LITERATURE REVIEW

Facial emotion detection, also known as facial expression recognition, is a field within computer vision and affective computing that aims to identify human emotions through facial expressions. This technology has applications in various areas, including psychology, human-computer interaction, security, marketing, and healthcare.

Key Areas in Facial Emotion Detection

Feature Extraction: Techniques to extract relevant features from facial images, such as geometric features (e.g., distances between facial landmarks) and appearance features (e.g., pixel values, texture).

Common methods: Active Shape Models (ASM), Active Appearance Models (AAM), and Convolutional Neural Networks (CNNs).

Classification Methods: Machine learning algorithms used to classify emotions based on extracted features. Traditional methods: Support Vector Machines (SVM), k-Nearest Neighbors (k-NN), and Random Forests. Deep learning methods: CNNs, Recurrent Neural Networks (RNNs), and Long Short-Term Memory networks (LSTMs).

Datasets: Publicly available datasets used to train and evaluate facial emotion detection models.

Popular datasets: CK+ (Extended Cohn-Kanade), FER-2013, JAFFE (Japanese Female Facial Expression), and AffectNet.

Applications: Real-world applications of facial emotion detection technology.

Examples: Automated customer service, mental health monitoring, driver fatigue detection, and adaptive learning systems.

Notable Works and Papers

1. **"Facial Expression Recognition: A Survey" by S. Li and W. Deng (2018)**:
   - A comprehensive survey on facial expression recognition, covering feature extraction, classification methods, and datasets.
2. **"Deep Emotion: Facial Expression Recognition Using Attentional Convolutional Network" by M. Mollahosseini, D. Chan, and M. H. Mahoor (2016)**:

    - Introduces a deep learning approach using attentional mechanisms to improve the accuracy of facial emotion detection.

3. **"Learning Deep Features for Facial Expression Recognition with CNN" by B. Barsoum, C. Zhang, C. C. Ferrer, and Z. Zhang (2016)**:
    - Explores the use of deep convolutional neural networks for facial expression recognition, achieving state-of-the-art performance.
4. **"FER-2013 Face Database" by I. Goodfellow, D. Erhan, P. Luc Carrier, A. Courville, and Y. Bengio (2013)**:
    - Introduces the FER-2013 dataset, which has become a standard benchmark for evaluating facial expression recognition algorithms.

Current Trends and Future Directions

1. **Multimodal Emotion Recognition**:
    - Combining facial expressions with other modalities, such as voice and physiological signals, for more robust emotion detection.
2. **Real-Time Emotion Detection**:
    - Developing efficient algorithms that can detect emotions in real-time, suitable for applications like virtual reality and interactive systems.
3. **Context-Aware Emotion Detection**:
    - Incorporating contextual information (e.g., surroundings, situational context) to improve the accuracy and relevance of emotion detection.
4. **Ethical Considerations**:
    - Addressing privacy concerns and ethical implications of using facial emotion detection technology, particularly in surveillance and marketing.

Dataset Information

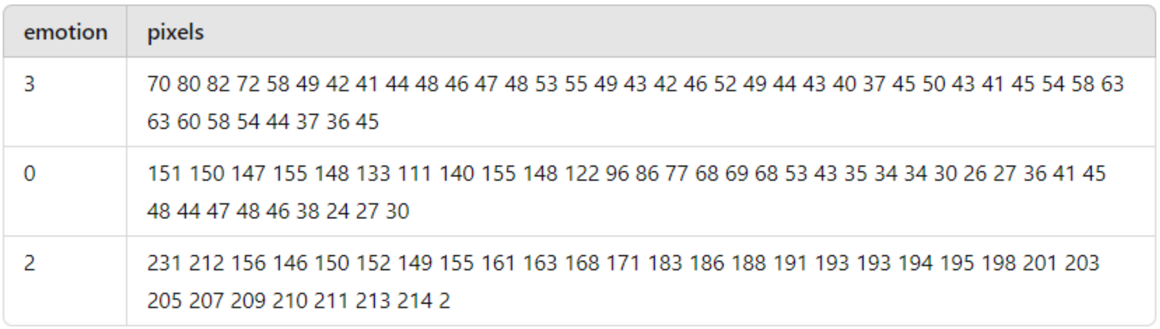
The FER-2013 dataset was introduced as part of the Kaggle competition "Challenges in Representation Learning: Facial Expression Recognition Challenge." It contains grayscale images of faces, with each image labeled as one of seven emotions.

Data Features and Properties

1. **Image Properties**:
   - **Resolution**: All images are 48x48 pixels.
   - **Color Space**: Grayscale.
2. **Emotions (Classes)**:
   - The dataset includes images labeled with one of seven emotions:
     1. **Angry** (label: 0)
     2. **Disgust** (label: 1)
     3. **Fear** (label: 2)
     4. **Happy** (label: 3)
     5. **Neutral** (label: 4)
     6. **Sad** (label: 5)
     7. **Surprise** (label: 6)
3. **Data Split**:
   - **Training Set**: 58,454 images.
   - **Validation Set**: 7,066
4. **Format**:
   - The images are stored in a CSV file, where each row corresponds to an image.
   - The columns include the pixel values (flattened into a single row) and the emotion label.

Example of the Data Format:

The CSV file format typically looks like this:



emotion	pixels
3	70 80 82 72 58 49 42 41 44 48 46 47 48 53 55 49 43 42 46 52 49 44 43 40 37 45 50 43 41 45 54 58 63 63 60 58 54 44 37 36 45
0	151 150 147 155 148 133 111 140 155 148 122 96 86 77 68 69 68 53 43 35 34 34 30 26 27 36 41 45 48 44 47 48 46 38 24 27 30
2	231 212 156 146 150 152 149 155 161 163 168 171 183 186 188 191 193 193 194 195 198 201 203 205 207 209 210 211 213 214 2

- **emotion**: The label of the emotion (integer between 0 and 6).
- **pixels**: A space-separated string of 2,304 grayscale pixel values (48x48).

Applications and Usage

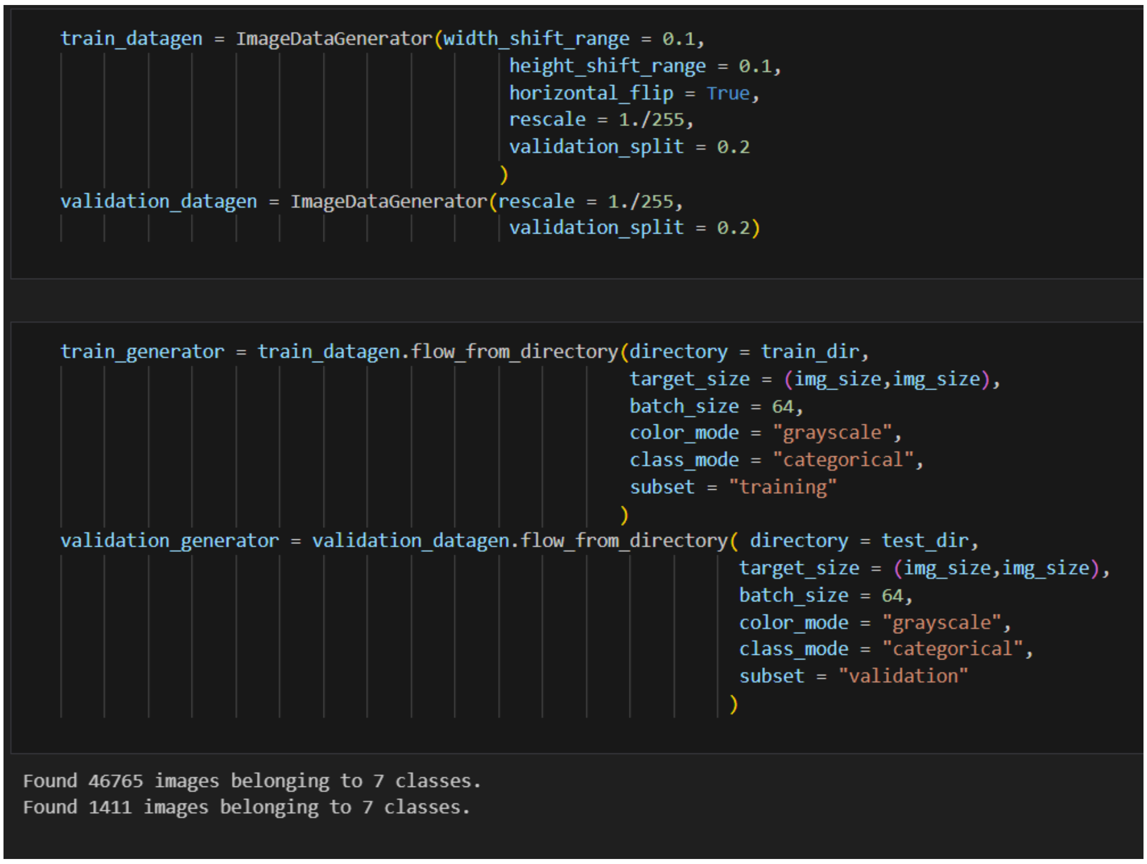
- **Training and Testing**: The dataset is split into training and test sets to facilitate model training and performance evaluation.
- **Benchmarking**: FER-2013 is commonly used to benchmark facial expression recognition algorithms.
- **Challenges**: The dataset is known for its challenging nature due to the low resolution of images and the presence of real-world noise.

Preprocessing Steps

Common preprocessing steps for FER-2013 include:

1. **Rescaling**: Normalizing pixel values (e.g., to a range of 0-1).
2. **Augmentation**: Applying data augmentation techniques such as rotation, scaling, and flipping to increase the diversity of training data.
3. **Cropping**: Cropping or padding images to focus on the face region.

Data Augmentation

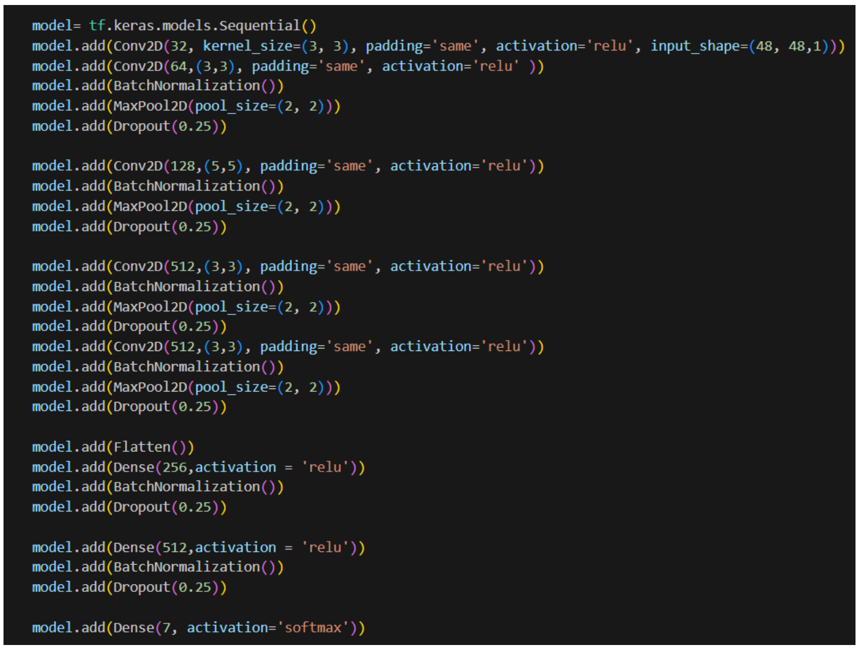


```
train_datagen = ImageDataGenerator(width_shift_range = 0.1,
                                    height_shift_range = 0.1,
                                    horizontal_flip = True,
                                    rescale = 1./255,
                                    validation_split = 0.2
                                   )
validation_datagen = ImageDataGenerator(rescale = 1./255,
                                         validation_split = 0.2)
```

```
train_generator = train_datagen.flow_from_directory(directory = train_dir,
                                                     target_size = (img_size,img_size),
                                                     batch_size = 64,
                                                     color_mode = "grayscale",
                                                     class_mode = "categorical",
                                                     subset = "training"
                                                    )
validation_generator = validation_datagen.flow_from_directory( directory = test_dir,
                                                              target_size = (img_size,img_size),
                                                              batch_size = 64,
                                                              color_mode = "grayscale",
                                                              class_mode = "categorical",
                                                              subset = "validation"
                                                             )
```

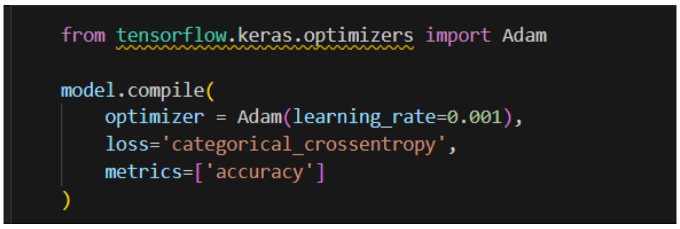
```
Found 46765 images belonging to 7 classes.
Found 1411 images belonging to 7 classes.
```

Data Modeling



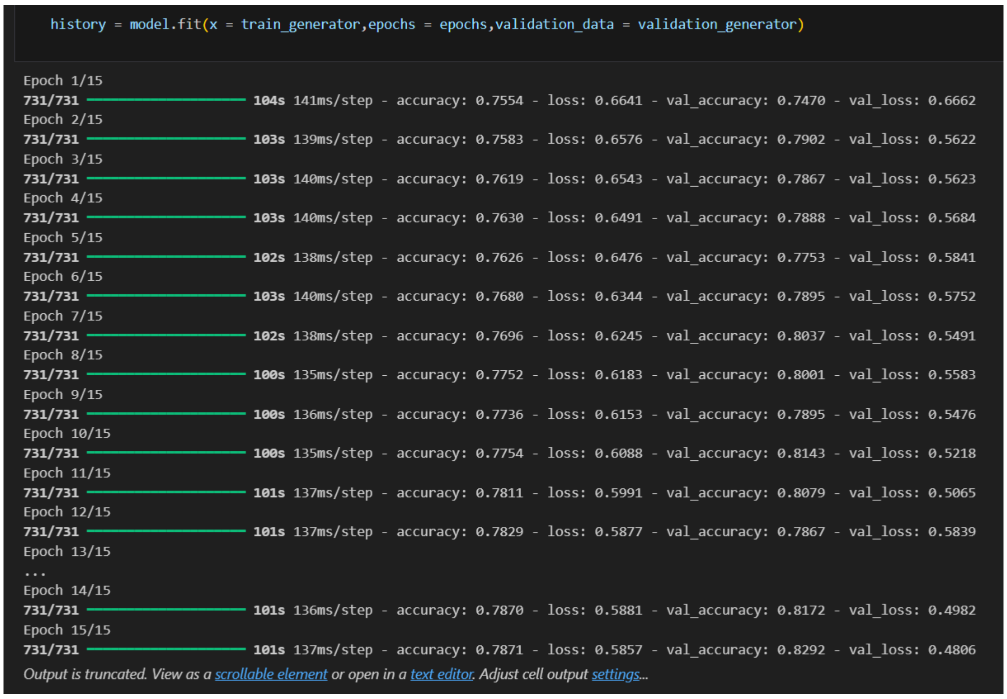
```
model= tf.keras.models.Sequential()
model.add(Conv2D(32, kernel_size=(3, 3), padding='same', activation='relu', input_shape=(48, 48,1)))
model.add(Conv2D(64,(3,3), padding='same', activation='relu' ))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128,(5,5), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(512,(3,3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(512,(3,3), padding='same', activation='relu'))
model.add(BatchNormalization())
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(256,activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Dense(512,activation = 'relu'))
model.add(BatchNormalization())
model.add(Dropout(0.25))

model.add(Dense(7, activation='softmax'))
```

Model Compile Process



```
from tensorflow.keras.optimizers import Adam

model.compile(
    optimizer = Adam(learning_rate=0.001),
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
```

Data Training

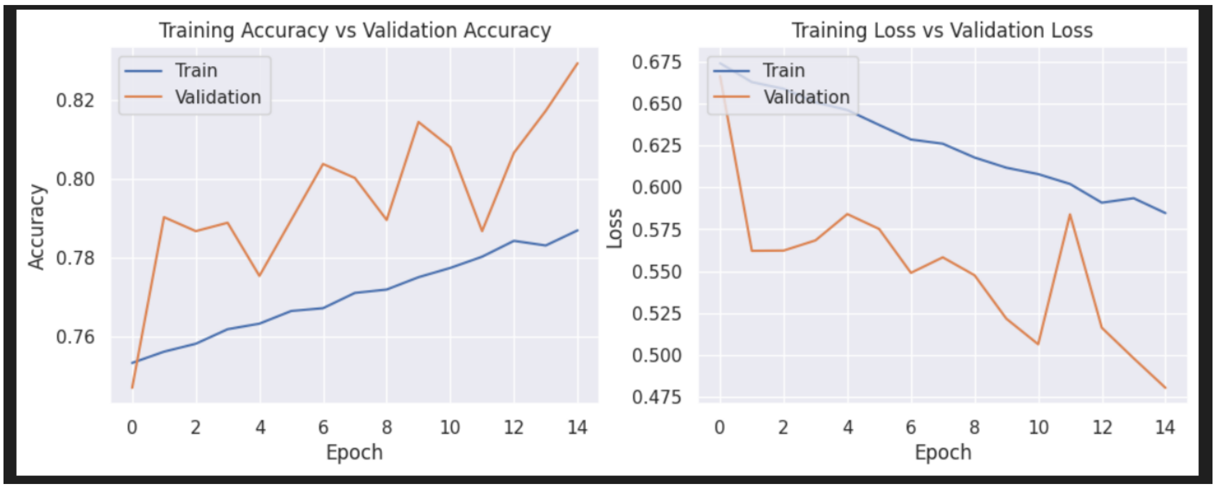


```
history = model.fit(x = train_generator,epochs = epochs,validation_data = validation_generator)
```

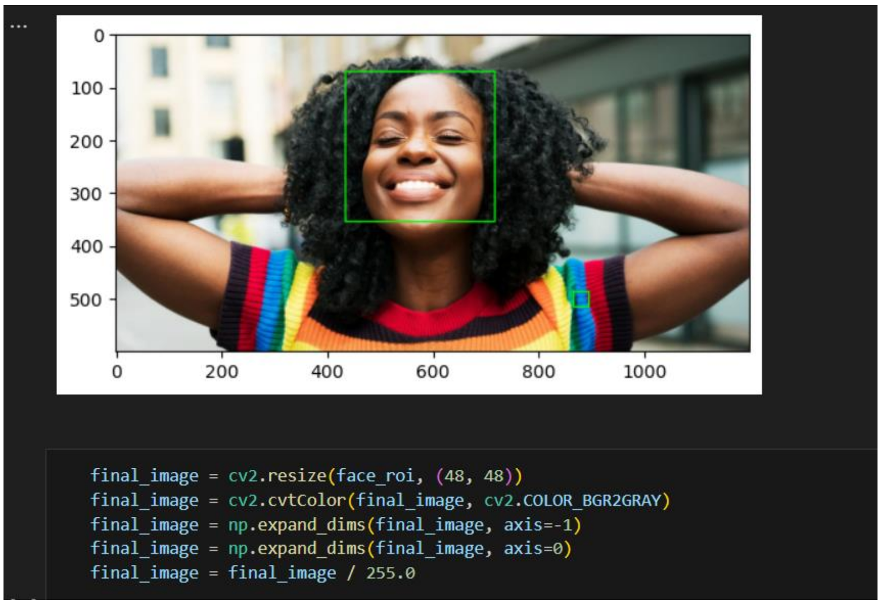
```
Epoch 1/15
731/731 ━━━━━━━━━━━━━━━━━━━━ 104s 141ms/step - accuracy: 0.7554 - loss: 0.6641 - val_accuracy: 0.7470 - val_loss: 0.6662
Epoch 2/15
731/731 ━━━━━━━━━━━━━━━━━━━━ 103s 139ms/step - accuracy: 0.7583 - loss: 0.6576 - val_accuracy: 0.7902 - val_loss: 0.5622
Epoch 3/15
731/731 ━━━━━━━━━━━━━━━━━━━━ 103s 140ms/step - accuracy: 0.7619 - loss: 0.6543 - val_accuracy: 0.7867 - val_loss: 0.5623
Epoch 4/15
731/731 ━━━━━━━━━━━━━━━━━━━━ 103s 140ms/step - accuracy: 0.7630 - loss: 0.6491 - val_accuracy: 0.7888 - val_loss: 0.5684
Epoch 5/15
731/731 ━━━━━━━━━━━━━━━━━━━━ 102s 138ms/step - accuracy: 0.7626 - loss: 0.6476 - val_accuracy: 0.7753 - val_loss: 0.5841
Epoch 6/15
731/731 ━━━━━━━━━━━━━━━━━━━━ 103s 140ms/step - accuracy: 0.7680 - loss: 0.6344 - val_accuracy: 0.7895 - val_loss: 0.5752
Epoch 7/15
731/731 ━━━━━━━━━━━━━━━━━━━━ 102s 138ms/step - accuracy: 0.7696 - loss: 0.6245 - val_accuracy: 0.8037 - val_loss: 0.5491
Epoch 8/15
731/731 ━━━━━━━━━━━━━━━━━━━━ 100s 135ms/step - accuracy: 0.7752 - loss: 0.6183 - val_accuracy: 0.8001 - val_loss: 0.5583
Epoch 9/15
731/731 ━━━━━━━━━━━━━━━━━━━━ 100s 136ms/step - accuracy: 0.7736 - loss: 0.6153 - val_accuracy: 0.7895 - val_loss: 0.5476
Epoch 10/15
731/731 ━━━━━━━━━━━━━━━━━━━━ 100s 135ms/step - accuracy: 0.7754 - loss: 0.6088 - val_accuracy: 0.8143 - val_loss: 0.5218
Epoch 11/15
731/731 ━━━━━━━━━━━━━━━━━━━━ 101s 137ms/step - accuracy: 0.7811 - loss: 0.5991 - val_accuracy: 0.8079 - val_loss: 0.5065
Epoch 12/15
731/731 ━━━━━━━━━━━━━━━━━━━━ 101s 137ms/step - accuracy: 0.7829 - loss: 0.5877 - val_accuracy: 0.7867 - val_loss: 0.5839
Epoch 13/15
...
Epoch 14/15
731/731 ━━━━━━━━━━━━━━━━━━━━ 101s 136ms/step - accuracy: 0.7870 - loss: 0.5881 - val_accuracy: 0.8172 - val_loss: 0.4982
Epoch 15/15
731/731 ━━━━━━━━━━━━━━━━━━━━ 101s 137ms/step - accuracy: 0.7871 - loss: 0.5857 - val_accuracy: 0.8292 - val_loss: 0.4806
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

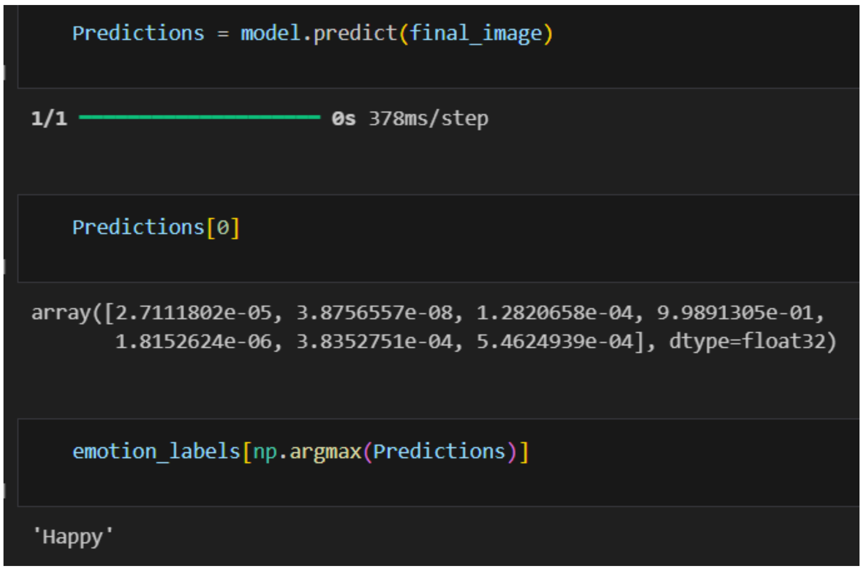
Results



Testing



```
final_image = cv2.resize(face_roi, (48, 48))
final_image = cv2.cvtColor(final_image, cv2.COLOR_BGR2GRAY)
final_image = np.expand_dims(final_image, axis=-1)
final_image = np.expand_dims(final_image, axis=0)
final_image = final_image / 255.0
```



```
Predictions = model.predict(final_image)
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 378ms/step
```

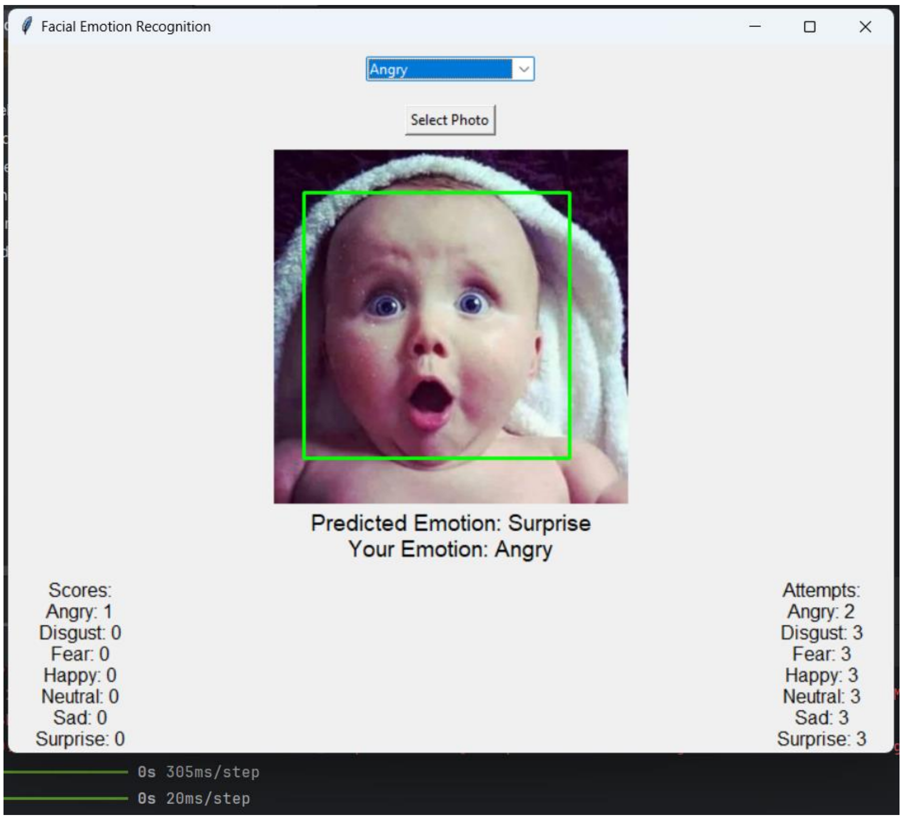
```
Predictions[0]
```

```
array([2.7111802e-05, 3.8756557e-08, 1.2820658e-04, 9.9891305e-01,
       1.8152624e-06, 3.8352751e-04, 5.4624939e-04], dtype=float32)
```

```
emotion_labels[np.argmax(Predictions)]
```

```
'Happy'
```

Application

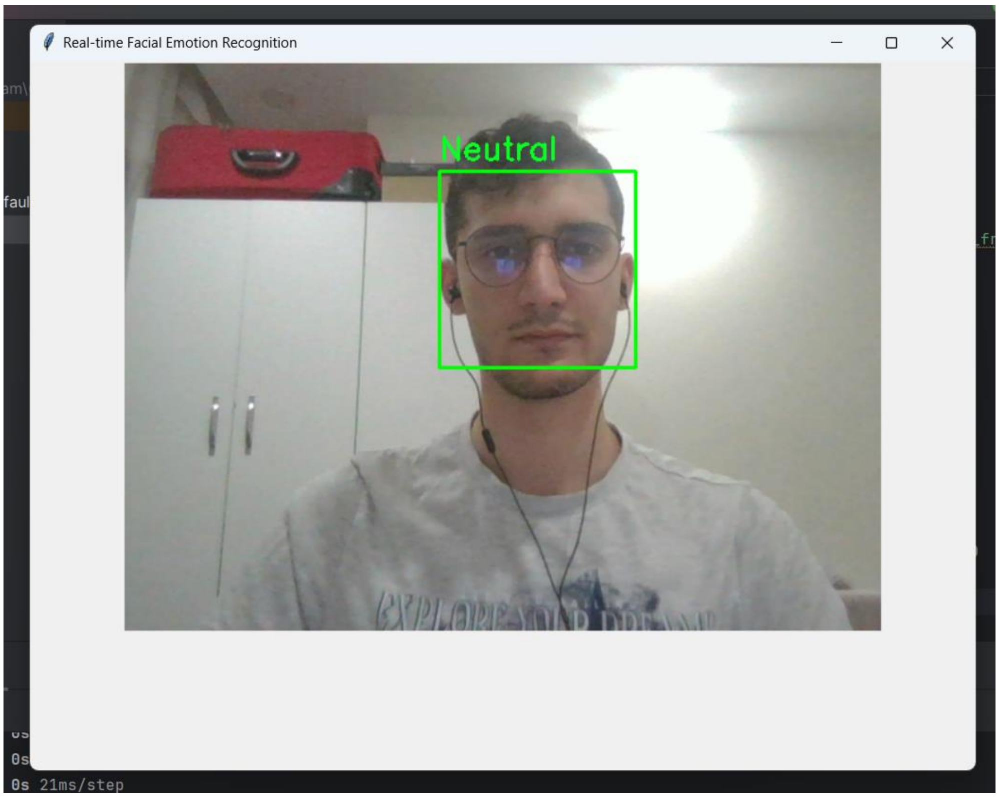


Predicted Emotion: Surprise
Your Emotion: Angry

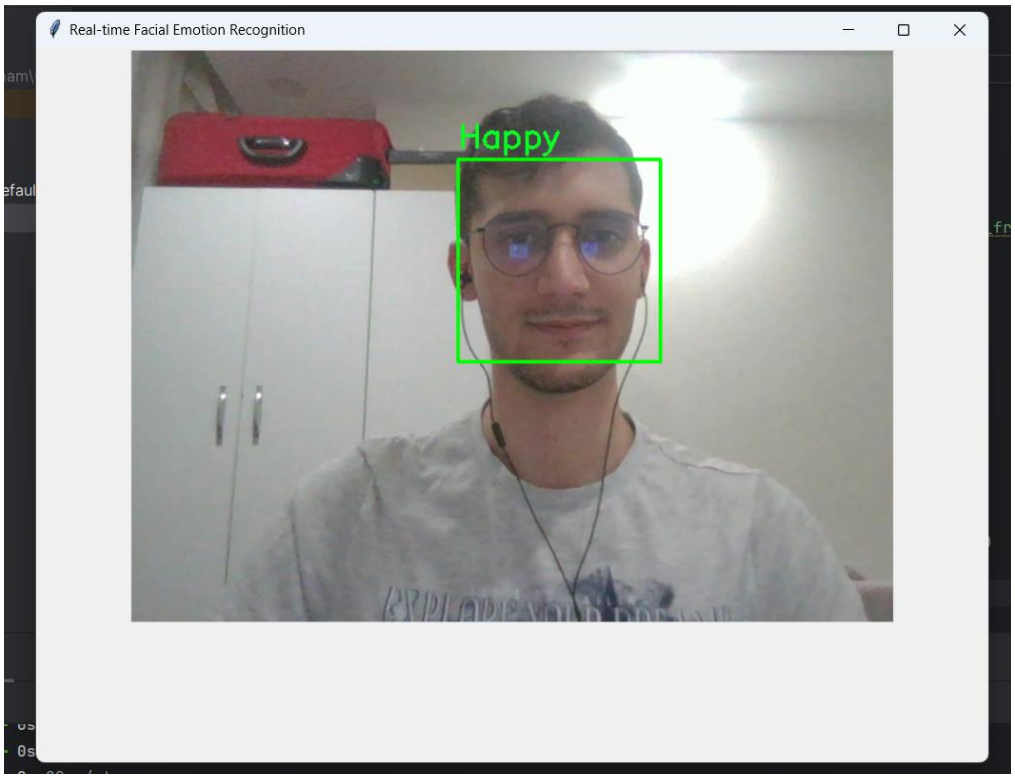
Scores:
Angry: 1
Disgust: 0
Fear: 0
Happy: 0
Neutral: 0
Sad: 0
Surprise: 0

Attempts:
Angry: 2
Disgust: 3
Fear: 3
Happy: 3
Neutral: 3
Sad: 3
Surprise: 3

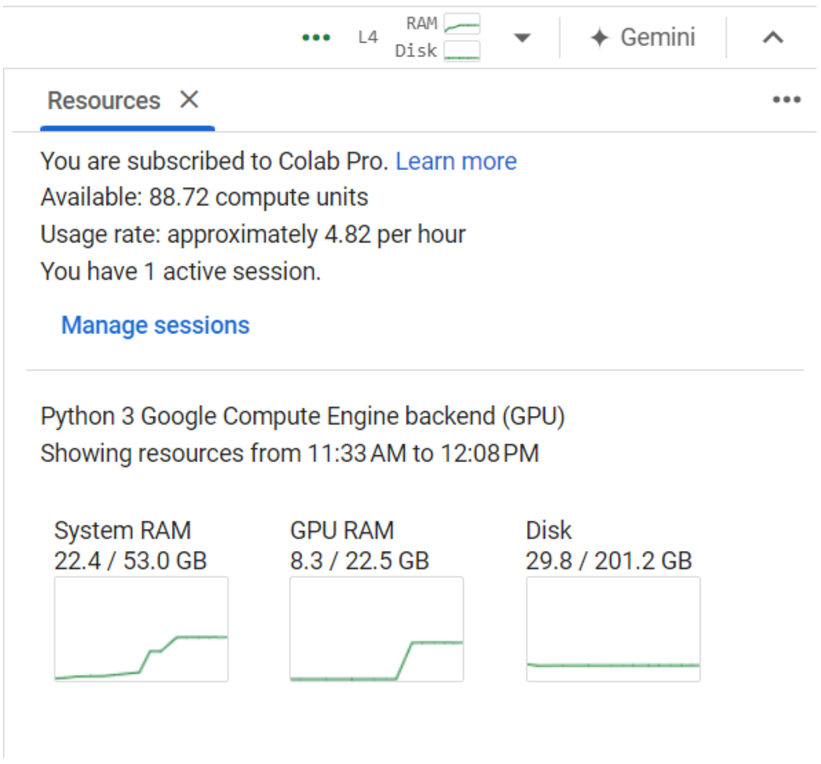
```
0s 305ms/step
0s 20ms/step
```



```
0s
0s 21ms/step
```



Colab Usage Information



You are subscribed to Colab Pro. Learn more
Available: 88.72 compute units
Usage rate: approximately 4.82 per hour
You have 1 active session.

Manage sessions

Python 3 Google Compute Engine backend (GPU)
Showing resources from 11:33 AM to 12:08 PM

System RAM
22.4 / 53.0 GB

GPU RAM
8.3 / 22.5 GB

Disk
29.8 / 201.2 GB