**Second Programming Exam.**

**Deep Learning, 2022**

姓名：吳佳展 學號：110318517

import tensorflow as tf

import tensorflow.keras as keras

from tensorflow.keras.models import Sequential, Model

from tensorflow.keras.layers import Input, Dense, Dropout, Activation

from tensorflow.keras.layers import BatchNormalization, Flatten

from tensorflow.keras.layers import Conv2D, MaxPooling2D

import pandas as pd

import numpy as np

from sklearn import preprocessing

import matplotlib.pyplot as plt

def Plot\_loss(X, y, mode):

plt.figure(figsize=(12,6))

plt.plot(X)

plt.plot(y)

if mode =='loss':

plt.title('Model loss')

plt.ylabel('loss')

plt.xlabel('epoch')

plt.legend(['Train\_loss', 'Val\_loss'], loc='upper right')

else:

plt.title('Model Accuracy')

plt.ylabel('Accuracy')

plt.xlabel('epoch')

plt.legend(['Train\_ACC', 'Val\_ACC'], loc='lower right')

def Predict():

df\_test = pd.read\_csv('class\_tst.csv',header=None)

X\_data = df\_test.values

X\_data\_normalized = preprocessing.scale(X\_data)

ans = fnn\_model.predict(X\_data\_normalized)

fin\_ans = ans.argmax(-1) + 1

result = pd.concat([df\_test,pd.DataFrame(fin\_ans)],axis=1)

return result

def Export\_csv(data):

cols= [i for i in range(0,9)]

df\_ans.to\_csv('Q1/class\_answer.csv',header=cols,index=None)

# Get data

df = pd.read\_csv('class\_trn.csv',header=None)

X = df.iloc[:,0:8].values

y = df.iloc[:,8].values

# Normalize data

X\_normalized = preprocessing.scale(X)

# split train(80), test, val data(20)

train\_len, test\_len = int(0.8\*len(df)), int(0.2\*len(df))

X\_train, y\_train = X\_normalized[:train\_len], y[:train\_len]

X\_test, y\_test = X\_normalized[train\_len:], y[train\_len:]

y\_train, y\_test = y\_train-1, y\_test - 1

y\_train = y\_train.reshape(len(y\_train),1)

y\_test = y\_test.reshape(len(y\_test),1)

y\_train = np.eye(5, dtype='float32')[y\_train[:, 0]]

y\_test = np.eye(5, dtype='float32')[y\_test[:, 0]]

# Create model

inputs = Input(shape=X\_train.shape[1:])

x = Dense(40, activation='relu')(inputs)

x = Dense(35, activation='relu')(x)

x = Dense(20, activation='relu')(x)

x = Dense(20, activation='relu')(x)

x = Dense(40, activation='relu')(x)

x = Flatten()(x)

outputs = Dense(5, activation='softmax')(x)

fnn\_model = Model(inputs=inputs, outputs=outputs)

fnn\_model.summary()

# Training model

learning\_rate = 1e-4

optimizer = keras.optimizers.Adam(learning\_rate=learning\_rate)

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

history = fnn\_model.fit(X\_train, y\_train, batch\_size=32, epochs=100, validation\_data=(X\_test, y\_test), verbose=2)

fnn\_model.save('Q1/class.h5')

# Plot loss, accuracy picture

Plot\_loss(history.history['loss'], history.history['val\_loss'], 'loss')

Plot\_loss(history.history['accuracy'], history.history['val\_accuracy'], 'acc')

# Use model predict

df\_ans = Predict()

# Export csv

Export\_csv(df\_ans)