Project Development Summary for PDF

🎭 Mood Detection Using Deep Learning - Project Summary

The **Mood Detection Project** was developed to classify facial expressions in real-time or from images using a deep learning model. The project was implemented using **MobileNetV2**, a lightweight convolutional neural network (CNN) architecture, which was fine-tuned on a custom dataset containing 7 categories of emotions such as **Happy, Angry, Sad, Neutral, Fear, Disgust, and Surprise**.

📚 Step 1: Dataset Preparation and Preprocessing

To begin with, a dataset was created and stored inside a folder named `training` that contained 7 subfolders labeled `0, 1, 2, 3, 4, 5, 6`, where each folder represented a different emotion. Using OpenCV and NumPy, the dataset was preprocessed by converting images to grayscale, resizing them to 224x224 pixels (as required by MobileNetV2), and normalizing pixel values to the range `[0, 1]`. The data was split into training and validation sets to ensure the model generalized well.

🧠 Step 2: Model Selection and Training

MobileNetV2 was chosen due to its efficiency and high accuracy for image classification tasks. The pre-trained model was loaded with `imagenet` weights and fine-tuned on the custom dataset. The model architecture was modified by adding a GlobalAveragePooling layer followed by a Dense layer with 7 output neurons representing the emotions. The model was trained using the `categorical_crossentropy` loss function and the Adam optimizer. After training, the model achieved an accuracy of around **35%** due to the limited size of the dataset. The model was saved as `mobilenetv2_trained_model.h5` for further use.

🎥 Step 3: Real-Time Mood Detection Setup

For real-time emotion detection, Haar Cascades were used to detect faces from a video feed. Once a face was detected, the region of interest (ROI) was cropped, resized to 224x224 pixels, and fed to the trained model. The predicted emotion was displayed on the screen along with a rectangle drawn around the detected face.

To enhance the user experience, a **side monitor panel** was added to display a continuous log of predictions during the real-time detection. This panel was dynamically updated with the latest predictions, providing an intuitive view of mood changes.

📝 Step 4: Building the Jupyter Notebook

The entire process was documented in a Jupyter Notebook (`facial_expression.ipynb".ipynb`), including:

- Preprocessing steps
- Model training and evaluation
- Real-time video mood detection
- Visualization of predictions and logs

The notebook allows easy reproducibility of results and facilitates further improvements if required.

Here's a **detailed step-by-step guide** on how to download the required files and proceed with the Mood Detection Project.

Step 1: Download the Required Files

1. Download the FER-2013 Dataset from Kaggle

This dataset contains images labeled with different emotions such as **Angry, Happy, Sad, Neutral, etc.**

📌 **Follow these steps to download:**

1. Open your web browser and go to: FER-2013 Kaggle Dataset
2. Click on the **"Download"** button (You may need to sign in to Kaggle).
3. Extract the downloaded `.zip` file to access the **CSV file (fer2013.csv)**.

---

2. Download the Haar Cascade XML File

This file is required for **face detection** using OpenCV.

📌 **Follow these steps to download:**

1. Open the following GitHub link: Haar Cascade File
2. Click on the **"Raw"** button.
3. Right-click and select **"Save As"** to download the file.
4. Store it in a folder where your project code is located.

Step 2: Set Up Your Environment

Before running the project, ensure you have installed the necessary dependencies.

📌 **Run the following commands in your Jupyter Notebook or terminal:**

pip install tensorflow keras opencv-python numpy matplotlib pandas

---

Step 3: Load and Preprocess the Data

1. **Read the dataset (fer2013.csv)**
   Use Pandas to read the CSV file and extract image pixel data.
2. **Reshape the data**
   Convert the pixel values into NumPy arrays and reshape them to fit the model.
3. **Normalize the data**
   Scale pixel values between 0 and 1.

Example Code:

```
import pandas as pd
import numpy as np
import cv2
import matplotlib.pyplot as plt

# Load dataset
df = pd.read_csv("fer2013.csv")

# Convert pixel values to arrays
X = np.array([np.fromstring(pixels, dtype=int, sep=' ') for pixels in df['pixels']])
X = X.reshape(-1, 48, 48, 1) / 255.0  # Normalize
```

Step 4: Train the Model

We use **MobileNetV2** (or another CNN model) to classify facial expressions.

Example:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense

# Define CNN Model
model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(48,48,1)),
    MaxPooling2D(2,2),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(7, activation='softmax')  # 7 classes for 7 emotions
])

# Compile Model
model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

```
# Train Model
model.fit(X_train, y_train, epochs=20, validation_data=(X_test, y_test))
```

Step 5: Real-Time Mood Detection

1. **Use OpenCV to capture frames from the webcam.**
2. **Detect faces using Haar Cascade.**
3. **Predict emotion using the trained model.**

Example:

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
cap = cv2.VideoCapture(0)

while True:
    ret, frame = cap.read()
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.1, 4)

    for (x, y, w, h) in faces:
        face_roi = gray[y:y+h, x:x+w]
        face_resized = cv2.resize(face_roi, (48,48)).reshape(1,48,48,1) / 255.0
        prediction = model.predict(face_resized)
        emotion_label = np.argmax(prediction)

        cv2.putText(frame, str(emotion_label), (x, y-10), cv2.FONT_HERSHEY_SIMPLEX, 0.9,
(36,255,12), 2)
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 0, 0), 2)

    cv2.imshow('Mood Detector', frame)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

cap.release()
cv2.destroyAllWindows()
```

Final Notes

- This project **detects human faces in real-time and classifies emotions** using a trained CNN model.
- Ensure that you **have the dataset, Haar Cascade XML, and trained model file** before running the program.
- You can increase **epochs and adjust the model architecture** to improve accuracy.

Now, you can **copy-paste** this into a README file for GitHub! 😊 🚀