**Second Programming Exam.**

**Deep Learning, 2022**

姓名：吳佳展 學號：110318517

from matplotlib import pyplot as plt

import pandas as pd

import numpy as np

from sklearn.preprocessing import MinMaxScaler

from tensorflow.keras.layers import LSTM, Dropout, Dense, Activation, GRU

from tensorflow.keras.models import Sequential, load\_model

from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping

def Get\_data():

df\_train = pd.read\_csv('predict2\_trn.csv')

df\_test = pd.read\_csv('predict2\_tst.csv')

X\_train, y\_train = df\_train.iloc[:,:12], df\_train.iloc[:,-4:]

X\_test = df\_test.iloc[:,:12]

return X\_train, y\_train, X\_test

def Normalize():

df\_train = pd.read\_csv('predict2\_trn.csv')

scaler = MinMaxScaler(feature\_range=(0,1))

scaled = scaler.fit\_transform(df\_train)

X\_scaled, y\_scaled = scaled[:,:12], scaled[:,-4:]

return scaler, X\_scaled, y\_scaled

def Build\_Model(X\_train, y\_train):

# The GRU architecture

regressorGRU = Sequential()

# First GRU layer with Dropout regularisation

regressorGRU.add(GRU(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1],1)))

regressorGRU.add(Dropout(0.2))

# Second GRU layer

regressorGRU.add(GRU(units=50, return\_sequences=True, input\_shape=(X\_train.shape[1],1)))

regressorGRU.add(Dropout(0.2))

# Third GRU layer

regressorGRU.add(GRU(units=50))

regressorGRU.add(Dropout(0.2))

# The output layer

regressorGRU.add(Dense(y\_train.shape[1]))

regressorGRU.summary()

# Compiling the RNN

# regressorGRU.compile(optimizer=SGD(lr=0.01, decay=1e-7, momentum=0.9, nesterov=False),loss='mean\_squared\_error', metrics=['mse'])

regressorGRU.compile(optimizer='adam', loss='mean\_squared\_error', metrics=['mse'])

return regressorGRU

def Train\_Model(X, y):

re\_X = X.reshape(X.shape[0], X.shape[1],1)

file1 = 'Q3/predict2.h5'#儲存訓練的最小誤差

checkpoint = ModelCheckpoint(file1, monitor='val\_loss', verbose=2, save\_best\_only = True, mode='min')

earlyStopping = EarlyStopping(monitor='val\_loss', patience=50, verbose=2, mode='auto')

callbacks\_list= [checkpoint,earlyStopping]

gru\_model = Build\_Model(re\_X, y)

history = gru\_model.fit(re\_X, y, epochs=80, batch\_size=35, callbacks=callbacks\_list, validation\_split=0.1)

# savemodel

gru\_model.save(file1)

return history.history ['loss'], history.history ['val\_loss']

def Predict(model, X\_test):

temp\_x = np.concatenate((X\_test.iloc[:,:].values, X\_test.iloc[:,-4:].values),axis=1)

temp\_x = scaler.fit\_transform(temp\_x)

y\_hat = model.predict(temp\_x[:,:12])

# inverse

all\_test = np.concatenate((temp\_x[:,:12], y\_hat),axis=1)

actual\_test = scaler.inverse\_transform(all\_test)

return y\_hat, actual\_test

def plot\_loss(x, y):

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

plt.plot(x)

plt.plot(y)

plt.title('GRU Model loss')

plt.ylabel('MSE')

plt.xlabel('epoch')

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

def Export\_csv(data):

cols = []

for i in range(-11,5,1):

if i <0 :

cols.append(f't-{i\*(-1)}')

elif i==0:

cols.append('t')

else:

cols.append(f't+{i}')

df\_fn = pd.DataFrame(data)

df\_fn.to\_csv('Q3/predict2\_answer.csv', index=None, header=cols)

# Get data

X\_train, y\_train, X\_test = Get\_data()

# data normalize

scaler, X\_scaled, y\_scaled = Normalize()

# train model

loss, val\_loss = Train\_Model(X\_scaled, y\_scaled)

# plot loss

plot\_loss(loss, val\_loss)

# load model

pre\_model= load\_model('Q3/predict2.h5')

# predict value

y\_hat, actual = Predict(pre\_model, X\_test)

# export csv

Export\_csv = Export\_csv(actual)