Assignment 3: How to create your own Facial Recognition Model

What You Will Build:
In this assignment, you will write Python code to:

1. Collect facial expression images using your webcam
2. Train a CNN (Convolutional Neural Network) to recognize smiles
3. Evaluate and save your model

Before we begin making our own facial recognition
Here is a little background information about OpenCV, facial recognition, and image processing.

Please watch the following videos:
OpenCV and Facial Recognition
Convolution

Step 0: Creating File

In your folder create a file called smile_training.py, make sure you are in your virtual environment and in the correct folder.

Step 1: Setup and Imports

```
import tensorflow as tf
from tensorflow.keras import layers, models
import numpy as np
import cv2
import os
from sklearn.model_selection import train_test_split

# Set random seed for reproducibility
tf.random.set_seed(42)
np.random.seed(42)
```

Step 2: Create CNN Model

Paste this block and read the comments:

```
# Part 2: Create CNN Model
def create_model():
    """Create a CNN model for smile detection"""
    model = models.Sequential([
        # First Convolutional Layer
        layers.Conv2D(32, (3, 3), activation='relu', input_shape=(64, 64,
1)),
        layers.MaxPooling2D((2, 2)),

        # Second Convolutional Layer
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),

        # Third Convolutional Layer
        layers.Conv2D(64, (3, 3), activation='relu'),
        layers.MaxPooling2D((2, 2)),

        # Flatten layer
        layers.Flatten(),

        # Dense layers
        layers.Dense(128, activation='relu'),
        layers.Dropout(0.5),
        layers.Dense(1, activation='sigmoid')
    ])
    return model
```

Step 3: Collect Training Data

```
def collect_training_data():
    """Collect training data using webcam"""
    cap = cv2.VideoCapture(0)
    face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades +
'haarcascade_frontalface_default.xml')

    # Create directories for storing images
    os.makedirs('smile_data/smiling', exist_ok=True)
    os.makedirs('smile_data/not_smiling', exist_ok=True)

    smiling_count = 0
    not_smiling_count = 0

    print("Press 's' to capture smiling images")
    print("Press 'n' to capture not smiling images")
    print("Press 'q' to quit")

    while True:
        ret, frame = cap.read()
        if not ret:
            break

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = face_cascade.detectMultiScale(gray, 1.3, 5)

        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)
            face_roi = gray[y:y+h, x:x+w]

        cv2.imshow('Capture Training Data', frame)
        key = cv2.waitKey(1) & 0xFF
```

```
        if key == ord('s') and len(faces) > 0:
            face_roi = cv2.resize(face_roi, (64, 64))

cv2.imwrite(f'smile_data/smiling/smile_{smiling_count}.jpg',
face_roi)
            smiling_count += 1
            print(f"Captured smiling image {smiling_count}")

        elif key == ord('n') and len(faces) > 0:
            face_roi = cv2.resize(face_roi, (64, 64))

cv2.imwrite(f'smile_data/not_smiling/not_smile_{not_smiling_count}.jp
g', face_roi)
            not_smiling_count += 1
            print(f"Captured not smiling image {not_smiling_count}")

        elif key == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
    return smiling_count, not_smiling_count
```

Step 4: Prepare Dataset

Paste this block and think about the TODO question. (TODO question is in the second to last line in the following code block.)

```
def prepare_dataset():
    """Prepare the dataset for training"""
    X = []
    y = []

    smile_dir = 'smile_data/smiling'
    for img_name in os.listdir(smile_dir):
        img_path = os.path.join(smile_dir, img_name)
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        img = img / 255.0  # Normalize pixel values
        X.append(img)
        y.append(1)

    not_smile_dir = 'smile_data/not_smiling'
    for img_name in os.listdir(not_smile_dir):
        img_path = os.path.join(not_smile_dir, img_name)
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        img = img / 255.0  # Normalize pixel values
        X.append(img)
        y.append(0)

    X = np.array(X)
    y = np.array(y)

    # Reshape for CNN input
    X = X.reshape(-1, 64, 64, 1)

    # TODO: What would happen if we changed test_size from 0.2 to 0.5?
    return train_test_split(X, y, test_size=0.2, random_state=42)
```

Part 5: Train and Save the Model

```
def train_model():
    """Train the smile detection model"""
    print("Let's collect training data...")
    smiling_count, not_smiling_count = collect_training_data()
    print(f"\nCollected {smiling_count} smiling images and
{not_smiling_count} not smiling images")

    print("\nPreparing dataset...")
    X_train, X_test, y_train, y_test = prepare_dataset()

    print("\nCreating and compiling model...")
    model = create_model()
    model.compile(optimizer='adam',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])

    print("\nTraining model...")
    history = model.fit(X_train, y_train,
                        epochs=20,
                        batch_size=32,
                        validation_data=(X_test, y_test))

    model.save('smile_detector.h5')
    print("\nModel saved as 'smile_detector.h5'")

    test_loss, test_acc = model.evaluate(X_test, y_test)
    print(f"\nTest accuracy: {test_acc:.4f}")

    return model, history
```

Step 6: Main Function

```
if __name__ == "__main__":
```

```
    model, history = train_model()
```

```

```

Final Step: How to Run the Code

Important: Remember to save your code

After pasting **all the code parts together** into one file (`smile_training.py`), follow these steps:

Run the script from the terminal:

```
python smile_training.py
```

Follow the on-screen instructions:

- Press s to capture smiling images.
- Press n to capture not-smiling images.
- Press q to quit the data collection.

Take around 100 photos each for both smiling and not smiling, you can move your face around (move left, move right, closer to the camera, farther away from the camera, fo under different lighting, etc). You can also have your friends or classmates take pictures of their faces. Repeated faces are okay in this case (in real life, you probably would want more diverse data), but since you are the one testing the actual code, it will be okay. Personally, I took 100 pictures of myself smiling and not smiling.

After you finish collecting the data, press q and you model will start to train itself using all the photos you've taken.

Testing your model

Open this Google Doc on your Raspberry Pi and download a test file I've created for you all onto your Raspberry Pi. The testing file will be in this Google Drive folder . The file is called test_train_model.py. Save this file into the folder where the smile_training.py is located.

Run the script from the terminal:

```
python test_train_model.py
```

And smile at your camera :D