757	0.3	0.4023	0.6422	0.4247	0.6579	0.4775
ResNet50_	0.3605	0.4576	0.3387	0.5978	0.4371	0.6508	0.4125

Figure 6: *Multiple model f1 labelwise scores*

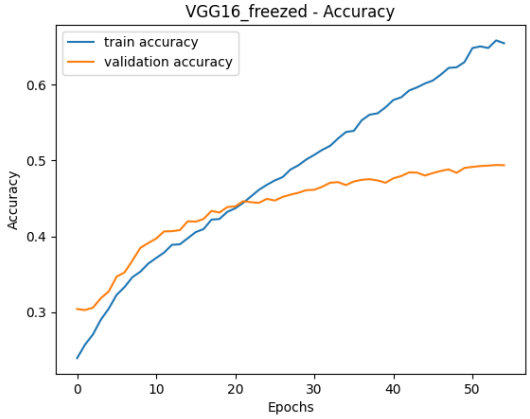

Figure 7: *Accuracy curve VGG*

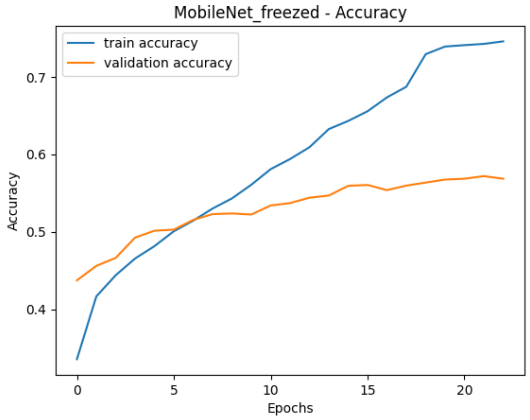

Figure 8: *Accuracy curve MobileNetV2*

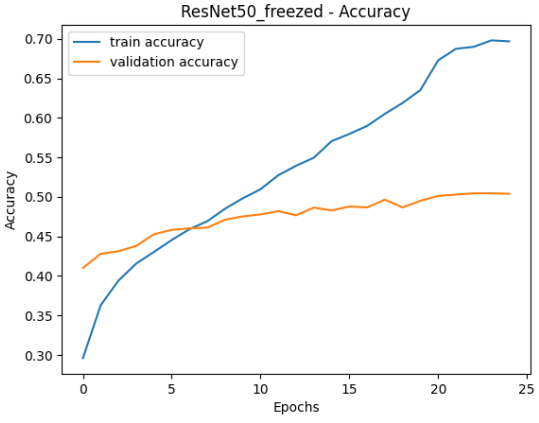

Figure 9: *Accuracy curve ResNet50*

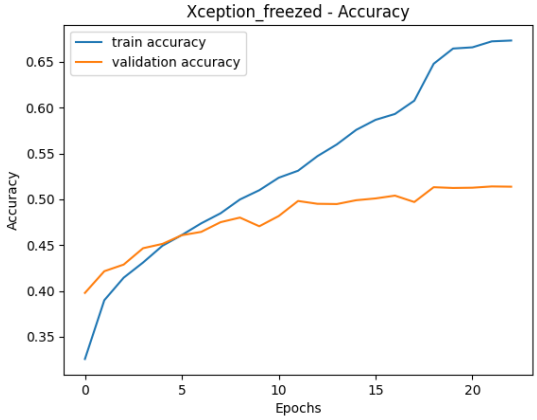

Figure 10: *Accuracy curve Xception*

MobileNetV2 demonstrates superior performance, with higher accuracy, ROC AUC, average TPR, Macro F1, and most individual emotion F1 scores. This could be due to the architecture differences between the two models. MobileNetV2 uses depthwise separable convolutions, which are more efficient and require less computational resources than the standard convolutions used in VGG16. This might enable it to learn more effective features from the data even when the layers are frozen. Furthermore, MobileNetV2 is designed to provide good performance on mobile and embedded devices, indicating its ability to maintain high accuracy even when computational resources are constrained.

We would be choosing MobileNetV2 and VGG16 for further experimentation because of the following reasons:

Performance: Both MobileNetV2 and VGG16 demonstrate competitive performance across various metrics.

Efficiency: MobileNetV2, due to its efficient architecture, can be a good candidate for environments with limited computational resources.

Simplicity: VGG16, with its simpler and more understandable architecture, can be a good model to experiment with, as the changes and their effects can be easier to interpret.

Versatility: VGG16 and MobileNetV2 offer a good mix of performance, efficiency, and simplicity, making them versatile choices for further experimentation.

Benchmarking: MobileNetV2 and VGG16 have been widely used as benchmarks in many image classification tasks. Using these models allows for easier comparison with other works.

Based on these results, we decided to further explore the VGG16 and MobileNetV2 models.

4.2 Fine-tuning VGG16 and MobileNetV2

In the next stage, we fine-tuned the VGG16 and MobileNetV2 models by incorporating three major modifications: data augmentation, batch normalization, and unfreezing the last six layers of the models. The fine-tuned VGG16 model displayed a higher performance metrics compared to MobileNetV2.

Model	Accuracy	ROC AUC	Avg TPR	Avg FPR	Macro F1
MobileNet_modified	0.5458	0.8481	0.4814	0.0826	0.4941
VGG16_modified	0.584	0.8761	0.5976	0.065	0.6032

Figure 11: *Model performances on different parameters*

Model	Angry-F1	Disgust-F1	Fear-F1	Happy-F1	Sad-F1	Surprise-F1	Neutral-F1
MobileNet_modified	0.4142	0.2769	0.4142	0.7273	0.4155	0.7238	0.4871
VGG16_modified	0.5073	0.5769	0.5189	0.7965	0.516	0.7698	0.5367

Figure 12: *Multiple model f1 labelwise scores*

In this experiment, VGG16 performed better than MobileNetV2. The primary reason for this difference can be attributed to the architecture of the two models. VGG16 has a deeper architecture with more layers, which allows it to learn more complex and hierarchical features from the input data. In contrast, MobileNetV2 has a more efficient and compact architecture designed for mobile and edge devices, trading off some accuracy for a reduced number of parameters and computational complexity.

Given the better performance of VGG16 in this experiment, we would be using the VGG16 model with the current setting (data augmentation, unfreezing the last six layers, and adding batch normalization layers) for further experimentation and model development. This configuration should provide a strong foundation for building an accurate and robust emotion recognition system.

4.3 Incorporating Gaussian Noise to modified VGG16

We further enhanced the robustness of our model by adding Gaussian noise to the input data. This modification aimed to increase the model's tolerance to minor perturbations in the input images, thereby improving its generalization ability to real-world conditions where images may contain noise due to various factors.

Model	Accuracy	ROC AUC	Avg TPR	Avg FPR	Macro F1
VGG16_mod_noise	0.5979	0.8733	0.5143	0.07	0.5232
VGG16_modified	0.584	0.8761	0.5357	0.0657	0.6032

Figure 13: *Model performances on different parameters*

Model	Angry-F1	Disgust-F1	Fear-F1	Happy-F1	Sad-F1	Surprise-F1	Neutral-F1
VGG16_mod_noise	0.5127	0.1333	0.4302	0.7998	0.5164	0.7266	0.5435
VGG16_modified	0.5073	0.5769	0.5189	0.7965	0.516	0.7698	0.5367

Figure 14: *Multiple model f1 labelwise scores*

Both VGG16_mod_noise and VGG16_modified were trained under similar conditions, the primary difference being the inclusion of a Gaussian Noise layer in VGG16_mod_noise. The Gaussian Noise layer is used to add noise to the inputs with a mean of 0 and some standard deviation. This noise can act as a form of regularization, making the model more robust and preventing overfitting by forcing the model to learn the same concept, even though the input data is changed slightly due to the noise.

However, the addition of a Gaussian Noise layer in VGG16_mod_noise doesn't seem to have improved the model's performance significantly, as it achieved an accuracy of 0.5979, compared to 0.5840 for VGG16_modified. The difference in accuracy is not considerable, suggesting that the noise layer didn't contribute substantially to the performance in this case.

In terms of class-specific F1-scores, there is a noticeable improvement in the 'Disgust' class for the VGG16_modified model, which can be a result of the model being able to learn more discriminative features for this class without the noise layer. However, the 'Happy' class performed slightly better with the noise layer. This may be due to the noise providing additional robustness for this class.

The average TPR is higher for VGG16_modified, suggesting that the model without the noise layer was able to recall the positive class better. A possible explanation for this might be that the noise layer, while adding robustness, could also make it harder for the model to correctly identify the positive class in some cases, leading to a lower TPR.

Choosing the VGG16_modified model (without Gaussian noise layer) for further experimentation because:
As the performance difference in terms of accuracy is not significant, it might be beneficial to choose the simpler model. This aligns with the principle of Occam's razor in model selection, which states that given two models with similar performance, the simpler one should be chosen.

When comparing the F1 scores, we see that the model without the noise layer (VGG16_modified) has better overall Macro F1 score (0.6031) than the model with the noise layer (VGG16_mod_noise, Macro F1: 0.5232). This suggests that the VGG16_modified model has a more balanced performance across classes, not just favouring the majority classes.

4.4 Preprocessing with Face Detection

In our final set of experiments, we preprocessed the dataset by incorporating face detection. This modification involved detecting and cropping faces in the images before feeding them into the model. Despite its theoretical advantages, this variation did not lead to a significant improvement in the model's performance.

Model	Accuracy	ROC AUC	Avg TPR	Avg FPR	Macro F1
VGG16_mod_face	0.5525	0.8769	0.6029	0.0643	0.6131
VGG16_modified	0.584	0.8761	0.6021	0.0643	0.6031

Figure 15: *Model performances on different parameters*

Model	Angry-F1	Disgust-F1	Fear-F1	Happy-F1	Sad-F1	Surprise-F1	Neutral-F1
VGG16_mod_face	0.5152	0.6813	0.4882	0.8027	0.5086	0.7606	0.5354
VGG16_modified	0.5073	0.5769	0.5189	0.7965	0.516	0.7698	0.5367

Figure 16: *Multiple model f1 labelwise scores*

We evaluated the performance of two variations of the VGG16 model: one with a face detection and cropping mechanism (VGG16_mod_face) and the other without this mechanism (VGG16_modified)

Overall, the VGG16_modified model without face

detection demonstrated superior performance, achieving an accuracy of 58.4%, compared to 55.25% for the VGG16_mod_face model. Interestingly, the ROC AUC scores for both models were very similar, indicating that both models were comparable in distinguishing between the different emotion classes.

The average TPR and FPR were also identical for both models, suggesting that the models had similar performance in terms of sensitivity and specificity. However, the VGG16_modified model had a slightly higher Macro F1-score and individual F1-scores for most of the emotion classes, indicating that it had better overall performance in terms of precision and recall.

The results suggest that the face detection and cropping mechanism in VGG16_mod_face did not significantly improve the model's performance. One possible reason could be that the dataset used in this study was already cropped around the face region, making additional face detection redundant. Another reason could be that the face detection mechanism might have inadvertently removed useful contextual information from the images. For instance, the position of the face within the image or the presence of other objects could potentially provide cues about the person's emotional state.

We decided to select the VGG16_modified model without face detection as the final model. The reasons for this decision are as follows:

Better Overall Performance: The VGG16_modified model showed superior performance across several metrics, including accuracy and Macro F1-score. This indicates that the model was better at both predicting the correct emotion class and balancing precision and recall across the classes.

Simplicity: The VGG16_modified model was simpler and more efficient, as it did not require the additional step of face detection and cropping. This could make the model easier to implement and more efficient to run in real-world applications.

Robustness to Different Image Types: The VGG16_modified model might be more robust to images where the face is not centered or where there are multiple faces, as it does not rely on face detection.

4.5 Final Model and Performance

From our previous experiments and analysis we selected the model VGG16_modified. The changes from the basic VGG16 made are:

- Unfreezing the last 6 layers
- Batch Normalization
- Data Augmentation

Results for VGG16_modified are given below

Accuracy: 0.584006667137146

F1 scores: {
'Angry': 0.5072765072765072,
'Disgust': 0.576923076923077,
'Fear': 0.5188866799204771,
'Happy': 0.7965317919075143,
'Sad': 0.5159817351598174,
'Surprise': 0.7697929354445797,
'Neutral': 0.5366639806607573
}

Precision scores: {
'Angry': 0.49292929292929294,
'Disgust': 0.625,
'Fear': 0.5117647058823529,
'Happy': 0.8251497005988024,
'Sad': 0.5128593040847201,
'Surprise': 0.7783251231527094,
'Neutral': 0.5252365930599369
}

Recall scores: {
'Angry': 0.5224839400428265,
'Disgust': 0.5357142857142857,
'Fear': 0.5262096774193549,
'Happy': 0.7698324022346369,
'Sad': 0.5191424196018377,
'Surprise': 0.7614457831325301,
'Neutral': 0.5485996705107083
}

True Positive Rates: {
Angry: 0.52
Disgust: 0.54
Fear: 0.53
Happy: 0.77
Sad: 0.52
Surprise: 0.76
Neutral: 0.55

}

False Positive Rates: {
Angry: 0.08
Disgust: 0.01
Fear: 0.08
Happy: 0.05
Sad: 0.11
Surprise: 0.03
Neutral: 0.10

}

Confusion Matrix:

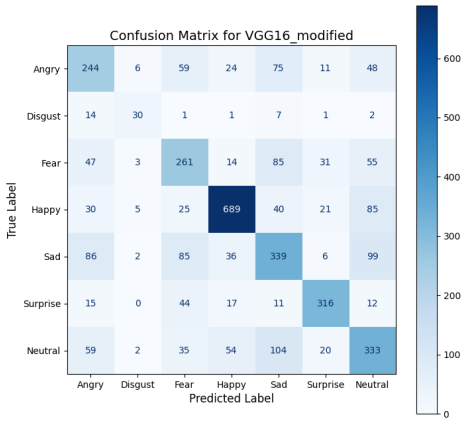

Figure 17: *VGG16modified Confusion matrix*

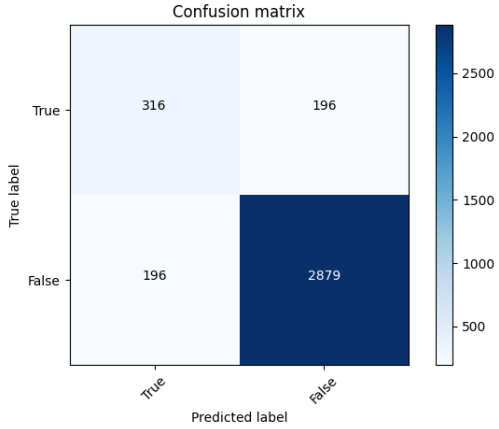

Figure 18: *VGG16modified Average Confusion matrix*

Classification Report: {

'Angry': {
'precision': 0.49292929292929294,
'recall': 0.5224839400428265,
'f1-score': 0.5072765072765072,
'support': 467
},
'Disgust': {
'precision': 0.625,
'recall': 0.5357142857142857,
'f1-score': 0.576923076923077,
'support': 56
},
'Fear': {
'precision': 0.5117647058823529,
'recall': 0.5262096774193549,
'f1-score': 0.5188866799204771,
'support': 496
},
'Happy': {
'precision': 0.8251497005988024,
'recall': 0.7698324022346369,
'f1-score': 0.7965317919075143,
'support': 895
},
'Sad': {
'precision': 0.5128593040847201,
'recall': 0.5191424196018377,
'f1-score': 0.5159817351598174,
'support': 653
},
'Surprise': {
'precision': 0.7783251231527094,
'recall': 0.7614457831325301,
'f1-score': 0.7697929354445797,
'support': 415
},
'Neutral': {
'precision': 0.5252365930599369,
'recall': 0.5485996705107083,
'f1-score': 0.5366639806607573,
'support': 607
},

'accuracy': 0.6163276678740597,

'macro avg': {
'precision': 0.6101806742439735,
'recall': 0.5976325969508828,
'f1-score': 0.6031509581846758,
'support': 3589
},

'weighted avg': {
'precision': 0.6225307661227734,
'recall': 0.6163276678740597,
'f1-score': 0.6190094025448716,
'support': 3589
}
}

ROC AUC: 0.8760710426056891

Loss Curve

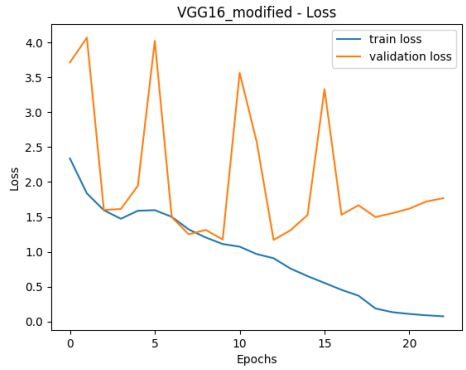

Figure 19: *VGG16modified Loss Curve*

Accuracy Curve

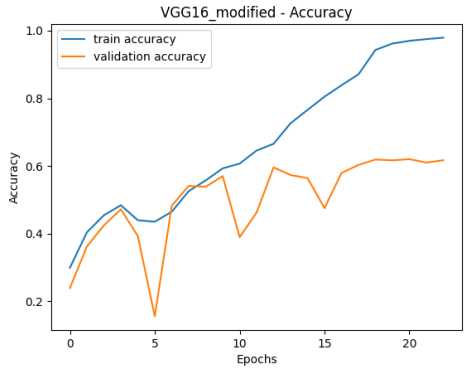

Figure 20: *VGG16modified Accuracy Curve*

Predictions

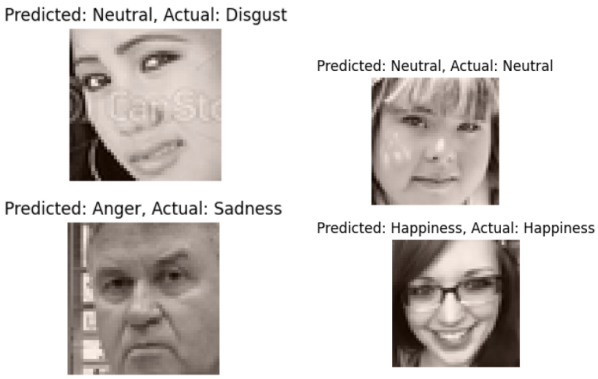

Figure 21: *VGG16-Modified Predictions*

5. Conclusion and Future Work

In this project, we applied deep learning techniques for facial expression recognition, leveraging the power of convolutional neural networks and the versatility of transfer learning. We experimented with several pre-trained models including VGG16, ResNet50, MobileNetV2, and Xception, as well as various fine-tuning strategies and additional techniques such as data augmentation, batch normalization, and Gaussian noise.

Our results indicate that the VGG16 model, after fine-tuning and without the use of face detection preprocessing, achieved the best performance. This model demonstrated superior results in terms of accuracy, Macro F1-score, and individual F1-scores for most of the emotion classes.

While the addition of a face detection and cropping mechanism did not significantly improve the performance of our model, it did highlight the importance of considering the context in which the model will be deployed. Our results suggest that for datasets where faces are already well-centered and isolated, such a mechanism might not provide additional benefits and could even remove potentially useful contextual information.

Moving forward, there are several potential avenues for improving and expanding upon this work:

Explore Different Model Architectures: While we experimented with several popular pre-trained models, there are many other architectures that could potentially yield better performance. Future work could explore models such as EfficientNet, DenseNet, or NASNet.

Incorporate Additional Preprocessing Techniques: We could explore other image preprocessing techniques like image enhancement methods (contrast adjustment, histogram equalization) or other data augmentation techniques to possibly improve the model's performance.

Multi-modal Emotion Recognition: To capture a more comprehensive picture of a person's emotional state, we could extend this work to incorporate other modalities, such as audio or text, in a multi-modal emotion recognition system.

Real-Time Emotion Recognition: Our work could be extended to develop a real-time emotion recognition system that could be used in various applications such as interactive games, mental health monitoring, or customer satisfaction analysis.

Fairness and Bias Analysis: Lastly, a crucial aspect to explore in future work is the fairness and bias of the models. It would be important to ensure that the models perform equally well across different demographic groups and do not exhibit any unfair biases.

In conclusion, this project represents a promising step towards effective emotion recognition using deep learning, and we look forward to seeing how these techniques continue to evolve and improve in the future.

6. References

- https://saturncloud.io/blog/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way/
- https://dspace.library.uvic.ca/bitstream/handle/1828/13388/Adil_Mohammed_Adnan_MEng_2021.pdf
- https://github.com/ryangawei/CNN-Facial-Expression-Recognition