Fault detection of sensors with machine learning using LSTM neural networks

Vít Zeman

***Abstract*—Fault detection is one of the most important parts of diagnostics, as it can lead to the prevention of accidents and could lower the cost of maintenance. Depending on the type of fault, it can be compensated for in the short term. Some types of fault can be detected relatively easily, e.g. sudden jump in value. Today, as machine learning (ML) continues to expand, it can be used here for the detection of more complex faults, such as drift. This article will focus on creating a dataset and creating a neural network using Long-Short-term Memory (LSTM).**

***Index Terms*—Fault detection, machine learning, neural networks, LSTM, drift, offset, outlier**

I. Introduction

THE aim of this work is to propose a method for the detection of faults in temperature sensors over long periods of time from other sensors with different correlations. This method should contain an LSTM neural network.

The data used for this task are collected from the house in Colorado, USA, and consist of temperature sensors as well as other data such as energy consumption of multiple heating elements, energy created using solar panels, relative humidity, and others.

The code was written in Julia programming language[1] and is stored on Github[2].

II. Data

As with many other ML algorithms, the data set is one of the most crucial parts of the whole process. That, with the combination of non-triviality of creating custom dataset led to a significant time spent on this part.

The measured data are continuous measurements in time, each sample is averaged over an hour, and there are 16,885 samples for each measurement. That is, approximately 704 days of continuous measurements.

For data in this scope, it is not unusual that some measurements are incorrect. During the creation of the data set, multiple problems were encountered. First, some values were missing; thankfully, this problem was only with energy data. Therefore, it was assumed that when this occurred, the measured value was set to 0.

Second, some measured data were set to -6,999, which in the case of temperatures defiles the laws of physics, as the measured temperature was in degrees of Fahrenheit. This occurred in multiple temperature measurements at the same time and was continuous. Therefore, data from 12,267 to 12,277 were omitted.

[1]https://julialang.org/
[2]https://github.com/vitzeman/SemProject.jl

The omission leads to split to test and train data as the splits make final split 72.69 to 27.31, which is close to traditionally used 70/30 split.

For data set creation, every temperature measurement (that is, columns 8-24 and 29) and energy measurements (columns 31-47) were used. As a defective sensor, the temperature sensor in the master bedroom was used(column 13).

A. Faults

In this work, 3 faults were considered:

1) Drift - Data slightly increases each time step,
2) Offset - Data are shifted in y axis,
3) Outlier - Data jumps for shorter time than offset to values bigger than offset.

In figure 1 can be seen the demonstration of these faults on a constant function. In figure 2 visualisation of faults on provided data can be seen.

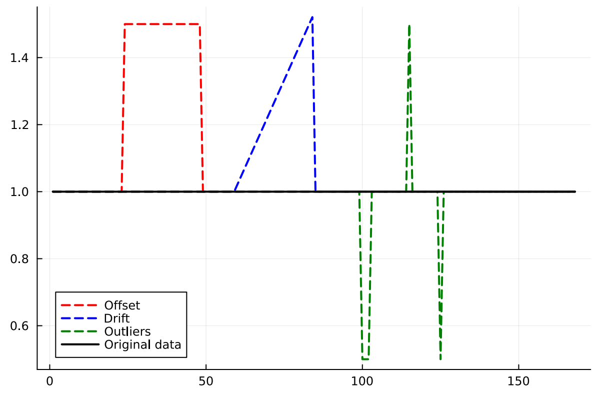


Fig. 1. Example of faults on simple line

B. Data set generation

For dataset reation the data were created at random location and for a random duration. The dataset has four classes:

1) Faultless,
2) Drift,
3) Offset and
4) Outliers.

It is generated at random with the distribution of 0.4 for faultless data and 0.2 for each of the faults. Each fault had a randomly generated value and duration.

The duration of drift was randomly generated in an interval of 120-216 hours (5-9 days). Its slopes were randomly selected

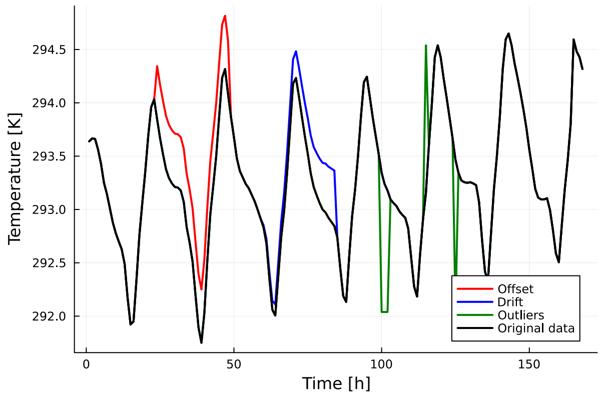


Fig. 2. Example of generated faults temperature data

so that the temperature would increase or decrease in 5 days by 0.75 - 1.25 kelvin.

The offset duration was randomly generated over an interval of 72-168 hours (3-7 days). And its offset value is to be in an interval from 0.5-5 kelvins.

The duration of the outlier was randomly generated in an interval of 72-168 hours (3-7 days). Their value was set to be in interval ±[10;30] kelvins.

C. Metrics

Metrics used in this work are the accuracy and the confusion matrix. Accuracy can be used as computed as the number of times the prediction was correct.

III. Neural networks

Although the idea of a neural network is not new. The rise of computational power in the new millennium allowed more exploration and expansion of different architectures. One family of these is recurrent neural networks (RNNs) and its subtype LSTM.

The idea behind RNN is that the previous input influences the current prediction[3]. That is where the loop comes in, which can be seen in Figure 3, its function is to allow information to persist in the network. However, in practise, RNNs are limited in how far back they can look due to the vanishing or exploding of gradient.

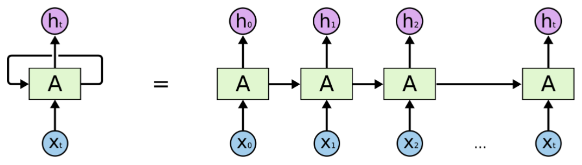


Fig. 3. RRN scheme on the left and unrolled RNN on the right [1]. The unrolled version shows how the cell works in time.

Therefore, a new type of network was proposed, the LSTM network[2], which was designed to deal with long-term dependencies. Its architecture can be seen in Figure 4. The idea is that in each iteration the cell can add or remove information to the long-term memory of the cell, through so-called gates.

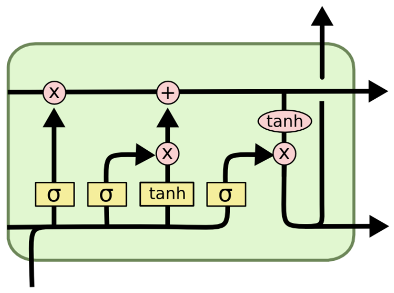


Fig. 4. LSTM architecture [1]. The sigmoid blocks, the values of which are between zero and one, determine how much information is important for given input. The Hyperbolic Tangent blocks, the values of which are between minus one and plus one, propose a new addition to the memory.[2]

IV. Proposed models

The simple network proposed from the LSTM cell followed by the Dense layer and softmax and can be seen in figure 5 This network proves the concept and the approach is valid.

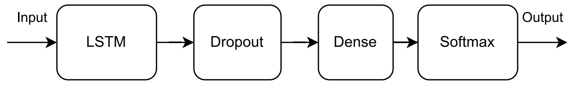
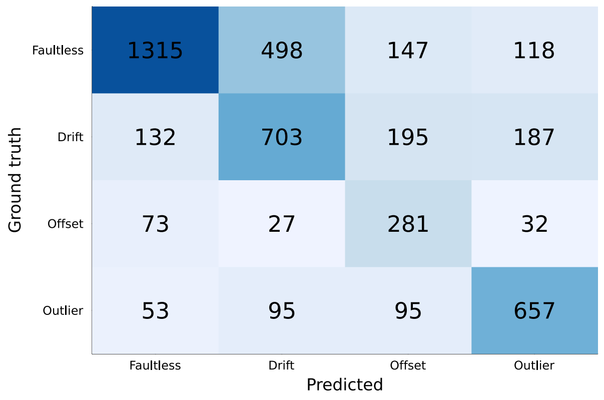


Fig. 5. Example of faults on simple line

As loss function, cross-entropy was chosen, and training was performed with ADAM optimiser.

V. Results

Although this network is simple, it was able to achieve 64% accuracy. In the figure 6 can be seen confusion matrix. From this can be observed that the most misclassification occurred with the combination of drift and faultless. This seems logical as these 2 are the most simmilar.



Ground truth \ Predicted	Faultless	Drift	Offset	Outlier
Faultless	1315	498	147	118
Drift	132	703	195	187
Offset	73	27	281	32
Outlier	53	95	95	657

Fig. 6. Confussion matrix of the simple model

VI. Conclusion

In this work, a data set was generated and an architecture using LSTM was proposed. The precision of the final model was 64%, the confusion matrix can be seen in figure 6. From what can be seen, the network had the biggest problems with the distinguishing drift from normal course.

REFERENCES

[1] Understanding lstm networks – colah's blog.
[2] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997.
[3] R. C. Staudemeyer and E. R. Morris. Understanding lstm-a tutorial into long short-term memory recurrent neural networks. 2019.