```
import matplotlib as mpl
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from scipy import stats
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
import sklearn
import sys
import io
def plot_learning_curves(loss, val_loss):
    plt.figure()
    plt.plot(np.arange(len(loss)), loss, "b.-", label="Training loss")
    plt.plot(np.arange(len(val_loss)), val_loss, "r.-", label="Validation loss")
    plt.gca().xaxis.set major locator(mpl.ticker.MaxNLocator(integer=True))
    plt.legend(fontsize=14)
    plt.xlabel("Epochs")
    plt.ylabel("Loss")
   plt.grid(True)
n_steps = 50
forecast = 10
#EXTRACT FLOW, Z SCORE, OUTLIERS
data_west = pd.read_csv('denoised_data_1015_east.csv')
data_west_o = np.array(data_west.FLOW)
#EXRTEND DATA
array_to_concatinate = data_west o
for iter in range (35):
    data_west_o = np.concatenate([data_west_o,array_to_concatinate])
#SCALE AND RESHAPE DATA
scaler = MinMaxScaler()
array = data_west_o.reshape(-1, 1)
array scaled = scaler.fit transform(array)
flow_reshaped = array_scaled[:(len(array_scaled) - (len(array_scaled) % (n steps+forecast)))
#TRAIN SET, VALIDATION SET, TEST SET
test = int(0.7 * flow reshaped.shape[0])
valid = int(0.9 * flow_reshaped.shape[0])
X train = flow reshaped[:test, :n steps]
X valid = flow reshaped[test:valid, :n steps]
X test = flow reshaped[valid:, :n steps]
print(X test.shape)
print(X_test[-1:].shape)
#prepare targets
Y = np.empty((flow reshaped.shape[0], n steps, forecast))
for step ahead in range(1, forecast + 1):
    Y[:, :, step_ahead - 1] = flow_reshaped[:, step_ahead:step_ahead + n_steps, 0]
y_train = Y[:test]
y_valid = Y[test:valid]
y_test = Y[valid:]
#MODEL
def last time step mse(Y true, Y pred):
     return keras.metrics.mean_squared_error(Y_true[:, -1], Y_pred[:, -1])
```

```
model = keras.models.Sequential()
model.add(keras.layers.InputLayer(input_shape=[None, 1]))
for rate in (1, 2, 4, 8) \times 2:
    model.add(keras.layers.Conv1D(filters=20, kernel_size=2, padding="causal",
                                   activation="relu", dilation_rate=rate))
model.add(keras.layers.Conv1D(filters=10, kernel size=1))
model.compile(loss="mse", optimizer="adam", metrics=[last time step mse])
early stopping cb = keras.callbacks.EarlyStopping(patience=15, restore best weights=True)
history = model.fit(X train, y train, epochs=1000,
                     validation data=(X valid, y valid), callbacks=[early stopping cb])
model.save("vawenet_1015_east.h5")
plot_learning_curves(history.history["loss"], history.history["val_loss"])
#50 MIN FORECAST
#50 MIN FORECAST
#flow
flow_unscaled = array[:(len(array) - (len(array) % (n_steps + forecast)))].reshape(-1, (n_st
y_test_unscaled = flow_unscaled[valid:, n_steps:, 0]
y_real_rescaled = y_test_unscaled[-1, :].reshape(-1, 1)
print(y real rescaled.shape)
flow_not_reshaped = array[:(len(array) - (len(array) % (n_steps+forecast)))]
#flow prediction
y_pred = model.predict(X_test[-1, :].reshape(-1, n_steps, 1)) #shape (1, 50, 10)
y_{pred} = y_{pred}[-1,-1,:].reshape(-1,1)
y pred rescaled = scaler.inverse transform(y pred).reshape(-1, 1) #shape (10, 1)
print(y pred rescaled.shape)
#time
time_not_reshaped = np.array(data_west['TIME'][:(len(data_west['TIME']) - (len(data_west['TI
time_reshaped = np.array(data_west['TIME'][:(len(data_west['TIME']) - (len(data_west['TIME']
    reshape(-1, (n_steps+forecast), 1)
valid time = int(0.9 * time reshaped.shape[0])
y time test = time reshaped[valid time:, n steps:, 0]
print(y time test[-1, :].shape)
def plot prediction(y real resacled, y pred rescaled, flow not reshaped, time not reshaped,
    plt.figure()
    plt.title("50 minutes prediction", fontsize=14)
    plt.plot(time not reshaped[-300:-forecast], flow not reshaped[-300:-forecast], 'b-')
    plt.plot(y_time_test[-1, :], y_real_resacled, 'ro-', label = 'Real values')
plt.plot(y_time_test[-1, :], y_pred_rescaled, 'gx-', label = 'Predicted values')
    plt.legend(loc="upper left")
    plt.xlabel("Time (in 5 minutes intervals)")
    plt.ylabel('Volume (veh/h)')
plot_prediction(y_real_rescaled, y_pred_rescaled, flow_not_reshaped, time_not_reshaped, y_ti
plt.show()
```

Г⇒

```
(1072, 50, 1)
(1, 50, 1)
Train on 7499 samples, validate on 2142 samples
Epoch 1/1000
Epoch 2/1000
Epoch 3/1000
Epoch 4/1000
Epoch 5/1000
704/7499 [=>.....] - ETA: 3s - loss: 0.0074 - last_time_s
______
KeyboardInterrupt
                            Traceback (most recent call last)
<ipython-input-4-3fcaab120557> in <module>()
   73 early stopping cb = keras.callbacks.EarlyStopping(patience=15, restore be
   74 history = model.fit(X_train, y_train, epochs=1000,
---> 75
                  validation_data=(X_valid, y_valid), callbacks=[early_
   76
   77 model.save("vawenet 1015 east.h5")
                    3 frames
/usr/local/lib/python3.6/dist-packages/tensorflow/python/client/session.py in __c
  1456
          ret = tf_session.TF_SessionRunCallable(self._session._session,
  1457
                                     self. handle, args,
-> 1458
                                     run metadata ptr)
  1459
          if run metadata:
  1460
            proto data = tf session.TF GetBuffer(run metadata ptr)
```

KeyboardInterrupt:

SEARCH STACK OVERFLOW