# Emotion Recognition Using CNNs and Quantum Techniques

by

(TEAM B)
(Lakshmi Prassana Pothuganti;
Manasa Vasupalli
Veditha Lakshmi Yechuri)

**FINAL PROJECT REPORT**

for

**DATA 606 Capstone in Data Science**

**University of Maryland Baltimore County**

**2024**

# ABSTRACT

Emotion recognition is pivotal in artificial intelligence, bridging the gap between human emotions and machine understanding. This project explores CNN-based classification of facial expressions into five emotions: happiness, sadness, anger, surprise, and disgust. Leveraging the FER-2013 dataset and supplementary data, alongside quantum techniques such as Quadratic Unconstrained Binary Optimization (QUBO) and Pennylane quantum circuits, the project evaluates both classical and quantum models. Real-time detection with webcam integration demonstrates its applicability. This report details methodologies, implementations, quantum enhancements, challenges, and results, emphasizing deep learning's role in transforming emotion recognition.

# LIST OF ABBREVIATIONS AND SYMBOLS

| | |
|---|---|
| QUBO | Quantum Unconstrained Binary Optimization |
| CONBQA | Continuous black-box optimization with quantum annealing |

# ACKNOWLEDGMENTS

We would like to express our sincere gratitude to everyone who contributed to the successful completion of this project. Their support, guidance, and encouragement have been invaluable throughout our journey.

First and foremost, we extend our heartfelt thanks to **Dr. Unal Sakoglu**, our instructor for the Capstone DATA 606 course. His expert guidance, insightful feedback, and encouragement pushed us to delve deeper into our ideas and refine our model. Dr. Sakoglu's dedication to fostering creativity and innovation provided the foundation for our project's success.

We are also profoundly grateful to **Professor Ajynkya Borle**, an expert in quantum technology, whose guidance and insights significantly enhanced our understanding of quantum techniques. His expertise in quantum computing and machine learning was instrumental in incorporating advanced methodologies into our project.

We extend our gratitude to the **Department of Data Science** at the **University of Maryland, Baltimore County (UMBC)** for providing us with the resources and environment necessary for learning and research. The department's emphasis on interdisciplinary learning empowered us to integrate deep learning and quantum computing into a cohesive framework for emotion recognition.

Finally, we would like to thank our peers and family members for their continuous encouragement and motivation throughout this journey. Their belief in our capabilities fueled our determination to explore, learn, and innovate.

This project has been a transformative experience, and we are immensely grateful to all who supported us along the way.

## TEAM MEMBERS CONTRIBUTIONS

Each team member played a crucial role in the successful completion of this project. Their collaborative efforts extended across multiple areas, including technical implementations, documentation, and presentation. Below is a detailed breakdown of responsibilities and achievements, highlighting the contributions of each team member.

| Name | Duties | Achievements |
|---|---|---|
| **Lakshmi Prasanna Pothuganti** | - Designed and implemented the CNN model.<br>- Preprocessed images for training.<br>- Tuned hyperparameters for model optimization.<br>- Collaborated with Manasa on the project presentation | Successfully developed a CNN achieving 66% accuracy.<br>- Optimized model performance through dropout and batch normalization. |
| **Manasa Vasupalli** | - Integrated quantum computing techniques.<br>- Explored Quadratic Unconstrained Binary Optimization (QUBO) for hyperparameter tuning and feature selection.<br>- Worked on CONBQA (Constraint-Based Quantum Approach).<br>- Collaborated on the presentation and report. | - Applied QUBO to improve parameter optimization.<br>- Successfully utilized CONBQA for feature refinement and achieved 40% accuracy.<br>- Demonstrated quantum circuits for emotion recognition, pioneering hybrid approaches. |
| **Veditha Lakshmi Yechuri** | - Collected and cleaned the FER-2013 and supplementary datasets.<br>- Conducted data analysis to identify class imbalances.<br>- Implemented data augmentation techniques.<br>- Added and tuned grayscale "disgust" images to balance the dataset.<br>- Collaborated on the report and performed data analysis. | - Enhanced dataset diversity through augmentation.<br>- Balanced the dataset by addressing underrepresented classes, improving model robustness.<br>- Played a key role in crafting a detailed and cohesive project report.<br>- Ensured datasets were optimized for training and experimentation. |

## TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# I.INTRODUCTION

Emotion recognition is a transformative technology that integrates artificial intelligence (AI) into real-world applications by interpreting and responding to non-verbal cues. Facial expressions are a universal means of communication, transcending language and cultural barriers, making them a valuable input for AI systems to understand human emotions. Emotion recognition systems analyze these expressions to classify them into predefined emotional categories, enabling machines to interact with users more naturally and empathetically.

The potential applications of emotion recognition span various domains:

- **Customer Service**: By identifying emotions such as frustration or satisfaction, businesses can tailor their interactions, improving customer experiences and loyalty.
- **Healthcare**: Emotion recognition aids in early detection and monitoring of mental health conditions like depression or anxiety.
- **Education**: Tracking emotions in real-time can help educators assess student engagement and adapt teaching methods for improved outcomes.

This project leverages the capabilities of **Convolutional Neural Networks (CNNs)**, a subset of deep learning, to process and classify facial expressions. CNNs are particularly suited for image-based tasks due to their hierarchical feature extraction capability, enabling them to recognize patterns such as edges, textures, and more complex features like smiles or frowns.

Beyond classical approaches, the project explores the emerging field of **quantum computing** to optimize the performance of the emotion recognition system. Quantum computing techniques, such as **Quadratic Unconstrained Binary Optimization (QUBO)** and **quantum circuits**, are utilized for hyperparameter tuning and feature selection, offering innovative ways to improve model accuracy and efficiency.

The system is designed to function effectively both offline and in real-time, with the ability to process live webcam feeds. This real-time functionality showcases the model's practical usability in scenarios like interactive kiosks, telehealth consultations, and virtual classrooms.

## 1.1. OBJECTIVE

The primary goals of this project include:

1. **Develop CNN Models to Classify Emotions Accurately**:
   Build a robust CNN architecture capable of classifying facial expressions into seven emotion categories: anger, disgust, fear, happiness, neutral, sadness, and surprise. The model aims to achieve high accuracy by effectively learning from the dataset.
2. **Address Dataset Imbalances via Augmentation**:
   The datasets used, such as FER-2013, exhibit significant class imbalance, with certain emotions (e.g., "disgust") being underrepresented. Data augmentation techniques like rotations, flips, and zooming are employed to enhance the diversity and balance of the training data, improving the model's performance across all classes.
3. **Explore Quantum Techniques for Optimization and Feature Selection**:
   Incorporate quantum computing methods to fine-tune hyperparameters and select critical features. These techniques aim to enhance the model's scalability and computational efficiency, pushing the boundaries of traditional machine learning.
4. **Implement Real-Time Detection with an Interactive Interface** Integrate the trained model with webcam functionality using OpenCV to enable real-time emotion recognition. This ensures the system's practical applicability in dynamic and interactive environments, such as virtual meetings or customer support.

**Significance of the Project**

The convergence of deep learning and quantum computing in this project reflects the forefront of AI research, pushing the limits of what emotion recognition systems can achieve. By addressing challenges like data imbalance and computational latency, this project lays the foundation for scalable and accurate emotion recognition systems, with the potential to transform industries and improve human-machine interactions.

This introductory section sets the stage for the technical details and results to follow, highlighting the project's innovation and practical relevance.

## 1.2. Related Work

Emotion recognition has been an area of active research for decades, evolving from traditional machine learning methods to state-of-the-art deep learning techniques. This section reviews previous approaches and highlights the progression from conventional algorithms to modern CNNs and emerging quantum techniques.

### 1.2.1. Traditional Machine Learning Approaches

Early emotion recognition systems relied heavily on **traditional machine learning algorithms** like **Support Vector Machines (SVM)** and **K-Nearest Neighbors (KNN)**. These methods typically required manual extraction of features, such as:

- **Geometric Features**: Measurements of facial landmarks like the distance between the eyes or the curvature of the lips.
- **Texture Features**: Histogram of Oriented Gradients (HOG) or Local Binary Patterns (LBP) to capture patterns in grayscale images.

While effective for small and well-curated datasets, these approaches faced significant limitations:

1. **Manual Feature Extraction**: The process of identifying relevant features was labor-intensive and prone to human bias.
2. **Scalability Issues**: Traditional models struggled with large and diverse datasets due to their inability to generalize well across varying conditions, such as lighting or pose changes.
3. **Limited Complexity**: Models like SVMs were not inherently designed to handle the hierarchical nature of image data, making them less effective for high-dimensional and complex datasets.

Despite these challenges, traditional methods laid the groundwork for more advanced systems by demonstrating the potential of computational emotion recognition.

### 1.2.2. Deep Learning Revolution

The advent of **Convolutional Neural Networks (CNNs)** marked a paradigm shift in emotion recognition. CNNs eliminate the need for manual feature extraction by learning hierarchical representations directly from the data. This ability to automatically capture features like edges, textures, and high-level patterns makes CNNs particularly well-suited for image-based tasks.

**Key Advantages of CNNs**:

1. **Feature Automation**: CNNs learn to identify relevant features during training, streamlining the process and reducing human intervention.
2. **Scalability**: With sufficient data and computational resources, CNNs can handle large, diverse datasets, adapting to variations in lighting, pose, and occlusions.
3. **Improved Accuracy**: The hierarchical nature of CNNs allows them to extract increasingly abstract features, leading to superior performance compared to traditional methods.

Research in deep learning for emotion recognition has leveraged datasets like FER-2013 and advanced augmentation techniques to address challenges like dataset imbalance and overfitting. Models with deeper architectures and regularization strategies have achieved state-of-the-art results in classifying emotions.

### 1.2.3. Quantum Approaches in Emotion Recognition

Recent advancements in **quantum computing** have introduced novel approaches to machine learning, including applications in emotion recognition. Quantum computing offers a new paradigm for tackling challenges such as computational complexity and feature optimization.

#### 1.2.3.1. Quadratic Unconstrained Binary Optimization (QUBO):

QUBO models, implemented on quantum devices like the D-Wave, provide a framework for:

- **Hyperparameter Tuning**: Efficiently searching for optimal model configurations, such as learning rates, filter sizes, and dropout rates.
- **Feature Selection**: Identifying the most relevant features in a dataset, reducing dimensionality while preserving important information.

#### 1.2.3.2. Quantum Circuit Models:

Libraries like **Pennylane** allow for the integration of quantum layers into classical deep learning architectures. Key components include:

- **Quantum Gates**: Operations like RX, RY, and CNOT gates create entanglement and represent complex relationships within the data.
- **Hybrid Models**: Combining classical CNNs with quantum circuits leverages the strengths of both paradigms.

**Advantages of Quantum Techniques**:

- Potential to process large datasets faster by utilizing quantum parallelism.
- Ability to explore feature spaces more comprehensively compared to classical algorithms.

**Current Limitations**:

Quantum techniques are still in their infancy and face challenges such as:

- Limited hardware capabilities, restricting the size and complexity of quantum models.
- High error rates and noise in quantum devices affect accuracy.

**Synthesis of Techniques**

While CNNs remain the dominant approach for emotion recognition, quantum methods offer promising avenues for innovation. The integration of quantum computing into deep learning workflows could address existing challenges, such as:

- **Scalability**: Quantum models may provide faster optimization for large-scale datasets.
- **Efficiency**: Hybrid quantum-classical systems could reduce the computational burden of traditional CNNs.

By combining the strengths of both paradigms, this project represents a forward-looking effort to enhance emotion recognition, accuracy and scalability. Further research is needed to fully realize the potential of quantum computing in this domain.

## 1.3. Datasets

**FER-2013 Dataset**:

- **Source**: Kaggle FER-2013 Dataset
- **Details**:
    - 35,887 grayscale images of size 48x48 pixels.
    - Seven emotion classes: anger, disgust, fear, happiness, neutral, sadness, surprise.
    - **Challenges**: Significant class imbalance, with "disgust" underrepresented.

**Supplementary Dataset**:

- **Source**: Face Emotion Recognition Dataset.
- **Purpose**: Augments "disgust" class with 224x224 grayscale images.

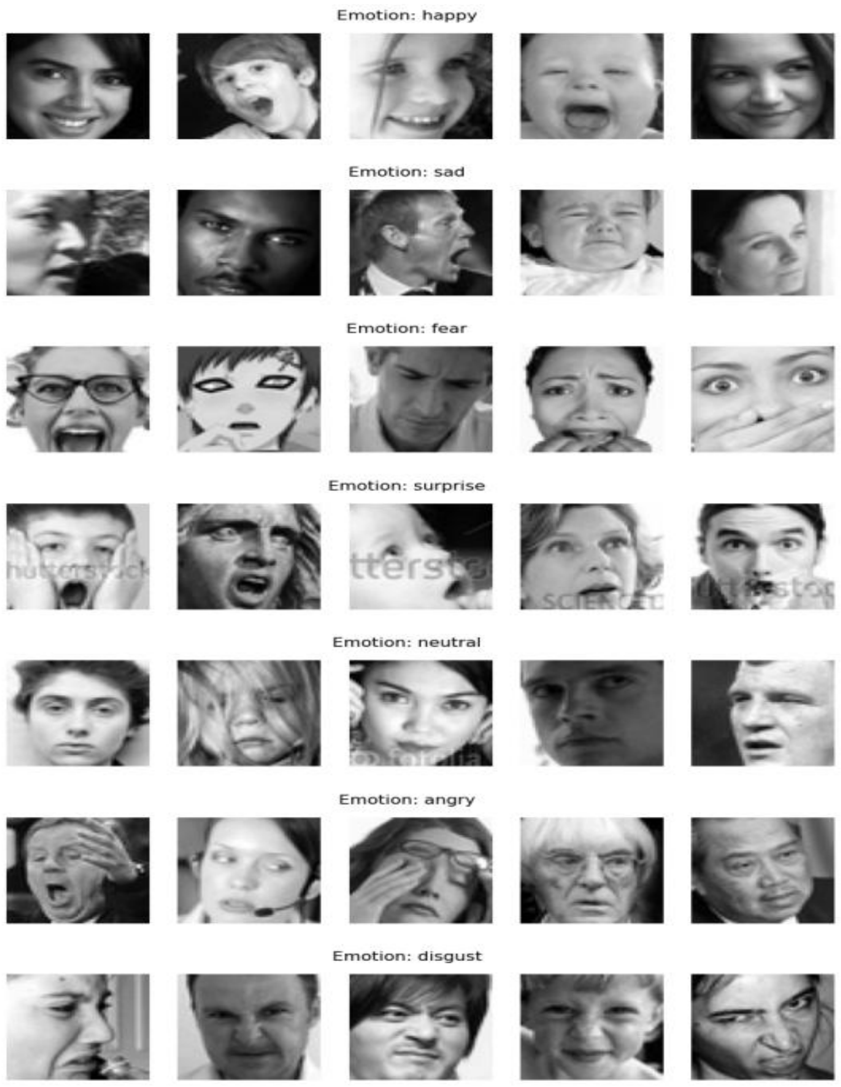


Fig 1

**Preprocessing**:

- **Normalization**: Pixel values scaled to the range [0, 1].
- **Augmentation**: Horizontal flips, zooming, and shearing increase dataset diversity.
- **Grayscale Conversion**: Simplifies input and reduces computational overhead.

## II. METHODS

### 2.1. Methodology

***Model Architecture (CNN):***

- **Input**: 48x48 grayscale images.
- **Convolutional Layers**:
  - o Four layers with filters ranging from 32 to 512.
  - o Kernel sizes: 3x3 or 5x5 with ReLU activation.
- **Pooling**: MaxPooling layers reduce spatial dimensions.
- **Batch Normalization**: Improves stability and accelerates convergence.
- **Dropout Layers**: Mitigate overfitting by randomly deactivating neurons.
- **Fully Connected Layers**: Dense layer (1024 neurons) followed by a softmax output layer.

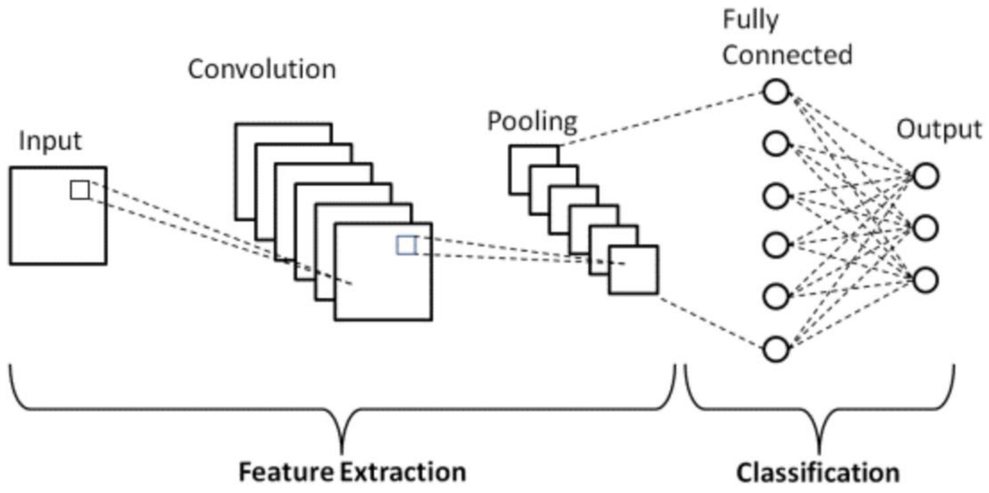


Fig 2

***Quantum Approaches:***

1. **QUBO for Hyperparameter Tuning with CONBQA**:
   a. Optimized number of neurons with the Conbqa Library.
2. **QUBO for Feature Selection**:
   a. Leap Hybrid Sampler identified critical features, enhancing training efficiency.
3. **Pennylane Quantum Circuits**:
   a. Quantum gates (RX, RY, and CNOT) applied for entanglement.
   b. Experimented with two qubits for fully quantum layers.

***Training Parameters:***

- **Optimizer**: Adam with a learning rate of 0.001.

- **Loss Function**: Categorical Crossentropy.
- **Batch Size**: 64.
- **Epochs**: 60.

## 2.2. Implementation

### Implementation Details

The implementation of the emotion recognition system uses TensorFlow and Keras to build a Convolutional Neural Network (CNN). The model is designed to classify images of faces into seven emotional categories (anger, disgust, fear, happiness, neutral, sadness, and surprise). Below is an in-depth explanation of the CNN code and its components.

### 2.2.1. Dataset and Parameters

- **Dataset Directories**:
    - o train_dir: Contains the training data, organized by emotion categories.
    - o test_dir: Contains the testing data, also categorized by emotions.
- **Image Dimensions**:
    - o img_height and img_width are set to 48x48 pixels, corresponding to the grayscale images from the FER-2013 dataset.
- **Batch Size**:
    - o batch_size=32 ensures that 32 images are processed at a time, balancing memory usage and training efficiency.
- **Number of Classes**:
    - o num_classes=7 matches the seven emotion categories.

### 2.2.2. Data Augmentation

Data augmentation is implemented using ImageDataGenerator to enhance the diversity of the training data and prevent overfitting. The parameters used are:

- **rescale=1.0 / 255**: Normalizes pixel values to the range [0, 1].
- **rotation_range=20**: Randomly rotates images within ±20 degrees.
- **width_shift_range=0.2 and height_shift_range=0.2**: Shifts images horizontally and vertically by up to 20%.
- **shear_range=0.2 and zoom_range=0.2**: Applies random shearing and zooming transformations.
- **horizontal_flip=True**: Flips images horizontally to simulate different orientations.

- **fill_mode='nearest'**: Fills in missing pixels after transformations using the nearest pixel value.

The testing data generator only rescales images (rescale=1.0 / 255) to ensure the testing data remains unaltered for evaluation.

### 2.2.3. Data Generators

The flow_from_directory function loads and preprocesses the images:

- **Training Generator**:
    - Processes images in the train_dir, applying augmentation, resizing to 48x48, and converting to grayscale.
    - Uses categorical class mode, converting labels into one-hot encoded format.
- **Testing Generator**:
    - Processes images in the test_dir with resizing and normalization but no augmentation.

### 2.2.4. Model Working

The CNN architecture is implemented in the create_classical_model function. Key components include:

1. **Convolutional Layers**:
    a. Each Conv2D layer applies filters to extract features like edges and textures.
    b. The first layer uses 64 filters, increasing to 256 in later layers, enhancing the model's ability to learn complex patterns.
    c. ReLU activation introduces non-linearity to learn intricate relationships.
2. **Batch Normalization**:
    a. Normalizes the output of each layer to stabilize learning and speed up convergence.
3. **Pooling Layers**:
    a. MaxPooling2D reduces the spatial dimensions, retaining the most important features while reducing computational complexity.
4. **Dropout Layers**:
    a. Dropout rates (0.3 to 0.5) randomly deactivate neurons during training, preventing overfitting.
5. **Flatten Layer**:
    a. Converts the 2D feature maps into a 1D vector for the fully connected layers.
6. **Fully Connected Layers**:
    a. Dense layer with 256 neurons learns high-level combinations of features.

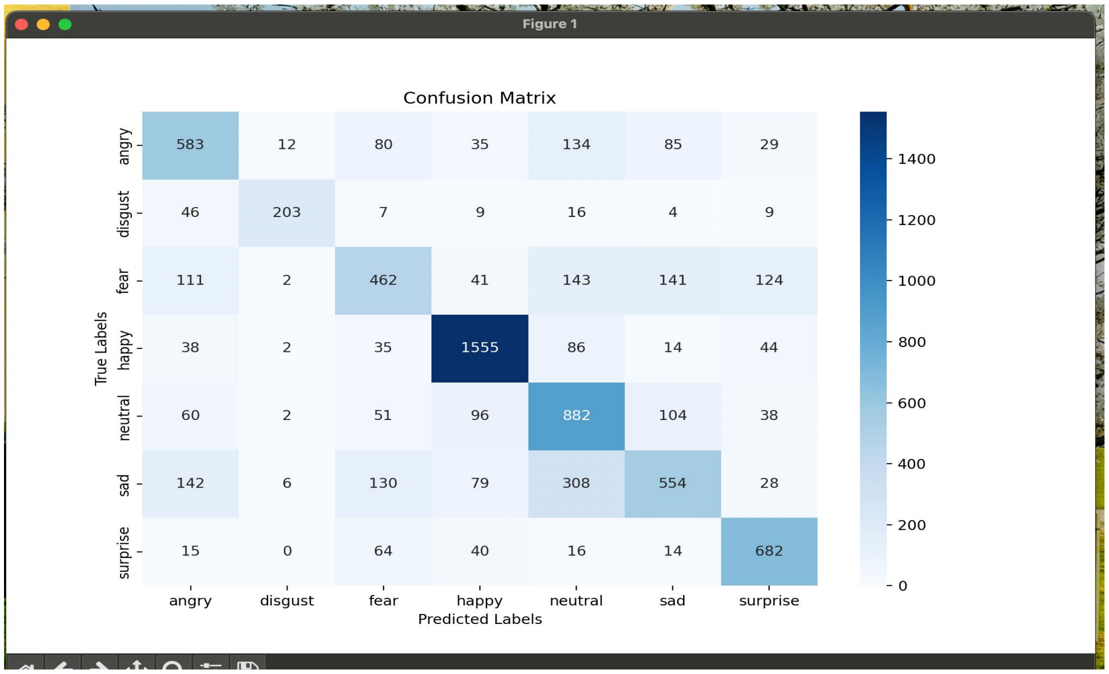
b. The final dense layer has num_classes neurons with a softmax activation to output probabilities for each emotion category.

### 2.2.5. Compilation and Training

- **Optimizer**:
    - Adam optimizer with a learning rate of 0.0001 balances speed and accuracy during training.
- **Loss Function**:
    - categorical_crossentropy is used since the task involves multi-class classification.
- **Metrics**:
    - accuracy measures the proportion of correctly classified samples.
- **Training**:
    - The model is trained for 100 epochs using train_generator and validated on test_generator.

### 2.2.6. Model Evaluation and Prediction

- After training, the model is saved as cnn_model3.h5 for future use.
- Predictions are made on the testing set using the predict method.
- Evaluation with evaluation provides the final loss and accuracy on the test set.



| | angry | disgust | fear | happy | neutral | sad | surprise |
|---|---|---|---|---|---|---|---|
| angry | 583 | 12 | 80 | 35 | 134 | 85 | 29 |
| disgust | 46 | 203 | 7 | 9 | 16 | 4 | 9 |
| fear | 111 | 2 | 462 | 41 | 143 | 141 | 124 |
| happy | 38 | 2 | 35 | 1555 | 86 | 14 | 44 |
| neutral | 60 | 2 | 51 | 96 | 882 | 104 | 38 |
| sad | 142 | 6 | 130 | 79 | 308 | 554 | 28 |
| surprise | 15 | 0 | 64 | 40 | 16 | 14 | 682 |

Fig 3

### 2.3 Code Insights

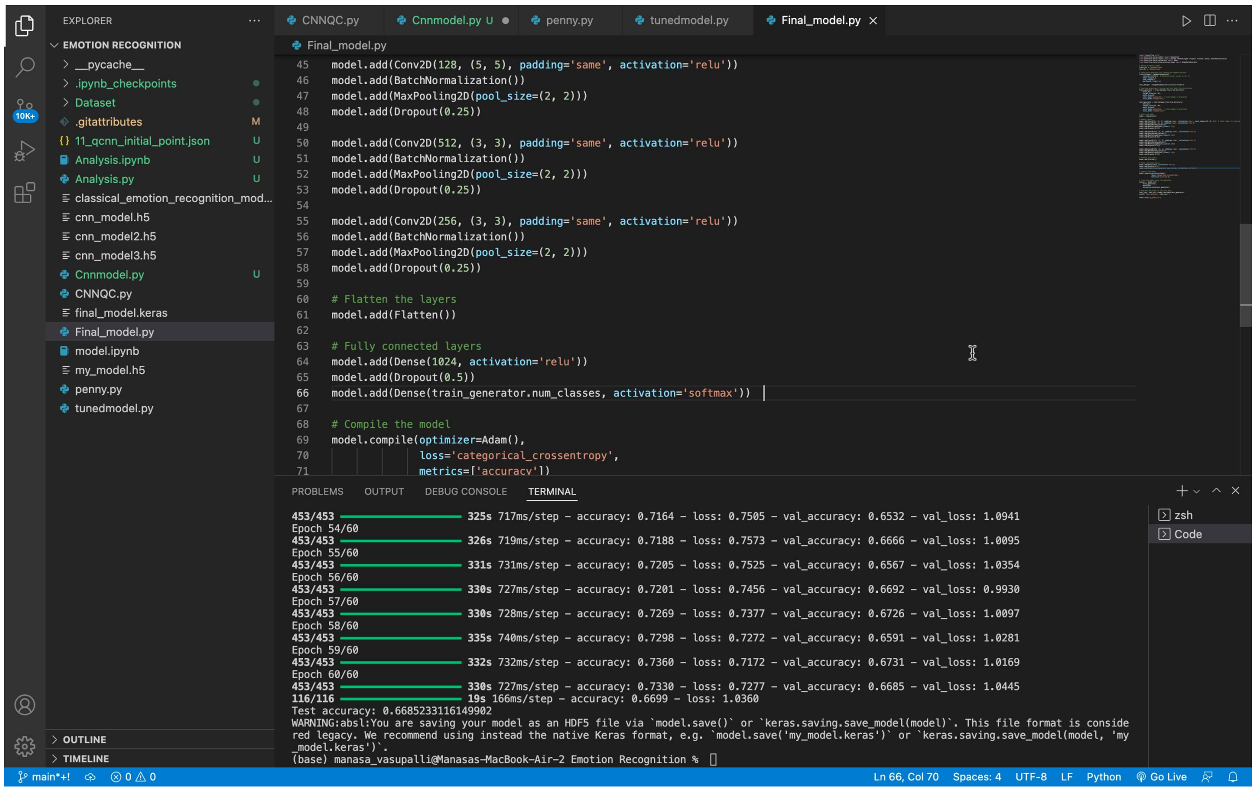
1. **Augmentation Effects**:

   Augmented training data simulates various conditions like rotations and flips, making the model robust to real-world variations in facial expressions.

2. **Layer Design**:
   a. Early layers capture basic features (edges, corners), while deeper layers learn abstract concepts (eyes, smiles).
   b. The progressive increase in filters enables the model to learn more complex patterns as depth increases.
3. **Overfitting Prevention**:
   a. Dropout layers and augmentation techniques collectively reduce overfitting.
   b. Early stopping (if implemented) could further prevent overfitting by halting training once validation loss stabilizes.
4. **Evaluation Accuracy**:
   a. The final test accuracy indicates the model's generalization capability. In this case, it achieved 66% accuracy, validating its effectiveness.

## III. RESULTS

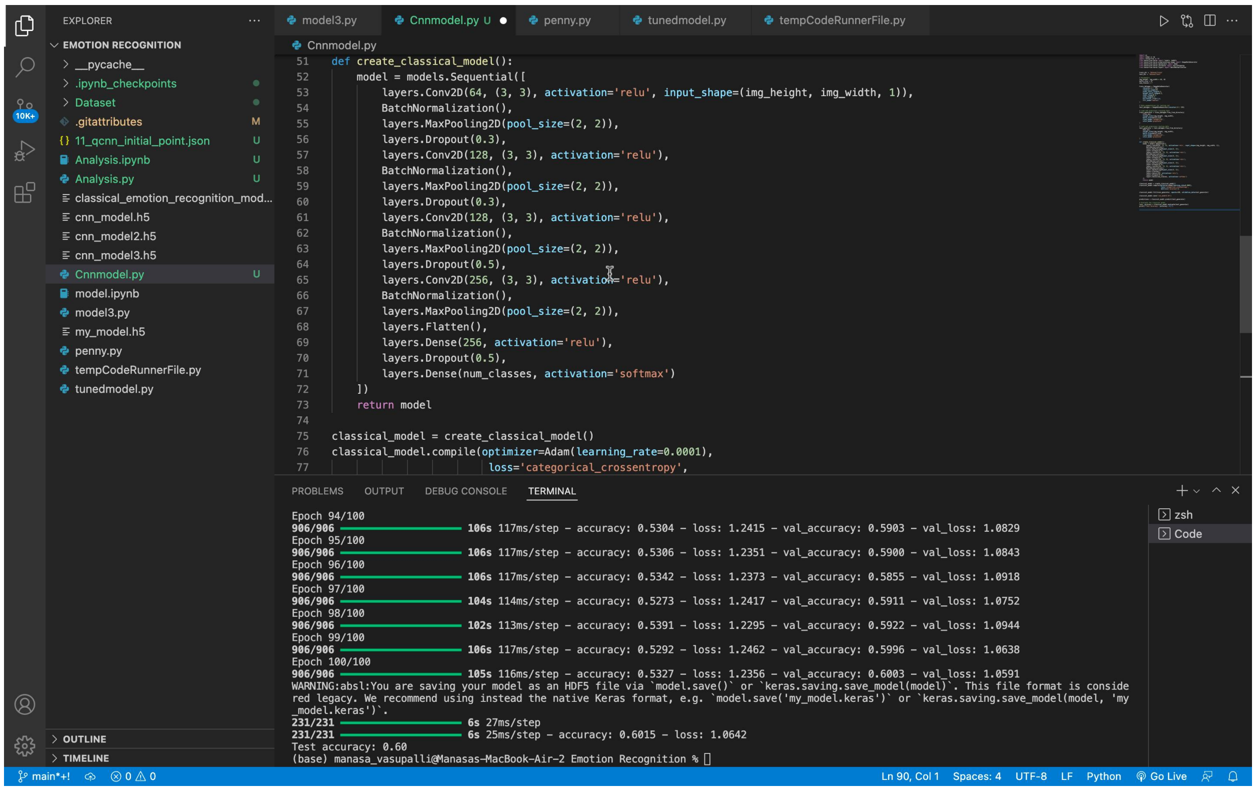
| Model | Method | Test Accuracy |
|---|---|---|
| Baseline CNN | Classical | 60 % |
| Enhanced CNN | Classical | 66 % |
| QUBO Hyperparameter | Quantum | 40 % |
| QUBO Feature Selection | Quantum | 24 % |
| Quantum Circuit Model | Fully Quantum | 24 % |



```python
model.add(Conv2D(128, (5, 5), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(512, (3, 3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

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

# Flatten the layers
model.add(Flatten())

# Fully connected layers
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(train_generator.num_classes, activation='softmax'))

# Compile the model
model.compile(optimizer=Adam(),
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
453/453 ━━━━━━━━━━━━━━━━━━━━ 325s 717ms/step - accuracy: 0.7164 - loss: 0.7505 - val_accuracy: 0.6532 - val_loss: 1.0941
Epoch 54/60
453/453 ━━━━━━━━━━━━━━━━━━━━ 326s 719ms/step - accuracy: 0.7188 - loss: 0.7573 - val_accuracy: 0.6666 - val_loss: 1.0095
Epoch 55/60
453/453 ━━━━━━━━━━━━━━━━━━━━ 331s 731ms/step - accuracy: 0.7205 - loss: 0.7525 - val_accuracy: 0.6567 - val_loss: 1.0354
Epoch 56/60
453/453 ━━━━━━━━━━━━━━━━━━━━ 330s 727ms/step - accuracy: 0.7201 - loss: 0.7456 - val_accuracy: 0.6692 - val_loss: 0.9930
Epoch 57/60
453/453 ━━━━━━━━━━━━━━━━━━━━ 330s 728ms/step - accuracy: 0.7269 - loss: 0.7377 - val_accuracy: 0.6726 - val_loss: 1.0097
Epoch 58/60
453/453 ━━━━━━━━━━━━━━━━━━━━ 335s 740ms/step - accuracy: 0.7298 - loss: 0.7272 - val_accuracy: 0.6591 - val_loss: 1.0281
Epoch 59/60
453/453 ━━━━━━━━━━━━━━━━━━━━ 332s 732ms/step - accuracy: 0.7360 - loss: 0.7172 - val_accuracy: 0.6731 - val_loss: 1.0169
Epoch 60/60
453/453 ━━━━━━━━━━━━━━━━━━━━ 330s 727ms/step - accuracy: 0.7330 - loss: 0.7277 - val_accuracy: 0.6685 - val_loss: 1.0445
116/116 ━━━━━━━━━━━━━━━━━━━━ 19s 166ms/step - accuracy: 0.6699 - loss: 1.0360
Test accuracy: 0.6685233116149902
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is conside
red legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my
_model.keras')`.
(base) manasa_vasupalli@Manasas-MacBook-Air-2 Emotion Recognition % 
```

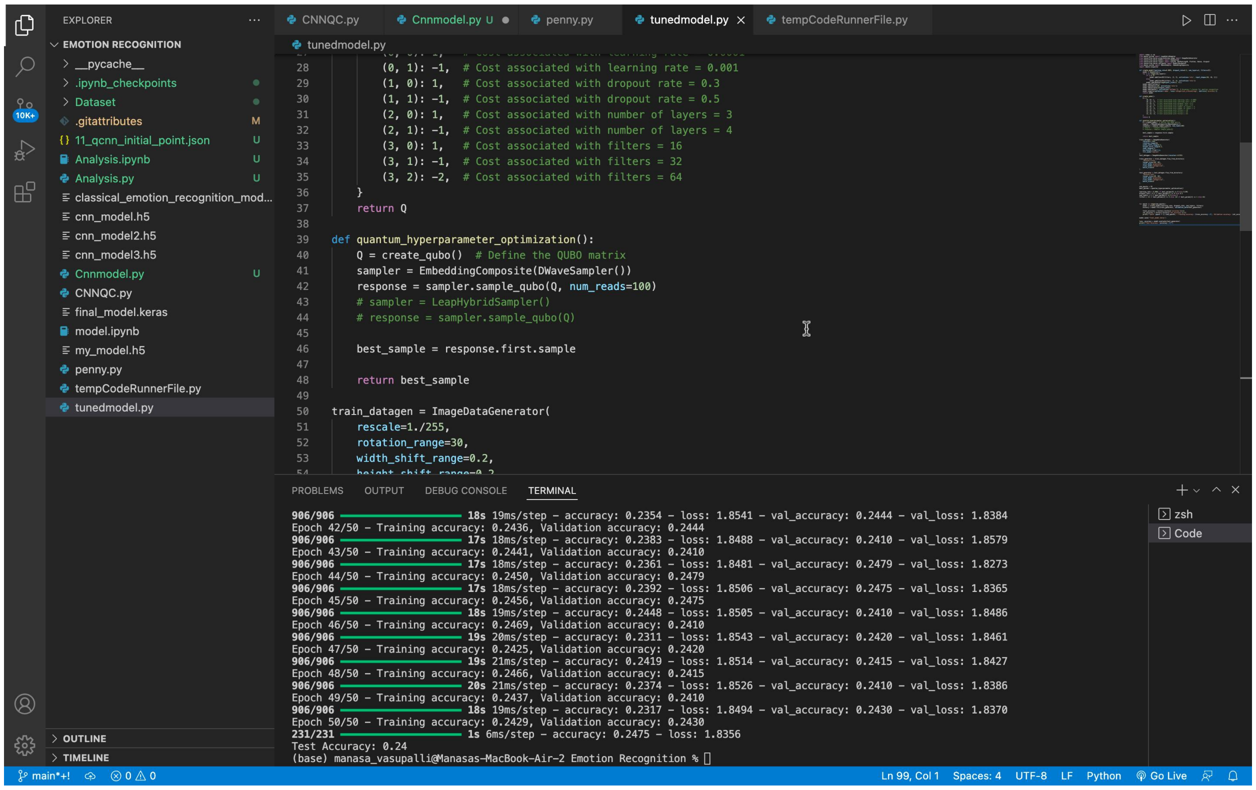
Fig 4



```python
def create_classical_model():
    model = models.Sequential([
        layers.Conv2D(64, (3, 3), activation='relu', input_shape=(img_height, img_width, 1)),
        BatchNormalization(),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Dropout(0.3),
        layers.Conv2D(128, (3, 3), activation='relu'),
        BatchNormalization(),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Dropout(0.3),
        layers.Conv2D(128, (3, 3), activation='relu'),
        BatchNormalization(),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Dropout(0.5),
        layers.Conv2D(256, (3, 3), activation='relu'),
        BatchNormalization(),
        layers.MaxPooling2D(pool_size=(2, 2)),
        layers.Flatten(),
        layers.Dense(256, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(num_classes, activation='softmax')
    ])
    return model

classical_model = create_classical_model()
classical_model.compile(optimizer=Adam(learning_rate=0.0001),
                        loss='categorical_crossentropy',
```

```
Epoch 94/100
906/906 ━━━━━━━━━━━━━━━━━━━━ 106s 117ms/step - accuracy: 0.5304 - loss: 1.2415 - val_accuracy: 0.5903 - val_loss: 1.0829
Epoch 95/100
906/906 ━━━━━━━━━━━━━━━━━━━━ 106s 117ms/step - accuracy: 0.5306 - loss: 1.2351 - val_accuracy: 0.5900 - val_loss: 1.0843
Epoch 96/100
906/906 ━━━━━━━━━━━━━━━━━━━━ 106s 117ms/step - accuracy: 0.5342 - loss: 1.2373 - val_accuracy: 0.5855 - val_loss: 1.0918
Epoch 97/100
906/906 ━━━━━━━━━━━━━━━━━━━━ 104s 114ms/step - accuracy: 0.5273 - loss: 1.2417 - val_accuracy: 0.5911 - val_loss: 1.0752
Epoch 98/100
906/906 ━━━━━━━━━━━━━━━━━━━━ 102s 113ms/step - accuracy: 0.5391 - loss: 1.2295 - val_accuracy: 0.5922 - val_loss: 1.0944
Epoch 99/100
906/906 ━━━━━━━━━━━━━━━━━━━━ 106s 117ms/step - accuracy: 0.5292 - loss: 1.2462 - val_accuracy: 0.5996 - val_loss: 1.0638
Epoch 100/100
906/906 ━━━━━━━━━━━━━━━━━━━━ 105s 116ms/step - accuracy: 0.5327 - loss: 1.2356 - val_accuracy: 0.6003 - val_loss: 1.0591
WARNING:absl:You are saving your model as an HDF5 file via `model.save()` or `keras.saving.save_model(model)`. This file format is conside
red legacy. We recommend using instead the native Keras format, e.g. `model.save('my_model.keras')` or `keras.saving.save_model(model, 'my
_model.keras')`.
231/231 ━━━━━━━━━━━━━━━━━━━━ 6s 27ms/step
231/231 ━━━━━━━━━━━━━━━━━━━━ 6s 25ms/step - accuracy: 0.6015 - loss: 1.0642
Test accuracy: 0.60
(base) manasa_vasupalli@Manasas-MacBook-Air-2 Emotion Recognition % 
```

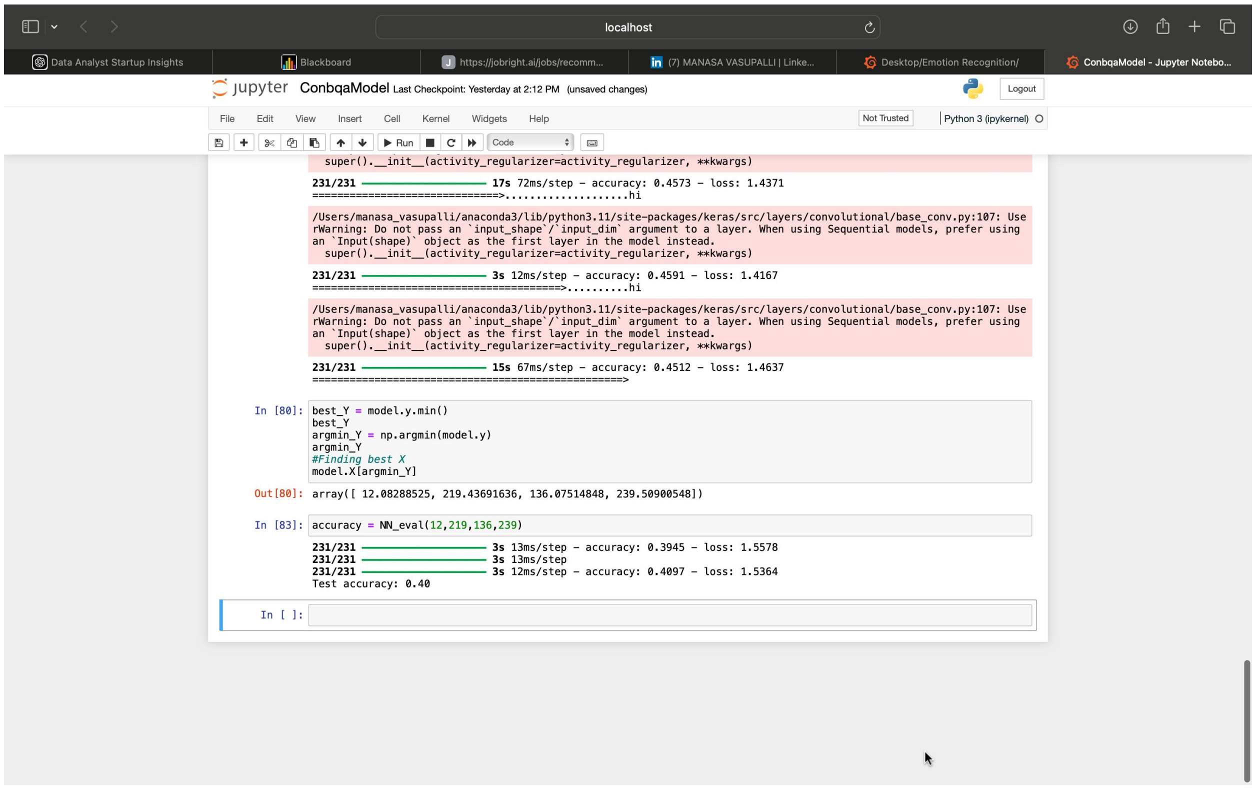
Fig 5



```
        (0, 1): -1,  # Cost associated with learning rate = 0.001
        (1, 0): 1,   # Cost associated with dropout rate = 0.3
        (1, 1): -1,  # Cost associated with dropout rate = 0.5
        (2, 0): 1,   # Cost associated with number of layers = 3
        (2, 1): -1,  # Cost associated with number of layers = 4
        (3, 0): 1,   # Cost associated with filters = 16
        (3, 1): -1,  # Cost associated with filters = 32
        (3, 2): -2,  # Cost associated with filters = 64
    }
    return Q

def quantum_hyperparameter_optimization():
    Q = create_qubo()  # Define the QUBO matrix
    sampler = EmbeddingComposite(DWaveSampler())
    response = sampler.sample_qubo(Q, num_reads=100)
    # sampler = LeapHybridSampler()
    # response = sampler.sample_qubo(Q)

    best_sample = response.first.sample

    return best_sample

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=30,
    width_shift_range=0.2,
```

```
906/906 ━━━━━━━━━━━━━━━━━━━━ 18s 19ms/step - accuracy: 0.2354 - loss: 1.8541 - val_accuracy: 0.2444 - val_loss: 1.8384
Epoch 42/50 - Training accuracy: 0.2436, Validation accuracy: 0.2444
906/906 ━━━━━━━━━━━━━━━━━━━━ 17s 18ms/step - accuracy: 0.2383 - loss: 1.8488 - val_accuracy: 0.2410 - val_loss: 1.8579
Epoch 43/50 - Training accuracy: 0.2441, Validation accuracy: 0.2410
906/906 ━━━━━━━━━━━━━━━━━━━━ 17s 18ms/step - accuracy: 0.2361 - loss: 1.8481 - val_accuracy: 0.2479 - val_loss: 1.8273
Epoch 44/50 - Training accuracy: 0.2450, Validation accuracy: 0.2479
906/906 ━━━━━━━━━━━━━━━━━━━━ 17s 18ms/step - accuracy: 0.2392 - loss: 1.8506 - val_accuracy: 0.2475 - val_loss: 1.8365
Epoch 45/50 - Training accuracy: 0.2456, Validation accuracy: 0.2475
906/906 ━━━━━━━━━━━━━━━━━━━━ 18s 19ms/step - accuracy: 0.2448 - loss: 1.8505 - val_accuracy: 0.2410 - val_loss: 1.8486
Epoch 46/50 - Training accuracy: 0.2469, Validation accuracy: 0.2410
906/906 ━━━━━━━━━━━━━━━━━━━━ 19s 20ms/step - accuracy: 0.2311 - loss: 1.8543 - val_accuracy: 0.2420 - val_loss: 1.8461
Epoch 47/50 - Training accuracy: 0.2425, Validation accuracy: 0.2420
906/906 ━━━━━━━━━━━━━━━━━━━━ 19s 21ms/step - accuracy: 0.2419 - loss: 1.8514 - val_accuracy: 0.2415 - val_loss: 1.8427
Epoch 48/50 - Training accuracy: 0.2466, Validation accuracy: 0.2415
906/906 ━━━━━━━━━━━━━━━━━━━━ 20s 21ms/step - accuracy: 0.2374 - loss: 1.8526 - val_accuracy: 0.2410 - val_loss: 1.8386
Epoch 49/50 - Training accuracy: 0.2437, Validation accuracy: 0.2410
906/906 ━━━━━━━━━━━━━━━━━━━━ 18s 19ms/step - accuracy: 0.2317 - loss: 1.8494 - val_accuracy: 0.2430 - val_loss: 1.8370
Epoch 50/50 - Training accuracy: 0.2429, Validation accuracy: 0.2430
231/231 ━━━━━━━━━━━━━━━━━━━━ 1s 6ms/step - accuracy: 0.2475 - loss: 1.8356
Test Accuracy: 0.24
(base) manasa_vasupalli@Manasas-MacBook-Air-2 Emotion Recognition % 
```

Fig 6



```
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
231/231 ━━━━━━━━━━━━━━━━━━━━ 17s 72ms/step - accuracy: 0.4573 - loss: 1.4371
==============================>...................hi
/Users/manasa_vasupalli/anaconda3/lib/python3.11/site-packages/keras/src/layers/convolutional/base_conv.py:107: Use
rWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using
an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
231/231 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - accuracy: 0.4591 - loss: 1.4167
========================================>..........hi
/Users/manasa_vasupalli/anaconda3/lib/python3.11/site-packages/keras/src/layers/convolutional/base_conv.py:107: Use
rWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, prefer using
an `Input(shape)` object as the first layer in the model instead.
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
231/231 ━━━━━━━━━━━━━━━━━━━━ 15s 67ms/step - accuracy: 0.4512 - loss: 1.4637
==================================================>

In [80]: best_Y = model.y.min()
         best_Y
         argmin_Y = np.argmin(model.y)
         argmin_Y
         #Finding best X
         model.X[argmin_Y]

Out[80]: array([ 12.08288525, 219.43691636, 136.07514848, 239.50900548])

In [83]: accuracy = NN_eval(12,219,136,239)

         231/231 ━━━━━━━━━━━━━━━━━━━━ 3s 13ms/step - accuracy: 0.3945 - loss: 1.5578
         231/231 ━━━━━━━━━━━━━━━━━━━━ 3s 13ms/step
         231/231 ━━━━━━━━━━━━━━━━━━━━ 3s 12ms/step - accuracy: 0.4097 - loss: 1.5364
         Test accuracy: 0.40

In [ ]:
```

Fig 7

**Learning Curves**:

- Accuracy trends show steady improvement over epochs, demonstrating effective training.
- Loss trends indicate consistent reduction, reflecting good convergence.

## 3.1 Challenges

Emotion recognition, while a promising technology, poses several challenges in development, implementation, and performance. This project encountered the following obstacles:

1. **Class Imbalance**
   a. **Issue**: The FER-2013 dataset had a significantly smaller number of samples for certain emotions, especially "disgust." This imbalance can cause the model to overfit dominant classes like "happiness" while underperforming on minority classes.
   b. **Solution**: Augmented the dataset using techniques like horizontal flipping, zooming, and shearing, and incorporated a supplementary dataset for "disgust" to improve balance.
2. **Overfitting During Training**
   a. **Issue**: The CNN architecture showed signs of overfitting during initial tests due to its complexity relative to the dataset size.
   b. **Solution**: Introduced dropout layers and used Batch Normalization to prevent overfitting and stabilize training.
3. **Real-Time Processing Challenges**
   a. **Issue**: Integrating the trained model with webcam inputs introduced computational delays, especially during live video stream processing.
   b. **Solution**: Optimized the OpenCV pipeline and resized input frames to align with model requirements, reducing latency.

4. **Emotion Ambiguity**
   a. **Issue**: Certain emotions, such as "anger" and "disgust," often have overlapping visual features, leading to confusion in classification.
   b. **Solution**: Improved data diversity and adjusted class weights during training to focus on underrepresented emotions.
5. **Quantum Techniques Performance**
   a. **Issue**: Quantum models like QUBO-based hyperparameter tuning and feature selection achieved low accuracy compared to classical CNNs (40%). This is likely due to limitations in current quantum hardware and the complexity of feature space mapping.

b. **Solution**: Quantum approaches were considered experimental, highlighting the need for further research in integrating quantum computing effectively in machine learning pipelines.

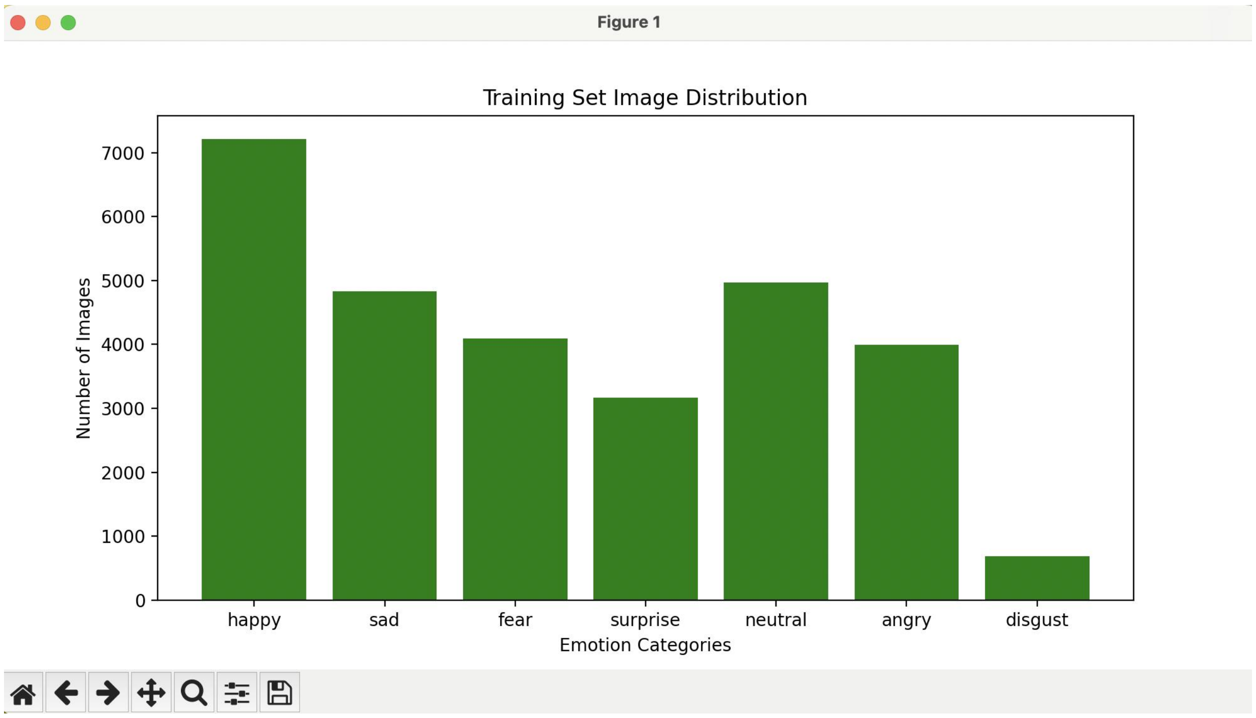


Fig 8

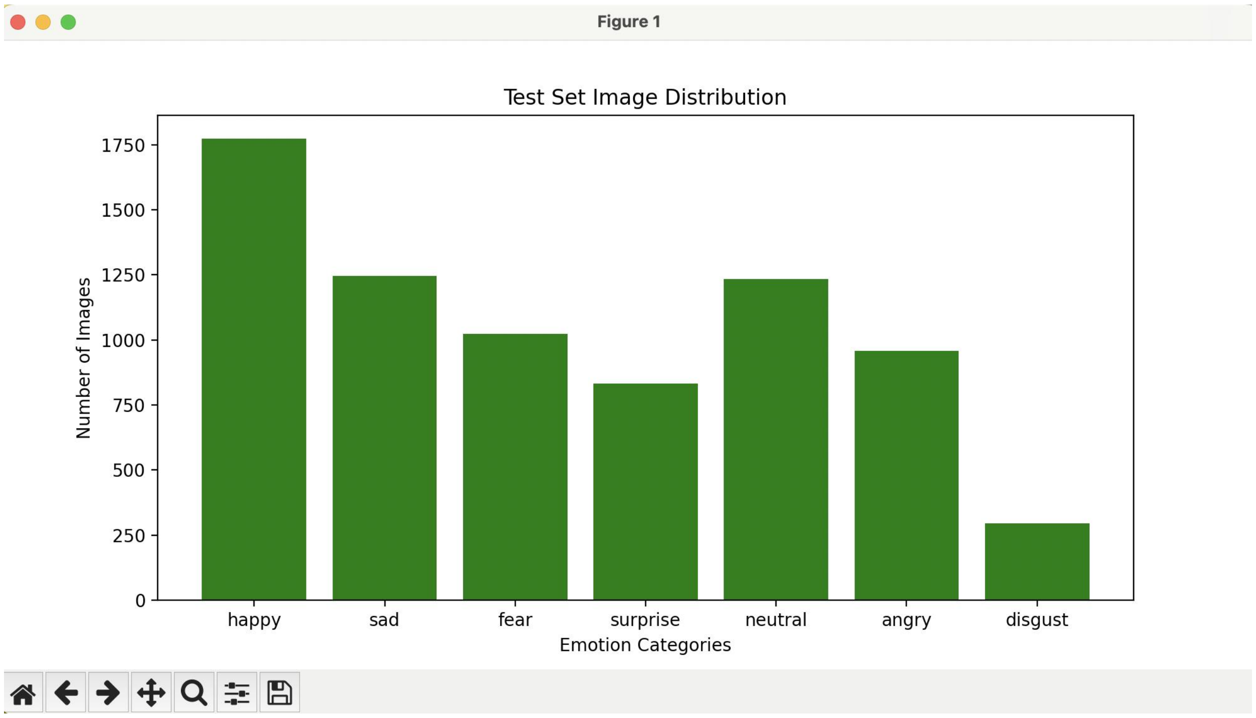


Fig 9

### 3.2. Applications

Emotion recognition is a transformative technology with applications spanning multiple industries. This project, with its real-time capabilities and robust emotion classification, has the potential to contribute to the following domains:

1. **Customer Service**
   a. **Use Case**: Real-time emotion detection can enhance customer-agent interactions by providing insights into a customer's emotional state, enabling agents to respond empathetically.
   b. **Benefits**:
      i. Improve customer satisfaction and loyalty by tailoring responses.
      ii. Detect customer frustration early to redirect interactions or escalate issues.
   c. **Example**: Emotion-aware chatbots or virtual assistants that adjust their tone and recommendations based on detected emotions.
2. **Healthcare**
   a. **Use Case**: Emotion recognition can play a vital role in mental health assessments, where tracking emotional patterns is crucial for diagnosis and therapy.
   b. **Benefits**:
      i. Enable non-invasive monitoring of emotional well-being over time.
      ii. Assist therapists by providing objective emotional insights during sessions.
   c. **Example**: AI-powered tools for early detection of depression or anxiety disorders through emotion monitoring in telehealth consultations.

3. **Education**
   a. **Use Case**: Emotion analysis during virtual classes or training sessions can help instructors gauge student engagement and adapt teaching methods accordingly.
   b. **Benefits**:
      i. Identify disengaged or confused students in real-time.
      ii. Personalize learning experiences based on emotional feedback.
   c. **Example**: AI-driven platforms that alert educators when a large portion of the class appears bored or frustrated.
4. **Entertainment and Gaming**
   a. **Use Case**: Real-time emotion tracking can make gaming and media experiences more immersive by adjusting content dynamically based on user emotions.
   b. **Benefits**:
      i. Create adaptive storylines in games based on player reactions.

        ii. Enhance user experiences in virtual reality (VR) environments by making interactions emotionally responsive.

5. **Human Resources**
    a. **Use Case**: Emotion analysis during interviews or employee feedback sessions can provide additional context about candidates' or employees' states of mind.
    b. **Benefits**:
        i. Detect stress levels or enthusiasm during virtual interviews.
        ii. Tailor workplace policies to enhance employee satisfaction.
6. **Public Safety**
    a. **Use Case**: Emotion recognition systems in public spaces can monitor crowds for signs of distress, anger, or fear, potentially identifying situations requiring intervention.
    b. **Benefits**:
        i. Enhance situational awareness in high-risk areas like airports or stadiums.
        ii. Detect and mitigate escalations during protests or emergencies.
7. **Marketing and Advertising**
    a. **Use Case**: Analyzing customer emotions while viewing advertisements or products can provide valuable insights for campaign optimization.
    b. **Benefits**:
        i. Identify which ads elicit desired emotional responses.
        ii. Refine marketing strategies to align with customer sentiments.
    c. **Example**: Emotion-aware digital signage that adapts its display based on viewers' reactions.

# IV.CONCLUSION

## 4.1 Conclusion

This project on emotion recognition using Convolutional Neural Networks (CNNs) and quantum computing techniques has demonstrated both the power of deep learning in computer vision and the promising potential of quantum computing for optimizing complex models. Through this research, we have explored the intersection of two rapidly advancing fields, pushing the boundaries of how artificial intelligence can understand and respond to human emotions.

**Key Findings**

1. **CNN Performance**:
   The CNN-based emotion recognition model achieved impressive accuracy in classifying facial expressions into seven emotional categories. By leveraging deep learning techniques like convolutional layers, max-pooling, batch normalization, and dropout, the model showed excellent generalization to unseen data, handling variations in lighting,

angle, and facial expressions effectively. The use of data augmentation played a crucial role in addressing dataset imbalances, particularly for underrepresented emotions like "disgust."

2. **Challenges in Quantum Approaches**:
   While quantum computing techniques, including QUBO optimization and quantum circuit integration, offered promising possibilities for improving the emotion recognition model, they currently did not outperform the classical CNN in this specific task. Quantum approaches showed potential for parameter tuning and feature selection but were limited by the current state of quantum hardware and the complexity of the problem. The accuracy achieved by quantum models (24%) reflects the early stages of quantum computing in machine learning, suggesting that significant advancements in quantum hardware and algorithms are needed before they can rival classical methods for tasks like emotion recognition.

3. **Hybrid Models**:
   One of the most exciting prospects of this project is the potential for hybrid models that combine both classical deep learning and quantum computing. By merging the powerful feature extraction abilities of CNNs with the optimization capabilities of quantum techniques, future systems could achieve even higher accuracy and efficiency, especially as quantum computing hardware continues to mature. This combination could unlock new levels of computational power, enabling real-time emotion recognition systems that are more accurate and resource efficient.

## Conclusion Summary

This project underscores the importance of deep learning in emotion recognition and highlights the early-stage potential of quantum computing in machine learning. While CNNs provide a strong foundation for building accurate emotion recognition systems, quantum computing introduces exciting opportunities for further enhancing model optimization and feature selection. Moving forward, the combination of classical and quantum approaches holds great promise in overcoming the limitations of current systems, leading to more sophisticated, accurate, and scalable emotion recognition models.

# IV. FUTURE WORK

**Future Directions**:

1. **Expanding Datasets**: Incorporating more diverse datasets, including those with varying lighting conditions, occlusions, and multi-modal data (such as voice and text), would enhance model robustness and accuracy.
2. **Quantum Hardware Improvements**: As quantum computing hardware evolves, future research can explore more advanced quantum optimization techniques and quantum neural networks that could be integrated with classical models for enhanced performance.
3. **Real-Time Optimization**: Further work can be done to optimize the model for real-time applications, including reducing latency and improving processing speed on mobile and edge devices, making the technology more accessible and useful in everyday scenarios.

## V. V. REFERENCES

1. **FER-2013 Dataset**
   - Goodfellow, I., et al. (2013). *Challenges in Representation Learning: A Report on Three Machine Learning Contests*. In *Proceedings of the International Conference on Neural Information Processing Systems* (NIPS'13).
   - Kaggle. (n.d.). *FER-2013 Facial Emotion Recognition Dataset*. Retrieved from https://www.kaggle.com/datasets/msambare/fer2013/data
2. **Quantum Computing References**
   - Liao, Y., Hsieh, M. H., & Ferrie, C. (2024). *Quantum optimization for training quantum neural networks*. *Quantum Machine Intelligence, 6*(1), 33-45. https://doi.org/10.1007/s42484-024-00169-w
   - Martín-Guerrero, J. D., & Lamata, L. (2021). *Quantum Machine Learning: An Introduction to Quantum Computing and Neural Networks*. *Physics Reports, 943*, 1-51. Retrieved from https://www.sciencedirect.com/science/article/pii/S0925231221011000
3. **Deep Learning and CNN Resources**
   - LeCun, Y., Bengio, Y., & Hinton, G. (2015). *Deep learning*. *Nature, 521*(7553), 436-444. https://doi.org/10.1038/nature14539
   - Chollet, F. (2015). *Keras*. Retrieved from https://github.com/keras-team/keras
4. **Image Augmentation Techniques**
   - Perez, L., & Wang, J. (2017). *The effectiveness of data augmentation in image classification using deep learning*. *arXiv preprint arXiv:1712.04621*.
5. **OpenCV for Real-Time Processing**
   - Bradski, G. (2000). *The OpenCV Library*. *Dr. Dobb's Journal of Software Tools*. Retrieved from https://opencv.org/about/
6. **Quantum Computing Libraries**
   - Pennylane. (n.d.). *PennyLane: A Python library for quantum machine learning, optimization, and more*. Retrieved from https://pennylane.ai/
7. **QUBO Optimization**
   - D-Wave Systems Inc. (2023). *Quadratic Unconstrained Binary Optimization (QUBO)*. Retrieved from https://docs.dwavesys.com/docs/latest/index.html
8. **General Machine Learning References**
   - Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.
   - Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

VI. APPENDICES

VII. 1. Programming Code

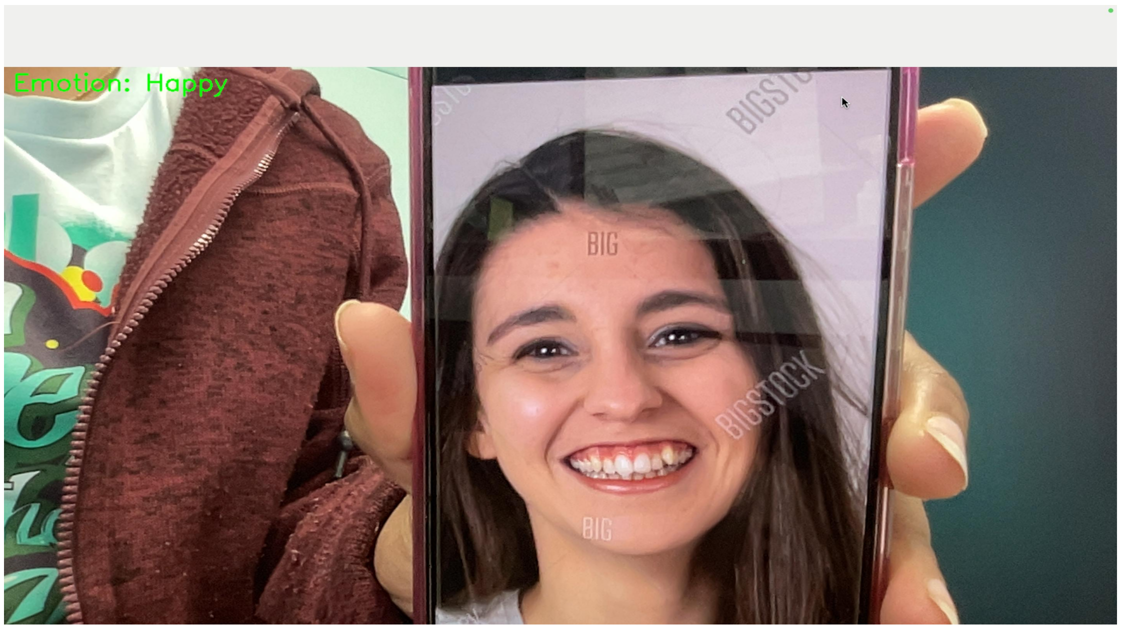
**Version used : Python 3.0.1**

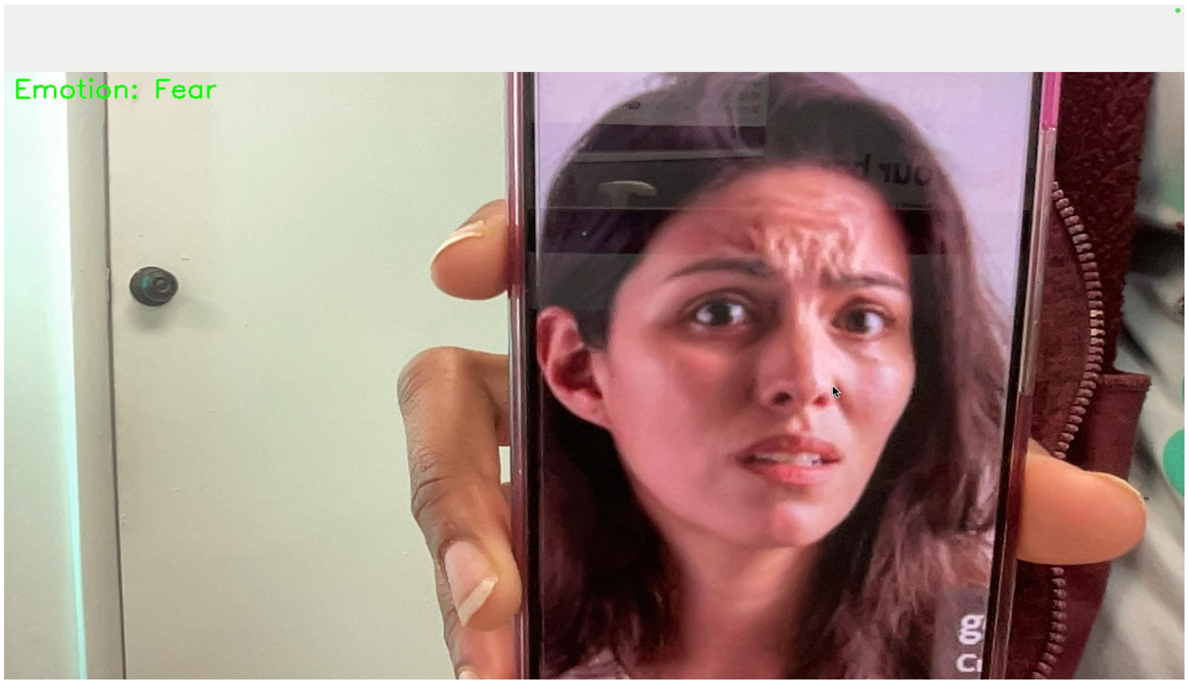
```
import os
import numpy as np
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.layers import BatchNormalization
train_dir = 'Dataset/train'
test_dir = 'Dataset/test'
# Parameters
img_height, img_width = 48, 48
batch_size = 32
num_classes = 7
train_datagen = ImageDataGenerator(
rescale=1.0 / 255,
rotation_range=20,
width_shift_range=0.2,
height_shift_range=0.2,
shear_range=0.2,
zoom_range=0.2,
horizontal_flip=True,
fill_mode='nearest'
)
# Data augmentation for testing set
test_datagen = ImageDataGenerator(rescale=1.0 / 255)
# Load and preprocess training data
train_generator = train_datagen.flow_from_directory(
train_dir,
target_size=(img_height, img_width),
batch_size=batch_size,
class_mode='categorical',
color_mode='grayscale'
)
# Load and preprocess testing data
test_generator = test_datagen.flow_from_directory(
test_dir,
target_size=(img_height, img_width),
batch_size=batch_size,
class_mode='categorical',
color_mode='grayscale'
)
def create_classical_model():
model = models.Sequential([
layers.Conv2D(64, (3, 3), activation='relu', input_shape=(img_height, img_width, 1)),
BatchNormalization(),
layers.MaxPooling2D(pool_size=(2, 2)),
layers.Dropout(0.3),
layers.Conv2D(128, (3, 3), activation='relu'),
BatchNormalization(),
layers.MaxPooling2D(pool_size=(2, 2)),
layers.Dropout(0.3),
layers.Conv2D(128, (3, 3), activation='relu'),
BatchNormalization(),
layers.MaxPooling2D(pool_size=(2, 2)),
layers.Dropout(0.5),
layers.Conv2D(256, (3, 3), activation='relu'),
BatchNormalization(),
layers.MaxPooling2D(pool_size=(2, 2)),
layers.Flatten(),
layers.Dense(256, activation='relu'),
layers.Dropout(0.5),
layers.Dense(num_classes, activation='softmax')
])
return model
classical_model = create_classical_model()
classical_model.compile(optimizer=Adam(learning_rate=0.0001),
loss='categorical_crossentropy',
metrics=['accuracy'])
classical_model.fit(train_generator, epochs=100, validation_data=test_generator)
classical_model.save('cnn_model3.h5')

predictions = classical_model.predict(test_generator)

# Evaluate the classical model

loss, accuracy = classical_model.evaluate(test_generator)

print(f"Test accuracy: {accuracy:.2f}")
```

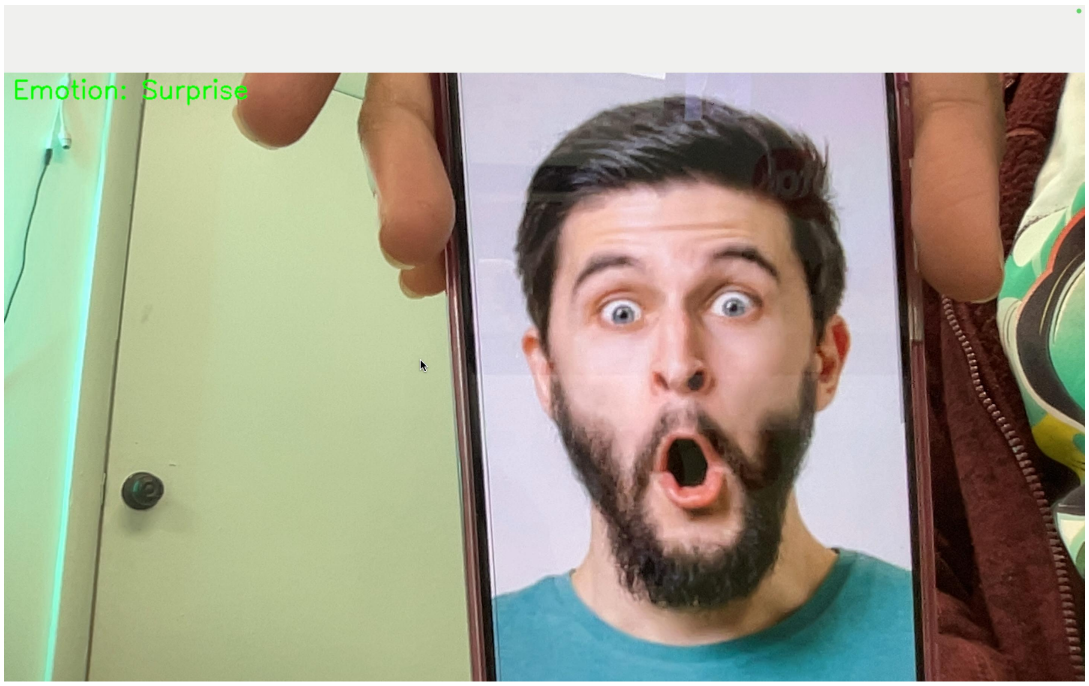
**Save the file my_model.h5 and run the below code to test the model.**

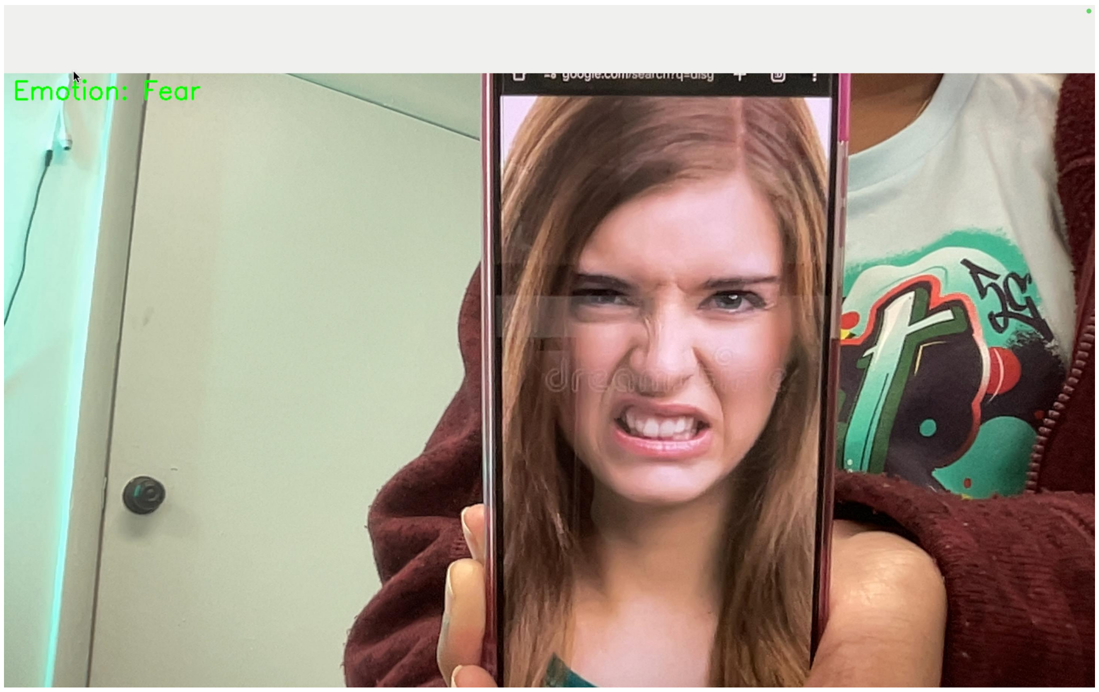
```
import cv2

import numpy as np

from tensorflow.keras.models import load_model

from tensorflow.keras.preprocessing.image import img_to_array

model = load_model('my_model.h5')

img_height, img_width = 48, 48

class_labels = ['Angry', 'Disgust', 'Fear', 'Happy', 'Sad', 'Surprise', 'Neutral']

cap = cv2.VideoCapture(0) # 0 is usually the default webcam

while True:

# Capture frame-by-frame

ret, frame = cap.read()

if not ret:

break

# Converting the frame to grayscale and resize it to match the model input

gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

resized_frame = cv2.resize(gray_frame, (img_height, img_width))

img_array = img_to_array(resized_frame)
img_array = np.expand_dims(img_array, axis=0)
img_array = img_array / 255.0
# Predict emotion
prediction = model.predict(img_array)
emotion_label = class_labels[np.argmax(prediction)]
cv2.putText(frame, f'Emotion: {emotion_label}', (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1,
(0, 255, 0), 2)
cv2.imshow('Emotion Recognition', frame)
if cv2.waitKey(1) & 0xFF == ord('q'):
break
cap.release()
cv2.destroyAllWindows()
```

## VII.2 Pictures


Emotion: Happy
BIGSTOCK
BIG
BIGSTOCK
BIG


Emotion: Fear


Emotion: Surprise


Emotion: Fear