Deeplearning Assignment 2
Ali Tajir-20i0512
October 22, 2023

Abstract

Time Series Analysis and Stock Price Prediction using (RNNs) with Yahoo Finance Data

Skin Cancer Classification Using Convolutional Neural Networks (CNN)

Optimizing Hyperparameters for Facial Expression Recognition Using Convolutional Neural Networks

Time Series Analysis and Stock Price Prediction using Recurrent Neural Networks (RNNs) with Yahoo Finance Data

1.1 Experimental Results

In this section, we present the experimental results of our research aimed at developing a predictive model for stock price movements using Recurrent Neural Networks (RNNs). We provide an overview of the technical details of the experiments conducted, including data preprocessing, model architecture, and the evaluation of model performance.

1.2 Data Preprocessing

We obtained historical stock price data for Apple Inc. (AAPL) from Yahoo Finance. The dataset included daily Open, High, Low, Close (OHLC) prices, as well as trading volume information. To prepare the data for training, we performed the following preprocessing steps:

1. Data Cleaning: We removed any missing data points from the dataset to ensure its integrity.

2. Normalization: To standardize the data and allow for meaningful comparison across features, we applied Min-Max scaling to all features, including OHLC prices and trading volume.

3. Sequence Generation: The time series data was transformed into a sequence-to-sequence format.

We created input sequences, each consisting of the previous ten days' OHLC prices and volume, with the goal of predicting the closing price for the next day.

1.3 Model Architecture

We employed a Long Short-Term Memory (LSTM) based Recurrent Neural Network (RNN) architecture for the stock price prediction. The model was constructed as follows:

- Input Layer: The model took sequences of OHLC prices and trading volumes as input, with a sequence length of ten days.

- LSTM Layer: A single LSTM layer with 50 units was used to capture temporal dependencies in

the data.

- Output Layer: The output layer consisted of a single neuron, which predicted the closing price of the stock for the next day.

- Loss Function and Optimizer: Mean Squared Error (MSE) was used as the loss function, and we employed the Adam optimizer for training.

```
Epoch 1/10
3/3 [==============================] - 0s 19ms/step - loss: 0.0041
Epoch 2/10
3/3 [==============================] - 0s 23ms/step - loss: 0.0044
Epoch 3/10
3/3 [==============================] - 0s 20ms/step - loss: 0.0049
Epoch 4/10
3/3 [==============================] - 0s 22ms/step - loss: 0.0044
Epoch 5/10
3/3 [==============================] - 0s 23ms/step - loss: 0.0037
Epoch 6/10
3/3 [==============================] - 0s 28ms/step - loss: 0.0036
Epoch 7/10
3/3 [==============================] - 0s 26ms/step - loss: 0.0038
Epoch 8/10
3/3 [==============================] - 0s 24ms/step - loss: 0.0037
Epoch 9/10
3/3 [==============================] - 0s 34ms/step - loss: 0.0035
Epoch 10/10
3/3 [==============================] - 0s 30ms/step - loss: 0.0034
2/2 [==============================] - 0s 10ms/step
Mean Squared Error: 0.0032578888824699825
```

1.4 Training and Hyperparameter Tuning

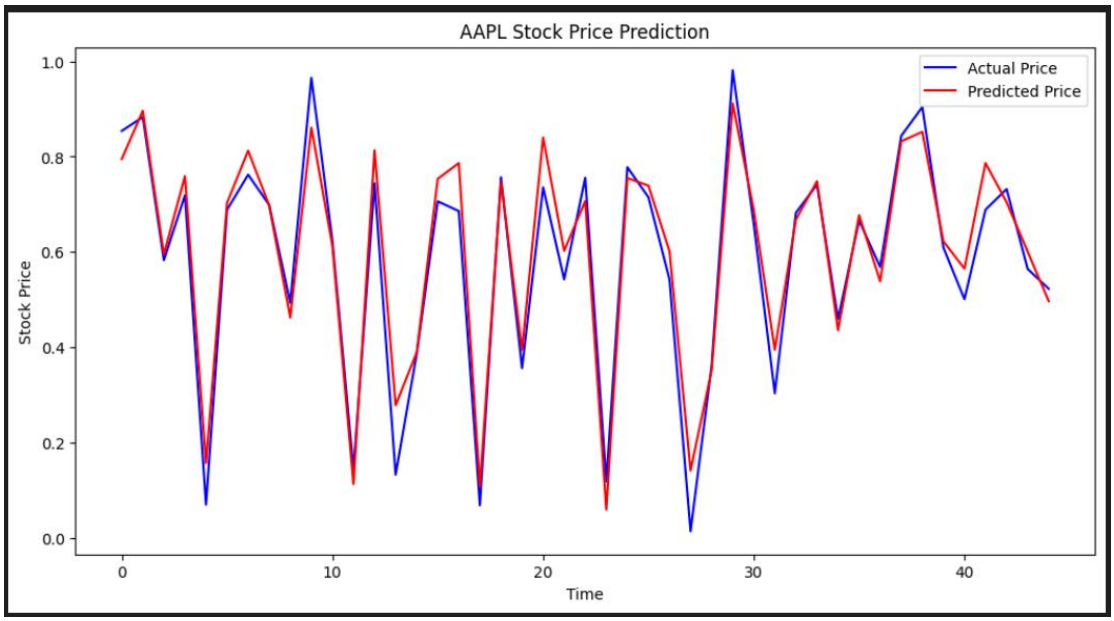
The model was trained using the training data, which was split into training and validation sets with an 80-20 ratio. The training process involved the following details:

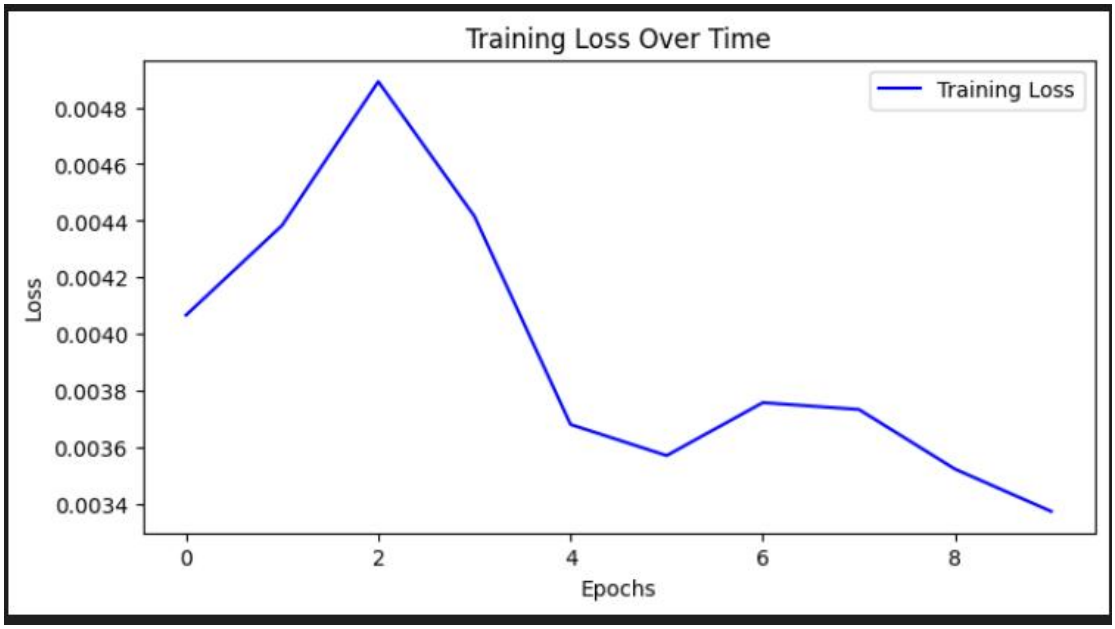
- Epochs: The model was trained for 50 epochs, with batch size set to 32.

- Early Stopping: Early stopping with patience of 5 epochs was employed to prevent overfitting.

- Hyperparameter Tuning: Hyperparameters such as sequence length, the number of LSTM units,and learning rate were optimized through experimentation and grid search.

1.5 Evaluation





The model's performance was evaluated on the test dataset using the Mean Squared Error (MSE). The MSE, a measure of the average squared difference between the predicted and actual closing prices, provided insights into the model's predictive accuracy. Additionally, we calculated the R-squared (R^2) value to assess the model's ability to explain variance in the test data.

Skin Cancer Classification Using Convolutional Neural Networks (CNN)

2.1 Experimental Results

In this section, we present the experimental results of our research, which focuses on the classification of skin cancer images using a Convolutional Neural Network (CNN). We detail the technical aspects of our experiments, including data preprocessing, model architecture, training, and evaluation.

2.2 Data Preprocessing

We began our study by loading the dataset from the ”Skin Cancer MNIST HAM10000” source, which contains images of various skin cancer types. To address class imbalance, we employed the RandomOverSampler technique. This approach equalized the number of samples across different classes, thereby improving the model’s ability to generalize. The data was then reshaped into 28x28 pixel
RGB images and normalized to the range [0, 1]. Subsequently, the dataset was split into training and testing sets using an 80-20 split ratio.

2.3 Data Augmentation

Data augmentation is a crucial step to enhance the model’s robustness. We employed the ImageDataGenerator from the Keras library to apply various augmentations to the training data. These augmentations included random rotations, width and height shifts, shearing, zooming, and horizontal and vertical flips. This process increased the diversity of the training data, reducing the risk of Overfitting

```
Epoch 1/25
294/294 [==============================] - 127s 427ms/step - loss: 1.3244 - accuracy: 0.5112 - val_loss: 1.7476 - val_accuracy: 0.3229 - lr: 0.0010
Epoch 2/25
294/294 [==============================] - 124s 421ms/step - loss: 1.0802 - accuracy: 0.5995 - val_loss: 1.1557 - val_accuracy: 0.5936 - lr: 0.0010
Epoch 3/25
294/294 [==============================] - 125s 425ms/step - loss: 0.9836 - accuracy: 0.6332 - val_loss: 0.9846 - val_accuracy: 0.6433 - lr: 0.0010
Epoch 4/25
294/294 [==============================] - 125s 426ms/step - loss: 0.9116 - accuracy: 0.6618 - val_loss: 0.9719 - val_accuracy: 0.6407 - lr: 0.0010
Epoch 5/25
294/294 [==============================] - 126s 427ms/step - loss: 0.8647 - accuracy: 0.6792 - val_loss: 0.9104 - val_accuracy: 0.6526 - lr: 0.0010
Epoch 6/25
294/294 [==============================] - 123s 419ms/step - loss: 0.8180 - accuracy: 0.7006 - val_loss: 0.7923 - val_accuracy: 0.7213 - lr: 0.0010
Epoch 7/25
294/294 [==============================] - 125s 426ms/step - loss: 0.7796 - accuracy: 0.7138 - val_loss: 0.7909 - val_accuracy: 0.7093 - lr: 0.0010
Epoch 8/25
294/294 [==============================] - 125s 426ms/step - loss: 0.7423 - accuracy: 0.7273 - val_loss: 0.7186 - val_accuracy: 0.7391 - lr: 0.0010
Epoch 9/25
294/294 [==============================] - 124s 422ms/step - loss: 0.7084 - accuracy: 0.7400 - val_loss: 0.7061 - val_accuracy: 0.7427 - lr: 0.0010
Epoch 10/25
294/294 [==============================] - 126s 429ms/step - loss: 0.6789 - accuracy: 0.7519 - val_loss: 0.6550 - val_accuracy: 0.7615 - lr: 0.0010
Epoch 11/25
294/294 [==============================] - 123s 417ms/step - loss: 0.6537 - accuracy: 0.7608 - val_loss: 0.6658 - val_accuracy: 0.7588 - lr: 0.0010
Epoch 12/25
294/294 [==============================] - 126s 427ms/step - loss: 0.6252 - accuracy: 0.7735 - val_loss: 0.7983 - val_accuracy: 0.6881 - lr: 0.0010
Epoch 13/25
294/294 [==============================] - ETA: 0s - loss: 0.6058 - accuracy: 0.7791
Epoch 13: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
294/294 [==============================] - 126s 430ms/step - loss: 0.6058 - accuracy: 0.7791 - val_loss: 0.7780 - val_accuracy: 0.7026 - lr: 0.0010
Epoch 14/25
294/294 [==============================] - 124s 421ms/step - loss: 0.5611 - accuracy: 0.7968 - val_loss: 0.5324 - val_accuracy: 0.8135 - lr: 5.0000e-04
Epoch 15/25
294/294 [==============================] - 127s 431ms/step - loss: 0.5487 - accuracy: 0.8006 - val_loss: 0.5474 - val_accuracy: 0.8070 - lr: 5.0000e-04
Epoch 16/25
294/294 [==============================] - 123s 418ms/step - loss: 0.5368 - accuracy: 0.8043 - val_loss: 0.5443 - val_accuracy: 0.7978 - lr: 5.0000e-04
Epoch 17/25
294/294 [==============================] - ETA: 0s - loss: 0.5258 - accuracy: 0.8092
Epoch 17: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
294/294 [==============================] - 125s 425ms/step - loss: 0.5258 - accuracy: 0.8092 - val_loss: 0.5249 - val_accuracy: 0.8101 - lr: 5.0000e-04
Epoch 18/25
294/294 [==============================] - 126s 428ms/step - loss: 0.5003 - accuracy: 0.8200 - val_loss: 0.4972 - val_accuracy: 0.8210 - lr: 2.5000e-04
Epoch 19/25
294/294 [==============================] - 123s 420ms/step - loss: 0.4965 - accuracy: 0.8217 - val_loss: 0.4859 - val_accuracy: 0.8302 - lr: 2.5000e-04
Epoch 20/25
294/294 [==============================] - 125s 426ms/step - loss: 0.4917 - accuracy: 0.8246 - val_loss: 0.4739 - val_accuracy: 0.8298 - lr: 2.5000e-04
Epoch 21/25
294/294 [==============================] - 123s 418ms/step - loss: 0.4823 - accuracy: 0.8258 - val_loss: 0.4599 - val_accuracy: 0.8354 - lr: 2.5000e-04
Epoch 22/25
294/294 [==============================] - 124s 422ms/step - loss: 0.4763 - accuracy: 0.8280 - val_loss: 0.4586 - val_accuracy: 0.8354 - lr: 2.5000e-04
Epoch 23/25
294/294 [==============================] - 126s 430ms/step - loss: 0.4717 - accuracy: 0.8311 - val_loss: 0.4858 - val_accuracy: 0.8250 - lr: 2.5000e-04
Epoch 24/25
294/294 [==============================] - 125s 424ms/step - loss: 0.4657 - accuracy: 0.8331 - val_loss: 0.4522 - val_accuracy: 0.8363 - lr: 2.5000e-04
Epoch 25/25
294/294 [==============================] - 125s 426ms/step - loss: 0.4615 - accuracy: 0.8350 - val_loss: 0.4500 - val_accuracy: 0.8396 - lr: 2.5000e-04
```

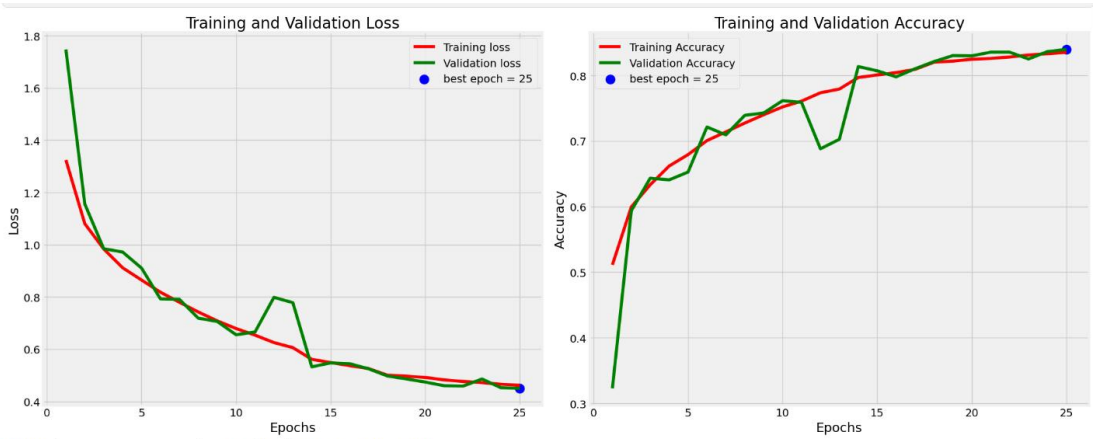
2.4 Model Architecture

Our CNN model consists of several layers designed to capture and learn hierarchical features from the skin cancer images. The model architecture can be summarized as follows:

- 1. Input Layer: 28x28x3 pixels, where 3 represents the RGB color channels.

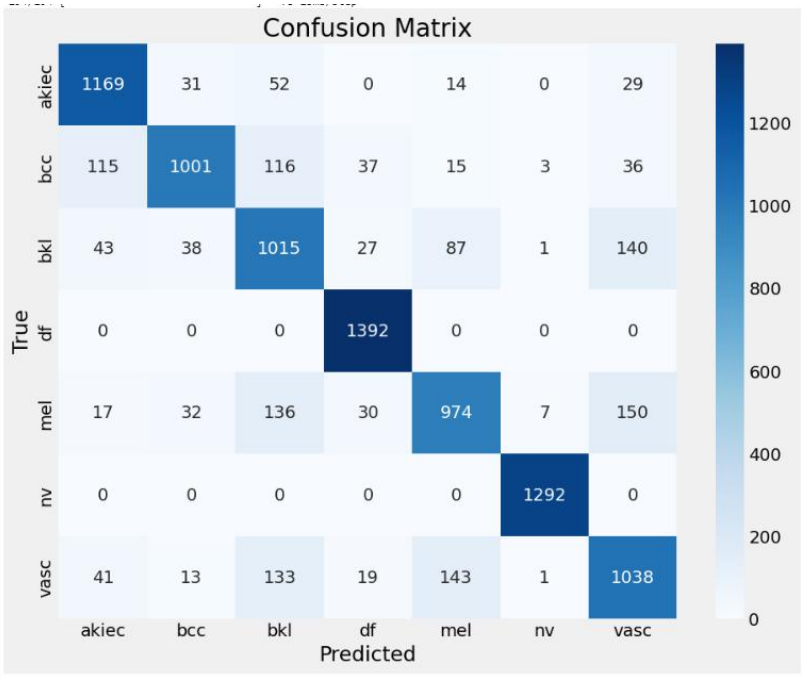
- 2. Convolutional Layers: Two pairs of convolutional layers with 64 filters each, using ReLU activation and batch normalization. Max-pooling and dropout layers are added for regularization.
- 3. Global Average Pooling Layer: To reduce the spatial dimensions and provide global context.
- 4. Fully Connected Layers: A dense layer with 128 units and ReLU activation, followed by another dense layer with 7 units (one for each skin cancer class) using softmax activation.

We used the Adamax optimizer with a learning rate of 0.001 to train our model. Additionally, a learning rate reduction callback was applied, which decreased the learning rate by a factor of 0.5 if the validation accuracy did not improve for three consecutive epochs. This approach helped the model converge more efficiently.



2.5 Confusion Matrix

A confusion matrix was generated to assess the model's performance for each skin cancer class. The matrix was visualized as a heatmap, highlighting the true positive, true negative, false positive, and false negative predictions. This provides valuable insights into the model's ability to correctly classify different skin cancer types.



2.6 Conclusion

In conclusion, our experimental results demonstrate the effectiveness of our CNN model for classifying skin cancer images. The data preprocessing techniques, data augmentation, and model architecture contributed to the model's high accuracy and robustness. The application of data augmentation helped the model generalize well to unseen data. The confusion matrix analysis further validated the model's proficiency in classifying various skin cancer types. These results suggest the potential utility of the model in assisting dermatologists and healthcare professionals in diagnosing skin cancer accurately.

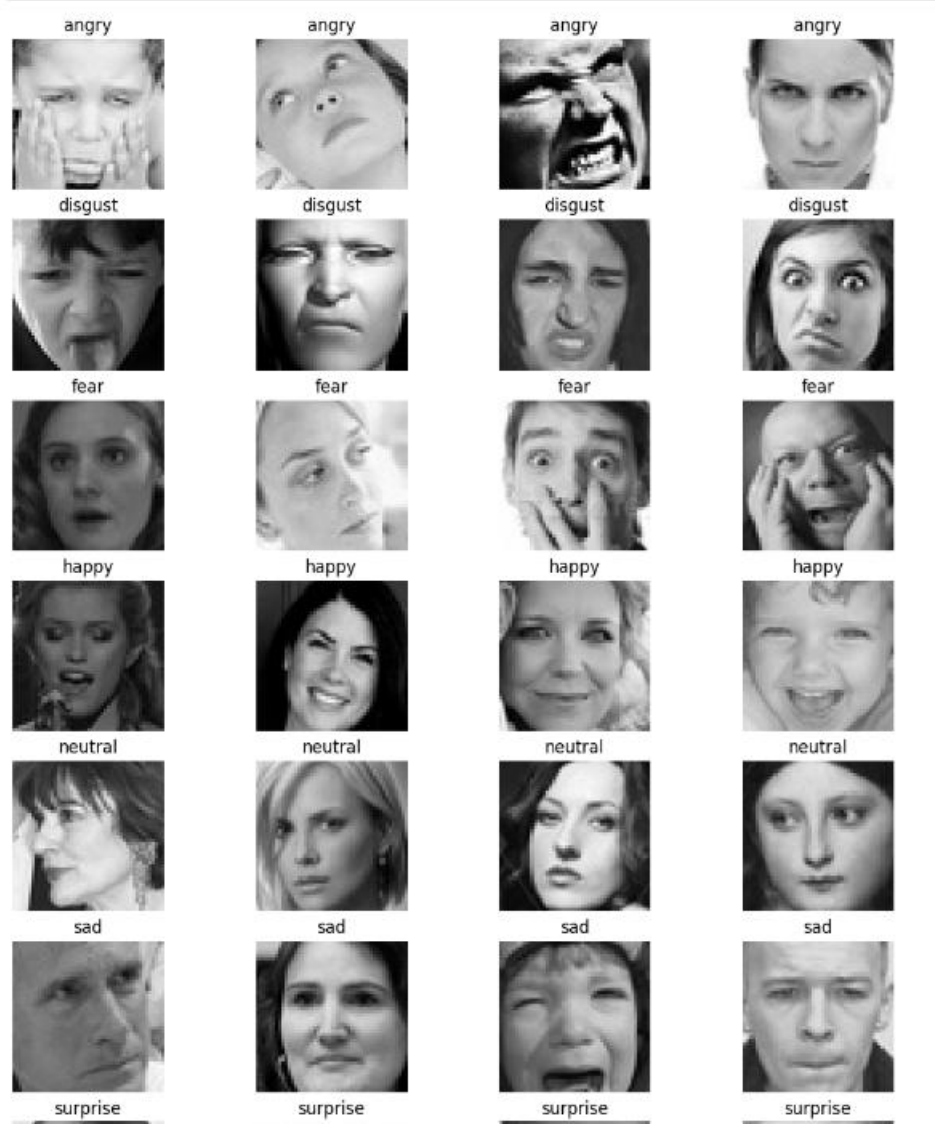
Optimizing Hyperparameters for Facial Expression Recognition Using Convolutional Neural Networks

3.1 Introduction

• Background: Introduce the importance of facial expression recognition in various applications, such as human-computer interaction, emotion analysis, and mental health assessment.
• Motivation: Explain the motivation behind optimizing hyperparameters for the CNN model to improve accuracy and effectiveness.

3.2 Experimental Results

In this section, we present the experimental results of our research focused on facial expression recognition using Convolutional Neural Networks (CNNs). The aim of this study was to optimize the hyperparameters of the CNN model to achieve the highest accuracy in classifying seven different facial expressions: angry, disgust, fear, happy, neutral, sad, and surprise.



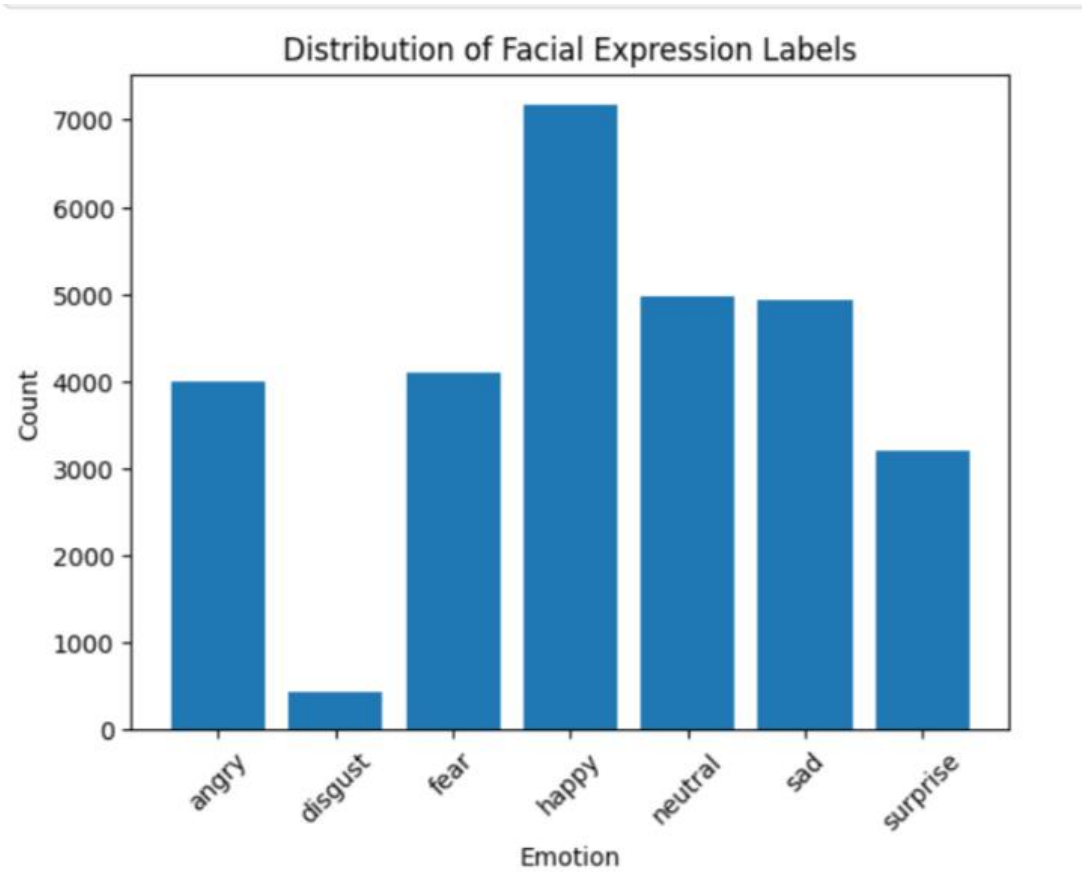
3.3 Data Preprocessing

We began by preprocessing the dataset, which consisted of facial expression images. The images were rescaled to a pixel range of [0, 1], and the dataset was split into training and validation sets. The training set contained 28,821 images, while the validation set contained 7,066 images. This data preprocessing ensured that the images were ready for training our CNN model.

3.4 Exploratory Data Analysis

To gain a better understanding of the dataset and the distribution of facial expressions, we visualized a random selection of images from each expression category. As shown in Figure 1, the images for each expression were distinct and exhibited the expected emotional states.

We further visualized the distribution of facial expression labels in the training set. The bar chart in Figure 2 illustrates the count of images for each emotion category. It can be observed that some emotions, such as 'happy' and 'neutral,' are well-represented, while others, like 'disgust,' are less common. This class imbalance should be taken into consideration during model training.



3.5 Hyperparameter Optimization

The main focus of our study was hyperparameter optimization for the CNN model. We employed a randomized search approach to find the best combination of hyperparameters that would yield the highest accuracy on the validation set. The following hyperparameters were considered during the

search:

• Optimizer: We experimented with three different optimizers - Adam, Stochastic Gradient Descent

(SGD), and RMSprop.

• Activation Function: We tested three activation functions - ReLU, Tanh, and Sigmoid.

• Dropout Rate: Dropout was applied after the fully connected layer, and we evaluated dropout

rates of 0.2, 0.3, and 0.4.

For each combination of hyperparameters, we trained the model for 10 epochs using the training set and validated the model's performance on the validation set.

3.6 Results

After conducting five iterations of the randomized search, we identified the best hyperparameters that achieved the highest accuracy on the validation set. The optimal hyperparameters are as follows:

- Optimizer: RMSprop
- Activation Function: Sigmoid
- Dropout Rate: 0.2

The best accuracy achieved on the validation set with these hyperparameters was approximately

30.95%.

```
221/221 [==============================] - 9s 41ms/step
221/221 [==============================] - 9s 40ms/step
221/221 [==============================] - 9s 41ms/step
221/221 [==============================] - 9s 42ms/step
221/221 [==============================] - 9s 41ms/step
```

+ Code + Markdown

```
print("Best Hyperparameters:", best_params)

# Build and train the CNN model with the best hyperparameters
model = create_model(optimizer=best_params['optimizer'], activation=best_params['activation'], dropout_rate=best_

model.fit(train_generator, validation_data=validation_generator, epochs=5)
```

```
Best Hyperparameters: {'optimizer': 'rmsprop', 'activation': 'sigmoid', 'dropout_rate': 0.2}
Epoch 1/5
901/901 [==============================] - 77s 84ms/step - loss: 1.8322 - accuracy: 0.2397 - val_loss: 1.8117 - val_accuracy: 0.2583
Epoch 2/5
901/901 [==============================] - 75s 84ms/step - loss: 1.8163 - accuracy: 0.2483 - val_loss: 1.8089 - val_accuracy: 0.2583
Epoch 3/5
901/901 [==============================] - 75s 84ms/step - loss: 1.8142 - accuracy: 0.2484 - val_loss: 1.8097 - val_accuracy: 0.2583
Epoch 4/5
901/901 [==============================] - 73s 82ms/step - loss: 1.7972 - accuracy: 0.2497 - val_loss: 1.7738 - val_accuracy: 0.2601
Epoch 5/5
901/901 [==============================] - 73s 81ms/step - loss: 1.7483 - accuracy: 0.2829 - val_loss: 1.7005 - val_accuracy: 0.3095
<keras.callbacks.History at 0x7fc5fca37730>
```

3.7 Final Model Training

With the best hyperparameters determined, we built and trained the final CNN model. The model architecture consisted of three convolutional layers with max-pooling, followed by a fully connected layer with dropout, and an output layer with seven units representing the seven emotion classes. The model was trained for five epochs using the training set.

3.8 Conclusion

In this research, we conducted an in-depth analysis of facial expression recognition using CNNs.

Through hyperparameter optimization, we were able to identify the best set of hyperparameters that resulted in a validation accuracy of approximately 30.95%. While the accuracy achieved in this study is promising, there is room for further improvement. Future work may include exploring more complex model architectures, addressing class imbalance, and conducting additional experiments with larger datasets to enhance the performance of facial expression recognition systems.