AI ASSIGNMENT - 3

Step 1: Import the necessary libraries

python

import pandas as pd

import numpy as np

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import StandardScaler

from tensorflow.keras.models import Sequential

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

from tensorflow.keras.preprocessing.image import ImageDataGenerator

Step 2: Load the dataset

python

# Assuming the dataset is stored in a directory named 'dataset' with subdirectories for each class

train\_data\_dir = 'dataset/train' # Path to the training data

test\_data\_dir = 'dataset/test' # Path to the testing data

# Specify the input shape of the images

input\_shape = (64, 64, 3) # Assuming the images are RGB and resized to 64x64 pixels

# Specify the batch size for training and testing

batch\_size = 32

# Create an ImageDataGenerator for data augmentation

train\_datagen = ImageDataGenerator(

rescale=1.0 / 255, # Normalize pixel values to the range [0, 1]

rotation\_range=20, # Randomly rotate the images by 20 degrees

zoom\_range=0.2, # Randomly zoom the images by 20%

horizontal\_flip=True # Randomly flip the images horizontally

)

test\_datagen = ImageDataGenerator(rescale=1.0 / 255) # Normalize pixel values of the test images

# Load and augment the training data

train\_generator = train\_datagen.flow\_from\_directory(

train\_data\_dir,

target\_size=input\_shape[:2],

batch\_size=batch\_size,

class\_mode='binary' # Change to 'categorical' if you have multiple classes

)

# Load and normalize the testing data

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

test\_data\_dir,

target\_size=input\_shape[:2],

batch\_size=batch\_size,

class\_mode='binary' # Change to 'categorical' if you have multiple classes

)

Step 3: Build the CNN model

python

model = Sequential()

model.add(Conv2D(32, (3, 3), activation='relu', input\_shape=input\_shape))

model.add(MaxPooling2D((2, 2)))

model.add(Flatten())

model.add(Dense(64, activation='relu'))

model.add(Dense(32, activation='relu'))

model.add(Dense(1, activation='sigmoid')) # Output layer

model.compile(optimizer='adam', loss='binary\_crossentropy', metrics=['accuracy'])

Step 4: Train the model

python

# Specify the number of steps per epoch and the number of epochs

steps\_per\_epoch = train\_generator.n // batch\_size

epochs = 10

model.fit(train\_generator, steps\_per\_epoch=steps\_per\_epoch, epochs=epochs)

Step 5: Test the model

python

# Evaluate the model on the testing data

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

print('Test loss:', loss)

print('Test accuracy:', accuracy)