Department of Computer Engineering

**Answers**

**Viva**

**Coding Efficiency**

**Timely Completion**

**Total**

**5**

**5**

**5**

**5**

**20**

**Start Date** :..

Course: Laboratory Practice V

**Dated Sign of Subject Teacher**

Date of Completion:............

**Mini Project -1**

**Title - Implement Human Face Recognition**

**Project title: "Celebrity Face Recognition"**

**Objective: To** build a **deep** neural network model **that** can recognize the identities of celebrities in the

**"CelebA" dataset**.

**Theory -** Human **face** recognition using **deep** neural **networks involves building** a neural network model that

**can** take an image **of a** human **face as** input and **accurately recognize the person** in **the** image**.** Here is a sample

code using TensorFlow to implement a basic face **recognition** system using a deep neural network:

Human face recognition **using** deep neural **networks** (DNNs**) involves** training a neural **network to** identify and

distinguish between different faces. The process typically involves **the** following **steps**:

**Data collection:** A **large dataset of face images is** collected, including **images** of different individuals and

**under** different lighting **and** pose conditions.

**Data preprocessing:** The face images **are** preprocessed to remove noise, **align the** faces**, and normalize the**

**illumination**.

**Feature extraction:** The preprocessed face images are **then** fed into a deep neural network to **extract**

**high**-level features that capture the important characteristics **of a face**. The neural network typically consists **of**

several layers of convolutional and pooling **operations**, followed by fully connected layers that produce **a**

**feature vector**.

Department of Computer Engineering

Course: Laboratory Practice V

**Training: The** extracted **features are** then used to train **the** neural network to distinguish between different

faces**. This** is typically **done using a** supervised learning approach, where **the network is trained on** a labeled

**dataset** of **face images and their corresponding** identities.

**Testing**: **After the neural network has been trained**, it **can be** tested on a **separate dataset to evaluate its**

performance. This typically involves measuring the accuracy of **the** network in correctly identifying **the**

**individuals in the test dataset**.

**Deployment: Once the** neural network **has** been trained and tested, it can be deployed in a real-**world**

application for face recognition. **This typically** involves capturing a **face** image, preprocessing it, and then

**feeding** it into **the** neural **network** to obtain **a** feature vector. **The feature** vector is **then** compared to a **database**

of **known faces** to determine **the** identity **of the** individual **in** the image.

**Overall**, human face recognition using DNNs **is** a complex process **that** requires a large amount of **data**,

sophisticated neural network architectures, and **careful** preprocessing and training. However, with the

increasing availability of large **datasets** and powerful computing resources, DNN**-**based face recognition

systems **have** become **increasingly** accurate and effective in real**-world** applications.

Example- **One** example of human face recognition using DNNs is **the** FaceNet algorithm, which **was**

**developed** by researchers **at** Google in 2015. **FaceNet is a** deep neural network **that** is trained to directly

optimize the **embedding of face images** into a **high**-dimensional **feature** space, where distances between faces

**correspond** to **their similarity**.

**The FaceNet** architecture consists of a **deep** convolutional neural network **that** takes a **raw face image** as input

and produces a **128-**dimensional feature vector as output. The network **is** trained **on** a large **dataset** of face

**images with labels that** correspond to different individuals.

During training, **the FaceNet network is optimized** to minimize **the** distance **between the feature vectors** of

**images that** depict **the same person** and **maximize the distance** between **feature vectors** of **images that** depict

different **people**. **This is** done **using** a loss function **called** the triplet loss**, which** compares **the** distance between

**the** feature vectors **of an** anchor image, a positive **image** (of **the** same **person as the** anchor), **and** a **negative**

Department of Computer Engineering

Course: Laboratory Practice V

**image (of** a different person**).** The goal is to minimize **the distance** between the anchor and positive images

**while maximizing the** distance between **the** anchor and **negative** images.

After training, **the** FaceNet network can be used for face recognition by capturing a face **image**, preprocessing

**it, and** then feeding **it into the** network to obtain a **128-**dimensional feature vector. **This** feature vector **is then**

compared to a **database** of known **faces** by computing **the Euclidean distance between the feature vector** and

**the feature vectors** of **the faces** in **the** database**. The** closest **matching** face **in the database is** then returned **as the**

recognized identity.

**Overall, FaceNet** is an example of a DNN-based **face** recognition **system that has** achieved **high** accuracy **and**

robustness in real-**world** applications. It has been used in a variety of applications, including security systems,

access **control**, and **social** media tagging.

import tensorflow as **tf**

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense**,** Flatten, Conv2D, **MaxPooling2D**, Dropout

from **tensorflow.keras.preprocessing.image** import **ImageDataGenerator**

**import** numpy **as** np

import pandas **as** pd

import **matplotlib.pyplot as** plt

**import** cv2

# **Define constants**

**img\_height** = **128**

**img\_width** = **128**

**batch size = 32**

**epochs** = 10

num **classes**

=

**10**

# **Load the "CelebA" dataset**

df **–** nd road **csv('**list **attr 1**

Department of Computer Engineering

df = df.sample**(frac**=1).reset\_index(drop=True) **#** shuffle **the dataset**

Course: Laboratory Practice V

df[**'image\_id**'] = df[**'**image\_id'].apply(**lambda x: x**[:-4]) # **remove file** extension **from** image IDs

**df\_train** = **df**[:int(len(df)**\*0.8**)] # **80% for** training

df\_val = df[int(len(df)\***0.8**):int(len(df)\*0.9)] # **10%** for **validation**

df\_test = **df[int(**len(df)\*0.9):] # **10%** for **testing**

# Define data **generators** for training**, validation**, and testing sets

**train\_datagen** = ImageDataGenerator(**rescale**=1./255**)**

val\_datagen = ImageDataGenerator**(**rescale=1./**255)**

test\_datagen = ImageDataGenerator**(**rescale=1./255**)**

**train\_generator = train\_datagen.flow\_from\_dataframe(**

**dataframe**=df\_train,

**directory=**'**img\_align\_celeba**',

x\_col='image\_id',

y\_col='**Smiling**',

**target\_size**=(img\_height**,** img\_width**),**

**batch\_size=batch\_size**,

**class** \_mode='binary')

**val\_generator =** val\_datagen.flow\_from\_dataframe(

**dataframe=df\_val,**

directory='img\_align\_celeba',

x\_col='image\_id',

y\_col=**'Smiling**',

target\_size=(img\_height**,** img\_width),

**batch\_size=batch**

Department of Computer Engineering

**class\_mode**='**binary**')

test\_generator = test\_datagen.flow\_from\_dataframe(

**dataframe**=df\_test,

directory='**img\_align\_celeba**',

x\_col='image\_id',

y\_col='**Smiling**',

**target\_size**=(**img\_height, img\_width)**,

**batch\_size**=**batch\_size,**

class\_mode=**'binary**')

# **Define the neural network model**

**model** = Sequential([

**Conv2D**(32, (**3,3)**, **activation='**relu', input\_shape=(img\_height**,** img\_width**, 3))**,

**MaxPooling2D**((**2,2))**,

**Conv2D**(**64**, (3,3**),** activation**='relu**'**)**,

MaxPooling2D**((2,2))**,

Conv2D(128**, (3,3)**, activation='relu'**)**,

**MaxPooling2D**((2,2)),

Conv2D(128**,** (**3,3)**, activation='relu'),

MaxPooling2D((**2,2**)**)**,

Flatten(),

Dropout(0.5),

**Dense(**512**, activation**=**'**relu'),

**Dense(num\_classes,** activation**='sigmoid'**)

# **Compile the model**

model.compile(optimizer=**'adam**',

Course: Laboratory Practice V

Department of Computer Engineering

**loss=**'**binary\_crossentropy**',

metrics**=**[**'**accuracy'])

# **Train the model**

history = model.fit**(**train\_generator,

epochs=epochs,

**validation\_data=**val\_generator)

**#Evaluate the model on the test set**

loss**,** accuracy = model.evaluate(test\_generator)

**print("Test** accuracy:**"**, accuracy**)**

# **Predict the** smiling attribute **of** a sample image

img = cv2.imread**('**sample\_image.jpg**')**

img = cv2.resize(img, (img\_height**,** img\_width**))**

img **= np.expand\_dims(**img**, axis=**0)

**Conclusion- In this way** Human **Face Recognition** Implemented.

**Assignment Question**

1. What do you mean by Exploratory **Analysis?**

2. **What do** you mean by **Correlation Matrix?**

3. **What is Conv2D used for?**

Course: Laboratory Practice V