import sys

import warnings

import argparse

import numpy as np

import pandas as pd

from data.data import process\_data

from model import model

from keras.models import Model

from keras.callbacks import EarlyStopping

warnings.filterwarnings("ignore")

def train\_model(model, X\_train, y\_train, name, config):

model.compile(loss="mse", optimizer="rmsprop", metrics=['mape'])

# early = EarlyStopping(monitor='val\_loss', patience=30, verbose=0, mode='auto')

hist = model.fit(

X\_train, y\_train,

batch\_size=config["batch"],

epochs=config["epochs"],

validation\_split=0.05)

model.save('C:/Users/DELL/Downloads/traffic flow prediction/TrafficFlowPrediction-master/model/' + name + '.h5')

df = pd.DataFrame.from\_dict(hist.history)

df.to\_csv('C:/Users/DELL/Downloads/traffic flow prediction/TrafficFlowPrediction-master/model/' + name + ' loss.csv', encoding='utf-8', index=False)

def train\_seas(models, X\_train, y\_train, name, config):

temp = X\_train

for i in range(len(models) - 1):

if i > 0:

p = models[i - 1]

hidden\_layer\_model = Model(input=p.input,

output=p.get\_layer('hidden').output)

temp = hidden\_layer\_model.predict(temp)

m = models[i]

m.compile(loss="mse", optimizer="rmsprop", metrics=['mape'])

m.fit(temp, y\_train, batch\_size=config["batch"],

epochs=config["epochs"],

validation\_split=0.05)

models[i] = m

saes = models[-1]

for i in range(len(models) - 1):

weights = models[i].get\_layer('hidden').get\_weights()

saes.get\_layer('hidden%d' % (i + 1)).set\_weights(weights)

train\_model(saes, X\_train, y\_train, name, config)

def main(argv):

parser = argparse.ArgumentParser()

parser.add\_argument(

"--model",

default="lstm",

help="Model to train.")

args = parser.parse\_args()

lag = 12

config = {"batch": 256, "epochs": 60}

file1 = 'C:/Users/DELL/Downloads/traffic flow prediction/TrafficFlowPrediction-master/data/train.csv'

file2 = 'C:/Users/DELL/Downloads/traffic flow prediction/TrafficFlowPrediction-master/data/test.csv'

X\_train, y\_train, \_, \_, \_ = process\_data(file1, file2, lag)

if args.model == 'lstm':

X\_train = np.reshape(X\_train, (X\_train.shape[0], X\_train.shape[1], 1))

m = model.get\_lstm([12, 64, 64, 1])

train\_model(m, X\_train, y\_train, args.model, config)

if args.model == 'gru':

X\_train = np.reshape(X\_train, (X\_train.shape[0], X\_train.shape[1], 1))

m = model.get\_gru([12, 64, 64, 1])

train\_model(m, X\_train, y\_train, args.model, config)

if args.model == 'saes':

X\_train = np.reshape(X\_train, (X\_train.shape[0], X\_train.shape[1]))

m = model.get\_saes([12, 400, 400, 400, 1])

train\_seas(m, X\_train, y\_train, args.model, config)

if \_\_name\_\_ == '\_\_main\_\_':

main(sys.argv)