MBIT SCHOOL

ADVANCED DEEP LEARNING MASTER DISSERTATION

Deep Learning For Predictive Maintenance

Author:
Hugo PÉREZ BLASCO

Supervisor:
Dr. Manuel
SÁNCHEZ-MONTAÑÉS ISLA

July 19, 2020

MBIT SCHOOL

Abstract

Software Engineer

Deep Learning For Predictive Maintenance

by Hugo PÉREZ BLASCO

Predictive maintenance encompasses a variety of topics, including but not limited to: failure prediction, failure diagnosis (root cause analysis), failure detection, failure type classification, and recommendation of mitigation or maintenance actions after failure (*Predictive Maintenance: Step 2A of 3, train and evaluate regression models* 2015).

Predictive Maintenance is also a domain where data is collected over time to monitor the state of an asset with the goal of finding patterns to predict failures which can also benefit from certain deep learning algorithms (Fidan Boylu, 2017). This study uses simulated aircraft sensor values to predict when an aircraft engine will fail in the future so that maintenance can be planned in advance (Griffo, 2019).

The goal of this dissertation if to do a trade-off over the different deep learning architectures and models that compound the current state of the art concerning the prediction and classification over time series data.

Contents

List of Figures

List of Tables

List of Abbreviations

RUL	**R**emaining **U**seful **L**ife
RNN	**R**ecurrent **N**eural **N**etwork

Chapter 1

Objectives

1.1 Problem Description

Aircraft lifetime and maintenance is a recurrent problem for companies due to its high costs. Being able to predict when an aircraft is going to have failures or malfunctioning is a key topic for improving costs and performance of this machines.

The scenario uses data from simulated aircraft sensor to predict when an aircraft engine will fail in the future. Two different approaches are used for this problem:

- A regression models that manages to answer the question: How many more cycles an in-service engine will last before it fails?
- A binary classification model that manages to answer the question: Is this engine going to fail within a specific number of time cycles?

1.2 Data Summary

The data provided is divided in three different sets:

1.2.1 Training data

The training data consists of multiple multivariate time series with "cycle" as the time unit, together with 21 sensor readings for each cycle. Each time series can be assumed as being generated from a different engine of the same type. Each engine is assumed to start with different degrees of initial wear and manufacturing variation, and this information is unknown to the user. In this simulated data, the engine is assumed to be operating normally at the start of each time series. It starts to degrade at some point during the series of the operating cycles. The degradation progresses and grows in magnitude. When a predefined threshold is reached, then the engine is considered unsafe for further operation. In other words, the last cycle in each time series can be considered as the failure point of the corresponding engine. Taking the sample training data shown in the following table as an example, the engine with id=1 fails at cycle 192.

1.2.2 Text data

The testing data has the same data schema as the training data. The only difference is that the data does not indicate when the failure occurs (in other words, the last time period does NOT represent the failure point). Taking the sample testing data shown in the following table as an example, the engine with id=1 runs from cycle 1 through cycle 31. It is not shown how many more cycles this engine can last before it fails.

TABLE 1.1: Training Data

Id	**Cycle**	**Setting 1**	**Setting 2**	**S1**	**S2**	**S3**	**...**
1	1	-0.0007	-0.0004	100.0	518.67	641.82	...
1	2	0.0019	-0.0003	100.0	518.67	642.15	...
1	3	-0.0043	0.0003	100.0	518.67	642.35	...
...	...						...
1	191	0	-0.0004	100.0	518.67	643.34	...
1	192	0.0009	0	100.0	518.67	643.54	...
2	1	-0.0018	0.0006	100.0	518.67	641.89	...
2	2	0.0009	-0.0003	100.0	518.67	641.82	...
2	3	0.0018	0.0003	100.0	518.67	641.55	...
...	...						...

TABLE 1.2: Test Data

Id	**Cycle**	**Setting 1**	**Setting 2**	**S1**	**S2**	**S3**	**...**
1	1	0.0023	0.0003	100.0	518.67	643.02	...
1	2	-0.0027	-0.0003	100.0	518.67	641.71	...
1	3	0.0003	0.0001	100.0	518.67	642.46	...
...	...						...
1	30	-0.0025	0.0004	100.0	518.67	642.79	...
1	31	-0.0006	0.0004	100.0	518.67	642.58	...
2	1	-0.0009	0.0006	100.0	518.67	641.89	...
2	2	-0.0011	0.0002	100.0	518.67	642.51	...
2	3	0.0002	0.0003	100.0	518.67	642.58	...
...	...						...

TABLE 1.3: Ground Truth Data

RUL
112
98
69
82
91
...

1.2.3 Ground truth data

The ground truth data provides the number of remaining working cycles for the engines in the testing data. Taking the sample ground truth data shown in the following table as an example, the engine with id=1 in the testing data can run another 112 cycles before it fails.

1.3 Dissertation goals

Once the data and the problem to solve is known, the goal of this dissertation is to expose and analyze the different approaches and alternatives used for the problem resolution.

All of the different methodologies, technical solutions and results will be explained step by step and compared within each other.

Finally some conclusions must be extracted in order to clarify which one is the best approach to solve the problem concerning this dissertation context.

Chapter 2

Methodology and Techniques

2.1 Deep Learning for sequence processing

This dissertation is focused on the use of Deep Learning techniques to solve predictive maintenance problems.

Some of this techniques are going to be used to solve the problem exposed in chapter 1.

This methodologies and techniques are chosen based on the state-of-art of the topic and also on its frequency of use.

2.1.1 Study methodology

The methodology for the study follows the next steps:

- Data analysis and pre-processing.
- Technical solution and model definition.
- Model and data visualization.
- Analysis of the results.
- Hyper-parameters and model adjustment
- Conclusions

2.1.2 Trade-off

The results of each approach and technique will be compared in a trade-off extracting conclusion about its feasibility to solve the exposed problem.

Also, future directions and improvements will be discussed in agreement with the results of the dissertation.

Chapter 3

Data Analysis and Pre-processing

3.1 Data labelling

The first step is to generate the labels required to train the model. Two different labels must be generated for each question we want to answer:

- Remaining Useful Life (RUL) for the regression problem.
- Boolean flag indicating if the RUL is less than an specific cycle value.

3.1.1 Remaining Useful Life

In order to face the regression model to answer the question *how many more cycles an in-service engine will last before it fails?* each entrance of the train dataset must include the number of remaining cycles until the aircraft fails.

This way, all the rows associated with the same aircraft identifier will have a decreasing number for this field. In the last step, where the cycle number represents the moment the aircraft fails, the RUL must be 0. See table 3.1

3.1.2 Failure Cycle Window

In the case of the binary classification model, the question to answer is *is this engine going to fail within w1 cycles?*.

The w1 represents the window of cycles that we want to take in account to classify. In this dissertation the number of cycles will be set to 30.

TABLE 3.1: Training Data with RUL column

Id	Cycle	Setting 1	Setting 2	S1	...	RUL
1	1	-0.0007	-0.0004	100.0	...	191
1	2	0.0019	-0.0003	100.0	...	190
1	3	-0.0043	0.0003	100.0	...	189
...	...					...
1	191	0	-0.0004	100.0	...	1
1	192	0.0009	0	100.0	...	0
2	1	-0.0018	0.0006	100.0	...	286
2	2	0.0009	-0.0003	100.0	...	285
2	3	0.0018	0.0003	100.0	...	284
...	...					...

TABLE 3.2: Training Data with label for classification

Id	Cycle	Setting 1	Setting 2	S1	...	RUL	label1
1	1	-0.0007	-0.0004	100.0	...	191	0
1	2	0.0019	-0.0003	100.0	...	190	0
1	3	-0.0043	0.0003	100.0	...	189	0
...	...						...
1	191	0	-0.0004	100.0	...	1	1
1	192	0.0009	0	100.0	...	0	1
2	1	-0.0018	0.0006	100.0	...	286	0
2	2	0.0009	-0.0003	100.0	...	285	0
2	3	0.0018	0.0003	100.0	...	284	0
...	...						...

Along with the RUL, it is possible to generate a label (label1) for each entrance indicating if the aircraft failed on this cycle window. To calculate it, just set to 1 ("true") all the RUL values lesser than w1. See table 3.2

3.2 Feature Engineering and Normalization

The information of the aircraft for each time step is formed by 21 different sensor and 3 setting numerical values. The distribution of this values is shown in the figure 3.1.

The data distribution shows that most of the sensor data follows a gaussian distribution, avoiding the existence of outliers. With this premise and also taking in account that some of the value ranges can take negative values, the normalization method chosen is min max scaler.

Min max scaler normalization transforms the values in the distribution to the range [0-1]. This normalization is required to avoid the network to priorize on features with higher values and also to avoid exploding gradient problems.

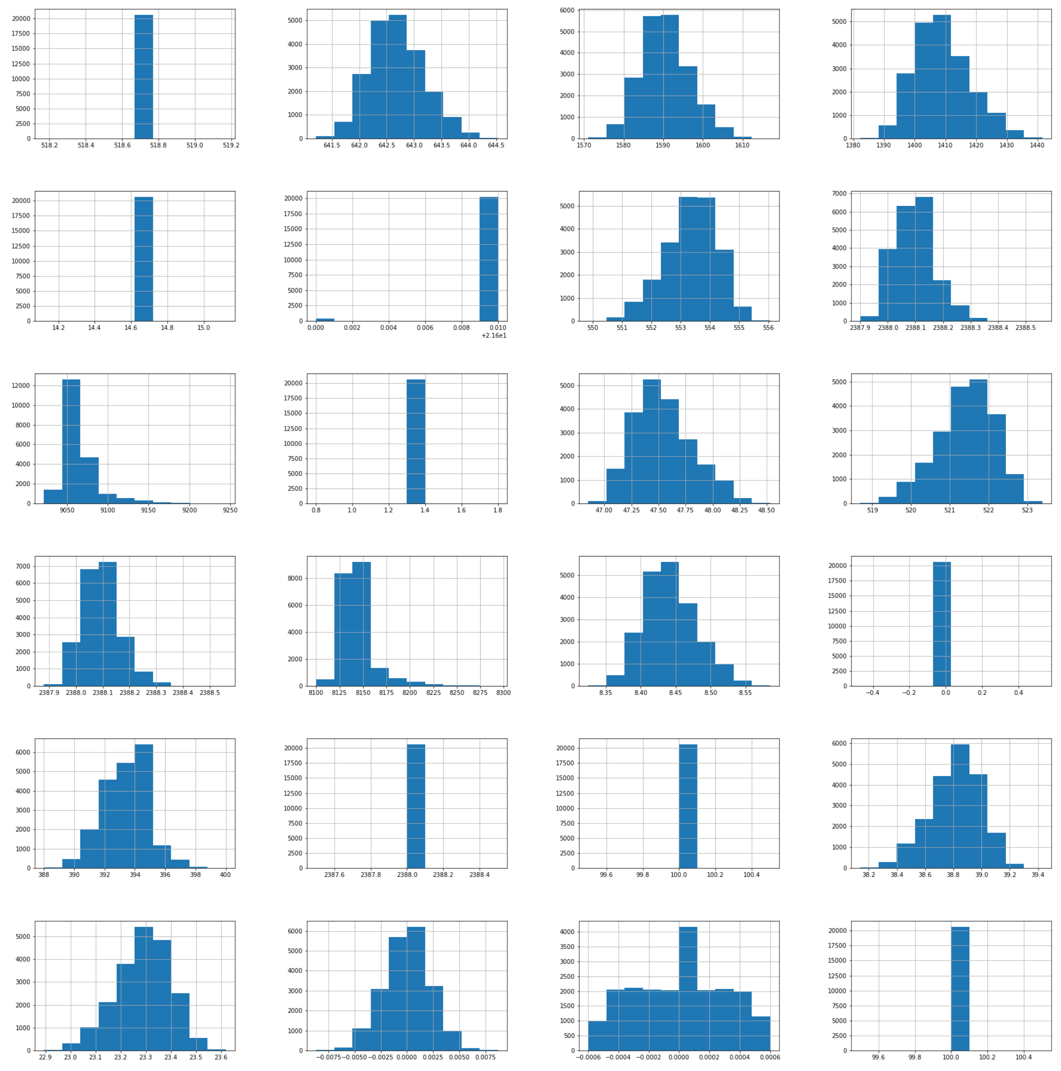

FIGURE 3.1: Sensor data distribution

Chapter 4

Long Short-Term Memory

4.1 Understanding LSTMs

Long Short Term Memory networks – usually just called *LSTMs* – are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter and Schmidhuber (1997).

LSTMs are explicitly designed to avoid the long-term dependency problem. This allows the network to remember information for long periods of time.

All recurrent neural networks have the form of a chain of repeating modules of neural network. But the core of the LSTMs is the *cell state*, that allows the network to keep a simple state information through the processing of all the time steps in the chain. An scheme of a LSTM architecture is described in figure 4.1.

4.2 Binary Classification

The next sections describe the methodology applied to solve the binary classification problem using LSTMs

4.2.1 Data Generation

Before the ingestion of the data into the LSTM model, it has to be adapted to the architecture of the network.

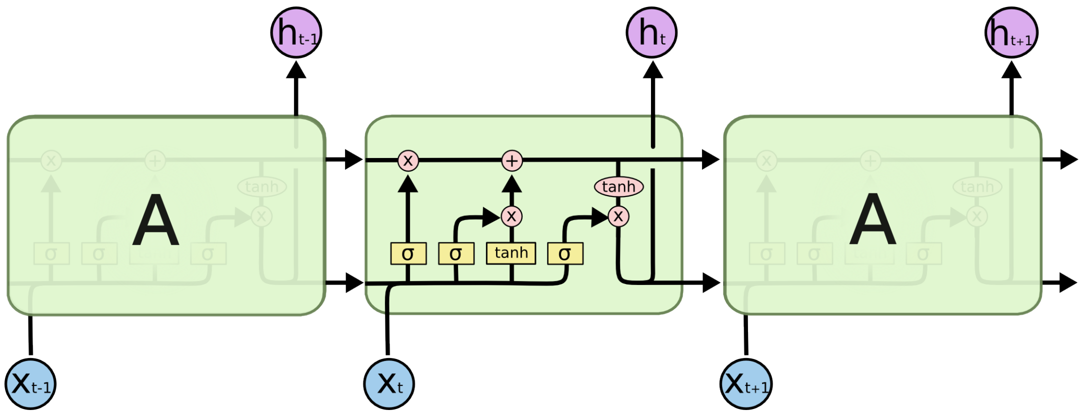

FIGURE 4.1: LSTM architecture.

Being *samples* the total number of data, *timesteps* the amount of rows to be ingested for each aircraft and *features* the number of sensor data the input of the LSTM layer is defined by a tensor of shape:

```
(samples, timesteps, features)
```

The time steps used for the study will be 50, which means that for each aircraft batches of 50 time steps will be taken. Using the *zip* function over the data of the aircraft with id=1, which contains 192 cycles. The batches will be distributed this way:

```
(0, 50)     -> from row 0 to row 50
(1, 51)     -> from row 1 to row 51
(2, 52)     -> from row 2 to row 52
...
(111, 192)  -> from row 111 to row 192
```

After the pre-processing, the input tensor will have the shape:

```
(15631, 50, 25)
```

To generate the labels the values of the column *label 1*, which was generated on the data preparation (see chapter 3), must be grouped by aircraft indicating if the engine failed within *w1* cycles.

The result label tensor has the shape:

```
(15631, 1)
```

4.2.2 Model definition

Work in progress...

Appendix A

Frequently Asked Questions

A.1 How do I change the colors of links?

The color of links can be changed to your liking using:
`\hypersetup{urlcolor=red}`, or
`\hypersetup{citecolor=green}`, or
`\hypersetup{allcolor=blue}`.
If you want to completely hide the links, you can use:
`\hypersetup{allcolors=.}`, or even better:
`\hypersetup{hidelinks}`.
If you want to have obvious links in the PDF but not the printed text, use:
`\hypersetup{colorlinks=false}`.

Bibliography

Fidan Boylu, Ahmad Afsar (July 2017). *LSTMs for Predictive Maintenance*. https://github.com/Azure/lstms_for_predictive_maintenance.

Griffo, Umberto (Apr. 2019). *Recurrent Neural Networks for Predictive Maintenance*. https://github.com/umbertogriffo/Predictive-Maintenance-using-LSTM.

Predictive Maintenance: Step 2A of 3, train and evaluate regression models (Apr. 2015). Microsoft Azure AI. URL: https://gallery.azure.ai/Experiment/Predictive-Maintenance-Step-2A-of-3-train-and-evaluate-regression-models-2 (visited on 07/01/2020).