PRODIGY_ML_03

TASK_03

1.Install Libraries Using pip:

```
pip install tensorflow keras numpy opencv-python scikit-learn matplotlib
```

2. Mount Google Drive in Colab:

```
from google.colab import drive

drive.mount('/content/drive')
```

3. Get the Path to the Folder in Colab:

```
data_dir =
'/content/drive/MyDrive/GestureRecognitionDataset/leapGestRecog'
```

4. List the Files and Folders:

```
import os

# Check if the path exists
if os.path.exists(data_dir):
    print(f"Directory exists: {data_dir}")
    print("Subdirectories:")
    print(os.listdir(data_dir))
else:
    print(f"Directory not found: {data_dir}")
```

5.Code:

```
import os
import cv2
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelBinarizer
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D,
Flatten, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
```

6.Set the directory path:

```
data_dir =
'/content/drive/MyDrive/GestureRecognitionDataset/leapGestRecog'
gestures = ['01_palm', '02_l', '03_fist', '04_fist_moved', '05_thumb',
'06_index', '07_ok', '08_palm_moved', '09_c', '10_down']
image_size = 64

def load_images_from_folder(folder, label):
    images = []
    for filename in os.listdir(folder):
        img_path = os.path.join(folder, filename)
        img = cv2.imread(img_path, cv2.IMREAD_GRAYSCALE)
        if img is not None:
            img = cv2.resize(img, (image_size, image_size))
            images.append((img, label))
    return images

def load_dataset(data_dir):
    images = []
    for idx, gesture in enumerate(gestures):
        folder_path = os.path.join(data_dir, f'{str(idx).zfill(2)}')
```

```
        if not os.path.exists(folder_path):
            print(f"Folder {folder_path} does not exist. Skipping.")
            continue

        subfolders = os.listdir(folder_path)

        if not subfolders:
            print(f"No subfolders found in {folder_path}. Skipping.")
            continue

        for subfolder in subfolders:
            subfolder_path = os.path.join(folder_path, subfolder)
            if os.path.isdir(subfolder_path):
                images += load_images_from_folder(subfolder_path, idx)
            else:
                print(f"{subfolder_path} is not a directory. Skipping.")

    return images
```

7.Load and preprocess data:

```
dataset = load_dataset(data_dir)
if not dataset:
    raise ValueError("Dataset could not be loaded. Please check the directory paths.")

X, y = zip(*dataset)
X = np.array(X).reshape(-1, image_size, image_size, 1) / 255.0
y = np.array(y)
```

8.Encode labels to one-hot vectors:

```
lb = LabelBinarizer()
y = lb.fit_transform(y)
```

9.Split data into training and testing sets:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

10.Build the CNN model:

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(image_size, image_size, 1)),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    Conv2D(128, (3, 3), activation='relu'),
    MaxPooling2D(pool_size=(2, 2)),

    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dense(10, activation='softmax')
])

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

11.Train the model:

```
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
```

12.Evaluate the model:

```
test_loss, test_accuracy = model.evaluate(X_test, y_test)
```

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

13.Plot the training history:

```
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Test Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
plt.show()

plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Test Loss')
plt.title('Model Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

14.Save the model:

```
model.save('/content/drive/MyDrive/your-folder/gesture_recognition_model.h5' )
```