Name: Gaurav Soni

Project Title: Facial Emotion Classification

Data Description

Dataset Source:

- Name of the Dataset: FER-2013
- Source: Kaggle Link to Dataset FER-2013

Data Summary:

The FER-2013 dataset consists of 48x48 pixel grayscale images of faces. The images have been automatically registered so that each face is centered and occupies a similar amount of space in the image. The dataset is designed for facial expression recognition tasks.

Each image in the dataset represents a face showing one of seven different emotions. The dataset is divided into training and testing directories, with subdirectories for each emotion category: Angry, Disgust, Fear, Happy, Sad, Surprise, Neutral.

The dataset includes separate directories for training and testing data Each directory contains subdirectories named after the emotion categories, which hold the corresponding images.

Sr. no.	Emotion	Label	No. instances in training	No. instances in Testing
1.	Fear	0	4097	1024
2.	Disgust	1	436	111
3.	Sad	2	4830	1247
4.	Neutral	3	4965	1233
5.	Surprise	4	3171	831
6.	Happy	5	7215	1774
7.	Angry	6	3995	958

The dataset provides a well-balanced distribution of images across different emotions, though some categories like "Disgust" have fewer samples compared to others like "Happy". This balance ensures that the model can learn to recognize a variety of expressions, but additional techniques like data augmentation may be necessary for categories with fewer samples to improve model performance.

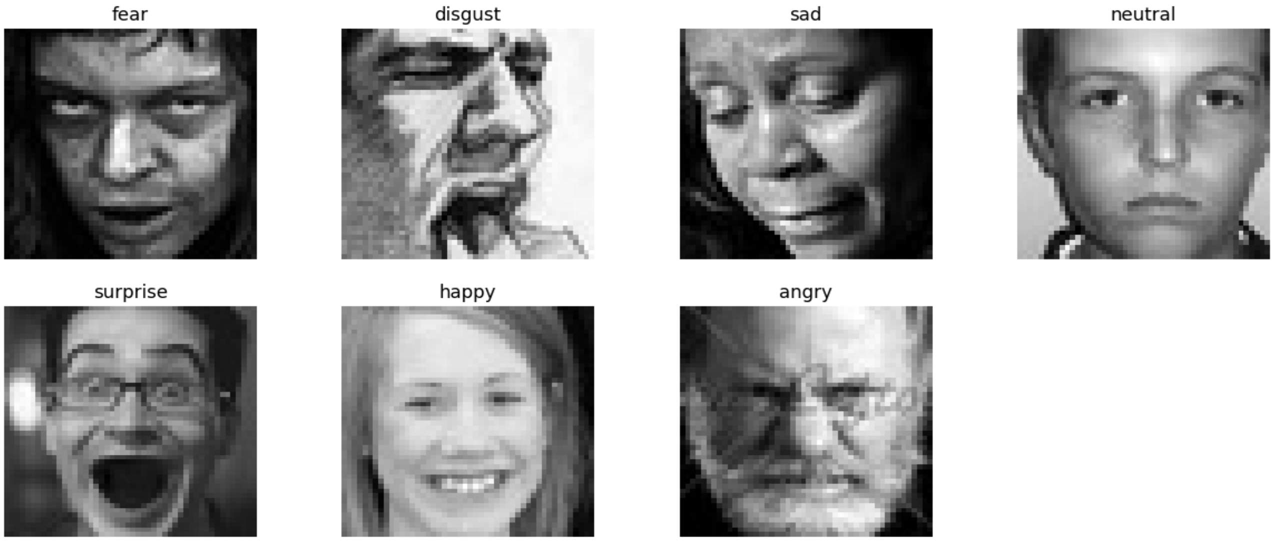


Figure 1 data Sample

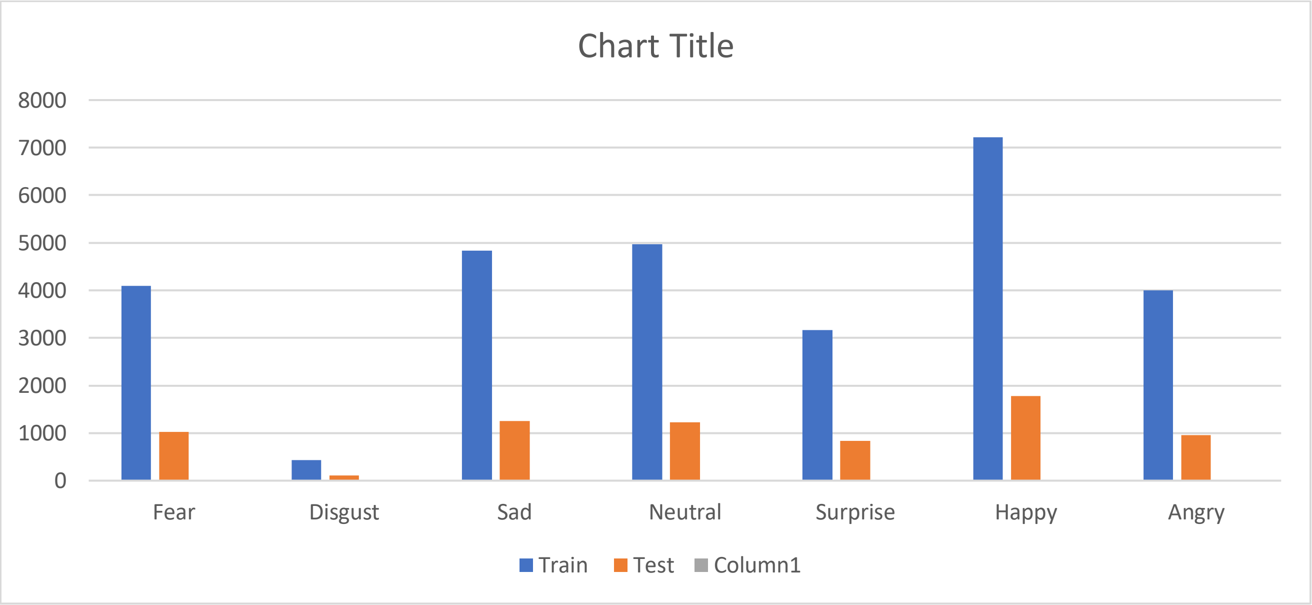


Figure 2 Data Distribution

Conclusion:

The FER-2013 dataset is a comprehensive collection of facial images categorized by emotional expression, providing a robust foundation for training and testing an image classifier. The structured directories and well-defined categories facilitate efficient data loading and preprocessing for deep learning tasks.

Data Cleaning:

To ensure the dataset is clean and uniform, I performed a data cleaning step where only images with specific file extensions were retained. This step involved filtering the dataset to include only images with the following extensions: ['jpeg', 'jpg', 'png']. This was done to eliminate any unsupported or corrupted files and ensure consistency across the dataset for better training performance.

Objective:

The objective of this project is to classify images based on the emotions they depict using various Convolutional Neural Network (CNN) models. The goal is to evaluate and compare the performance of different CNN architectures to identify the most effective model for accurate emotion recognition. This involves training each model on the FER-2013 dataset and assessing their classification accuracy and other relevant metrics.

Models

Model 1: Custom CNN:

Model Parameters:

The Custom CNN model was designed with specific parameters to classify facial expressions into seven distinct emotion categories. The input images were 48x48 pixels in grayscale format, with a batch size of 64 and a total of 10 training epochs. The dataset was split into training and validation sets, with 20% of the training data reserved for validation purposes.

Model 1 and model has same Architecture but model 2 is with image augmentation. The Architecture is as follow:

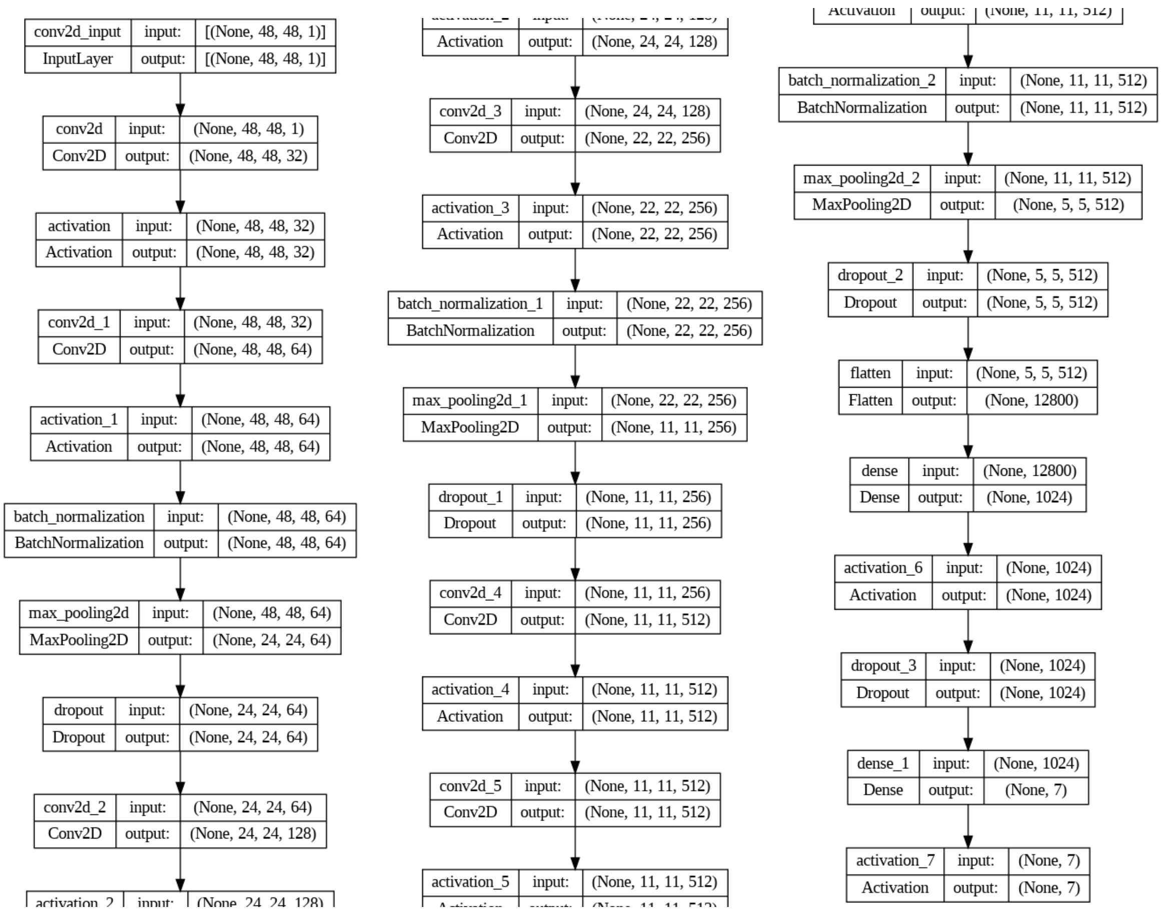

conv2d_input | input: | [(None, 48, 48, 1)]
InputLayer | output: | [(None, 48, 48, 1)]
conv2d | input: | (None, 48, 48, 1)
Conv2D | output: | (None, 48, 48, 32)
activation | input: | (None, 48, 48, 32)
Activation | output: | (None, 48, 48, 32)
conv2d_1 | input: | (None, 48, 48, 32)
Conv2D | output: | (None, 48, 48, 64)
activation_1 | input: | (None, 48, 48, 64)
Activation | output: | (None, 48, 48, 64)
batch_normalization | input: | (None, 48, 48, 64)
BatchNormalization | output: | (None, 48, 48, 64)
max_pooling2d | input: | (None, 48, 48, 64)
MaxPooling2D | output: | (None, 24, 24, 64)
dropout | input: | (None, 24, 24, 64)
Dropout | output: | (None, 24, 24, 64)
conv2d_2 | input: | (None, 24, 24, 64)
Conv2D | output: | (None, 24, 24, 128)
activation_2 | input: | (None, 24, 24, 128)
Activation | output: | (None, 24, 24, 128)
conv2d_3 | input: | (None, 24, 24, 128)
Conv2D | output: | (None, 22, 22, 256)
activation_3 | input: | (None, 22, 22, 256)
Activation | output: | (None, 22, 22, 256)
batch_normalization_1 | input: | (None, 22, 22, 256)
BatchNormalization | output: | (None, 22, 22, 256)
max_pooling2d_1 | input: | (None, 22, 22, 256)
MaxPooling2D | output: | (None, 11, 11, 256)
dropout_1 | input: | (None, 11, 11, 256)
Dropout | output: | (None, 11, 11, 256)
conv2d_4 | input: | (None, 11, 11, 256)
Conv2D | output: | (None, 11, 11, 512)
activation_4 | input: | (None, 11, 11, 512)
Activation | output: | (None, 11, 11, 512)
conv2d_5 | input: | (None, 11, 11, 512)
Conv2D | output: | (None, 11, 11, 512)
activation_5 | input: | (None, 11, 11, 512)
Activation | output: | (None, 11, 11, 512)
batch_normalization_2 | input: | (None, 11, 11, 512)
BatchNormalization | output: | (None, 11, 11, 512)
max_pooling2d_2 | input: | (None, 11, 11, 512)
MaxPooling2D | output: | (None, 5, 5, 512)
dropout_2 | input: | (None, 5, 5, 512)
Dropout | output: | (None, 5, 5, 512)
flatten | input: | (None, 5, 5, 512)
Flatten | output: | (None, 12800)
dense | input: | (None, 12800)
Dense | output: | (None, 1024)
activation_6 | input: | (None, 1024)
Activation | output: | (None, 1024)
dropout_3 | input: | (None, 1024)
Dropout | output: | (None, 1024)
dense_1 | input: | (None, 1024)
Dense | output: | (None, 7)
activation_7 | input: | (None, 7)
Activation | output: | (None, 7)

Figure 3 model 1 and model 2 caption

Data Generators:

data generators were employed to efficiently handle the large dataset and ensure smooth training and validation processes. The use of data generators helps in dynamically loading and preprocessing the data in batches, which is crucial for handling large datasets that cannot fit entirely into memory.

Training Generator: The training generator was responsible for loading and preprocessing the training images. It identified a total of 22,968 images across the seven emotion classes. The generator applied real-time data augmentation techniques such as rotation, scaling, and horizontal flipping to enhance the diversity of the training data. This augmentation helps improve the model's generalization ability by exposing it to a wider variety of image variations.

Validation Generator: The validation generator loaded and preprocessed the validation images, identifying 5,741 images belonging to the seven classes. This generator did not apply data augmentation, ensuring that the validation performance reflected the model's true ability to generalize to unseen data. The validation generator provided a consistent evaluation metric for monitoring the model's performance during training.

Test Generator: The test generator was used to load and preprocess the test images, identifying 7,178 images across the seven emotion classes. Similar to the validation generator, the test generator did not apply data augmentation. The test data was used to evaluate the final model performance after training was completed, providing an unbiased assessment of the model's accuracy and generalization capability.

Overall, the data generators played a crucial role in managing the dataset efficiently, applying necessary preprocessing steps, and ensuring the model was trained and evaluated on appropriately augmented and preprocessed data. Total Trainable Parameters are 17,044,871.

To enhance training efficiency and model performance, several callbacks were implemented. The model was saved during training based on its validation performance, ensuring the best model was retained. Early stopping was used to prevent overfitting by monitoring training progress and halting once performance plateaued. Additionally, the ReduceLROnPlateau callback adjusted the learning rate dynamically when the validation loss stopped improving, with a reduction factor of 0.2 and a patience of 6 epochs. This comprehensive training strategy ensured that the Custom CNN model was trained effectively, balancing performance and efficiency.

Training Description:

The Custom CNN model was trained using a carefully structured process. The training data was split into training and validation sets with an 80-20 split. The model's performance was continually monitored using callbacks, ensuring that training was efficient and prevented overfitting. By utilizing `ReduceLROnPlateau`, the model adjusted its learning rate dynamically to maintain steady improvement in validation loss.

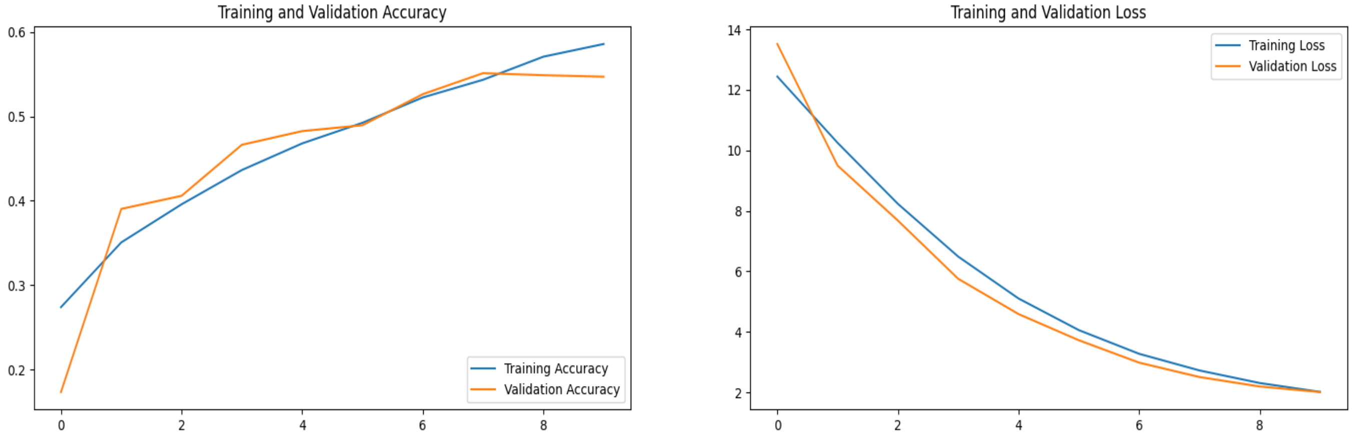


Figure 4 train & Validation Accuracy/Loss for model 1

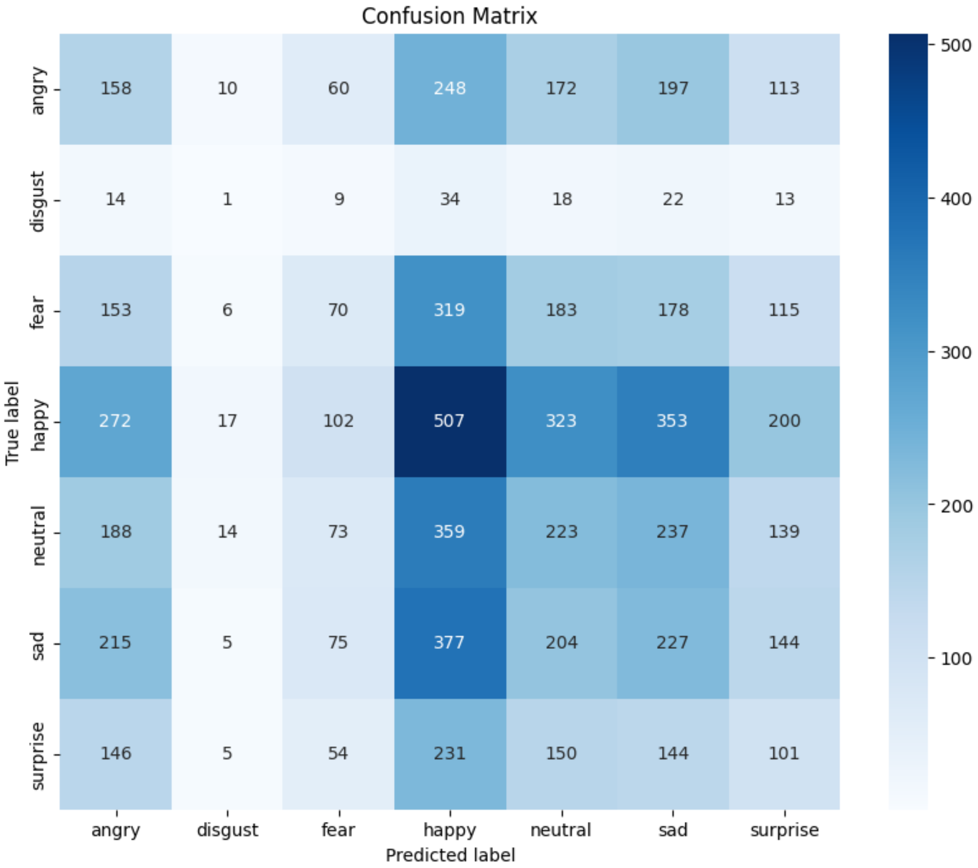


Figure 5 Confusion matrix of model 1

The final training accuracy of the Custom CNN model was **65.58%,** with a validation accuracy of **56.71%.**

Model 2: Custom CNN with Image Augmentation

The **Custom CNN with Image Augmentation** model was configured with specific parameters tailored to classify facial expressions into seven emotion categories. The input images were standardized to 48x48 pixels in grayscale format to maintain consistency. The model was trained with a batch size of 64, ensuring efficient use of computational resources. The training process spanned 100 epochs, providing ample opportunity for the model to learn and fine-tune its parameters. The classification task was aimed at identifying one of seven distinct emotion categories, reflecting the seven classes present in the dataset.

Image Data Generator:

To enhance the diversity of the training data and improve the model's robustness, various image augmentation techniques were applied. The pixel values were rescaled from [0, 255] to [0, 1], and random rotations within a 40-degree range were introduced. The data augmentation also included random horizontal shifts up to 20% of the total width and random vertical shifts up to 20% of the total height. Additionally, random shearing intensity, random zoom within a 20% range, and random horizontal flips were incorporated. Any new pixels created from these transformations were filled using the 'nearest' strategy. A validation split of 20% was used to evaluate the model's performance. The training generator applied these augmentation techniques to the dataset, while the validation generator used the original images without augmentation. Similar to Model 1, this model also comprised a substantial number of trainable parameters, ensuring it had sufficient capacity to learn complex patterns.

Callbacks:

Several callbacks were implemented to enhance training efficiency and model performance. The best model was saved based on its validation performance, ensuring that the optimal version was retained. Early stopping was used to monitor training progress and halt if performance plateaued, preventing overfitting. Additionally, the `ReduceLROnPlateau` callback was employed to dynamically adjust the learning rate. This callback monitored the validation loss and reduced the learning rate by a factor of 0.2 if no improvement was observed for 6 epochs. The settings for this callback included a verbosity level for detailed output and a minimum delta of 0.0001 to qualify as an improvement.

Training Description

The Custom CNN with Image Augmentation model was trained using an augmented dataset to increase the diversity of training examples. This helped the model generalize better to unseen data. The augmentation techniques provided varied perspectives of the same images, making the model more robust to different image variations.

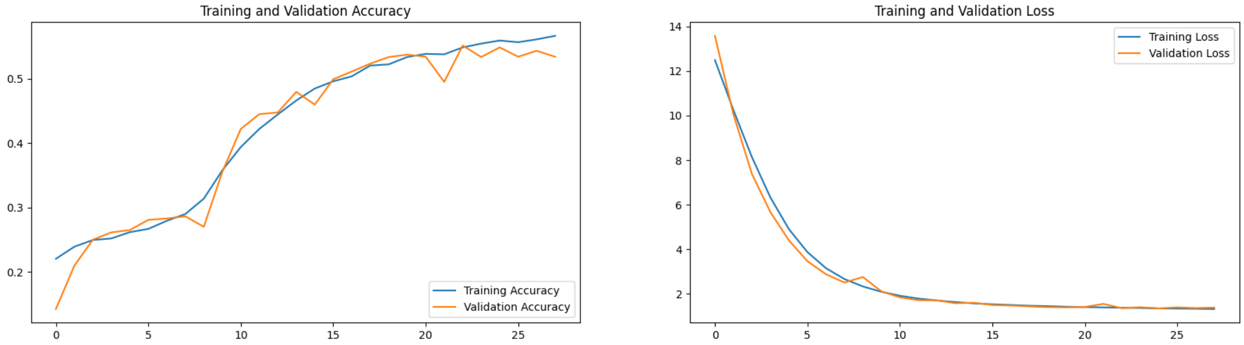
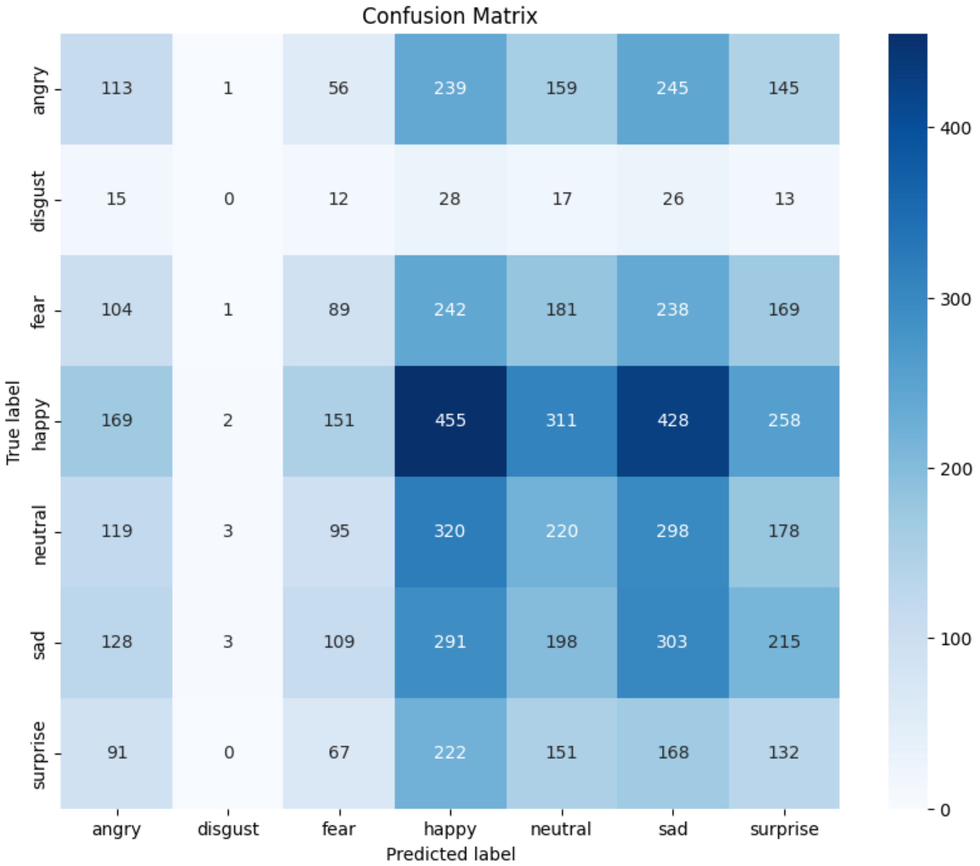


Figure 6 train & Validation Accuracy/Loss for model 2



True label \ Predicted label	angry	disgust	fear	happy	neutral	sad	surprise
angry	113	1	56	239	159	245	145
disgust	15	0	12	28	17	26	13
fear	104	1	89	242	181	238	169
happy	169	2	151	455	311	428	258
neutral	119	3	95	320	220	298	178
sad	128	3	109	291	198	303	215
surprise	91	0	67	222	151	168	132

Figure 7 Confusion matrix of model 2

Result:

The final training accuracy of this model was **57.08%,** while the validation accuracy was **59.14%.** These results suggest that while image augmentation improved the model's validation performance compared to its training performance, further optimization and experimentation with different CNN architectures and hyperparameters could potentially enhance the results even more.

Model 3: Transfer Learning VGGNET

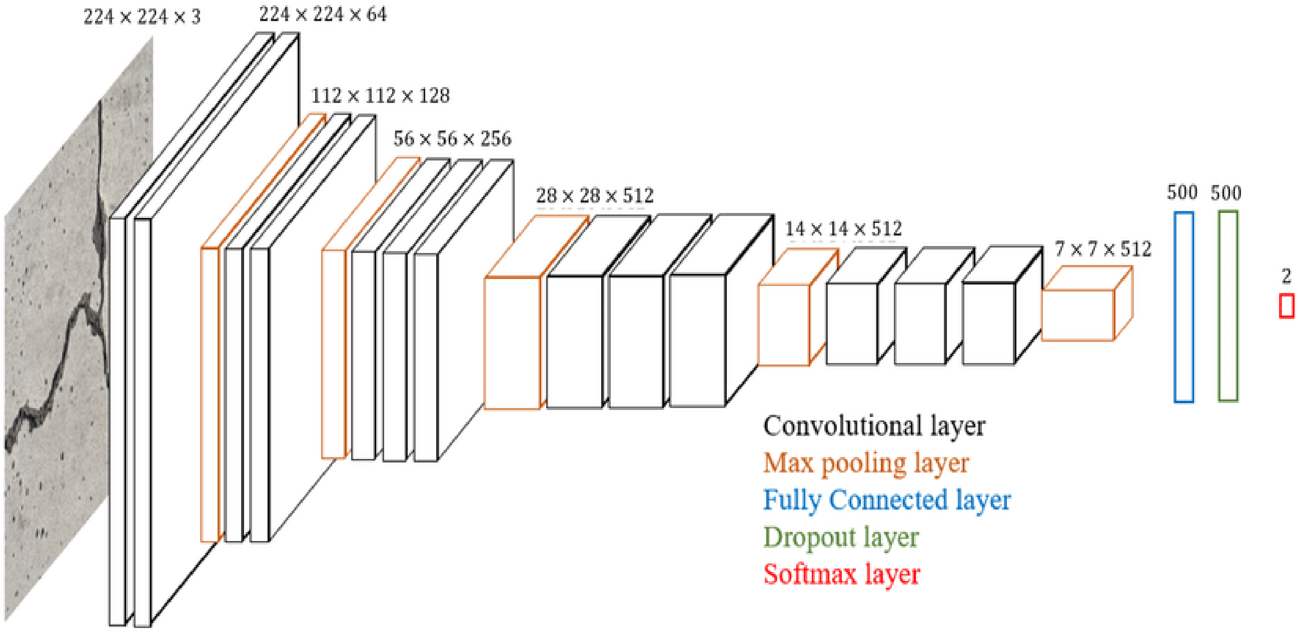


Figure 8 VGG16 architecture

Model Parameters:

The Transfer Learning VGGNET model utilized a pre-trained VGGNET architecture, with specific parameters designed for classifying facial expressions into seven emotion categories. The input images were resized to 224x224 pixels in RGB format to match the requirements of the VGGNET model. The model was trained with a batch size of 64 over 100 epochs. This approach leveraged the robust feature extraction capabilities of the pre-trained VGGNET model, fine-tuning it specifically for the task of emotion classification.

Data Augmentation:

To enhance the training dataset's diversity and robustness, several image augmentation techniques were employed. Pixel values were rescaled from [0, 255] to [0, 1] to standardize the input images. Random rotations within a 10-degree range were applied to simulate different viewing angles. A zoom range of 20% introduced variations in the scale of the images. Horizontal shifts up to 10% of the total width and vertical shifts up to 10% of the total height were used to create spatial transformations. Horizontal flips were randomly applied to create mirror images, enhancing the model's ability to generalize. Additionally, new pixels created from these transformations were filled using the 'nearest' strategy to maintain image integrity. These augmentation techniques collectively contributed to creating a more varied and robust training dataset.

Data Generators:

Data generators played a crucial role in efficiently managing the dataset for the Transfer Learning VGGNET model. The training generator applied various augmentation techniques, such as rescaling, random rotations, zoom, shifts, and flips, to enhance the diversity and robustness of the training dataset. This process resulted in the identification and processing of 28,709 images across the seven emotion classes. Similarly, the test generator was responsible for loading and preprocessing the test images, which involved rescaling the pixel values. This generator handled 7,178 images, ensuring that the model's performance could be accurately evaluated on a standardized test set. The use of these data generators ensured that the training and testing processes were both efficient and effective.

The Transfer Learning VGGNET model inherited the complexity of the VGGNET architecture. Only the last three dense layers were trained, while the rest of the pre-trained layers were frozen. This allowed the model to benefit from the pre-trained feature extraction while adapting the final classification layers to the specific emotion classification task.

Callbacks:

To optimize training efficiency and model performance, several callbacks were implemented. The `ModelCheckpoint` callback was used to save the best model based on its validation performance, ensuring that the optimal model weights were preserved. Early stopping was employed to monitor the validation loss and halt training if the performance plateaued, restoring the best weights to prevent overfitting. The `ReduceLROnPlateau` callback was also utilized to reduce the learning rate when the validation loss showed no improvement for six epochs. This reduction was achieved by a factor of 0.2, with a minimum delta of 0.0001 to qualify as an improvement, and verbose output enabled for monitoring progress. These callbacks collectively enhanced the model's training process, contributing to more efficient and effective learning.

Training Description:

The Transfer Learning VGGNET model was trained using the augmented dataset to increase the diversity of training examples, which helped the model generalize better to unseen data. The training process involved 28,709 images.

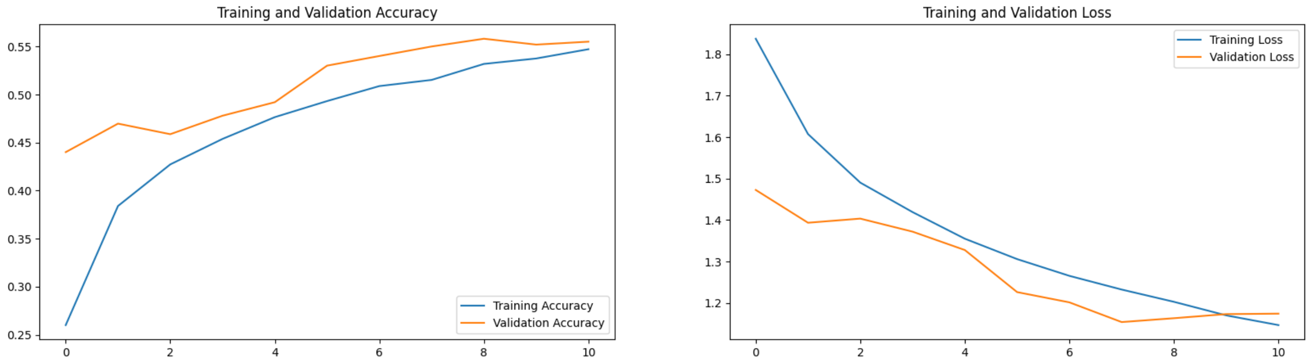


Figure 9 train & Validation Accuracy/Loss for model 3

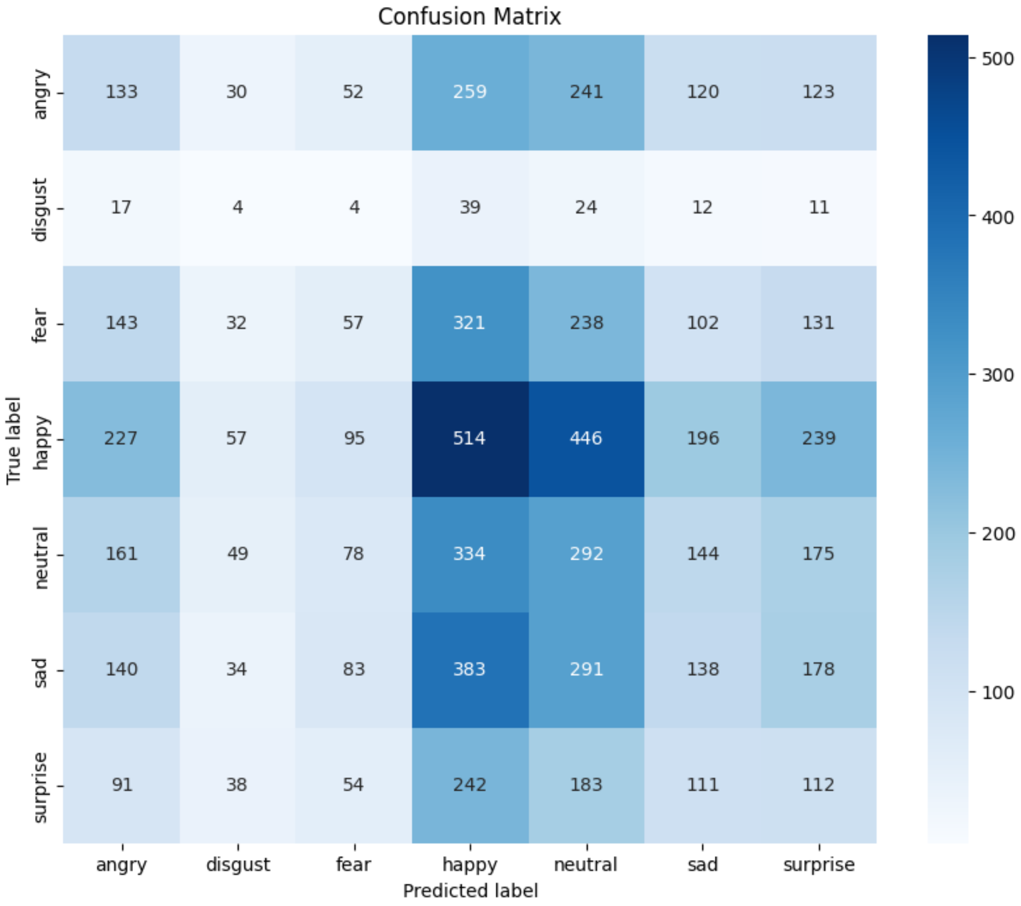


Figure 10 Confusion matrix of model 3

Result:

the final training accuracy of this model was 55.93% while the validation accuracy was 55%. The use of transfer learning, combined with data augmentation and a robust training strategy, significantly enhanced the model's performance in classifying facial expressions into the seven emotion categories. Further fine-tuning and experimentation with different hyperparameters and CNN architectures could potentially yield even better results.

Model 4: Transfer Learning - ResNet50

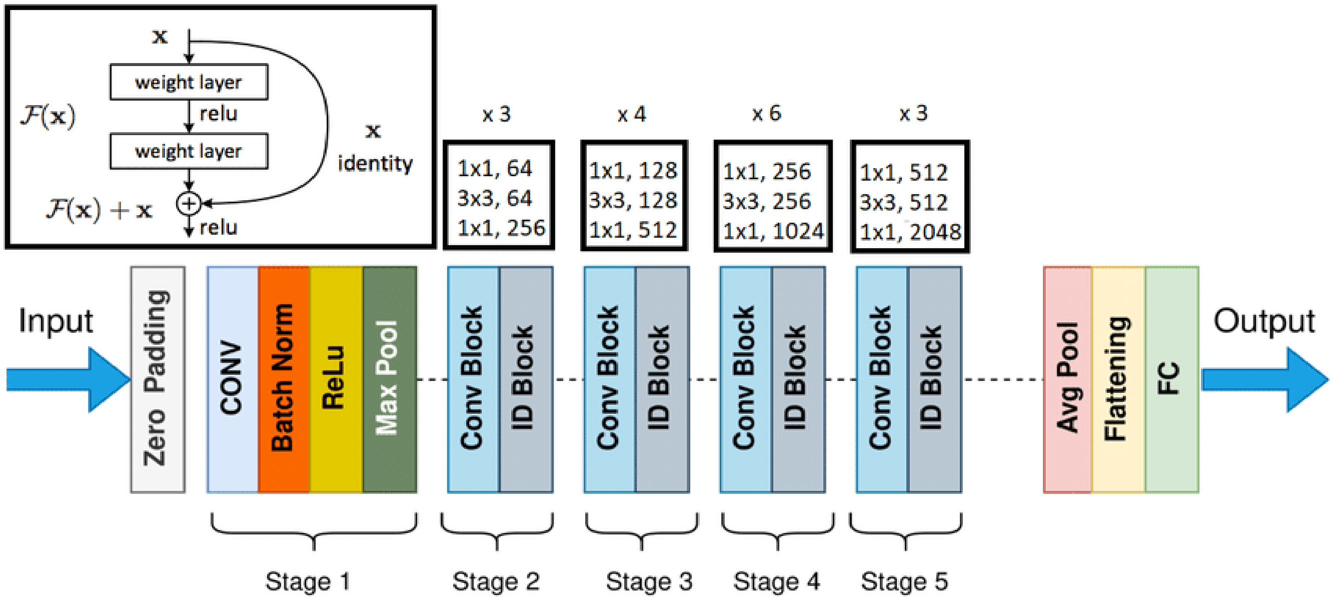


Figure 11 Resnet 50 Architecture

Image Augmentation:

To enhance the diversity and robustness of the training dataset, several image augmentation techniques were utilized.

Pixel values were rescaled from the original range of [0, 255] to [0, 1] to standardize the input data. Random rotations within a 10-degree range were applied to introduce variability in the orientation of the images. Zoom adjustments were made within a 20% range to simulate different levels of magnification. Additionally, random horizontal and vertical shifts were performed, with shifts up to 10% of the total width and height, respectively, to account for variations in image positioning. Horizontal flips were also included to further diversify the training examples. For handling newly created pixels from these transformations, the 'nearest' fill mode was employed, ensuring that these pixels were filled appropriately based on their nearest neighbors.

Data Generators:

Data generators played a crucial role in managing the dataset:

- **Training Generator:** Applied the aforementioned augmentation techniques to enhance the training dataset, processing 28,709 images across seven emotion classes. This generator rescaled pixel values to [0, 1], resized images to (224, 224), used the RGB color mode, shuffled the data, and processed it in batches.
- **Test Generator:** Used for loading and preprocessing the test images, handling 7,178 images across seven emotion classes. This generator only rescaled pixel values to [0, 1], resized images to (224, 224), used the RGB color mode, and did not shuffle the data.
-

Class Weights

To address class imbalance, class weights were introduced:

- **Class Weights Dictionary:**
    - Class 0: 1.0266
    - Class 1: 9.4066
    - Class 2: 1.0010
    - Class 3: 0.5684
    - Class 4: 0.8260
    - Class 5: 0.8491
    - Class 6: 1.2934

Callbacks:

Several callbacks were implemented to enhance training efficiency and model performance:

- **Model Saving:** The best model was saved based on validation performance using the `ModelCheckpoint` callback.
- **Early Stopping:** Monitored validation loss and halted training if performance plateaued, restoring the best weights.
- **Learning Rate Reduction:** The `ReduceLROnPlateau` callback was used with the following settings:
    - **Monitor:** Validation loss
    - **Factor:** 0.2 (reduces learning rate by a factor of 0.2)
    - **Patience:** 6 epochs (no improvement in validation loss for 6 epochs)
    - **Verbosity:** Verbose output
    - **Minimum Delta:** 0.0001 (minimum change to qualify as an improvement)

Training Description

- The Transfer Learning ResNet50 model was trained using the augmented dataset to increase the diversity of training examples, helping the model generalize better to unseen data. The training involved processing 28,709 images,

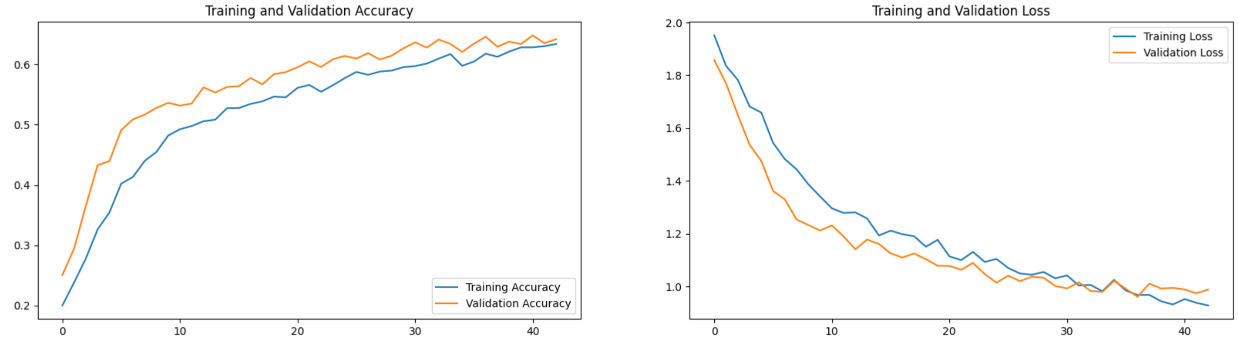


Figure 12 train & Validation Accuracy/Loss for model 4

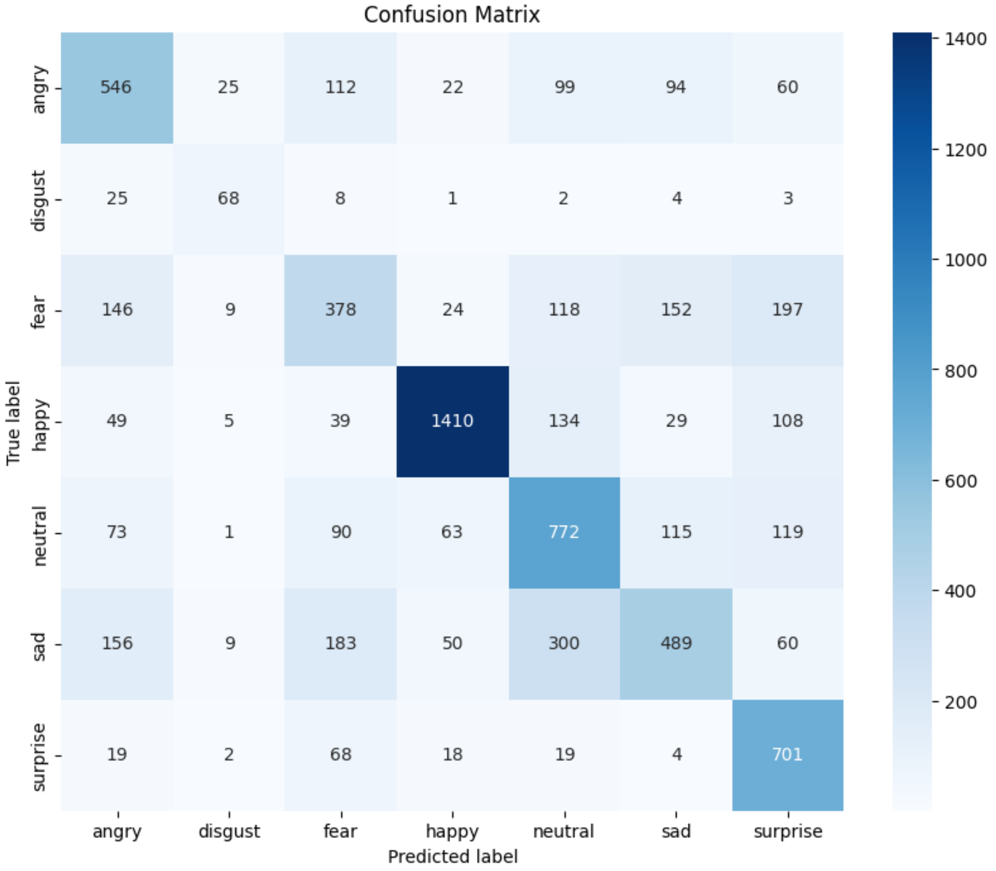


Figure 13 Confusion matrix of model 4

Result:

achieving a final training accuracy of 62.61% and a validation accuracy of 60.80%. The application of transfer learning, along with data augmentation, class weights, and robust training strategies, contributed to the model's effective performance in classifying facial expressions into the seven emotion categories. Continued fine-tuning and experimentation with hyperparameters and CNN architectures could potentially improve results further.

Key Findings:

In this project, several models were developed and evaluated for classifying facial expressions using the Kaggle FER-2013 dataset. Each model employed different approaches and configurations, resulting in varied performance outcomes.

1. Custom CNN:

- Training Accuracy: **65.58%,**
- Validation Accuracy:56.71%

This model demonstrated a moderate training accuracy but showed a relatively lower validation accuracy, indicating potential overfitting or inadequate generalization to unseen data.

2. 2. Custom CNN with Image Augmentation:

- Training Accuracy: 57.08%
- Validation Accuracy: 59.14%

Incorporating image augmentation techniques improved the validation accuracy compared to the baseline Custom CNN. However, the training accuracy decreased, suggesting that while augmentation helped with generalization, it may have also introduced challenges in model convergence.

3. 3. Transfer Learning – VGG16:

- Training Accuracy: 55.93%
- Validation Accuracy: 55.00%

Utilizing VGG16 with transfer learning resulted in the highest validation accuracy among the models. This indicates that leveraging pre-trained features and fine-tuning on the specific dataset contributed significantly to better performance.

4. 4. Transfer Learning - ResNet50:

- Training Accuracy: 62.61%
- Validation Accuracy: 60.80%

The ResNet50 model, another transfer learning approach, achieved competitive performance, with validation accuracy slightly below that of VGGNET. This model benefited from residual connections and a deeper architecture, which aided in learning complex features.

Conclusion:

The experimentation with different models highlighted the effectiveness of transfer learning in improving classification performance on facial emotion recognition tasks. Among the models tested, ResNET 50 achieved the highest validation accuracy, underscoring the value of leveraging pre-trained networks to capture intricate features in facial images. Although the VGG 16 model also performed well, the ResNET 50 model stood out for its superior generalization capability. The Custom CNNs, while providing useful baseline results, demonstrated the benefits of advanced architectures and augmentation techniques in achieving better accuracy and robustness.

Additional Plan of Action:

To further enhance the performance of the facial emotion recognition system, several steps will be undertaken.

1. **Exploration of Additional Architectures:**

- Expanding the exploration to include more advanced and diverse CNN architectures will be a key focus. This may involve experimenting with state-of-the-art models such as EfficientNet, InceptionV3, or DenseNet, which are known for their efficiency and accuracy in image classification tasks.

2. **Enhanced Fine-Tuning:**

- Further fine-tuning of the existing models will be conducted to optimize their performance. This includes adjusting hyperparameters, exploring different learning rates, and implementing more sophisticated regularization techniques to reduce overfitting and improve generalization.

3. **Improving Image Quality and Data Labeling:**

- Given that the current input images are 48x48 pixel grayscale images, which are relatively lower in resolution and quality, efforts will be made to obtain higher resolution images. Improved image quality can significantly impact model performance. Additionally, addressing the challenge of classifying difficult emotions will involve refining the dataset with more accurate and well-labeled images, which can help in achieving better accuracy and reliability in the model predictions.

By incorporating these strategies, the goal is to achieve a more robust and accurate emotion classification model that can handle diverse and complex facial expressions with higher precision.