Introduction to CNN with Python

Face Recognition

Thiha Htun
2312172
Supervisor- Dr. Debbrota Paul Chowdhury

Introduction

Face Recognition

- Face recognition is a biometric technique that identifies or verifies a person from an image or video.
- It plays a key role in modern security systems, authentication, and surveillance.
- Traditional methods struggle with variations in lighting, pose, and occlusion.
- Deep learning has significantly improved accuracy using Convolutional Neural Networks (CNNs).
- This project uses a CNN-based model to detect and recognize human faces from images.

Importing Libraries

```
import os #used to interact with the operating system such as creating directories and reading file path
import cv2 # importing the OpenCV, computer vision library to handle images and videos
import pandas as pd #importing panda library to manipulate csv files
import tensorflow as tf #TensorFlow library-building and training models
from tensorflow import keras # keras API-help to build Deep Learning models using layers.
from tensorflow.keras import layers #layers module-building layers
from sklearn.model_selection import train_test_split #used for splitting train and test from dataset to evaulate model's performance
```

Loading CSV File and Images

```
csv_path = "/Users/thihahtun/Algo/Faces_rec/Dataset.csv"     # locate the path of the csv file from Dataset
df = pd.read_csv(csv_path) #Load the csv file into DataFrame
df.head() #print the first 5 rows for verification
X = [] #list to store the features of Images
y  = [] #list to store the labels of images
face_folder = "/Users/thihahtun/Algo/Faces_rec/faces/faces" # locate to the faces folder containing images
IMG_SIZE = (128,128) #for resizing all images to (128,128)pixels

for idx, row in df.iterrows(): #iterate each row from dataframe
    image_path = os.path.join(face_folder,row['id'])# construct full image path by using id rows from csv file
    if os.path.exists(image_path): #check that path either exists
        img = cv2.imread(image_path) #read image using csv
```

Loading CSV File and Images

```
        img = cv2.resize(img, IMG_SIZE) #resize to 128,128 pixels
        img = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY) #change to
            grayscale
        img = img/255.0 #normalise between 0 and 1
        X.append(img) #append to list X after editing
        y.append(row['label']) #append labels to List y
X = np.array(X, dtype =np.float32) #convert X to numpy array
X = X.reshape(-1,128,128,1) #reshape to match with model input (
    input,width,height,channels)
y = np.array(y) #convert y labels to numpy array
```

Conversion of y labels to Numeric

```
from sklearn.preprocessing import LabelEncoder #importing label
    encoder
label_encoder = LabelEncoder() #create an instance of encoder
y_encoded = label_encoder.fit_transform(y) #converting string
    labels to numeric labels
num_classes = len(label_encoder.classes_) #count the total
    number of unique of classes
```

Model Building

```
model = keras.Sequential([
    layers.Conv2D(32,3,activation ='relu',input_shape
        =(128,128,1)), #first conv layer with (3,3) 32 kernels,
        ReLu =max(0,m)
    layers.MaxPooling2D(), #reduce size by (2,2) pooling

    layers.Conv2D(64,3,activation ='relu'), #second conv layer
        with 64 filters
    layers.MaxPooling2D(), #reduce size by (2,2) pooling

    layers.Flatten(), #flatten the features of 2D into 1D for
        Dense layer
    layers.Dense(128,activation='relu'), #fully connected layer
        with 128 filters and activation=ReLu
    layers.Dense(num_classes,activation = 'softmax') #output
        layer with the number of unique classes, softmax for
        categorical classification
```

Training Model

```
X_train,X_test,y_train,y_test = train_test_split(X,y_encoded,
    test_size=0.3,random_state=42) #split the dataset to trian
    set and test set=20% and ensures the split is same every time
model.compile(
    optimizer ="adam", #adam optimizer is used for automatic
        updation of learning rate
    loss = "sparse_categorical_crossentropy", #suitable for
        integer_encoded labels
    metrics = ["accuracy"] #keep track accuracy in each epoch
)
model.fit(X_train,y_train,epochs=10) #train the model with
    training features and labels for 10 epochs
Epoch 9/10
57/57 7s 124ms / step - accuracy: 0.9902 - loss: 0.0761
Epoch 10/10
57/57 7s 128ms / step - accuracy: 0.9989 - loss: 0.0258
```

Model Prediction

```
model.evaluate(X_test,y_test) #test model on test set
y_pred = model.predict(X_test) #predict the probabilities of each class on test
y_pred_classes = np.argmax(y_pred, axis=1) #get the index of highest probabilities
predicted_labels = label_encoder.inverse_transform(y_pred_classes) #convert the predicted labels to the original labels
for i in range(5): #iterate for 5 images
    plt.imshow(X_test[i].reshape(128, 128), cmap='gray') #dispaly, resize and convert to gray scale
    plt.title(f"Predicted: {predicted_labels[i]} | Actual: {label_encoder.inverse_transform([y_test[i]])[0]}") #label the images
    plt.axis('off') #hide the axis
    plt.show() #show images
```

Model Prediction of the Image

```
img_path = '/Users/thihahtun/Algo/Akshay Kumar_25.jpg' #path to the image that I downloaded
img = cv2.imread(img_path) #using openCV, read image
img = cv2.resize(img, IMG_SIZE) #resize the image
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) #convert color to Gray
img = img / 255.0 #normalize to (0,255)
img = img.reshape(1, 128, 128, 1) #reshape image to match with model input
import matplotlib.pyplot as plt #import matplotlib for visualization
plt.imshow(img.reshape(128, 128), cmap='gray') #reshape back to 2D
plt.title("Preprocessed Input Image") #set title of image
plt.axis('off')  # Hides the axis
plt.show() #show image
```

Model Prediction of the Image

```
pred = model.predict(img) #predict the image
predicted_label = label_encoder.inverse_transform([np.argmax(
    pred)])[0] #change the label back to original label
print("Predicted Label:", predicted_label) #disply the result
    prediction
Predicted Label: Akshay Kumar #output
```

Final Thoughts

Conclusion

- From this task, I learned how to properly preprocess the images before feeding to the Deep Learning model.
- This hands-on step taught me how crucial preprocessing is for reliable results in face recognition tasks, and it reinforced how CNNs rely heavily on input consistency and proper data preparation
- I also understood the importance of checking the image visually to catch issues like wrong color channels, wrong orientation, or unexpected formats.

Thank you!

thiha_ug_23@cse.nits.ac.in

Computer Science and Engineering Department