Loading the data

Imports and Google Drive mounting

```
try:
  # %tensorflow_version only exists in Colab.
  %tensorflow_version 2.x
except Exception:
  pass

# TensorFlow and tf.keras
import tensorflow as tf
from tensorflow import keras

print(tf.__version__)

# Helper libraries
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import sklearn as sk
import pandas as pd
from pandas import read_csv
from datetime import datetime
import math
import os

# fix random seed for reproducibility
seed = 2020
np.random.seed(seed)
```

```
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_absolute_error

from tensorflow.keras.datasets import cifar100
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten, BatchNormalization, Activation
from tensorflow.keras.layers import Conv1D, MaxPooling1D, GlobalMaxPooling1D
from tensorflow.keras.layers import LSTM, GRU
from tensorflow.keras.regularizers import L1L2
from tensorflow.keras.constraints import max_norm
from tensorflow.keras.optimizers import Adam, SGD
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.models import load_model
from tensorflow.keras import optimizers
from tensorflow.keras import Input
from tensorflow.keras.models import Model
from tensorflow.keras.constraints import MaxNorm
```

```
TensorFlow 2.x selected.
2.1.0
```

```
from google.colab import drive
drive.mount('/content/gdrive')

!ls '/content/gdrive/My Drive/Colab Notebooks/DL2020/GA3/'
```

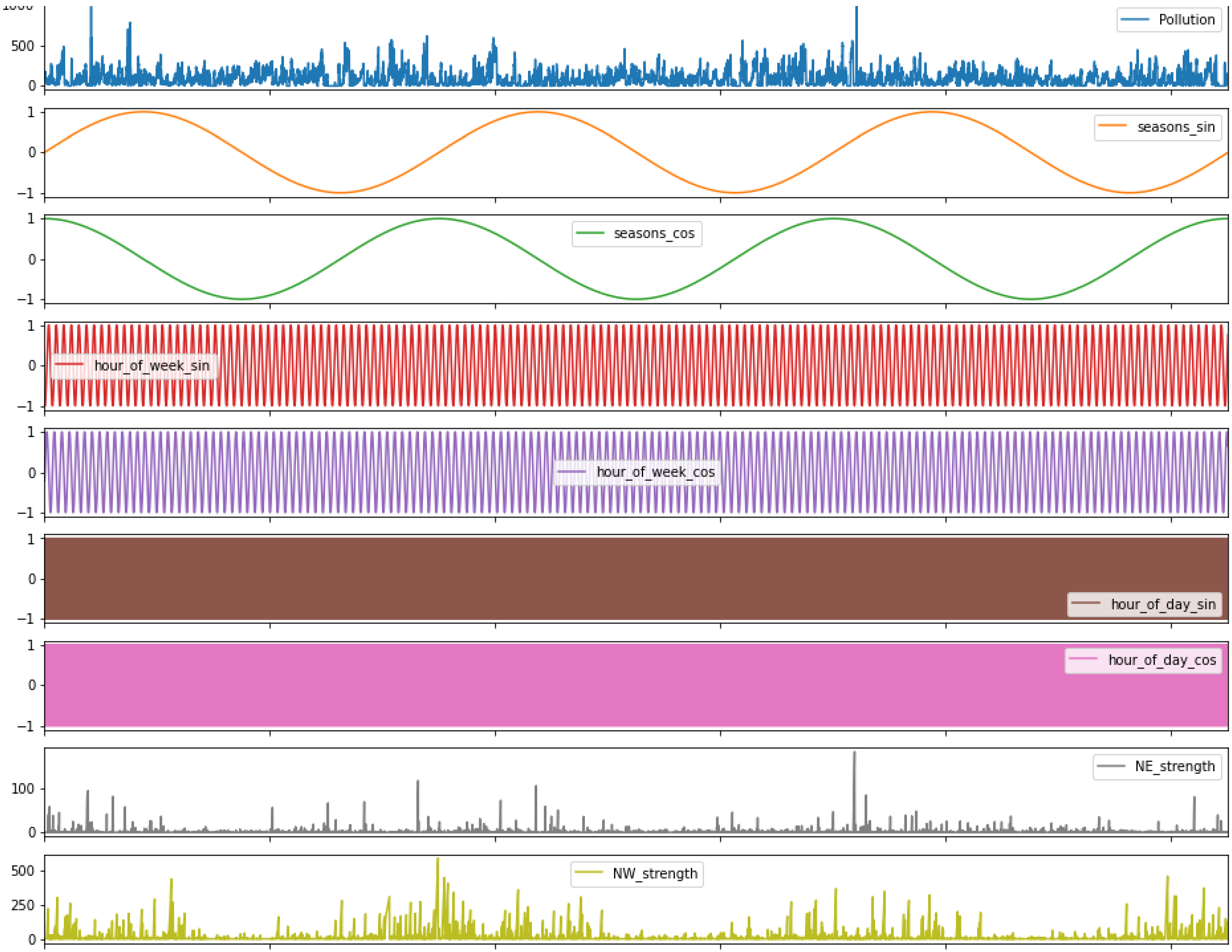
```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True
checkpoints                       preprocessed_train_data_all.csv
pollution_data.csv                preprocessed_train_data.csv
preprocessed_pollution_data.csv   preprocessed_val_data.csv
preprocessed_test_data.csv
```

Loading the dataset

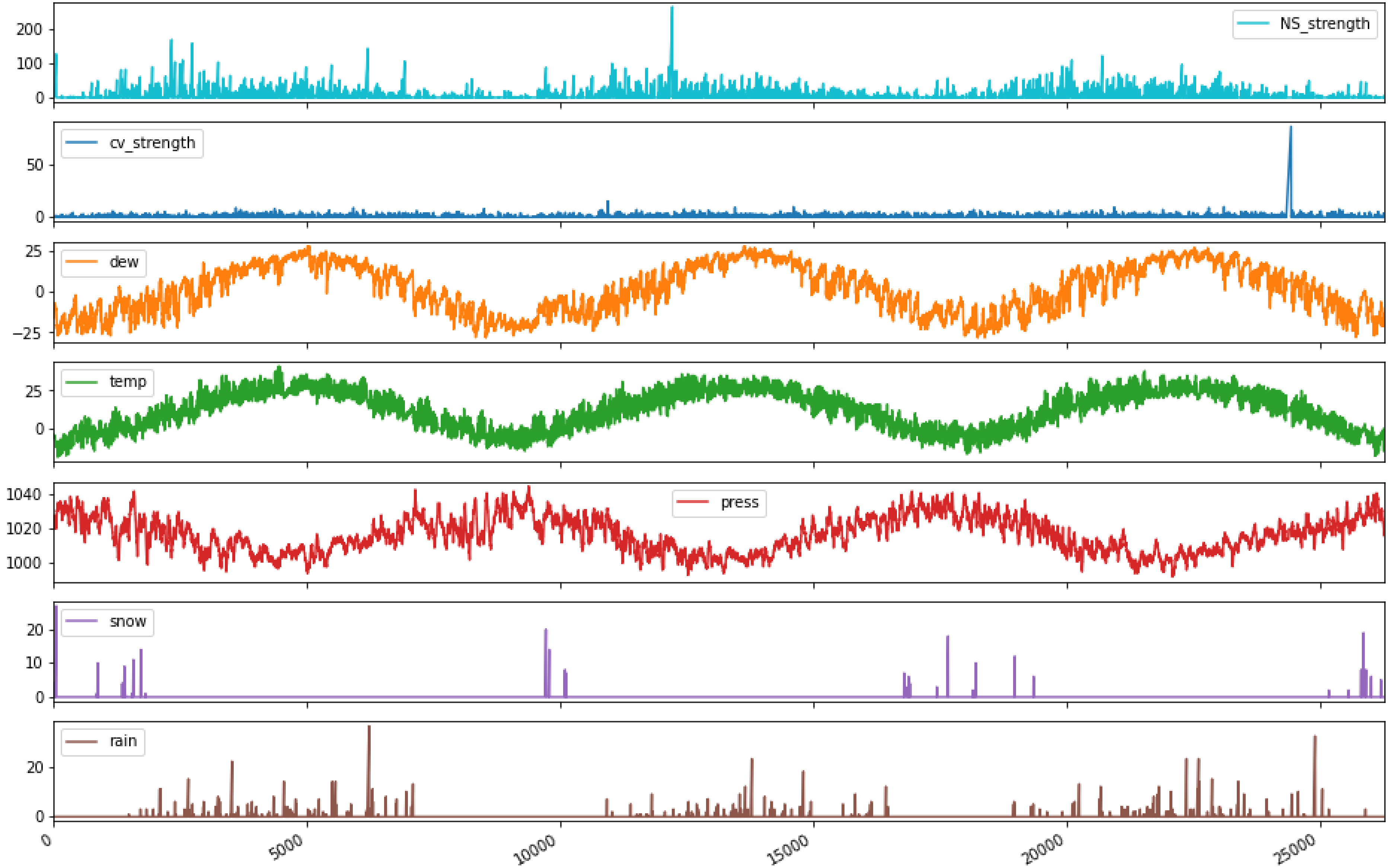
```
# this code only loads train and test datasets
```

```
# remember you also need to split off a validation dataset
# either by reusing and adapting the splitting code from the preprocessing notebook
# or by writing your own code
# like in the previous assignments, this means you should have 4 sets:

# train_all (train+validate), train, validate and test

# The PATH setting below assumes you just uploaded the data file to your Colab session
# When using Drive: replace this by the path where you put the data file
DATAPATH = '/content/gdrive/My Drive/Colab Notebooks/DL2020/GA3/'
TRAINDATAALLFILE = DATAPATH+'preprocessed_train_data_all.csv'
TRAINDATAFILE = DATAPATH+'preprocessed_train_data.csv'
VALDATAFILE = DATAPATH+'preprocessed_val_data.csv'
TESTDATAFILE = DATAPATH+'preprocessed_test_data.csv'

train_all_dataset = read_csv(TRAINDATAALLFILE, header=0, index_col=0)
train_dataset = read_csv(TRAINDATAFILE, header=0, index_col=0)
val_dataset = read_csv(VALDATAFILE, header=0, index_col=0)
test_dataset = read_csv(TESTDATAFILE, header=0, index_col=0)

#check - note that you can make many other plots directly from pandas dataframes:
# https://pandas.pydata.org/pandas-docs/version/0.16/visualization.html
train_dataset.loc[:, train_dataset.columns != 'Year'].plot(subplots=True,figsize=(16,26))
```

```
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/tools.py:307: MatplotlibDeprecationWarning:
The rowNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get_subplotspec().row
  layout[ax.rowNum, ax.colNum] = ax.get_visible()
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/tools.py:307: MatplotlibDeprecationWarning:
The colNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get_subplotspec().col
  layout[ax.rowNum, ax.colNum] = ax.get_visible()
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/tools.py:313: MatplotlibDeprecationWarning:
The rowNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get_subplotspec().row
  if not layout[ax.rowNum + 1, ax.colNum]:
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/tools.py:313: MatplotlibDeprecationWarning:
The colNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get_subplotspec().col
  if not layout[ax.rowNum + 1, ax.colNum]:
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/tools.py:307: MatplotlibDeprecationWarning:
The rowNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get_subplotspec().row
  layout[ax.rowNum, ax.colNum] = ax.get_visible()
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/tools.py:307: MatplotlibDeprecationWarning:
The colNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get_subplotspec().col
  layout[ax.rowNum, ax.colNum] = ax.get_visible()
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/tools.py:313: MatplotlibDeprecationWarning:
The rowNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get_subplotspec().row
  if not layout[ax.rowNum + 1, ax.colNum]:
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/tools.py:313: MatplotlibDeprecationWarning:
The colNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get_subplotspec().col
  if not layout[ax.rowNum + 1, ax.colNum]:
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f3290398438>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f329036d6d8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f32903246d8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f32902da6d8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f329028f6d8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f32902c36d8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f32902796d8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f329022e6a0>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f329022e710>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f32901986d8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f329014d6d8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f32901816d8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f32901366d8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f32900eb6d8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f32900a06d8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f32900556d8>],
      dtype=object)
```



Pollution
seasons_sin
seasons_cos
hour_of_week_sin
hour_of_week_cos
hour_of_day_sin
hour_of_day_cos
NE_strength
NW_strength
500
0
1
−1
100
250



Extract train, validate and test features and labels

```
# extract data from dataframes, selecting the features you want
```

```
# note that the labels simply contain the pollution data for now
# depending on the window sizes used in training, the correct values will be cut out
features = np.arange(1,17)
pollution = 1

# Important: Tensorflow 2.x gives an error if you omit
# the np.asarray(...,,dtype=np.float32)
train_all_values = np.asarray(train_all_dataset.values[:,features],dtype=np.float32)
train_all_labels = np.asarray(train_all_dataset.values[:,pollution],dtype=np.float32)
train_values = np.asarray(train_dataset.values[:,features],dtype=np.float32)
train_labels = np.asarray(train_dataset.values[:,pollution],dtype=np.float32)
val_values = np.asarray(val_dataset.values[:,features],dtype=np.float32)
val_labels = np.asarray(val_dataset.values[:,pollution],dtype=np.float32)
test_values = np.asarray(test_dataset.values[:,features],dtype=np.float32)
test_labels = np.asarray(test_dataset.values[:,pollution],dtype=np.float32)
```

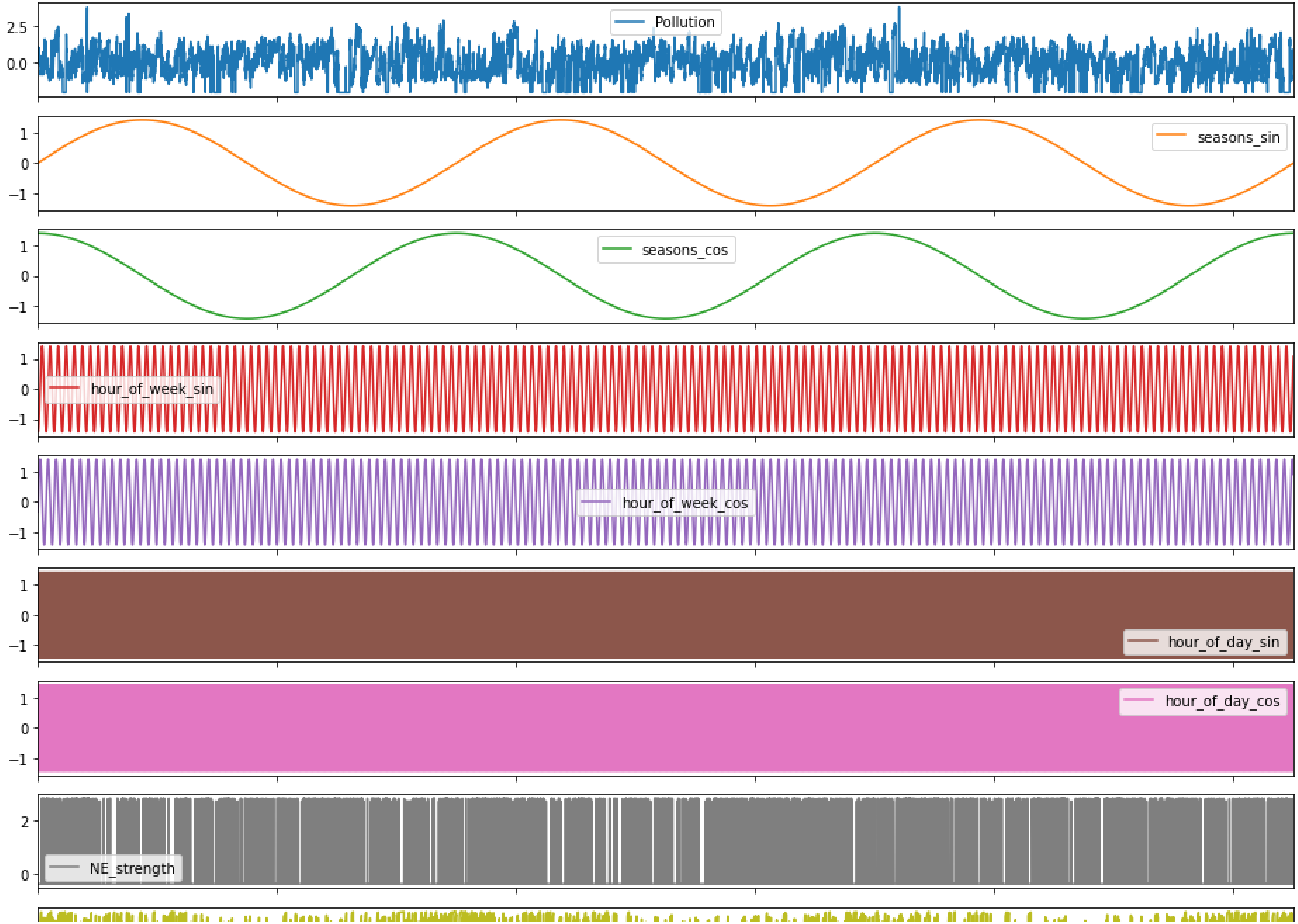
Normalising the data

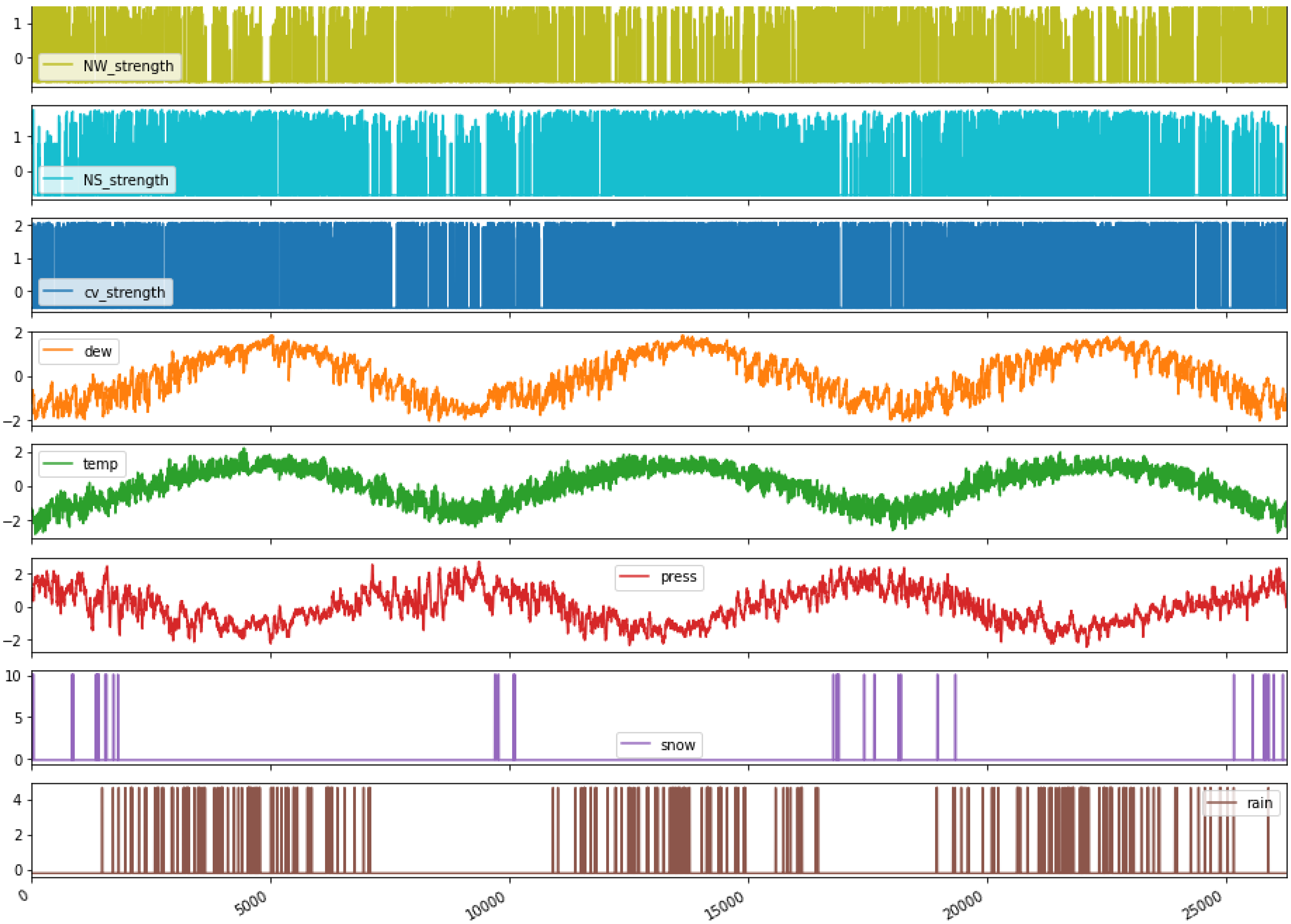
```
# imports to show that there are many different scalers
# especially with recurrent NNs, the choice of scaler can make a difference
# look up what they do before choosing which one to try
# from sklearn.preprocessing import StandardScaler
# from sklearn.preprocessing import MinMaxScaler
# from sklearn.preprocessing import MaxAbsScaler
# from sklearn.preprocessing import RobustScaler
# from sklearn.preprocessing import Normalizer # BAD for this case
# from sklearn.preprocessing import QuantileTransformer
from sklearn.preprocessing import PowerTransformer # BEST for this case

# Example: train standard scalers, apply to train and test data
# adapt to do all you need to do ...
SS1 = PowerTransformer()
SS1.fit(train_values)

train_scaled = SS1.transform(train_values)
val_scaled = SS1.transform(val_values)
```

```
val_scaled = SS1.transform(val_values)

SS2 = PowerTransformer()
SS2.fit(train_all_values)

train_all_scaled = SS2.transform(train_all_values)
test_scaled = SS2.transform(test_values)

pd.DataFrame(train_scaled, columns=train_dataset.columns[1:]).plot(subplots=True,figsize=(16,26))
```

```
/usr/local/lib/python3.6/dist-packages/numpy/core/_methods.py:195: RuntimeWarning: overflow encountered in multiply
  x = um.multiply(x, x, out=x)
/usr/local/lib/python3.6/dist-packages/numpy/core/_methods.py:195: RuntimeWarning: overflow encountered in multiply
  x = um.multiply(x, x, out=x)
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/tools.py:307: MatplotlibDeprecationWarning:
The rowNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get_subplotspec().row
  layout[ax.rowNum, ax.colNum] = ax.get_visible()
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/tools.py:307: MatplotlibDeprecationWarning:
The colNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get_subplotspec().col
  layout[ax.rowNum, ax.colNum] = ax.get_visible()
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/tools.py:313: MatplotlibDeprecationWarning:
The rowNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get_subplotspec().row
  if not layout[ax.rowNum + 1, ax.colNum]:
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/tools.py:313: MatplotlibDeprecationWarning:
The colNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get_subplotspec().col
  if not layout[ax.rowNum + 1, ax.colNum]:
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/tools.py:307: MatplotlibDeprecationWarning:
The rowNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get_subplotspec().row
  layout[ax.rowNum, ax.colNum] = ax.get_visible()
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/tools.py:307: MatplotlibDeprecationWarning:
The colNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get_subplotspec().col
  layout[ax.rowNum, ax.colNum] = ax.get_visible()
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/tools.py:313: MatplotlibDeprecationWarning:
The rowNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get_subplotspec().row
  if not layout[ax.rowNum + 1, ax.colNum]:
/usr/local/lib/python3.6/dist-packages/pandas/plotting/_matplotlib/tools.py:313: MatplotlibDeprecationWarning:
The colNum attribute was deprecated in Matplotlib 3.2 and will be removed two minor releases later. Use ax.get_subplotspec().col
  if not layout[ax.rowNum + 1, ax.colNum]:
array([<matplotlib.axes._subplots.AxesSubplot object at 0x7f328f961198>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f328f980f98>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f328e124f98>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f328e0d8eb8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f328e098128>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f328e049358>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f32903249e8>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f32902eb080>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f32902eb198>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f328f9d5400>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f328e138978>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f328df5e400>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f328df13400>,
       <matplotlib.axes._subplots.AxesSubplot object at 0x7f328dec9400>,
```

```
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f328de7f400>,
 <matplotlib.axes._subplots.AxesSubplot object at 0x7f328deb3400>],
dtype=object)
```





NW_strength
NS_strength
cv_strength
dew
temp
press
snow
rain
0
5000
10000
15000
20000
25000

Dataset manipulations

```
# dense network:
# here you only need to take into account that we are predicting 6 steps ahead
# this means that the features of the first timestep (index 0)
# are used to predict the 7th pollution value (index 6) so the first 6 and the last 6 feature-samples labels are omitted
def create_dataset_dense(train, test, ahead=6): # can use this with different 'ahead' values, but default is set to 6
    return train[:-ahead,:], test[ahead:]

# window-based and recurrent networks:
# now, you use a window of k history values to predict
# this means that the features of the first k timesteps (indices 0 to k-1)
# are used to predict the k+6th pollution value (index k+6-1)
# output dimension of train data is samples x window x features
def create_dataset_windowed(train, test, ahead=6, window_size=1):
  samples = train.shape[0]-ahead-(window_size-1)
  dataX= []
  for i in range(samples):
      a = train[i:(i+window_size), :]
      dataX.append(a)

  return np.array(dataX), test[ahead+window_size-1:]

# reshape arrays so that they can be used as input for the top model of the ensemble
def create_dataset_top(train, test, ahead=6, window_size=1):
  return np.concatenate(train, axis=1), test[ahead+window_size-1:]
```

Help functions for visualization

```
def plot_history(history):
  plt.figure(figsize=(6,4))

  plt.xlabel('Epoch')
  plt.ylabel('Mae')
```

```
  plt.plot(history.epoch, np.array(history.history['mae']),'g-',
           label='Train MAE')
  try:
    plt.plot(history.epoch, np.array(history.history['val_mae']),'r-',
             label = 'Validation MAE')
  except:
    pass
  plt.legend()
  plt.show()

def PlotResults(labels,predictions,binsize=10):
  num_samples = len(labels)

  fig = plt.figure(figsize=(16,16))
  spec = gridspec.GridSpec(ncols=4, nrows=4, figure=fig)
  ax1 = fig.add_subplot(spec[0, :])
  ax2 = fig.add_subplot(spec[1, :])
  ax3 = fig.add_subplot(spec[2:,0:2])
  ax4 = fig.add_subplot(spec[2:,2:])

  ax1.plot(labels,'k-',label='Labels')
  ax1.plot(predictions,'r-',label='Predictions')
  ax1.set_ylabel('Pollution')
  ax1.legend()

  errors=np.absolute(labels-predictions)
  ax2.plot(errors,'k-')
  ax2.set_ylabel("Absolute error")

  ax3.scatter(labels,predictions)
  ax3.set_xlabel('Labels')
  ax3.set_ylabel('Predictions')

  bins = np.arange(0,(np.ceil(np.max(errors)/binsize)+1)*binsize,binsize)

  ax4.hist(errors,bins=bins)
  ax4.set_xlabel('Absolute error')
  ax4.set_ylabel('Frequency')
```

```
  ax4.set_ylabel('Frequency')

  plt.show()
```

Step 1

```
X_train_d,r_train_d = create_dataset_dense(train_scaled,train_labels)
X_val_d,r_val_d = create_dataset_dense(val_scaled,val_labels)
X_train_all_d,r_train_all_d = create_dataset_dense(train_all_scaled,train_all_labels)
X_test_d,r_test_d = create_dataset_dense(test_scaled,test_labels)

# Train a Dense network on the scaled features
def dense_model(learning_rate=0.01, hidden=[256,1024,1024,128], dropouts=[0.5,0.5,0.5,0.1]):
  # create linear model
  model = Sequential()
  model.add(Dense(hidden[0], kernel_initializer='he_uniform', input_shape=(train_scaled.shape[1],)))
  model.add(BatchNormalization())
  model.add(Activation('relu'))
  model.add(Dropout(dropouts[0]))

  for idx in range(1, len(hidden)):
    model.add(Dense(hidden[idx], kernel_initializer='he_uniform'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(Dropout(dropouts[idx]))

  model.add(Dense(1))
  model.add(Activation('linear'))
  # no dropout at the end

  optim = tf.keras.optimizers.Adam(lr=learning_rate)
  model.compile(loss='mae',
              optimizer=optim,
              # keep extra metrics: mse and mae without regularisation terms
              metrics=['mse', 'mae'])
  return model
```

```
  return model
```

```
model = dense_model()
model.summary()

batch_size = 5*168
epochs = 70

# cp_path = '/content/gdrive/My Drive/Colab Notebooks/DL2020/GA3/checkpoints/step_1/cp.ckpt'
# cp_callback = tf.keras.callbacks.ModelCheckpoint(cp_path,
#                                                   monitor='val_mae',
#                                                   verbose=1,
#                                                   save_best_only=True,
#                                                   save_weights_only=True)
# lr_callback = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_mae',
#                                                     factor=0.9,
#                                                     patience=5,
#                                                     verbose=1,
#                                                     min_lr=0.0001)

# history = model.fit(X_train_d,
#                       r_train_d,
#                       shuffle=True,
#                       batch_size=batch_size,
#                       epochs=epochs,
#                       validation_data=(X_val_d,r_val_d),
#                       callbacks=[cp_callback, lr_callback])
# model.load_weights(cp_path)

history = model.fit(X_train_all_d,
                    r_train_all_d,
                    shuffle=True,
                    batch_size=batch_size,
                    epochs=epochs)
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
dense (Dense)                (None, 256)               4352      
_________________________________________________________________
batch_normalization (BatchNo (None, 256)               1024      
_________________________________________________________________
activation (Activation)      (None, 256)               0         
_________________________________________________________________
dropout (Dropout)            (None, 256)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 1024)              263168    
_________________________________________________________________
batch_normalization_1 (Batch (None, 1024)              4096      
_________________________________________________________________
activation_1 (Activation)    (None, 1024)              0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0         
_________________________________________________________________
dense_2 (Dense)              (None, 1024)              1049600   
_________________________________________________________________
batch_normalization_2 (Batch (None, 1024)              4096      
_________________________________________________________________
activation_2 (Activation)    (None, 1024)              0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 1024)              0         
_________________________________________________________________
dense_3 (Dense)              (None, 128)               131200    
_________________________________________________________________
batch_normalization_3 (Batch (None, 128)               512       
_________________________________________________________________
activation_3 (Activation)    (None, 128)               0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 128)               0         
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 129       
_________________________________________________________________
activation_4 (Activation)    (None, 1)                 0         
=================================================================
Total params: 1,458,177
Trainable params: 1,453,313
```

```
Trainable params: [illegible]
Non-trainable params: 4,864
_________________________________________________________________
Train on 35034 samples
Epoch 1/70
35034/35034 [==============================] - 2s 46us/sample - loss: 78.5146 - mse: 13661.8945 - mae: 78.5146
Epoch 2/70
35034/35034 [==============================] - 0s 9us/sample - loss: 44.0658 - mse: 5203.6519 - mae: 44.0658
Epoch 3/70
35034/35034 [==============================] - 0s 8us/sample - loss: 37.3092 - mse: 3571.8965 - mae: 37.3092
Epoch 4/70
35034/35034 [==============================] - 0s 8us/sample - loss: 36.7322 - mse: 3503.4966 - mae: 36.7322
Epoch 5/70
35034/35034 [==============================] - 0s 8us/sample - loss: 36.3990 - mse: 3457.7317 - mae: 36.3990
Epoch 6/70
35034/35034 [==============================] - 0s 8us/sample - loss: 36.0665 - mse: 3418.4182 - mae: 36.0665
Epoch 7/70
35034/35034 [==============================] - 0s 8us/sample - loss: 35.9245 - mse: 3398.2158 - mae: 35.9245
Epoch 8/70
35034/35034 [==============================] - 0s 8us/sample - loss: 35.8215 - mse: 3389.9365 - mae: 35.8215
Epoch 9/70
35034/35034 [==============================] - 0s 8us/sample - loss: 35.6267 - mse: 3354.0217 - mae: 35.6267
Epoch 10/70
35034/35034 [==============================] - 0s 8us/sample - loss: 35.4905 - mse: 3326.1458 - mae: 35.4905
Epoch 11/70
35034/35034 [==============================] - 0s 8us/sample - loss: 35.2678 - mse: 3320.1406 - mae: 35.2678
Epoch 12/70
35034/35034 [==============================] - 0s 8us/sample - loss: 35.1846 - mse: 3303.9519 - mae: 35.1846
Epoch 13/70
35034/35034 [==============================] - 0s 8us/sample - loss: 35.2897 - mse: 3323.4863 - mae: 35.2897
Epoch 14/70
35034/35034 [==============================] - 0s 8us/sample - loss: 35.1027 - mse: 3303.5291 - mae: 35.1027
Epoch 15/70
35034/35034 [==============================] - 0s 9us/sample - loss: 34.8917 - mse: 3261.8740 - mae: 34.8917
Epoch 16/70
35034/35034 [==============================] - 0s 8us/sample - loss: 34.8479 - mse: 3262.4751 - mae: 34.8479
Epoch 17/70
35034/35034 [==============================] - 0s 8us/sample - loss: 34.6774 - mse: 3225.4436 - mae: 34.6774
Epoch 18/70
35034/35034 [==============================] - 0s 8us/sample - loss: 34.6194 - mse: 3225.2275 - mae: 34.6194
Epoch 19/70
35034/35034 [==============================] - 0s 8us/sample - loss: 34.7590 - mse: 3252.2400 - mae: 34.7590
Epoch 20/70
```

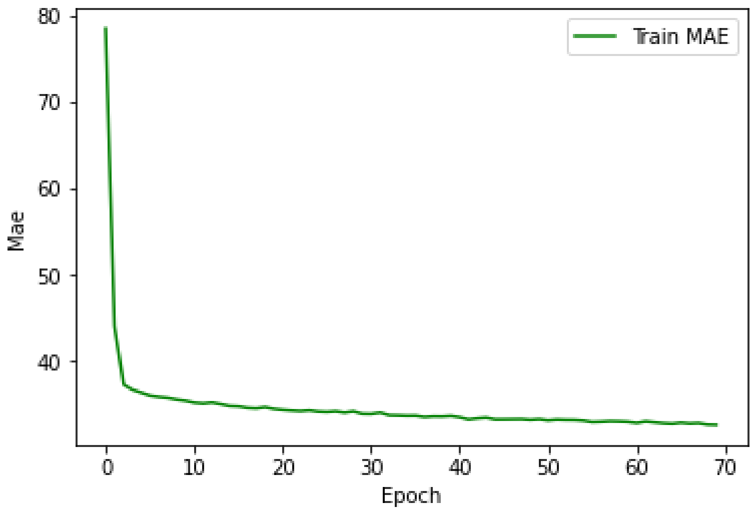
```
35034/35034 [==============================] - 0s 8us/sample - loss: 34.5515 - mse: 3195.1504 - mae: 34.5515
Epoch 21/70
35034/35034 [==============================] - 0s 9us/sample - loss: 34.4425 - mse: 3205.3689 - mae: 34.4425
Epoch 22/70
35034/35034 [==============================] - 0s 9us/sample - loss: 34.3655 - mse: 3200.5151 - mae: 34.3655
Epoch 23/70
35034/35034 [==============================] - 0s 8us/sample - loss: 34.3033 - mse: 3168.3855 - mae: 34.3033
Epoch 24/70
35034/35034 [==============================] - 0s 9us/sample - loss: 34.3860 - mse: 3185.9153 - mae: 34.3860
Epoch 25/70
35034/35034 [==============================] - 0s 9us/sample - loss: 34.2461 - mse: 3172.3406 - mae: 34.2461
Epoch 26/70
35034/35034 [==============================] - 0s 8us/sample - loss: 34.1995 - mse: 3165.9343 - mae: 34.1995
Epoch 27/70
35034/35034 [==============================] - 0s 9us/sample - loss: 34.2764 - mse: 3162.7942 - mae: 34.2764
Epoch 28/70
35034/35034 [==============================] - 0s 8us/sample - loss: 34.1207 - mse: 3155.5232 - mae: 34.1207
Epoch 29/70
35034/35034 [==============================] - 0s 8us/sample - loss: 34.2727 - mse: 3143.4817 - mae: 34.2727
Epoch 30/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.9865 - mse: 3148.1396 - mae: 33.9865
Epoch 31/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.9636 - mse: 3133.0679 - mae: 33.9636
Epoch 32/70
35034/35034 [==============================] - 0s 8us/sample - loss: 34.1207 - mse: 3154.8799 - mae: 34.1207
Epoch 33/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.8180 - mse: 3113.9597 - mae: 33.8180
Epoch 34/70
35034/35034 [==============================] - 0s 9us/sample - loss: 33.7960 - mse: 3107.2698 - mae: 33.7960
Epoch 35/70
35034/35034 [==============================] - 0s 9us/sample - loss: 33.7567 - mse: 3114.9546 - mae: 33.7567
Epoch 36/70
35034/35034 [==============================] - 0s 9us/sample - loss: 33.7672 - mse: 3091.2883 - mae: 33.7672
Epoch 37/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.6124 - mse: 3075.5461 - mae: 33.6124
Epoch 38/70
35034/35034 [==============================] - 0s 9us/sample - loss: 33.6878 - mse: 3084.3408 - mae: 33.6878
Epoch 39/70
35034/35034 [==============================] - 0s 9us/sample - loss: 33.6742 - mse: 3105.8569 - mae: 33.6742
Epoch 40/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.7394 - mse: 3085.9954 - mae: 33.7394
Epoch 41/70
```

```
35034/35034 [==============================] - 0s 8us/sample - loss: 33.5806 - mse: 3050.9263 - mae: 33.5806
Epoch 42/70
35034/35034 [==============================] - 0s 9us/sample - loss: 33.3304 - mse: 3024.0122 - mae: 33.3304
Epoch 43/70
35034/35034 [==============================] - 0s 9us/sample - loss: 33.4608 - mse: 3065.6443 - mae: 33.4608
Epoch 44/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.5393 - mse: 3051.9995 - mae: 33.5393
Epoch 45/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.3328 - mse: 3041.8708 - mae: 33.3328
Epoch 46/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.3387 - mse: 3034.3071 - mae: 33.3387
Epoch 47/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.3511 - mse: 3040.2632 - mae: 33.3511
Epoch 48/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.3621 - mse: 3051.0061 - mae: 33.3621
Epoch 49/70
35034/35034 [==============================] - 0s 9us/sample - loss: 33.2823 - mse: 3026.6216 - mae: 33.2823
Epoch 50/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.3578 - mse: 3058.5618 - mae: 33.3578
Epoch 51/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.2197 - mse: 3014.4700 - mae: 33.2197
Epoch 52/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.3109 - mse: 3045.2559 - mae: 33.3109
Epoch 53/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.2751 - mse: 3025.4312 - mae: 33.2751
Epoch 54/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.2664 - mse: 3006.1978 - mae: 33.2664
Epoch 55/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.1946 - mse: 3017.5742 - mae: 33.1946
Epoch 56/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.0301 - mse: 2993.9436 - mae: 33.0301
Epoch 57/70
35034/35034 [==============================] - 0s 9us/sample - loss: 33.0644 - mse: 2972.7712 - mae: 33.0644
Epoch 58/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.1257 - mse: 2992.3965 - mae: 33.1257
Epoch 59/70
35034/35034 [==============================] - 0s 8us/sample - loss: 33.1044 - mse: 3015.5137 - mae: 33.1044
Epoch 60/70
35034/35034 [==============================] - 0s 9us/sample - loss: 33.0605 - mse: 2999.5271 - mae: 33.0605
Epoch 61/70
35034/35034 [==============================] - 0s 8us/sample - loss: 32.9215 - mse: 2969.3818 - mae: 32.9215
Epoch 62/70
```

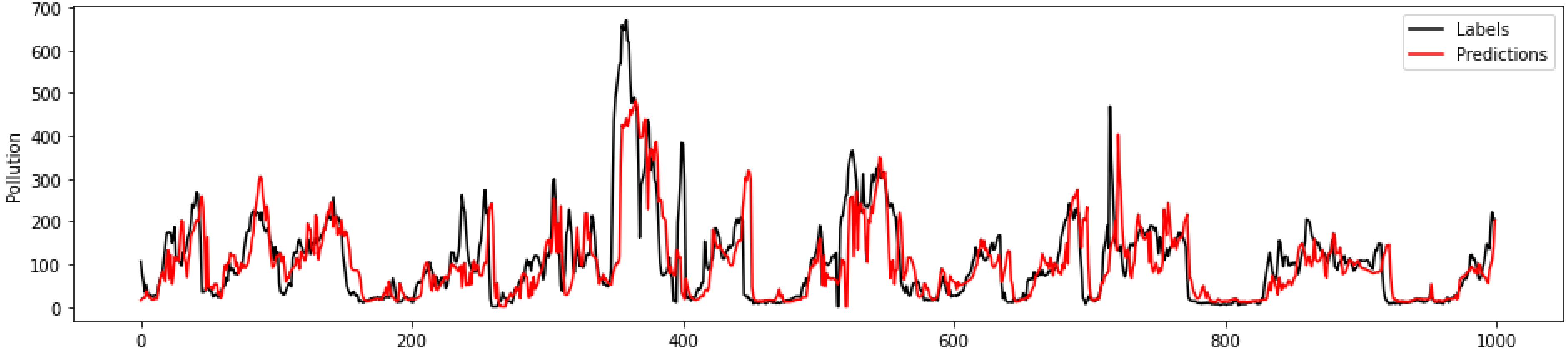
```
35034/35034 [==============================] - 0s 8us/sample - loss: 33.1054 - mse: 2992.5251 - mae: 33.1054
Epoch 63/70
35034/35034 [==============================] - 0s 8us/sample - loss: 32.9936 - mse: 2974.5037 - mae: 32.9936
Epoch 64/70
35034/35034 [==============================] - 0s 9us/sample - loss: 32.8913 - mse: 2953.0444 - mae: 32.8913
Epoch 65/70
35034/35034 [==============================] - 0s 8us/sample - loss: 32.8328 - mse: 2951.7742 - mae: 32.8328
Epoch 66/70
35034/35034 [==============================] - 0s 8us/sample - loss: 32.9495 - mse: 2997.6707 - mae: 32.9495
Epoch 67/70
35034/35034 [==============================] - 0s 9us/sample - loss: 32.8606 - mse: 2957.0000 - mae: 32.8606
Epoch 68/70
35034/35034 [==============================] - 0s 9us/sample - loss: 32.9068 - mse: 2975.1118 - mae: 32.9068
Epoch 69/70
35034/35034 [==============================] - 0s 8us/sample - loss: 32.7313 - mse: 2946.4006 - mae: 32.7313
Epoch 70/70
35034/35034 [==============================] - 0s 8us/sample - loss: 32.6970 - mse: 2945.5493 - mae: 32.6970
```

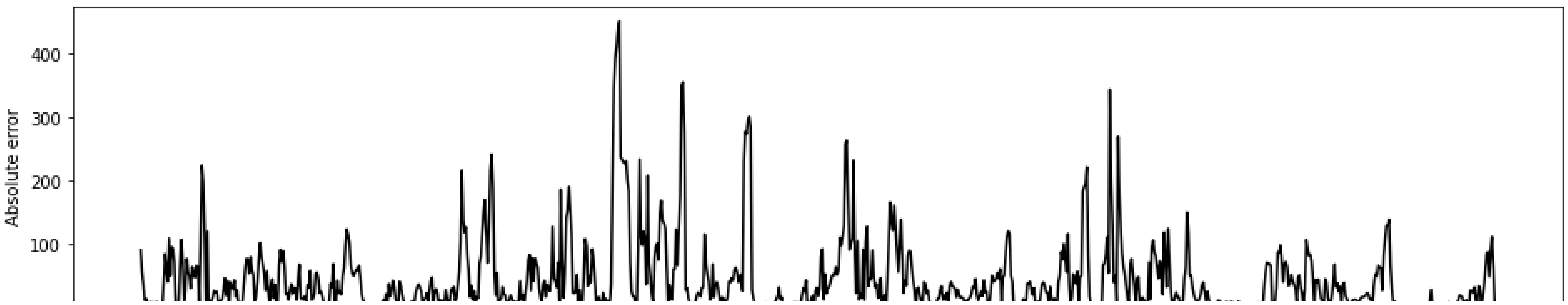
```
plot_history(history)

# y_train = model.predict(X_train_d)
# y_val = model.predict(X_val_d)
y_train_all = model.predict(X_train_all_d)
y_test = model.predict(X_test_d)

# mae_train = mean_absolute_error(r_train_d,y_train)
# mae_val = mean_absolute_error(r_val_d,y_val)
mae_train_all = mean_absolute_error(r_train_all_d,y_train_all)
mae_test = mean_absolute_error(r_test_d,y_test)

# print(f"train      mae: {mae_train_all}")
# print(f"test mae: {mae_test}")
print(f"train      mae: {mae_train_all}")
print(f"validation mae: {mae_test}")

# Visualise first 1000 predictions for validation and test
# PlotResults(r_val_d[:1000],y_val[:1000,0])
```

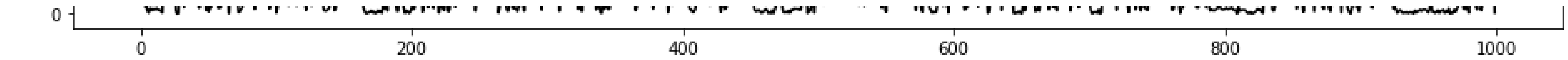
```
PlotResults(r_test_d[:1000],y_test[:1000,0])
```

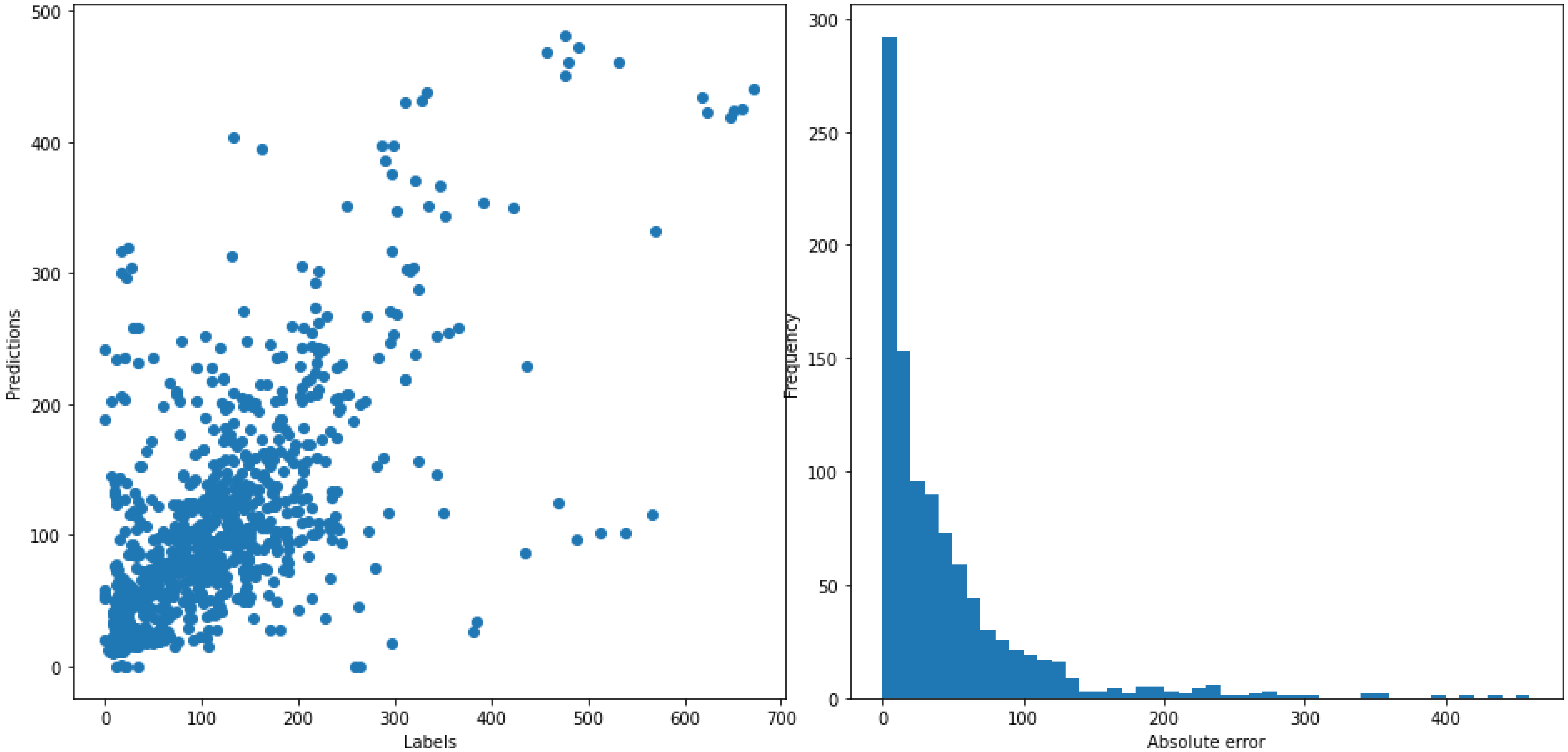



```
train      mae: 30.360923767089844
validation mae: 34.866722106933594
```









LIME

```
# predictions = model.predict(X_val_d)
# r_val_d_T = np.transpose(np.array([r_val_d]))
# mae_val = abs(r_val_d_T-predictions)
# mae_val_partitioned = np.argpartition(mae_val,-5,axis=0)
# largest = mae_val_partitioned[-5:]
# smallest = mae_val_partitioned[:5]
```

```
# print(f'Largest mistakes: {mae_val[largest]}')
# print(f'Smallest mistakes: {mae_val[smallest]}')

predictions = model.predict(X_test_d)
r_test_d_T = np.transpose(np.array([r_test_d]))
mae_test = abs(r_test_d_T-predictions)
mae_test_partitioned = np.argpartition(mae_test,-5,axis=0)
largest = mae_test_partitioned[-5:]
smallest = mae_test_partitioned[:5]
print(f'Largest mistakes: {mae_test[largest]}')
print(f'Smallest mistakes: {mae_test[smallest]}')
```

```
Largest mistakes: [[[419.20596]]

 [[451.2628 ]]

 [[516.565  ]]

 [[437.78207]]

 [[460.53723]]]
Smallest mistakes: [[[ 4.7851257]]

 [[ 3.4017372]]

 [[ 8.788279 ]]

 [[12.0851345]]

 [[12.711214 ]]]
```

```
!pip install lime

import lime
import lime.lime_tabular

feature_names = ['Pollution',
                 'seasons_sin',
                 'seasons_cos',
```

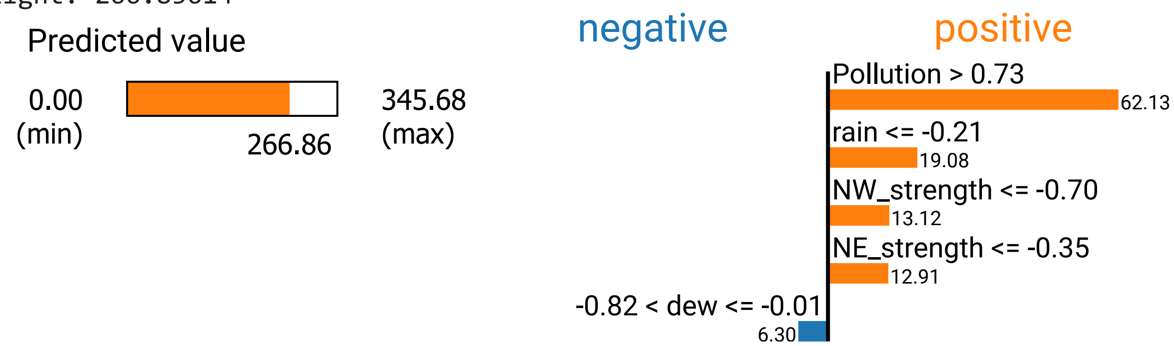
```
              'hour_of_week_sin',
              'hour_of_week_cos',
              'hour_of_day_sin',
              'hour_of_day_cos',
              'NE_strength',
              'NW_strength',
              'SE_strength',
              'cv_strength',
              'dew',
              'temp',
              'press',
              'snow',
              'rain']

explainer = lime.lime_tabular.LimeTabularExplainer(train_scaled, feature_names=feature_names, class_names=['Pollution'], verbose=True
```

```
Requirement already satisfied: lime in /usr/local/lib/python3.6/dist-packages (0.1.1.37)
Requirement already satisfied: scikit-image>=0.12; python_version >= "3.0" in /usr/local/lib/python3.6/dist-packages (from lime)
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from lime) (1.4.1)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from lime) (1.18.2)
Requirement already satisfied: matplotlib; python_version >= "3.0" in /usr/local/lib/python3.6/dist-packages (from lime) (3.2.0)
Requirement already satisfied: scikit-learn>=0.18 in /usr/local/lib/python3.6/dist-packages (from lime) (0.22.2.post1)
Requirement already satisfied: progressbar in /usr/local/lib/python3.6/dist-packages (from lime) (2.5)
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image>=0.12; python_version
Requirement already satisfied: pillow>=4.3.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image>=0.12; python_version
Requirement already satisfied: PyWavelets>=0.4.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image>=0.12; python_vers
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.6/dist-packages (from scikit-image>=0.12; python_version
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from matplotlib; python_version >= "3.0"-
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotl
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib; python_version >= "
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages (from matplotlib; python_version >
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-learn>=0.18->lime) (0.14.1)
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.6/dist-packages (from networkx>=2.0->scikit-image>=0.1
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from cycler>=0.10->matplotlib; python_version >= "
Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages (from kiwisolver>=1.0.1->matplotlib; python_
```

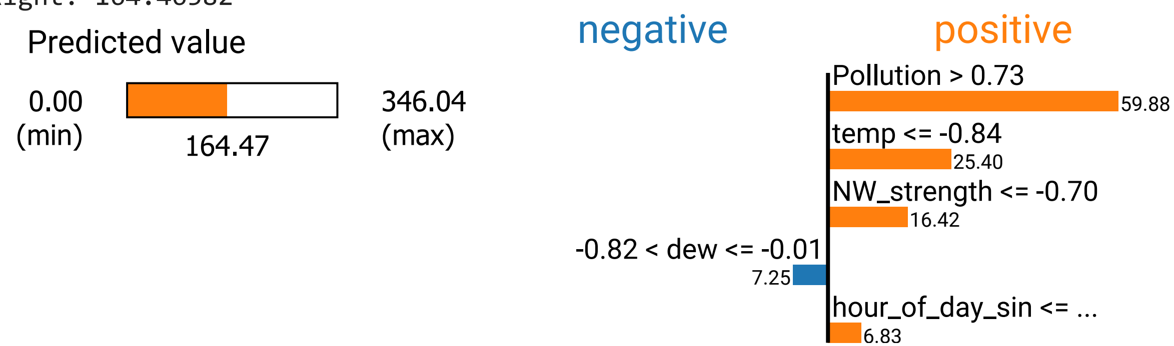
```
# Explain largest and smallest mistakes
print('Largest mistakes:')
for i in largest:
```

```
  print(f'Correct pollution: {r_val_d[i]}')
  exp = explainer.explain_instance(X_val_d[i[0]], model.predict, num_features=5, num_samples=5000)
  exp.show_in_notebook(show_table=False)
  exp.score

print('Smallest mistakes:')
for i in smallest:
  print(f'Correct pollution: {r_val_d[i]}')
  exp = explainer.explain_instance(X_val_d[i[0]], model.predict, num_features=5, num_samples=5000)
  exp.show_in_notebook(show_table=False)
  exp.score
```

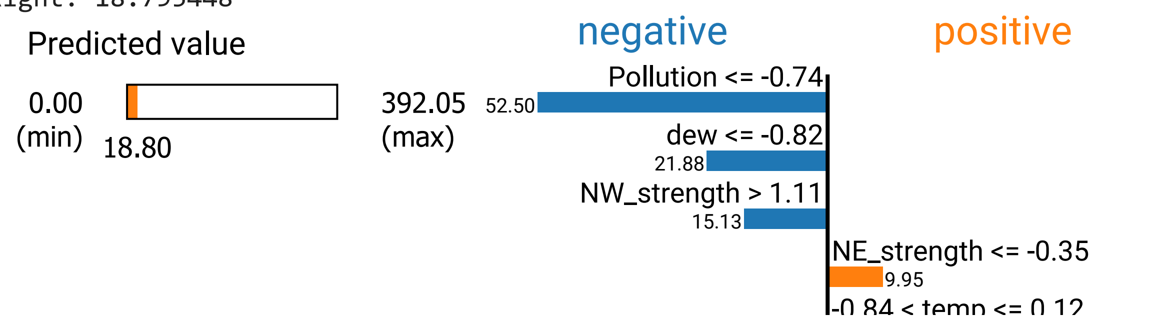
```
Largest mistakes:
Correct pollution: [289.]
Intercept -9.073544251839309
Prediction_local [91.86833571]
Right: 266.85614
```



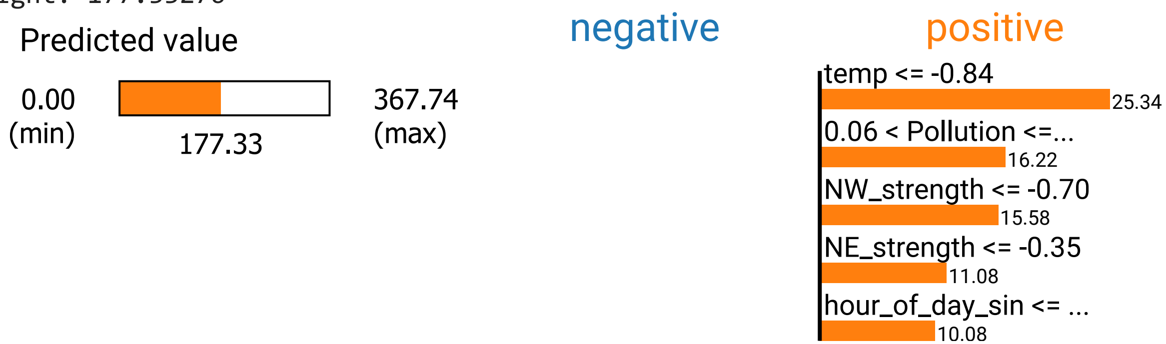

```
Correct pollution: [227.]
Intercept 10.458152778999157
Prediction_local [111.73522731]
Right: 164.46582
```



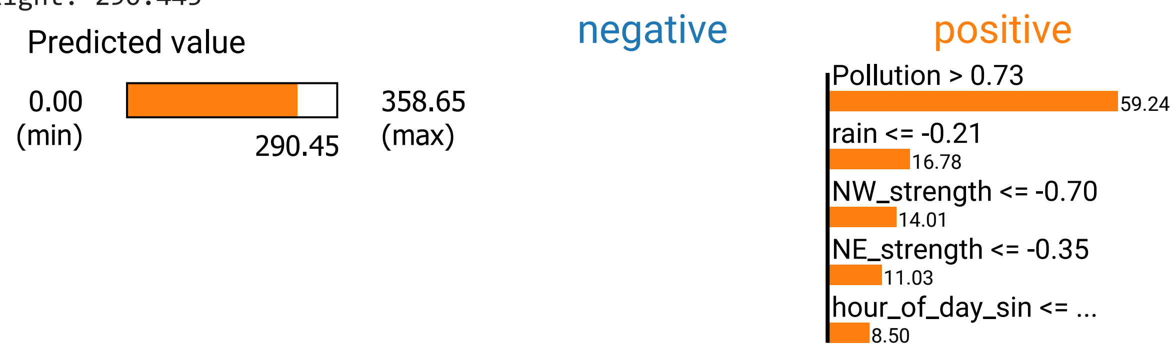

```
Correct pollution: [14.]
Intercept 56.26142526155941
Prediction_local [-15.80824271]
Right: 18.795448
```



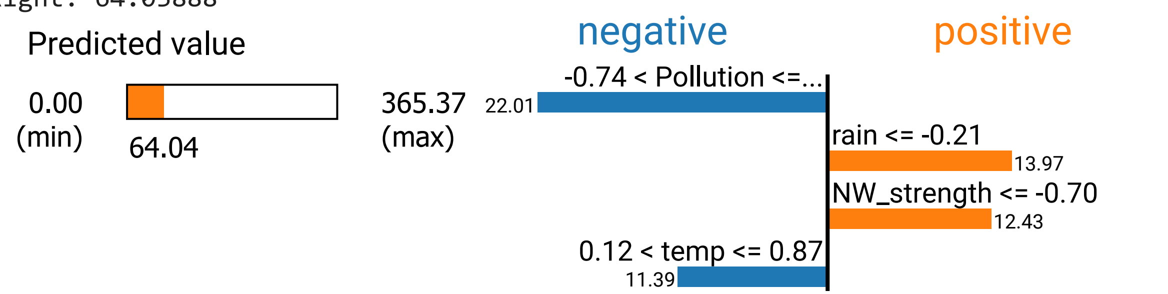

```
Correct pollution: [201.]
Intercept 9.095140395595095
Prediction_local [87.39353018]
Right: 177.33276
```



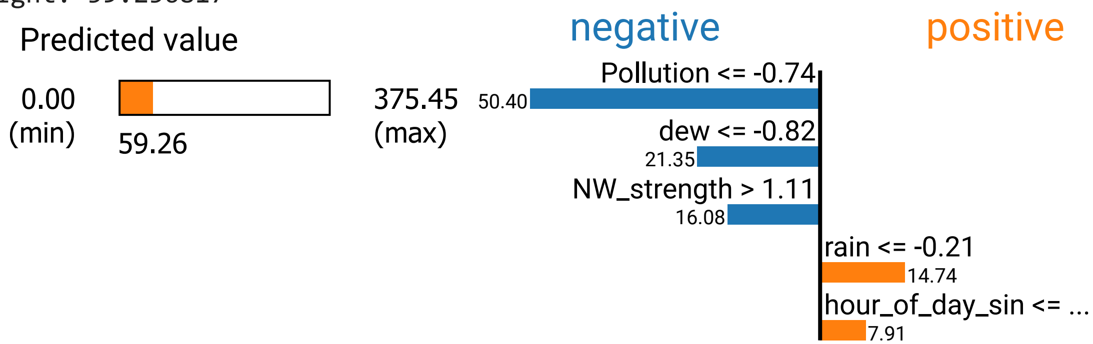

```
Correct pollution: [370.]
Intercept -9.622402032133586
Prediction_local [99.93343762]
Right: 290.445
```



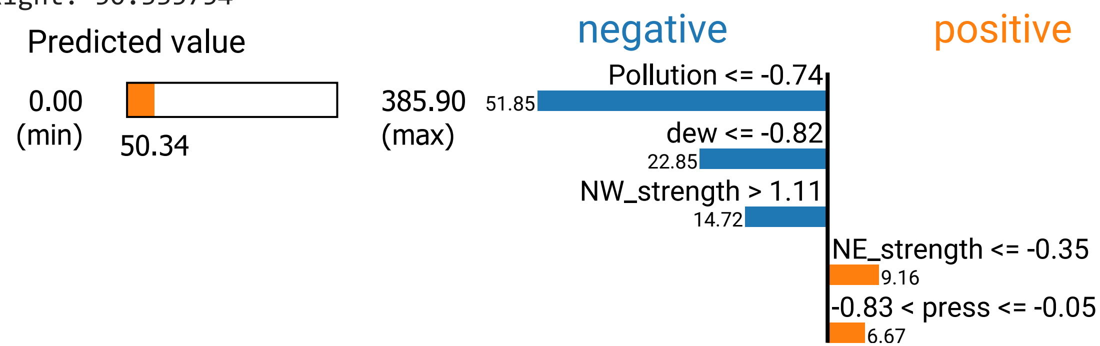

```
Smallest mistakes:
Correct pollution: [59.]
Intercept 28.656407253076047
Prediction_local [30.40274477]
Right: 64.03888
```



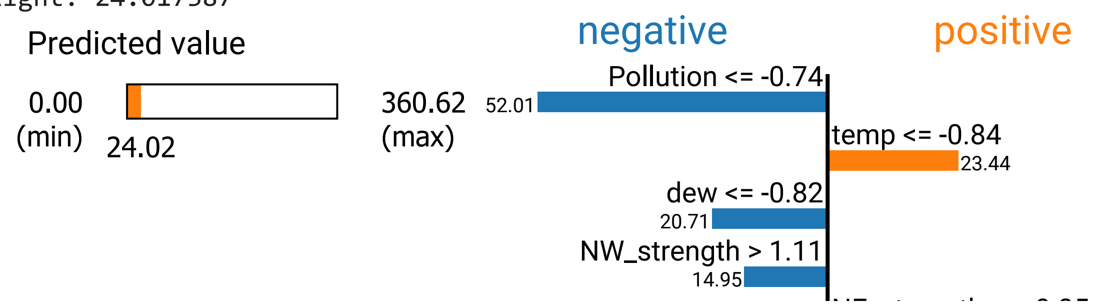

```
Correct pollution: [23.]
Intercept 49.04832864562894
Prediction_local [-16.13255713]
Right: 59.256817
```



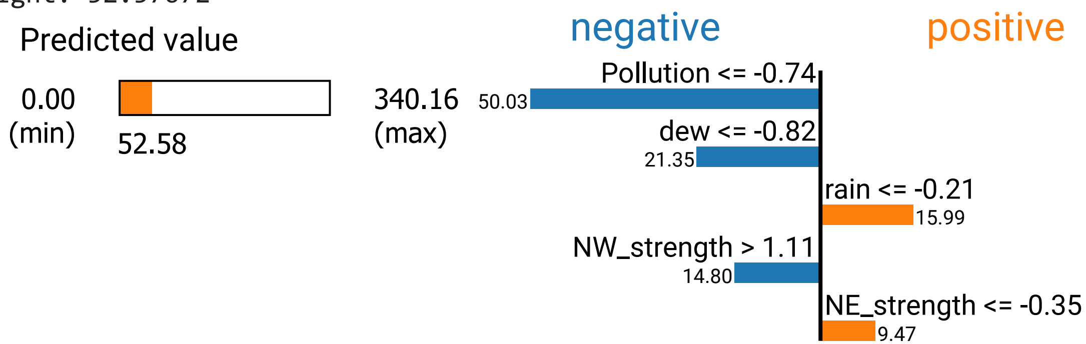

```
Correct pollution: [20.]
Intercept 56.931562378563356
Prediction_local [-16.65907207]
Right: 50.335754
```




```
Correct pollution: [14.]
Intercept 48.11350116855321
Prediction_local [-3.73077516]
Right: 24.017387
```




```
Correct pollution: [23.]
Intercept 41.12963793365804
Prediction_local [-19.58710775]
Right: 52.57672
```



Step 2

Window-based CNN

```
# current code for pollution-only, add other features for multi-feature model,
# remove indices as in commented code if you want to use all features
train_features = [0,]

CNN_WINDOW_SIZE = 168
ahead = 6

# X_train_w,r_train_w = create_dataset_windowed(train_scaled[:,train_features], train_labels, window_size=CNN_WINDOW_SIZE)
# X_val_w,r_val_w = create_dataset_windowed(val_scaled[:,train_features], val_labels, window_size=CNN_WINDOW_SIZE)
X_train_all_w,r_train_all_w = create_dataset_windowed(train_all_scaled[:,train_features],train_all_labels, window_size=CNN_WINDOW_SIZ
X_test_w,r_test_w = create_dataset_windowed(test_scaled[:,train_features],test_labels, window_size=CNN_WINDOW_SIZE)

def conv_model(CNN_WINDOW_SIZE=24, learning_rate=0.01, hidden=[64,128,256]):
  model = Sequential()
```

```
  model.add(Conv1D(filters=hidden[0],
                    kernel_size=2,
                    padding='same',
                    kernel_initializer='he_uniform',
                    input_shape=(CNN_WINDOW_SIZE,len(train_features))))
  model.add(BatchNormalization())
  model.add(Activation('relu'))
  model.add(MaxPooling1D(pool_size=2))

  for idx in range(1, len(hidden)):
    model.add(Conv1D(filters=hidden[idx],
                      kernel_size=2,
                      padding='same',
                      kernel_initializer='he_uniform'))
    model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(MaxPooling1D(pool_size=2))

  model.add(Flatten())
  model.add(Dense(1, kernel_initializer='glorot_uniform'))
  model.add(Activation('linear'))

  optim = tf.keras.optimizers.Adam(lr=learning_rate)
  model.compile(loss='mae',
              optimizer=optim,
              # keep extra metrics: mse and mae without regularisation terms
              metrics=['mse', 'mae'])

  return model
```

```
batch_size = 8*168 # 8 weeks
epochs = 30
learning_rate = 0.001

model = conv_model(CNN_WINDOW_SIZE=CNN_WINDOW_SIZE, learning_rate=learning_rate, hidden=[64,128,256])
model.summary()
```

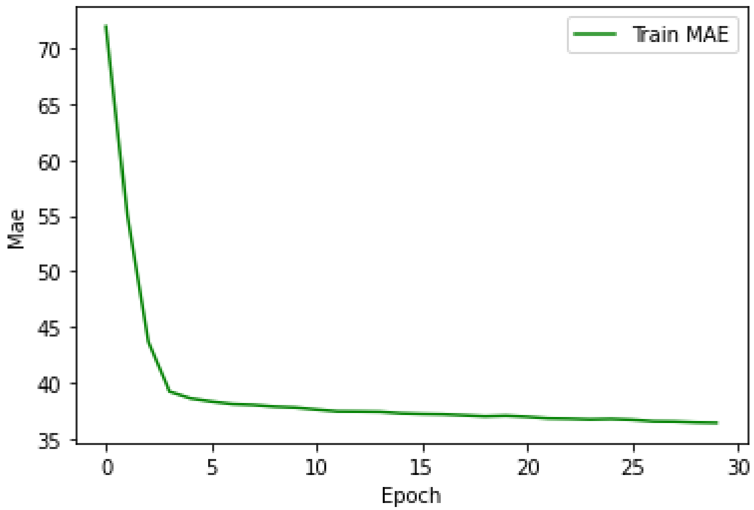
```
# cp_path = '/content/gdrive/My Drive/Colab Notebooks/DL2020/GA3/checkpoints/step_2/cp.ckpt'
# cp_callback = tf.keras.callbacks.ModelCheckpoint(cp_path,
#                                                   monitor='val_mae',
#                                                   verbose=1,
#                                                   save_best_only=True,
#                                                   save_weights_only=True)

# history = model.fit(X_train_w, r_train_w,
#                      shuffle=True,
#                      batch_size=batch_size,
#                      epochs=epochs,
#                      validation_data=(X_val_w,r_val_w),
#                      callbacks=[cp_callback])
# model.load_weights(cp_path)

history = model.fit(X_train_all_w,
                    r_train_all_w,
                    shuffle=True,
                    batch_size=batch_size,
                    epochs=epochs)
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 168, 64)           192
_________________________________________________________________
batch_normalization_4 (Batch (None, 168, 64)           256
_________________________________________________________________
activation_5 (Activation)    (None, 168, 64)           0
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 84, 64)            0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 84, 128)           16512
_________________________________________________________________
batch_normalization_5 (Batch (None, 84, 128)           512
_________________________________________________________________
activation_6 (Activation)    (None, 84, 128)           0
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 42, 128)           0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 42, 256)           65792
_________________________________________________________________
batch_normalization_6 (Batch (None, 42, 256)           1024
_________________________________________________________________
activation_7 (Activation)    (None, 42, 256)           0
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 21, 256)           0
_________________________________________________________________
flatten (Flatten)            (None, 5376)              0
_________________________________________________________________
dense_5 (Dense)              (None, 1)                 5377
_________________________________________________________________
activation_8 (Activation)    (None, 1)                 0
=================================================================
Total params: 89,665
Trainable params: 88,769
Non-trainable params: 896
_________________________________________________________________
Train on 34867 samples
Epoch 1/30
34867/34867 [==============================] - 5s 141us/sample - loss: 71.9665 - mse: 11722.4336 - mae: 71.9666
Epoch 2/30
```

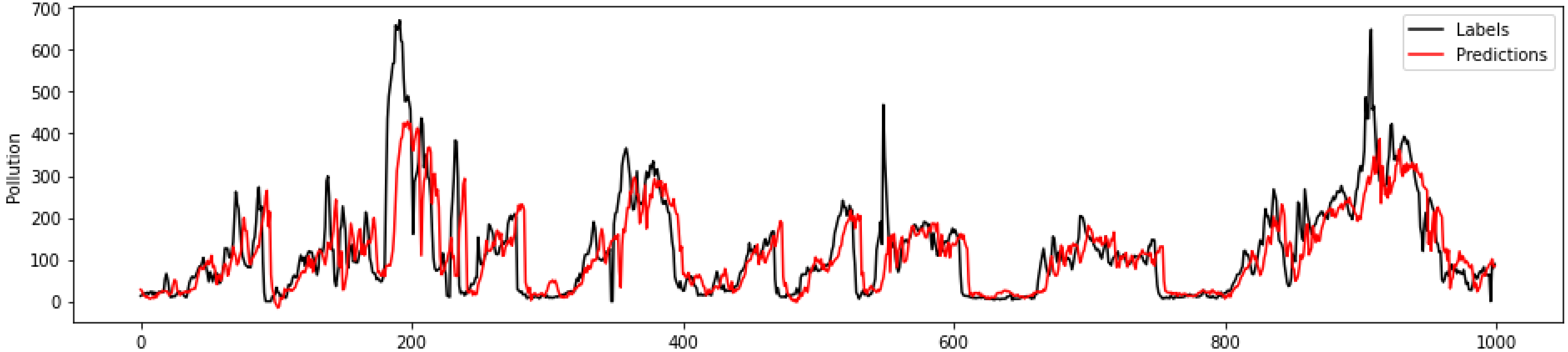
```
Epoch 2/30
34867/34867 [==============================] - 2s 49us/sample - loss: 55.0473 - mse: 6389.7373 - mae: 55.0473
Epoch 3/30
34867/34867 [==============================] - 2s 50us/sample - loss: 43.6378 - mse: 4429.9492 - mae: 43.6378
Epoch 4/30
34867/34867 [==============================] - 2s 50us/sample - loss: 39.2266 - mse: 3833.0310 - mae: 39.2266
Epoch 5/30
34867/34867 [==============================] - 2s 50us/sample - loss: 38.6068 - mse: 3780.6689 - mae: 38.6068
Epoch 6/30
34867/34867 [==============================] - 2s 50us/sample - loss: 38.3187 - mse: 3751.9556 - mae: 38.3187
Epoch 7/30
34867/34867 [==============================] - 2s 50us/sample - loss: 38.0931 - mse: 3731.8708 - mae: 38.0931
Epoch 8/30
34867/34867 [==============================] - 2s 50us/sample - loss: 38.0126 - mse: 3713.8384 - mae: 38.0126
Epoch 9/30
34867/34867 [==============================] - 2s 50us/sample - loss: 37.8579 - mse: 3701.4709 - mae: 37.8579
Epoch 10/30
34867/34867 [==============================] - 2s 51us/sample - loss: 37.7775 - mse: 3684.1226 - mae: 37.7775
Epoch 11/30
34867/34867 [==============================] - 2s 50us/sample - loss: 37.5990 - mse: 3665.1528 - mae: 37.5990
Epoch 12/30
34867/34867 [==============================] - 2s 50us/sample - loss: 37.4350 - mse: 3645.4639 - mae: 37.4350
Epoch 13/30
34867/34867 [==============================] - 2s 50us/sample - loss: 37.4208 - mse: 3638.4688 - mae: 37.4208
Epoch 14/30
34867/34867 [==============================] - 2s 50us/sample - loss: 37.3966 - mse: 3634.8154 - mae: 37.3966
Epoch 15/30
34867/34867 [==============================] - 2s 50us/sample - loss: 37.2579 - mse: 3627.5024 - mae: 37.2579
Epoch 16/30
34867/34867 [==============================] - 2s 50us/sample - loss: 37.2016 - mse: 3617.1348 - mae: 37.2016
Epoch 17/30
34867/34867 [==============================] - 2s 51us/sample - loss: 37.1703 - mse: 3611.6221 - mae: 37.1703
Epoch 18/30
34867/34867 [==============================] - 2s 51us/sample - loss: 37.0839 - mse: 3592.8589 - mae: 37.0839
Epoch 19/30
34867/34867 [==============================] - 2s 50us/sample - loss: 36.9892 - mse: 3588.2493 - mae: 36.9892
Epoch 20/30
34867/34867 [==============================] - 2s 50us/sample - loss: 37.0456 - mse: 3583.0808 - mae: 37.0456
Epoch 21/30
34867/34867 [==============================] - 2s 50us/sample - loss: 36.9462 - mse: 3572.5754 - mae: 36.9462
Epoch 22/30
34867/34867 [==============================] - 2s 50us/sample - loss: 36.8073 - mse: 3556.1616 - mae: 36.8073
Epoch 23/30
```

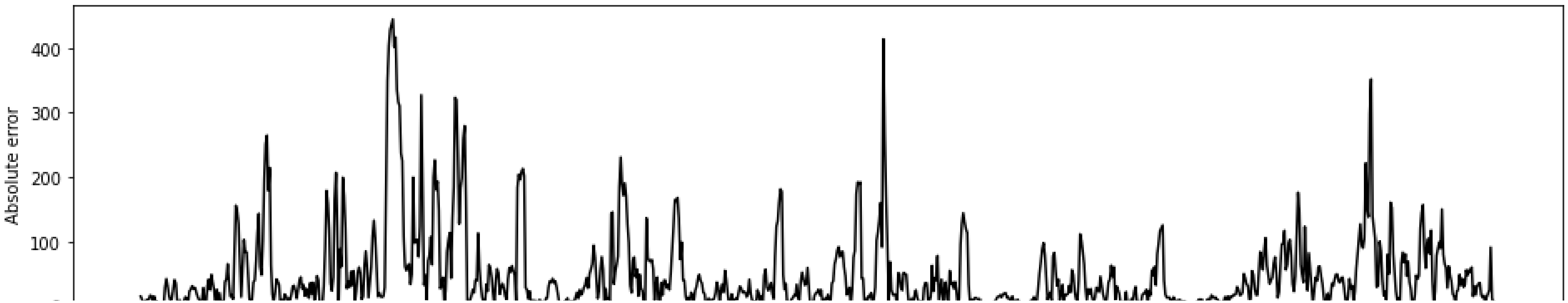
```
34867/34867 [==============================] - 2s 50us/sample - loss: 36.7746 - mse: 3558.0537 - mae: 36.7746
Epoch 24/30
34867/34867 [==============================] - 2s 50us/sample - loss: 36.7149 - mse: 3546.3904 - mae: 36.7149
Epoch 25/30
34867/34867 [==============================] - 2s 50us/sample - loss: 36.7548 - mse: 3548.9934 - mae: 36.7548
Epoch 26/30
34867/34867 [==============================] - 2s 50us/sample - loss: 36.6818 - mse: 3527.5796 - mae: 36.6818
Epoch 27/30
34867/34867 [==============================] - 2s 50us/sample - loss: 36.5393 - mse: 3526.1572 - mae: 36.5393
Epoch 28/30
34867/34867 [==============================] - 2s 49us/sample - loss: 36.5183 - mse: 3508.5530 - mae: 36.5183
Epoch 29/30
34867/34867 [==============================] - 2s 49us/sample - loss: 36.4431 - mse: 3505.9695 - mae: 36.4431
Epoch 30/30
34867/34867 [==============================] - 2s 50us/sample - loss: 36.4045 - mse: 3503.3608 - mae: 36.4045
```

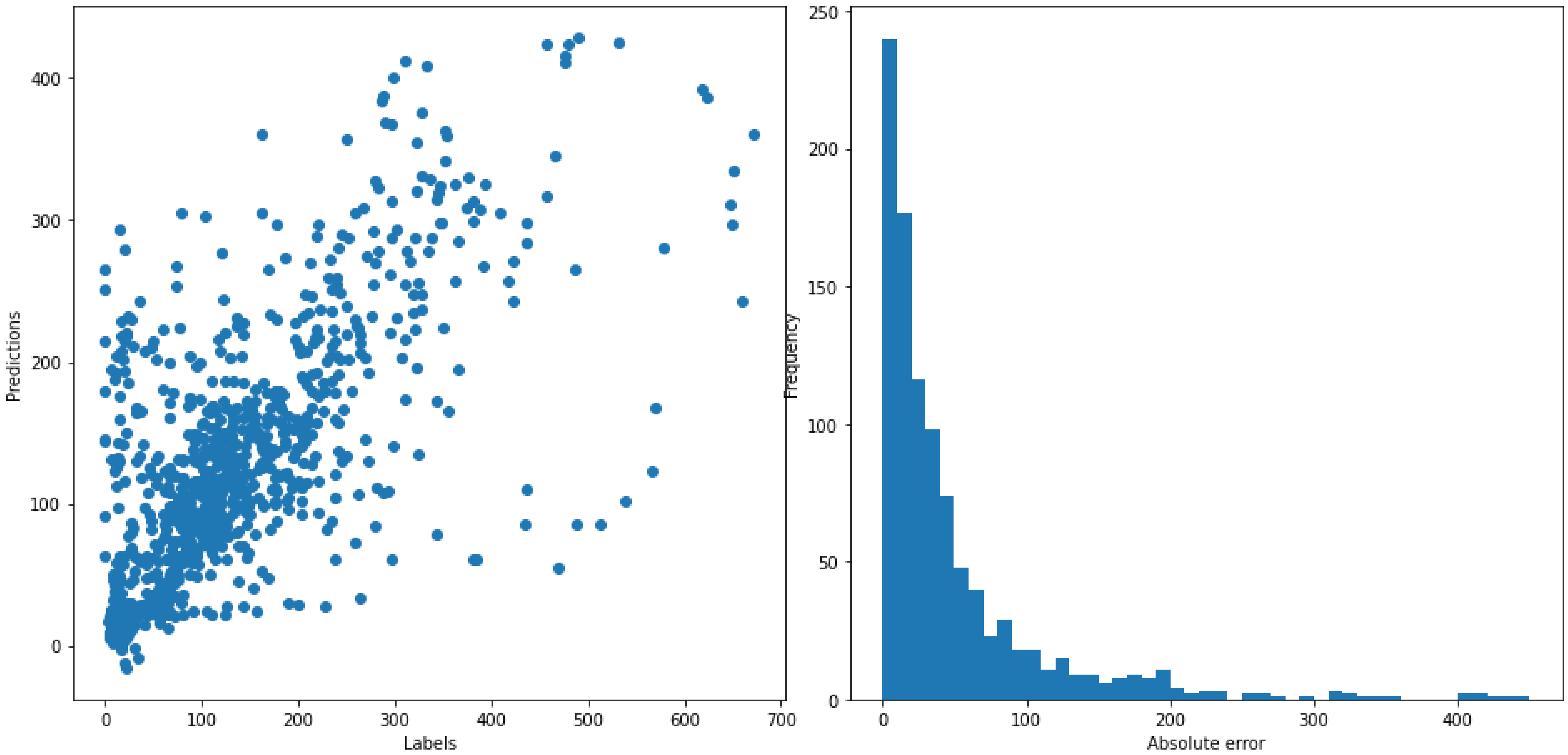
```
plot_history(history)

# y_train = model.predict(X_train_w)
# y_val = model.predict(X_val_w)
y_train_all = model.predict(X_train_all_w)
y_test = model.predict(X_test_w)

# mae_train = mean_absolute_error(r_train_w,y_train)
# mae_val = mean_absolute_error(r_val_w,y_val)
mae_train_all = mean_absolute_error(r_train_all_w,y_train_all)
mae_test = mean_absolute_error(r_test_w,y_test)

# print(f"train      mae: {mae_train}")
# print(f"validation mae: {mae_val}")
print(f"train all mae: {mae_train_all}")
print(f"test      mae: {mae_test}")

# Visualise first 1000 predictions for validation and test
# PlotResults(r_val_w[:1000],y_val[:1000,0])
PlotResults(r_test_w[:1000],y_test[:1000,0])
```




```
train all mae: 36.263267517089844
test      mae: 36.93035888671875
```







LSTM/GRU

```
# current code for pollution-only, add other features for multi-feature model,
# remove indices as in commented code if you want to use all features
train_features = [0,]

GRU_WINDOW_SIZE = 20
ahead = 6
```

```
# For GRU, input has to be between 0 and 1
# max_value_train = np.max(train_scaled[:,train_features])
# min_value_train = np.min(train_scaled[:,train_features])
# max_value_val = np.max(val_scaled[:,train_features])
# min_value_val = np.min(val_scaled[:,train_features])
max_value_train_all = np.max(train_all_scaled[:,train_features])
min_value_train_all = np.min(train_all_scaled[:,train_features])
max_value_test = np.max(test_scaled[:,train_features])
min_value_test = np.min(test_scaled[:,train_features])

# train_scaled_gru = (train_scaled[:,train_features]-min_value_train)/(max_value_train-min_value_train)
# val_scaled_gru = (val_scaled[:,train_features]-min_value_val)/(max_value_val-min_value_val)
train_all_scaled_gru = (train_all_scaled[:,train_features]-min_value_train_all)/(max_value_train_all-min_value_train_all)
test_scaled_gru = (test_scaled[:,train_features]-min_value_test)/(max_value_test-min_value_test)

# X_train_gru,r_train_gru = create_dataset_windowed(train_scaled_gru, train_labels, window_size=GRU_WINDOW_SIZE)
# X_val_gru,r_val_gru = create_dataset_windowed(val_scaled_gru, val_labels, window_size=GRU_WINDOW_SIZE)
X_train_all_gru,r_train_all_gru = create_dataset_windowed(train_all_scaled_gru, train_all_labels, window_size=GRU_WINDOW_SIZE)
X_test_gru,r_test_gru = create_dataset_windowed(test_scaled_gru, test_labels, window_size=GRU_WINDOW_SIZE)

def gru_model(GRU_WINDOW_SIZE=24, units=512, learning_rate=0.001):
  model = Sequential()

  model.add(GRU(units=units,
              return_sequences=False,
              implementation=1,
              input_shape=(GRU_WINDOW_SIZE,1)))
  model.add(Dropout(0.5))
  model.add(Dense(1))
  model.add(Activation('linear'))

  optim = tf.keras.optimizers.SGD(learning_rate=learning_rate,momentum=0.9,clipnorm=9)
  model.compile(loss='mae',
              optimizer=optim,
              # keep extra metrics: mse and mae without regularisation terms
              metrics=['mse', 'mae'])
  return model
```

```
    return model
```

```
batch_size = 2*168
epochs = 50
learning_rate = 0.01

model = gru_model(GRU_WINDOW_SIZE=GRU_WINDOW_SIZE, units=512, learning_rate=learning_rate)
model.summary()

# cp_path = '/content/gdrive/My Drive/Colab Notebooks/DL2020/GA3/checkpoints/step_2/cp.ckpt'
# cp_callback = tf.keras.callbacks.ModelCheckpoint(cp_path,
#                                                    monitor='val_mae',
#                                                    verbose=1,
#                                                    save_best_only=True,
#                                                    save_weights_only=True)
# es_callback = tf.keras.callbacks.EarlyStopping(monitor='val_mae',
#                                                  min_delta=0.0001,
#                                                  patience=5)
# history = model.fit(X_train_gru,
#                       r_train_gru,
#                       shuffle=True,
#                       batch_size=batch_size,
#                       epochs=epochs,
#                       validation_data=(X_val_gru,r_val_gru),
#                       callbacks=[cp_callback, es_callback])
# model.load_weights(cp_path)

history = model.fit(X_train_all_gru,
                    r_train_all_gru,
                    shuffle=True,
                    batch_size=batch_size,
                    epochs=epochs)
```

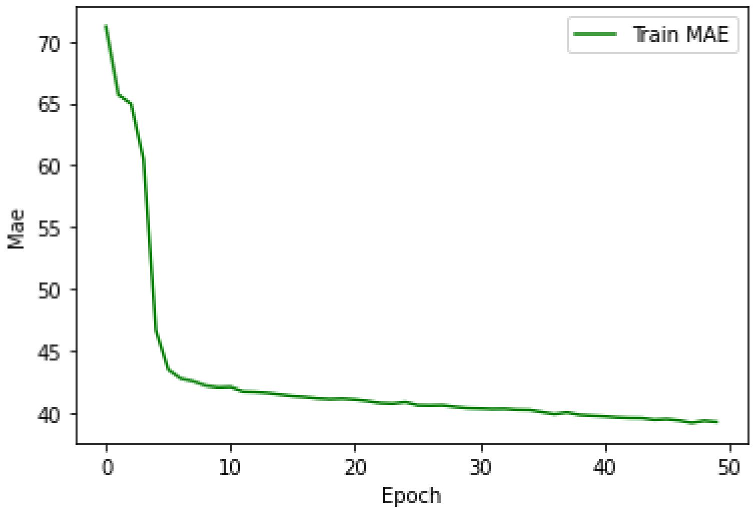
```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
gru (GRU)                    (None, 512)               791040
_________________________________________________________________
dropout_4 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_6 (Dense)              (None, 1)                 513
_________________________________________________________________
activation_9 (Activation)    (None, 1)                 0
=================================================================
Total params: 791,553
Trainable params: 791,553
Non-trainable params: 0
_________________________________________________________________
Train on 35015 samples
Epoch 1/50
35015/35015 [==============================] - 2s 69us/sample - loss: 71.2187 - mse: 10740.4189 - mae: 71.2187
Epoch 2/50
35015/35015 [==============================] - 1s 42us/sample - loss: 65.6855 - mse: 9135.8750 - mae: 65.6854
Epoch 3/50
35015/35015 [==============================] - 1s 42us/sample - loss: 64.9366 - mse: 8905.2334 - mae: 64.9367
Epoch 4/50
35015/35015 [==============================] - 1s 42us/sample - loss: 60.5093 - mse: 8061.3975 - mae: 60.5093
Epoch 5/50
35015/35015 [==============================] - 1s 42us/sample - loss: 46.6006 - mse: 5129.3594 - mae: 46.6006
Epoch 6/50
35015/35015 [==============================] - 1s 42us/sample - loss: 43.4271 - mse: 4405.7178 - mae: 43.4271
Epoch 7/50
35015/35015 [==============================] - 1s 42us/sample - loss: 42.7156 - mse: 4272.8228 - mae: 42.7156
Epoch 8/50
35015/35015 [==============================] - 2s 43us/sample - loss: 42.4921 - mse: 4234.6924 - mae: 42.4921
Epoch 9/50
35015/35015 [==============================] - 1s 42us/sample - loss: 42.1491 - mse: 4189.7827 - mae: 42.1491
Epoch 10/50
35015/35015 [==============================] - 2s 43us/sample - loss: 42.0081 - mse: 4172.7471 - mae: 42.0081
Epoch 11/50
35015/35015 [==============================] - 1s 43us/sample - loss: 42.0503 - mse: 4174.9243 - mae: 42.0503
Epoch 12/50
35015/35015 [==============================] - 1s 43us/sample - loss: 41.6807 - mse: 4139.4043 - mae: 41.6807
Epoch 13/50
```

```
Epoch 13/50
35015/35015 [==============================] - 1s 43us/sample - loss: 41.6494 - mse: 4130.2139 - mae: 41.6494
Epoch 14/50
35015/35015 [==============================] - 1s 43us/sample - loss: 41.5756 - mse: 4127.7944 - mae: 41.5756
Epoch 15/50
35015/35015 [==============================] - 2s 44us/sample - loss: 41.4433 - mse: 4117.9678 - mae: 41.4433
Epoch 16/50
35015/35015 [==============================] - 2s 43us/sample - loss: 41.3191 - mse: 4095.5193 - mae: 41.3191
Epoch 17/50
35015/35015 [==============================] - 1s 43us/sample - loss: 41.2442 - mse: 4095.8853 - mae: 41.2442
Epoch 18/50
35015/35015 [==============================] - 2s 43us/sample - loss: 41.1335 - mse: 4095.1890 - mae: 41.1335
Epoch 19/50
35015/35015 [==============================] - 2s 43us/sample - loss: 41.0751 - mse: 4077.7515 - mae: 41.0751
Epoch 20/50
35015/35015 [==============================] - 2s 43us/sample - loss: 41.1148 - mse: 4081.1523 - mae: 41.1148
Epoch 21/50
35015/35015 [==============================] - 1s 43us/sample - loss: 41.0450 - mse: 4079.3882 - mae: 41.0450
Epoch 22/50
35015/35015 [==============================] - 2s 43us/sample - loss: 40.9255 - mse: 4050.8711 - mae: 40.9255
Epoch 23/50
35015/35015 [==============================] - 2s 43us/sample - loss: 40.7574 - mse: 4035.2146 - mae: 40.7574
Epoch 24/50
35015/35015 [==============================] - 2s 43us/sample - loss: 40.7147 - mse: 4037.5176 - mae: 40.7147
Epoch 25/50
35015/35015 [==============================] - 1s 42us/sample - loss: 40.8396 - mse: 4037.1360 - mae: 40.8396
Epoch 26/50
35015/35015 [==============================] - 2s 43us/sample - loss: 40.5780 - mse: 4010.3914 - mae: 40.5780
Epoch 27/50
35015/35015 [==============================] - 1s 43us/sample - loss: 40.5598 - mse: 4028.8118 - mae: 40.5598
Epoch 28/50
35015/35015 [==============================] - 1s 43us/sample - loss: 40.5789 - mse: 4019.4075 - mae: 40.5789
Epoch 29/50
35015/35015 [==============================] - 2s 43us/sample - loss: 40.4464 - mse: 4007.7205 - mae: 40.4464
Epoch 30/50
35015/35015 [==============================] - 2s 43us/sample - loss: 40.3474 - mse: 3996.6465 - mae: 40.3474
Epoch 31/50
35015/35015 [==============================] - 2s 43us/sample - loss: 40.3192 - mse: 3989.1357 - mae: 40.3192
Epoch 32/50
35015/35015 [==============================] - 2s 43us/sample - loss: 40.2790 - mse: 3992.3813 - mae: 40.2790
Epoch 33/50
35015/35015 [==============================] - 2s 43us/sample - loss: 40.2918 - mse: 3993.8240 - mae: 40.2918
Epoch 34/50
```

```
35015/35015 [==============================] - 2s 44us/sample - loss: 40.2196 - mse: 3994.0249 - mae: 40.2196
Epoch 35/50
35015/35015 [==============================] - 2s 43us/sample - loss: 40.2009 - mse: 3979.3872 - mae: 40.2009
Epoch 36/50
35015/35015 [==============================] - 2s 43us/sample - loss: 40.0268 - mse: 3957.9846 - mae: 40.0269
Epoch 37/50
35015/35015 [==============================] - 1s 43us/sample - loss: 39.8587 - mse: 3937.4265 - mae: 39.8587
Epoch 38/50
35015/35015 [==============================] - 1s 43us/sample - loss: 40.0057 - mse: 3965.1650 - mae: 40.0057
Epoch 39/50
35015/35015 [==============================] - 2s 43us/sample - loss: 39.7988 - mse: 3925.2485 - mae: 39.7988
Epoch 40/50
35015/35015 [==============================] - 1s 43us/sample - loss: 39.7337 - mse: 3909.7881 - mae: 39.7337
Epoch 41/50
35015/35015 [==============================] - 2s 44us/sample - loss: 39.6805 - mse: 3909.2004 - mae: 39.6805
Epoch 42/50
35015/35015 [==============================] - 2s 44us/sample - loss: 39.5995 - mse: 3902.3271 - mae: 39.5995
Epoch 43/50
35015/35015 [==============================] - 2s 43us/sample - loss: 39.5505 - mse: 3894.9324 - mae: 39.5505
Epoch 44/50
35015/35015 [==============================] - 2s 43us/sample - loss: 39.5393 - mse: 3882.5322 - mae: 39.5393
Epoch 45/50
35015/35015 [==============================] - 2s 43us/sample - loss: 39.4026 - mse: 3867.6694 - mae: 39.4026
Epoch 46/50
35015/35015 [==============================] - 2s 44us/sample - loss: 39.4573 - mse: 3872.0000 - mae: 39.4573
Epoch 47/50
35015/35015 [==============================] - 2s 44us/sample - loss: 39.3481 - mse: 3853.4172 - mae: 39.3481
Epoch 48/50
35015/35015 [==============================] - 2s 43us/sample - loss: 39.1527 - mse: 3848.9690 - mae: 39.1527
Epoch 49/50
35015/35015 [==============================] - 2s 43us/sample - loss: 39.3224 - mse: 3852.9966 - mae: 39.3224
Epoch 50/50
35015/35015 [==============================] - 2s 43us/sample - loss: 39.2304 - mse: 3834.1753 - mae: 39.2304
```

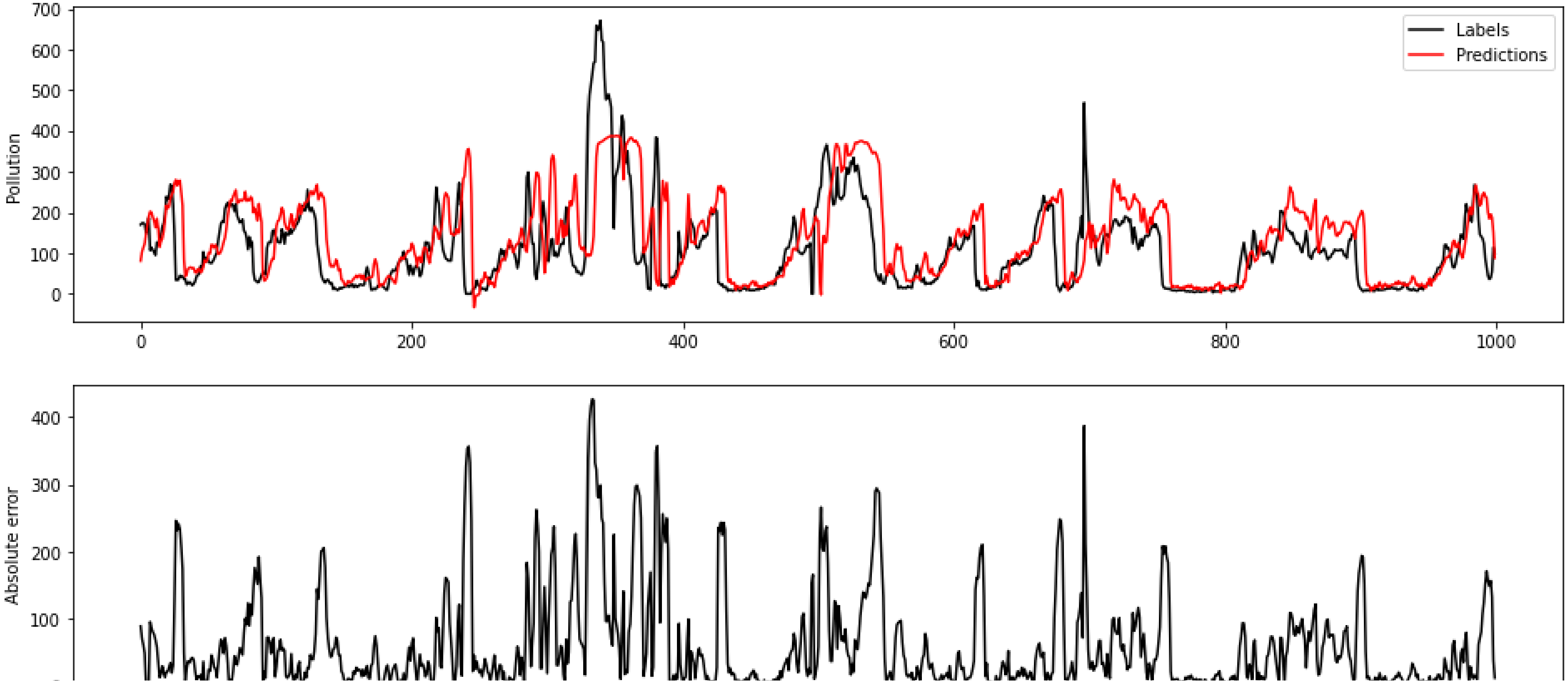
```
plot_history(history)

# y_train = model.predict(X_train_gru)
# y_val = model.predict(X_val_gru)
y_train_all = model.predict(X_train_all_gru)
```

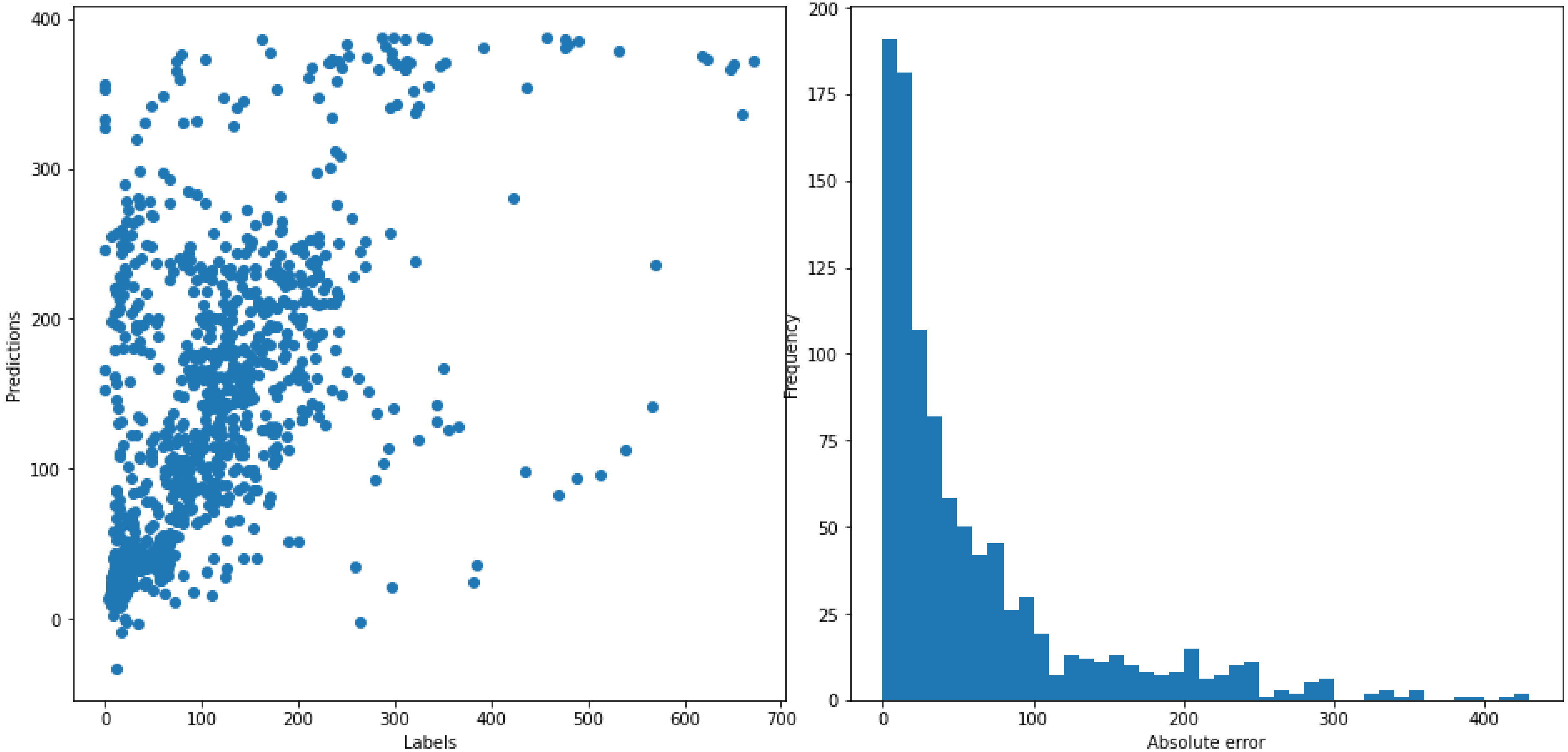
```
y_test = model.predict(X_test_gru)

# mae_train = mean_absolute_error(r_train_gru,y_train)
# mae_val = mean_absolute_error(r_val_gru,y_val)
mae_train_all = mean_absolute_error(r_train_all_gru,y_train_all)
mae_test = mean_absolute_error(r_test_gru,y_test)

# print(f"train      mae: {mae_train}")
# print(f"validation mae: {mae_val}")
print(f"train      mae: {mae_train_all}")
print(f"test mae: {mae_test}")

# Visualise first 1000 predictions for validation and test
# PlotResults(r_val_gru[:1000],y_val[:1000,0])
PlotResults(r_test_gru[:1000],y_test[:1000,0])
```



train   mae: 38.366157531738 28

test mae: 49.90842056274414





Step 3

```
# current code for pollution-only, add other features for multi-feature model,
# remove indices as in commented code if you want to use all features
train_features = np.arange(14) # 16: all, 14: without snow and rain, 13: also leave pressure out, 12: also leave temperature out

CNN_WINDOW_SIZE = 2*168 # 2 weeks
ahead = 6
```

```
ahead = 6

# X_train_w,r_train_w = create_dataset_windowed(train_scaled[:,train_features], train_labels, window_size=CNN_WINDOW_SIZE)
# X_val_w,r_val_w = create_dataset_windowed(val_scaled[:,train_features], val_labels, window_size=CNN_WINDOW_SIZE)
X_train_all_w,r_train_all_w = create_dataset_windowed(train_all_scaled[:,train_features],train_all_labels, window_size=CNN_WINDOW_SIZ
X_test_w,r_test_w = create_dataset_windowed(test_scaled[:,train_features],test_labels, window_size=CNN_WINDOW_SIZE)

def conv_model(CNN_WINDOW_SIZE=24, learning_rate=0.01, hidden=[32,64,128,256]):
  model = Sequential()

  model.add(Conv1D(filters=hidden[0],
                  kernel_size=2,
                  padding='same',
                  kernel_initializer='he_uniform',
                  input_shape=(CNN_WINDOW_SIZE,len(train_features))))
  # model.add(BatchNormalization())
  model.add(Activation('relu'))
  model.add(MaxPooling1D(pool_size=2))

  for idx in range(1, len(hidden)):
    model.add(Conv1D(filters=hidden[idx],
                    kernel_size=2,
                    padding='same',
                    kernel_initializer='he_uniform'))
    # model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(MaxPooling1D(pool_size=2))

  model.add(Flatten())
  model.add(Dense(1, kernel_initializer='glorot_uniform'))
  model.add(Activation('linear'))

  optim = tf.keras.optimizers.Adam(learning_rate=learning_rate)
  model.compile(loss='mae',
                optimizer=optim,
                # keep extra metrics: mse and mae without regularisation terms
                metrics=['mse', 'mae'])
```

```
  return model
```

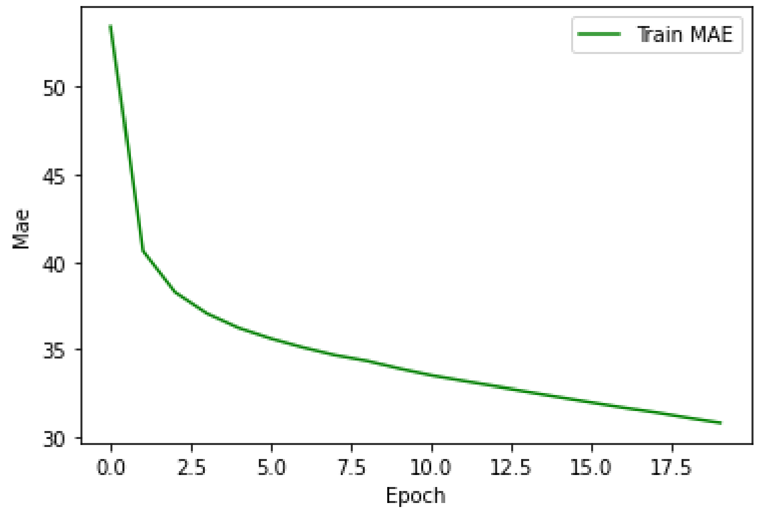
```
batch_size = 24
epochs = 20
learning_rate = 0.0001

model = conv_model(CNN_WINDOW_SIZE=CNN_WINDOW_SIZE, learning_rate=learning_rate, hidden=[32,64,128,256])
model.summary()

# cp_path = '/content/gdrive/My Drive/Colab Notebooks/DL2020/GA3/checkpoints/step_3/cp.ckpt'
# cp_callback = tf.keras.callbacks.ModelCheckpoint(cp_path,
#                                                   monitor='val_mae',
#                                                   verbose=1,
#                                                   save_best_only=True,
#                                                   save_weights_only=True)

# history = model.fit(X_train_w, r_train_w,
#                      shuffle=True,
#                      batch_size=batch_size,
#                      epochs=epochs,
#                      validation_data=(X_val_w,r_val_w),
#                      callbacks=[cp_callback])
# model.load_weights(cp_path)

history = model.fit(X_train_all_w,
                    r_train_all_w,
                    shuffle=True,
                    batch_size=batch_size,
                    epochs=epochs)
```

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv1d_3 (Conv1D)            (None, 336, 32)           928       
_________________________________________________________________
activation_10 (Activation)   (None, 336, 32)           0         
_________________________________________________________________
max_pooling1d_3 (MaxPooling1 (None, 168, 32)           0         
_________________________________________________________________
conv1d_4 (Conv1D)            (None, 168, 64)           4160      
_________________________________________________________________
activation_11 (Activation)   (None, 168, 64)           0         
_________________________________________________________________
max_pooling1d_4 (MaxPooling1 (None, 84, 64)            0         
_________________________________________________________________
conv1d_5 (Conv1D)            (None, 84, 128)           16512     
_________________________________________________________________
activation_12 (Activation)   (None, 84, 128)           0         
_________________________________________________________________
max_pooling1d_5 (MaxPooling1 (None, 42, 128)           0         
_________________________________________________________________
conv1d_6 (Conv1D)            (None, 42, 256)           65792     
_________________________________________________________________
activation_13 (Activation)   (None, 42, 256)           0         
_________________________________________________________________
max_pooling1d_6 (MaxPooling1 (None, 21, 256)           0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 5376)              0         
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 5377      
_________________________________________________________________
activation_14 (Activation)   (None, 1)                 0         
=================================================================
Total params: 92,769
Trainable params: 92,769
Non-trainable params: 0
_________________________________________________________________
Train on 34699 samples
Epoch 1/20
34699/34699 [==============================] - 8s 221us/sample - loss: 53.4678 - mse: 6566.6738 - mae: 53.4678
Epoch 2/20
```

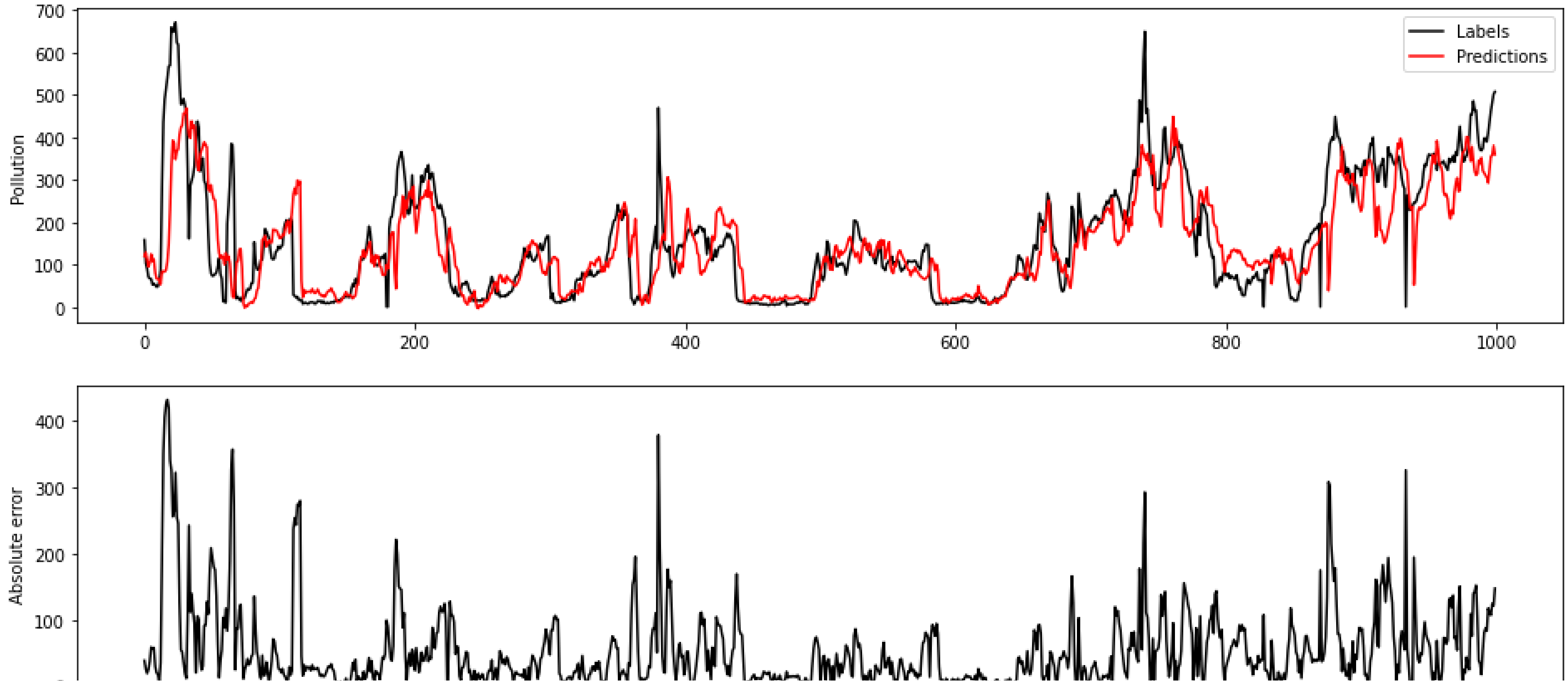
```
Epoch 2/20
34699/34699 [==============================] - 6s 181us/sample - loss: 40.6294 - mse: 3859.4417 - mae: 40.6294
Epoch 3/20
34699/34699 [==============================] - 6s 184us/sample - loss: 38.2617 - mse: 3574.7573 - mae: 38.2617
Epoch 4/20
34699/34699 [==============================] - 6s 185us/sample - loss: 37.0370 - mse: 3434.5955 - mae: 37.0370
Epoch 5/20
34699/34699 [==============================] - 6s 183us/sample - loss: 36.2029 - mse: 3347.6392 - mae: 36.2030
Epoch 6/20
34699/34699 [==============================] - 6s 180us/sample - loss: 35.5974 - mse: 3270.3047 - mae: 35.5974
Epoch 7/20
34699/34699 [==============================] - 7s 187us/sample - loss: 35.0903 - mse: 3217.6736 - mae: 35.0903
Epoch 8/20
34699/34699 [==============================] - 7s 198us/sample - loss: 34.6493 - mse: 3166.6382 - mae: 34.6493
Epoch 9/20
34699/34699 [==============================] - 7s 192us/sample - loss: 34.3211 - mse: 3121.6841 - mae: 34.3211
Epoch 10/20
34699/34699 [==============================] - 6s 184us/sample - loss: 33.8776 - mse: 3079.3103 - mae: 33.8776
Epoch 11/20
34699/34699 [==============================] - 6s 185us/sample - loss: 33.4903 - mse: 3039.8474 - mae: 33.4903
Epoch 12/20
34699/34699 [==============================] - 6s 184us/sample - loss: 33.1742 - mse: 2996.2019 - mae: 33.1742
Epoch 13/20
34699/34699 [==============================] - 6s 184us/sample - loss: 32.8745 - mse: 2960.5127 - mae: 32.8745
Epoch 14/20
34699/34699 [==============================] - 6s 185us/sample - loss: 32.5650 - mse: 2924.0540 - mae: 32.5650
Epoch 15/20
34699/34699 [==============================] - 6s 184us/sample - loss: 32.2604 - mse: 2883.9709 - mae: 32.2604
Epoch 16/20
34699/34699 [==============================] - 6s 184us/sample - loss: 31.9605 - mse: 2843.4626 - mae: 31.9605
Epoch 17/20
34699/34699 [==============================] - 7s 190us/sample - loss: 31.6633 - mse: 2809.0984 - mae: 31.6633
Epoch 18/20
34699/34699 [==============================] - 6s 187us/sample - loss: 31.3906 - mse: 2778.4553 - mae: 31.3906
Epoch 19/20
34699/34699 [==============================] - 6s 184us/sample - loss: 31.0827 - mse: 2738.3994 - mae: 31.0827
Epoch 20/20
34699/34699 [==============================] - 6s 185us/sample - loss: 30.7971 - mse: 2704.5264 - mae: 30.7971
```

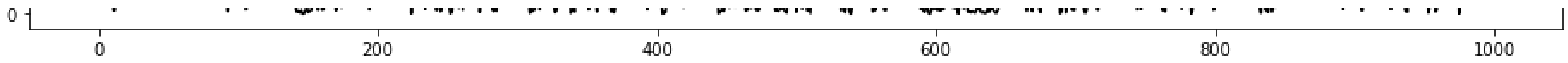
```
plot_history(history)
```

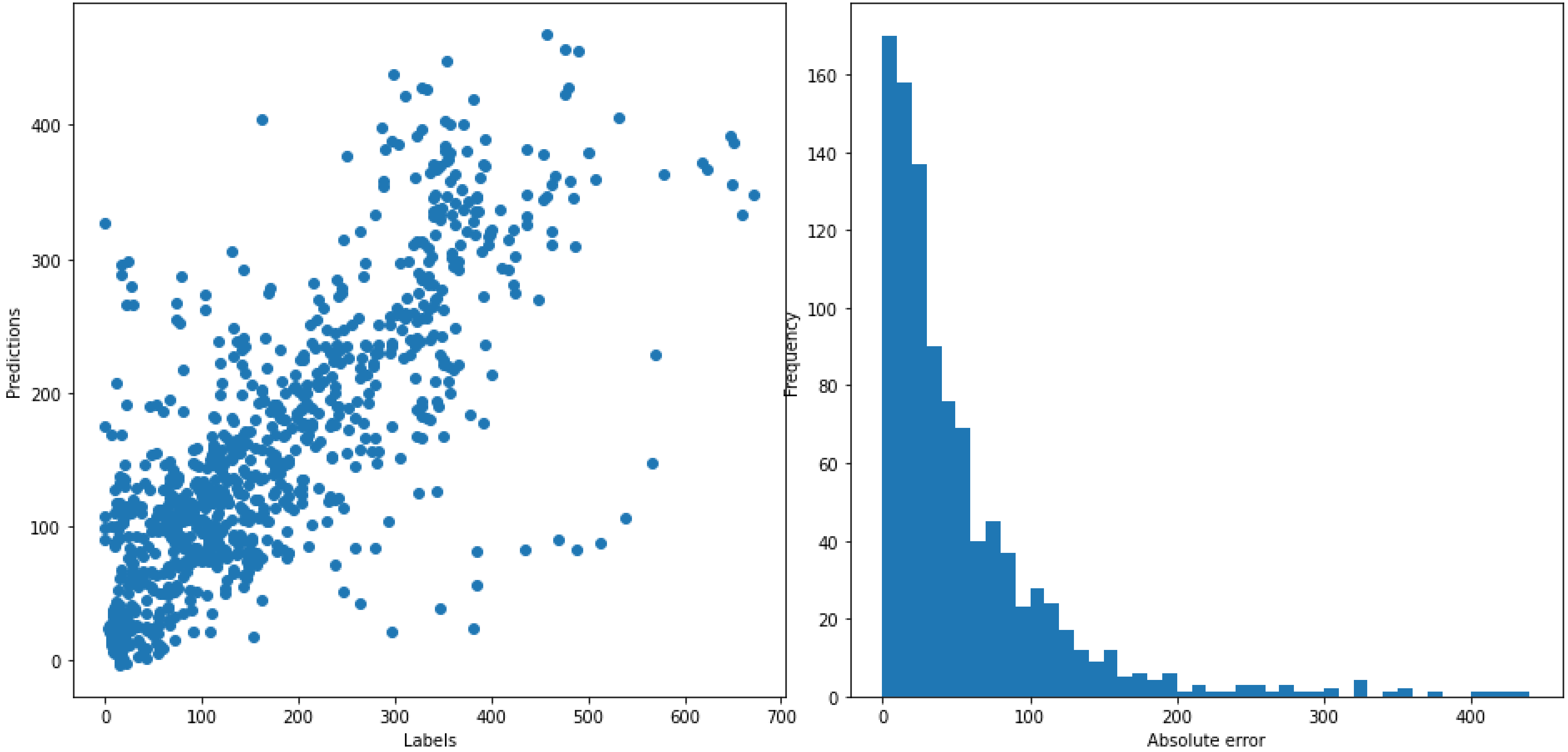
```
# y_train = model.predict(X_train_w)
# y_val = model.predict(X_val_w)
y_train_all = model.predict(X_train_all_w)
y_test = model.predict(X_test_w)

# mae_train = mean_absolute_error(r_train_w,y_train)
# mae_val = mean_absolute_error(r_val_w,y_val)
mae_train_all = mean_absolute_error(r_train_all_w,y_train_all)
mae_test = mean_absolute_error(r_test_w,y_test)

# print(f"train      mae: {mae_train}")
# print(f"validation mae: {mae_val}")
print(f"train all mae: {mae_train_all}")
print(f"test      mae: {mae_test}")

# Visualise first 1000 predictions for validation and test
# PlotResults(r_val_w[:1000],y_val[:1000,0])
PlotResults(r_test_w[:1000],y_test[:1000,0])
```




```
train all mae: 30.459682464599610
test      mae: 36.19908905029297
```







Step 4

Models

```
def conv_model(CNN_WINDOW_SIZE=24, learning_rate=0.01, hidden=[32,64,128,256], n_features=16):
  model = Sequential()
```

```
  model.add(Conv1D(filters=hidden[0],
                   kernel_size=2,
                   padding='same',
                   kernel_initializer='he_uniform',
                   input_shape=(CNN_WINDOW_SIZE,n_features)))
  # model.add(BatchNormalization())
  model.add(Activation('relu'))
  model.add(MaxPooling1D(pool_size=2))

  for idx in range(1, len(hidden)):
    model.add(Conv1D(filters=hidden[idx],
                     kernel_size=2,
                     padding='same',
                     kernel_initializer='he_uniform'))
    # model.add(BatchNormalization())
    model.add(Activation('relu'))
    model.add(MaxPooling1D(pool_size=2))

  model.add(Flatten())
  model.add(Dense(1, kernel_initializer='glorot_uniform'))
  model.add(Activation('linear'))

  optim = tf.keras.optimizers.Adam(learning_rate=learning_rate)
  model.compile(loss='mae',
                optimizer=optim,
                # keep extra metrics: mse and mae without regularisation terms
                metrics=['mse', 'mae'])

  return model
```

```
batch_size = 24
epochs = 20
learning_rate = 0.0001
CNN_WINDOW_SIZE = 2*168 # 2 weeks
ahead = 6
```

```
# 1
train_features = np.arange(16) # 16: all

# X_train_w,r_train_w = create_dataset_windowed(train_scaled[:,train_features], train_labels, window_size=CNN_WINDOW_SIZE)
# X_val_w,r_val_w = create_dataset_windowed(val_scaled[:,train_features], val_labels, window_size=CNN_WINDOW_SIZE)
X_train_all_w,r_train_all_w = create_dataset_windowed(train_all_scaled[:,train_features],train_all_labels, window_size=CNN_WINDOW_SIZ
X_test_w,r_test_w = create_dataset_windowed(test_scaled[:,train_features],test_labels, window_size=CNN_WINDOW_SIZE)

model = conv_model(CNN_WINDOW_SIZE=CNN_WINDOW_SIZE, learning_rate=learning_rate, hidden=[32,64,128,256], n_features=len(train_feature
# model.summary()

# cp_path = '/content/gdrive/My Drive/Colab Notebooks/DL2020/GA3/checkpoints/step_4/cp.ckpt'
# cp_callback = tf.keras.callbacks.ModelCheckpoint(cp_path,
#                                                   monitor='val_mae',
#                                                   verbose=1,
#                                                   save_best_only=True,
#                                                   save_weights_only=True)

# history = model.fit(X_train_w, r_train_w,
#                     shuffle=True,
#                     batch_size=batch_size,
#                     epochs=epochs,
#                     validation_data=(X_val_w,r_val_w),
#                     callbacks=[cp_callback])
# model.load_weights(cp_path)

history = model.fit(X_train_all_w, r_train_all_w, shuffle=True, batch_size=batch_size, epochs=epochs)

# y_train_1 = model.predict(X_train_w)
# y_val_1 = model.predict(X_val_w)
y_train_all_1 = model.predict(X_train_all_w)
y_test_1 = model.predict(X_test_w)

# mae_train = mean_absolute_error(r_train_w,y_train_1)
# mae_val = mean_absolute_error(r_val_w,y_val_1)
mae_train_all = mean_absolute_error(r_train_all_w,y_train_all_1)
mae_test = mean_absolute_error(r_test_w,y_test_1)
```

```
# print(f"train      mae: {mae_train}")
# print(f"validation mae: {mae_val}")
print(f"train all mae: {mae_train_all}")
print(f"test      mae: {mae_test}")
```

```
Train on 34699 samples
Epoch 1/20
34699/34699 [==============================] - 7s 203us/sample - loss: 54.8534 - mse: 6857.3096 - mae: 54.8535
Epoch 2/20
34699/34699 [==============================] - 6s 187us/sample - loss: 40.8519 - mse: 3880.9929 - mae: 40.8518
Epoch 3/20
34699/34699 [==============================] - 7s 197us/sample - loss: 38.1572 - mse: 3535.9573 - mae: 38.1571
Epoch 4/20
34699/34699 [==============================] - 7s 207us/sample - loss: 36.9534 - mse: 3386.5479 - mae: 36.9534
Epoch 5/20
34699/34699 [==============================] - 7s 194us/sample - loss: 36.1111 - mse: 3287.3606 - mae: 36.1111
Epoch 6/20
34699/34699 [==============================] - 7s 196us/sample - loss: 35.4299 - mse: 3210.5422 - mae: 35.4299
Epoch 7/20
34699/34699 [==============================] - 7s 199us/sample - loss: 34.9252 - mse: 3159.7725 - mae: 34.9252
Epoch 8/20
34699/34699 [==============================] - 7s 202us/sample - loss: 34.4660 - mse: 3102.1282 - mae: 34.4660
Epoch 9/20
34699/34699 [==============================] - 7s 204us/sample - loss: 33.9536 - mse: 3044.1445 - mae: 33.9536
Epoch 10/20
34699/34699 [==============================] - 7s 203us/sample - loss: 33.6187 - mse: 3009.4475 - mae: 33.6187
Epoch 11/20
34699/34699 [==============================] - 7s 205us/sample - loss: 33.1836 - mse: 2955.1345 - mae: 33.1835
Epoch 12/20
34699/34699 [==============================] - 7s 203us/sample - loss: 32.8078 - mse: 2910.6738 - mae: 32.8078
Epoch 13/20
34699/34699 [==============================] - 7s 200us/sample - loss: 32.4859 - mse: 2867.5222 - mae: 32.4860
Epoch 14/20
34699/34699 [==============================] - 7s 194us/sample - loss: 32.1952 - mse: 2826.7388 - mae: 32.1952
Epoch 15/20
34699/34699 [==============================] - 7s 196us/sample - loss: 31.7864 - mse: 2783.0283 - mae: 31.7864
Epoch 16/20
34699/34699 [==============================] - 7s 196us/sample - loss: 31.4701 - mse: 2743.9824 - mae: 31.4700
Epoch 17/20
34699/34699 [==============================] - 7s 195us/sample - loss: 31.1860 - mse: 2707.5129 - mae: 31.1860
Epoch 18/20
34699/34699 [==============================] - 7s 194us/sample - loss: 30.8736 - mse: 2669.2893 - mae: 30.8735
Epoch 19/20
34699/34699 [==============================] - 7s 196us/sample - loss: 30.5661 - mse: 2630.2693 - mae: 30.5661
Epoch 20/20
34699/34699 [==============================] - 7s 194us/sample - loss: 30.2766 - mse: 2597.5564 - mae: 30.2766
train all mae: 30.693071365356445
```

```
train all mae: [illegible]
test      mae: 36.721683502197266
```

```
# 2
train_features = np.arange(14) # 14: without snow and rain

# X_train_w,r_train_w = create_dataset_windowed(train_scaled[:,train_features], train_labels, window_size=CNN_WINDOW_SIZE)
# X_val_w,r_val_w = create_dataset_windowed(val_scaled[:,train_features], val_labels, window_size=CNN_WINDOW_SIZE)
X_train_all_w,r_train_all_w = create_dataset_windowed(train_all_scaled[:,train_features],train_all_labels, window_size=CNN_WINDOW_SIZ
X_test_w,r_test_w = create_dataset_windowed(test_scaled[:,train_features],test_labels, window_size=CNN_WINDOW_SIZE)

model = conv_model(CNN_WINDOW_SIZE=CNN_WINDOW_SIZE, learning_rate=learning_rate, hidden=[32,64,128,256], n_features=len(train_feature
# model.summary()

# cp_path = '/content/gdrive/My Drive/Colab Notebooks/DL2020/GA3/checkpoints/step_4/cp.ckpt'
# cp_callback = tf.keras.callbacks.ModelCheckpoint(cp_path,
#                                                    monitor='val_mae',
#                                                    verbose=1,
#                                                    save_best_only=True,
#                                                    save_weights_only=True)

# history = model.fit(X_train_w, r_train_w,
#                      shuffle=True,
#                      batch_size=batch_size,
#                      epochs=epochs,
#                      validation_data=(X_val_w,r_val_w),
#                      callbacks=[cp_callback])
# model.load_weights(cp_path)

history = model.fit(X_train_all_w, r_train_all_w, shuffle=True, batch_size=batch_size, epochs=epochs)

# y_train_2 = model.predict(X_train_w)
# y_val_2 = model.predict(X_val_w)
y_train_all_2 = model.predict(X_train_all_w)
y_test_2 = model.predict(X_test_w)

# mae_train = mean_absolute_error(r_train_w,y_train_2)
# mae_val = mean_absolute_error(r_val_w,y_val_2)
```

```
# mae_val = mean_absolute_error(r_val_w,y_val_2)
mae_train_all = mean_absolute_error(r_train_all_w,y_train_all_2)
mae_test = mean_absolute_error(r_test_w,y_test_2)

# print(f"train      mae: {mae_train}")
# print(f"validation mae: {mae_val}")
print(f"train all mae: {mae_train_all}")
print(f"test      mae: {mae_test}")
```

```
Train on 34699 samples
Epoch 1/20
34699/34699 [==============================] - 7s 212us/sample - loss: 52.8202 - mse: 6273.0283 - mae: 52.8202
Epoch 2/20
34699/34699 [==============================] - 7s 197us/sample - loss: 40.0717 - mse: 3761.2554 - mae: 40.0717
Epoch 3/20
34699/34699 [==============================] - 7s 199us/sample - loss: 37.8851 - mse: 3519.9519 - mae: 37.8851
Epoch 4/20
34699/34699 [==============================] - 7s 196us/sample - loss: 36.8386 - mse: 3413.6392 - mae: 36.8386
Epoch 5/20
34699/34699 [==============================] - 7s 194us/sample - loss: 36.1003 - mse: 3347.7534 - mae: 36.1004
Epoch 6/20
34699/34699 [==============================] - 7s 193us/sample - loss: 35.4671 - mse: 3277.6780 - mae: 35.4671
Epoch 7/20
34699/34699 [==============================] - 7s 195us/sample - loss: 34.9650 - mse: 3222.1902 - mae: 34.9650
Epoch 8/20
34699/34699 [==============================] - 7s 197us/sample - loss: 34.4782 - mse: 3171.0457 - mae: 34.4783
Epoch 9/20
34699/34699 [==============================] - 7s 197us/sample - loss: 34.0872 - mse: 3125.9915 - mae: 34.0872
Epoch 10/20
34699/34699 [==============================] - 7s 194us/sample - loss: 33.6866 - mse: 3076.1357 - mae: 33.6866
Epoch 11/20
34699/34699 [==============================] - 7s 194us/sample - loss: 33.3307 - mse: 3041.9434 - mae: 33.3308
Epoch 12/20
34699/34699 [==============================] - 7s 195us/sample - loss: 32.9633 - mse: 3003.2407 - mae: 32.9634
Epoch 13/20
34699/34699 [==============================] - 7s 203us/sample - loss: 32.5760 - mse: 2952.4656 - mae: 32.5761
Epoch 14/20
34699/34699 [==============================] - 7s 195us/sample - loss: 32.2687 - mse: 2924.3916 - mae: 32.2687
Epoch 15/20
34699/34699 [==============================] - 7s 196us/sample - loss: 31.9347 - mse: 2884.2375 - mae: 31.9346
Epoch 16/20
34699/34699 [==============================] - 7s 194us/sample - loss: 31.6328 - mse: 2843.5540 - mae: 31.6328
Epoch 17/20
34699/34699 [==============================] - 7s 194us/sample - loss: 31.3857 - mse: 2811.0469 - mae: 31.3857
Epoch 18/20
34699/34699 [==============================] - 7s 195us/sample - loss: 31.0480 - mse: 2773.1462 - mae: 31.0479
Epoch 19/20
34699/34699 [==============================] - 7s 192us/sample - loss: 30.7533 - mse: 2741.5356 - mae: 30.7533
Epoch 20/20
34699/34699 [==============================] - 7s 193us/sample - loss: 30.4540 - mse: 2708.9795 - mae: 30.4540
train all mae: 30.0151309967041
```

```
train all mae: 36.0151236987841
test      mae: 36.05653381347656
```

```
# 3
train_features = np.arange(13) # 13: also leave pressure out

# X_train_w,r_train_w = create_dataset_windowed(train_scaled[:,train_features], train_labels, window_size=CNN_WINDOW_SIZE)
# X_val_w,r_val_w = create_dataset_windowed(val_scaled[:,train_features], val_labels, window_size=CNN_WINDOW_SIZE)
X_train_all_w,r_train_all_w = create_dataset_windowed(train_all_scaled[:,train_features],train_all_labels, window_size=CNN_WINDOW_SIZ
X_test_w,r_test_w = create_dataset_windowed(test_scaled[:,train_features],test_labels, window_size=CNN_WINDOW_SIZE)

model = conv_model(CNN_WINDOW_SIZE=CNN_WINDOW_SIZE, learning_rate=learning_rate, hidden=[32,64,128,256], n_features=len(train_feature
# model.summary()

# cp_path = '/content/gdrive/My Drive/Colab Notebooks/DL2020/GA3/checkpoints/step_4/cp.ckpt'
# cp_callback = tf.keras.callbacks.ModelCheckpoint(cp_path,
#                                                    monitor='val_mae',
#                                                    verbose=1,
#                                                    save_best_only=True,
#                                                    save_weights_only=True)

# history = model.fit(X_train_w, r_train_w,
#                       shuffle=True,
#                       batch_size=batch_size,
#                       epochs=epochs,
#                       validation_data=(X_val_w,r_val_w),
#                       callbacks=[cp_callback])
# model.load_weights(cp_path)

history = model.fit(X_train_all_w, r_train_all_w, shuffle=True, batch_size=batch_size, epochs=epochs)

# y_train_3 = model.predict(X_train_w)
# y_val_3 = model.predict(X_val_w)
y_train_all_3 = model.predict(X_train_all_w)
y_test_3 = model.predict(X_test_w)

# mae_train = mean_absolute_error(r_train_w,y_train_3)
# mae_val = mean_absolute_error(r_val_w,y_val_3)
```

```
# mae_val = mean_absolute_error(r_val_w,y_val_3)
mae_train_all = mean_absolute_error(r_train_all_w,y_train_all_3)
mae_test = mean_absolute_error(r_test_w,y_test_3)

# print(f"train      mae: {mae_train}")
# print(f"validation mae: {mae_val}")
print(f"train all mae: {mae_train_all}")
print(f"test      mae: {mae_test}")
```

```
Train on 34699 samples
Epoch 1/20
34699/34699 [==============================] - 7s 210us/sample - loss: 54.0307 - mse: 6617.4917 - mae: 54.0307
Epoch 2/20
34699/34699 [==============================] - 7s 190us/sample - loss: 40.2426 - mse: 3807.7834 - mae: 40.2426
Epoch 3/20
34699/34699 [==============================] - 7s 191us/sample - loss: 37.9463 - mse: 3515.8484 - mae: 37.9462
Epoch 4/20
34699/34699 [==============================] - 7s 191us/sample - loss: 36.8989 - mse: 3408.2424 - mae: 36.8989
Epoch 5/20
34699/34699 [==============================] - 7s 191us/sample - loss: 36.1279 - mse: 3325.7302 - mae: 36.1279
Epoch 6/20
34699/34699 [==============================] - 7s 193us/sample - loss: 35.6114 - mse: 3265.9702 - mae: 35.6114
Epoch 7/20
34699/34699 [==============================] - 7s 196us/sample - loss: 35.1045 - mse: 3212.6860 - mae: 35.1044
Epoch 8/20
34699/34699 [==============================] - 7s 191us/sample - loss: 34.7127 - mse: 3173.8320 - mae: 34.7126
Epoch 9/20
34699/34699 [==============================] - 7s 190us/sample - loss: 34.2677 - mse: 3125.3076 - mae: 34.2677
Epoch 10/20
34699/34699 [==============================] - 7s 189us/sample - loss: 33.9107 - mse: 3088.5574 - mae: 33.9106
Epoch 11/20
34699/34699 [==============================] - 7s 190us/sample - loss: 33.5652 - mse: 3050.0444 - mae: 33.5652
Epoch 12/20
34699/34699 [==============================] - 7s 191us/sample - loss: 33.1941 - mse: 3000.4568 - mae: 33.1941
Epoch 13/20
34699/34699 [==============================] - 7s 193us/sample - loss: 32.8671 - mse: 2969.6519 - mae: 32.8671
Epoch 14/20
34699/34699 [==============================] - 7s 192us/sample - loss: 32.5818 - mse: 2933.4343 - mae: 32.5818
Epoch 15/20
34699/34699 [==============================] - 7s 192us/sample - loss: 32.2261 - mse: 2899.2178 - mae: 32.2261
Epoch 16/20
34699/34699 [==============================] - 7s 193us/sample - loss: 31.9384 - mse: 2861.4075 - mae: 31.9384
Epoch 17/20
34699/34699 [==============================] - 7s 192us/sample - loss: 31.6317 - mse: 2831.4285 - mae: 31.6318
Epoch 18/20
34699/34699 [==============================] - 7s 192us/sample - loss: 31.3363 - mse: 2790.1589 - mae: 31.3363
Epoch 19/20
34699/34699 [==============================] - 7s 193us/sample - loss: 31.0215 - mse: 2754.9412 - mae: 31.0215
Epoch 20/20
34699/34699 [==============================] - 7s 191us/sample - loss: 30.7538 - mse: 2717.4949 - mae: 30.7538
train all mae: 30.32161521911621
```

```
test      mae: 35.549591064453125
```

```
# 4
train_features = np.arange(12) # 12: also leave temperature out

# X_train_w,r_train_w = create_dataset_windowed(train_scaled[:,train_features], train_labels, window_size=CNN_WINDOW_SIZE)
# X_val_w,r_val_w = create_dataset_windowed(val_scaled[:,train_features], val_labels, window_size=CNN_WINDOW_SIZE)
X_train_all_w,r_train_all_w = create_dataset_windowed(train_all_scaled[:,train_features],train_all_labels, window_size=CNN_WINDOW_SIZ
X_test_w,r_test_w = create_dataset_windowed(test_scaled[:,train_features],test_labels, window_size=CNN_WINDOW_SIZE)

model = conv_model(CNN_WINDOW_SIZE=CNN_WINDOW_SIZE, learning_rate=learning_rate, hidden=[32,64,128,256], n_features=len(train_feature
# model.summary()

# cp_path = '/content/gdrive/My Drive/Colab Notebooks/DL2020/GA3/checkpoints/step_4/cp.ckpt'
# cp_callback = tf.keras.callbacks.ModelCheckpoint(cp_path,
#                                                   monitor='val_mae',
#                                                   verbose=1,
#                                                   save_best_only=True,
#                                                   save_weights_only=True)

# history = model.fit(X_train_w, r_train_w,
#                     shuffle=True,
#                     batch_size=batch_size,
#                     epochs=epochs,
#                     validation_data=(X_val_w,r_val_w),
#                     callbacks=[cp_callback])
# model.load_weights(cp_path)

history = model.fit(X_train_all_w, r_train_all_w, shuffle=True, batch_size=batch_size, epochs=epochs)

# y_train_4 = model.predict(X_train_w)
# y_val_4 = model.predict(X_val_w)
y_train_all_4 = model.predict(X_train_all_w)
y_test_4 = model.predict(X_test_w)

# mae_train = mean_absolute_error(r_train_w,y_train_4)
# mae_val = mean_absolute_error(r_val_w,y_val_4)
```

```
# mae_val = mean_absolute_error(r_val_w,y_val_4)
mae_train_all = mean_absolute_error(r_train_all_w,y_train_all_4)
mae_test = mean_absolute_error(r_test_w,y_test_4)

# print(f"train      mae: {mae_train}")
# print(f"validation mae: {mae_val}")
print(f"train all mae: {mae_train_all}")
print(f"test      mae: {mae_test}")
```

```
Train on 34699 samples
Epoch 1/20
34699/34699 [==============================] - 7s 207us/sample - loss: 52.7946 - mse: 6387.0820 - mae: 52.7945
Epoch 2/20
34699/34699 [==============================] - 7s 190us/sample - loss: 40.1069 - mse: 3812.5237 - mae: 40.1069
Epoch 3/20
34699/34699 [==============================] - 7s 190us/sample - loss: 37.9810 - mse: 3552.3857 - mae: 37.9810
Epoch 4/20
34699/34699 [==============================] - 7s 191us/sample - loss: 37.0116 - mse: 3439.1738 - mae: 37.0116
Epoch 5/20
34699/34699 [==============================] - 7s 189us/sample - loss: 36.3666 - mse: 3364.1562 - mae: 36.3666
Epoch 6/20
34699/34699 [==============================] - 7s 190us/sample - loss: 35.8381 - mse: 3306.1174 - mae: 35.8381
Epoch 7/20
34699/34699 [==============================] - 7s 194us/sample - loss: 35.4023 - mse: 3251.7136 - mae: 35.4023
Epoch 8/20
34699/34699 [==============================] - 7s 190us/sample - loss: 34.9676 - mse: 3201.4102 - mae: 34.9676
Epoch 9/20
34699/34699 [==============================] - 7s 194us/sample - loss: 34.6092 - mse: 3158.4285 - mae: 34.6092
Epoch 10/20
34699/34699 [==============================] - 7s 198us/sample - loss: 34.2632 - mse: 3128.3069 - mae: 34.2632
Epoch 11/20
34699/34699 [==============================] - 7s 192us/sample - loss: 33.9271 - mse: 3084.5957 - mae: 33.9271
Epoch 12/20
34699/34699 [==============================] - 7s 191us/sample - loss: 33.5960 - mse: 3039.9888 - mae: 33.5959
Epoch 13/20
34699/34699 [==============================] - 7s 191us/sample - loss: 33.2688 - mse: 3004.7747 - mae: 33.2688
Epoch 14/20
34699/34699 [==============================] - 6s 186us/sample - loss: 32.9851 - mse: 2965.5227 - mae: 32.9851
Epoch 15/20
34699/34699 [==============================] - 7s 190us/sample - loss: 32.6499 - mse: 2937.2498 - mae: 32.6499
Epoch 16/20
34699/34699 [==============================] - 7s 192us/sample - loss: 32.3263 - mse: 2891.3926 - mae: 32.3264
Epoch 17/20
34699/34699 [==============================] - 7s 188us/sample - loss: 31.9796 - mse: 2857.7935 - mae: 31.9796
Epoch 18/20
34699/34699 [==============================] - 7s 197us/sample - loss: 31.7465 - mse: 2819.9917 - mae: 31.7465
Epoch 19/20
34699/34699 [==============================] - 7s 196us/sample - loss: 31.4289 - mse: 2783.5835 - mae: 31.4289
Epoch 20/20
34699/34699 [==============================] - 7s 188us/sample - loss: 31.1717 - mse: 2758.3677 - mae: 31.1717
train all mae: 30.537208557128906
```

```
test      mae: 36.466339111328125
```

```
# 5
train_features = [0,] # only pollution

# X_train_w,r_train_w = create_dataset_windowed(train_scaled[:,train_features], train_labels, window_size=CNN_WINDOW_SIZE)
# X_val_w,r_val_w = create_dataset_windowed(val_scaled[:,train_features], val_labels, window_size=CNN_WINDOW_SIZE)
X_train_all_w,r_train_all_w = create_dataset_windowed(train_all_scaled[:,train_features],train_all_labels, window_size=CNN_WINDOW_SIZ
X_test_w,r_test_w = create_dataset_windowed(test_scaled[:,train_features],test_labels, window_size=CNN_WINDOW_SIZE)

model = conv_model(CNN_WINDOW_SIZE=CNN_WINDOW_SIZE, learning_rate=learning_rate, hidden=[32,64,128,256], n_features=len(train_feature
# model.summary()

# cp_path = '/content/gdrive/My Drive/Colab Notebooks/DL2020/GA3/checkpoints/step_4/cp.ckpt'
# cp_callback = tf.keras.callbacks.ModelCheckpoint(cp_path,
#                                                   monitor='val_mae',
#                                                   verbose=1,
#                                                   save_best_only=True,
#                                                   save_weights_only=True)

# history = model.fit(X_train_w, r_train_w,
#                     shuffle=True,
#                     batch_size=batch_size,
#                     epochs=epochs,
#                     validation_data=(X_val_w,r_val_w),
#                     callbacks=[cp_callback])
# model.load_weights(cp_path)

history = model.fit(X_train_all_w, r_train_all_w, shuffle=True, batch_size=batch_size, epochs=epochs)

# y_train_5 = model.predict(X_train_w)
# y_val_5 = model.predict(X_val_w)
y_train_all_5 = model.predict(X_train_all_w)
y_test_5 = model.predict(X_test_w)

# mae_train = mean_absolute_error(r_train_w,y_train_5)
# mae_val = mean_absolute_error(r_val_w,y_val_5)
```

```
# mae_val = mean_absolute_error(r_val_w,y_val_5)
mae_train_all = mean_absolute_error(r_train_all_w,y_train_all_5)
mae_test = mean_absolute_error(r_test_w,y_test_5)

# print(f"train      mae: {mae_train}")
# print(f"validation mae: {mae_val}")
print(f"train all mae: {mae_train_all}")
print(f"test      mae: {mae_test}")
```

```
Train on 34699 samples
Epoch 1/20
34699/34699 [==============================] - 7s 190us/sample - loss: 50.5201 - mse: 5662.3936 - mae: 50.5200
Epoch 2/20
34699/34699 [==============================] - 6s 172us/sample - loss: 41.0793 - mse: 3996.4307 - mae: 41.0793
Epoch 3/20
34699/34699 [==============================] - 6s 170us/sample - loss: 39.6769 - mse: 3839.4705 - mae: 39.6769
Epoch 4/20
34699/34699 [==============================] - 6s 173us/sample - loss: 38.8301 - mse: 3750.6023 - mae: 38.8301
Epoch 5/20
34699/34699 [==============================] - 6s 172us/sample - loss: 38.3320 - mse: 3694.9348 - mae: 38.3320
Epoch 6/20
34699/34699 [==============================] - 6s 169us/sample - loss: 38.0554 - mse: 3672.5378 - mae: 38.0554
Epoch 7/20
34699/34699 [==============================] - 6s 172us/sample - loss: 37.7645 - mse: 3641.6311 - mae: 37.7645
Epoch 8/20
34699/34699 [==============================] - 6s 170us/sample - loss: 37.5976 - mse: 3621.0688 - mae: 37.5977
Epoch 9/20
34699/34699 [==============================] - 6s 170us/sample - loss: 37.4052 - mse: 3604.5259 - mae: 37.4052
Epoch 10/20
34699/34699 [==============================] - 6s 172us/sample - loss: 37.3063 - mse: 3583.9077 - mae: 37.3063
Epoch 11/20
34699/34699 [==============================] - 6s 171us/sample - loss: 37.0894 - mse: 3572.0010 - mae: 37.0894
Epoch 12/20
34699/34699 [==============================] - 6s 175us/sample - loss: 36.9698 - mse: 3555.3555 - mae: 36.9698
Epoch 13/20
34699/34699 [==============================] - 6s 176us/sample - loss: 36.8417 - mse: 3530.8772 - mae: 36.8417
Epoch 14/20
34699/34699 [==============================] - 6s 175us/sample - loss: 36.7225 - mse: 3527.4028 - mae: 36.7225
Epoch 15/20
34699/34699 [==============================] - 6s 176us/sample - loss: 36.5896 - mse: 3510.2017 - mae: 36.5896
Epoch 16/20
34699/34699 [==============================] - 6s 173us/sample - loss: 36.4849 - mse: 3485.8623 - mae: 36.4849
Epoch 17/20
34699/34699 [==============================] - 6s 173us/sample - loss: 36.3171 - mse: 3470.1851 - mae: 36.3171
Epoch 18/20
34699/34699 [==============================] - 6s 173us/sample - loss: 36.2053 - mse: 3465.5117 - mae: 36.2053
Epoch 19/20
34699/34699 [==============================] - 6s 177us/sample - loss: 36.1076 - mse: 3448.4753 - mae: 36.1077
Epoch 20/20
34699/34699 [==============================] - 6s 179us/sample - loss: 36.0124 - mse: 3434.7559 - mae: 36.0124
train all mae: 35.67976379394531
```

```
test      mae: 37.383750915527344
```

Average

↳ 1 cell hidden

Weighted average

```
# create new dataset
# X_train_t,r_train_t = create_dataset_top((y_train_1, y_train_2, y_train_3, y_train_4, y_train_5), train_labels, window_size=CNN_WIN
# X_val_t,r_val_t = create_dataset_top((y_val_1, y_val_2, y_val_3, y_val_4, y_val_5), val_labels, window_size=CNN_WINDOW_SIZE)
X_train_all_t,r_train_all_t = create_dataset_top((y_train_all_1, y_train_all_2, y_train_all_3, y_train_all_4, y_train_all_5), train_a
X_test_t,r_test_t = create_dataset_top((y_test_1, y_test_2, y_test_3, y_test_4, y_test_5), test_labels, window_size=CNN_WINDOW_SIZE)

def top_model(n_models=5, learning_rate=0.01):
  model = Sequential(Dense(1, activation='linear', kernel_initializer=tf.constant_initializer(value=1/n_models), kernel_constraint=tf

  optim = tf.keras.optimizers.Adam(learning_rate=learning_rate)
  model.compile(loss='mae',
                optimizer=optim,
                # keep extra metrics: mse and mae without regularisation terms
                metrics=['mse', 'mae'])

  return model
```

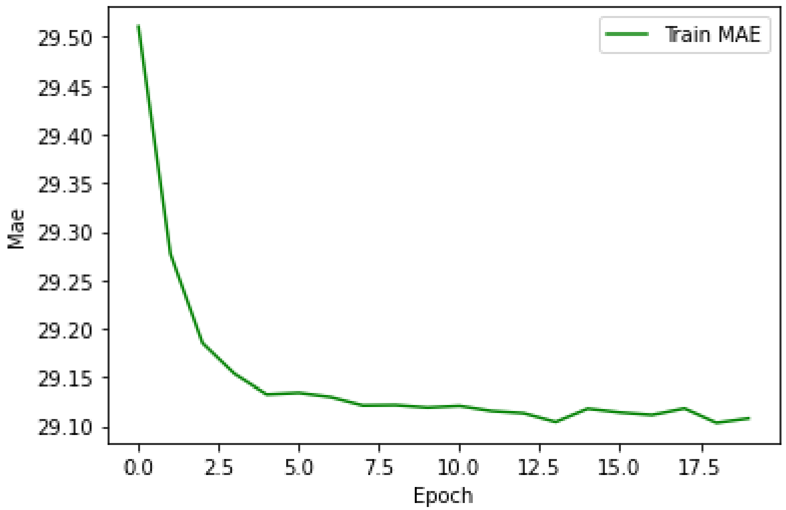
```
batch_size = 32
epochs = 20
learning_rate = 0.001

model = top_model(n_models=5, learning_rate=learning_rate)
model.summary()

# cp_path = '/content/gdrive/My Drive/Colab Notebooks/DL2020/GA3/checkpoints/step_4/cp.ckpt'
```

```
# cp_callback = tf.keras.callbacks.ModelCheckpoint(cp_path,
#                                                   monitor='val_mae',
#                                                   verbose=1,
#                                                   save_best_only=True,
#                                                   save_weights_only=True)

# history = model.fit(X_train_t, r_train_t,
#                     shuffle=True,
#                     batch_size=batch_size,
#                     epochs=epochs,
#                     validation_data=(X_val_t,r_val_t),
#                     callbacks=[cp_callback])
# model.load_weights(cp_path)

history = model.fit(X_train_all_t, r_train_all_t, shuffle=True, batch_size=batch_size, epochs=epochs)
```

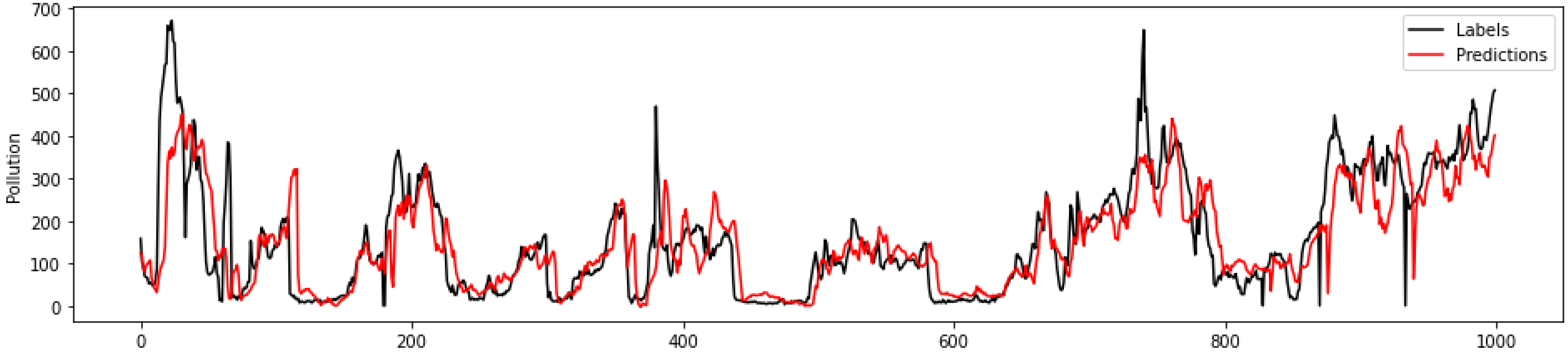
```
Model: "sequential_9"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_13 (Dense)             (None, 1)                 5
=================================================================
Total params: 5
Trainable params: 5
Non-trainable params: 0
_________________________________________________________________
Train on 34699 samples
Epoch 1/20
34699/34699 [==============================] - 3s 89us/sample - loss: 29.5108 - mse: 2583.5007 - mae: 29.5108
Epoch 2/20
34699/34699 [==============================] - 3s 85us/sample - loss: 29.2769 - mse: 2557.5396 - mae: 29.2769
Epoch 3/20
34699/34699 [==============================] - 3s 82us/sample - loss: 29.1851 - mse: 2542.3557 - mae: 29.1851
Epoch 4/20
34699/34699 [==============================] - 3s 83us/sample - loss: 29.1535 - mse: 2533.5254 - mae: 29.1535
Epoch 5/20
34699/34699 [==============================] - 3s 83us/sample - loss: 29.1325 - mse: 2534.1399 - mae: 29.1325
Epoch 6/20
34699/34699 [==============================] - 3s 83us/sample - loss: 29.1342 - mse: 2531.3281 - mae: 29.1342
Epoch 7/20
34699/34699 [==============================] - 3s 84us/sample - loss: 29.1299 - mse: 2531.9302 - mae: 29.1299
Epoch 8/20
34699/34699 [==============================] - 3s 82us/sample - loss: 29.1212 - mse: 2530.6082 - mae: 29.1212
Epoch 9/20
34699/34699 [==============================] - 3s 82us/sample - loss: 29.1216 - mse: 2528.7749 - mae: 29.1217
Epoch 10/20
34699/34699 [==============================] - 3s 83us/sample - loss: 29.1191 - mse: 2528.1057 - mae: 29.1191
Epoch 11/20
34699/34699 [==============================] - 3s 83us/sample - loss: 29.1208 - mse: 2528.4241 - mae: 29.1208
Epoch 12/20
34699/34699 [==============================] - 3s 83us/sample - loss: 29.1156 - mse: 2528.0950 - mae: 29.1156
Epoch 13/20
34699/34699 [==============================] - 3s 82us/sample - loss: 29.1133 - mse: 2528.0969 - mae: 29.1133
Epoch 14/20
34699/34699 [==============================] - 3s 83us/sample - loss: 29.1043 - mse: 2527.8591 - mae: 29.1043
Epoch 15/20
34699/34699 [==============================] - 3s 82us/sample - loss: 29.1180 - mse: 2528.7607 - mae: 29.1180
Epoch 16/20
```

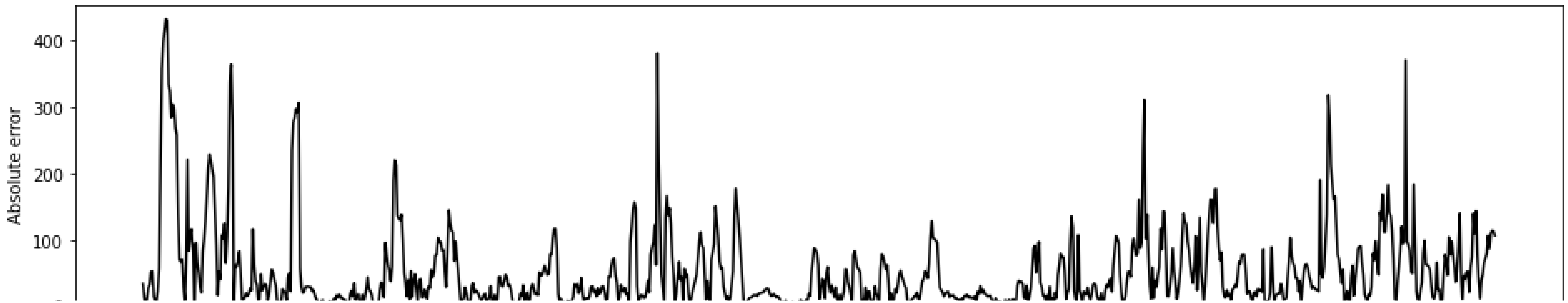
```
Epoch 16/20
34699/34699 [==============================] - 3s 81us/sample - loss: 29.1141 - mse: 2526.6499 - mae: 29.1141
Epoch 17/20
34699/34699 [==============================] - 3s 83us/sample - loss: 29.1115 - mse: 2524.7542 - mae: 29.1115
Epoch 18/20
34699/34699 [==============================] - 3s 82us/sample - loss: 29.1182 - mse: 2528.4539 - mae: 29.1183
Epoch 19/20
34699/34699 [==============================] - 3s 83us/sample - loss: 29.1034 - mse: 2525.4360 - mae: 29.1034
Epoch 20/20
34699/34699 [==============================] - 3s 83us/sample - loss: 29.1080 - mse: 2525.6501 - mae: 29.1080
```

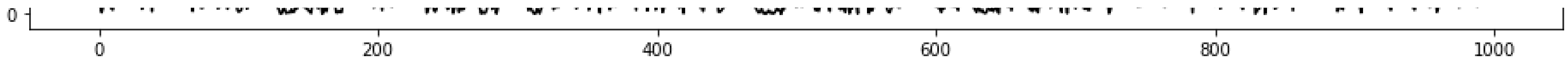
```
plot_history(history)

# y_train = model.predict(X_train_t)
# y_val = model.predict(X_val_t)
y_train_all = model.predict(X_train_all_t)
y_test = model.predict(X_test_t)

# mae_train = mean_absolute_error(r_train_t,y_train)
# mae_val = mean_absolute_error(r_val_t,y_val)
mae_train_all = mean_absolute_error(r_train_all_t,y_train_all)
mae_test = mean_absolute_error(r_test_t,y_test)

# print(f"train      mae: {mae_train}")
# print(f"validation mae: {mae_val}")
print(f"train all mae: {mae_train_all}")
print(f"test      mae: {mae_test}")

# Visualise first 1000 predictions for test
# PlotResults(r_val_t[:1000],y_val[:1000,0])
PlotResults(r_test_t[:1000],y_test[:1000,0])
```

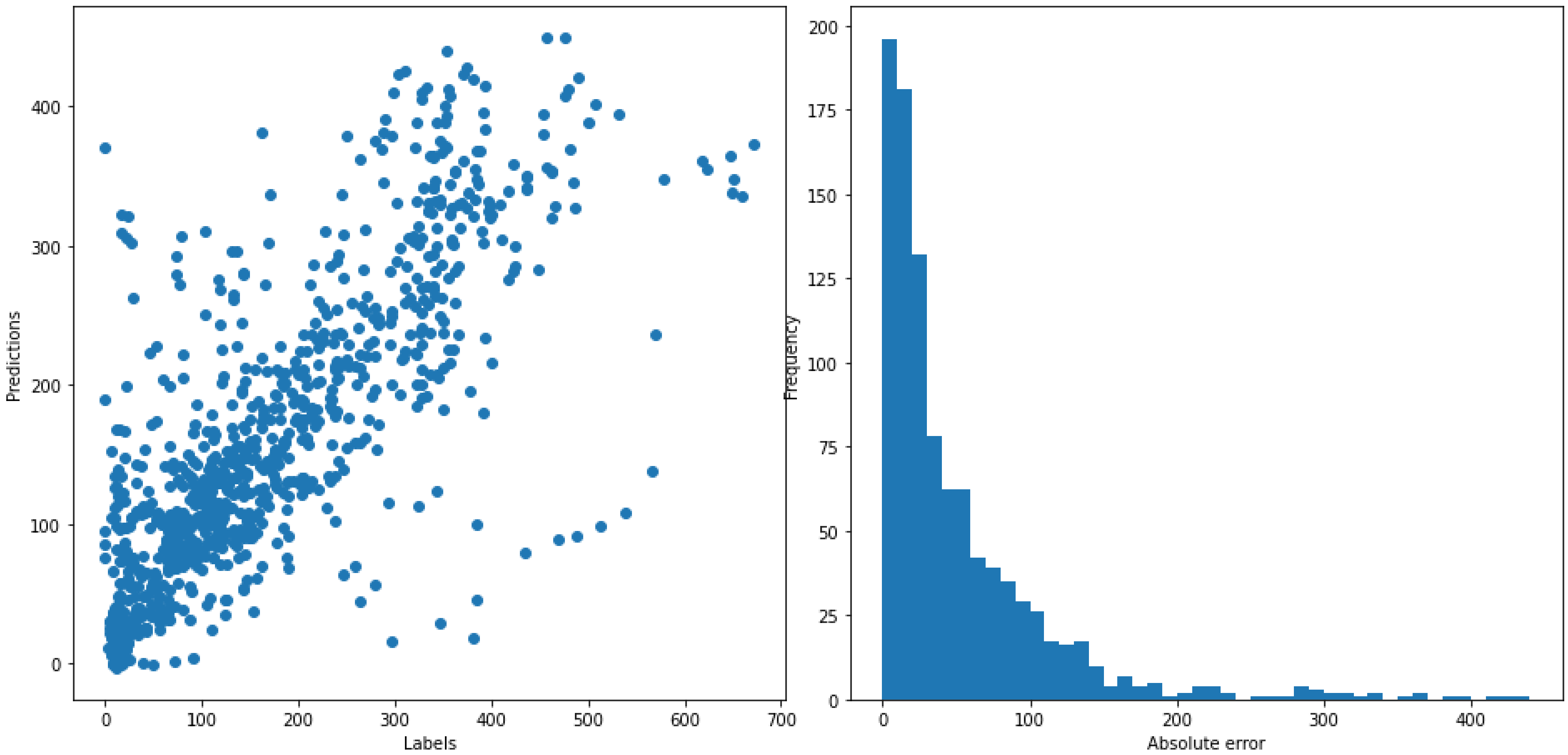



```
train all mae: 29.08261489868164
test      mae: 35.12261962890625
```










```
# get weights of each model
model.layers[0].get_weights()
```

```
[array([[ 0.31255028],
        [ 0.34277973],
        [ 0.34602535],
        [ 0.06957645],
        [-0.08152214]], dtype=float32)]
```