TRANSPORT AND
TELECOMMUNICATION
INSTITUTE

Intelligence System

Course Work

Topic: Outliers detection using LSTM NN.

Student name: Eduards

Student surname: Gruberts

St. code: 4901-2MDA

Lecturer: Kijonoka Jeļena.

RIGA

Content

Problem statement

Description of the selected problem

It is difficult to imagine our daily life without the transportation of goods or public transport, the scope of this business is based on vehicles, and In case of failure, drivers observe it post factum.

For many industries, vehicle technical condition is a critical case.

The solution of this issue may be in regular technical inspection, but this procedure is quite expensive.

In my work, I will research the possibility of detecting non-typical system operation based on aggregated telemetric data.

Justification of the choice of the method for solving the problem

Data collected from machines represent time series in which time goes along the X-axis and the observed value is Y.

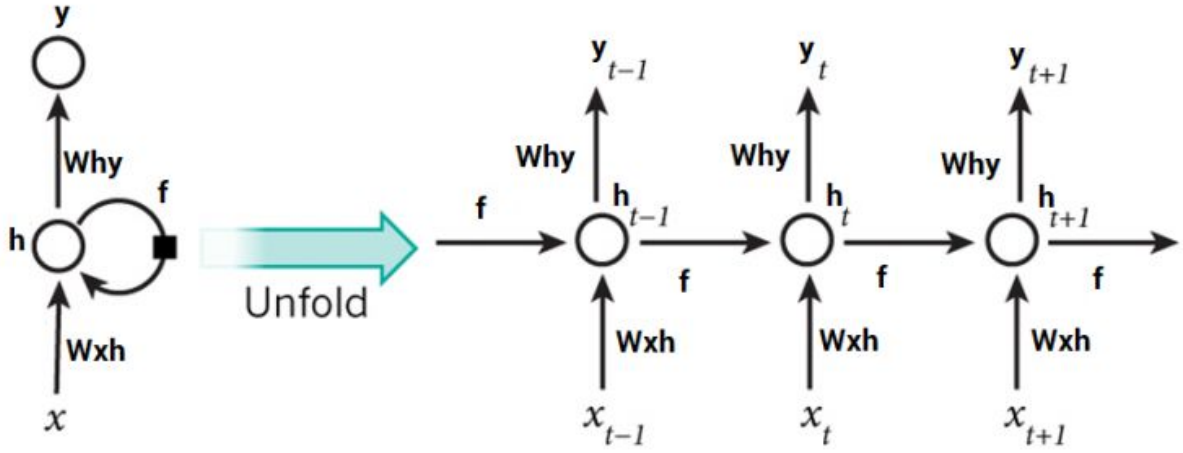
The data are interrelated and the value at the moment is closely related to the value in the previous step.

Therefore, the task of detecting anomalies is reduced to the task of forecasting and determining how much the predicted value differs from the real one.

For the forecasting problem in neural network apply method recurrent neural network.

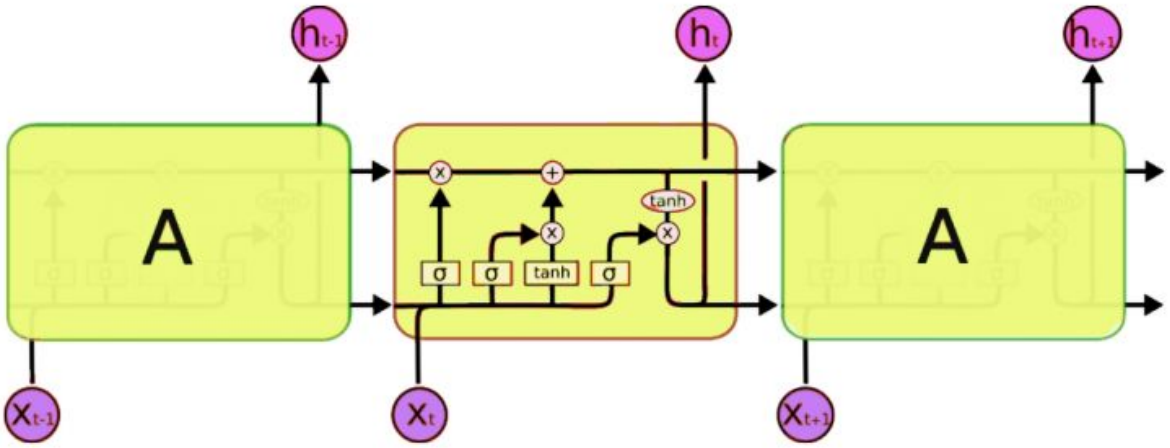
Description of the studied method or algorithm, that will be used of the problem solving

Recurrent neural networks (RNN) are a class of neural networks that is powerful for modeling sequence data such as time series or natural language. But unfortunately the RNN has some limitations, such as vanishing gradient problem.



Pic.1. RNN schema

Long Short-Term Memory (LSTM) is a variety of recurrent neural network which are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series.



Pic.2. LSTM schema

Formalization of the solution to the selected problem by the method used

The problem of determining the anomaly is reduced to predicting engine operation by the value of the parameter engine revolutions per minute (RPM)

The input for a neural network is a time series with the following ODB parameters:

- Engine RPM
- Engine Load
- Motion
- Vehicle Speed
- Intake MAP (Intake manifold absolute pressure)
- MAF air flow rate
- Throttle Position
- Coolant Temperature
- Intake Air Temperature
- Ambient Air Temperature
- Direct Fuel Rail Pressure
- Commanded EGR
- EGR Error
- Fuel Level
- Barometric Pressure
- Control Module Voltage

Supervised learning problem will be framed as predicting the engine RPM at the current hour (t) given the engine RPM measurement and ODB parameters at the prior time step.

Software implementation

```
from math import sqrt
from numpy import concatenate
from matplotlib import pyplot
from pandas import set_option
from pandas import read_csv
from pandas import DataFrame
from pandas import concat
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import mean_squared_error

from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import GRU
import tensorflow as tf
from tensorflow import keras

print(tf.__version__)



#set_option('display.max_columns', None)

# Set CPU as available physical device
#my_devices = tf.config.experimental.list_physical_devices(device_type='CPU')
#tf.config.experimental.set_visible_devices(devices= my_devices, device_type='CPU')
```

```
def series_to_supervised(data):
                    n_vars = 1 if type(data) is list else data.shape[1]

                    #df = DataFrame(data)
                    cols, names = list(), list()

                    cols.append(data.shift(1))
                names += [('var%d(t-%d)' % (j + 1, 1)) for j in range(n_vars)]

                    cols.append(data.shift(-0)[0])
                    names += ['var1(1)']

                    agg = concat(cols, axis=1)
                    agg.columns = names

                    return agg

# load dataset
df = read_csv('fm_volvo_odb_telemetry.csv', index_col=0)
df = df.interpolate(method='linear', axis=0)

# ensure all data is float
values = df.values.astype('float32')
scaler = MinMaxScaler(feature_range=(0, 1))
scaled = DataFrame(scaler.fit_transform(values))

#Transform series to supervised (just add the engine RPM value to the end)
reframed = series_to_supervised(scaled)
```

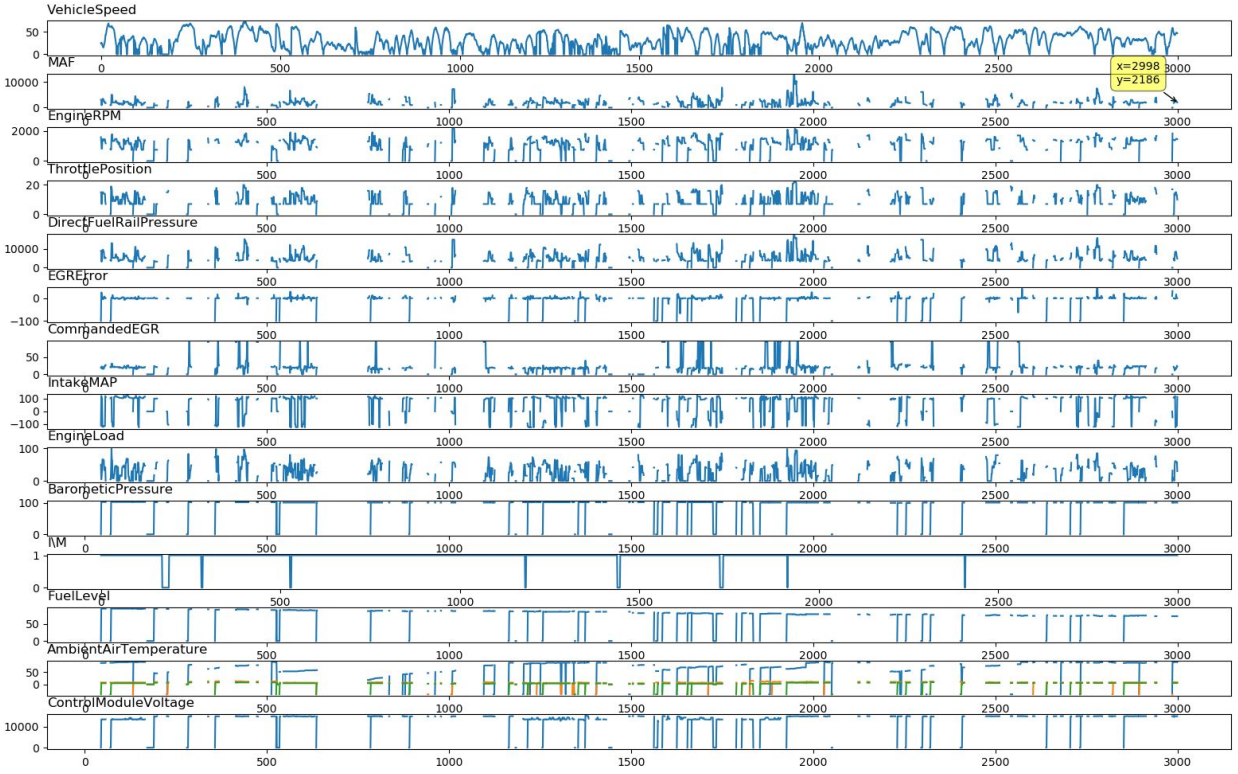
```
#Filter NaN
reframed.dropna(inplace=True)

#split dataset to train and test
test_size = 1000
test = reframed.iloc[:test_size, :]
train = reframed.iloc[test_size:, :]

#Split supervised input-output pairs
train_X, train_y = train.values[:, :-1], train.values[:, -1]
test_X, test_y = test.values[:, :-1], test.values[:, -1]

#Transform data for keras format
train_X = train_X.reshape((train_X.shape[0], 1, train_X.shape[1]))
test_X = test_X.reshape((test_X.shape[0], 1, test_X.shape[1]))

#design network
model = Sequential()
model.add(GRU(200, input_shape=(train_X.shape[1], train_X.shape[2])))
model.add(Dense(1))

model.compile(
        loss='mae',
        optimizer='adam')
```

```
#train model
history = model.fit(
	train_X,
	train_y,
	epochs=100,
	batch_size=72,
	validation_data=(test_X, test_y),
	verbose=1,
	shuffle=False)

#Make a prediction
yhat = model.predict(test_X)
test_X = test_X.reshape((test_X.shape[0], test_X.shape[2]))

# invert scaling for forecast
inv_yhat = concatenate((yhat, test_X[:, 1:]), axis=1)
inv_yhat = scaler.inverse_transform(inv_yhat)
inv_yhat = inv_yhat[:, 0]

# invert scaling for actual
test_y = test_y.reshape((len(test_y), 1))
inv_y = concatenate((test_y, test_X[:, 1:]), axis=1)
inv_y = scaler.inverse_transform(inv_y)
inv_y = inv_y[:, 0]

# calculate RMSE
rmse = sqrt(mean_squared_error(inv_y, inv_yhat))
print('Test RMSE: %.3f' % rmse)
```

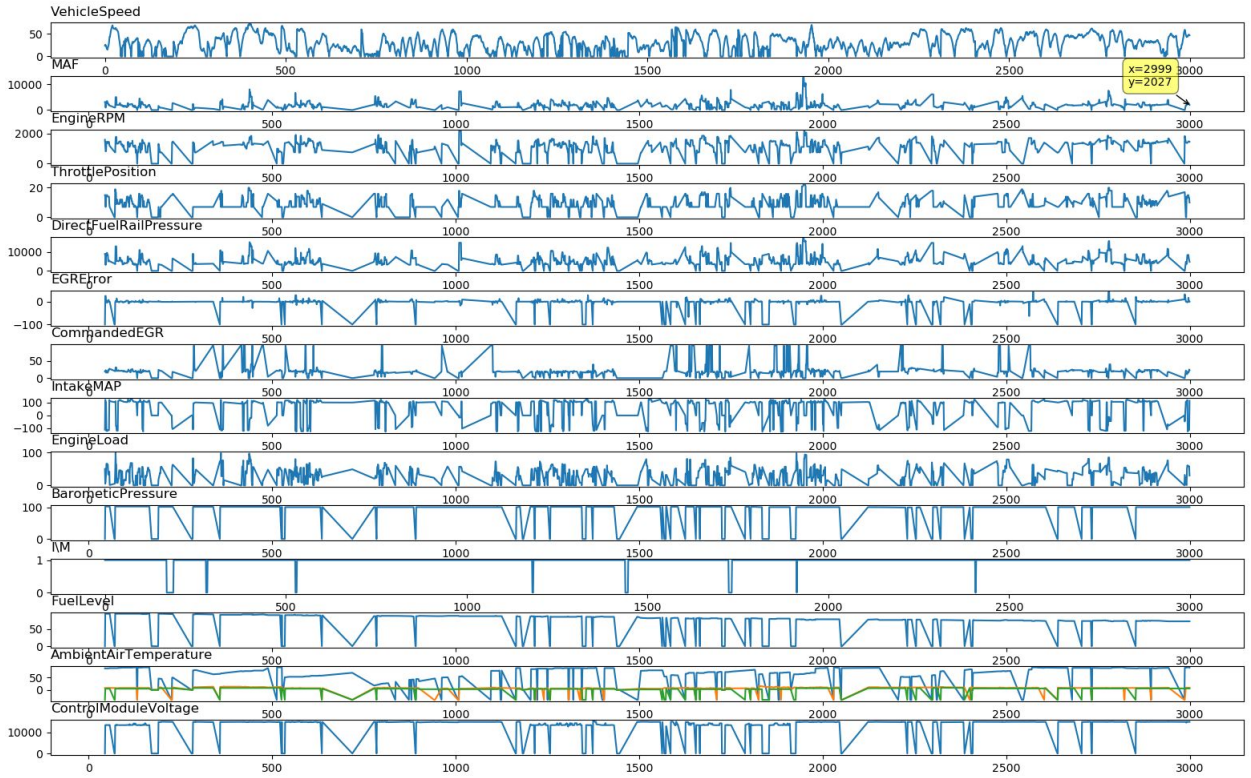
Demonstration

During data preparation a serious issue was noticed, due to some electronic collision data being seriously corrupted (Pic.3), to consolidate the data, was applied several methods, linear interpolation (Pic.4) and akim interpolation (Pic.5).

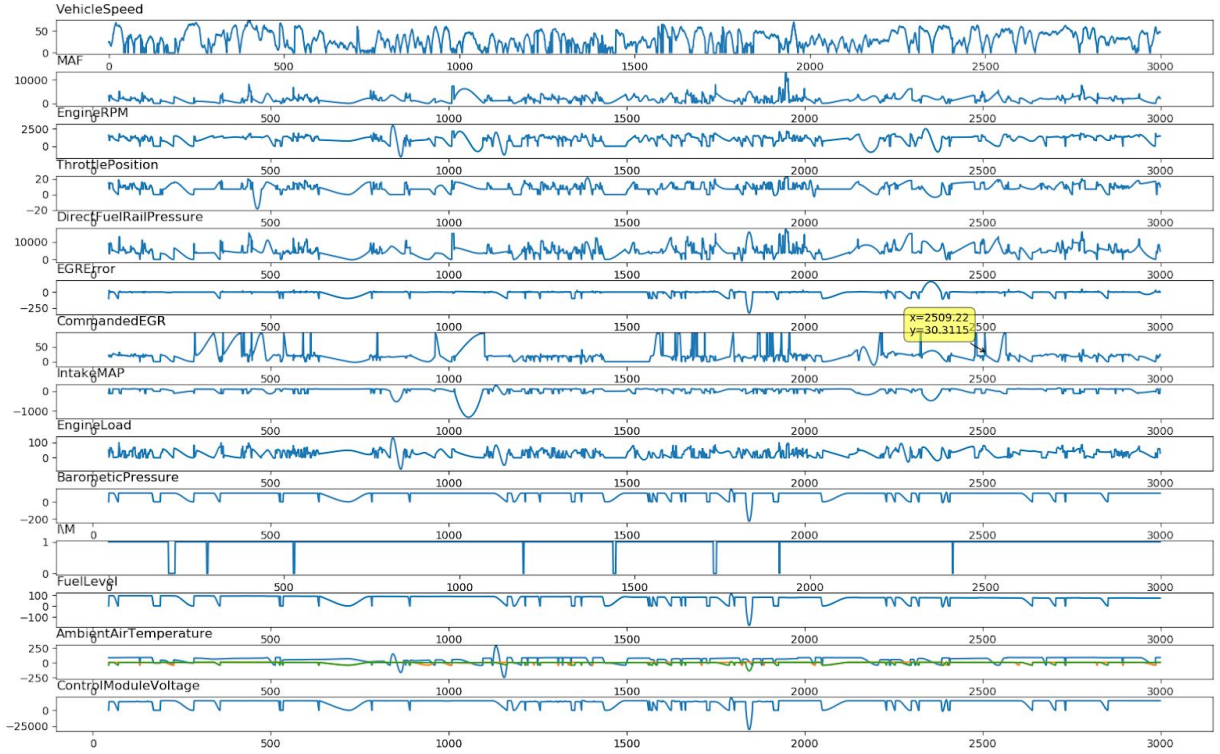
In order to check that my approach is working, I used the synthetic data from kegel



Pic.3. corrupted data



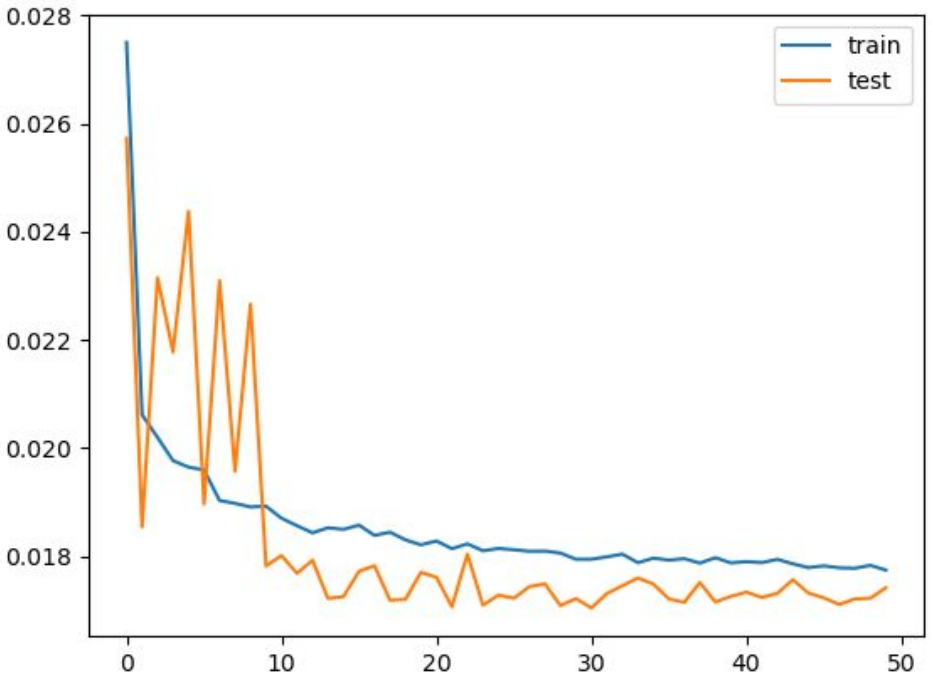
Pic.4. linear interpolation



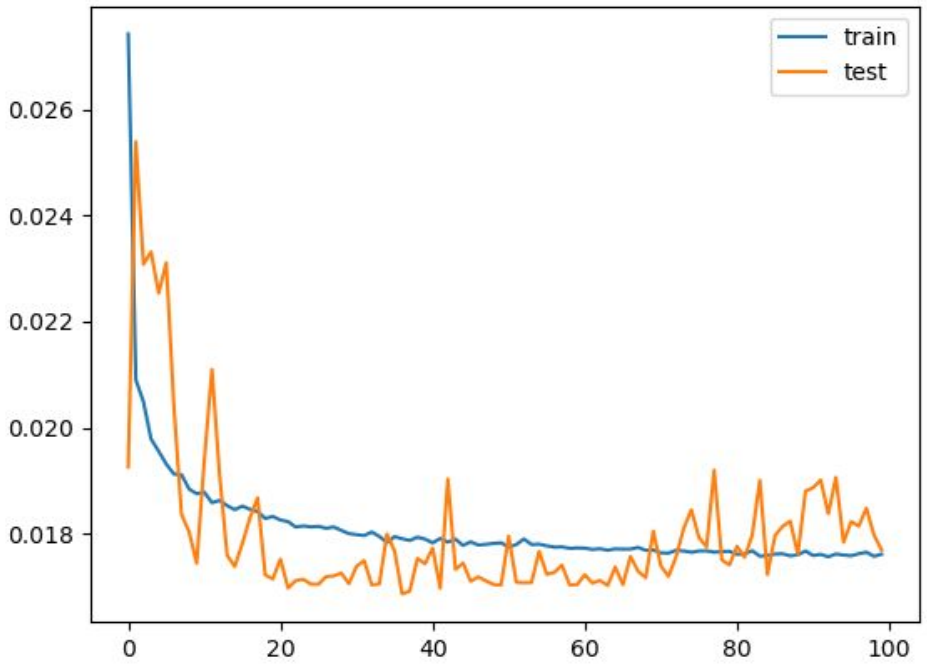
Pic.5. akima interpolation

Plan of experiments

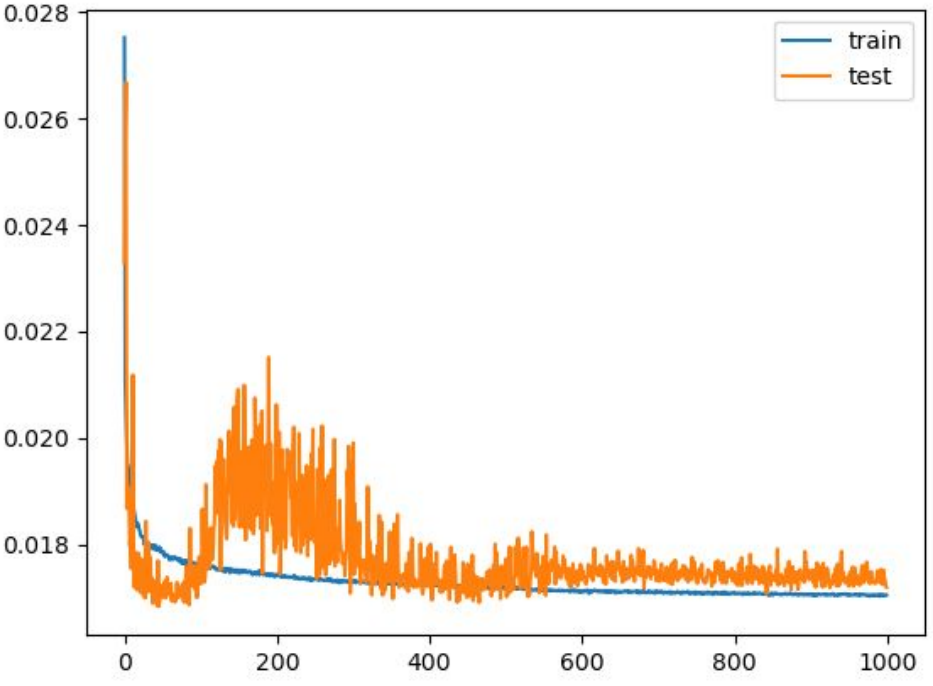
Data consolidation	Training data set	Testing data set	NN	epoch	units	loss	val loss	RMSE	Graphic
linear	333758	1000	LSTM	50	50	0.0176	0.0174	191.803	Pic.6
linear	333758	1000	LSTM	100	50	0.0175	0.0177	189.633	Pic.7
linear	333758	1000	LSTM	1000	50	0.0170	0.0172	190.158	Pic.8
linear	333758	1000	LSTM	100	200	0.0175	0.0175	189.158	Pic.9
linear	333758	1000	GRU	50	50	0.0177	0.0173		Pic.10
linear	333758	1000	GRU	100	50	0.0176	0.0177	189.757	Pic.11
linear	333758	1000	GRU	100	50	0.0170	0.0173	190.720	Pic.12



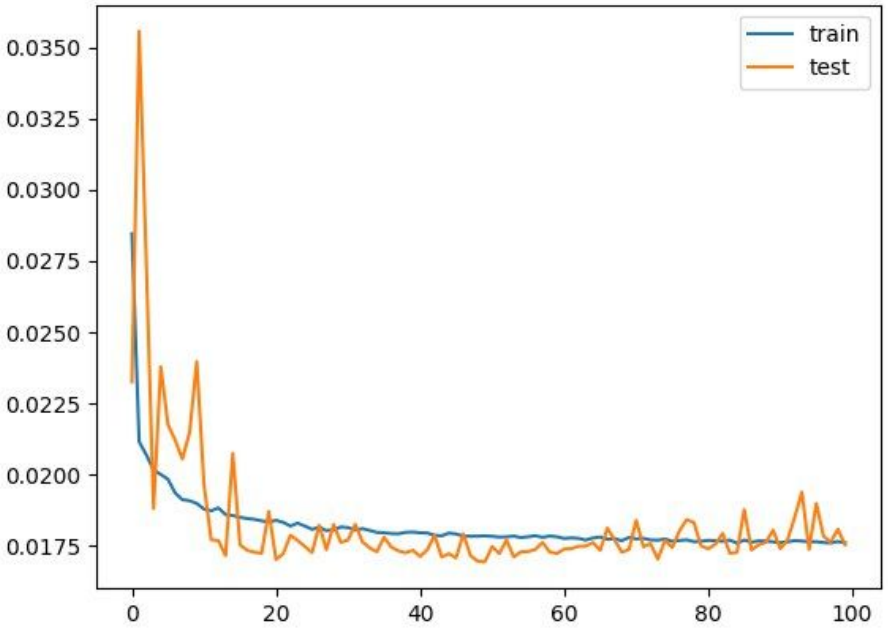
Pic.6.


train
test
0.026
0.024
0.022
0.020
0.018
0
20
40
60
80
100

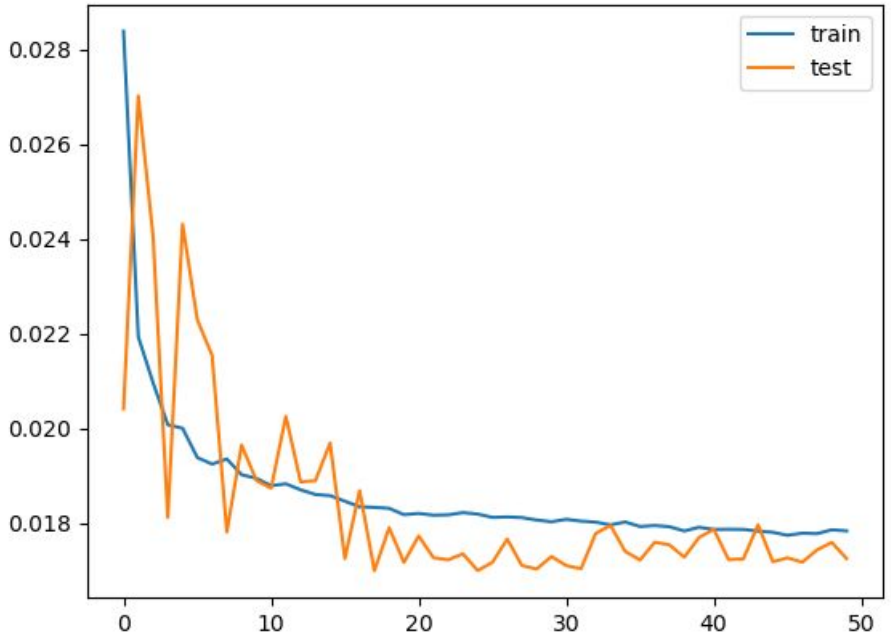
Pic.7.


train
test
0.028
0.026
0.024
0.022
0.020
0.018
0
200
400
600
800
1000

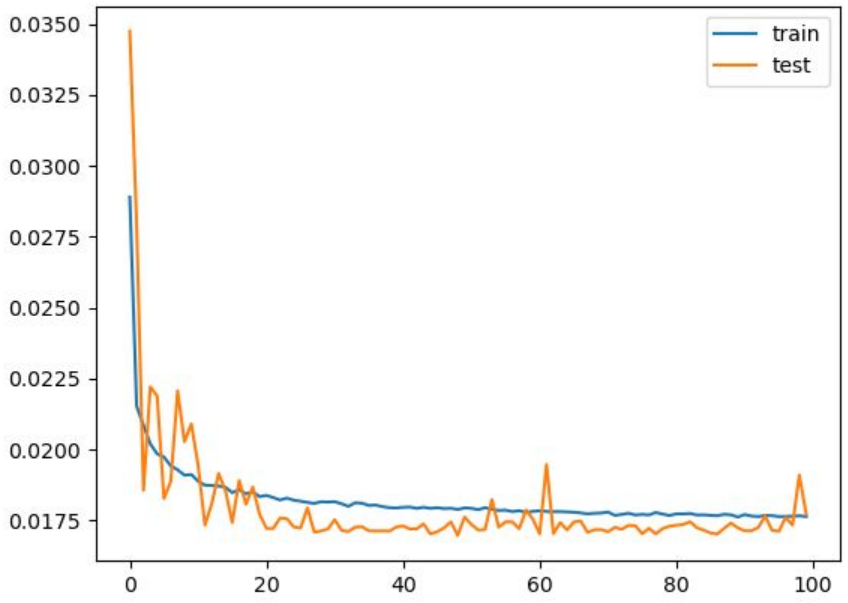
Pic.8.


train
test
0.0350
0.0325
0.0300
0.0275
0.0250
0.0225
0.0200
0.0175
0
20
40
60
80
100

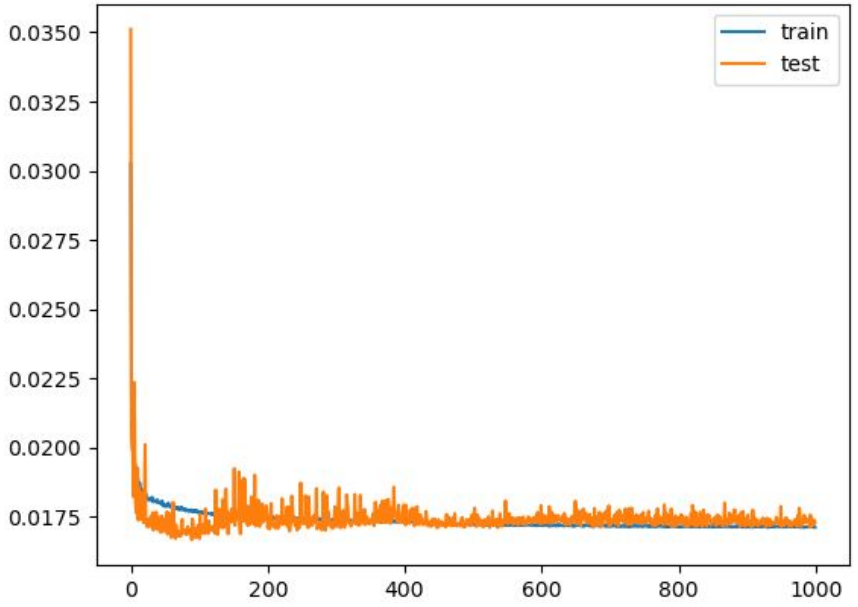
Pic.9.


train
test
0.028
0.026
0.024
0.022
0.020
0.018
0
10
20
30
40
50

Pic.10.


train
test
0.0350
0.0325
0.0300
0.0275
0.0250
0.0225
0.0200
0.0175
0
20
40
60
80
100

Pic.11.


train
test
0.0350
0.0325
0.0300
0.0275
0.0250
0.0225
0.0200
0.0175
0
200
400
600
800
1000

Pic.12.

Conclusion

While solving this problem, I fully understood the meanings of four V's of Big Data, because I was faced with the fact that the quantity was sufficient but the quality of data was very low.

Keras library provides a very convenient interface for the development of neural networks, abstracting the developer to write a large amount of code, making it possible to focus exclusively on research.

But unfortunately, sometimes it became not clear what exactly the developers of Keras had in mind under the hood of this or that parameter.