1. Handwritten digit classification

import matplotlib.pyplot as plt

from tensorflow.keras.datasets import mnist

(train\_images,train\_labels),(\_,\_)=mnist.load\_data()

num\_images=5

plt.figure(figsize=(8,4))

for i in range(num\_images):

  plt.subplot(4,5,i+1)

  plt.imshow(train\_images[i],cmap='gray')

  plt.title(f"Label:(train\_labels[i])")

  plt.axis('off')

  plt.tight\_layout()

plt.show()

[######## 1.2 ]

import tensorflow as tf

from tensorflow.keras.datasets import mnist

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense,Flatten

import matplotlib.pyplot as plt

(train\_images,train\_labels),(test\_images,test\_labels)=mnist.load\_data()

train\_images,test\_images=train\_images/255.0,test\_images/255.0

model=Sequential([

    Flatten(input\_shape=(28,28)),

    Dense(128,activation='relu'),

    Dense(64,activation='relu'),

    Dense(10,activation='softmax')

])

model.compile(optimizer='adam',

              loss='sparse\_categorical\_crossentropy',

              metrics=['accuracy'])

history=model.fit(train\_images,train\_labels,epochs=5,batch\_size=32,validation\_split=0.2)

test\_loss,test\_acc=model.evaluate(test\_images,test\_labels)

print(f'Test accuracy:(test\_acc)')

plt.plot(history.history['accuracy'])

plt.plot(history.history['val\_accuracy'])

plt.title('model accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend(['train','validation'],loc='upper left')

plt.show()

plt.plot(history.history['loss'])

plt.plot(history.history['val\_loss'])

plt.title('model loss')

plt.xlabel('Epoch')

plt.ylabel('loss')

plt.legend(['train','validation'],loc='upper left')

plt.show()

1. ##### 2

import tensorflow as tf

from tensorflow.keras import layers,models

from tensorflow.keras.datasets import imdb

from tensorflow.keras.preprocessing.sequence import pad\_sequences

import matplotlib.pyplot as plt

(train\_data,train\_labels),(test\_data,test\_labels)=imdb.load\_data(num\_words=10000)

max\_length=256

train\_data=pad\_sequences(train\_data,maxlen=max\_length)

test\_data=pad\_sequences(test\_data,maxlen=max\_length)

model=models.Sequential()

model.add(layers.Embedding(input\_dim=10000,output\_dim=16,input\_length=max\_length))

model.add(layers.GlobalAveragePooling1D())

model.add(layers.Dense(16,activation='relu'))

model.add(layers.Dense(1,activation='sigmoid'))

model.compile(optimizer='adam',

                loss='binary\_crossentropy',

                metrics=['accuracy'])

model.summary()

history=model.fit(train\_data,train\_labels,

                      epochs=10,

                      batch\_size=512,

                      validation\_split=0.2)

test\_loss,test\_acc=model.evaluate(test\_data,test\_labels)

print(f"Test Accuracy:(test\_acc)")

def plot\_history(history):

  plt.plot(history.history['accuracy'],label='training accuracy')

  plt.plot(history.history['val\_accuracy'],label='validaion accuracy')

  plt.xlabel('Epochs')

  plt.ylabel('Accuracy')

  plt.legend()

  plt.show()

plt.plot(history.history['loss'],label='training loss')

plt.plot(history.history['val\_loss'],label='validationLoss')

plt.xlabel('Epochs')

plt.ylabel('loss')

plt.legend()

plt.show()

plot\_history(history)

1. ##### 3 [ News Wires ]

import numpy as np

import tensorflow as tf

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense,Embedding,LSTM,Dropout

from tensorflow.keras.preprocessing.sequence import pad\_sequences

from tensorflow.keras.datasets import reuters

from tensorflow.keras.utils import to\_categorical

import matplotlib.pyplot as plt

(x\_train, y\_train), (x\_test, y\_test)= reuters.load\_data(num\_words=10000)

max\_len = 200

x\_train = pad\_sequences(x\_train, maxlen= max\_len)

x\_test = pad\_sequences(x\_train, maxlen= max\_len)

y\_train=to\_categorical(y\_train)

y\_test=to\_categorical(y\_test)

model=Sequential()

model.add(Embedding(input\_dim=10000,output\_dim=128,input\_length=max\_len))

model.add(LSTM(64,return\_sequences=True))

model.add(LSTM(64))

model.add(Dropout(0.5))

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

model.add(Dense(46,activation='softmax'))

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

history=model.fit(x\_train,y\_train,epochs=3,batch\_size=64,validation\_split=0.2)

loss,accuracy=model.evaluate(x\_test,y\_test)

print(f"Test Loss:{loss:.4f}")

print(f"Test Accuracy:{accuracy:.4f}")

def plot\_history(history):

  plt.figure(figsize=(12,6))

  plt.subplot(1,2,1)

  plt.plot(history.history['accuracy'])

  plt.plot(history.history['val\_accuracy'])

  plt.title('Model Accuracy')

  plt.xlabel('Epoch')

  plt.ylabel('Accuracy')

  plt.legend(['Train','Validation'],loc='upper left')

  plt.subplot(1,2,2)

  plt.plot(history.history['val\_accuracy'])

  plt.title('Model Loss')

  plt.xlabel('Epoch')

  plt.ylabel('Accuracy')

  plt.legend(['Train','Validation'],loc='upper left')

  plt.tight\_layout()

  plt.show()

plot\_history(history)

House Prediction

from sklearn.datasets import fetch\_california\_housing

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

from sklearn.preprocessing import StandardScaler

from tensorflow.keras.models import Sequential

from tensorflow.keras.layers import Dense

from sklearn.metrics import r2\_score

housing=fetch\_california\_housing()

X,y=housing.data,housing.target

scaler=StandardScaler()

x=scaler.fit\_transform(X)

X\_train,X\_test,y\_train,y\_test=train\_test\_split(X,y,test\_size=0.2,random\_state=42)

model=Sequential([

    Dense(64,activation='relu',input\_shape=(X\_train.shape[1],)),

    Dense(64,activation='relu'),

    Dense(1)

])

model.compile(optimizer='adam',loss='mean\_squared\_error')

history=model.fit(X\_train,y\_train,epochs=100,batch\_size=32,validation\_split=0.2,verbose=0)

loss=model.evaluate(X\_test,y\_test)

print(f'Test Loss: {loss}')

y\_pred\_train =model.predict(X\_train)

y\_pred\_test=model.predict(X\_test)

r2\_train=r2\_score(y\_train,y\_pred\_train)

r2\_test=r2\_score(y\_test,y\_pred\_test)

print(f'Training R^2 Score:(r2\_train)')

print(f'Test R^2 Score:(r2\_test)')

import matplotlib.pyplot as plt

plt.plot(history.history['loss'],label='Training Loss')

plt.plot(history.history['val\_loss'],label='Validation Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.legend()

plt.title('Training and Validation Loss')

plt.show()

plt.plot(history.history['val\_loss'],label='Validation R^2')

plt.plot(history.history['loss'],label='Training R^2')

plt.xlabel('Epoch')

plt.ylabel('R^2 score')

plt.legend()

plt.title('Training and Validation R^2 Score')

plt.show()

CNN for handwritten digit classification

import tensorflow as tf

from tensorflow.keras import layers, models

import matplotlib.pyplot as plt

mnist = tf.keras.datasets.mnist

(train\_images, train\_labels), (test\_images, test\_labels) = mnist.load\_data()

train\_images, test\_images = train\_images / 255.0, test\_images / 255.0

train\_images = train\_images.reshape(train\_images.shape[0], 28, 28, 1)

test\_images = test\_images.reshape(test\_images.shape[0], 28, 28, 1)

model = models.Sequential([

    layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(28, 28, 1)),

    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(64, (3, 3), activation='relu'),

    layers.MaxPooling2D((2, 2)),

    layers.Conv2D(64, (3, 3), activation='relu'),

    layers.Flatten(),

    layers.Dense(64, activation='relu'),

    layers.Dense(10, activation='softmax')

])

model.compile(optimizer='adam',

              loss='sparse\_categorical\_crossentropy',

              metrics=['accuracy'])

history = model.fit(train\_images, train\_labels, epochs=5, batch\_size=64, validation\_data=(test\_images, test\_labels))

test\_loss, test\_acc = model.evaluate(test\_images, test\_labels)

print('Test accuracy:', test\_acc)

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend(loc='lower right')

plt.title('Training and Validation Accuracy')

plt.show()

plt.plot(history.history['loss'], label='Training Loss')

plt.plot(history.history['val\_loss'], label='Validation Loss')

plt.xlabel('Epoch')

plt.ylabel('Loss')

plt.legend(loc='upper right')

plt.title('Training and Validation Loss')

plt.show()

CNN for simple image classification

import tensorflow as tf

from tensorflow.keras import layers

model = tf.keras.Sequential([

        layers.Conv2D(32, (3, 3), activation='relu', input\_shape=(150, 150, 3)),

        layers.MaxPooling2D((2, 2)),

        layers.Conv2D(64,(3,3),activation='relu'),

        layers.MaxPooling2D((2,2)),

        layers.Conv2D(128,(3,3),activation='relu'),

        layers.MaxPooling2D((2,2)),

        layers.Flatten(),

        layers.Dense(512,activation='relu'),

        layers.Dense(1,activation='sigmoid')])

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

train\_datagen=tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255)

test\_datagen=tf.keras.preprocessing.image.ImageDataGenerator(rescale=1./255)

train\_generator=train\_datagen.flow\_from\_directory("C:\Users\Administrator\Downloads\image1.jpeg",target\_size=(150,150),batch\_size=20,class\_mode='binary')

test\_generator=test\_datagen.flow\_from\_directory("C:\Users\Administrator\Downloads\image2.jpeg",target\_size=(150,150),batch\_size=20,class\_mode='binary')

history=model.fit(train\_generator,step\_per\_epoch=100,epochs=30,validation\_data=test\_generator,validation\_steps=50 )

test\_loss,test\_acc=model.evaluate(test\_generator,steps=50)

print("test accuracy:",test\_acc)

EX 8 One hot encoding

import numpy as np

text = "hello"

char\_to\_int = {char: i for i, char in enumerate(set(text))}

int\_to\_char = {i: char for char, i in char\_to\_int.items()}

char\_indices = [char\_to\_int[char] for char in text]

one\_hot\_encoded = np.zeros((len(text), len(char\_to\_int)))

for i, index in enumerate(char\_indices):

    one\_hot\_encoded[i, index] = 1

print(one\_hot\_encoded)

from sklearn.preprocessing import OneHotEncoder

words = ['vitw', 'college', 'srk', 'jntuk', 'good']

word\_to\_int = {word: i for i, word in enumerate(set(words))}

int\_to\_word = {i: word for word, i in word\_to\_int.items()}

word\_indices = [word\_to\_int[word] for word in words]

word\_indices = [[index] for index in word\_indices]

encoder = OneHotEncoder(sparse=False)

one\_hot\_encoded = encoder.fit\_transform(word\_indices)

print(one\_hot\_encoded)

EX 9

import matplotlib.pyplot as plt

from keras.datasets import imdb

from keras.models import Sequential

from keras.layers import Embedding, LSTM, Dense

from keras.preprocessing.sequence import pad\_sequences

max\_features = 1000

maxlen = 100

(x\_train, y\_train),(x\_test, y\_test)= imdb.load\_data(num\_words= max\_features)

x\_train= pad\_sequences(x\_train, maxlen= maxlen)

x\_test= pad\_sequences(x\_test, maxlen= maxlen)

model =Sequential()

model.add(Embedding(max\_features, 128, input\_length=maxlen))

model.add(LSTM(64, dropout=0.2, recurrent\_dropout=0.2))

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

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

history=model.fit(x\_train, y\_train, batch\_size=32, epochs=5, validation\_data=(x\_test, y\_test))

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Training and Validation Accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend()

plt.show()

EX 10

import matplotlib.pyplot as plt

from keras.datasets import imdb

from keras.models import Sequential

from keras.layers import Embedding, LSTM, Dense

from keras.preprocessing.sequence import pad\_sequences

max\_features = 1000

maxlen = 100

(x\_train, y\_train),(x\_test, y\_test)= imdb.load\_data(num\_words= max\_features)

x\_train= pad\_sequences(x\_train, maxlen= maxlen)

x\_test= pad\_sequences(x\_test, maxlen= maxlen)

model =Sequential()

model.add(Embedding(max\_features, 128, input\_length=maxlen))

model.add(LSTM(64, dropout=0.2, recurrent\_dropout=0.2))

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

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

history=model.fit(x\_train, y\_train, batch\_size=32, epochs=5, validation\_data=(x\_test, y\_test))

plt.plot(history.history['accuracy'], label='Training Accuracy')

plt.plot(history.history['val\_accuracy'], label='Validation Accuracy')

plt.title('Training and Validation Accuracy')

plt.xlabel('Epoch')

plt.ylabel('Accuracy')

plt.legend()

plt.show()