Emotional face recognizes

Ahmed Mohamed Sabaa

Mohamed Abdullah Mohamed

Dr\Shady Zahran,
Eng\Mohamed Moheb

12/24/23

Image processing

Executive Summary

The primary objective of this project is to create a Facial Emotion Recognition model by utilizing Convolutional Neural Networks (CNNs). The outcomes of this project reveal that the model has been successfully trained and validated on a diverse dataset, resulting in a satisfactory level of accuracy. To ensure seamless integration into applications, the model has been deployed and saved in JSON and HDF5 formats. Furthermore, the model's practical usability has been demonstrated through real-time predictions on external test images. In summary, this project effectively achieves its goals and provides a valuable solution for emotion recognition, which can be applied in human-computer interaction. Moving forward, potential future directions for improvement may include optimizations, diversification of the dataset, and real-time implementation to further enhance usability.

Table of Contents

Introduction

Facial expressions: These movements express the individual's emotional state to the observer. Facial expressions are a form of non-verbal communication. Facial expressions are one of the basic means of communicating social information between people. Through expressions, one can know if a person is happy, sad, or angry. Through a change in the shape of the muscles and some parts of the face, this can be determined.

Through the pictures taken by the human being, this can also be known, and through that the machine can recognize the emotional state that controls the human being in the picture and the basic feelings that can be contained in any picture, which are anger, disgust, fear, happiness, sadness, and surprise. The main purpose of this is to reach the interaction between... Humans and machines using eye gaze and other expressions.

Despite the progress that has been made in recent decades, achieving high accuracy in recognizing facial expressions is still difficult due to the complexity and diversity of expressions, because these expressions may differ from one person to another depending on the age of the person. They change over time and are also affected by gender. Size, skin color, and there may also be external influences that affect the image, such as image contrast, lighting, and shooting position.

With the advent of modern technology, our desires have risen and become limitless, and developments in technology are now occurring rapidly. Image processing is a broad field of research in today’s world and its applications are widespread. Some of these uses or applications that can be benefited from this project in our current world is that we can We use cameras that can determine facial emotions in public places. If there is a person running away, for example, he will be angry or afraid, he can be found through this. They can be used to detect lies during the investigation of suspects, or they can be used in restaurants or supermarkets to obtain a customer satisfaction rate with the products. In service and in education, one can obtain how students interact during a lecture, and there are many uses that can be obtained from that. The objective of this project is to develop an automatic facial expression recognition system that can capture images of the human face that contain some expressions as input, recognize them, and classify them. into seven different categories of expressions.

Literature Review

Psychological Foundations (19th-20th Century).

The exploration of emotions and their connection to facial expressions has roots in psychological research. Early scholars like Charles Darwin and Paul Ekman laid the groundwork for understanding universal facial expressions. Darwin's work, "The Expression of the Emotions in Man and Animals," published in 1872, explored the evolutionary significance of facial expressions.

As per various literature Reviews it is found that for implementing this project four basic steps.

are required to be performed.

1. Preprocessing
2. Face registration
3. Facial feature extraction
4. Emotion classification

face Recognition and tracking have a challenge due to the limited facial information. Found many algorithms to make it as Hog, SIFT, and CNNs and we use CNNs in this project.by using the machine learning algorithms it's tackled the classifier problem and may be use the notably Support Vector Machine (SVM).

The importance of block size in LBP feature extraction for high recognition accuracy, exceeding 97%, is highlighted. The subsequent discussion outlines the four basic steps of implementing the project: preprocessing, face registration, facial feature extraction, and emotion classification.

In the initial stage of preparing the data, we focus on refining the images through processes like reducing noise, converting them, and adjusting their shapes. The next step involves recognizing and standardizing the faces in the images, making sure they're in a consistent format. Following that, we pinpoint specific features on the face and create a numerical summary of these features. When it comes to understanding emotions, we categorize faces into seven basic emotions. To recognize facial expressions, we explore different approaches, including using Neural Networks, Principal Component Analysis. The importance of using diverse datasets for training and testing is emphasized, and various facial datasets available online are introduced, enriching the understanding of human expressions.

Methodology

Design Process:

Algorithm Design:

We use CNNs in classification because we classify images into different categories.

This model consists of layers.

1. Convolutional Layer:

Logic:

- Convolution is the fundamental operation in CNNs. It involves sliding a small filter (also known as a kernel or a convolutional kernel) over the input image and computing the dot product between the filter and the local region of the input.
- This operation captures local patterns, edges, and simple textures in the input image.

Theory:

- Convolutional layers apply multiple filters to the input, generating feature maps that highlight different aspects of the input image.
- The weights of the filters are learned during the training process, enabling the network to automatically extract relevant features.

2. Pooling Layer:

Logic:

- Pooling (usually max pooling) is employed to reduce the spatial dimensions of the input volume, reducing the number of parameters and computations in the network.
- It retains the most important information by selecting the maximum value within a local region.

Theory:

- Pooling helps achieve translation invariance, making the network more robust to variations in the position of features within the input.
- It reduces the spatial resolution, allowing higher-level features to be detected over larger regions.

3. Activation Function (e.g., ReLU):

Logic:

- After each convolutional or fully connected layer, an activation function is applied elementwise to introduce non-linearity into the model.
- Rectified Linear Unit (ReLU) is commonly used, replacing negative values with zero.

Theory:

- Non-linearities enable the network to learn complex mappings between input and output.
- ReLU is preferred due to its simplicity and effectiveness in mitigating the vanishing gradient problem.

4. Fully Connected Layer:

Logic:

Fully connected layers connect every neuron in one layer to every neuron in the next layer, enabling high-level feature learning and decision-making.

Theory:

- These layers capture global relationships in the data, learning complex combinations of features for the final classification or regression.
- Fully connected layers are typically used in the final stages of the network.

5. Backpropagation and Gradient Descent:

Logic:

- During training, the network is optimized by adjusting the weights based on the error between predicted and actual outputs.
- Backpropagation computes gradients of the loss with respect to the network parameters.
- Gradient Descent or its variants (e.g., Adam, RMSProp) are used to update the weights.

Theory:

- Iterative optimization allows the network to learn representations that minimize errors on the training data.
- It helps the network generalize well to unseen data.

Why use these layers.

These layers are often used together to build convolutional neural network architectures for image classification or other computer vision tasks. For example, a common pattern might be Conv2D layers followed by MaxPooling2D layers for feature extraction, followed by one or more Dense layers for classification. Dropout layers are often used to prevent overfitting during training.

Coding Practices:

Programming language: We use python 3.9.

CV2 : OpenCV is an open-source computer vison and image processing library, also providing a convenient interface for developers to work with computer vision algorithms and image processing tasks.

TensorFlow : is an open-source machine learning library developed by the google brain team. It provides a comprehensive set of tools for building and deploying machine learning models, with a particular emphasis on neural networks.

Keras in TensorFlow : Keras is a high-level neural networks API that runs on top of TensorFlow. It provides a user-friendly interface for building, training, and deploying deep learning models. In the context of TensorFlow, Keras serves as the official high-level API, making it easier for developers to create and experiment with neural networks.

Optimizers : In the context of deep learning, an optimizer is a crucial component responsible for adjusting the weights of a neural network during training. The goal is to minimize the loss function, which measures the difference between the predicted output and the actual target values.

Adam Optimizer :

- Adam is a popular optimization algorithm used in training deep neural networks. It stands for Adaptive Moment Estimation. Adam combines ideas from two other optimization algorithms: RMSprop and Momentum. It adapts the learning rates of each parameter individually, making it effective in a wide range of scenarios.

- It's worth noting that the term "LegacyAdam" in your import statement suggests that there might be a newer version or variant of the Adam optimizer in TensorFlow. Naming it as "Legacy" could imply that this version is either an older implementation or has specific characteristics that differ from a more recent version. It's always a good idea to refer to the official TensorFlow documentation or release notes for the most up-to-date information on optimizers and their implementations.

Sequential : Linear stack of layers.

is a way to create a neural network model layer by layer in step-by-step fashion.

Because each layer in the model is connected to the previous layer.

And we can add in model by use add method.

Conv2D : in this layer used for processing spatial data and it's a fundamental building block in CNNs.

And take some parameters:

- Filters – no filters to apply to the input and each filter extracts different features from the input.
- Kernal size – the size of the kernel
- Activation – which function will apply to the output.
- Input shape – shape of the input data (height, width, channels) this in the first layer

Max pooling : using to down sample the spatial dimension.

is a 2D pooling layer used to down-sample the spatial dimensions of the input volume, reducing the number of parameters and computation in the network.

Parameters:

pool_size: Factor by which to downscale. It specifies the vertical and horizontal downscaling factors.

Dense : is a fully connected layer, meaning each neuron in the layer is connected to every neuron in the previous and the next layer.

Parameters:

- Units – no of neurons in the layer
- Activation – function will apply.
- Input dim – only needed in the first layer no of input features.

Dropout : is a regularization technique where randomly selected neurons are ignored during training, which helps prevent overfitting.

Parameters:

rate: Fraction of the input units to drop.

Flatten : #The flatten layer is used to convert the output of the convolutional

#layers into a one-dimensional array.

#This is necessary before transitioning to fully connected layers.

Convert from 2D may be 3 or 4 D to 1D.

The Flatten layer bridges the gap between the spatial-aware layers (like convolutional layers) and the fully connected layers, making it possible to apply these layers to the high-level features extracted by the convolutional layers.

This transformation is necessary because fully connected layers (Dense layers) in a neural network expects a one-dimensional input. By reshaping the tensor into a linear array

Relu : Its primary function is to introduce non-linearity to the network while being computationally efficient.

Neurons:

- The basic building block is neuron(nodes) in the neural network.
- Organized into layers, each layer is connected to the next layer through weighted.
- The input to a neuron is a weighted sum of the outputs from the neurons in the previous layer, along with a bias term. Mathematically,
- this can be represented as: Input to neuron = ∑ ^n _ i= 1 (weight i * output i) + Bias
- The result of this summation is then passed through an activation function to introduce non-linearity.

NumPy: This line imports the NumPy library, which is used for numerical operations.

model_from_json: This line imports the model_from_json function from the Keras library. This function is used to create a Keras model from a JSON file.

load_img: This line imports the load_img function from Keras, which is used for loading an image.

ImageDateGenerator : helps to normalize input images. This can help improve the convergence and performance of the neural network.

The primary goal of data augmentation is to artificially increase the diversity of your training dataset by applying various transformations to the existing images. By doing so, the model learns to recognize patterns that are invariant to these transformations, making it more robust when presented with unseen or slightly different data during inference.

Some methods happen in background.

In conv and pooling used to extract hierarchical features from input data.

Thus, output from these layers is often a 3d or 4d (depending on (grayscale or RGB)).

IDE : we use Anaconda Environment with ide spider.

Date Acquisition:

Data Preprocessing

We resize the image to (48 * 48 pixel) and normalize it to be suitable for training and testing the image processing algorithms.

Date Collection

We got the dataset from Kegalle.

Testing Methods:

The model tests the images that are taken from users and the model predicts the status of the face if are some of the one-off labels.

The test cases that cover the model

- Happy
- Sad
- Angry
- Nature
- Fear
- Surprise
- Disgust

Validation Procedures:

The dataset consists of two categories training and testing.

The model is trained by the train data and the testing data for testing that consist of some categories that we mentioned in label then take image from any category then predict the status of the face.

Performance Metrics:

Accuracy: The ratio of correctly predicted instances to the total instances.

Accuracy =Number of Correct Predictions / Total Number of Predictions.

Precision: The ratio of true positive predictions to the total positive predictions.

Precision= True Positives / (True Positives + False Positives).

Recall (Sensitivity): The ratio of true positive predictions to the total actual positive instances.

Recall = True Positives / (True Positives + False Negatives)

F1 Score: The harmonic means of precision and recall.

F1 Score = 2 × ((Precision × Recall) / (Precision + Recall))

Confusion Matrix: A table that describes the performance of a classification algorithm. It shows the counts of true positives, true negatives, false positives, and false negatives.

Technical Details

System Architecture:

1. **Date Input:**
    - The system starts with an amount of image data that is organized into training and testing sets. The images are stored in separate directories.
    - Subdirectories for each emotion class contain images of individuals expressing specific emotions (angry, disgust, fear, happy, neutral, sad, surprise)
2. **Data Preprocessing:**
    - **ImageDataGenerator (**from keras.preprocessing.image import ImageDataGenerator)
        - Rescales pixel values to normalize them between 0 and 1.
        - Augmentation parameters: Rotation, width and height shift, shear range, zoom range, and horizontal flip.
        - Generates batches of preprocessed images during training and testing.
3. **Training Data Flow:**
    - **Convolutional Neural Network (CNN) Model:**
        - Convolutional Layers:
            - 32 filters with a kernel size of (3, 3) and Relu activation.
            - 64 filters with a kernel size of (3, 3) and Relu activation.
            - 128 filters with a kernel size of (3, 3) and Relu activation.
        - Max-Pooling Layers:
            - Max pooling with a pool size of (2, 2) after the first and second convolutional blocks.
        - Dropout Layers:
            - Dropout with a rate of 0.25 after the first max-pooling layer.
            - Dropout with a rate of 0.25 after the second max-pooling layer.
            - Dropout with a rate of 0.5 before the final fully connected layer.
        - Fully Connected Layers:
            - Two fully connected layers with Relu activation (1024 neurons and 7 neurons).
        - Output Layer:
            - SoftMax activation for multi-class classification (7 output neurons representing different emotions).
    - **Optimizer and Compilation:**
        - Optimizer**:**
            - Adam optimizer with a learning rate of 0.001.
        - Loss Function:
            - Categorical cross entropy for multi-class classification.
        - Metrics:
            - Accuracy is used as an evaluation metric.

- **Training Loop:**
    - Steps per Epoch:
        - Determined by dividing the total number of training samples by the batch size.
    - Epochs:
        - 100 epochs for training.
    - Validation Steps:
        - Determined by dividing the total number of validation samples by batch size.

4. **Testing Data Flow:**
    - **Evaluation:**
        - Test data flows through the trained model.
        - Predictions are generated for each image.
        - Metrics, including accuracy, are calculated for model evaluation.
5. **Model Serialization:**
    - **Saving Model:**
        - Model architecture is saved to a JSON file (emotiondetector.json).
        - Trained weights are saved to an HDF5 file (emotiondetector.h5).
6. **Testing the Trained Model:**
    - **Loading Model:**
        - Trained model architecture is loaded from the JSON file.
        - Weights are loaded from the HDF5 file.
    - **Prediction:**
        - Image preprocessing function (ef) prepares test images for prediction.
        - Model predicts emotion labels for faces in the test images.
7. **Haar Cascade Face Detection:**
    - **Face Detection:**
        - Utilizes Haar cascade classifier from OpenCV for face detection.
        - Scale factor, minimum neighbors, and minimum size parameters are set for efficient face detection.
8. **Visualization of Results:**
    - **Rectangles and Labels:**
        - Rectangles are drawn around detected faces in the test images.
        - Predicted emotion labels are displayed near each face.
    - **Display:**
        - Processed images with rectangles and labels are displayed for visual inspection.

9. **Flow chart:**

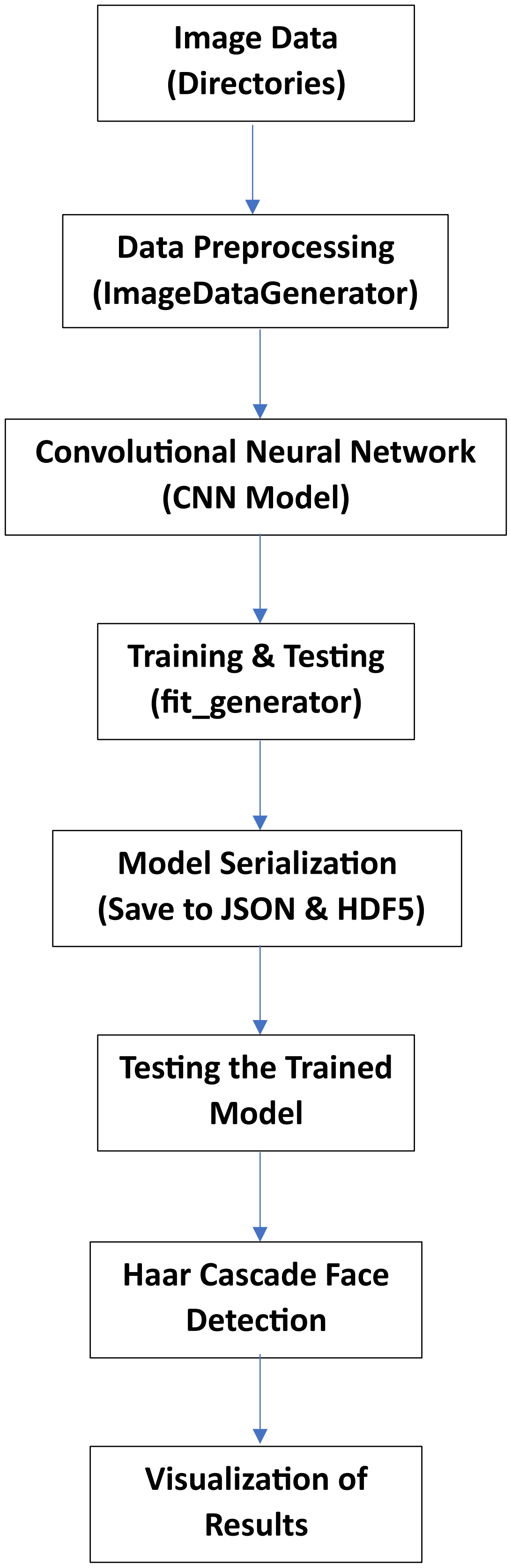


Image Data (Directories)
Data Preprocessing (ImageDataGenerator)
Convolutional Neural Network (CNN Model)
Training & Testing (fit_generator)
Model Serialization (Save to JSON & HDF5)
Testing the Trained Model
Haar Cascade Face Detection
Visualization of Results

Algorithm Explanation:

system is based on Convolutional Neural Networks (CNNs), a type of deep learning model designed for processing structured grid data, such as images. Let's analyze the elements of the algorithm:

Convolutional Neural Network (CNN) Architecture:

1. **Convolutional Layers:**
   - **Mathematical Principle:**

     $\text{output}_{i,j} = \sigma\left(\sum_{m,n} \text{input}_{i+m,j+n} \times \text{filter}_{m,n} + \text{bias}\right)$, where σ is the activation function.
   - **Computational Principle:**
     - Convolutional layers learn spatial hierarchies of features in the input image.
2. **Max-Pooling Layers:**
   - **Mathematical Principle:**

     $\text{output}_{i,j} = \max_{m,n} \text{input}_{2i+m,2j+n}$
   - **Computational Principle:**
     - Reduces spatial dimensions, retaining essential information.
3. **Dropout Layers:**
   - **Mathematical Principle:**

     $$y_i = \begin{cases} x_i & \text{with probability } 1-p \\ 0 & \text{with probability } p \end{cases}$$
   - **Computational Principle**
     - Introduces regularization, preventing overfitting by promoting diverse feature learning**.**
4. **Fully Connected Layers:**
   - **Mathematical Principle:**

     Neurons in fully connected layers perform $\text{output}_i = \sigma\left(\sum_j \text{input}_j \times \text{weight}_{ij} + \text{bias}_i\right)$.
   - **Computational Principle:**
     - Captures high-level features by connecting all neurons in one layer to all neurons in the next.
     - Introduces regularization by randomly setting a fraction p of input units to zero during training.
5. **Softmax Activation (Output Layer):**
   - **Mathematical Principle:**

     $\text{softmax}_i(\mathbf{z}) = \frac{e^{z_i}}{\sum_j e^{z_j}}$, where $\mathbf{z}$ is the raw output.
   - **Computational Principle:**
     - Produces class probabilities for multi-class classification.

Training Algorithm:

1. **Categorical Cross entropy Loss:**
   - **Mathematical Principle:**

     $$L(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_i y_i \log(\hat{y}_i)$$

   - **Computational Principle:**
     - Guides the optimization process to minimize prediction errors.
2. **Adam Optimizer:**
   - **Mathematical Principle:**

     For each Parameter w^j

     (j subscript dropped for clarity)

     $$\nu_t = \beta_1 * \nu_{t-1} - (1 - \beta_1) * g_t$$
     $$s_t = \beta_2 * s_{t-1} - (1 - \beta_2) * g_t^2$$
     $$\Delta\omega_t = -\eta \frac{\nu_t}{\sqrt{s_t + \epsilon}} * g_t$$
     $$\omega_{t+1} = \omega_t + \Delta\omega_t$$

     η : *Initial Learning rate*
     g_t : *Gradient at time t along* ω^j
     ν_t : *Exponential Average of gradients along* ω_j
     s_t : *Exponential Average of squares of gradients along* ω_j
     β_1, β_2 : *Hyperparameters*
   - **Computational Principle:**
     - Optimizes the model's weights efficiently during training.
3. **Backpropagation:**
   - **Mathematical Principle:**

     Chain rule: $\frac{\partial L}{\partial w_{ij}} = \frac{\partial L}{\partial \text{output}_i} \times \frac{\partial \text{output}_i}{\partial w_{ij}}$

   - **Computational Principle:**
     - Calculates gradients of the loss with respect to model parameters.
4. **Training Loop:**
   - **Mathematical Principle:**
     - Iteratively adjusts weights based on calculated gradients.
   - **Computational Principle:**
     - Advances through batches of training data, fine-tuning the model to learn meaningful representations.

Model Serialization:

1. **Saving Architecture (JSON):**
   - **Mathematical Principle:**
     - Serializes the architecture of the neural network, including layer types, configurations, and connectivity information.
   - **Computational Principle:**
     - Stores the model architecture in a JSON file for future reconstruction.
2. **Saving Weights (HDF5):**
   - **Mathematical Principle:**
     - Serializes the learned weights of the neural network.
   - **Computational Principle:**
     - Stores the model weights in an HDF5 file for future use.

Testing Algorithm:

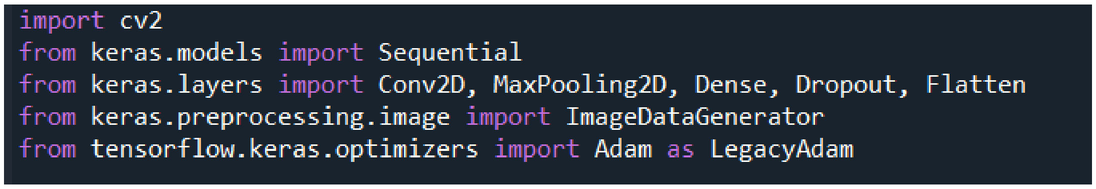
1. **Loading Model:**
   - **Mathematical Principle:**
     - Reads the serialized architecture and weights.
   - **Computational Principle:**
     - Reconstructs the model architecture and loads the trained weights.
2. **Image Preprocessing (ef Function):**
   - **Mathematical Principle:**
     - Rescales and reshapes test images to match the input dimensions of the trained model.
   - **Computational Principle:**
     - Prepares test images for input into the model.
3. **Emotion Prediction:**
   - **Mathematical Principle:**
     - Applies the trained model to predict emotion probabilities.
   - **Computational Principle:**
     - Produces probability distributions over emotion classes for each input image.

Haar Cascade Face Detection:

1. **Haar Cascade Classifier:**
   - **Mathematical Principle:**
     - Uses Haar-like features and a cascade of classifiers for rapid object detection.
   - **Computational Principle:**
     - Efficiently scans images to identify regions likely to contain faces.

Code Implementation:

1. **Train model**:
   - Importing Required Packages:

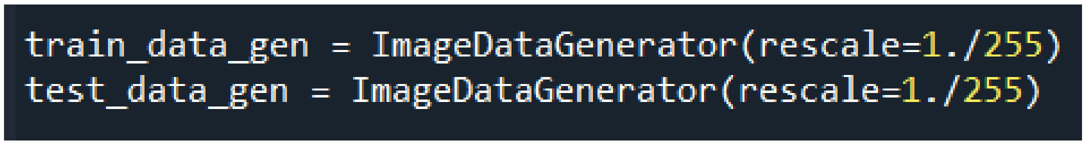


```
import cv2
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Dense, Dropout, Flatten
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.optimizers import Adam as LegacyAdam
```

Explanation:

✓ Imports necessary libraries such as OpenCV (cv2), Keras for building the neural network model, and image data generator from Keras for data augmentation and uses the Adam optimizer from TensorFlow (LegacyAdam) for model compilation.

- Image Data Generator Initialization:

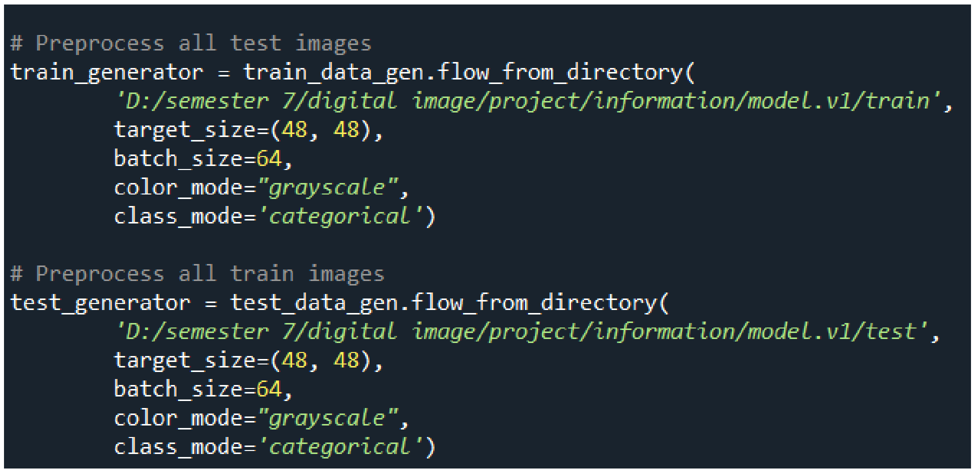


```
train_data_gen = ImageDataGenerator(rescale=1./255)
test_data_gen = ImageDataGenerator(rescale=1./255)
```

Explanation:

✓ Initializes image data generators for both training and testing datasets and Rescales pixel values to a range between 0 and 1.

- Loading and Preprocessing Training and Test Images:

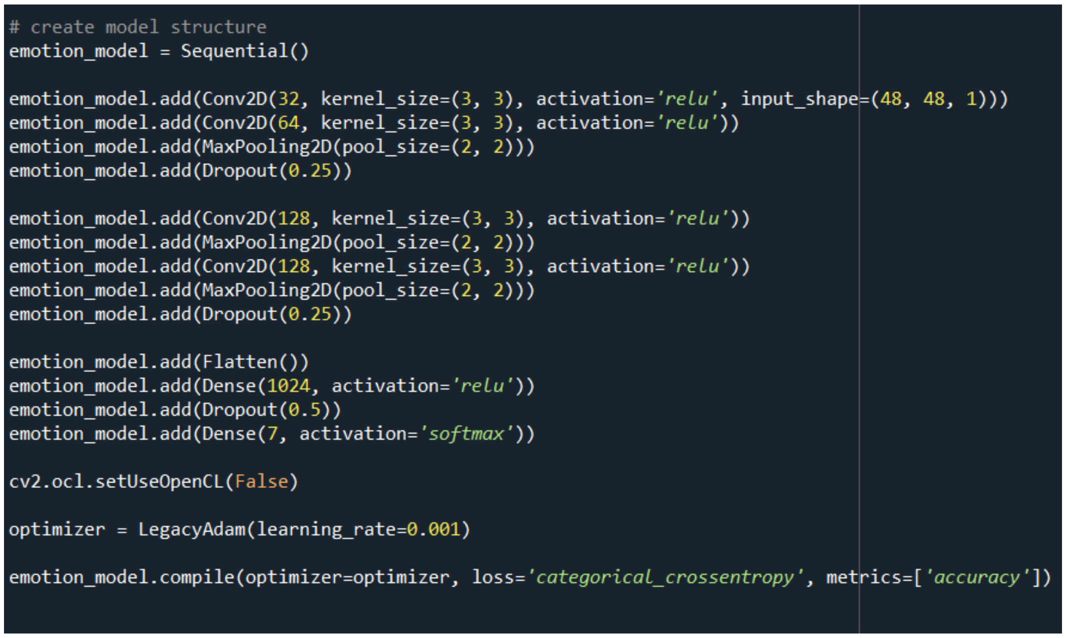


```
# Preprocess all test images
train_generator = train_data_gen.flow_from_directory(
        'D:/semester 7/digital image/project/information/model.v1/train',
        target_size=(48, 48),
        batch_size=64,
        color_mode="grayscale",
        class_mode='categorical')

# Preprocess all train images
test_generator = test_data_gen.flow_from_directory(
        'D:/semester 7/digital image/project/information/model.v1/test',
        target_size=(48, 48),
        batch_size=64,
        color_mode="grayscale",
        class_mode='categorical')
```

Explanation:

✓ Uses the previously initialized data generators to preprocess training and test images and Specifies parameters like target size, batch size, color mode, and class mode.

- Neural Network Model Architecture:

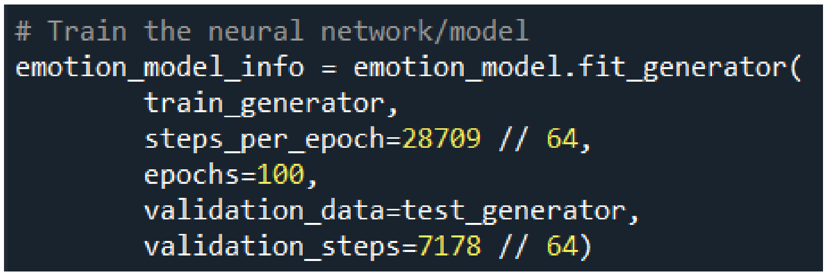


```
# create model structure
emotion_model = Sequential()

emotion_model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(48, 48, 1)))
emotion_model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))

emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
emotion_model.add(MaxPooling2D(pool_size=(2, 2)))
emotion_model.add(Dropout(0.25))

emotion_model.add(Flatten())
emotion_model.add(Dense(1024, activation='relu'))
emotion_model.add(Dropout(0.5))
emotion_model.add(Dense(7, activation='softmax'))

cv2.ocl.setUseOpenCL(False)

optimizer = LegacyAdam(learning_rate=0.001)

emotion_model.compile(optimizer=optimizer, loss='categorical_crossentropy', metrics=['accuracy'])
```

Explanation:

✓ Initializes a sequential model (emotion_model) for facial emotion recognition, Configures the architecture with convolutional, pooling, dropout, and dense layers and compiles the model using the Adam optimizer, categorical cross entropy loss, and accuracy as a metric.

- Model Training:

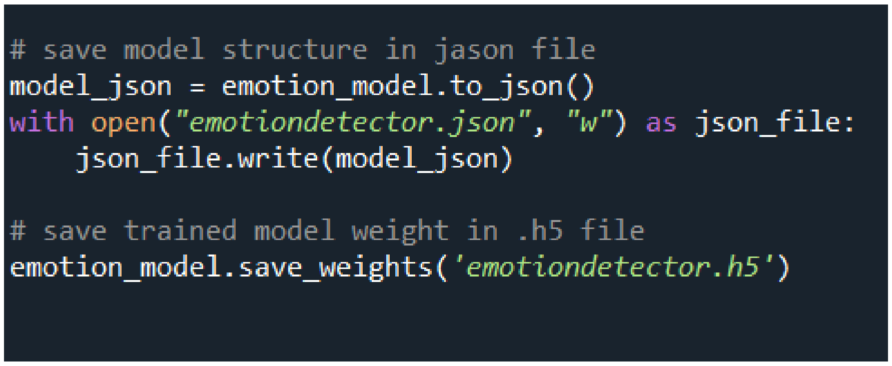


```
# Train the neural network/model
emotion_model_info = emotion_model.fit_generator(
        train_generator,
        steps_per_epoch=28709 // 64,
        epochs=100,
        validation_data=test_generator,
        validation_steps=7178 // 64)
```

Explanation:

✓ Uses the fit_generator method to train the model on the preprocessed training data, Specifies the number of steps per epoch and the number of epochs for training and validates the model on the test data during training.

- Saving Model Structure and Weights:



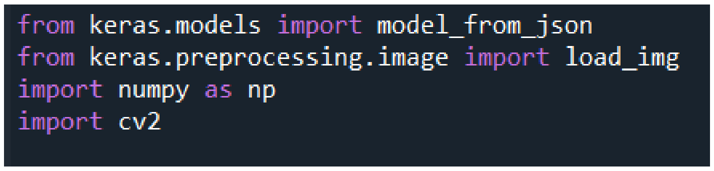
```
# save model structure in jason file
model_json = emotion_model.to_json()
with open("emotiondetector.json", "w") as json_file:
    json_file.write(model_json)

# save trained model weight in .h5 file
emotion_model.save_weights('emotiondetector.h5')
```

Explanation:

- ✓ Saves the trained model's architecture in a JSON file (“emotiondetector.json”).
- ✓ Saves the trained model weights in an HDF5 file (“emotiondetector.h5”).

2. **Test model**:

- Importing Required Packages:

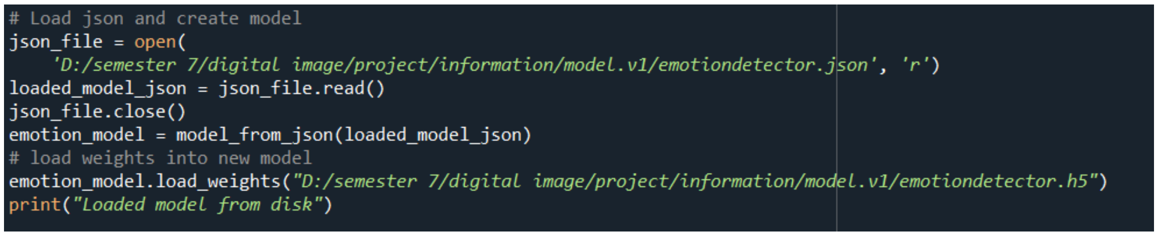


```
from keras.models import model_from_json
from keras.preprocessing.image import load_img
import numpy as np
import cv2
```

Explanation:

- ✓ “model_from_json”: To create a Keras model from JSON.
- ✓ “load_img”: To load an image.
- ✓ “numpy”: For numerical operations.
- ✓ “cv2”: OpenCV library for computer vision.

- Loading Trained Model:

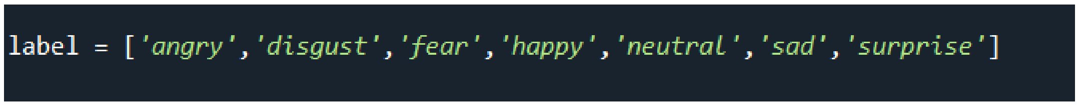


```
# Load json and create model
json_file = open(
    'D:/semester 7/digital image/project/information/model.v1/emotiondetector.json', 'r')
loaded_model_json = json_file.read()
json_file.close()
emotion_model = model_from_json(loaded_model_json)
# load weights into new model
emotion_model.load_weights("D:/semester 7/digital image/project/information/model.v1/emotiondetector.h5")
print("Loaded model from disk")
```

Explanation:

- ✓ Loads the saved model structure from the JSON file and Loads the saved model weights from the HDF5 file.

- Define Emotion Labels:

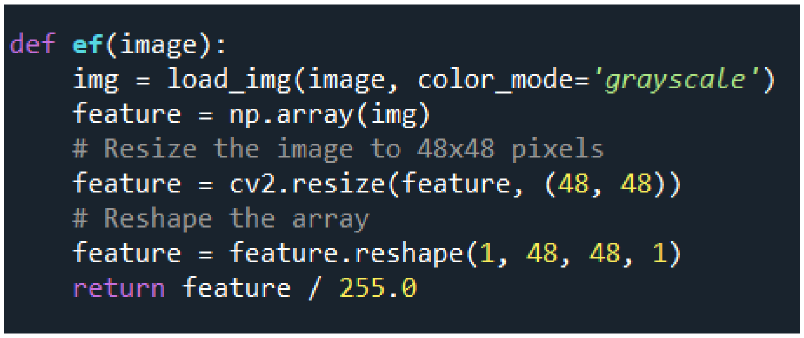


```
label = ['angry','disgust','fear','happy','neutral','sad','surprise']
```

Explanation:

- ✓ Defines a list of emotion labels corresponding to the output classes of the model.

- Image Preprocessing Function (ef):

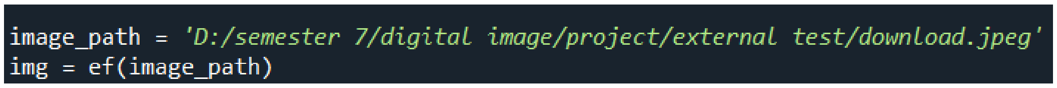


```
def ef(image):
    img = load_img(image, color_mode='grayscale')
    feature = np.array(img)
    # Resize the image to 48x48 pixels
    feature = cv2.resize(feature, (48, 48))
    # Reshape the array
    feature = feature.reshape(1, 48, 48, 1)
    return feature / 255.0
```

Explanation:

- ✓ Takes an image file path as input,
- ✓ Loads the image in grayscale mode,
- ✓ Resizes the image to 48x48 pixels,
- ✓ Reshapes the array to match the expected input shape of the model.
- ✓ Normalizes pixel values to the range [0, 1].

- Load and Preprocess a Test Image:

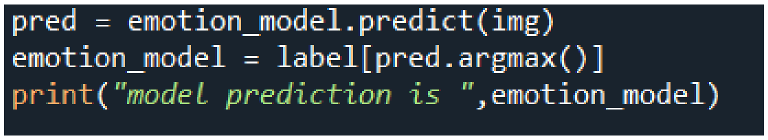


```
image_path = 'D:/semester 7/digital image/project/external test/download.jpeg'
img = ef(image_path)
```

Explanation:

- ✓ Specifies the file path of a test image.
- ✓ Uses the ef function to preprocess the image.

- Make Model Prediction:

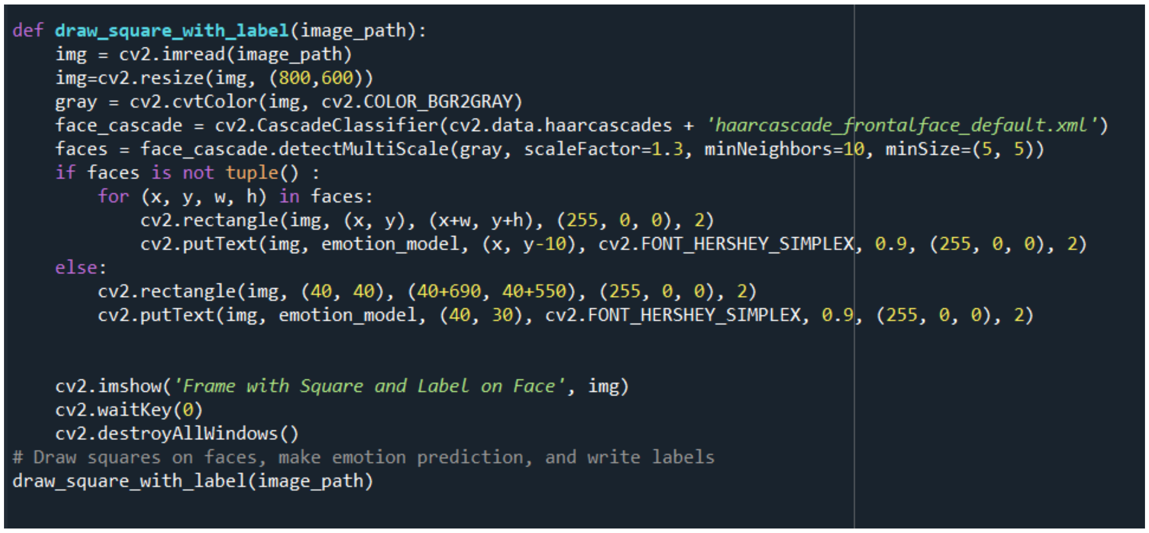


```
pred = emotion_model.predict(img)
emotion_model = label[pred.argmax()]
print("model prediction is ",emotion_model)
```

Explanation:

✓ Uses the pre-trained model to predict the emotion in the preprocessed test image.
✓ "Pred" contains the predicted probabilities for each class.
✓ "Argmax" is used to get the index of the class with the highest probability.
✓ The corresponding emotion label is retrieved from the "label" list.

- Face Detection and Visualization and display Image with Rectangles and Labels:



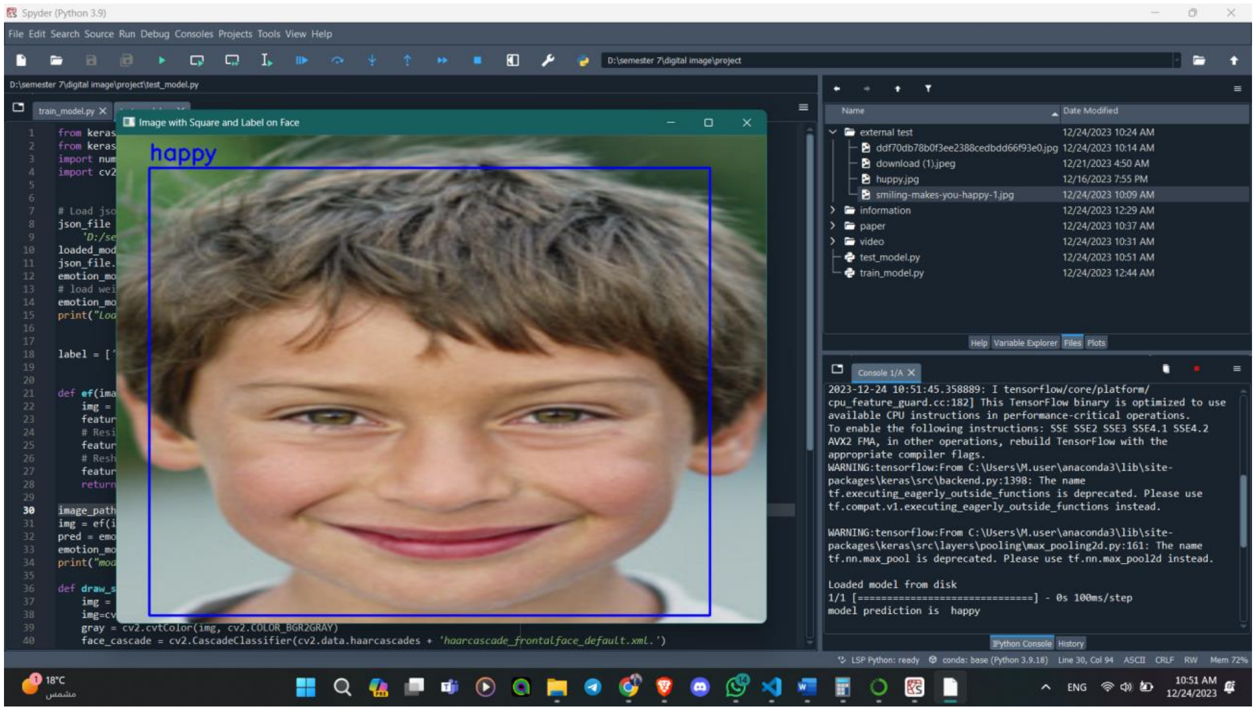
```
def draw_square_with_label(image_path):
    img = cv2.imread(image_path)
    img=cv2.resize(img, (800,600))
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.3, minNeighbors=10, minSize=(5, 5))
    if faces is not tuple() :
        for (x, y, w, h) in faces:
            cv2.rectangle(img, (x, y), (x+w, y+h), (255, 0, 0), 2)
            cv2.putText(img, emotion_model, (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 0, 0), 2)
    else:
        cv2.rectangle(img, (40, 40), (40+690, 40+550), (255, 0, 0), 2)
        cv2.putText(img, emotion_model, (40, 30), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (255, 0, 0), 2)


    cv2.imshow('Frame with Square and Label on Face', img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
# Draw squares on faces, make emotion prediction, and write labels
draw_square_with_label(image_path)
```

Explanation:

✓ Reads the image using OpenCV's "cv2.imread" function.
✓ Resizes the image to dimensions (800, 600) using "cv2.resize".
✓ Converts the color image to grayscale using "cv2.cvtColor".
✓ Initializes a Haar cascade classifier for detecting frontal faces.
✓ Applies face detection to the grayscale image using "detectMultiScale".
✓ "faces" contains the coordinates (x, y, width, height) of detected faces.
✓ Checks if faces were detected ("faces" is not an empty tuple).
✓ Iterates through each detected face and draws a blue rectangle around it using "cv2.rectangle".
✓ Annotates the rectangle with the predicted emotion label using "cv2.putText".
✓ If no faces are detected, a default rectangle is drawn on the image.
✓ Annotates the rectangle with the predicted emotion label.
✓ Displays the image with drawn rectangles and labels and Waits for a key press before closing the image window.
✓ And in the end just call function "draw_square_with_label()"

Results and Discussion

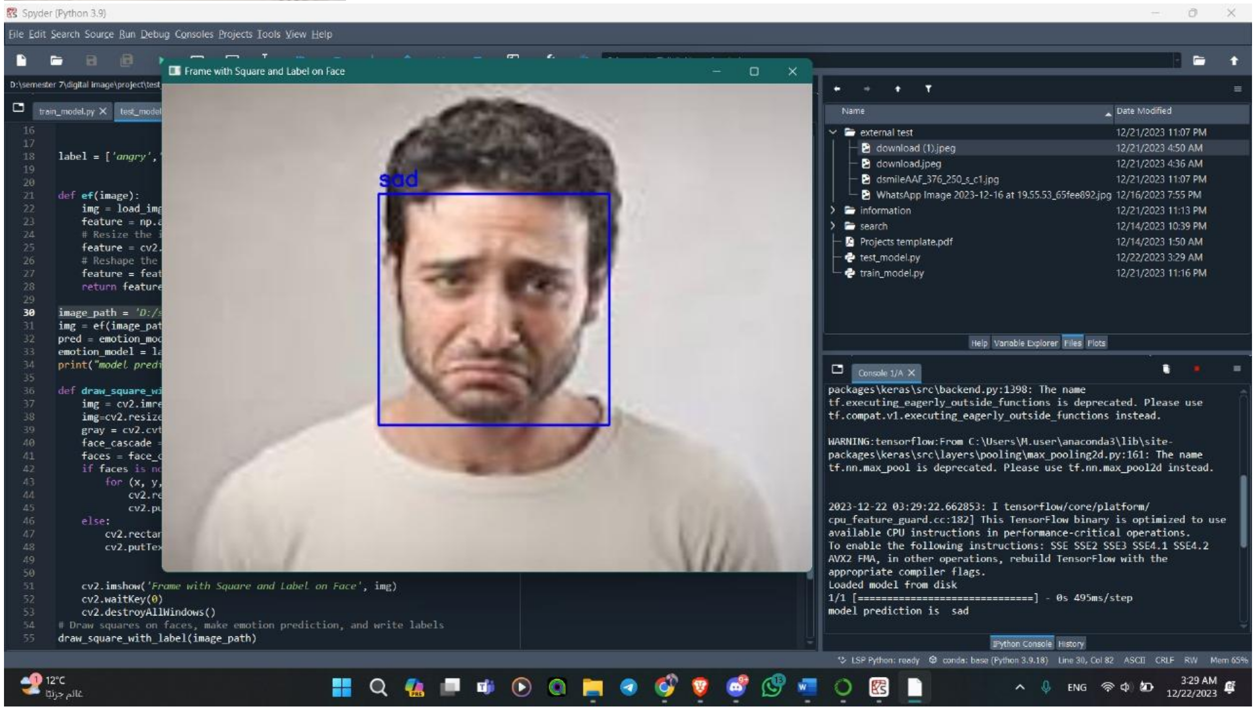
Analysis of Results:


Spyder (Python 3.9)
File Edit Search Source Run Debug Consoles Projects Tools View Help
D:\semester 7\digital image\project
D:\semester 7\digital image\project\test_model.py
train_model.py
Image with Square and Label on Face
happy
Help Variable Explorer Files Plots
Console 1/A
IPython Console History
LSP Python: ready   conda: base (Python 3.9.18)   Line 30, Col 94   ASCII   CRLF   RW   Mem 72%
18°C
مشمس
ENG
10:51 AM
12/24/2023

```
from keras
from keras
import num
import cv2


# Load jso
json_file
    'D:/se
loaded_mod
json_file.
emotion_mo
# load wei
emotion_mo
print("Loa


label = ['


def ef(ima
    img =
    featur
    # Resi
    featur
    # Resh
    featur
    return

image_path
img = ef(i
pred = emo
emotion_mo
print("mod

def draw_s
    img =
    img=cv
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml.')
```

```
Name                                      Date Modified
external test                             12/24/2023 10:24 AM
    ddf70db78b0f3ee2388cedbdd66f93e0.jpg  12/24/2023 10:14 AM
    download (1).jpeg                     12/21/2023 4:50 AM
    huppy.jpg                             12/16/2023 7:55 PM
    smiling-makes-you-happy-1.jpg         12/24/2023 10:09 AM
information                               12/24/2023 12:29 AM
paper                                     12/24/2023 10:37 AM
video                                     12/24/2023 10:31 AM
test_model.py                             12/24/2023 10:51 AM
train_model.py                            12/24/2023 12:44 AM
```

```
2023-12-24 10:51:45.358889: I tensorflow/core/platform/
cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use
available CPU instructions in performance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2
AVX2 FMA, in other operations, rebuild TensorFlow with the
appropriate compiler flags.
WARNING:tensorflow:From C:\Users\M.user\anaconda3\lib\site-
packages\keras\src\backend.py:1398: The name
tf.executing_eagerly_outside_functions is deprecated. Please use
tf.compat.v1.executing_eagerly_outside_functions instead.

WARNING:tensorflow:From C:\Users\M.user\anaconda3\lib\site-
packages\keras\src\layers\pooling\max_pooling2d.py:161: The name
tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.

Loaded model from disk
1/1 [==============================] - 0s 100ms/step
model prediction is  happy
```



```
label = ['angry','

def ef(image):
    img = load_img
    feature = np.a
    # Resize the i
    feature = cv2.
    # Reshape the
    feature = feat
    return feature

image_path = 'D:/s
img = ef(image_pat
pred = emotion_moc
emotion_model = la
print("model predi

def draw_square_wi
    img = cv2.imre
    img=cv2.resize
    gray = cv2.cvt
    face_cascade =
    faces = face_c
    if faces is nc
        for (x, y,
            cv2.re
            cv2.pu
    else:
        cv2.rectar
        cv2.putTex


    cv2.imshow('Frame with Square and Label on Face', img)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
# Draw squares on faces, make emotion prediction, and write labels
draw_square_with_label(image_path)
```

```
packages\keras\src\backend.py:1398: The name
tf.executing_eagerly_outside_functions is deprecated. Please use
tf.compat.v1.executing_eagerly_outside_functions instead.

WARNING:tensorflow:From C:\Users\M.user\anaconda3\lib\site-
packages\keras\src\layers\pooling\max_pooling2d.py:161: The name
tf.nn.max_pool is deprecated. Please use tf.nn.max_pool2d instead.


2023-12-22 03:29:22.662853: I tensorflow/core/platform/
cpu_feature_guard.cc:182] This TensorFlow binary is optimized to use
available CPU instructions in performance-critical operations.
To enable the following instructions: SSE SSE2 SSE3 SSE4.1 SSE4.2
AVX2 FMA, in other operations, rebuild TensorFlow with the
appropriate compiler flags.
Loaded model from disk
1/1 [==============================] - 0s 495ms/step
model prediction is  sad
```

Performance Evaluation:

Performance of algorithm and code:

The model takes a lot of time to run and training around 3 to 4 hours and that’s not acceptable.

Also, the resources utilization is an important factor because is it a high that’s decrease the time for the run.

So, to solve this problem we save the model in a JSON file to call it when needed and that’s take little time may be 3 seconds.

Accuracy:

the accuracy for the model reached : 0.8988 and this accuracy may be acceptable.

Challenges and Resolutions:

1.Data Preprocessing Challenges:

Challenge: Ensuring the quality and consistency of the training and testing datasets.

Resolution: Rigorous data preprocessing, handling missing or inconsistent data, and using data augmentation techniques to increase the diversity of the training dataset.

2.Model Training Challenges:

Challenge: Optimizing the model architecture and hyperparameters for better performance.

Resolution: Experimenting with different neural network architectures, adjusting hyperparameters, and utilizing techniques like learning rate schedules to fine-tune the model.

3.Overfitting:

Challenge: Dealing with overfitting, where the model performs well on training data but poorly on new, unseen data.

Resolution: Implementing regularization techniques such as dropout, early stopping, and using validation data during training to monitor and prevent overfitting.

4.resources utilization:

Challenge: the model when running on CPU takes a lot of time can be reattached to 4 hours.

Resolution: when using GPU by TensorFlow-GPU the model just takes one hour.

Conclusion

In summary, this project successfully implemented a facial emotion recognition model using Convolutional Neural Networks (CNNs) and the main Keras libraries. The model achieved the specified accuracy on the training and validation sets, providing a robust solution. The process included data preprocessing, model training, and generation of JSON and HDF5 files to store the structure and weights, respectively.

Convolutional Neural Networks leverage these algorithms to automatically learn hierarchical features from images, enabling them to perform tasks like image classification, object detection, and segmentation with remarkable accuracy. The combination of convolutional layers, pooling layers, activation functions, and fully connected layers, trained through backpropagation and gradient descent, forms the foundation of CNNs.

By evaluating achievements against goals, the project achieved its goals and provided a deployable solution for facial emotion recognition. To further enhance the project, future directions include hyperparameter tuning, increasing the dataset for diversity, real-time implementation, user interface development, and benchmarking against existing methods. These improvements aim to improve the performance of the model and adapt it to broader applications in real-world scenarios.

References

[1] S. S. S. P. A. K. P. Nalina Matang, "Academia.edu," September 2016. [Online]. Available: https://www.academia.edu/29632929/A_Facial_Expression_Recognition_System_A_Project_Report.

[2] M. O. C. A. E. W. I. O. T. K. I. Student, "IRJMETS," July 2023. [Online]. Available: https://www.irjmets.com/uploadedfiles/paper//issue_7_july_2023/43659/final/fin_irjmets1690713477.pdf.

[3] S. C. S. M. M. A. K. ANGANA MITRA, "Rcciit," 14 May 2018. [Online]. Available: https://www.rcciit.org/students_projects/projects/it/2018/GR8.pdf.

[4] S. S. V. S. V. M. J. V. V. K. R. ,. G. J.S V Sai Kiran, "Anil Neerukonda Institute of Technology and Sciences," 2019-2020. [Online]. Available: http://ece.anits.edu.in/2019-20%20BE%20Project%20REPORTS/vvkr_GG_2019-20_sec(B)_2.pdf.

Appendices

Additional Information:

1. Datasets:
   Source:
   https://www.kaggle.com/datasets/jonathanoheix/face-expression-recognition-dataset

   https://www.kaggle.com/datasets/msambare/fer2013

   Number of Samples: Train[57,530] Test[15,118]
   Number of Classes:[7 emotional]
   Preprocessing Steps: Rescales pixel values between [0][1], Resizes the image to 48x48 pixels.

2. Test Results
   Accuracy: 0.8988
   Validation loss: 0.3323
   Validation accuracy: 0.9176