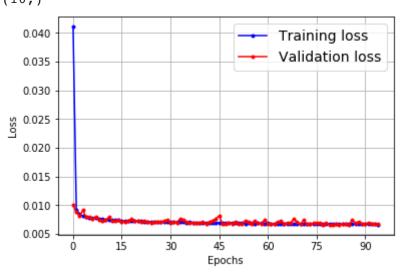
```
import matplotlib as mpl
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from scipy import stats
import seaborn as sns
import tensorflow as tf
from tensorflow import keras
import sklearn
import sys
import io
def plot learning curves(loss, 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 \text{ steps} = 50
forecast = 10
#EXTRACT FLOW, Z SCORE, OUTLIERS
data west = pd.read csv('denoised data 1015 north.csv')
data west o = np.array(data west.FLOW)
#EXRTEND DATA
array_to_concatinate = data_west_o[175:]
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) * 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 1014 north.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
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)
#t.ime
time not reshaped = np.array(data west['TIME'][:(len(data west['TIME']) - (len(data west[
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_reshape
    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
plt.show()
```

С→

```
Epoch 85/1000
Epoch 86/1000
Epoch 87/1000
Epoch 88/1000
Epoch 89/1000
Epoch 90/1000
Epoch 91/1000
Epoch 92/1000
Epoch 93/1000
Epoch 94/1000
Epoch 95/1000
7138/7138 [=====
     ========= ] - 5s 664us/sample - loss: 0.0066 - 1
(10, 1)
(10, 1)
(10,)
```





10/8/2019	wavenet_1014_north.ipynb - Colaboratory