

# Neural Networks for the Prediction of Fuel Oil Consumption for Containerships

Orfeas Bourchas

**Diploma Thesis**



School of Naval Architecture and Marine Engineering  
National Technical University of Athens

Supervisor: Assistant Prof. George Papalambrou

Committee Member : Prof. N. Kyrtatos

Committee Member : Associate Prof. C. Papadopoulos

November 2020



# Contents

<b>1</b>	<b>Abstract</b>	<b>7</b>
<b>2</b>	<b>Introduction</b>	<b>9</b>
2.1	The Problem	9
2.2	Fuel Oil Consumption Prediction related work	9
2.2.1	The Ahlgren and Thern approach [1]	9
2.2.2	Anan, Taizo, Higuchi, Hiroyuki, Hamada, Naoki Approach [2]	14
2.2.3	The Pedersen Approach [3]	19
2.2.3.1	Introduction	19
2.2.3.2	Regression methods	19
2.2.3.3	Training	20
2.2.3.4	Data	21
2.2.3.5	Results	24
2.2.4	Other related work	31
<b>3</b>	<b>Analysis of Ship Data</b>	<b>33</b>
3.1	Brief Introduction of Pandas, Jupyter Notebook & TensorFlow	33
3.2	Data Analysis	35
3.2.1	Data Acquisition	35
3.2.2	Data Preprocessing	36
<b>4</b>	<b>Neural Network Design</b>	<b>41</b>
4.1	First approach	41
4.2	Simple RNN [4]	41
4.3	LSTM [4]	45
4.4	The final approach	47
<b>5</b>	<b>Results</b>	<b>49</b>
5.1	One variable : Fuel Oil Consumption	49
5.1.1	1 Lookback	49
5.1.2	5 Lookbacks	53
5.1.3	10 Lookbacks	57
5.1.4	15 Lookbacks	60
5.1.5	20 Lookbacks	64
5.1.6	50 Lookbacks	67
5.2	Two variables:Fuel Oil Consumption and Fuel Oil Temperature	71
5.2.1	1 Lookback	71
5.2.2	5 Lookbacks	75
5.2.3	10 Lookbacks	78
5.2.4	15 Lookbacks	82

---

5.2.5	20 Lookbacks . . . . .	85
5.2.6	50 Lookbacks . . . . .	89
5.3	Three variables: Fuel Oil Consumption, Fuel Oil Temperature and Main Engine RPM . . . . .	92
5.3.1	1 Lookback . . . . .	93
5.3.2	5 Lookbacks . . . . .	96
5.3.3	10 Lookbacks . . . . .	100
5.3.4	15 Lookbacks . . . . .	103
5.3.5	20 Lookbacks . . . . .	107
5.3.6	50 Lookbacks . . . . .	110
<b>6</b>	<b>Conclusions and Future Work</b>	<b>115</b>
6.1	Conclusions . . . . .	115
6.2	Next steps . . . . .	115
	<b>Bibliography</b>	<b>117</b>

# Acknowledgments

This work has been carried out at the Laboratory of Marine Engineering (LME) at the School of Naval Architecture and Marine Engineering of the National Technical University of Athens, under the supervision of Assistant Professor George Papalambrou.

I would first like to thank my thesis supervisor Assistant Professor George Papalambrou for giving me the chance and motivation to work on this topic. I would also like to thank him for his patience, continuous support and immense knowledge. His guidance helped me in all the time working on this thesis.

I would like thank Professor Nikolaos Kyrtatos for evaluating my work and being a member of my supervisors committee.

I would also like to thank Associate Professor Christos Papadopoulos for evaluating my work and being a member of my supervisors committee.

I would like to express my sincere gratitude to Mr. Nikolaos Planakis, PhD candidate of School of Naval Architecture and Marine Engineering for sharing expertise, and sincere and valuable guidance during my last years in the institute.

I would also like to thank Bernhard Schulte Shipmanagement Hellas for providing me with the necessary data to carry out my thesis.

I am also sincerely grateful to my family, for the unceasing encouragement, support, motivation and attention throughout my studies.

I also place on record, my sense of gratitude to one and all, who directly or indirectly, have lent their hand in this venture. This accomplishment would not have been possible without them. Thank you.



# Chapter 1

## Abstract

In this thesis, a Neural Network for predicting the Fuel Oil Consumption (F.O.C.) of three containerships, two sister ships and another containership was studied and implemented. For this purpose several models were tested. The parameters that were used are:

- The variables that were used in each model,
- The number of steps back or lookbacks, as they are called in this thesis and
- The number of epochs that each model was trained.

The aim was to find the best parameters and create models that could not only predict well on the training dataset but also on the other ships' datasets. Initially the data of all three ships were preprocessed the same way. Then one of the sister ships was chosen to be the one which the dataset would be used for training, and some testing.

The Neural Networks are a sophisticated part of Machine Learning. There are a lot of types/categories of N.N. based on the architecture, the supervision and the way they train. In this thesis, the focus was on the Recurrent Neural Networks (R.N.N.) since these are the ones that tackle the timeseries problems such as the prediction of a value, in this case the FOC. Initially a simple RNN was the first thought to approach the problem. However, due to the large amount of data and some of the weaknesses of the simple RNNs the more advanced LSTM architecture was chosen due to its ability to cope with the simple RNNs problems.

After the model was selected and the hyper parameters were tuned the model began training on the 90% of the first ship's dataset. After the completion of each training the model's ability to predict was tested. First on the remaining 10% of the first ship's dataset. Then the % difference was calculated and plotted. Moreover, the model's performance was also tested on the second (sister ship) and on the third ships in order to evaluate its ability to generalize. The process was repeated for variety of combinations of variables and lookbacks and epochs. On each combination of variables and lookbacks the model's performance was tested using the epochs parameters to differentiate the models. So the best number of epochs to train a model, based on the number of lookbacks and the number of variables, was chosen. The results of the best models, showed that most of them were able to generalize and achieve almost 0% errors on the first ship and below 5% on the other two.





# Chapter 2

## Introduction

### 2.1 The Problem

The international shipping industry is responsible for the carriage of around 90% of world trade. Despite that, the impact of ship operations on the environment and the economics are major issues in the maritime industry. Based on a study carried out by the IMO the annual CO<sub>2</sub> emissions from ship operations reached approximately 800 million tons, which represents about 3% of worldwide CO<sub>2</sub> emissions during the year 2012. Moreover, the fuel expenses are the major costs for every shipping company, ranging from hundreds of millions to billions of dollars every year. Therefore, there is a need to reduce the excessive use of fuel oil. The main way of achieving it is by the accurate prediction of the fuel oil consumption which will result in better operational planning (such as dry docking).

### 2.2 Fuel Oil Consumption Prediction related work

There have been several attempts to accurately predict the Fuel Oil Consumption, as well as other variables. In the next few paragraphs some of this work will be briefly presented. Also, all of the tables, figures and equations in this section have been taken from the related papers-articles.

#### 2.2.1 The Ahlgren and Thern approach [1]

As part of their paper Mr. Ahlgren and Mr. Thern tried to predict the fuel oil consumption using three different ML algorithms. The data were manually collected by extracting data from the Valmarine machinery logging system and the ships logbook, for a period of 14-months, and were also validated by the ships schematics and they machinery crew. The Valmarine machinery logging system logs the temperatures, fuel flows and engine variables in the ship. The fuel oil consumption was mainly measured with volume flow meters.

The training process depended on the quality of the data. If the data were poor and full of noise, then the ML model being trained in wrong data. This would result in the model trying to minimise the error on the wrong data and thus probably being unable to accurately predict the future fuel flow. So as part of the preprocessing they tried to overcome the uncertainty of the fuel-flow by fitting the volume flow meter to the mass flow meter. In that way, it was possible to verify and thus minimize the uncertainty of the mass flow data. Moreover, all data was filtered for NaN (not a number), and as they assumed there was no back blow and all values below zero was set to zero. As final part of preprocessing analysis, they observed that during December 2014 and January 2015

the correlation between the flow meters are consistent within 0.003113-0.002274 standard deviations if the outliers were filtered out.

A supervised learning model was chosen for the purpose of this study. Supervised learning is the concept of manually choosing the best algorithm for a machine learning task, and this relies much on the experience of the data scientist as well as trying different setups. The ML-algorithms also has a large number of parameters that can be tuned, and some algorithms perform better with a scaling in the input data. For this reason two AutoML libraries were used in order to automate this process. The first one was Tree Pipeline Optimisation Tool (TPOT) library and the second one was the Auto Sklearn library. These tools optimise the supervised part of the ML-process and programmed to minimise the error for a machine learning pipeline. Both of the libraries were utilising the extensive Python Scikit-learn library as the ML-algorithm toolbox.

test_no	no_vars	var_1	var_2	var_3	var_4	vars
1 and 16	1	frp				frp
2 and 17	1	exh_T				exh_T
3 and 18	1	TC_rpm				TC_rpm
4 and 19	2	rpm	frp			rpm, frp
5 and 20	2	rpm	exh_T			rpm, exh_T
6 and 21	2	rpm	TC_rpm			rpm, TC_rpm
7 and 22	2	frp	exh_T			frp, exh_T
8 and 23	2	frp	TC_rpm			frp, TC_rpm
9 and 24	2	exh_T	TC_rpm			exh_T, TC_rpm
10 and 25	3	rpm	frp	exh_T		rpm, frp, exh_T
11 and 26	3	rpm	frp	TC_rpm		rpm, frp, TC_rpm
12 and 27	3	rpm	exh_T	TC_rpm		rpm, exh_T, TC_rpm
13 and 28	3	frp	exh_T	TC_rpm		frp, exh_T, TC_rpm
14 and 29	4	rpm	frp	exh_T	TC_RPM	rpm, frp, exh_T, TC_RPM

Table 2.1: Test, training setup and ML features.

The parameters that were available were fuel rack position (frp), exhaust gas temperature (exh\_T), engine RPM (rpm) and turbo charger RPM (TC<sub>rpm</sub>). These parameters were also chosen because they are often available in a standard logging engine system setup. Using these features it was possible to form a set of different input data. These combinations are given in Table 2.1 above. The test setup consists of 30 possible combinations for each model setup, and a total of 90 models for linear, TPOT and AutoSk.

Each of these combinations were trained with a ML, a linear model, and by two AutoML tools, TPOT and auto-sklearn.

A total of 90 models were trained by linear regression, TPOT optimisation for 10 generations and auto-sklearn regression. The training was fed into the fitting function with the corresponding features for each run and the corresponding measured fuel flow for the FO meter. The dataset was split, with the sklearn train test split tool, into 75% train data and 25% test data. The random seed for splitting the train and test set was manually set to 42 as to get a reproducibility.

The results demonstrate that the TPOT models are within 0.004833624 standard deviations and auto-sklearn 0.005572236 standard deviations of the measured value in all tests, in comparison with the linear models with 0.025050563. In Fig. 2.1 below it is demonstrated that the linear models show poor performance against the auto-ML methods. Note that the tests 0-14 are identical to 15-29 but on different fuel lines. The model trained on fuel line 2/4 (tests 15-29) demonstrated better performance. All AutoML model performed significantly better than the linear model baseline, which is also what could be expected.

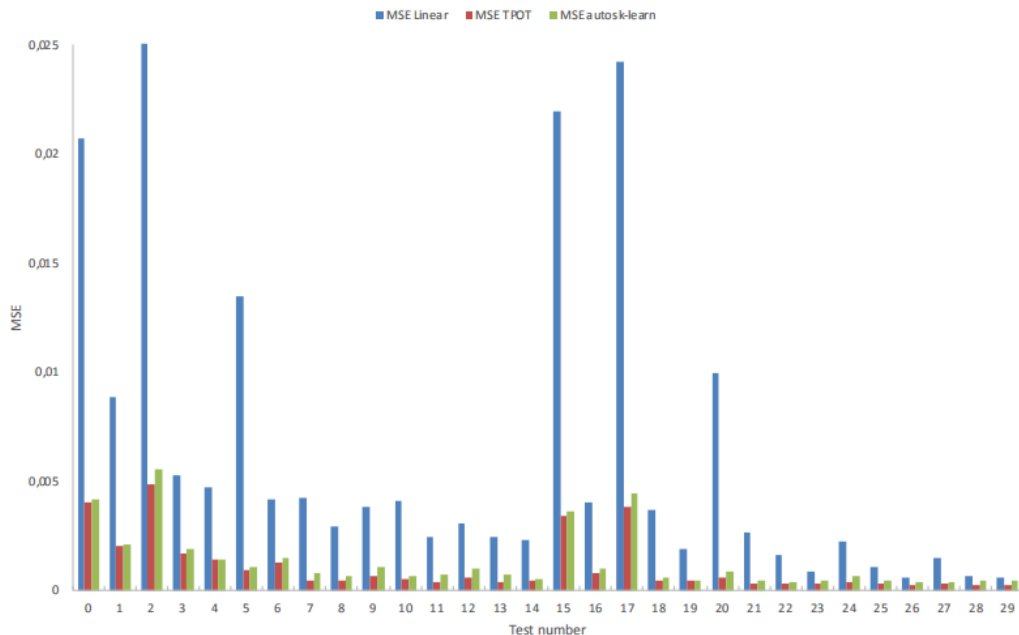


Figure 2.1: Mean squared error test results

In Fig. 2.2 the test numbers are sorted on TPOT mean squared error (MSE) from best to worst performance, in a comparison with the auto-sklearn. The TPOT models demonstrated better performance than auto-sklearn.

The six best performing tests are plotted in Fig. 2.3 as the difference between the model estimate and the measured value during an arbitrary day (2014-04-06 00:00 to 2014-04-07 00:00), it can be seen that there are some larger deviations of the model predictions at 15:00 and at 18:00. This is when the ship is manoeuvring in port, which means a lot of manoeuvring and as the data input for the model are averaged in 15-min interval dynamics from shorter time intervals are missed. As shown in Fig. 2.4, where only the TPOT and auto-sklearn models are shown, the model test TPOT 29 stands out.

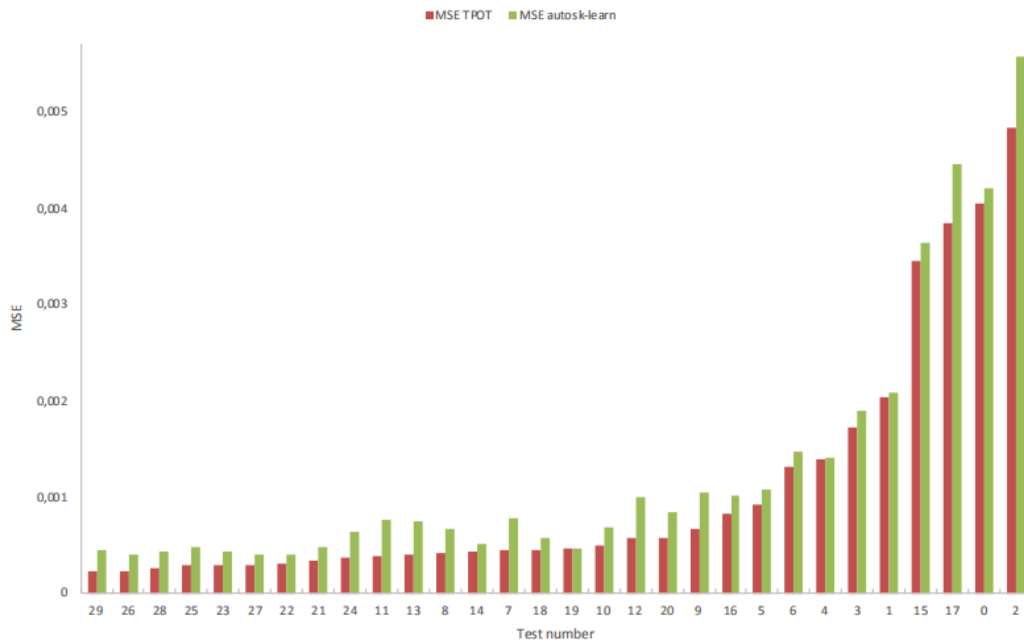


Figure 2.2: Mean squared error, comparison TPOt and auto-sklearn

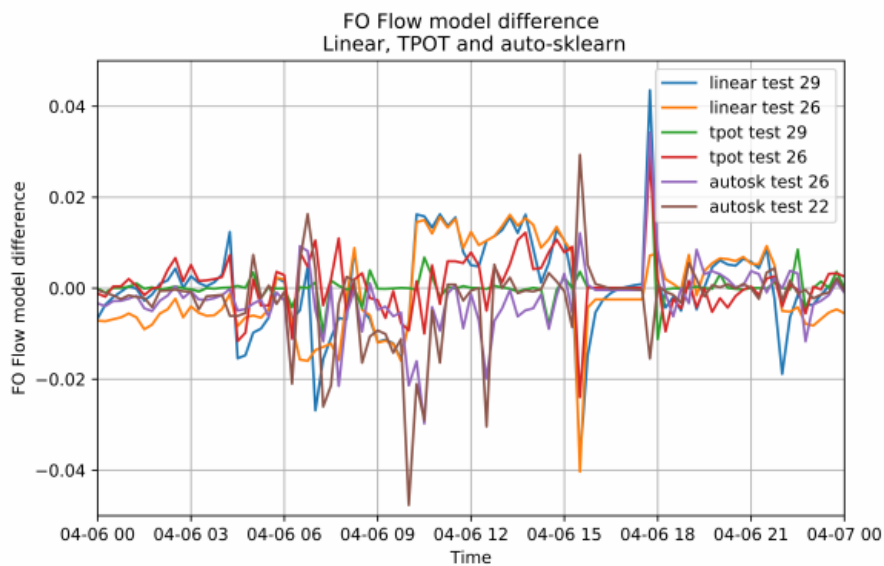


Figure 2.3: Best performing models, showing model difference

In Fig. 2.5 the fuel consumption of fuel line 2/4 is shown in absolute terms during the same time period, plotted with the model value over the measured value.

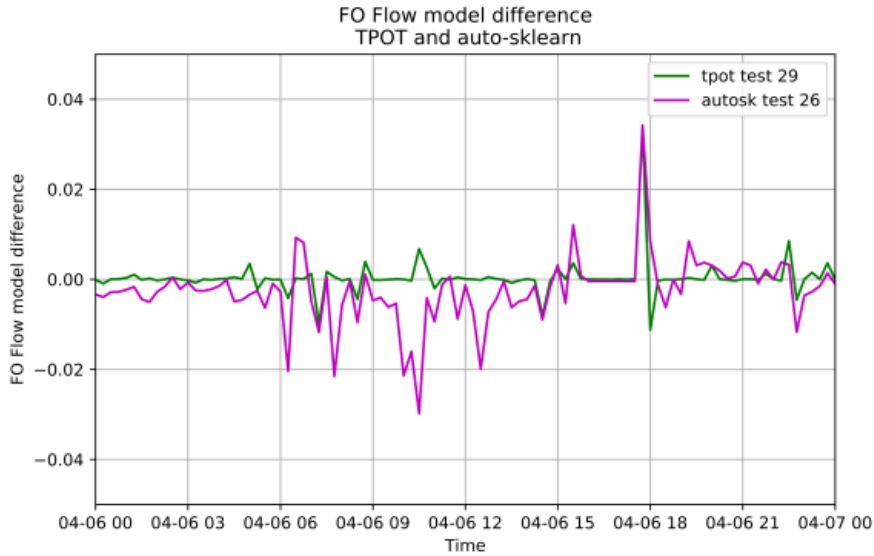


Figure 2.4: Best model performance TPOT and auto-sklearn

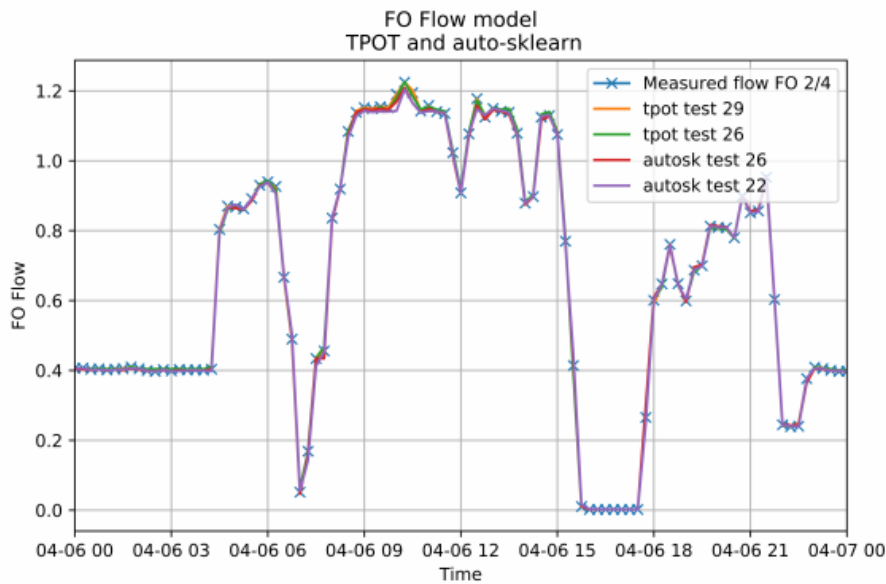


Figure 2.5: TPOT and auto-sklearn models compared to measured flow

In the table 2.2 below they ranked the features starting from the ones with the lowest MSE. From that table it is clear that if only using one variable, then the turbocharger RPM is the best model performing one. When using two variables, the engine fuel rack position provides the best addition. Both one and two variables show consistency in ranking between all tests, but when training with three variables the linear model gives best performance with the addition of the exhaust gas temperature. The TPOT model for both fuel lines and the auto-sklearn 2/4 give better results with the addition of the engine rpm. It is only the auto-sklearn model for the fuel line 1/3 which stands out in the AutoML models in the three-variable ranking.

no_vars	Linear 1/3	Linear 2/4	TPOT 1/3	TPOT 2/4	auto- sklearn 1/3	auto- sklearn 2/4
1	TC_rpm	TC_rpm	TC_rpm	TC_rpm	TC_rpm	TC_rpm
2	frp,TC_rpm	frp,TC_rpm	frp,TC_rpm	frp,TC_rpm	frp,TC_rpm	frp,TC_rpm
3	frp,exh_T, TC_rpm	frp,exh_T, TC_rpm	rpm,frp, TC_rpm	rpm,frp, TC_rpm	rpm,frp, exh_T	rpm,frp, TC_rpm

Table 2.2: Features for best ranked MSE.

To sum up the AutoML methods have demonstrated a large advantage over a standard linear regression training, and by only using a few features from the engines it was still possible to train a model which predicts consistently within a very small margin. By using AutoML methods, the time needed for manually evaluating and trying different machine learning algorithms was by far reduced and have also provided a very accurate model.

The AutoML models used in this study are open source and freely available to use as a concise and straight forward method for further refining the baseline for other black box ML-models, or real world predictions. The models can estimate the performance of each engine without installing flow meters for each individual engine. It is also possible given enough data, to use data of a lower interval such as daily or weekly bunker data, to train a model on engine features in which are recorded with a higher interval to make dynamic predictions.

### 2.2.2 Anan, Taizo, Higuchi, Hiroyuki, Hamada, Naoki Approach [2]

As it is commonly known, fuel costs and CO<sub>2</sub> emissions are the biggest concerns of shipping companies. For the last decades shipping companies have create performance departments with the goal to simulate the performance of the ship from the soar and use those simulations to reduce both fuel oil consumption and CO<sub>2</sub> emissions. However, most of the methods they use do not take into account the complex nature of the problem such as the complex interactions between wind, waves and sea currents that influence the state of the ship at real sea waters resulting in large margins of error. In their approach they present the technology that Fujitsu Laboratories have developed in order to cope with these discrepancies and better visualize the ship performance.

Over the past few years the maritime industry has been focused on developing energy saving technologies for ships varying from combustion efficiency to hull resistance. One of the most remarkable ones, is the weather routing technology. The aim of which is to select the route that minimizes the fuel consumption thus improving the fuel efficiency. It takes into consideration both the weather and the sea state and forecasts them and their anticipated impact on the ship resistances which impact the ship's speed and fuel consumption. The most common technique to calculate those resistances was by using existing physics models representing the hull, the waves and the winds. However, most of these models fail to represent precisely all the variables above and thus create large margins for error.

To cope with these problems the Fujitsu Laboratories used both their huge amount of stored data and implement their own AI called "Human Centric AI Zinrai". By using their technology Fujitsu Laboratories managed to estimate the ship's performance in real sea waters with a margin of error of 5% or less, which is an incredible level of accuracy.

Besides their AI technology, Fujitsu Laboratories also abandoned the conventional physics models and prioritizing high dimensional statistical analysis to determine the actual impact of the winds and waves on ship. This was accomplished by using the huge amount of stored data the Fujitsu Laboratories have acquired over the last years. The data were collected both from the operational reports and from engine logs. The data used were consisted of weather, sea states and ocean currents as well as of engine log variables and ship's speed.

As it is stated in their paper the features of the present technology are as follows:

1. Analysis based on actual ship operation data without physics models. The above mentioned high dimensional statistical analysis technology was implied and simultaneous analysis of various influences such as weather and sea conditions was successfully carried out using measurement data obtained from ship operation. This made it possible to estimate ship performance taking into account the complicated interactions of winds, waves, sea currents and the like, based on the raw data obtained in real sea waters, not data from water tank experiments.
2. Automatic grouping of actual measurement data and learning level adjustment. As shown in the figure 2.6, in a conventional physics model, the estimation accuracy cannot be improved because the physics phenomena are represented by a model that simplifies, for example, by classifying wind strength as either uniformly "weak" or "strong". By contrast, as shown in figure 2.7 the present technology automatically groups high dimensional data integrating various measured data by similarity, such as weather and sea conditions, allowing learning estimation according to each group. As a result, suppression of estimation errors due to averaging over the whole range was achieved.

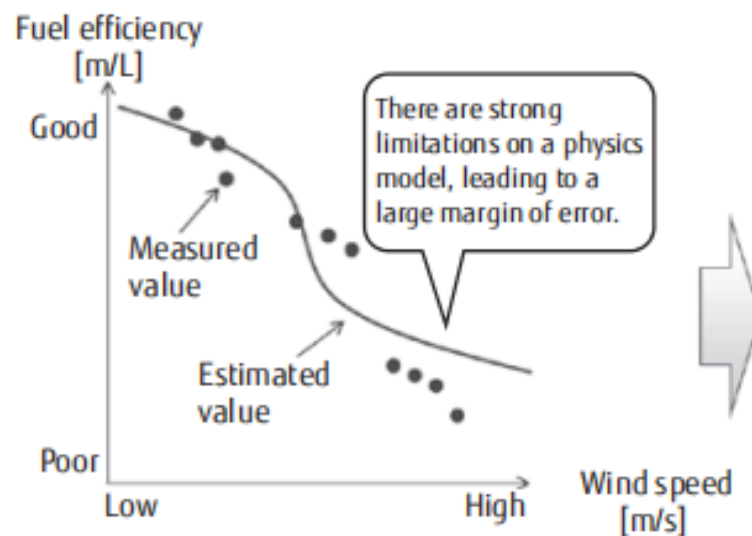


Figure 2.6: Conventional technology (physics model)

Regarding the system configuration and models of operation, the Fujitsu Laboratories included the previous technology into Fujitsu Intelligent Society Solution SPATIOWL, which is a cloud service that utilizes the location information to create a system for providing various ship related services. The Figure 2.8 demonstrates an example, in which

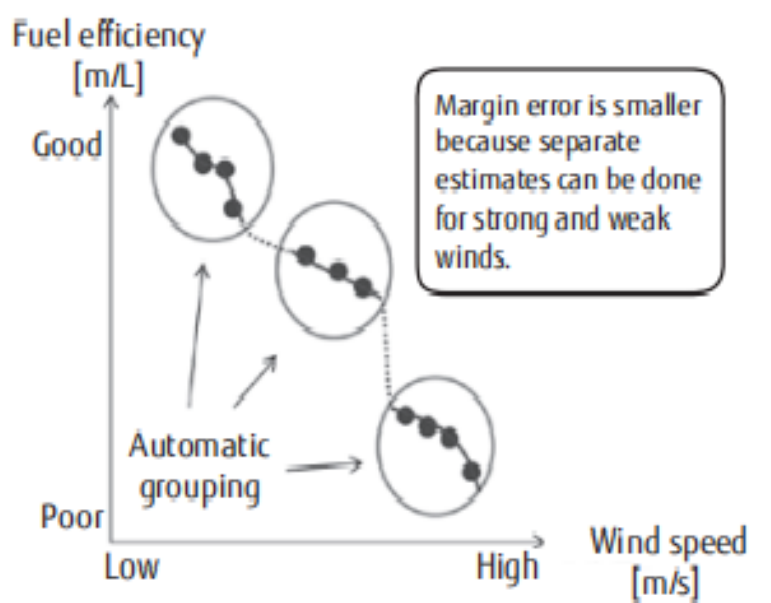


Figure 2.7: Newly developed technology

the AI and a weather routing simulator are mounted on SPATIOWL. This visualization provided by SPATIOWL allowed the organization to harvest accurate data that were after used to determine the best route for the selected ship, resulting in a lower fuel consumption. Last but not least this technology could also make a comparison of ship performance before and after maintenance.

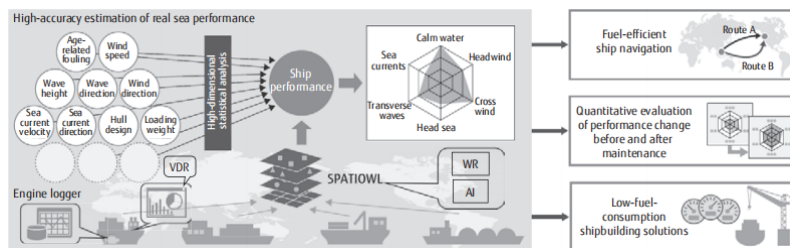


Figure 2.8: Example of incorporation of developed technology into SPATIOWL

During the development of the previous technology the Fujitsu Laboratories collaborated with Tokyo University of Marine Science and Technology. They collected data from several navigation, off the coast of Tateyama, from a ship owned by the University. The acquired variables were the following:

- Bow direction
- Speed Through Water
- True Wind Direction
- True Wind Speed
- Rudder Angle
- Controllable Pitch Propeller blade angle
- Shaft Revolutions



- Shaft Power
- Main Engine Revolutions
- Fuel Consumption

The performance of the ship was calculated using the obtained variables from the measurements. Since the developed method did not use any physics model any additional measured variable could be added to the analysis. The data was measured and accumulated at one second intervals during the ship operation.

In this evaluation the acquired data was divided into two sets one to be used for learning and one to be used for estimation. The ship's performance was visualized using only the learning data. The relationship between STW and nine other items was visualized. Similarly, with regard to the fuel consumption, the relationship between fuel consumption and nine other variables was visualized.

Whether fuel cost and ship's speed could be estimated from the visualization of the ship's and the performance was verified using the test data set. The results are shown in the graphs below. The horizontal axis of the first graph indicates the actual measured ship speed and the vertical axis represents the estimated ship speed. The closer the plotted points are to the diagonal line the closer the estimated value is to the measured one. On the left part of the graph shows the the measure values and the estimated ones arranged in chronological order. The more the 2 lines overlap the higher the accuracy.

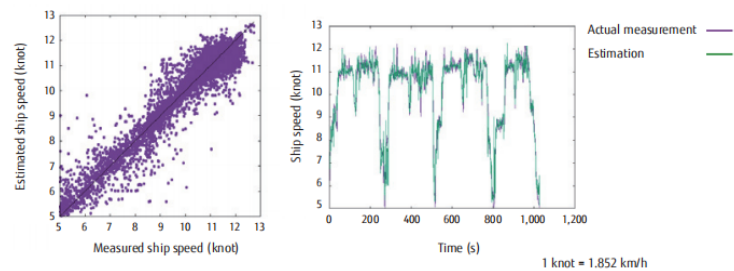


Figure 2.9: Estimated accuracy of ship speed

The horizontal axis of the second graph indicates the actual measured fuel consumption and the vertical axis represents the estimated fuel consumption. The closer the plotted points to are along to the diagonal line the closer the estimated value is to the measured one. Similarly to the previous graph the left part shows the the measure values and the estimated ones arranged in chronological order. The more the 2 lines overlap the higher the accuracy.

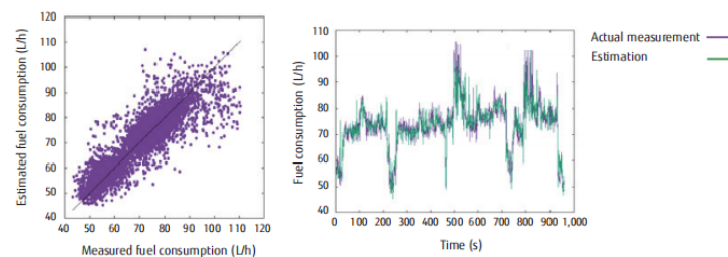


Figure 2.10: Estimated accuracy of fuel consumption

	MAPE%				Calculation time(s)
	Average	Standard Deviation	Max.	Min.	
STW(knot)	2.8	0.0001	3.0	2.6	11
Fuel Consumption(Lh)	4.4	0.0002	4.8	4.3	11

Table 2.3: Evaluation results of ship performance estimation.

On the table 2.3 below there are listed the results of the evaluation of the estimation error using cross validation.

The ten fold cross validation is a validation method that repeats verification a total of ten times, by dividing the entire data into ten blocks, selecting and learning nine out of these ten blocks and using the data of the remaining block in order to evaluate the estimation error in all blocks. As error index, the mean absolute percentage error(MAPE) was obtained by the following equation

$$MAPE = \frac{1}{T} \sum_{t=1}^T \frac{|f(x_t) - y_t|}{|y_t|}$$

where T is the number of samples of evaluation data, t is the ordinal number among all T samples,  $x_t$  is the input data used for estimation of the t-th sample,  $f(x_t)$  is the estimated value of the STW or the fuel consumption relative to  $x_t$  and  $y_t$  is the measured value of the STW or the fuel consumption of the t-th sample. The mean absolute percentage error with respect to the estimation of the STW was 2.8% and the mean absolute percentage error with respect to the estimation of the fuel consumption was 4.4%, showing that the proposed technology is capable of high accuracy estimation in both cases with an error of less than 5%. Generally, although it depends on the nature of the data and the accuracy, 3,000 data were found in this experiment to be the number of samples required to estimate both the STW and the fuel consumption with an error of less than 5%.

As shown in the previous table, the calculation time required for learning was just 11 seconds for both STW and fuel consumption, a level that qualifies as high-speed learning. The proposed technology was also applied in merchant ships actually carrying cargoes to verify its accuracy. From the merchant ships operation data and engine log data, the ship's speed and fuel consumption were visualized and the ship's speed and fuel consumption on actual routes were estimated. As a result similarly to the verification described above, it was found that both speed and fuel consumption can be estimated with an error of 5% or less.

Based on the results above, usage of this technology allows prediction of ship's speed and fuel consumption with high precision on routes to be travelled. Thus, the proposed technology is considered to be more suitable for optimum route selection than traditional weather routing algorithms that use physics models.

The results of the demonstration off Tateyama were incorporated into the weather routing of the TUMSAT and the fuel consumption reduction effect of this technology was evaluated. It was confirmed that when navigating optimum routes based on the ship performance visualized based on this technology, fuel consumption can be reduced by about 5% compared with navigation of the shortest route and significant reductions in fuel costs and CO<sub>2</sub> emissions can be achieved as a result.

As part of their future work the Fujitsu Laboratories plan to continue the research of this topic with collaboration with TUMSAT and also try to demonstrate the application of this technology to various types of ships.

## 2.2.3 The Pedersen Approach [3]

### 2.2.3.1 Introduction

In his paper Mr. Pedersen shows how the Gaussian Process Regression (GPR) can be used equally good or better than the Artificial Neural Networks (ANN) for short and long term predictions of the energy consumption on a ship.

Once more, the increase in fuel prices in combination with the need to be more economical as well as the environmental regulation have led the maritime industry in search for increased fuel efficiency on ship operations. This created the need for both long and short term prediction of energy consumption.

According to Mr. Pedersen, Energy and fuel efficient operations of ships have many facets. Both comprise voyage planning, efficient loading and discharging, if possible optimal trim of the loading conditions, and long term factors as wear on the engines and other mechanical part together with fouling of the hull and propeller. Since the conditions of the hull and propeller are difficult to assess it is also difficult to estimate the effect of fouling of the propulsion power. Traditionally the effect of fouling has been evaluated by comparing for example the actual fuel consumption with a theoretical estimate based on empirical methods and other available data such as model tests. Two well-known and robust methods are Holtrop (1984) and Harvald (1983) that are based on statistical data from model tests of a large number of ships, which makes them more suitable for ship models and loading conditions resembling to the ones tested. This is often not the case since model tests usually only are performed at the design and/or one ballast condition.

### 2.2.3.2 Regression methods

Gaussian Processes, GP, is a non-parametric model that provides a flexible framework for regression. The regression function does not take a predetermined form but is constructed from information derived from the data. The definition by Rasmussen and Williams (2006) is: "A Gaussian process is a collection of random variables, any Gaussian process finite number of which have a joint Gaussian distribution". GP is the most flexible class of non-parametric function estimators which can be interpreted as an infinite ANN, Neil (1994).

A GP can also be described as a multivariate Gaussian distribution over functions (instead of a scalar or vectors) and the general form can be written as in:

$$f(\mathbf{x}) \approx GP(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')) \text{ or } N(\mu, \Sigma) \quad (2.2.1)$$

where  $m(\mathbf{x})$  or  $\mu$  was the mean function  $E[f(\mathbf{x})]$  and  $k(\mathbf{x}, \mathbf{x}')$  or  $\Sigma$  was the covariance function  $E[(f(\mathbf{x}) - m(\mathbf{x}))(f'(\mathbf{x}') - m'(\mathbf{x}'))]$ . The mean function can be set to zero, and the covariance function used for this problem was the Squared Exponential ( $K_{SE}$ ).

$$cov(y) = K_{SE}(\mathbf{x}, \mathbf{x}') + \sigma_n^2 \quad (2.2.2)$$

$$K_{SE}(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp \left[ -\frac{1}{2} \sum_{d=1}^D \left( \frac{x_d - x'_d}{l_d} \right)^2 \right] \quad (2.2.3)$$

where  $\sigma_f^2$  was the predictive variance,  $\sigma_n^2$  was the noise variance, and  $l$  was the characteristic length-scale,  $D$  was the dimension of the input variables  $x$ ,  $\sigma_f^2$ ,  $\sigma_n^2$  and  $l$  were also referred to as the hyperparameters.

The length-scale can be thought of as the distance you have to move in input space for the function value to change. In the present problem, the input is multi-dimensional and

thus there is one length scale for each of the input variables and the hyperparameters thus defined as:  $\theta = l^1, l^2, \dots, l^d, \sigma_f^2, \sigma_n^2$ . The model was trained by optimizing the covariance function with respect to the hyperparameters by maximum likelihood also referred to as Automated Relevance Determination routine, ARD. This left one optimum length-scale for each of the input variables  $l^1, l^2, \dots, l^d$ .

Predictions by GPR were found by making a joint distribution of the training target values  $y = [y_1, \dots, y_N]$  and the test values  $y_t(x_t)$ , and from this the predictive function value and variance can be derived to 2.2.4 and the predictive variance  $\sigma_t^2$  can be found as the sum of the predicted covariance matrix  $cov(f(x_t))$  and the variance  $\sigma_n^2$ .

$$f(x_t) = \mathbf{x}_t \quad (2.2.4)$$

$$\sigma_t^2 = \mathbf{K}(\mathbf{x}_t, \mathbf{x}_t) - \mathbf{K}(\mathbf{x}_t, \mathbf{x}) \boldsymbol{\alpha} \mathbf{K}(\mathbf{x}, \mathbf{x}_t) \quad (2.2.5)$$

where

$$\boldsymbol{\alpha} = (\mathbf{K}(\mathbf{x}, \mathbf{x}) + \sigma_n^2 \mathbf{I})^{-1} \quad (2.2.6)$$

GPR was a fast method, but the complexity increases with the amount of input  $O(N^3)$ , so it was not appropriate for analysis of large data sets. The training data were explicitly used for prediction, so these could be computationally expensive.

One of the strengths of GPR was that from ARD, the length-scale was found, which determines how relevant each input variable was for the regression. Input variables with a small length-scale had a higher influence than the variables with high length-scales due to the short distance the input had to move in order to change the function. In the variable analysis all the length-scales were presented as the logarithm  $\log(l)$  due to the large variation in the length-scales. Furthermore the prediction variance was calculated for every prediction.

### 2.2.3.3 Training

In order to maximize the efficiency of the data, Mr. Pedersen divided them into several training/test sets. According to his research, the most efficient use was by training with all the data except one (N-1) and test with the single remaining input/output variable that was not used for training. This could be done alternately N times so all data points were used for test at a time. This was referred to as "Leave-One-Out" (LOO). Prediction errors were calculated for each training/test set, and the mean value of the test error from all the test sets was referred to as the "Cross-validation error".

The input  $x_i^d$  was normalized by the mean and standard deviation:  $\tilde{x}_i^d = \frac{x_i^d - \bar{x}^d}{std(x^d)}$ . This was done in order to avoid too large variations in the input variables leading to large variations in the length-scale and hence the predictions. Since GPR with zero-mean was assumed, the output variable was centred around 0 with following normalization:  $\tilde{y} = y_i - \bar{y}$ .

The initial guess of the hyperparameters occasionally resulted in local minima depending on the data length which was discovered by the prediction variance being very small or zero. To overcome this, the training/test set was also split up into MS parts accumulating to the full dataset 2.2.7. Each training was restarted with the final hyperparameters from the previous run in total MS-times as in 2.2.8. After a few tests, it was concluded that it was sufficient to restart the training twice depending on the data size, i.e. three trainings of the GPR, subsequently two restarts were used for all the trainings.

$$\theta(\mathbf{x}_{ms_{i+1}}) = \theta(\mathbf{x}_{ms_i}) \quad (2.2.7)$$

$$\mathbf{x}_{ms_i} \in \mathbf{x} \left( 0 : i \frac{|x|}{MS} \right) \quad (2.2.8)$$

Where  $\mathbf{x}$  was the total input data set and  $\mathbf{x}_{ms_i}$  was the multi start subset.

### 2.2.3.4 Data

From the existing empirical methods for predicting the propulsion power, it was clear that the necessary input variables were the following:

1. Draught midship [m]
2. Trim Ta-Tf (Draught aft- draught fore) [m]
3. Ship speed [knots]
4. Relative wind speed [m/s]
5. Relative wind direction [degree]
6. Wave height [m]
7. Relative wave direction [degree]
8. Water temperature [degree C]
9. Air temperature [degree C]

The output variable was either the propulsion power measured by a torsionmeter or the specific fuel consumption.

**Container ship data** The data were systematically collected from 5 sister container ships for a period up to 10 years. The logging periods were long and contained both dry-docking and hull cleanings, which gave an interesting insight into how these operations influenced the performance. The data were well organized, purged of irrelevant data and seemed to be very consistent, especially the manual observations of the wave height/direction and wind speed/direction.

The figure 2.11 shows the distribution of the logged speed and shaft power, it indicates a narrow speed profile with a mean around 23 knots and with almost no occurrences of speeds out of the range of 20-25 knots. The power distribution was much broader which indicates that the power was adjusted to meet the speed for different draught, environmental conditions, etc. On the table 2.4 below there is an overview of the datasets where the logging period until the first dry-docking is presented together with the total number of data points, number of dry dockings and hull cleanings within the period.

Ship ID	Period year to first docking	Total no. of NR	No. Docking	No. Hull cleaning
1	0-4.5	2337	2	7
2	0-4.5	2283	2	2
3	0-4.9	2268	1	3
4	0-5.0	2679	2	3
5	0-5.0	2564	2	4

Table 2.4: Noon reports from the container ships

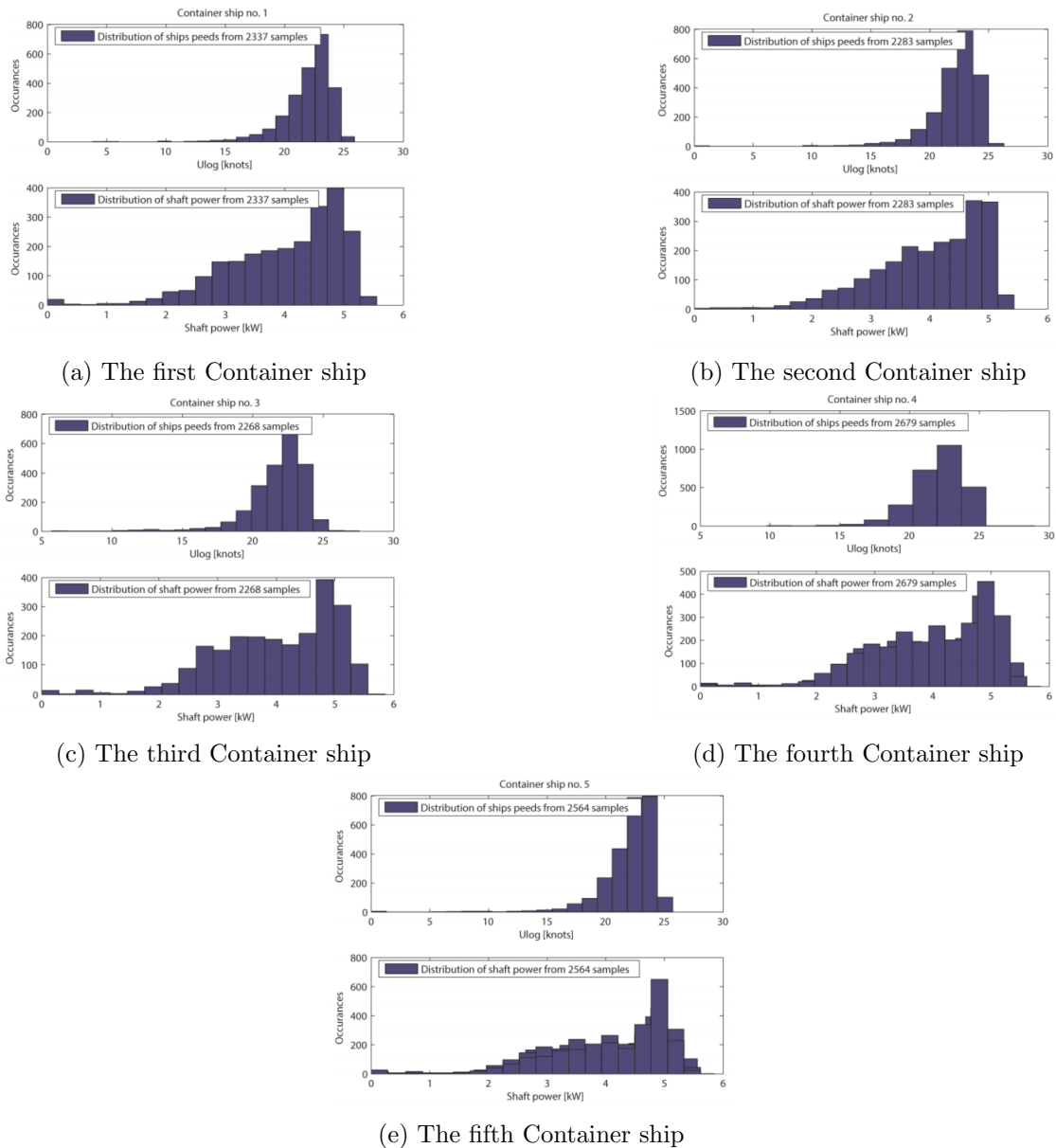


Figure 2.11: Speed and power distribution for the five container ships

In order to supplement the noon report hindcast data were used. Hindcasts are weather information at a certain time and position in the past. The data is received by a tool developed for Seatrend (Performance Monitoring tool developed at FORCE Technology, [www.force.dk](http://www.force.dk)) at FORCE Technology based on weather information from the NOAA (National Oceanic Atmospheric Administration, US Dept. of Commerce) database. This added several new variables to the noon reports, but it also limited the number of data, since not all areas were covered by the NOAA database. Many of the smaller seas i.e. the Mediterranean, North sea, Baltic sea, were not included. This reduced the number of data, approximately, by half.

Since the weather observations were instantaneous values, usually taken around the reporting time, and other important variables were average values from the previous noon report time, it was not correct to use them together. It would have been ideal to have "noon reports" for every hour to increase the accuracy of the hindcast weather information. As this was not possible, hindcasts were made between every noon report, but with one-hour

x	Data variable	Source	ID	Unit
1	Speed through water	Noon report	NRxls.Ulog	knots
2	Speed over ground	Noon report	NRxls.Uobs	knots
3	Sea water temperature	Noon report	NRxls.Tsw	deg
4	Mean draught (Ta+Tf)/2	Noon report	NRxls.Tm	m
5	Trim, Ta-Tf	Noon report	NRxls.Trim	m
6	True wind speed	Noon report	NRxls.WindSpeed	m/s
7	Relative wind direction	Noon report	NRxls.WindDir	deg
8	Average relative winds speed during report period	Hindcast	HC.Vrel	m/s
9	Average relative winds direction during report period	Hindcast	HC.gammarel	deg
10	Average significant wave height during report period	Hindcast	HC.mean.Hs	m
11	Average wave period during report period	Hindcast	HC.mean.Tp	s
12	Variance of the significant wave height during report period	Hindcast	HC.var.Hs	$xm^2$
13	Variance of the wave period during report period	Hindcast	HC.var.Tp	$s^2$
14	Variance of the wave direction during report period	Hindcast	HC.var.Td	$deg^2$
15	Variance of the winds speed during report period	Hindcast	HC.var.Ws	m/s
16	Variance of the winds direction during report period	Hindcast	HC.var.gamma	deg
17	Report date and tim (Matlab numeric value)	Noon report	NRxls.UTC	numeric
18	Average winds speed during report period	Hindcast	HC.Ws	m/s
19	Average winds direction during report period	Hindcast	HC.gamma	deg
27	Sea state	Noon report	NRxls.SeaState	m
28	Relative sea direction	Noon report	NRxls.TrueRelativeSeaDirection	deg
36	Average shaft power	Noon report	NRxls.PropPower	kW

Table 2.5: Container ship input data

intervals and equivalent position, and then the average of the values between the present and the previous noon report were found.

Similarly, the variance of the hindcasts for every noon report period was determined and made available for input to give more detailed weather information. The goal was to find weather that was equivalent to the effect spent in the same period, but since this was not possible, it was believed to be better than using only one observation (noon report time) to represent the past 24 hours. Table 2.5 shows the list of the data used for the

analysis.

### 2.2.3.5 Results

The evaluation was done by comparing the relative prediction errors of energy consumption. As described above, the cross-validation had been performed by Leave-One-Out. The cross-validation error  $\overline{\omega}_k$  was the mean error of the mean from the  $k$ 'th subsets of all the relative tests errors  $\omega_{n_k}$ . For *LOO*,  $K$  was equal to the length of the total data set  $N$ . In a similar way the cross-validation value of the relative predictive standard deviations had been determined.

Similarly the cross-validation value of the relative predictive standard deviations  $\overline{\sigma}_k$  had been determined:

$$\omega_{n_k} = \frac{\widehat{EC}_{n_k} - EC_{n_k}}{EC_{n_k}} \quad (2.2.9)$$

$$\omega_k = \frac{1}{N_k} \sum_{n_k=1}^{N_k} |\omega_{n_k}| \quad (2.2.10)$$

$$\overline{\omega}_K = \frac{1}{K} \sum_{k=1}^K |\omega_k| \quad (2.2.11)$$

$$\sigma_k = \sqrt{\frac{1}{N_k} \sum_{n_k=1}^{N_k} (\omega_{n_k} - \omega_k)^2} \quad (2.2.12)$$

$$\overline{\omega}_K = \frac{1}{K} \sum_{k=1}^K \sigma_k \quad (2.2.13)$$

Where:

$\widehat{EC}_{n_k}$  was the predicted energy consumption of the  $n$ 'th input data for data subset  $k$ .

$EC$  was the measured energy consumption of the  $n$ 'th input data for data subset  $k$

$N_k$  was the number of data points within each subset

$K$  was the number of subset,  $K = N$  for for *LOO*,  $N$  being the number data in the full data set

$\omega_{n_k}$  was the relative prediction error of the  $n$ 'th input data for data subset  $k$

$\omega_k$  was the average of the relative prediction errors for the  $k$ 'th subset

$\overline{\omega}_K$  was average of the relative prediction error.

**Results from Container ship #1** The data from Container ship #1 were used to make initial tests in order to determine the best combinations of the input variables. Table 2.6 shows the input variable combinations tested, with the cross validation errors and standard deviations listed in Table 2.7. From that the best prediction of the energy consumption (the lowest prediction error) was found by using the input combination 13, 14 and 21, all having relative prediction error around 4% they all had in common that they were based on noon reports and hindcasts, and that the time was included as a variable. Omitting the time increased the prediction errors of up to 2.4% (input combination 14 and 15). Using input data from the noon reports only, the best prediction error was found to be 4.9% with input variable combination 18, where all available noon report input, except the speed over ground, was used including the time variable. Without the time variable the



input combination 18 was identical to 17 which gave significant higher errors of 6.7%, for the noon reports data without time the rather simple input combination 9, including only the logged speed, sea water temperature and the mean draught, yields the best prediction of an error of 5.5%.

	Input combination ID	3	9	11	12	13	14	15	17	18	20	21
1	NR.Ulog	x	x	x	x	x	x	X	x	x	x	x
2	NR.Uobs				x	x	x	X			x	x
3	NR.Tsw	x	x	x	x	x	x	x	x	x	x	x
4	NR.Tm	x	x	x	x	x	x	x	x	x	x	x
5	NR.Trim	x		x	x	x	x	x	x	x	x	x
6	NR.True_wind_speed_m_s	x		x	x	x	x	x	x	x	x	x
7	NR.True_relative_wind _direction_deg	x		x	x	x	x	x	x	x	x	x
8	HC.Vrel				x	x					x	x
9	HC.gammarel				x	x					x	x
10	HC.mean.Hs				x	x	x	x			x	x
11	HC.mean.Tp				x	x	x	x			x	x
12	HC.var.Hs				x	x	x	x			x	x
13	HC.var.Tp				x	x	x	x			x	x
14	HC.var.Td				x	x	x	x			x	x
15	HC.var.Ws				x	x	x	x			x	x
16	HC.var.gamma				x	x	x	x			x	x
17	NR.UTC	<b>x</b>				<b>x</b>	<b>x</b>			<b>x</b>		<b>x</b>
18	HC.Ws						x	x				
19	HC.gamma						x	x				
27	NR.Sea_state_m								x	x	x	x
28	NR.True_relative_sea _direction_deg								x	x	x	x

Table 2.6: Input variable setup combination for Noon Report data of the containership data

Input variable setup	$\overline{\omega_K}$ %	$\overline{\sigma_K}$ %
3	5.81	6.11
9	5.45	6.76
11	8.01	9.59
12	5.40	5.22
13	3.96	3.99
14	4.00	3.96
15	6.43	6.21
17	7.64	7.81
18	4.92	5.45
20	4.52	5.05
21	3.98	4.03

Table 2.7: Relative cross validations errors predictive standard deviations.

Fig. 2.12 shows the relative prediction error with error bars based on the predicted standard deviation. It is seen how the error bars generally increase with high prediction errors or areas with sparse data density.

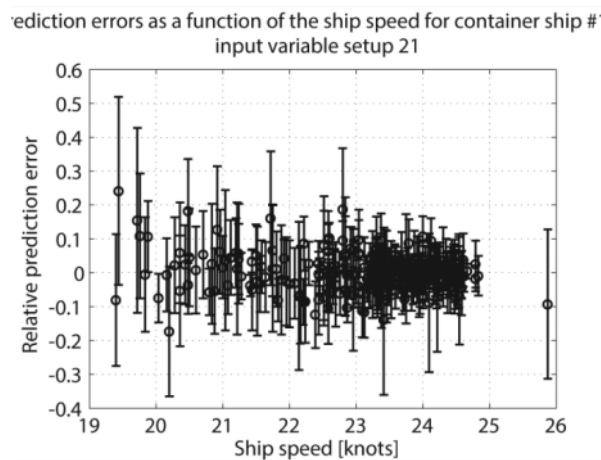


Figure 2.12: Relative prediction errors with error from the predicted standard deviation (input 21, 0-4.5 years)

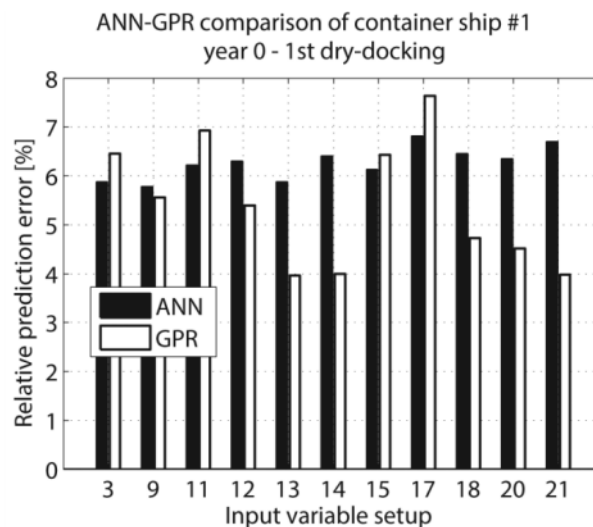


Figure 2.13: Comparison of prediction methods GPR and ANN)

In order to validate the prediction errors, an Artificial Neural Network (ANN) had been trained and tested with the same setups as the GPR. The ANN was a feed forward network as described in *Pedersen and Larsen (2009)*. Comparing the prediction methods in Fig. 2.13 showed that the cross validation errors for GPR in most cases were lower or in the same order of magnitude as ANN, was satisfactory for the further study of GPR with performance data.

**Results from Container ship #1-#5** Based on the findings of the previous sections, the remaining 4 container ships were trained and evaluated similarly as for container ship #1, i.e. using input variable combination 9, 18, 20 and 21, in order to represent data with and without hindcast data, and with and without time as a variable. Furthermore each of the data set was split into five subset: one for the first year, another for the first two years of data, a third one for the first three year and so forth until the first dry-docking.

Fig. 2.14 shows that for the data with hindcast input (20 and 21) were only affected very little by the increasing number of data and the ended up with prediction errors between 4 and 6%. For the data set using noon reports without the hindcast data there seemed to be a significant benefit using more data, since the prediction errors decreased with time for most of the data set.

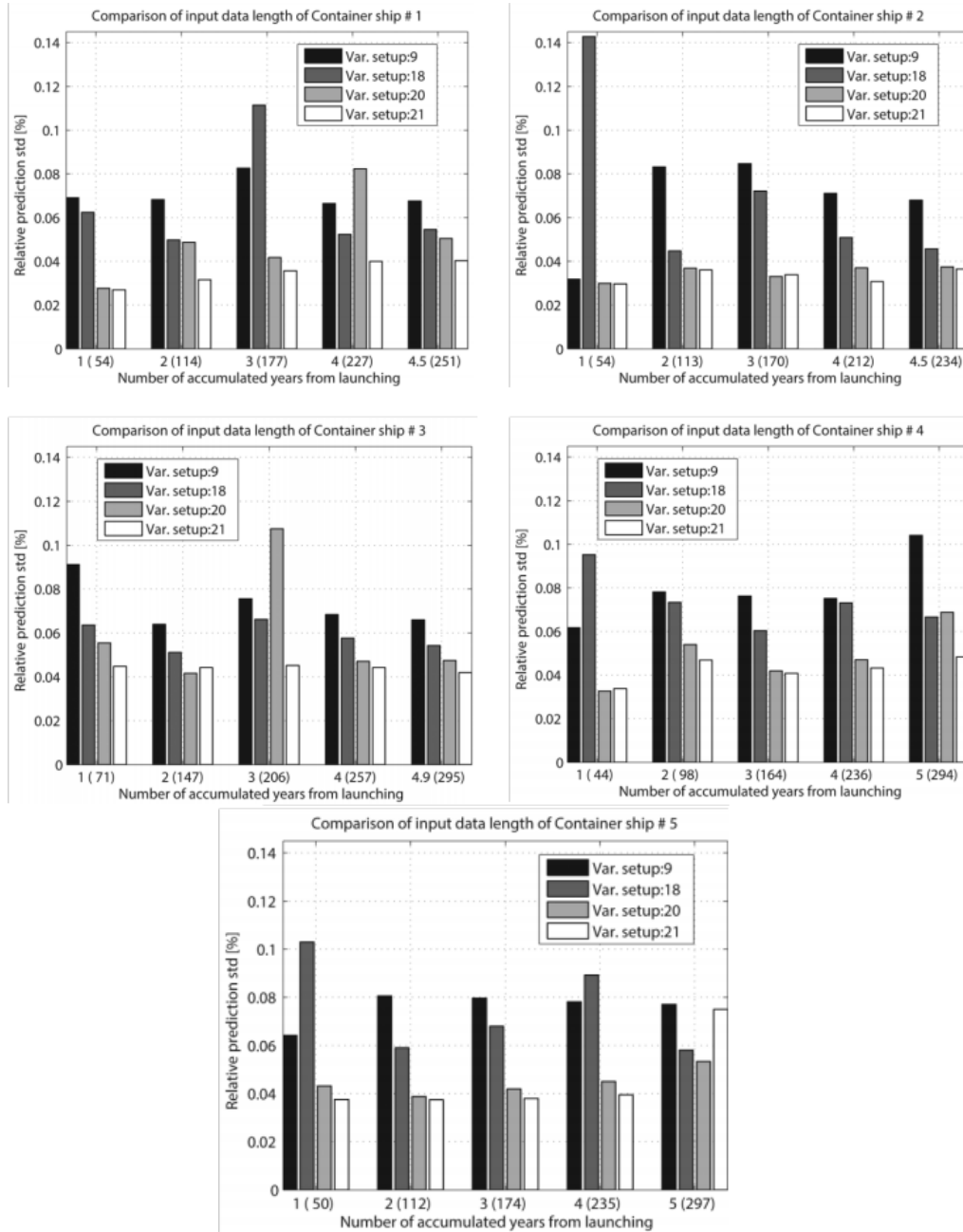
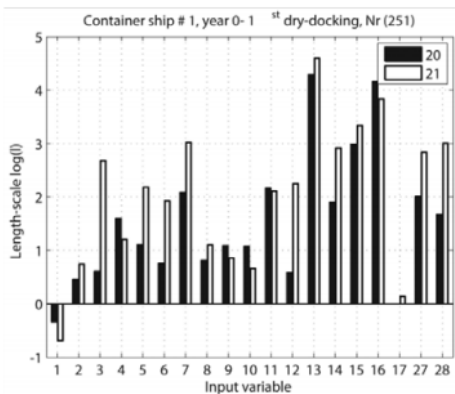


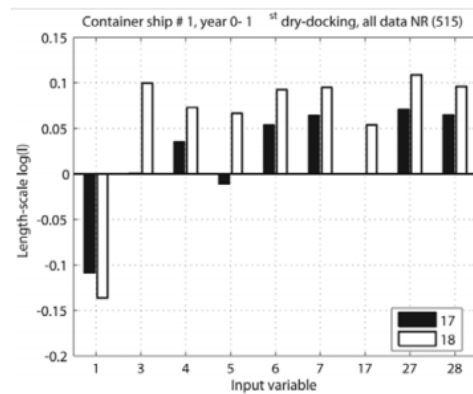
Figure 2.14: Cross-validation errors of container ship #1-#5

In order to evaluate the influence of each of the input variables, different trainings had been performed with Container ship #1. The length-scales for each of the input variables had then been evaluated for the different training setups. Initially the first year of container ship #1 data were used with all the relevant input variables including the hindcast data, see Table 2.7. In Fig. 2.15a the length-scales are illustrated in a bar diagram for the input variable setup where time was included in variable setup 21 and the one without time

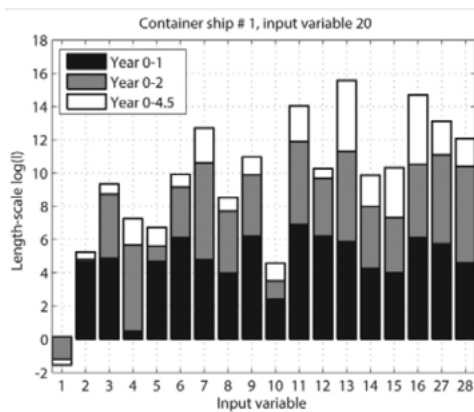
20 after training the data from the launching until the first dry-docking. Here all the available input variables were used, i.e. many data points were dismissed due to lack of hindcast availability. For both input 20 and 21 the figure shows a clear trend of the logged speed having the highest influence (*input1*). The speed over the ground (*input2*) is slightly higher together with hindcast wind speed and direction, and significant wave height have an equally influence. The hindcast wave period was generally less important. When the time (*input17*) was introduced in variable setup 21, it had a significant influence itself, but was also influenced the influence of other variables. The most dramatic drop between variable setup 20 and 21 was the seawater temperature (*input3*). The time was introduced as an estimator for the hull fouling since the propulsion performance was expected to drop over time of this. The significant influence of the time variable confirmed the strong relation between the expected to change in the performance over time. The draught and trim (*input4and5*) had a smaller impact than expected but this might be due to, only, small variation for these variables in the current data set.



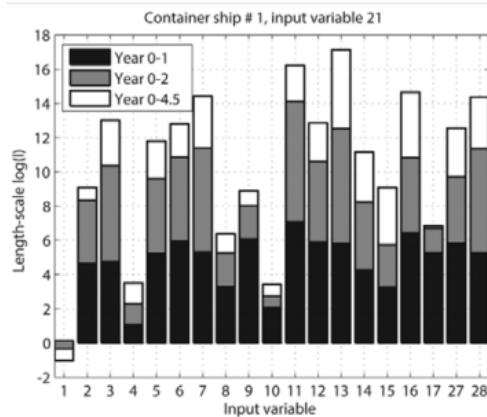
(a) Logarithmic length-scales of input variable setups 20 and 21 (with time as a variable)



(b) Logarithmic length-scales of input variable setups 17 and 18 (with time as a variable), using only noon report data.



(c) The accumulated length-scales of input variable setup 20 (without time), trained for the three periods of time 0-1, 0-2, 0-4.5 years (4.5 year is the time of the first dry docking).



(d) The accumulated length-scales of input variable setup 21 (including time – input variable 17), trained for the three periods of time 0-1, 0-2, 0-4.5 years (4.5 year is the time of the first dry docking).

Fig. 2.15b is presenting the length-scales based on the training of data with all noon report data available and no hindcast data, input variable setup 17 and 18, without and with the time as input variable. Again it was confirmed that the ship speed is the variable with the most significant influence with very small length-scales for both the input combi-

nations. For the variable setup 18, with time, the remaining input variables were all with the same order of magnitude. For input variable setup 17, the seawater temperature (input 3) and the trim (input 5) indicated to have more influence than the remaining variables. In order to assess the trend for shorter logging periods the model was also trained in data of 0 – 1 year, 0 – 2 years and 0-until the first dry-docking about 4.5 years. This had been performed for the variable setups 20 and 21 and the accumulated bars are shown in Fig. 2.15c and Fig. 2.15d. The figures shows that except for a few incidents the length-scales becomes shorter for longer data series.

The overall propulsion performance of a ship was expected to decrease over time mainly due to fouling of the hull and propeller. The change in performance can be determined by comparing the actual measured energy consumption  $EC$  propulsion power or fuel consumption with a calculated or predicted energy consumption  $\widehat{EC}$  of how the vessel should be able to perform.

With  $GPR$ ,  $\widehat{EC}$  predictions can be based on training of a previous period of time. The difference between the predicted and actual values, i.e. the prediction error, is thus a measure of the vessel propulsion performance.

In the analysis, the relative prediction error  $\omega = (\widehat{EC} - EC) / EC$  was used to evaluate the behaviour of the performance. The actual energy consumption  $EC$  was expected to increase over time due to the fouling, while  $\omega$  was expected to decrease since the predicted values were based on the training data which were assumed not to be affected by fouling. It was thus desirable to train on the shortest possible period of time in order to limit the effect of a trend in the training data. Yet the training set should include a reasonable variation in the input variables.

The training was performed on the entire training set and the testing on the remaining data set resulting in a predicted energy consumption  $\widehat{EC}$ . The change in propulsion performance due to fouling was assumed to be linear, and the relative prediction error  $\omega$  was estimated as a linear function of the time. In order to account for the predicted variance, a "weighted least square" regression was performed on  $\omega$  as a function of the time where the weights,  $\mathbf{w}$ , were the inverse relative predicted variance ( $1/\sigma_R^2$ ). This gave the prediction errors with large variance less influence on the linear regression model. As discussed previously the majority of the larger prediction errors also had a large standard deviation  $\sigma_R^2$ , so letting the inverse variance being the weights made these prediction errors less relevant.

Weighted least square regression was similar to normal least square, but with the residual ( $\omega_i - f(x, a)$ ) multiplied by the weights,  $\mathbf{w}_{ii}$  in the sum of square errors 2.2.14 which was minimized with respect to  $\alpha$ . The x-axis was represented by the time  $t_i$ .

$$E(\alpha) = \sum_{i=1}^n w_{ii} (\omega_i - f(t_i, \alpha))^2 \quad (2.2.14)$$

The function values of the performance trend could be define as a Vessel Performance Index, VPI:

$$VPI(t) = \alpha t + \beta \quad (2.2.15)$$

The performance trend was only continuous in periods without external disturbances that changed the propulsion performance. External disturbances could be known or unknown. The known disturbances were e.g. dry-docking and hull and propeller cleaning, and unknown disturbances could e.g. be sudden unknown damage of the propeller or rudder.

All known disturbances or events were available for the container ships and the trend was thus found between the known events, including dry-docking DD, hull cleaning HCL and propeller cleaning PCL for the container ship. All these events were expected to increase the relative prediction error  $\omega$ , because the energy consumption was expected to drop. But for the hull and propeller cleaning, this effect could sometimes be difficult to detect due to its limited impact on the relatively large data scatter. Therefore the trend detection had been performed both between all the known events and the dry-docking only.

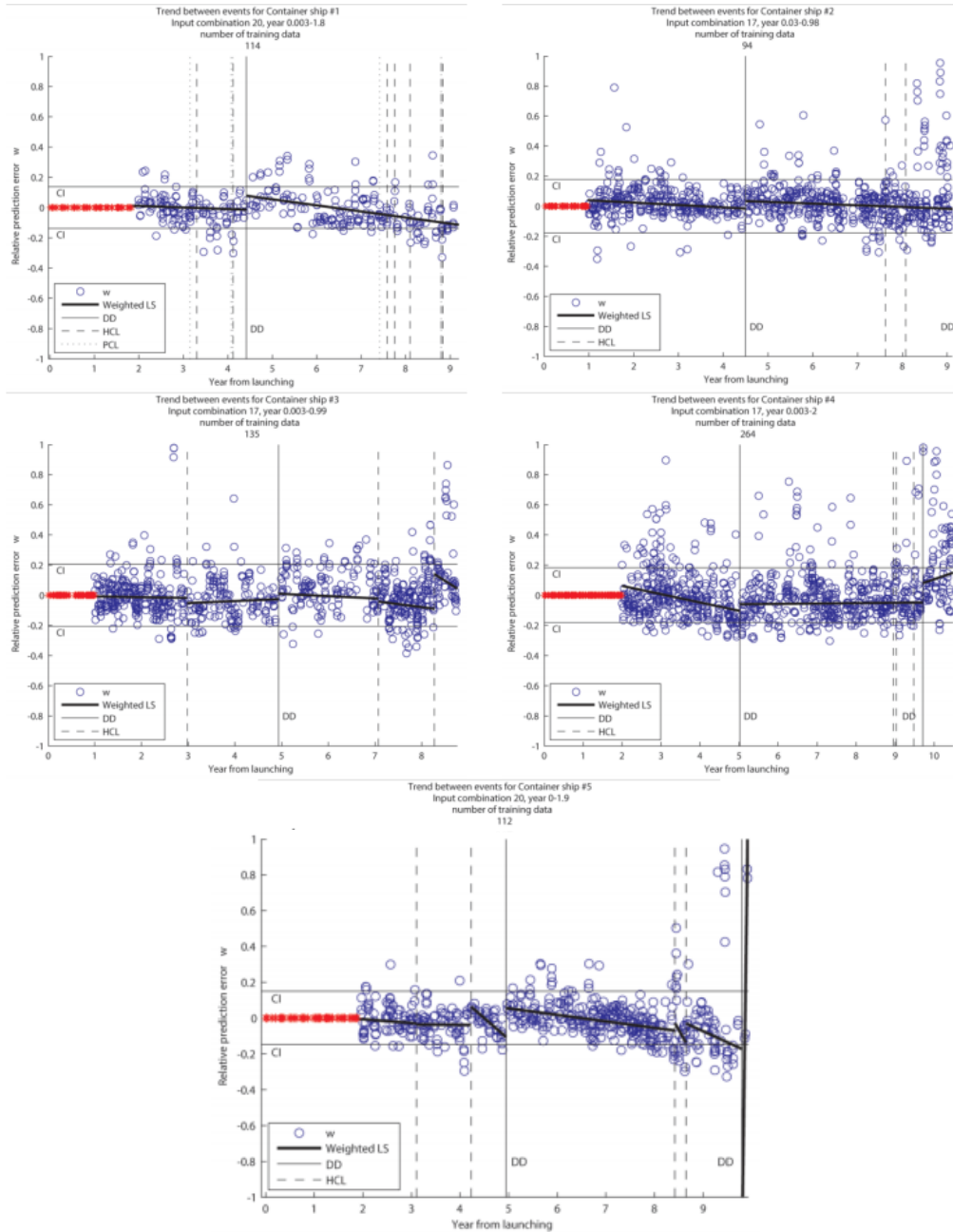


Figure 2.16: Performance trends for containership #1 – #5

The time variable in input variable setups 18 and 21 increased linearly with time. This means that if time had relevance for the regression model, it would be dominant for prediction far into the future. Input combinations including the time were thus inapplicable

for the trend detection. The best input variable setups without time was previously found to be 17 and 20, with 17 based exclusively on noon report data, and 20 including the hindcast data.

Given the considerations described above,  $\widehat{EC}$  for the five container ships was predicted based on two training periods, one and two years from launching, two input variable setups, 17 and 20. The trends were detected only between all the known events and the dry-dockings.

The performance trends of container ship #1 – #5 are illustrated in Fig. 2.16. For container ship #1 the intermediate events with short intervals gave too few data to make reasonable trends from and subsequently only the dry-docking was used, which gave a good picture of how the performance increased after a dry-docking, but afterwards dropped with a higher rate than before. Container ship #2 had fewer events and thus more continuous development where only a small increase in the performance was detected.

Container ship #3 had more events, but with reasonable time between and it thus became easier to develop trend for after the first hull cleaning there was a drop in the performance, indicating that the hull cleaning had actually had a negative effect on the propulsion performance. The following drydocking had a small positive effect.

Container ship #5 showed a very clear trend of the second hull-cleaning having a temporary significant positive effect, but the slope of the performance trend dropped dramatically, indicating that the anti fouling paint might had been damaged. The following dry-docking was bringing the vessel back to state which was actually better than at new.

With the methods described above, it was possible to detect the general trends of the change in the performance over time without any predefined definitions of the vessel. Using a combination of noon report data and hindcast data increased the prediction performance significant, even though it reduced the number of inputs.

The dry-docking, hull and propeller cleanings did not always have the intended effect. The hull cleanings may have a positive immediate effect, but the long-term trend can be negative. It might also be that the event has no immediate effect, but the long-term effect can be beneficial. In order to evaluate this, detailed information about the event is needed such as what part of the ship was cleaned and what equipment was used.

In general, the dry-docking had a more consistent effect with a positive change in VPI after the docking, but the slope afterwards varied from being steeper than before, as for Container ship #1, to being flatter than the previous trend, as for Container ship #4.

#### 2.2.4 Other related work

The rest of the related work, such as [5], [6], [7] to [17], that was either the same as the one presented above or was not taken into consideration for the data processing of the thesis and thus was not presented, can be found in the bibliography.





## Chapter 3

# Analysis of Ship Data

As stated before, the target of this thesis was to train a NN that would be able to predict the future fuel oil consumption using numerous variables. However in order to train any NN data are needed. In the case of this thesis the data were provided by the performance department of Bernhard Schulte shipmanagement Hellas.

### 3.1 Brief Introduction of Pandas, Jupyter Notebook & TensorFlow

The Pandas Library is a powerful data manipulation library that allows data engineers to visualize, filter and perform any actions needed on the data. It can better be explained in a few useful points

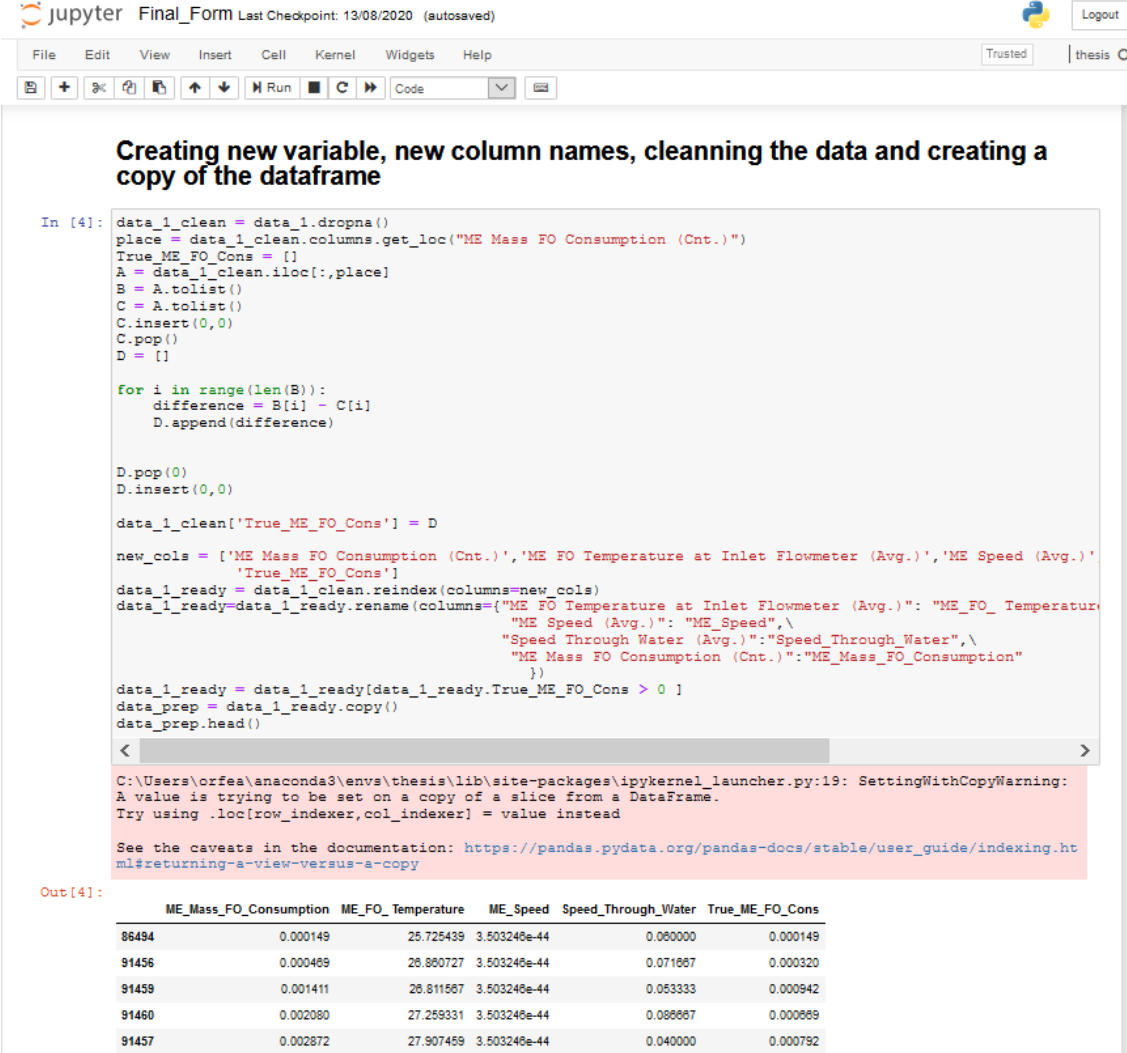
1. Pandas is a library that is used for data manipulation and analysis using powerful data structures.
2. The types of the data structures vary in Pandas, there can be Series 1-D (labelled homogeneous array, size immutable) DataFrames 2-D (labelled, size- mutable tabular structure with potentially heterogeneously typed columns) or Panels 3-D (labelled, size- mutable array.).
3. While the 1&2 -D are self-explanatory the Panel is a three-dimensional data structure with heterogeneous data. It is hard to represent the panel in graphical representation. But a panel can be illustrated as a container of DataFrame.
4. A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. Features of DataFrame: Potentially columns are of different types, Size – Mutable Labelled axes, (rows and columns), Can Perform Arithmetic operations.
5. In Pandas the data takes various forms like ndarray, series, map, lists, dictionaries, constants and also another DataFrame.

Pandas also provides some useful built-in functions. Some of the most common ones are :index, columns, dtype etc. to name a few. For more information the reader is encouraged to visit the pandas website [Pandas](#)

The TensorFlow is a subclassing API, created by Google, that provides a define-by-run interface for advanced research and especially Machine Learning research. Moreover, TensorFlow's high-level APIs are based on the Keras API standard for defining and training

neural networks. Keras enables fast prototyping, state-of-the-art research and production. For more information the reader is encouraged to visit the TensorFlow website [TensorFlow](#).

The Jupyter Notebook is an open-source web application that allows the user, as seen in the figure 3.1, to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more. For more information the reader is encouraged to visit the Jupyter Notebook website [Jupyter Notebook](#).



The screenshot shows a Jupyter Notebook interface with the title "Final\_Form" and a last checkpoint of "13/08/2020 (autosaved)". The notebook contains a code cell with the following Python code:

```
In [4]: data_1_clean = data_1.dropna()
place = data_1_clean.columns.get_loc("ME Mass FO Consumption (Cnt.)")
True_ME_FO_Cons = []
A = data_1_clean.iloc[:,place]
B = A.tolist()
C = A.tolist()
C.insert(0,0)
C.pop()
D = []

for i in range(len(B)):
    difference = B[i] - C[i]
    D.append(difference)

D.pop(0)
D.insert(0,0)

data_1_clean['True_ME_FO_Cons'] = D

new_cols = ['ME Mass FO Consumption (Cnt.)', 'ME FO Temperature at Inlet Flowmeter (Avg.)', 'ME Speed (Avg.)',
            'True_ME_FO_Cons']
data_1_ready = data_1_clean.reindex(columns=new_cols)
data_1_ready=data_1_ready.rename(columns={"ME FO Temperature at Inlet Flowmeter (Avg.)": "ME_FO_Temperature",
                                         "ME Speed (Avg.)": "ME_Speed",\
                                         "Speed Through Water (Avg.)": "Speed_Through_Water",\
                                         "ME Mass FO Consumption (Cnt.)": "ME_Mass_FO_Consumption"
                                         })
data_1_ready = data_1_ready[data_1_ready.True_ME_FO_Cons > 0 ]
data_prep = data_1_ready.copy()
data_prep.head()
```

The output of the code cell is a table with 6 columns: ME\_Mass\_FO\_Consumption, ME\_FO\_Temperature, ME\_Speed, Speed\_Through\_Water, and True\_ME\_FO\_Cons. The table shows 5 rows of data:

	ME_Mass_FO_Consumption	ME_FO_Temperature	ME_Speed	Speed_Through_Water	True_ME_FO_Cons
86494	0.000149	25.725439	3.503246e-44	0.060000	0.000149
91456	0.000469	26.860727	3.503246e-44	0.071667	0.000320
91459	0.001411	26.811567	3.503246e-44	0.053333	0.000942
91460	0.002080	27.259331	3.503246e-44	0.086667	0.000669
91457	0.002872	27.907459	3.503246e-44	0.040000	0.000792

Figure 3.1: Jupyter Notebook demo

## 3.2 Data Analysis

### 3.2.1 Data Acquisition

As mentioned in the previous chapter the variables that were needed to carry out a good predictive model were:

1. The Fuel Oil Consumption(FOC) [*kg or m<sup>3</sup>*]
2. The Ship's Speed Through Water (STW) [*kn*]
3. The Main Engine Shaft Power [*kW*]
4. The Main Engine Speed or RPM
5. The Fuel Oil Temperature  $^{\circ}C$
6. The True Wind Speed at Anemometer Height
7. The True Wind Direction at Anemometer Height

The data were collected from three containerships, the names and specific details of which will remain anonymous for confidential reasons.

Two of the ships were considered to be sister ships. Sister ships are called the ships that share the same design, same Main and Auxiliary Engines, same DWT and operate with approximately the same Service Speeds. According to the performance department two of the three ships shared these traits and were qualified to be sister ships.

The variables that were provided from the three ships were the following:

1. Id,
2. Vessel\_Object\_Id, (first\_ship 864911, second\_ship 864637, third\_ship 629311)
3. Entry\_Date,
4. ME\_Speed (Avg.),
5. ME\_Shaft\_Power (Avg.),
6. True\_Wind\_Speed\_at\_Anemometer\_Height (Avg.),
7. True\_Wind\_Direction\_at\_Anemometer\_Height (Avg.),
8. Speed\_Through\_Water (Avg.),
9. ME\_Mass\_FO\_Consumption (Cnt.),
10. ME\_Volume\_FO\_Consumption (Cnt.),
11. ME\_FO\_Temperature\_at\_Inlet\_Flowmeter (Avg.)

The amount of data acquired was, approximately, 1 million for each variable and  $\frac{1}{3}$  for each ship respectively. The Vessel\_Object\_Id was the ship's code and was used to match that values of the variables to each ship. The sister ship's Vessel\_Object\_Id began with the same two digits (86). The Entry\_Date was given with a minute accuracy and was used to sort the data of each ship to prepare them for the timeseries. The ME\_Speed (Avg.) was the average value of the ME's rpm in 1 minute interval. The same applied for the ME\_Shaft\_Power (Avg.), True\_Wind\_Speed\_at\_Anemometer\_Height (Avg.),

True\_Wind\_Direction\_at\_Anemometer\_Height (Avg.), ME\_FO\_Temperature\_at\_Inlet\_Flowmeter (Avg.) and the Speed\_Through\_Water (Avg.). The ME\_Mass\_FO\_Consumption and the ME\_Volume\_FO\_Consumption (Cnt.) on the other hand represented the total Mass and Volume (respectively) measured in that 1 minute interval.

### 3.2.2 Data Preprocessing

The data that were acquired, were measured using a variate of measurements. As a result the data contained a lot of noise and thus needed to be filtered. First of all, the data were uploaded using the pandas framework where each variable was in a separate column. Then, all the NaN data were deleted as it was considered that using the mean value would result in inconsistencies and ruining the ability of the ANN to train properly.

After the removal of the NaN values, the remaining data were split into 3 datasets, one for each ship, using the Vessel\_Object\_Id to separate them. The next step was to sort the variables using the Entry\_Date, which was the only one with no missing values.

Next, a new variable was created. This was the True\_ME\_FO\_Cons which was created with the following method. The first value of the ME Mass FO Consumption remained the same however every next value was created by subtracting from the ME\_Mass\_FO\_Consumption<sub>*i*</sub> value the ME\_Mass\_FO\_Consumption<sub>*i-1*</sub>:

$$True\_ME\_FO\_Cons_0 = ME\_Mass\_FO\_Consumption_0 \quad (3.2.1)$$

for  $i > 0$ :

$$True\_ME\_FO\_Cons_i = ME\_Mass\_FO\_Consumption_i - ME\_Mass\_FO\_Consumption_{i-1} \quad (3.2.2)$$

This new variable was created with the perspective to filter the data even more. Since the ME\_Mass\_FO\_Consumption would be the main variable to be used in this project, the newly created variable would help as an indicator of when there is any fuel oil consumption. This was a very important filter since despite the fact that all of the ships spent some time in a harbour the data were still recorded. As a result a huge chunk of the data did not have any fuel oil consumption rendering it useless for this thesis task. Moreover, the new variable also filtered the negative fuel oil consumption which was of course wrong and needed to either be removed or to be replaced by some median value. In this thesis it was considered that replacing it a median value would not provide any extra aid so it the data that had a negative fuel oil consumption were removed.

Next, the following variables were removed from the :

1. Id,
2. Vessel\_Object\_Id,864911,864637,629311
3. Entry\_Date,
4. ME\_Shaft\_Power (Avg.),
5. True\_Wind\_Speed\_at\_Anemometer\_Height (Avg.),
6. True\_Wind\_Direction\_at\_Anemometer\_Height (Avg.),
7. ME\_Volume\_FO\_Consumption (Cnt.),

The reason the ME\_Mass\_FO\_Consumption was kept instead of the ME\_Volume\_FO\_Consumption because of the better representation of the consumption due to different density of the fuels that were probably used. At this point it is worth to see the difference in the data before and after the preprocessing that are presented in the figures 3.3 & 3.4 below.

As it is clearly notice the fuel oil consumption in the figure 3.4 does not have any straight lines meaning that the 0 values had been successfully removed. Also the differences in the two figures arise from the removal of the NAN values and the 0FOC values.

The next feature that was used was the correlation, functions of pandas framework were used and the results are shown in the figure 3.2 below. From these results it was clear that for a model to successfully generalise more than one variable had to be used.

ME_Mass_FO_Consumption	1.0000	0.1986	0.0653	0.0776	0.0069
ME_FO_Temperature	0.1986	1.0000	0.4320	0.4606	0.0048
ME_Speed	0.0653	0.4320	1.0000	0.9717	0.0084
Speed_Through_Water	0.0776	0.4606	0.9717	1.0000	0.0079
True_ME_FO_Cons	0.0069	0.0048	0.0084	0.0079	1.0000
	ME_Mass_FO_Consumption	ME_FO_Temperature	ME_Speed	Speed_Through_Water	True_ME_FO_Cons

Figure 3.2: Correlation between Fuel Oil Consumption and the other variables

The last step of the preprocessing was to scale the data. This was accomplished by using the built in MinMaxScaler function of sklearn from the preprocessing library. This function-transformation transforms the data into new data that had a specified Max and Min values. The transformation is given by the the following equations :

$$X_{std} = (X - X.min(axis = 0)) / (X.max(axis = 0) - X.min(axis = 0)) \quad (3.2.3)$$

$$X_{scaled} = X_{std} * (max - min) + min \quad (3.2.4)$$

where min and max are the desired values.

In this project the min value was chosen to be 0 and the max value 1. As a result all the data would be between the values of 0,1 which would accelerate the computation in the following steps. Since the data were filtered to be above **zero**, in the previous steps, the transformation is just a division with the max value of each variable respectively. These values, for each variable, were saved since at the final part of processing the data would need to be scaled back to the original value in order to interpret the results. These values were the following :

$$[0.00013181, 0.01113438, 0.00400021, 0.04059073, 0.00237815]$$

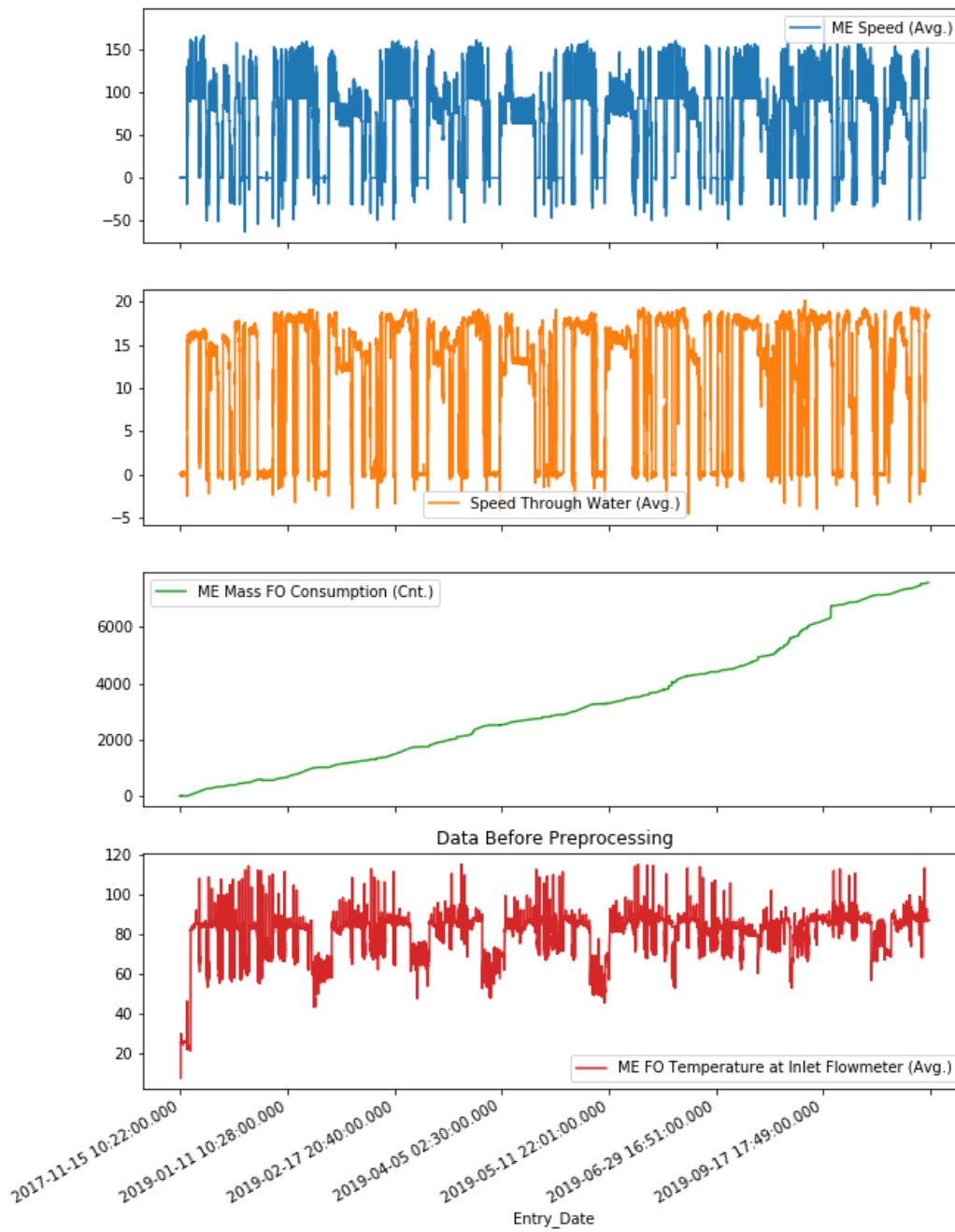


Figure 3.3: The data before preprocessing

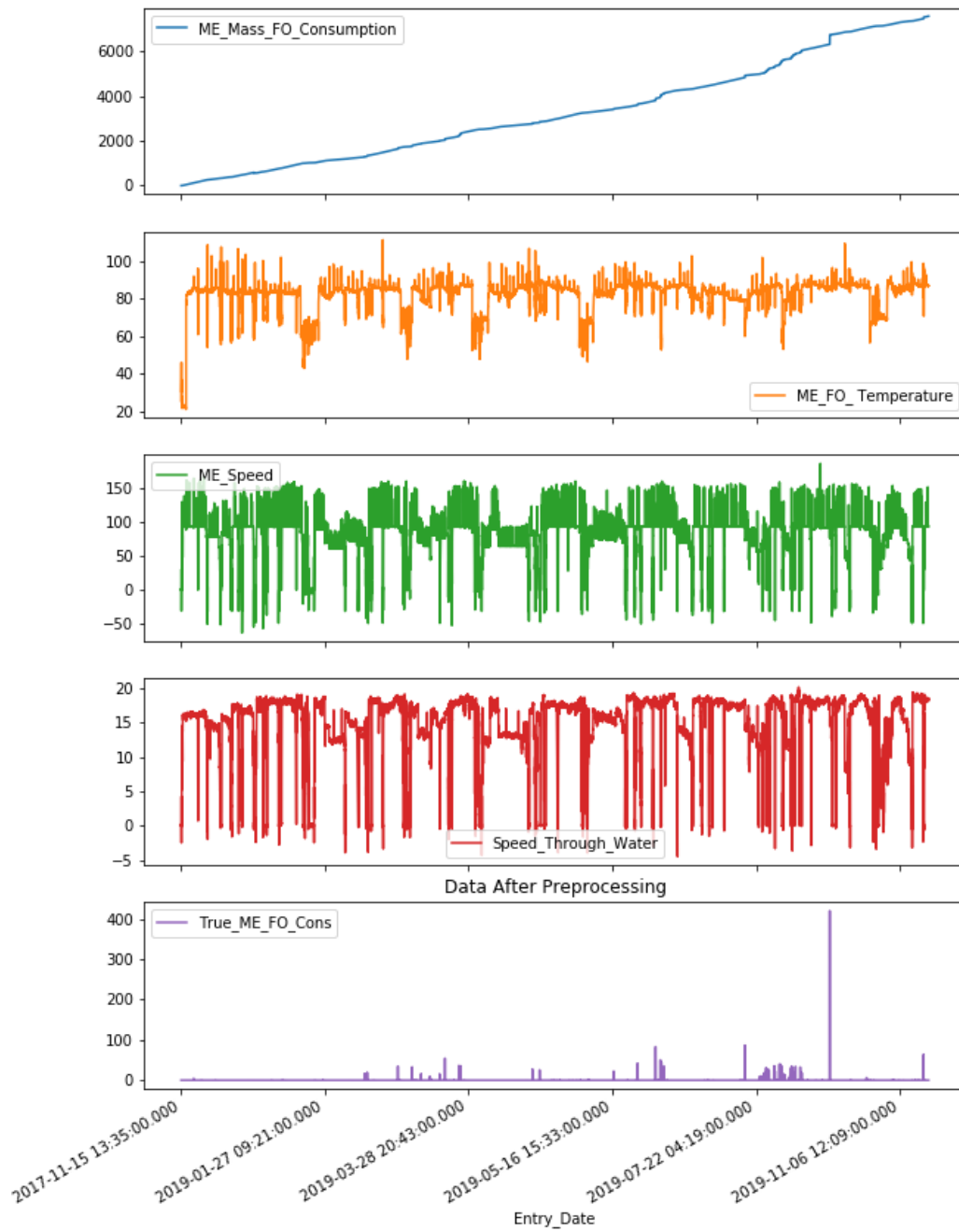


Figure 3.4: The data after preprocessing





## Chapter 4

# Neural Network Design

### 4.1 First approach

As we have seen before the data were in chronological order which meant we were dealing with time-series data. The best way to tackle problems that are consisting of this type of data, is to use a recurrent neural network. The reason is pretty simple. Imagine a ball in one frame. If you were asked where is the ball going to be in the next frame you could not decide. However if you were also given the previous position of the ball and more frames of the previous positions of the ball you probably would be able to accurately predict its position in the next frame or even in the next several frames. The same reasoning is applied in time-series data. Since the data are connected by the time parameter one can understand that the estimation of the fuel oil consumption value is not affected only from the present values of the speed, main engine's rpms and temperature but also from their previous values. Moreover this approach allows us to use the previous values of the fuel oil consumption as a variable that will help us make the prediction. To be more precise the previous values of the fuel oil consumption were the main variable of this thesis.

### 4.2 Simple RNN [4]

The keras backend provides the option to use the simpleRNN layer to tackle time-series problems. The RNN is similar to the feedforward neural network, with the exception that it also has connections pointing backwards. The simplest possible RNN is shown in the figure 4.1 below. This neural network consists of only one neuron. This one neuron receives inputs, produces the outputs and then feeds the output back to itself as shown in the figure 4.1. At each time step  $t$ , which is also called a frame, this recurrent neuron receives the inputs  $\mathbf{x}_{(t)}$  as well as its own output from the previous time step  $\mathbf{y}_{(t-1)}$ . Since there is no previous output for  $t = 0$  the "previous output" is set to 0 for this frame only. We can represent this tiny network against the time axis, as shown in figure 4.1. This is called unrolling the network through time (it's the same recurrent neuron represented once per time step).

A layer of recurrent neurons can easily be created. At each time step  $t$ , every neuron receives both the input vector  $\mathbf{x}_{(t)}$  and the output vector from the previous time step  $\mathbf{y}_{(t-1)}$ , as shown in figure 4.2. Note that both the inputs and outputs are vectors now (when there was just a single neuron, the output was a scalar).

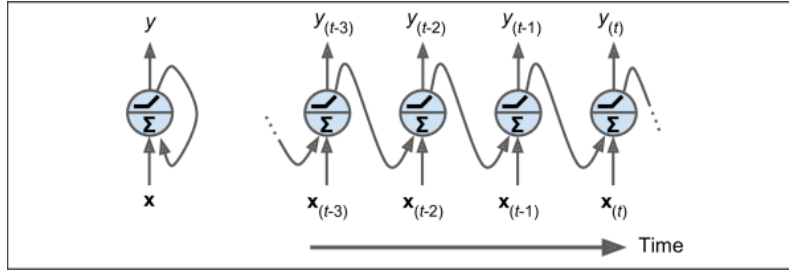


Figure 4.1: Simplest RNN (left) unrolled through time (right)

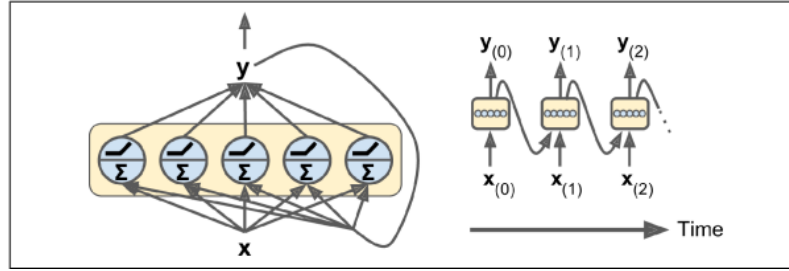


Figure 4.2: A layer of recurrent neurons (left) unrolled through time (right)

Each recurrent neuron has two sets of weights: one for the inputs  $\mathbf{x}_{(t)}$  and the other for the outputs of the previous time step,  $\mathbf{y}_{(t-1)}$ . Let's call these weight vectors  $\mathbf{w}_x$  and  $\mathbf{w}_y$ . If we consider the whole recurrent layer instead of just one recurrent neuron, we can place all the weight vectors in two weight matrices,  $\mathbf{W}_x$  and  $\mathbf{W}_y$ . The output vector of the whole recurrent layer can then be computed pretty much as you might expect, as shown in Equation 4.2.1 (b is the bias vector and  $\phi(\cdot)$  is the activation function).

$$\mathbf{y}(t) = \phi \left( \mathbf{W}_x^T \mathbf{x}_{(t)} + \mathbf{W}_y^T \mathbf{y}_{(t-1)} + b \right) \quad (4.2.1)$$

Just as with feedforward neural networks, we can compute a recurrent layer's output in one shot for a whole mini-batch by placing all the inputs at time step t in an input matrix  $\mathbf{X}_{(t)}$  (see equation 4.2.2).

$$\begin{aligned} \mathbf{Y}(t) &= \phi \left( \mathbf{X}_{(t)} \mathbf{W}_x + \mathbf{Y}_{(t-1)} \mathbf{W}_y + b \right) \\ &= \phi \left( \begin{bmatrix} \mathbf{X}_{(t)} & \mathbf{Y}_{(t-1)} \end{bmatrix} \mathbf{W} + b \right) \text{ with } \mathbf{W} = \begin{bmatrix} \mathbf{W}_x \\ \mathbf{W}_y \end{bmatrix} \end{aligned} \quad (4.2.2)$$

In this equation:

- $\mathbf{Y}_{(t)}$  is an  $m \times n_{neurons}$  matrix containing the layer's outputs at time step t for each instance in the mini-batch ( $m$  is the number of instances in the mini-batch and  $n_{neurons}$  is the number of neurons).
- $\mathbf{X}_{(t)}$  is an  $m \times n_{inputs}$  matrix containing the inputs for all instances ( $n_{inputs}$  is the number of input features).
- $\mathbf{W}_x$  is an  $n_{inputs} \times n_{neurons}$  matrix containing the connection weights for the inputs of the current time step.
- $\mathbf{W}_y$  is an  $n_{neurons} \times n_{neurons}$  matrix containing the connection weights for the outputs of the previous time step.

- $\mathbf{b}$  is a vector of size  $n_{neurons}$  containing each neuron's bias term.
- The weight matrices  $\mathbf{W}_x$  and  $\mathbf{W}_y$  are often concatenated vertically into a single weight matrix  $\mathbf{W}$  of shape  $(n_{inputs} + n_{neurons}) \times n_{neurons}$ .
- The notation  $[\mathbf{X}_{(t)} \mathbf{Y}_{(t-1)}]$  represents the horizontal concatenation of the matrices  $\mathbf{X}_{(t)}$  and  $\mathbf{Y}_{(t-1)}$ .

Notice that  $\mathbf{Y}_{(t)}$  is a function of  $\mathbf{X}_{(t)}$  and  $\mathbf{Y}_{(t-1)}$ , which is a function of  $\mathbf{X}_{(t-1)}$  and  $\mathbf{Y}_{(t-2)}$ , which is a function of  $\mathbf{X}_{(t-2)}$  and  $\mathbf{Y}_{(t-3)}$ , and so on. This makes  $\mathbf{Y}_{(t)}$  a function of all the inputs since time  $t = 0$  (that is,  $\mathbf{X}_{(0)}, \mathbf{X}_{(1)}, \dots, \mathbf{X}_{(t)}$ ). At the first time step,  $t = 0$ , there are no previous outputs, so they are typically assumed to be all zeros.

Since the output of a recurrent neuron at time step  $t$  is a function of all the inputs from previous time steps, you could say it has a form of *memory*. A part of a neural network that preserves some state across time steps is called *a memory cell* (or simply a *cell*). A single recurrent neuron, or a layer of recurrent neurons, is a very basic cell, capable of learning only short patterns (typically about 10 steps long, but this varies depending on the task).

In general a cell's state at time step  $t$ , denoted  $\mathbf{h}_{(t)}$  (the "h" stands for "hidden"), is a function of some inputs at that time step and its state at the previous time step:  $\mathbf{h}_{(t)} = f(\mathbf{h}_{(t-1)}, \mathbf{x}_{(t)})$ . Its output at time step  $t$ , denoted  $\mathbf{y}_{(t)}$ , is also a function of the previous state and the current inputs. In the case of the basic cells we have discussed so far, the output is simply equal to the state, but in more complex cells this is not always the case, as shown in figure 4.3.

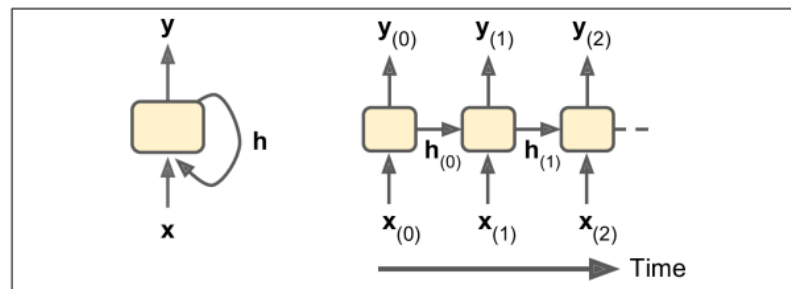


Figure 4.3: A cell's hidden state and its output may be different

A RNN can simultaneously take a sequence of inputs and produce a sequence of outputs (see the top-left network in figure 4.4). This type of *sequence – to – sequencenetwork* is useful for predicting time series such as stock prices: you feed it the prices over the last  $N$  days, and it must output the prices shifted by one day into the future (i.e., from  $N - 1$  days ago to tomorrow).

Alternatively, you could feed the network a sequence of inputs and ignore all outputs except for the last one (see the top-right network in figure 4.4). In other words, this is a *sequence – to – vectornetwork*. For example, you could feed the network a sequence of words corresponding to a movie review, and the network would output a sentiment score (e.g., from  $-1[\text{hate}]$  to  $+1[\text{love}]$ ).

Conversely, you could feed the network the same input vector over and over again at each time step and let it output a sequence (see the bottom-left network of figure 4.4). This is a *vector – to – sequencenetwork*. For example, the input could be an image (or the output of a CNN), and the output could be a caption for that image.

Lastly, you could have a *sequence – to – vectornetwork*, called an encoder, followed by a *vector – to – sequencenetwork*, called a decoder (see the bottom-right network of

figure 4.4). For example, this could be used for translating a sentence from one language to another. You would feed the network a sentence in one language, the encoder would convert this sentence into a single vector representation, and then the decoder would decode this vector into a sentence in another language. This two-step model, called an Encoder–Decoder, works much better than trying to translate on the fly with a single *sequence – to – sequence* RNN (like the one represented at the top left): the last words of a sentence can affect the first words of the translation, so you need to wait until you have seen the whole sentence before translating it.

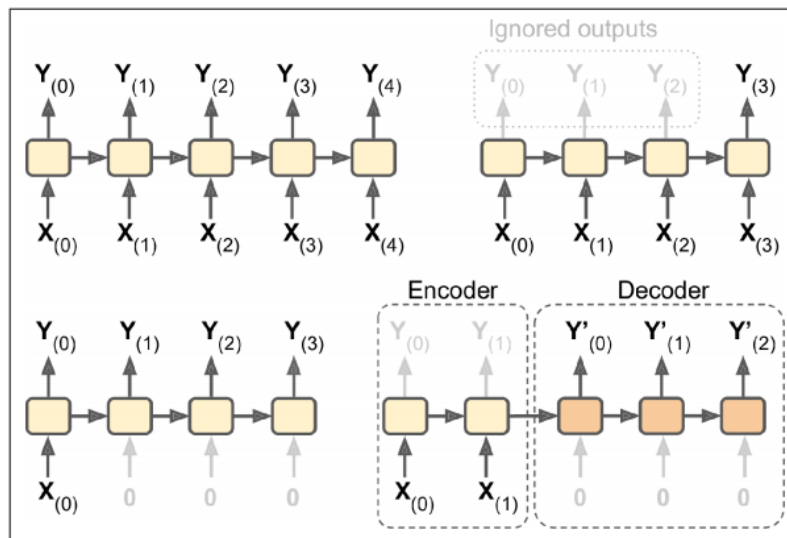


Figure 4.4: Seq-to-seq (top left), seq-to-vector (top right), vector-to-seq (bottom left), and Encoder–Decoder (bottom right) networks

In this thesis, a *sequence – to – sequence network* was used since the main purpose was to predict the future fuel oil consumption using the previous fuel oil consumption, as well as other variables, as inputs.

To train an RNN, the trick is to unroll it through time (like we just did) and then simply use regular backpropagation (see figure 4.5). This strategy is called backpropagation through time (BPTT).

Just like in regular backpropagation, there is a first forward pass through the unrolled network (represented by the dashed arrows). Then the output sequence is evaluated using a cost function  $C(Y_{(0)}, Y_{(1)}, \dots, Y_{(T)})$  (where  $T$  is the max time step). Note that this cost function may ignore some outputs, as shown in figure 4.5 (for example, in a sequence-to-vector RNN, all outputs are ignored except for the very last one). The gradients of that cost function are then propagated backward through the unrolled network (represented by the solid arrows). Finally the model parameters are updated using the gradients computed during BPTT. Note that the gradients flow backward through all the outputs used by the cost function, not just through the final output (for example, in figure 4.5 the cost function is computed using the last three outputs of the network,  $Y_{(2)}$ ,  $Y_{(3)}$ , and  $Y_{(4)}$ , so gradients flow through these three outputs, but not through  $Y_{(0)}$  and  $Y_{(1)}$ ). Moreover, since the same parameters  $W$  and  $b$  are used at each time step, backpropagation will do the right thing and sum over all time steps.

There are two problems with the simple RNN. The first one is the unstable gradients and the second one is the "Short Memory". The first problem is mostly tackled with the use of normalized layers and dropout which is a technique that a random subset of all the neurons is deactivated or else dropped out at each training iteration. In this thesis it was

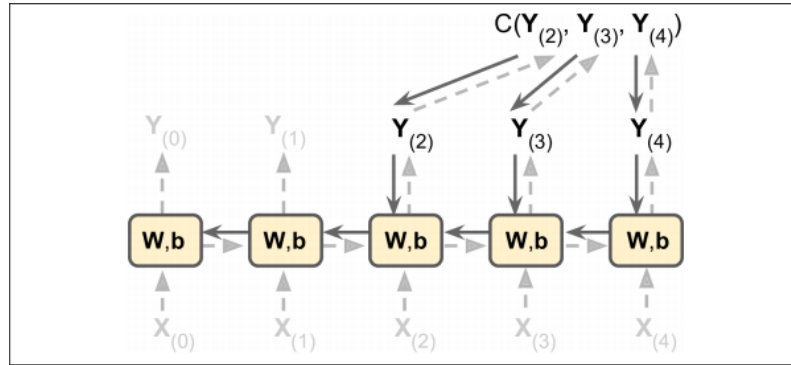


Figure 4.5: Backpropagation through time

noticed that the implementation of these methods made little to no improvement.

However, the "Short Memory" problem was the one that needed to be solved for this thesis since the data for each ship were approximately  $300k$ .

### 4.3 LSTM [4]

One of the few ways to tackle this problem is the use of a LSTM layer instead of a simpleRNN one. The initials(LSTM) stand for Long-Short Term Memory.

The Long Short-Term Memory (LSTM) cell was proposed in 1997 by Sepp Hochreiter and Jürgen Schmidhuber [4] and gradually improved over the years by several researchers.

LSTM cell as a black box, it can be used very much like a basic cell, except it will perform much better since training will converge faster, and it will detect long-term dependencies in the data. So let's see how an LSTM cell works and what its architecture looks like.

If you don't look at what's inside the box, the LSTM cell looks exactly like a regular cell, except that its state is split into two vectors:  $\mathbf{h}_{(t)}$  and  $\mathbf{c}_{(t)}$  ("c" stands for "cell"). You can think of  $\mathbf{h}_{(t)}$  as the short-term state and  $\mathbf{c}_{(t)}$  as the long-term state. So let's open the box. The architecture of the LSTM cell is presented in the figure 4.6 below.

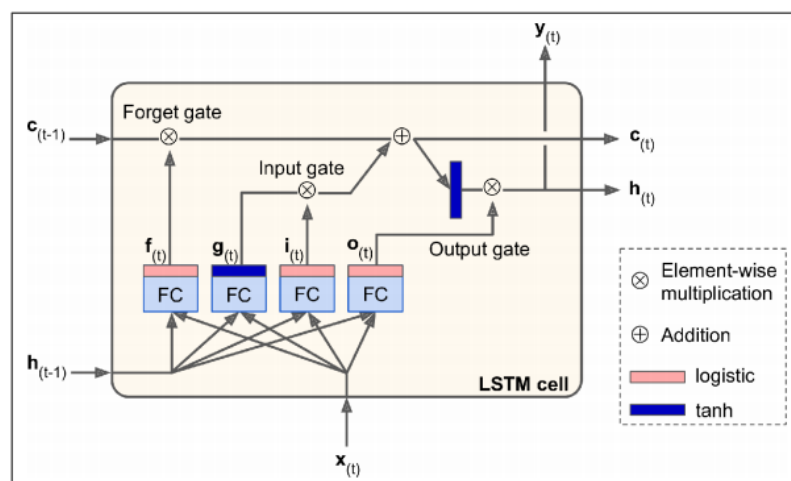


Figure 4.6: LSTM cell

The key idea is that the network can learn what to store in the long-term state, what to throw away, and what to read from it. As the long-term state  $\mathbf{c}_{(t-1)}$  traverses the network

from left to right, you can see that it first goes through a *forget gate*, dropping some memories and then it adds some new memories via the addition operation (which adds the memories that were selected by an *input gate*). The result  $\mathbf{c}_{(t)}$  is sent straight out, without any further transformation. So, at each time step, some memories are dropped and some memories are added. Moreover, after the addition operation, the long-term state is copied and passed through the *tanh* function, and then the result is filtered by the *output gate*. This produces the short-term state  $\mathbf{h}_{(t)}$  (which is equal to the cell's output for this time step,  $\mathbf{y}_{(t)}$ ). Now let's look at where new memories come from and how the gates work.

First, the current input vector  $\mathbf{x}_{(t)}$  and the previous short-term state  $\mathbf{h}_{(t-1)}$  are fed to four different fully connected layers. They all serve a different purpose:

- The main layer is the one that outputs  $\mathbf{g}_{(t)}$ . It has the usual role of analysing the current inputs  $\mathbf{x}_{(t)}$  and the previous (short-term) state  $\mathbf{h}_{(t-1)}$ . In a basic cell, there is nothing other than this layer, and its output goes straight out to  $\mathbf{y}_{(t)}$  and  $\mathbf{h}_{(t)}$ . In contrast, in an LSTM cell this layer's output does not go straight out, but instead its most important parts are stored in the long-term state (and the rest is dropped).
- The three other layers are *gate controllers*. Since they use the logistic activation function, their outputs range from 0 to 1. As you can see, their outputs are fed to element-wise multiplication operations, so if they output 0s they close the gate, and if they output 1s they open it. Specifically:
  - The forget gate (controlled by  $\mathbf{f}_{(t)}$ ) controls which parts of the long-term state should be erased.
  - The input gate (controlled by  $\mathbf{i}_{(t)}$ ) controls which parts of  $\mathbf{g}_{(t)}$  should be added to the long-term state.
  - Finally, the output gate (controlled by  $\mathbf{o}_{(t)}$ ) controls which parts of the long term state should be read and output at this time step, both to  $\mathbf{h}_{(t)}$  and to  $\mathbf{y}_{(t)}$ .

In short, an LSTM cell learns to recognize an important input (that's the role of the input gate), stores it in the long-term state, preserves it for as long as it is needed (that's the role of the forget gate), and extracts it whenever it is needed. This explains why these cells have been amazingly successful at capturing long-term patterns in time series, long texts, audio recordings, and more.

The equation 4.3.1 summarizes how to compute the cell's long-term state, its short-term state, and its output at each time step for a single instance.

$$\begin{aligned}
 \mathbf{i}_{(t)} &= \sigma(\mathbf{W}_{xi}^T \mathbf{x}_{(t)} + \mathbf{W}_{hi} \mathbf{h}_{(t-1)} + \mathbf{b}_i) \\
 \mathbf{f}_{(t)} &= \sigma(\mathbf{W}_{xf}^T \mathbf{x}_{(t)} + \mathbf{W}_{hf} \mathbf{h}_{(t-1)} + \mathbf{b}_f) \\
 \mathbf{o}_{(t)} &= \sigma(\mathbf{W}_{xo}^T \mathbf{x}_{(t)} + \mathbf{W}_{ho} \mathbf{h}_{(t-1)} + \mathbf{b}_o) \\
 \mathbf{g}_{(t)} &= \tanh(\mathbf{W}_{xg}^T \mathbf{x}_{(t)} + \mathbf{W}_{hg} \mathbf{h}_{(t-1)} + \mathbf{b}_g) \\
 \mathbf{c}_{(t)} &= \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)} \\
 \mathbf{y}_{(t)} &= \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh(\mathbf{c}_{(t)})
 \end{aligned} \tag{4.3.1}$$

In this equation:

- $\mathbf{W}_{(xi)}$ ,  $\mathbf{W}_{(xf)}$ ,  $\mathbf{W}_{(xo)}$ ,  $\mathbf{W}_{(xg)}$  are the weight matrices of each of the four layers for their connection to the input vector  $\mathbf{x}_{(t)}$ .

- $\mathbf{W}_{(hi)}$ ,  $\mathbf{W}_{(hf)}$ ,  $\mathbf{W}_{(xo)}$ , and  $\mathbf{W}_{(hg)}$  are the weight matrices of each of the four layers for their connection to the previous short-term state  $\mathbf{h}_{(t-1)}$ .
- $\mathbf{b}_i$ ,  $\mathbf{b}_f$ ,  $\mathbf{b}_o$ , and  $\mathbf{b}_g$  are the bias terms for each of the four layers. Note that TensorFlow initializes  $\mathbf{b}_f$  to a vector full of 1s instead of 0s. This prevents forgetting everything at the beginning of training.

Besides all mentioned above, LSTM cells can be modified to run on the GPU instead of the CPU, making them even faster as they utilize the parallel processing capabilities of the GPU.

## 4.4 The final approach

As it was said before the aim of this thesis was to accurately predict the fuel oil consumption one minute in the future, using different types of inputs. In order to achieve this goal a recurrent neural network was chosen. From the previous paragraphs it was clear that the LSTM cell was the one to be preferred. It must be said that a simple RNN was also used but did not bear any fruits and will not be presented in this thesis.

Before the LSTM was created the first ship's dataset was split into two parts one used for training and one used for testing. Since the dataset had approximately 300k values for each variable, 90% of it was used for training and the other 10% was saved for the testing set. The datasets of the second and third ship would entirely be used for testing the ability of the ship to generalize in a sister ship and in a ship of the same type respectively.

The first approach was to build an one LSTM layer with one input and of course one output. In the figure 4.7 there is a representation of the model's structure. This model used only the previous FOC values as input and tried to predict the FOC value of the next minute.

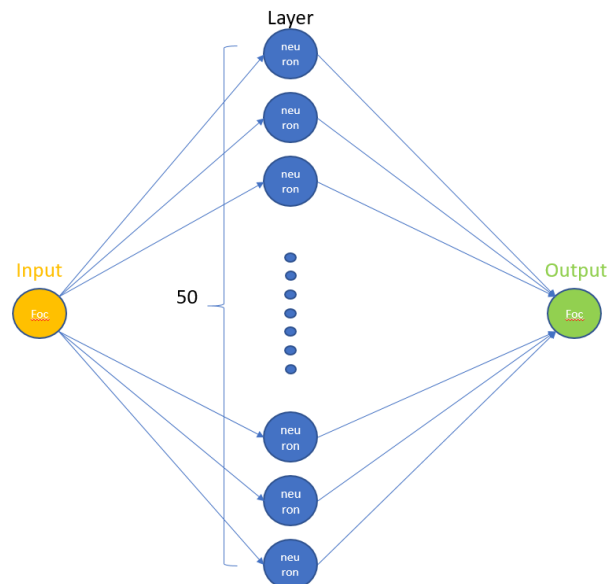


Figure 4.7: A NN with one input(FOC) one layer with 50 neuros and one output(FOC)

The reason 50 neurons were chosen was arbitrary, however most papers suggest the use of 32 for simple problems with little data and few variables.

This architecture, as well as the next ones, was tried in different models. To be more specific, there are two key "parameters" that needed to be tweaked. The first one was the look-back parameter or time-steps. This parameter represented the amount of past data that the model would see in order to determine the next value and the values that were chosen for this variable were [1 5 10 15 20 50] .

The second parameter was the epochs. The term epoch refers to one cycle through the full training dataset. The number of epochs plays an important role in the performance of the model not only on the testing set but also on new data. In other words the amount of epochs directly impact the model's ability to regularize. If a model is trained on a dataset for few epochs it might fail to find the patterns and thus fail to make accurate predictions in the testing set. On the other hand if a model is let to train on a dataset for a lot of epochs it can reach a point where it has memorize the whole training set. As a result it might not be able to make good predictions for the testing set but will most likely fail to make an accurate prediction on new data. This phenomenon is call overfitting in machine learning. This is the reason why each NN was trained for [5 10 15 20 30 50] . This way it was made possible to control if the model was overfitting.

After the training was finished(for each value of the epoch parameter) the model predicted the remaining 10% of the first ship's dataset and the full dataset for the second and third. The result were plotted in comparison of the real values for each ship respectively. Moreover the % difference was plotted for each ship. This was very important since the measured value of the last part of the datasets was in the order of 7000 kg a discrepancy of 200 kg would not be easily spotted or would not appeared to be so big in the first plot. This amount of an error would however not be acceptable in the first ship.



# Chapter 5

## Results

In this chapter the best results for each combination of variable and look-back will be presented. Look-back is the parameter that indicates the number of values per time step. In the table 5.1 a brief summary of the results is being presented.

No. Variables	Variables	N.N. Type	Best	Lookbacks					
				1	5	10	15	20	50
1	F.O.C.	L.S.T.M.	Model in	15	30	50	5	50	5
2	F.O.C.	L.S.T.M.	terms of	30	20	20	15	30	10
	F.O.T.								
3	F.O.C.	L.S.T.M.	Epochs	5	30	50	10	30	50
	F.O.T.								
	M.E.RPM								

Table 5.1: Brief summary of the results

### 5.1 One variable : Fuel Oil Consumption

In this part the best results for each value of the look-back parameter will be presented. Moreover these are models that used only the FOC variable as input and only one Layer, more combinations of these parameters will be presented below.

#### 5.1.1 1 Lookback

The best results for one lookback using only one 50 neuron LSTM layer were produced when the model was trained for 10, 15 and 30 epochs for the first ship and for 15 epochs for the second and third ship. So overall the best results were produced when the model was trained for only 15 epochs.

In the figures below the following conclusions have been reached:

- First of all the model succeeded to train incredibly well for the first ship, with an error almost 0%.
- Secondly, the model succeeded to generalize in sister ship level with an error below 0.5%.
- Thirdly, the model succeeded to generalize in type ship level with an error below 1% for the first part but failed to keep it below that in the last third of the dataset.

- Last but not least, the spikes that can be clearly seen in the 3 percentages plots happened for the following reasons:
  - As it was discussed in the preprocessing section, some of the values were removed. As a result, there were some sudden changes in the values of the FOC and are traced when the graph creates a step.
  - Moreover, as the time progresses fouling is created around the ship increasing the resistance of the ship and as a result increasing the consumption of the ship "disrupting the pattern".

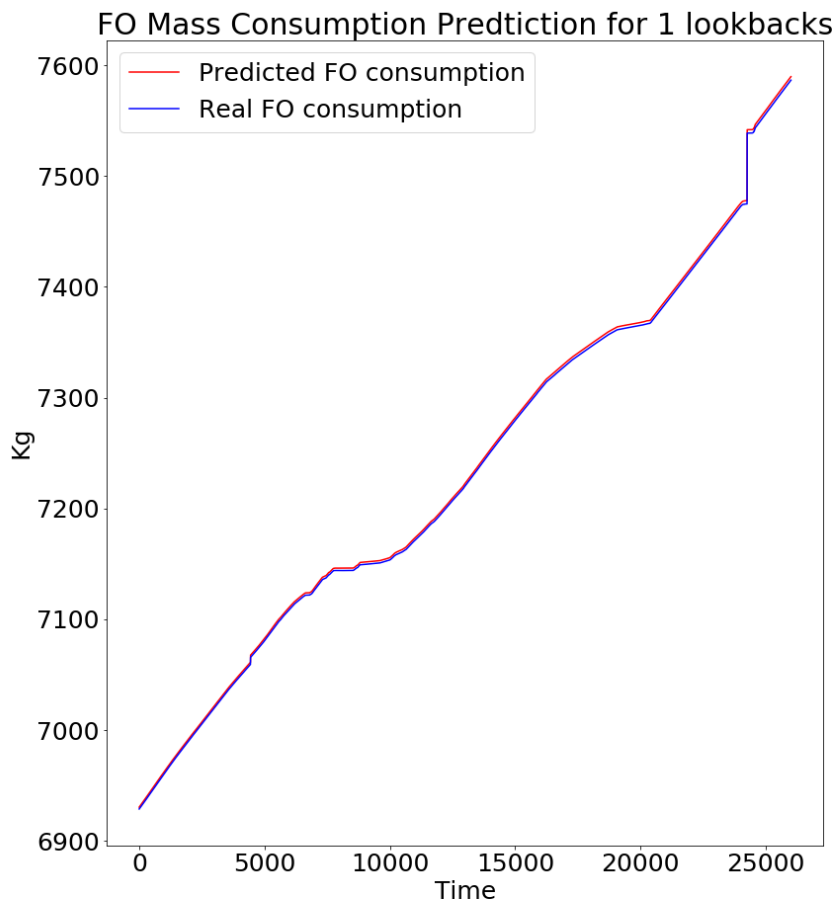


Figure 5.1: Results of the *first* ship after 15 epochs for 1 lookback

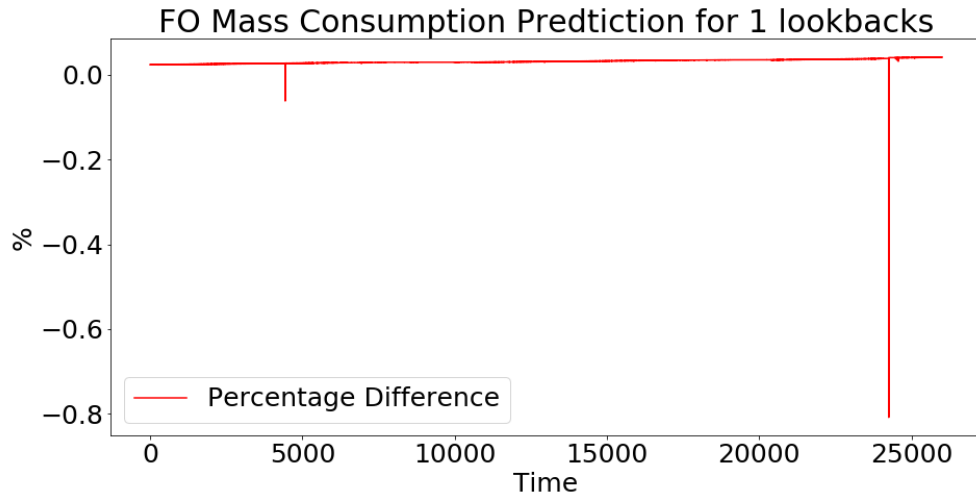


Figure 5.2: Results of the % difference of the *first* ship after 15 epochs for 1 lookback

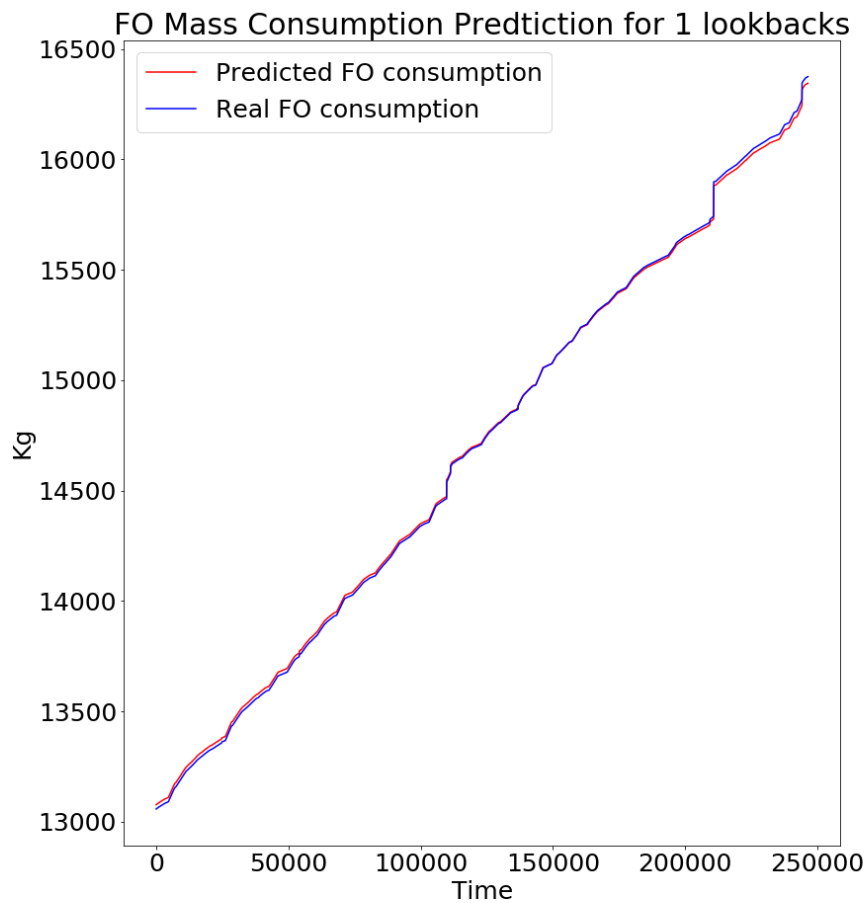


Figure 5.3: Results of the *second* ship after 15 epochs for 1 lookback

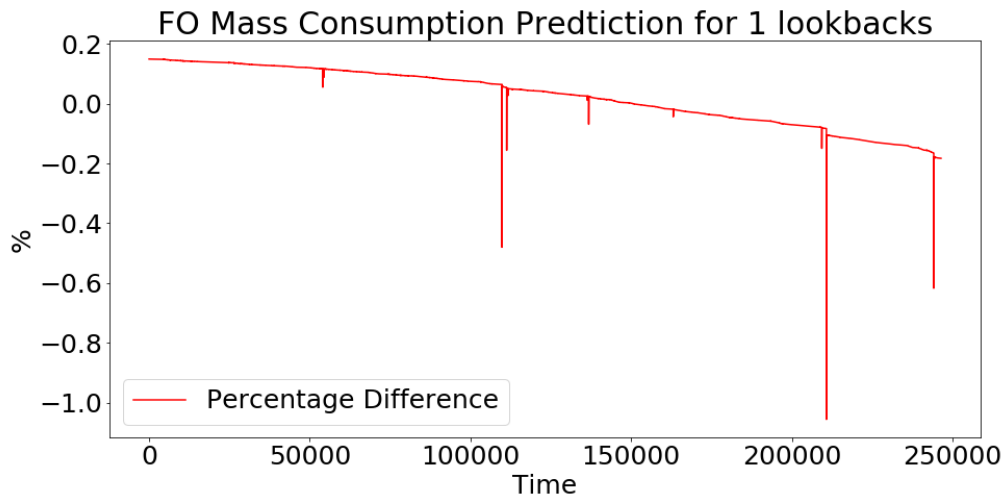


Figure 5.4: Results of the % difference of the *second* ship after 15 epochs for 1 lookback

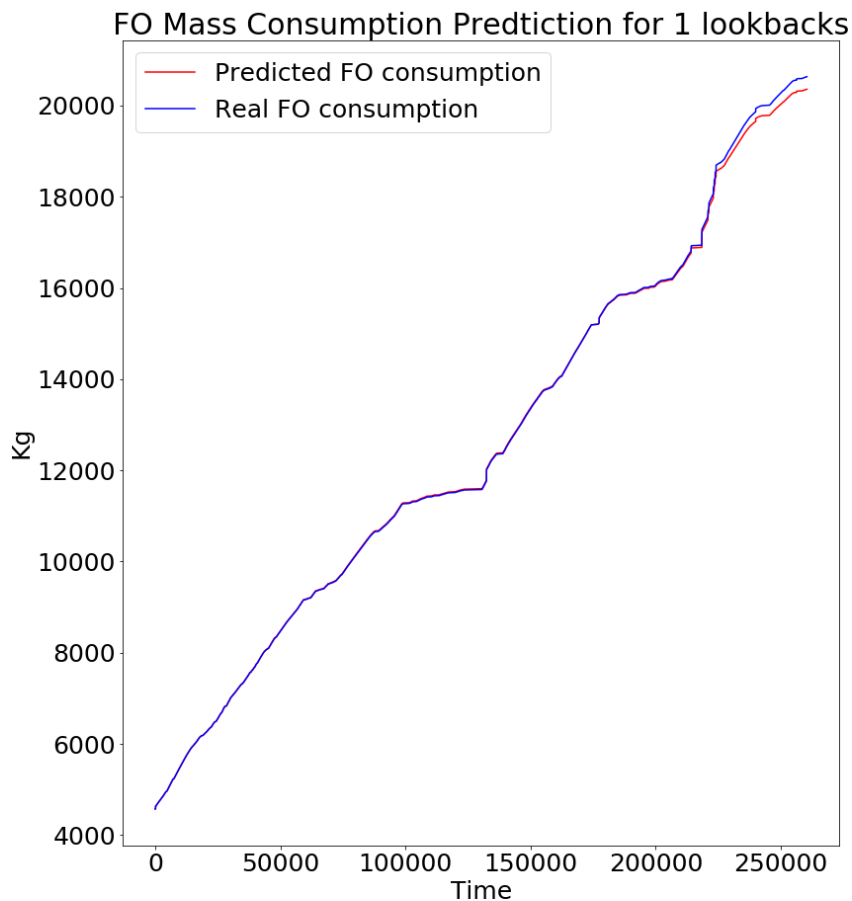


Figure 5.5: Results of the *third* ship after 15 epochs for 1 lookback

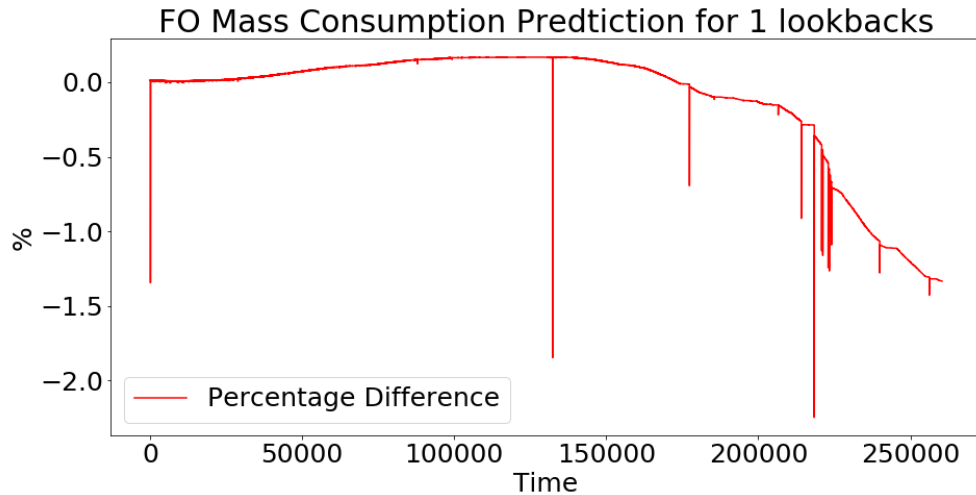


Figure 5.6: Results of the % difference of the *third* ship after 15 epochs for 1 lookback

### 5.1.2 5 Lookbacks

The best results for 5 lookbacks using only one 50 neuron LSTM layer were produced when the model was trained for 5, 20 and 30 epochs for the first ship and for 10 and 30 epochs for the second and third ship. So overall the best results were produced when the model was trained for 30 epochs.

From the figures below the following conclusions have been reached:

- First of all, as in the 1 lookback the model succeeded to train incredibly well for the first ship, with an error almost 0%.
- However, it failed to generalize so well for the second ship(sister ship) and for the third.
- Moreover, the spikes appear in the same places as before.

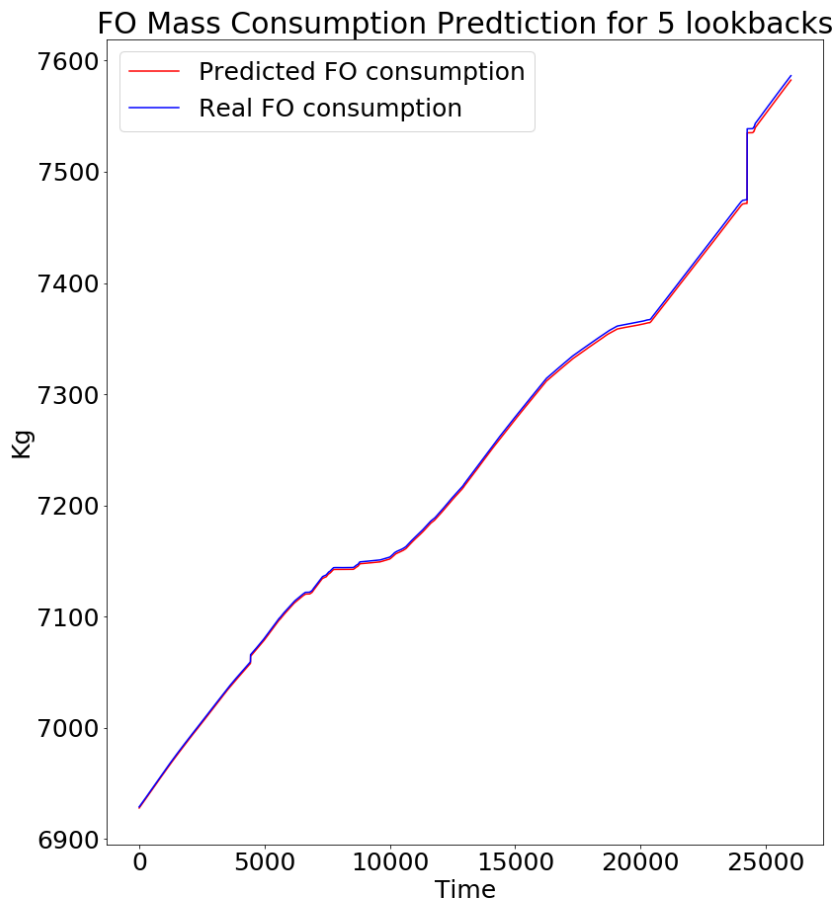


Figure 5.7: Results of the *first* ship after 5 epochs for 1 lookback

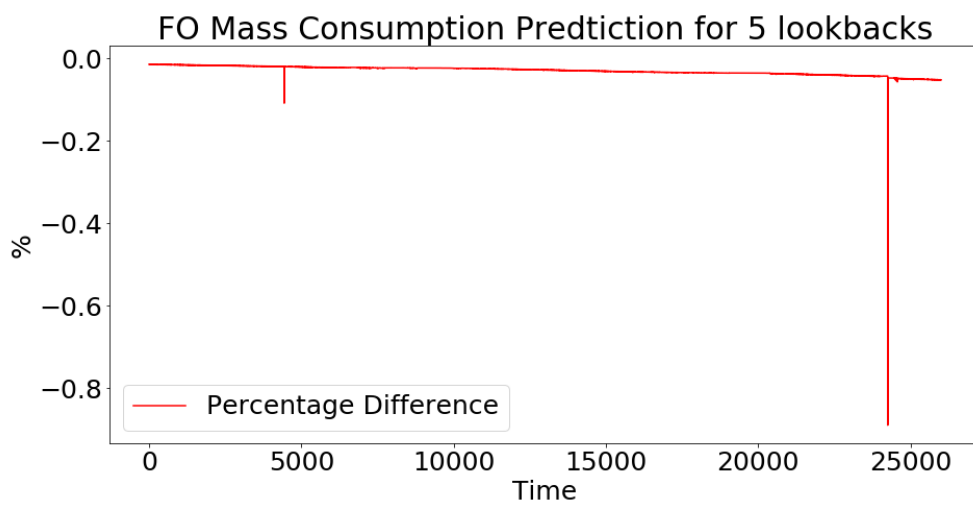


Figure 5.8: Results of the % difference of the *first* ship after 30 epochs for 5 lookback

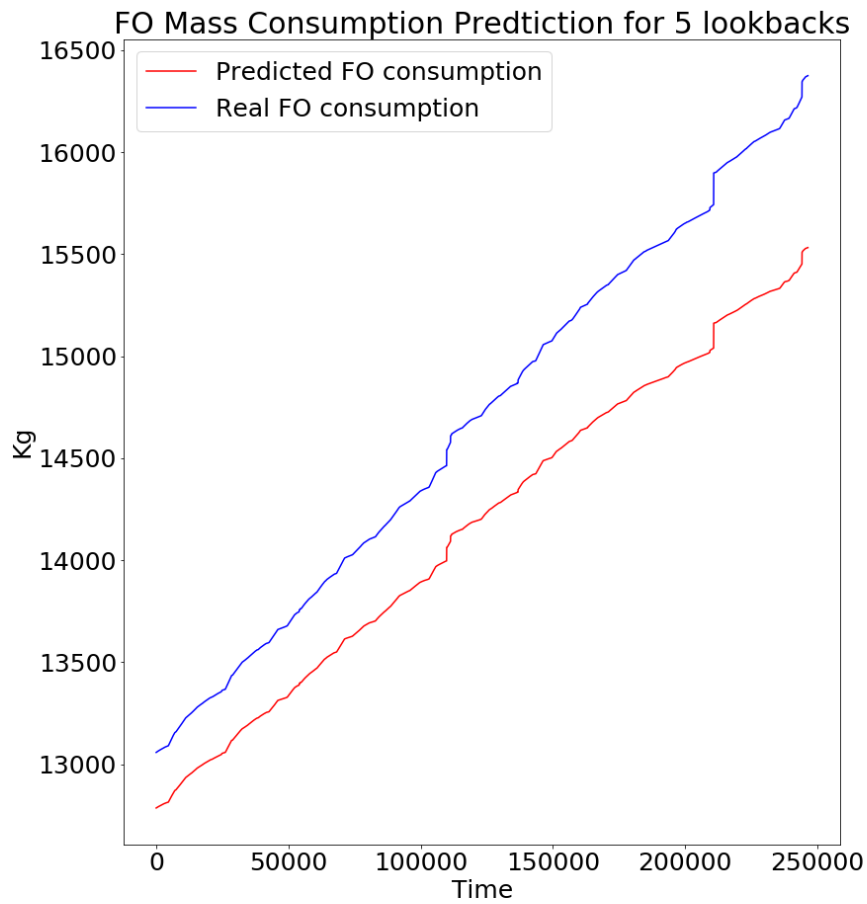


Figure 5.9: Results of the *second* ship after 5 epochs for 1 lookback

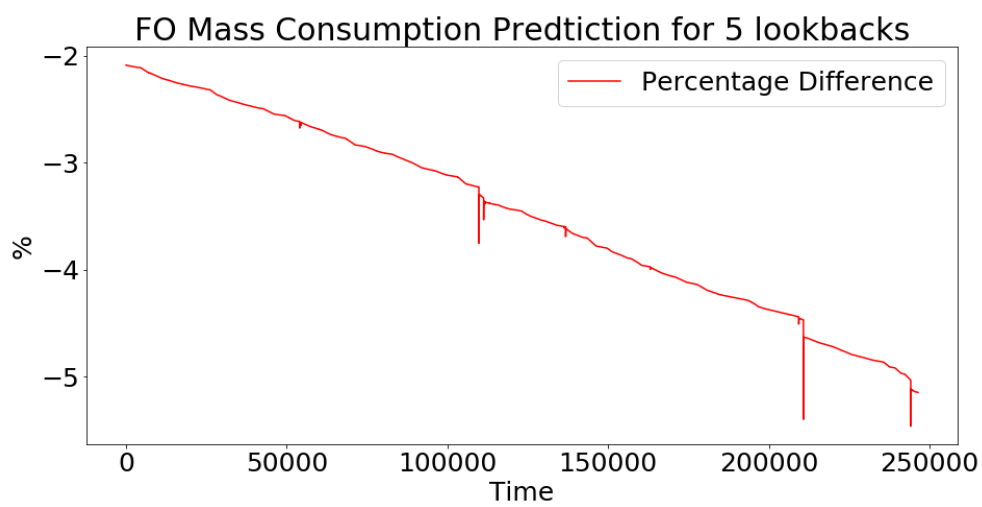


Figure 5.10: Results of the % difference of the *second* ship after 30 epochs for 5 lookback

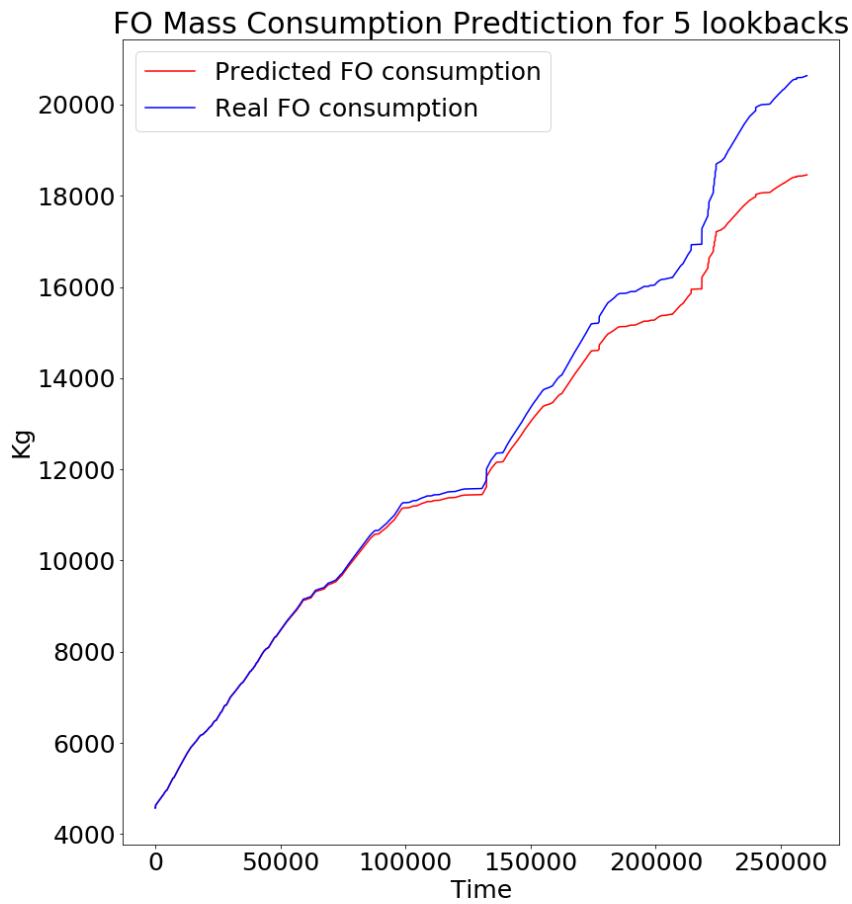


Figure 5.11: Results of the *third* ship after 5 epochs for 1 lookback

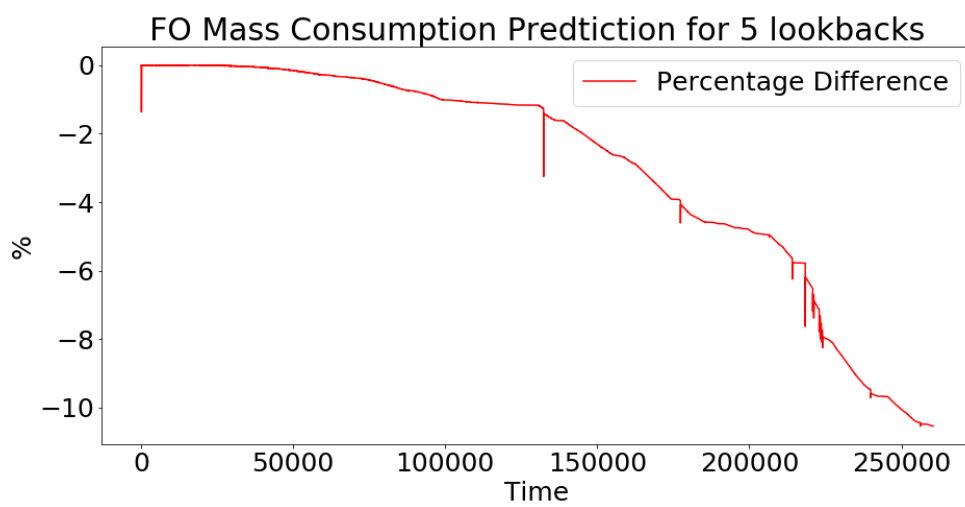


Figure 5.12: Results of the % difference of the *third* ship after 30 epochs for 5 lookback



### 5.1.3 10 Lookbacks

The best results for 10 lookbacks using only one 50 neuron LSTM layer were produced when the model was trained for 10, 20, 30 and 50 epochs for the first ship and for 30 and 50 epochs for the second and 50 epochs for the third ship. So overall the best results were produced when the model was trained for 50 epochs.

From the figures below the following conclusions have been reached:

- Once again the model was able to train very well for the first ship, with an error almost 0%.
- The model's results for the second and the third ship were satisfactory, since it appears the model was able to generalize but not as well as for 1 lookback.
- Once again the spikes appeared to be in the same places as before.

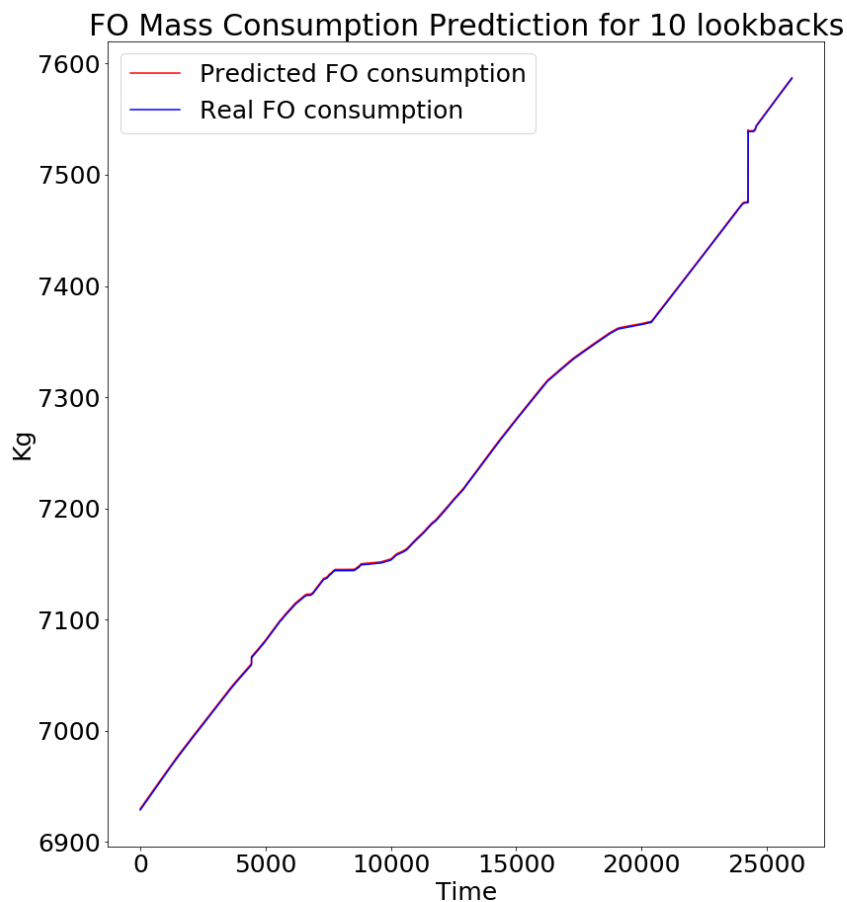


Figure 5.13: Results of the *first* ship after 50 epochs for 1 lookback

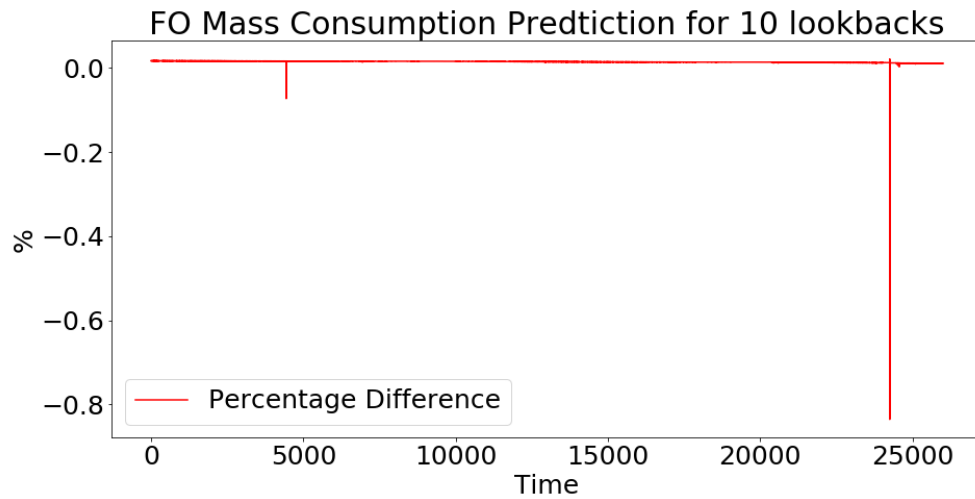
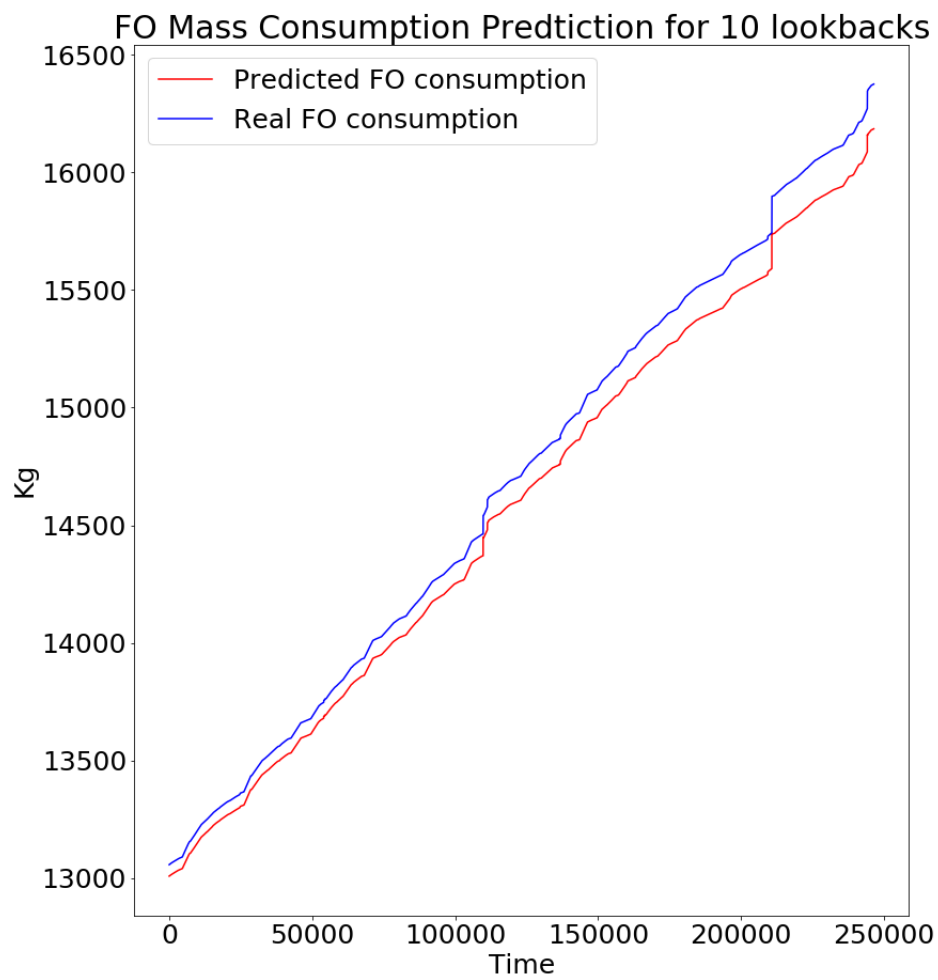
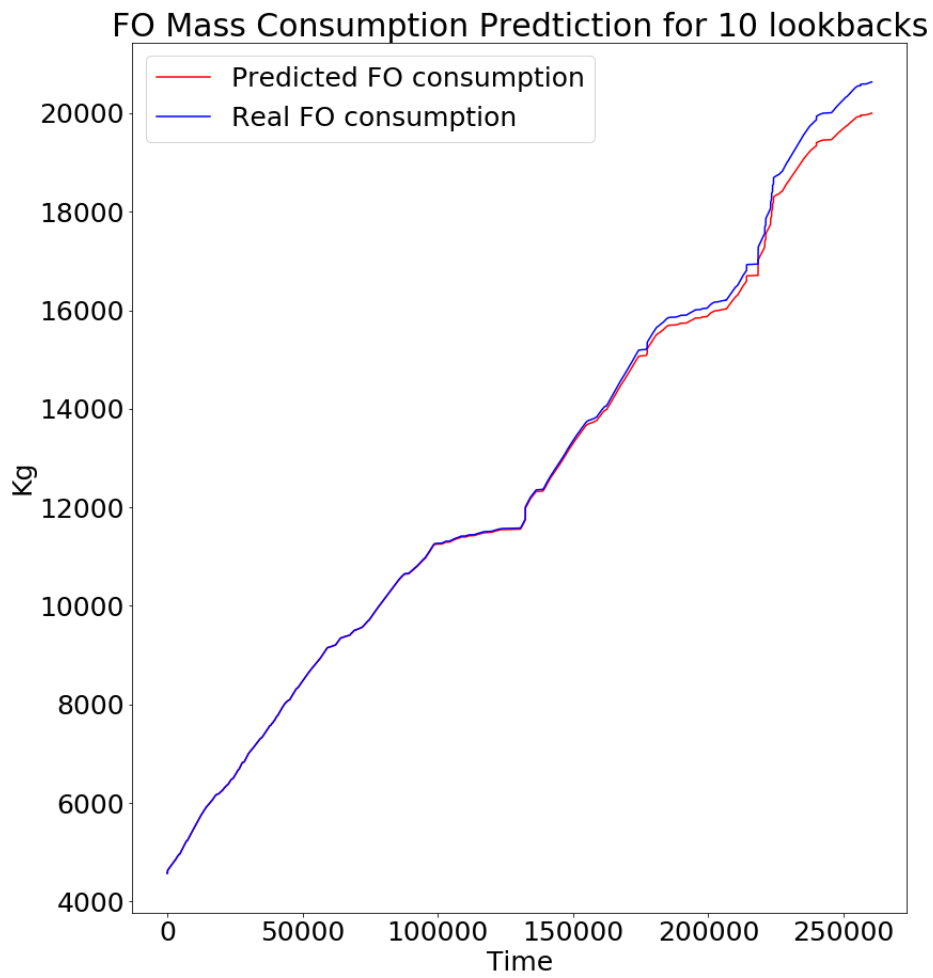
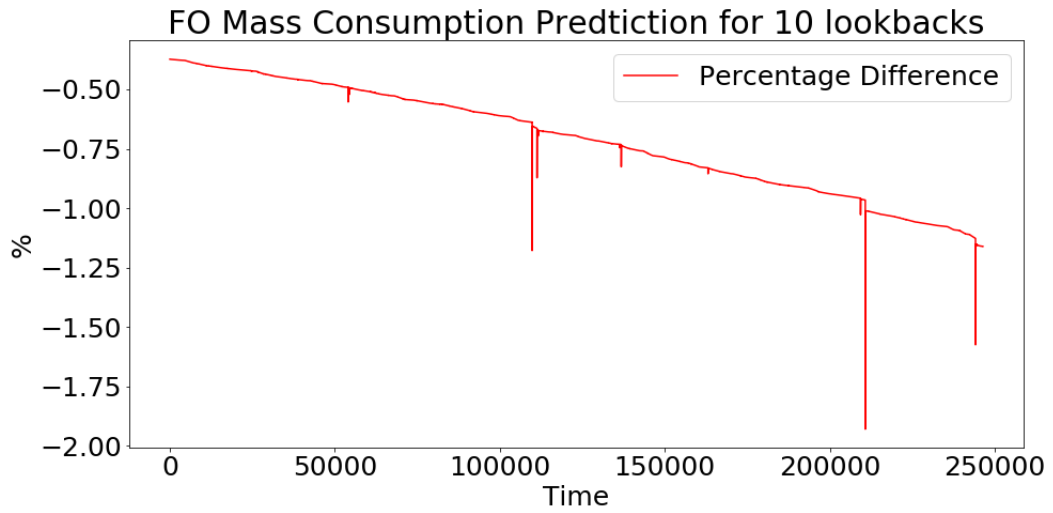
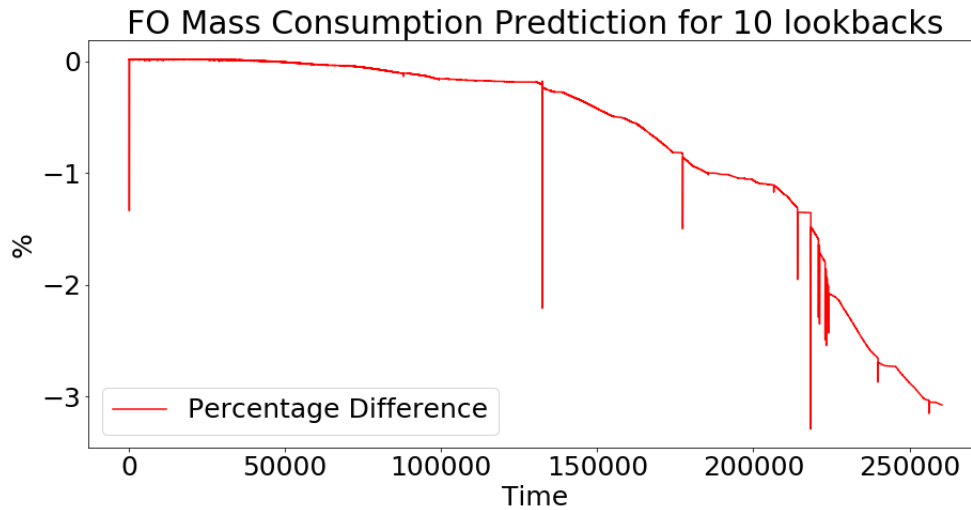


Figure 5.14: Results of the % difference of the *first* ship after 50 epochs for 10 lookback







#### 5.1.4 15 Lookbacks

The best results for 10 lookbacks using only one 50 neuron LSTM layer were produced when the model was trained for 5, 10, 15, 50 and 50 epochs for the first ship and for 5 and 20 epochs for the second and for the third ship. So overall the best results were produced when the model was trained for 5 epochs.

From the figures below the following conclusions have been reached:

- The results are similar to the results for 10 lookbacks meaning, the model was able to train very well for the first ship and generalize for the second and third ship in some degree.

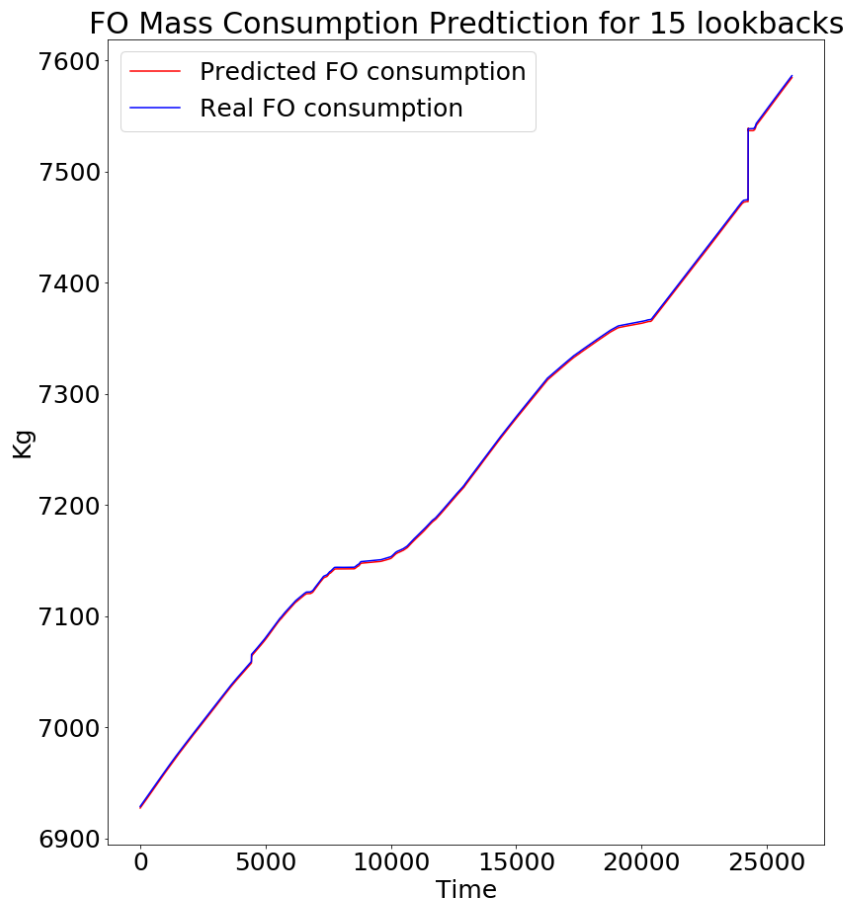


Figure 5.15: Results of the *first* ship after 5 epochs for 15 lookback

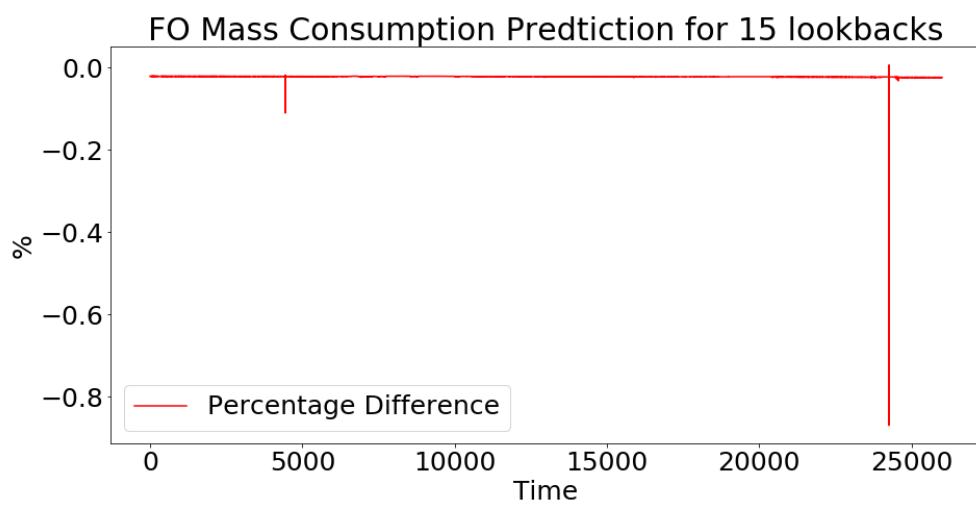


Figure 5.16: Results of the % difference of the *first* ship after 5 epochs for 15 lookback

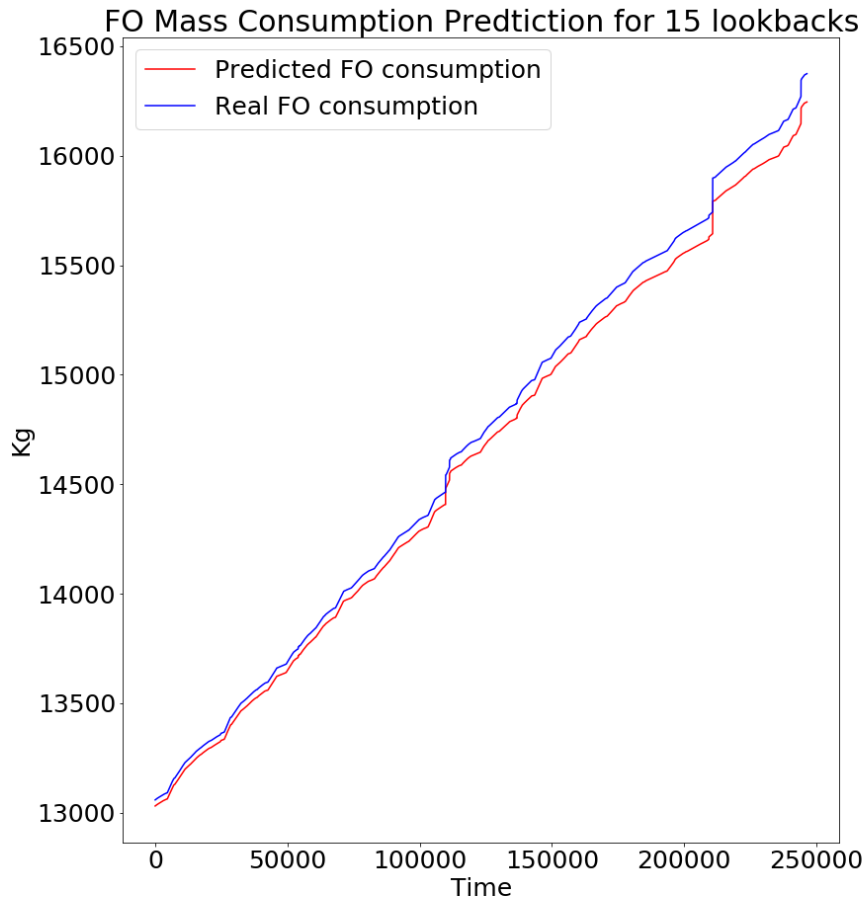


Figure 5.17: Results of the *second* ship after 5 epochs for 15 lookback

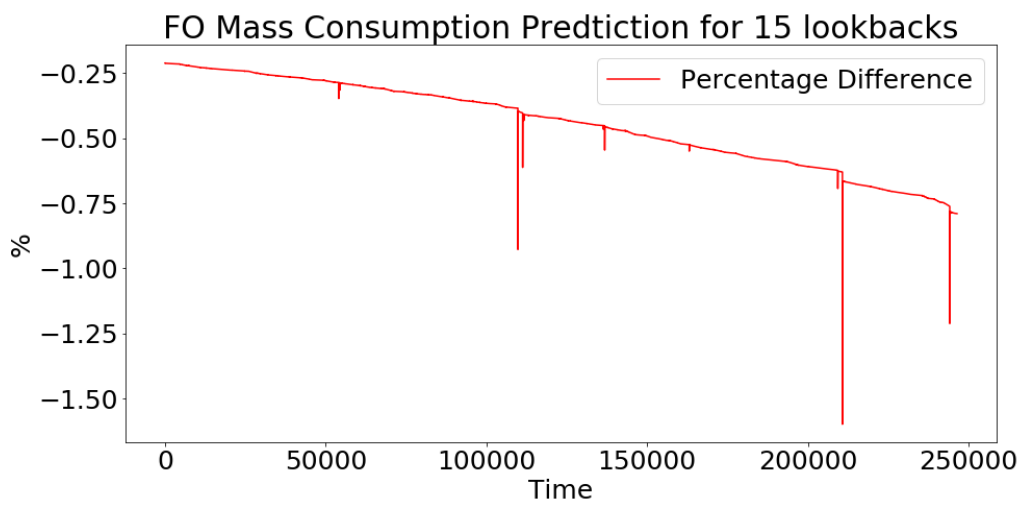


Figure 5.18: Results of the % difference of the *second* ship after 5 epochs for 15 lookback

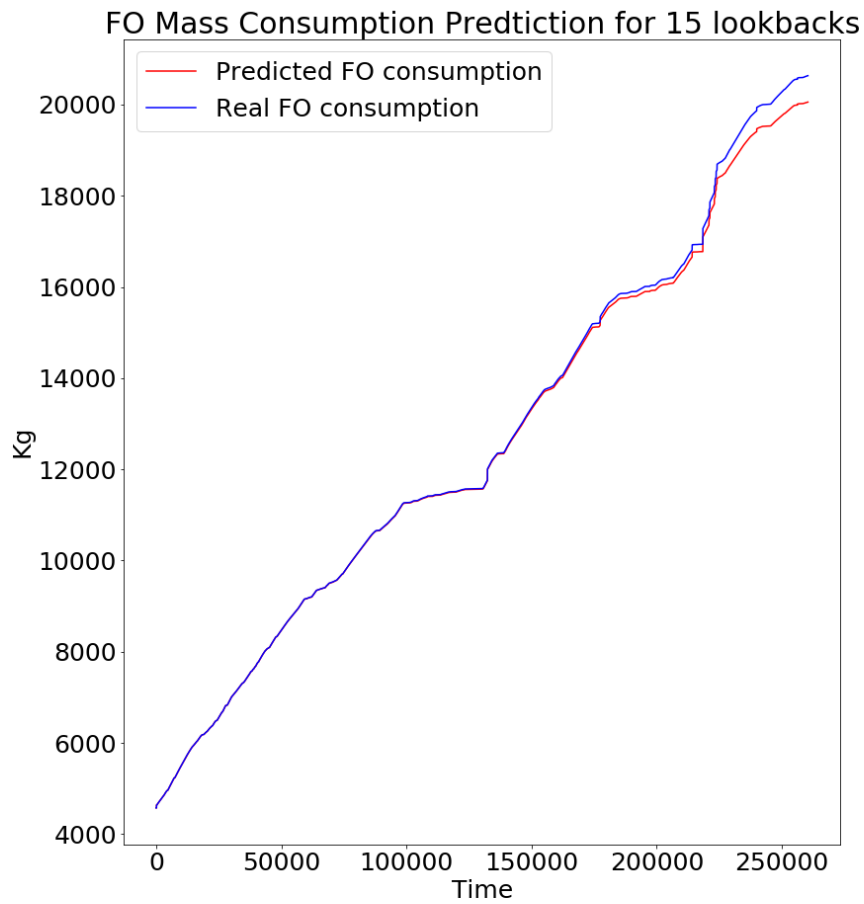


Figure 5.19: Results of the *third* ship after 5 epochs for 15 lookback

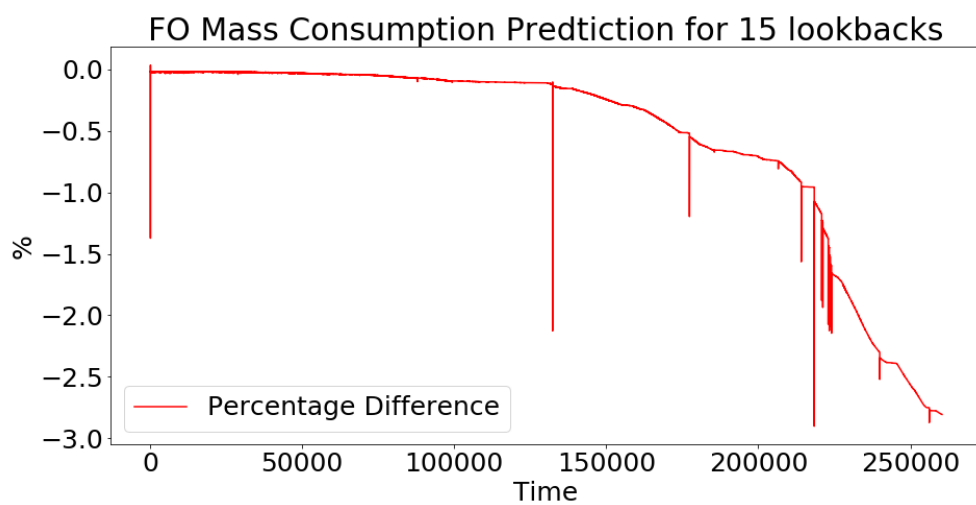


Figure 5.20: Results of the % difference of the *third* ship after 5 epochs for 15 lookback

### 5.1.5 20 Lookbacks

The best results for 20 lookbacks using only one 50 neuron LSTM layer were produced when the model was trained for 30 and 50 epochs for the first ship and for 15, 30 and 50 epochs for the second and 50 epochs for the third ship. So overall the best results were produced when the model was trained for 50 epochs.

From the figures below the following conclusions have been reached:

- Once again the model succeeded to train incredibly well for the first ship, with an error almost 0%.
- Moreover, the model achieve an incredible performance on the second ship's dataset with an error approximately 0.2%, even though as it can be seen from the plot it overshooted a little bit in the last part of the dataset.
- The most impressive result though, was that on the third ship's dataset, where the model achieve a similar performance as in the second dataset meaning that the model was able to generalize pretty well.

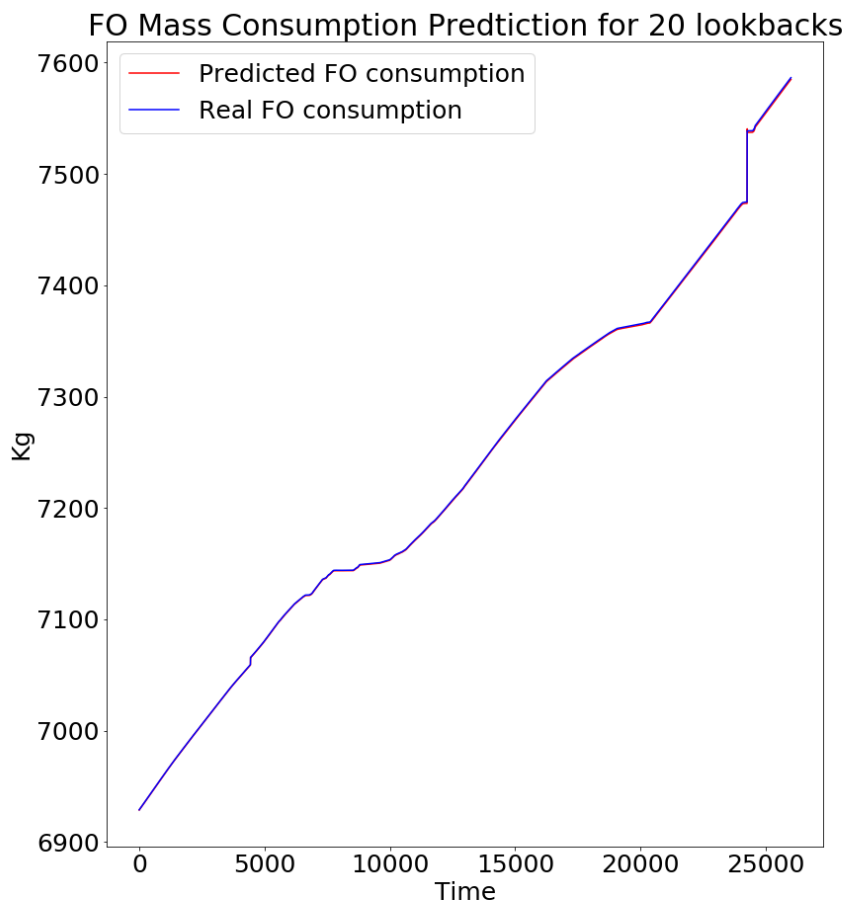


Figure 5.21: Results of the *first* ship after 50 epochs for 20 lookback



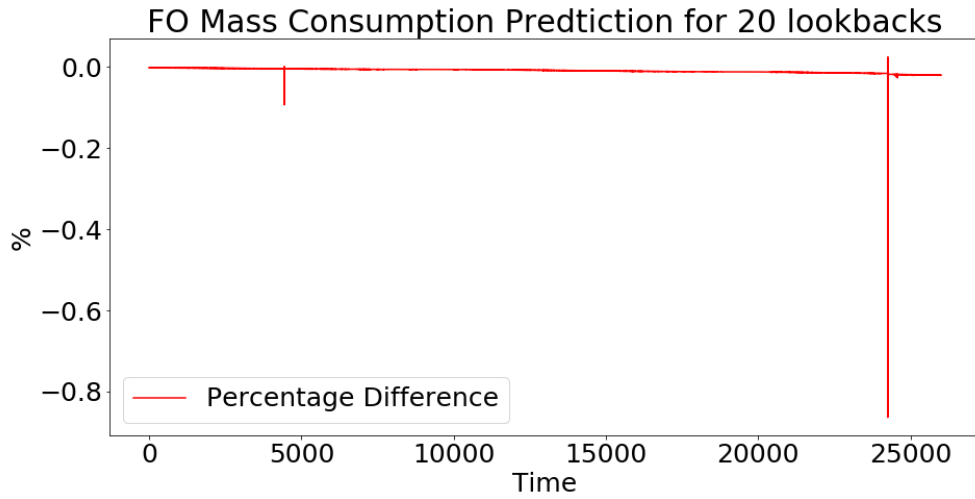


Figure 5.22: Results of the % difference of the *first* ship after 50 epochs for 20 lookback

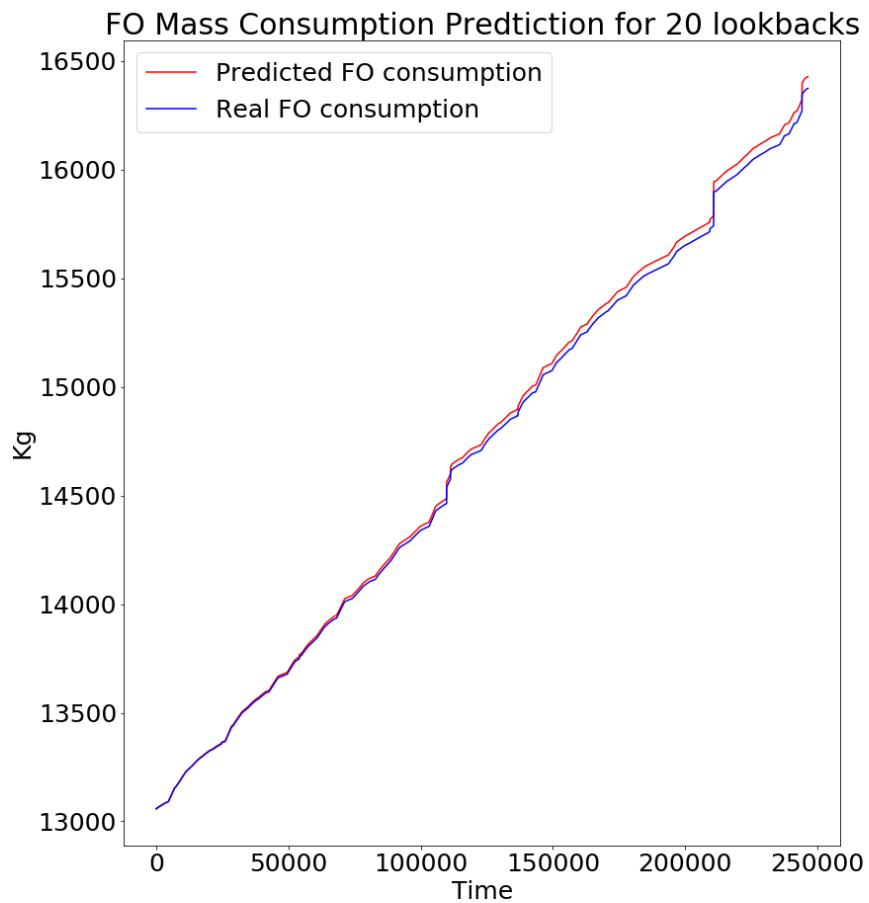


Figure 5.23: Results of the *second* ship after 50 epochs for 20 lookback

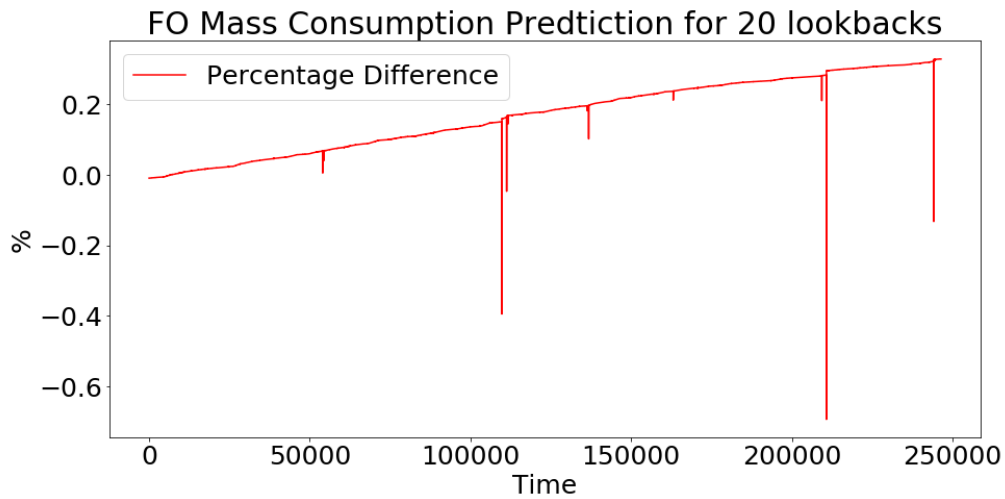


Figure 5.24: Results of the % difference of the *second* ship after 50 epochs for 20 lookback

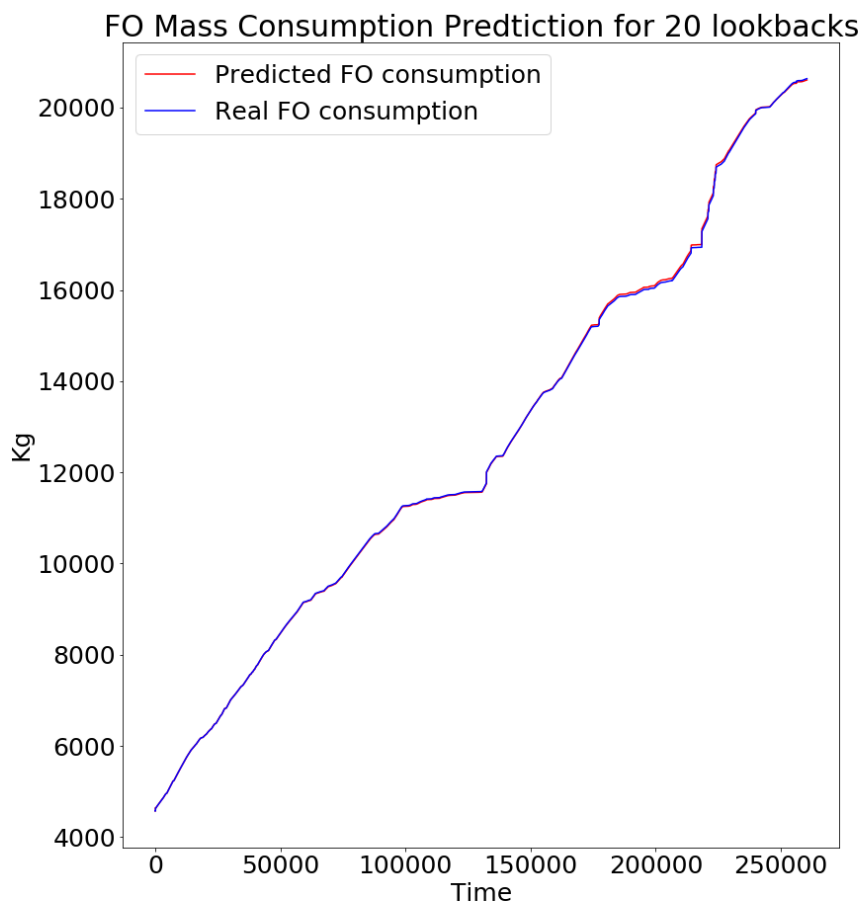


Figure 5.25: Results of the *third* ship after 50 epochs for 20 lookback

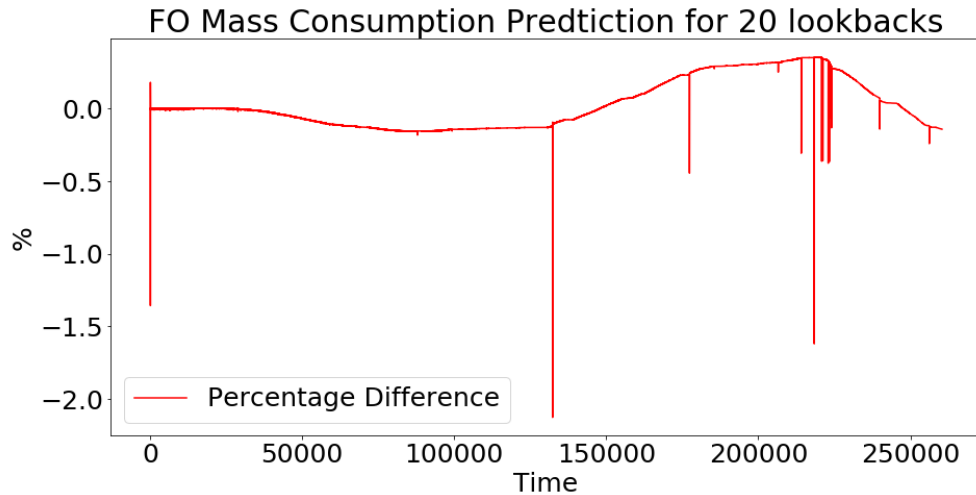


Figure 5.26: Results of the % difference of the *third* ship after 50 epochs for 20 lookback

### 5.1.6 50 Lookbacks

The best results for 50 lookbacks using only one 50 neuron LSTM layer were produced when the model was trained for 5 and 20 epochs for the first ship and for 30 epochs for the second, with the 5 and 50 epochs being close, and 5 epochs for the third ship. So overall the best results were produced when the model was trained for 5 epochs.

From the figures below the following conclusions have been reached:

- The results are similar to the results for 10 and 15 lookbacks meaning, the model was able to train very well for the first ship and generalize for the second and third ship in some degree.

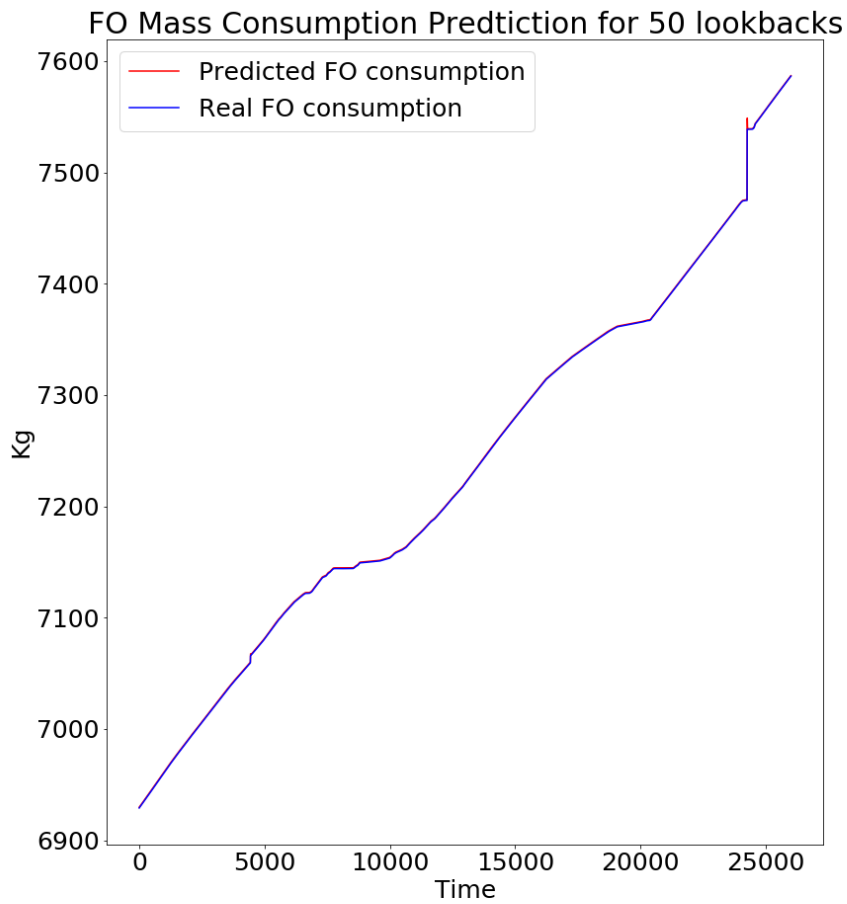


Figure 5.27: Results of the *first* ship after 5 epochs for 50 lookback

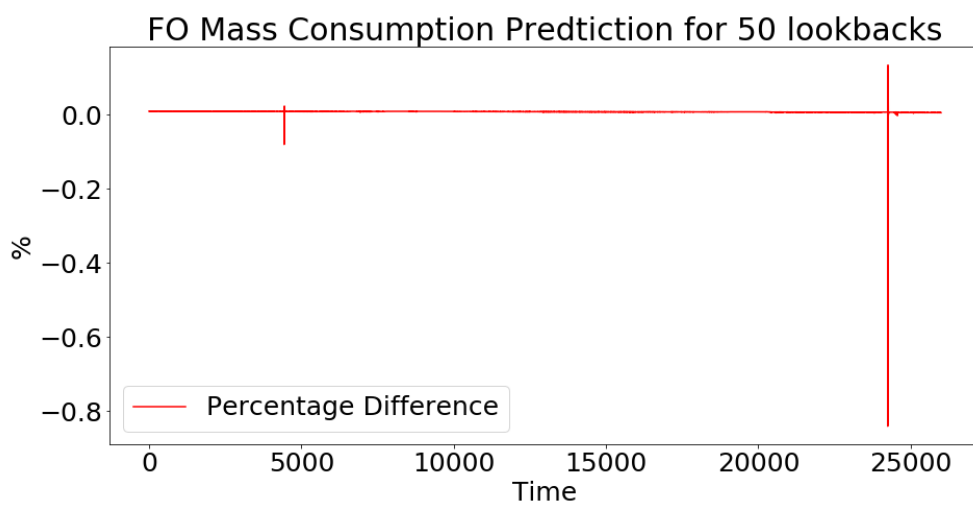


Figure 5.28: Results of the % difference of the *first* ship after 5 epochs for 50 lookback

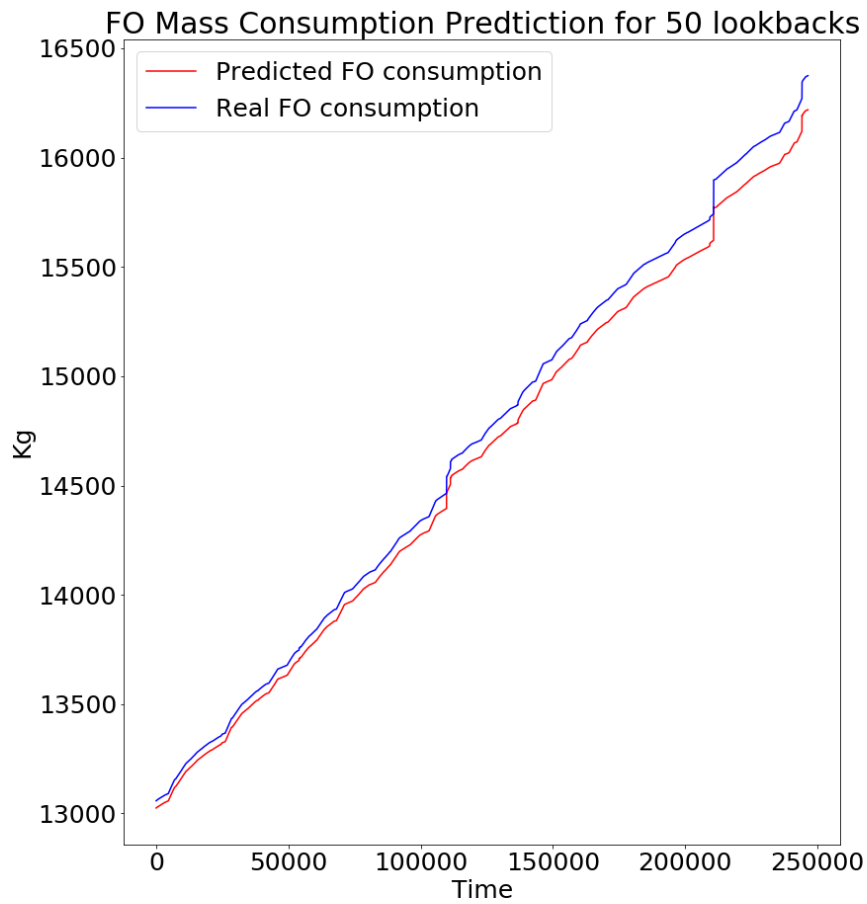


Figure 5.29: Results of the *second* ship after 5 epochs for 50 lookback

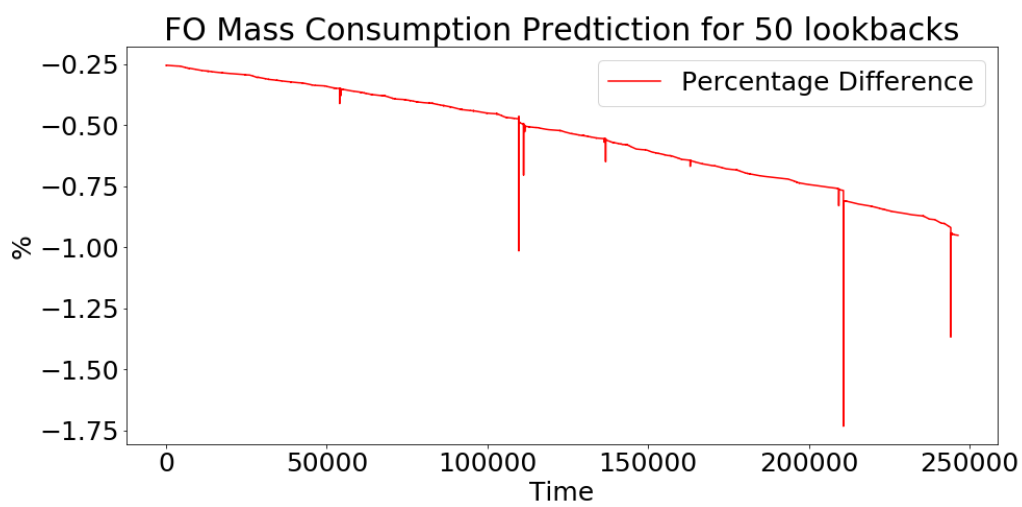


Figure 5.30: Results of the % difference of the *second* ship after 5 epochs for 50 lookback

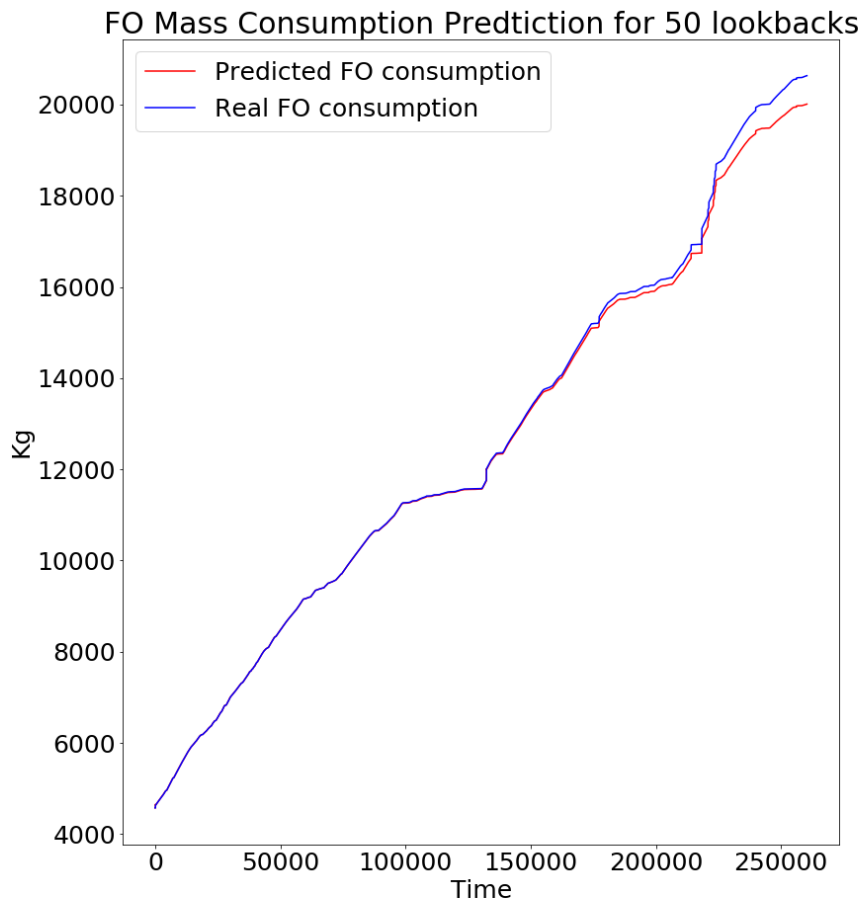


Figure 5.31: Results of the *third* ship after 5 epochs for 50 lookback

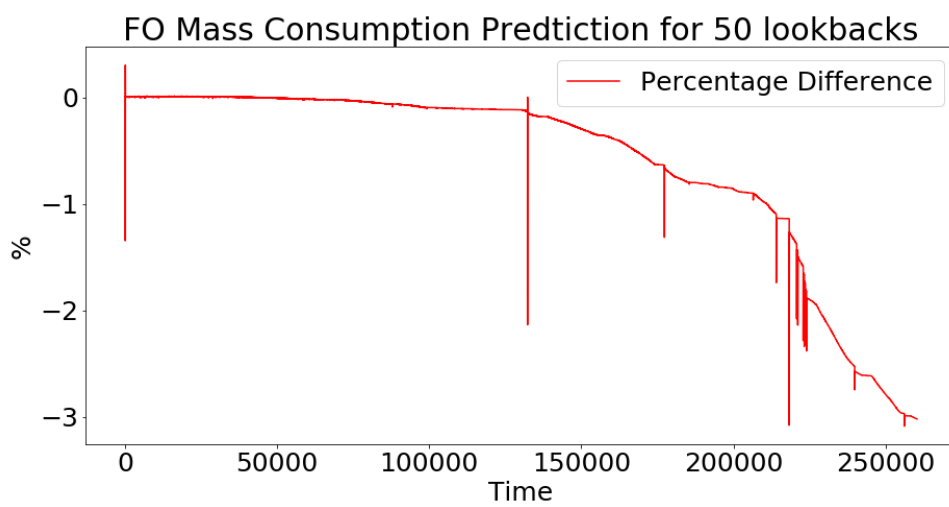


Figure 5.32: Results of the % difference of the *third* ship after 5 epochs for 50 lookback

## 5.2 Two variables: Fuel Oil Consumption and Fuel Oil Temperature

In the previous paragraph, the results of using only the FOC previous values. In this paragraph the fuel oil temperature variable was also used, to examine if this addition would increase the performance of the previous models. In the figure 5.33 the new layout of the NN is presented.

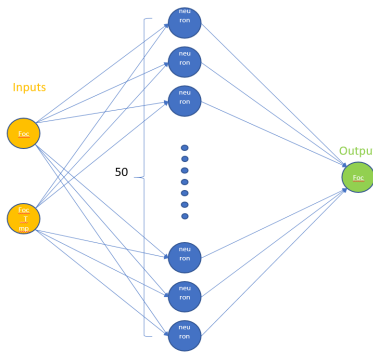


Figure 5.33: A NN with two inputs(FOC,FOC\_tmp) one layer with 50 neurons and one output(FOC)

### 5.2.1 1 Lookback

The best results for one lookback using only one 50 neuron LSTM layer were produced when the model was trained for 30 and 50 epochs for the first ship and for 5 and 30 epochs for the second and for 30 epochs for the third ship. So overall the best results were produced when the model was trained for 30 epochs.

In the figures below the following conclusions have been reached:

- First of all the model succeeded to train incredibly well for the first ship, with an error almost 0%.
- Secondly, the model seemed to be able to generalize in sister ship level with an error below 2%.
- Thirdly, the model succeeded to generalize in type ship level with an error below 1% for the first half of the dataset but failed to keep it below that in the other half of the dataset.

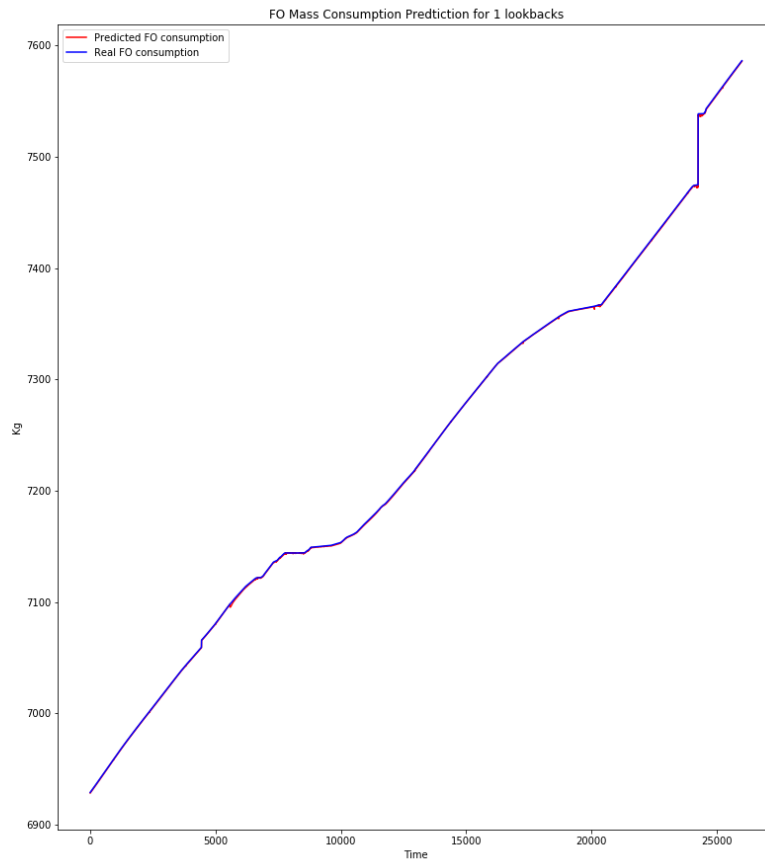


Figure 5.34: Results of the *first* ship after 30 epochs for 1 lookback

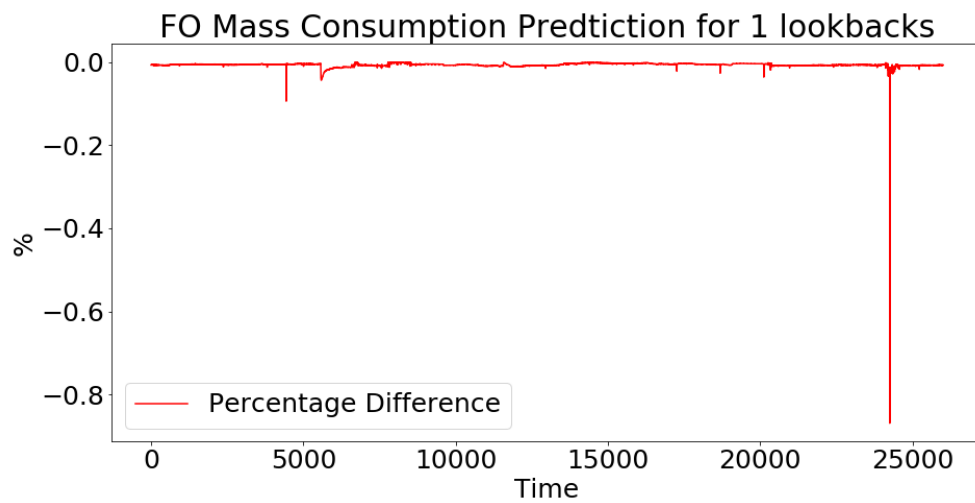


Figure 5.35: Results of the % difference of the *first* ship after 30 epochs for 1 lookback



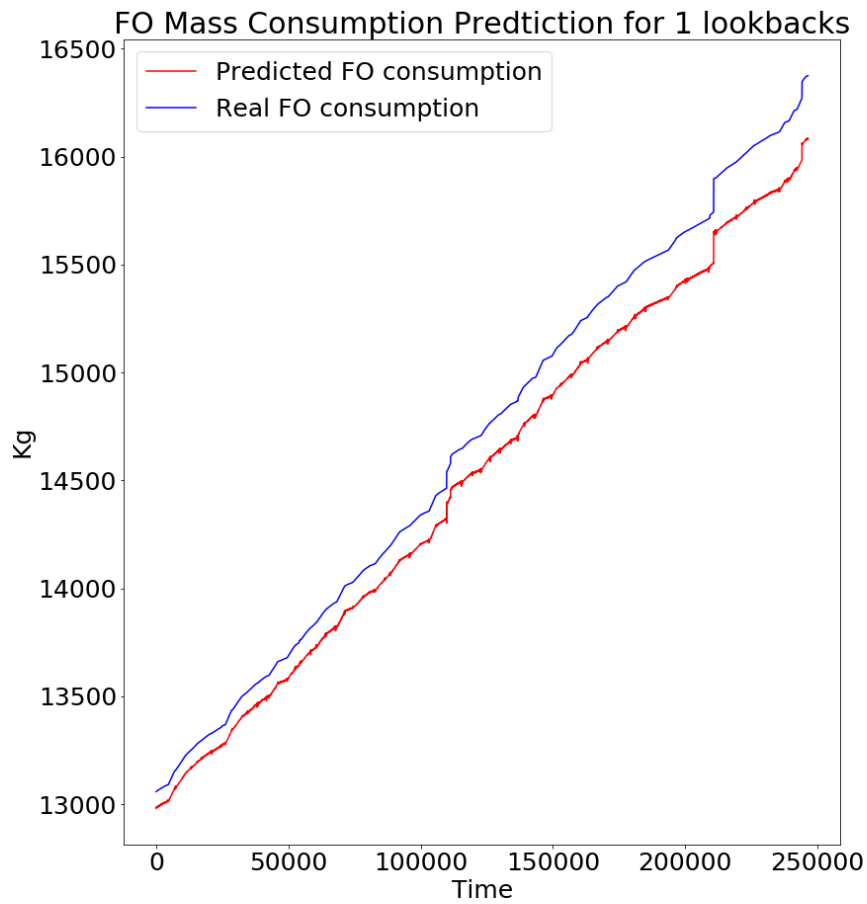


Figure 5.36: Results of the *second* ship after 30 epochs for 1 lookback

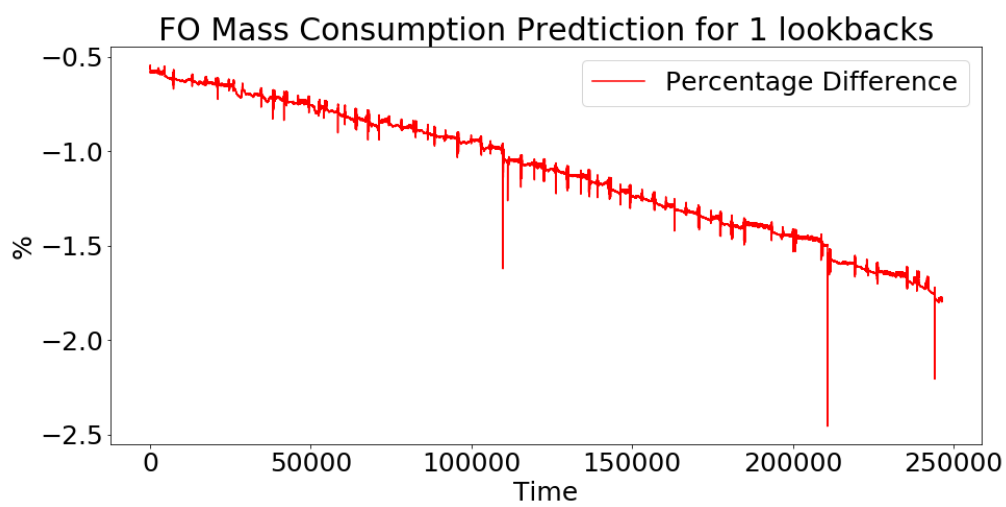


Figure 5.37: Results of the % difference of the *second* ship after 30 epochs for 1 lookback

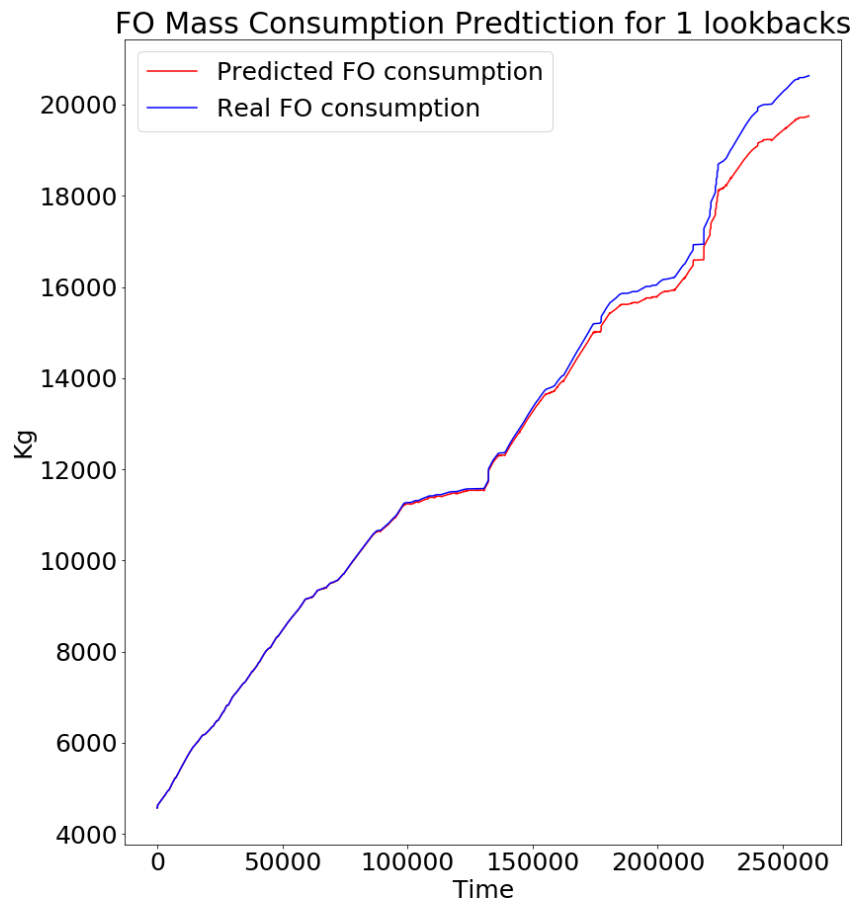


Figure 5.38: Results of the *third* ship after 30 epochs for 1 lookback

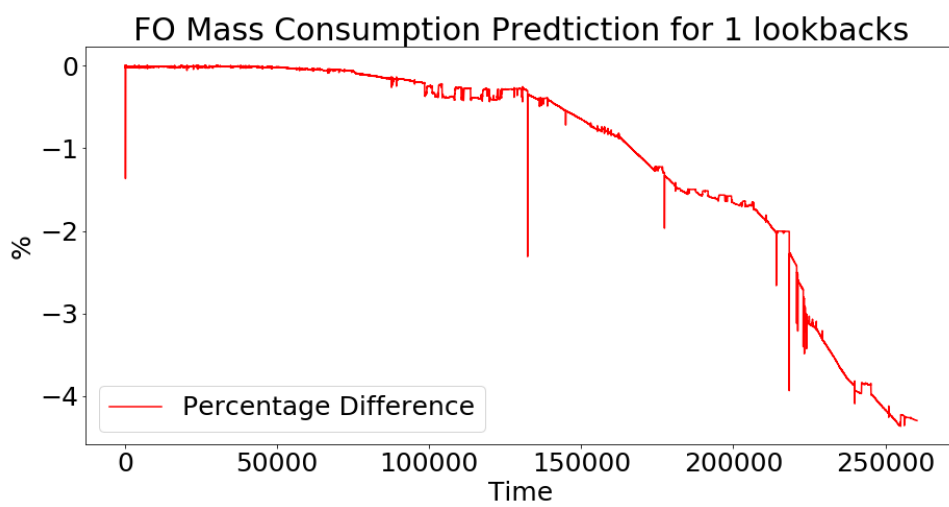


Figure 5.39: Results of the % difference of the *third* ship after 30 epochs for 1 lookback

### 5.2.2 5 Lookbacks

The best results for 5 lookbacks using only one 50 neuron LSTM layer were produced when the model was trained for 20 epochs for the first ship and for 5, 20 and 50 epochs for the second and for 15, 20 and 50 epochs for the third ship. So overall the best results were produced when the model was trained for 20 epochs.

From the figures below the following conclusions have been reached:

- First of all, as in the 1 lookback the model succeeded to train incredibly well for the first ship, with an error around 0%.
- However, it failed to generalize so well for the second ship(sister ship) and for the third.

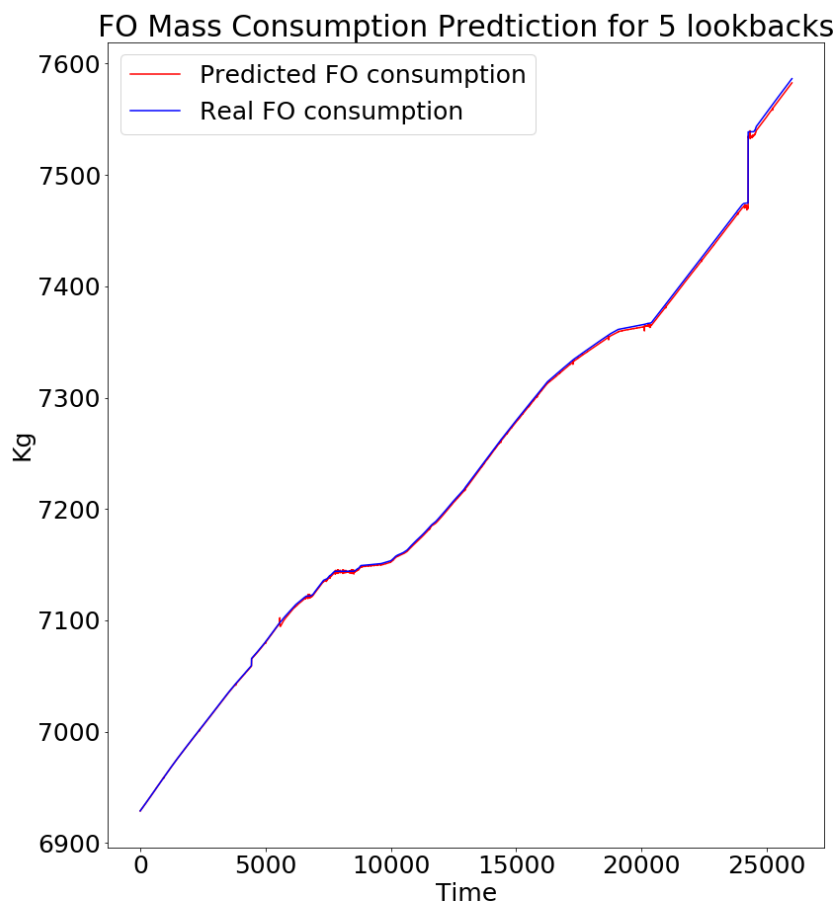


Figure 5.40: Results of the *first* ship after 20 epochs for 5 lookback

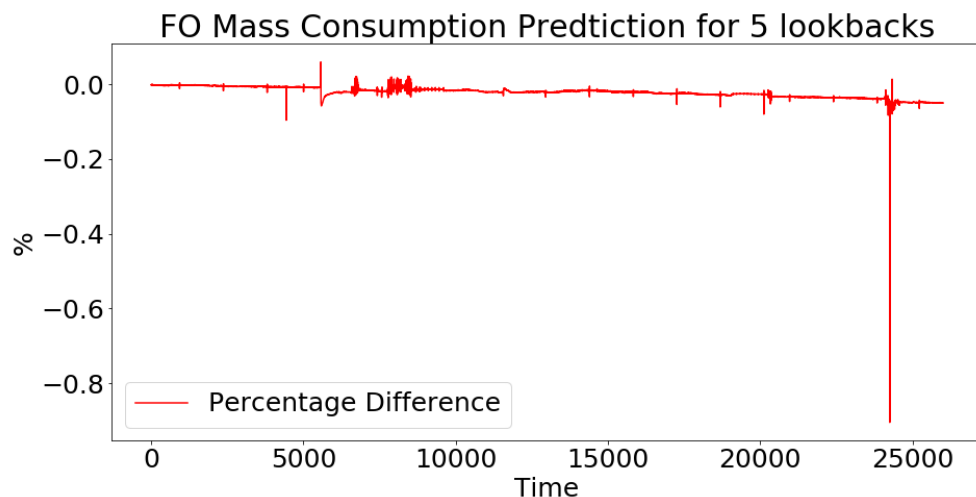


Figure 5.41: Results of the % difference of the *first* ship after 20 epochs for 5 lookback

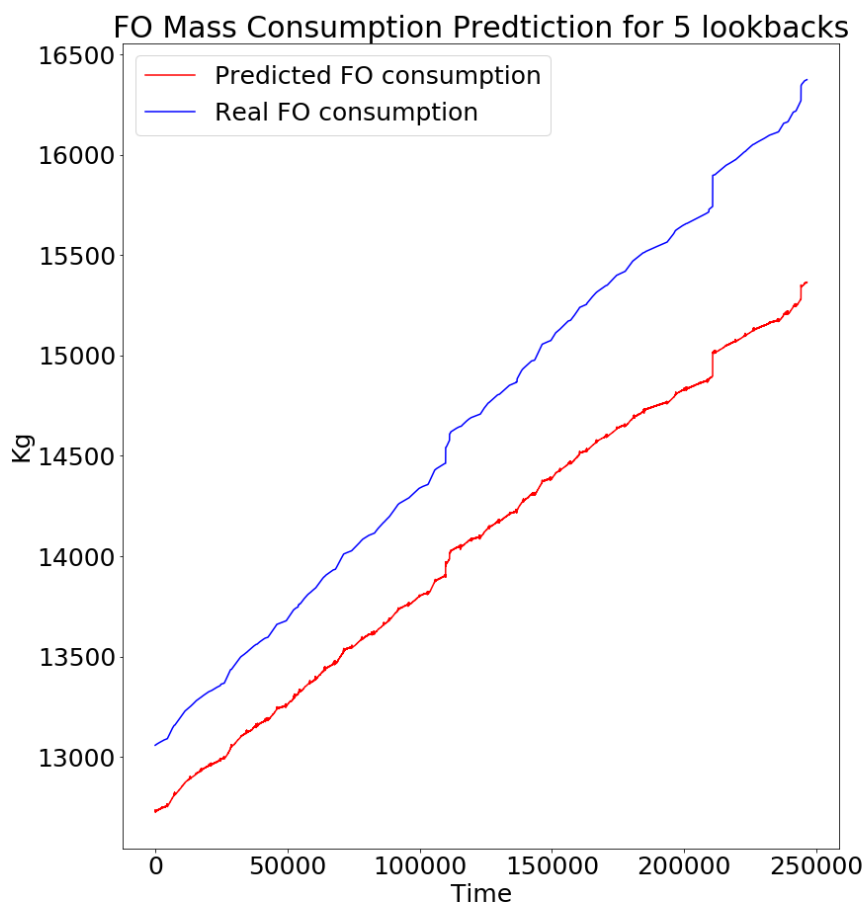


Figure 5.42: Results of the *second* ship after 20 epochs for 5 lookback

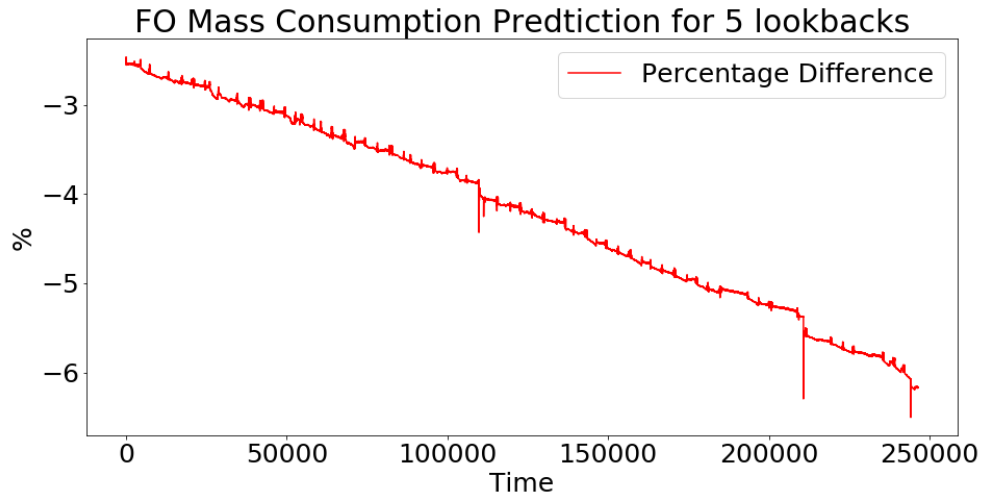


Figure 5.43: Results of the % difference of the *second* ship after 20 epochs for 5 lookback

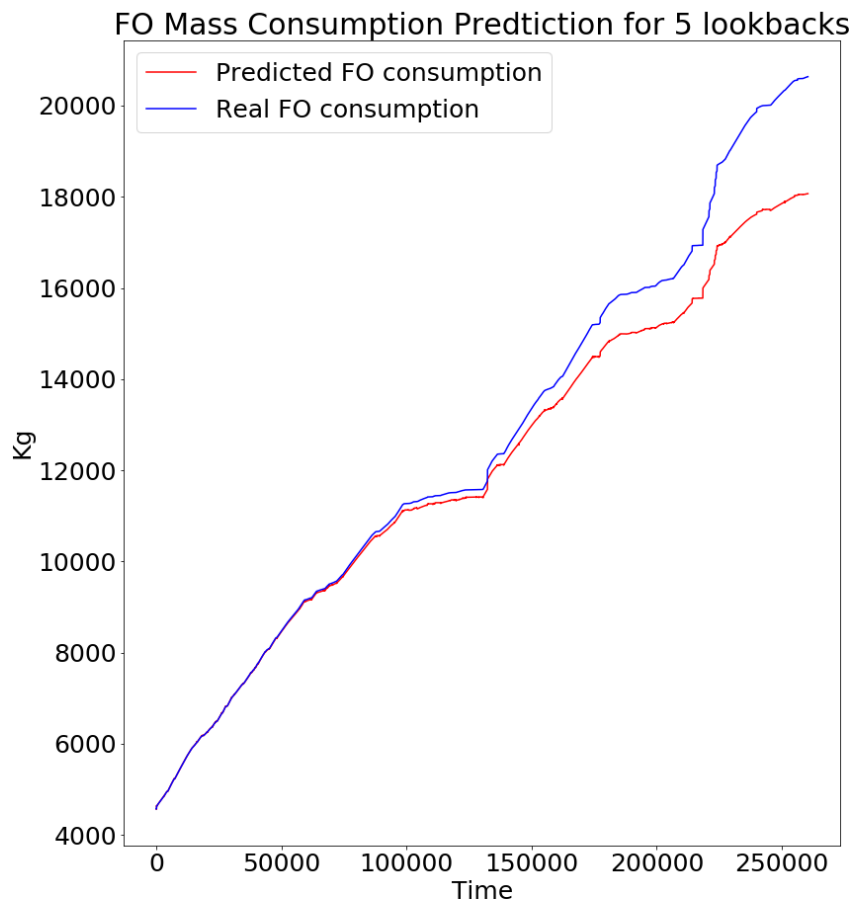


Figure 5.44: Results of the *third* ship after 20 epochs for 5 lookback

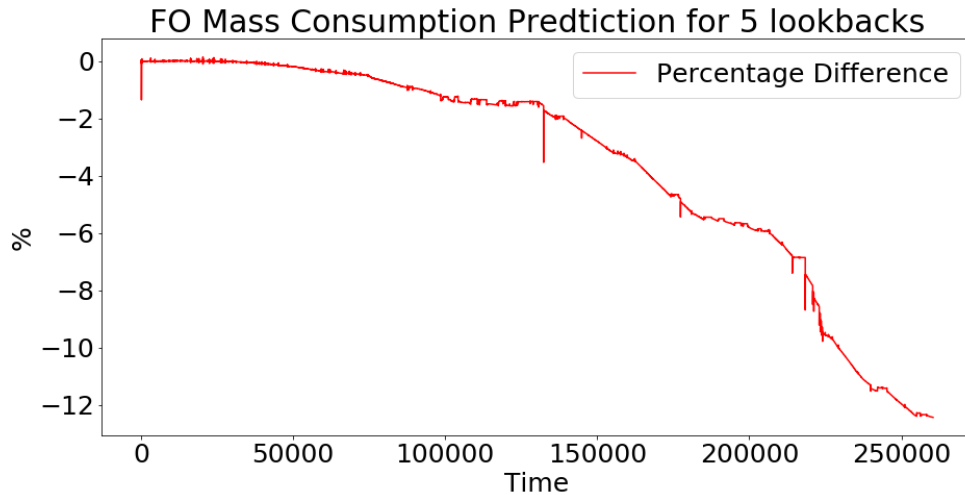


Figure 5.45: Results of the % difference of the *third* ship after 20 epochs for 5 lookback

### 5.2.3 10 Lookbacks

The best results for 10 lookbacks using only one 50 neuron LSTM layer were produced when the model was trained for 15, 20, 50 and 50 epochs for the first ship and for 15, 20, 30 and 50 epochs for the second and 20 epochs for the third ship. So overall the best results were produced when the model was trained for 20 epochs.

From the figures below the following conclusions have been reached:

- Once again the model was able to train very well for the first ship, with an error around 0%.
- The model's results for the second ship were satisfactory, with an error below 1.25%.
- The model's result on the third ship are similar to the second in the first half of the dataset with an error below 1% and then it drops to 3%

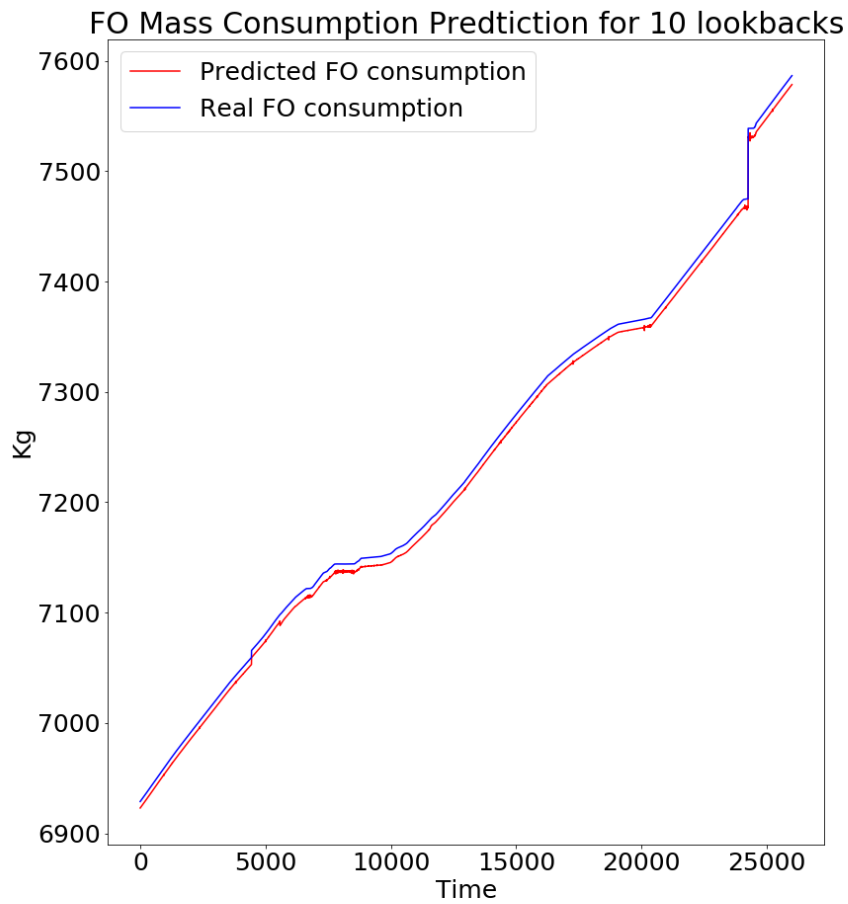


Figure 5.46: Results of the *first* ship after 20 epochs for 10 lookback

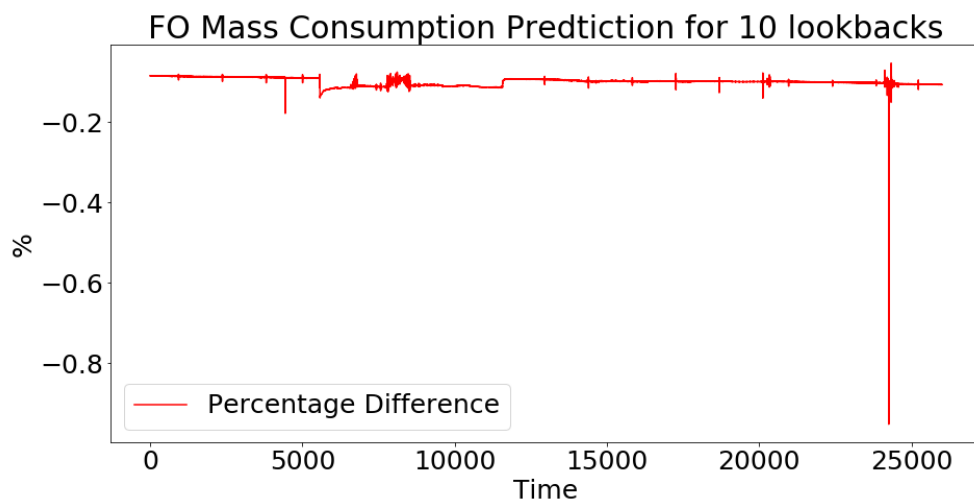


Figure 5.47: Results of the % difference of the *first* ship after 20 epochs for 10 lookback

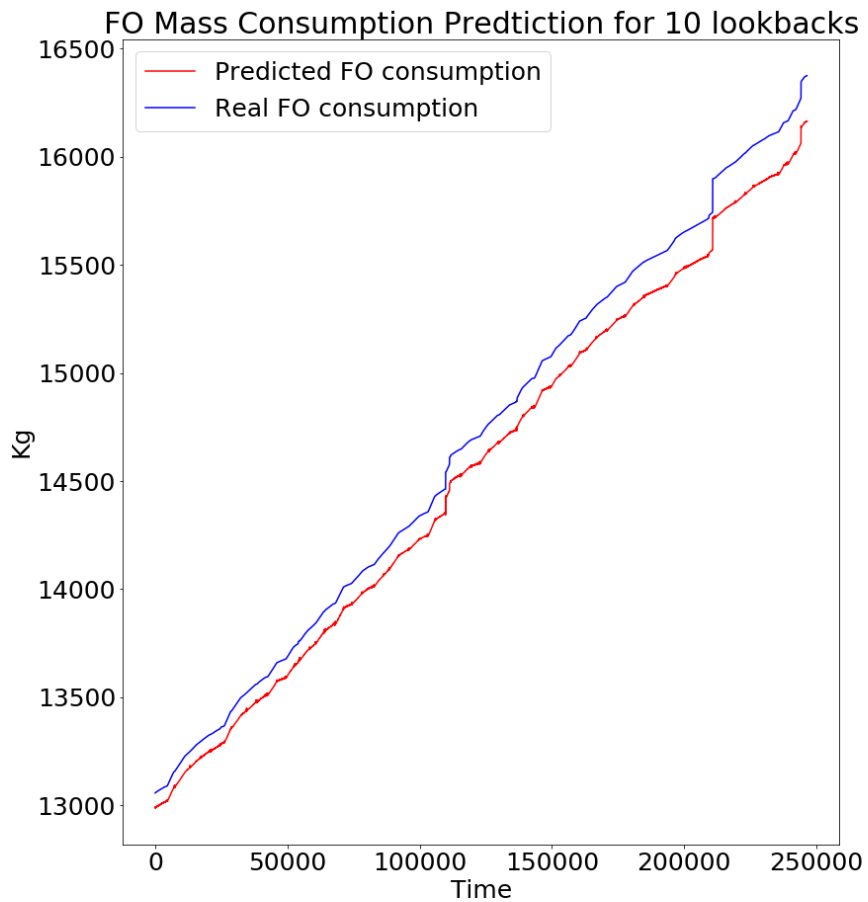


Figure 5.48: Results of the *second* ship after 20 epochs for 10 lookback

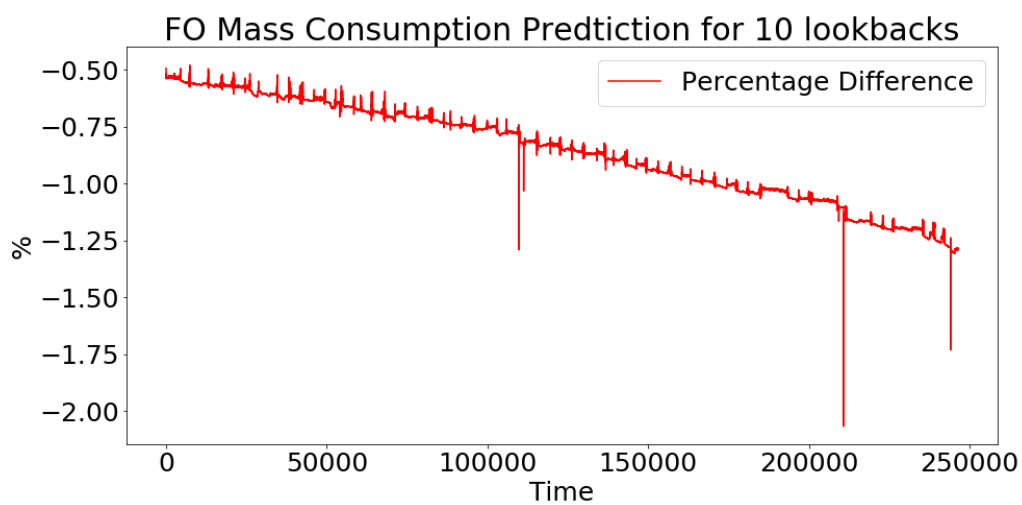


Figure 5.49: Results of the % difference of the *second* ship after 20 epochs for 10 lookback



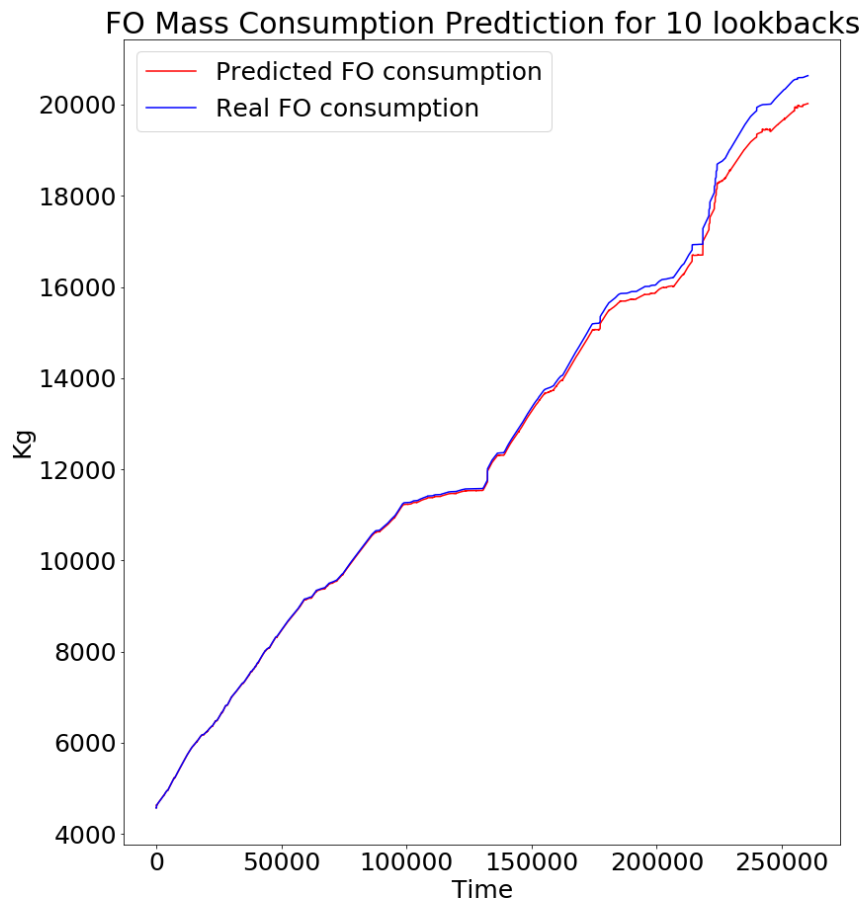


Figure 5.50: Results of the *third* ship after 20 epochs for 10 lookback

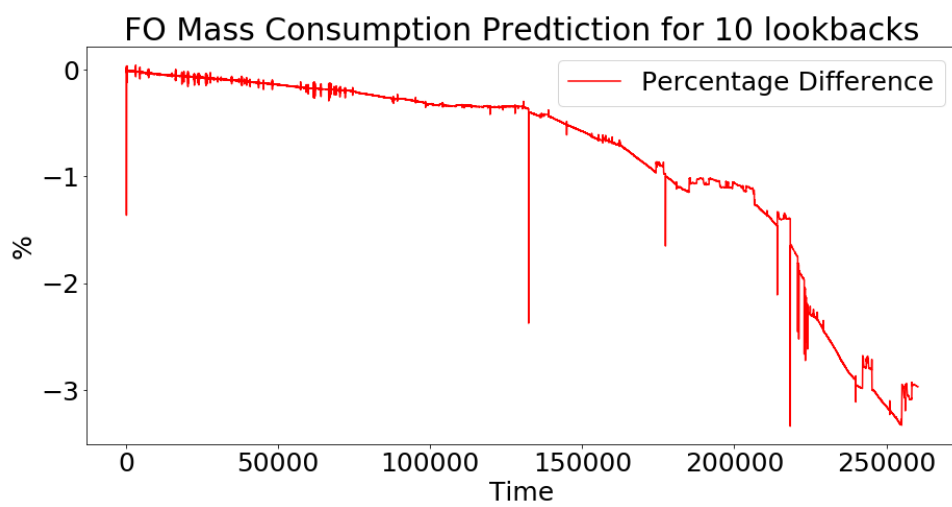


Figure 5.51: Results of the % difference of the *third* ship after 20 epochs for 10 lookback

### 5.2.4 15 Lookbacks

The best results for 10 lookbacks using only one 50 neuron LSTM layer were produced when the model was trained for 15, 20, 30, 50 and 50 epochs for the first ship and for 10 and 15 epochs for the second and for 15 and 30 epochs for the third ship. So overall the best results were produced when the model was trained for 15 epochs.

From the figures below the following conclusions have been reached:

- The results are similar to the results for 10 lookbacks meaning, the model was able to train very well for the first ship and generalize for the second and third ship in some degree.

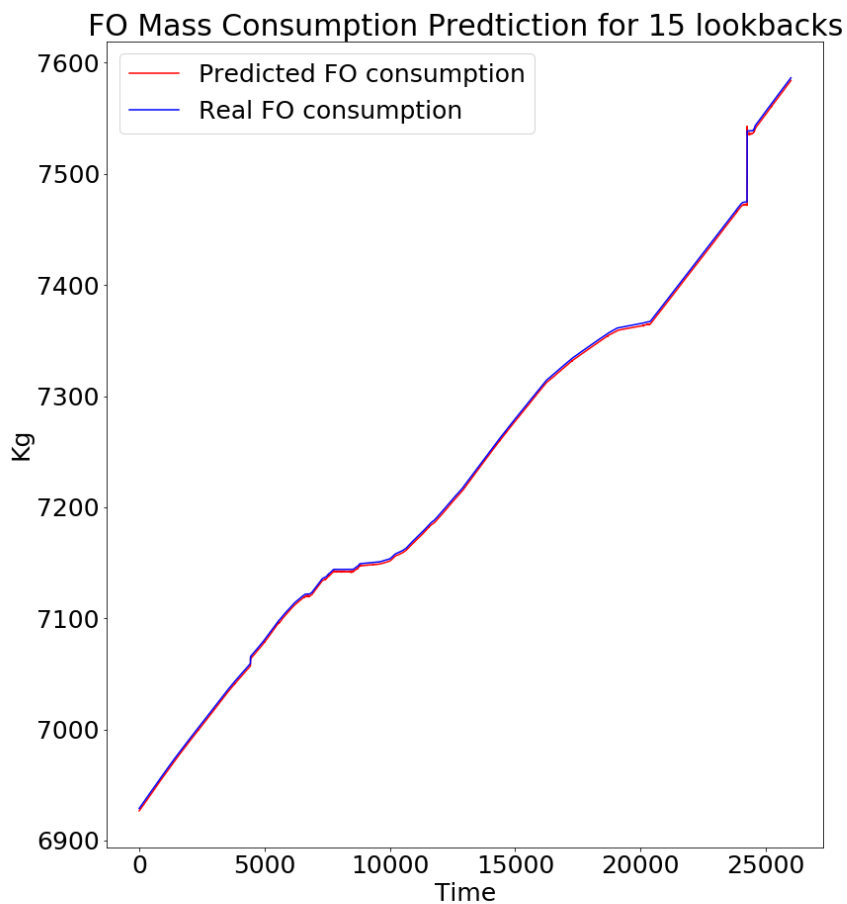


Figure 5.52: Results of the *first* ship after 15 epochs for 15 lookback

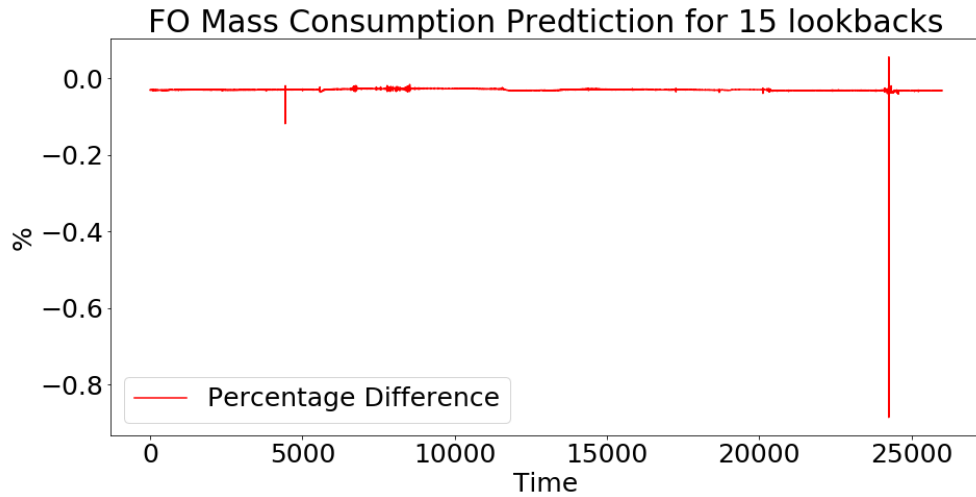


Figure 5.53: Results of the % difference of the *first* ship after 15 epochs for 15 lookback

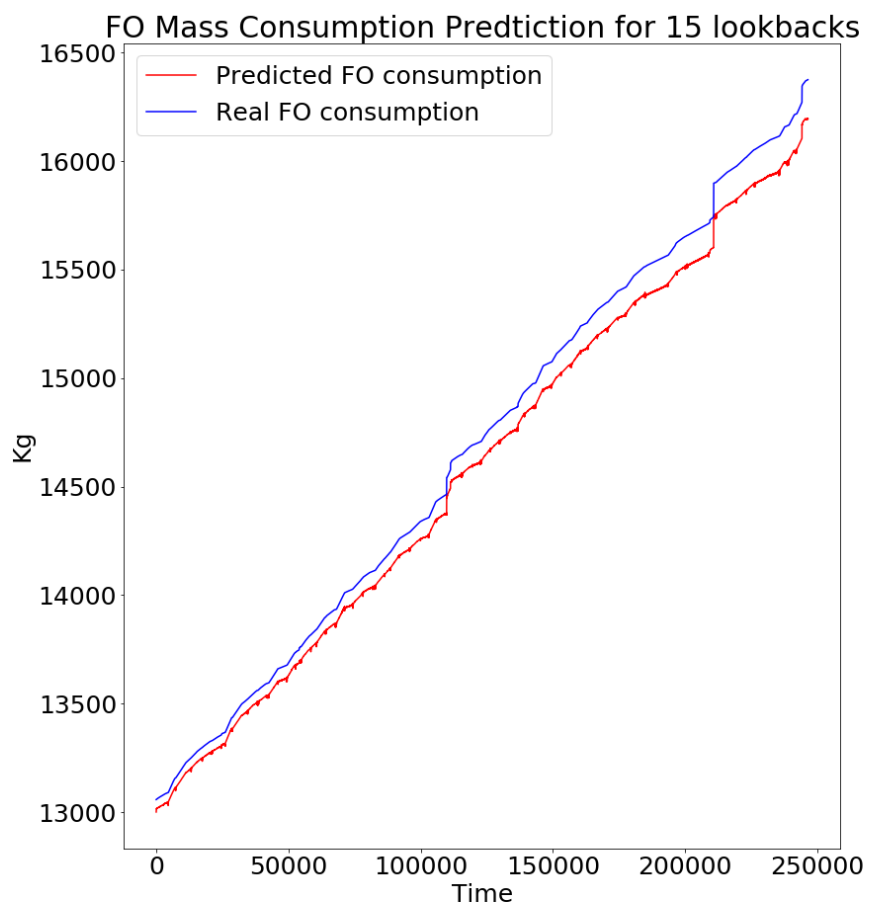


Figure 5.54: Results of the *second* ship after 15 epochs for 15 lookback

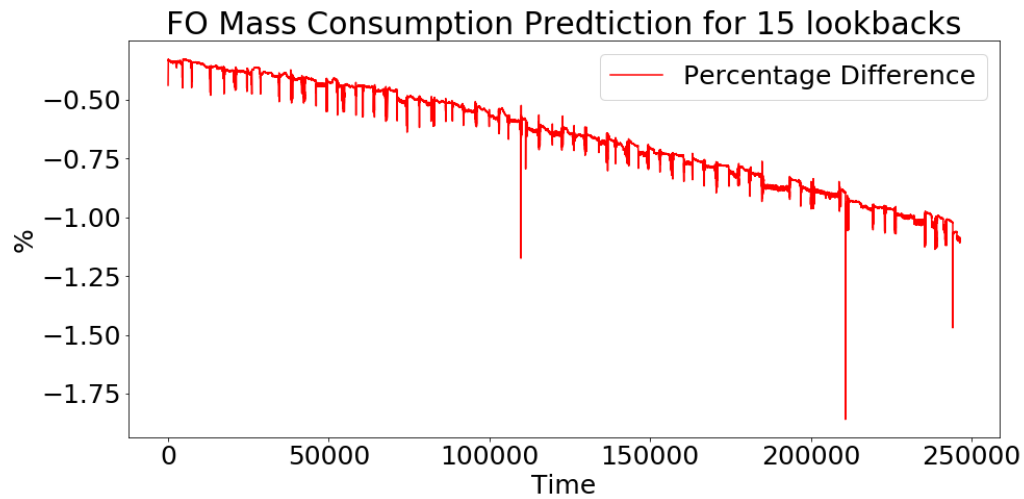


Figure 5.55: Results of the % difference of the *second* ship after 15 epochs for 15 lookback

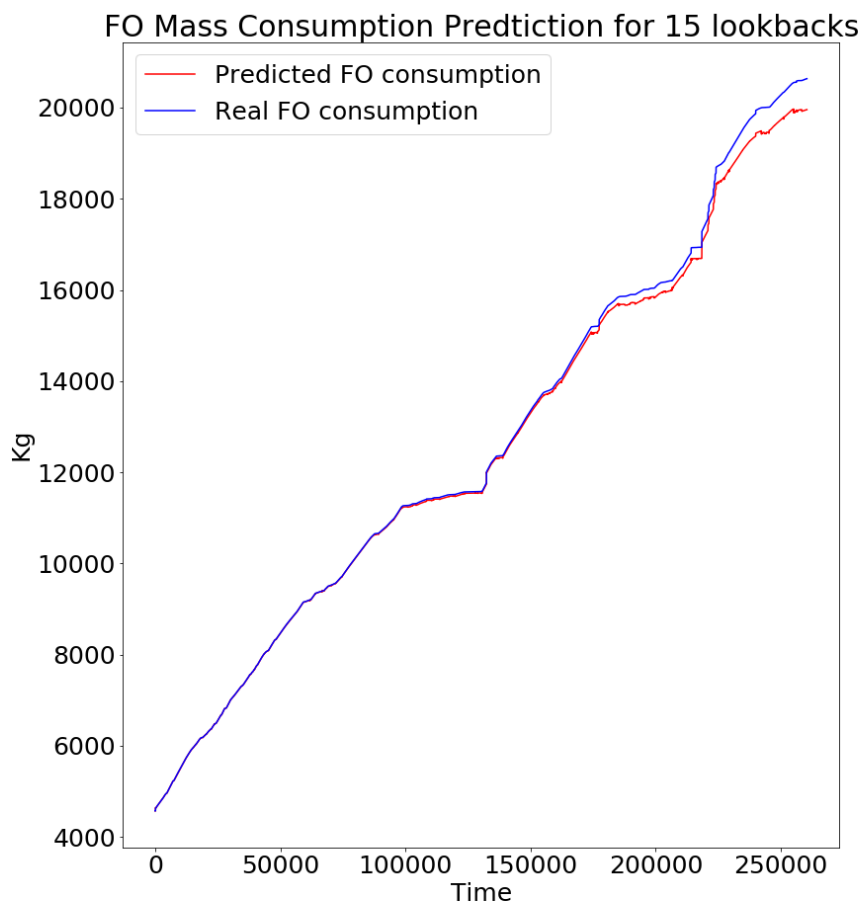


Figure 5.56: Results of the *third* ship after 15 epochs for 15 lookback

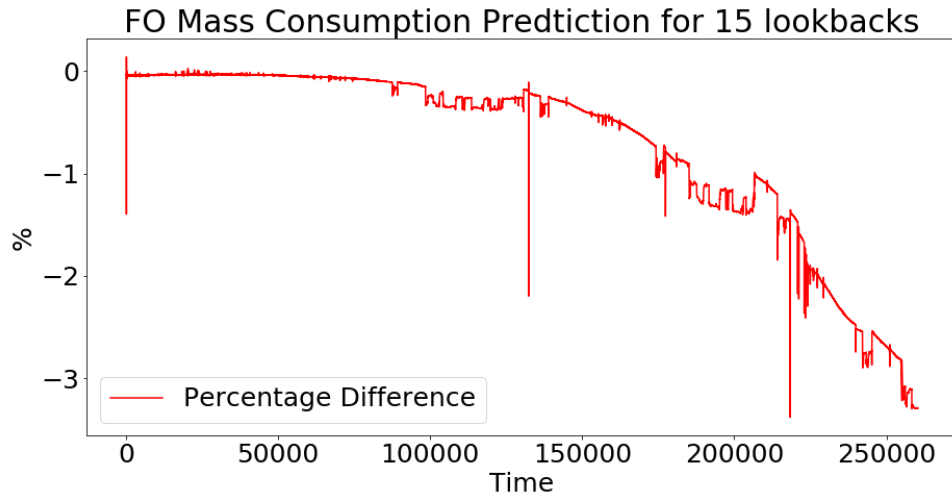


Figure 5.57: Results of the % difference of the *third* ship after 15 epochs for 15 lookback

### 5.2.5 20 Lookbacks

The best results for 20 lookbacks using only one 50 neuron LSTM layer were produced when the model was trained for 5, 20, 30 and 50 epochs for the first ship and for 5 and 30 epochs for the second and for 30 epochs for the third ship. So overall the best results were produced when the model was trained for 30 epochs.

From the figures below the following conclusions have been reached:

- Once again the model succeeded to train incredibly well for the first ship, with an error almost 0%.
- Moreover, the model achieve an good performance on the second ship's dataset with an error between (-0.5% and 0.5%).
- The model's result on the third ship's dataset was extremely well, with an error ranging between ( -0,5% and 0.5%), like in the performance in the second dataset. However in this one the overshoots were smaller than the ones the second.

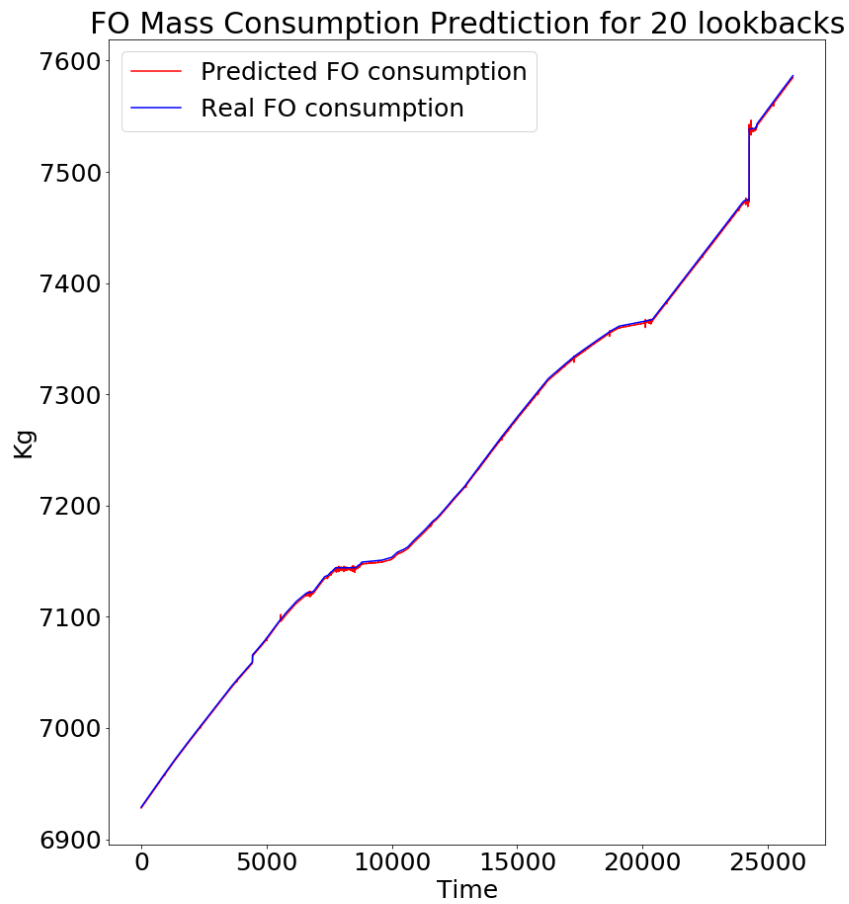


Figure 5.58: Results of the *first* ship after 30 epochs for 20 lookback

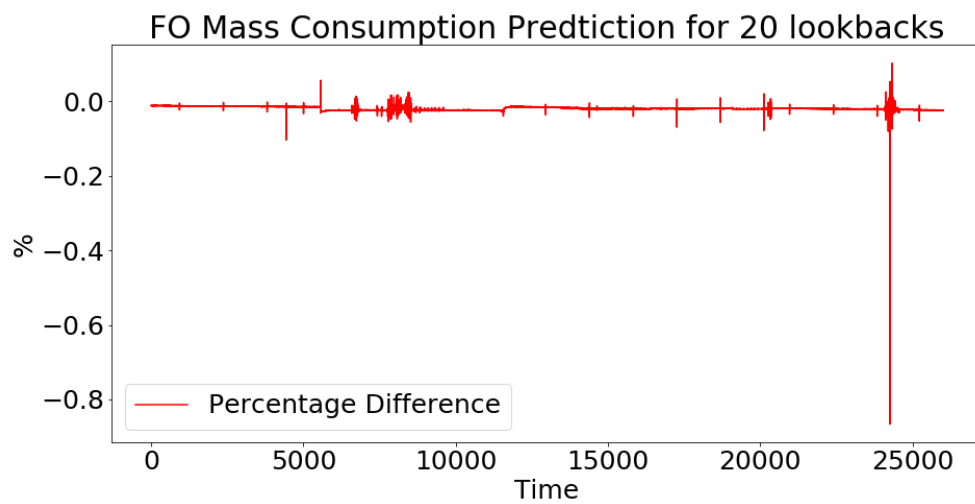


Figure 5.59: Results of the % difference of the *first* ship after 30 epochs for 20 lookback

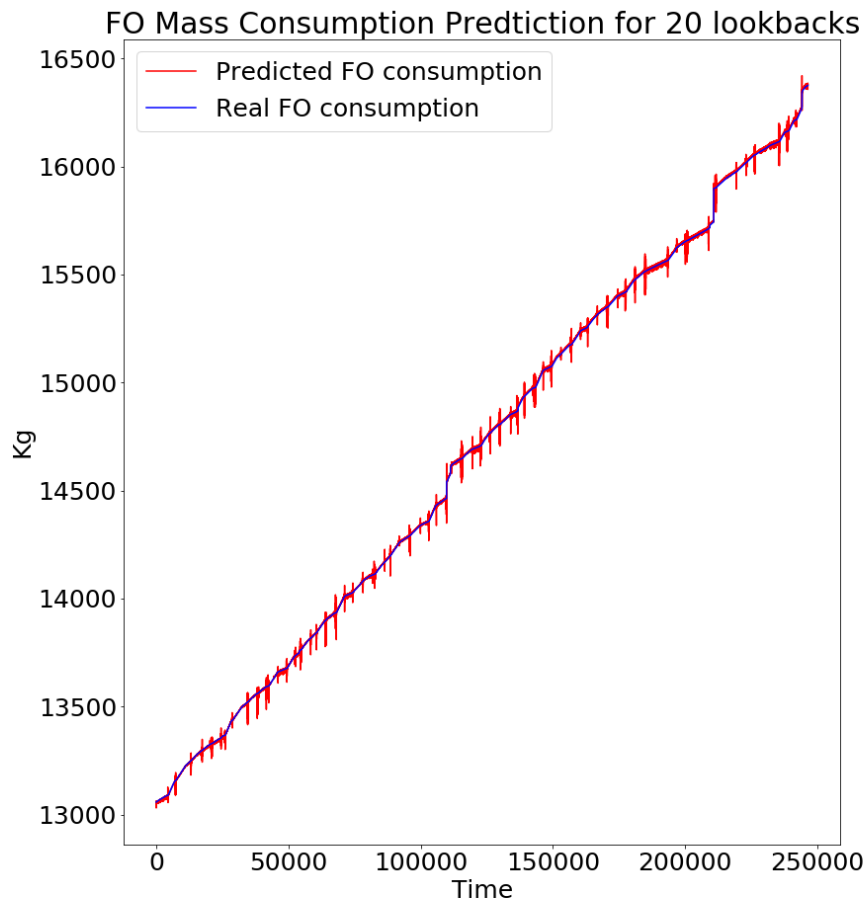


Figure 5.60: Results of the *second* ship after 30 epochs for 20 lookback

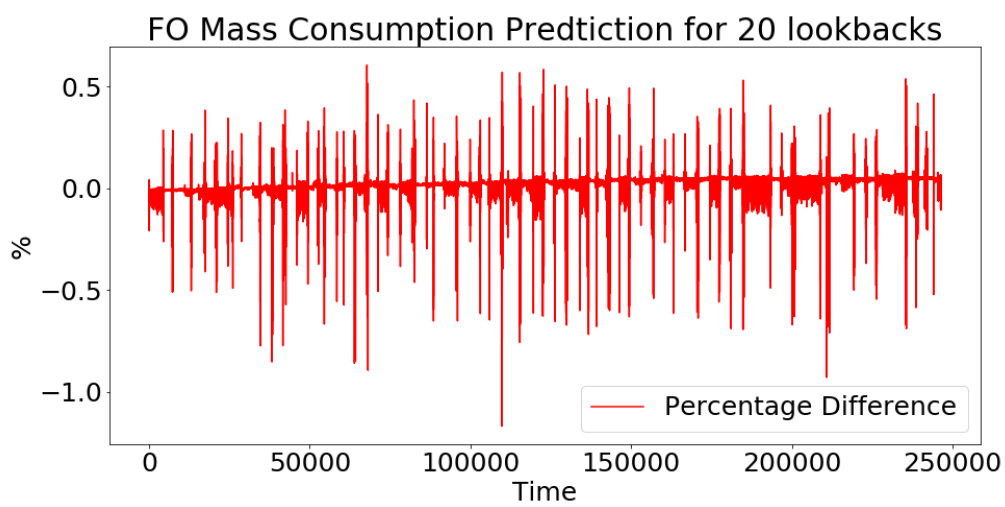


Figure 5.61: Results of the % difference of the *second* ship after 30 epochs for 20 lookback

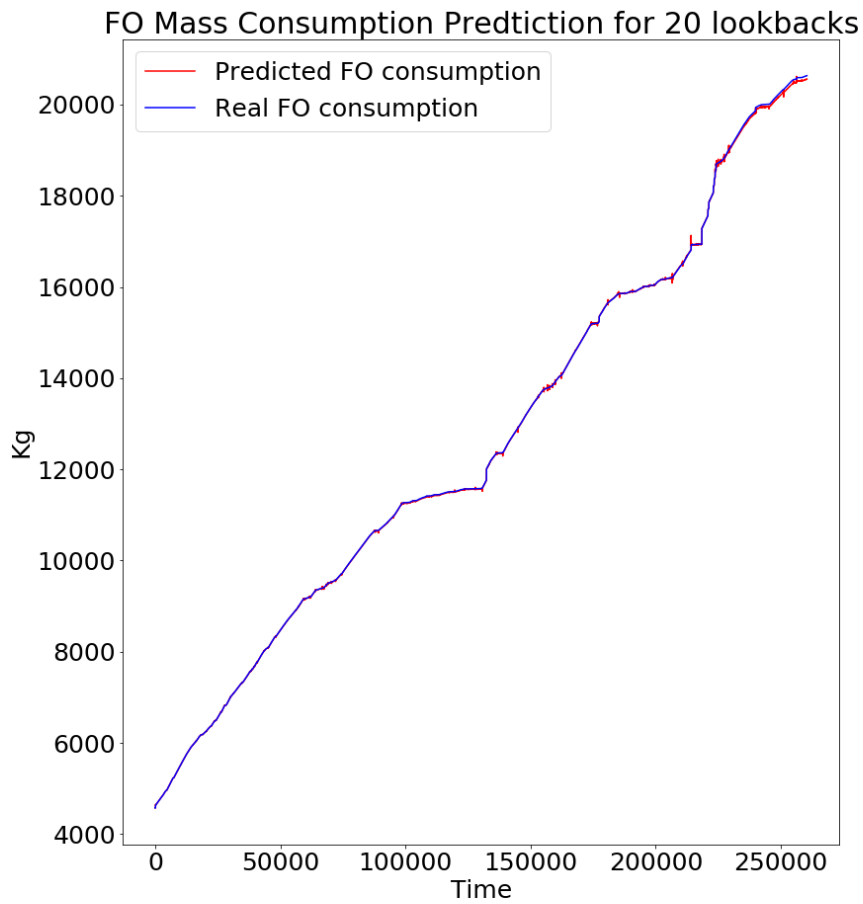


Figure 5.62: Results of the *third* ship after 30 epochs for 20 lookback

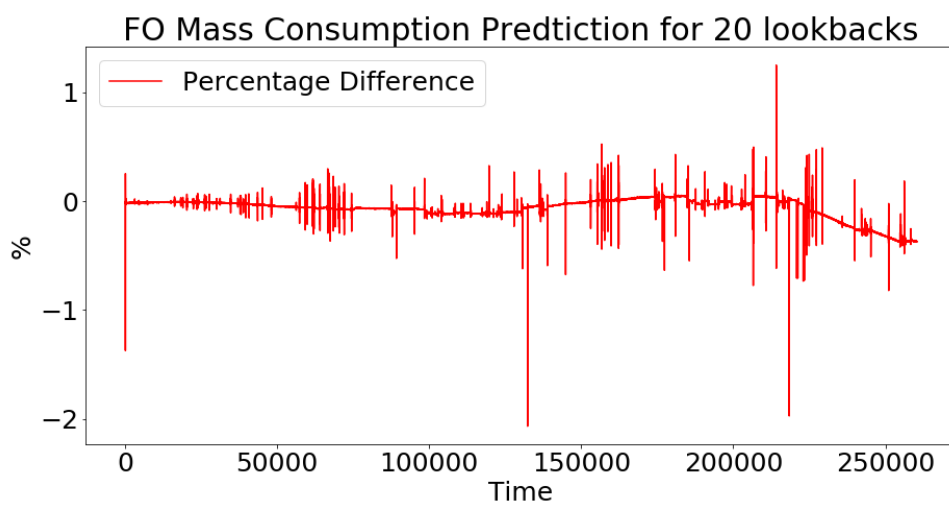


Figure 5.63: Results of the % difference of the *third* ship after 30 epochs for 20 lookback



### 5.2.6 50 Lookbacks

The best results for 50 lookbacks using only one 50 neuron LSTM layer were produced when the model was trained for 10, 20, 30 and 50 epochs for the first ship and for 5 and 10 epochs for the second and for 10, 20, 30 and 50 epochs for the third ship. So overall the best results were produced when the model was trained for 10 epochs, due to the difference between the 20, 30 and 50 epochs in the ability to generalize in the second ship.

From the figures below the following conclusions have been reached:

- Once again the model succeeded to train incredibly well for the first ship, with an error almost 0%.
- On the second ship's dataset the model was able to generalize very well with an error below 1%.
- The model's performance on the third dataset was the same as in the 10 and 15 lookbacks.

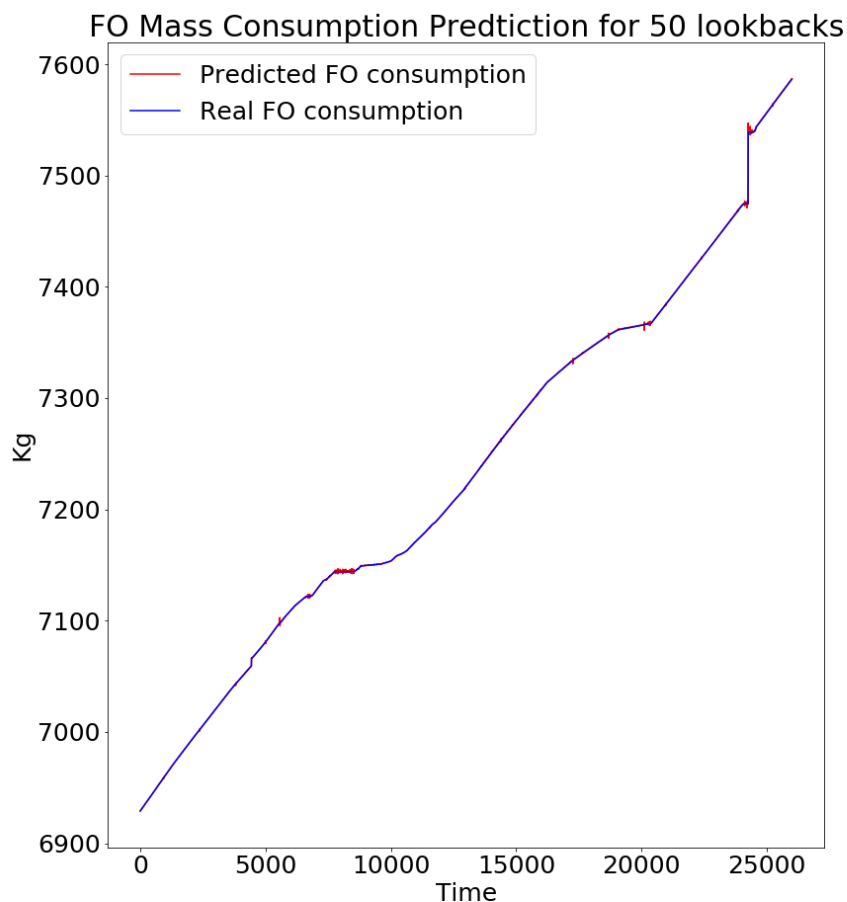


Figure 5.64: Results of the *first* ship after 10 epochs for 50 lookback

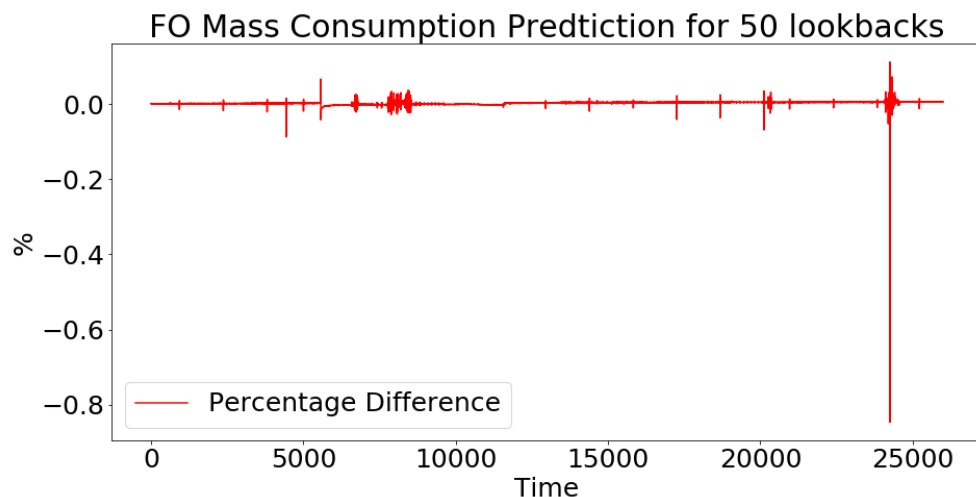


Figure 5.65: Results of the % difference of the *first* ship after 10 epochs for 50 lookback

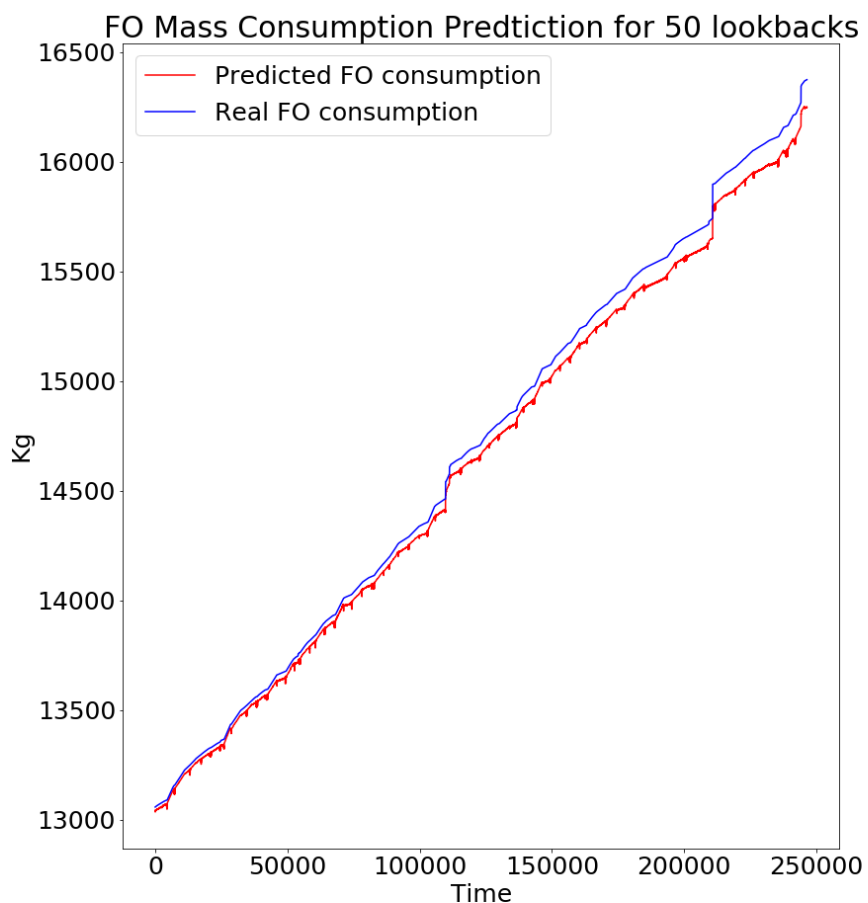


Figure 5.66: Results of the *second* ship after 10 epochs for 50 lookback

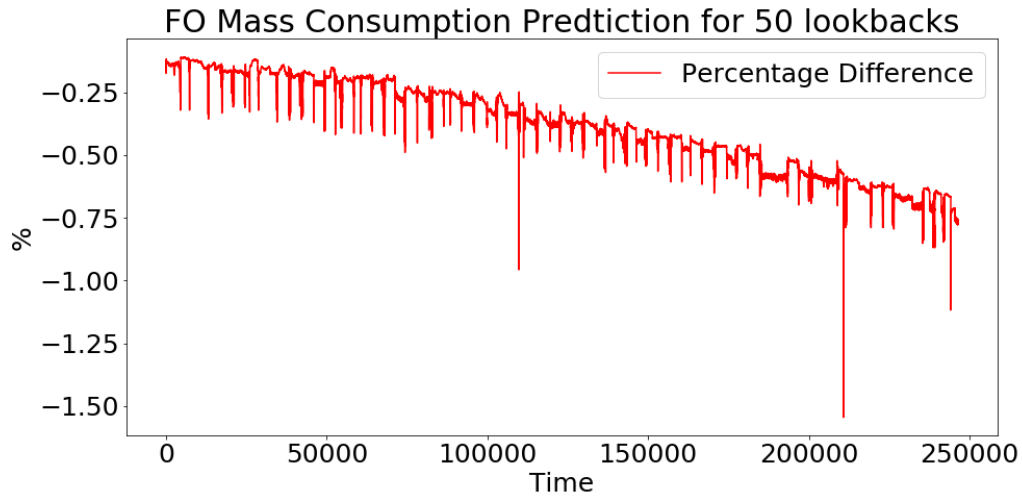


Figure 5.67: Results of the % difference of the *second* ship after 10 epochs for 50 lookback

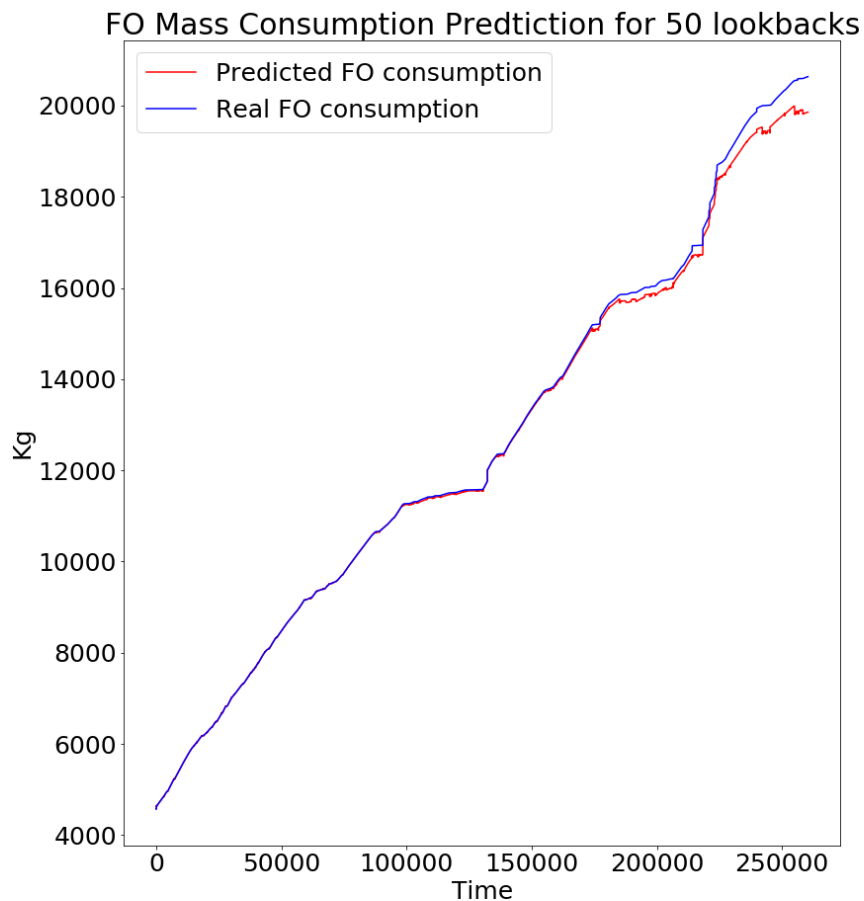


Figure 5.68: Results of the *third* ship after 10 epochs for 50 lookback

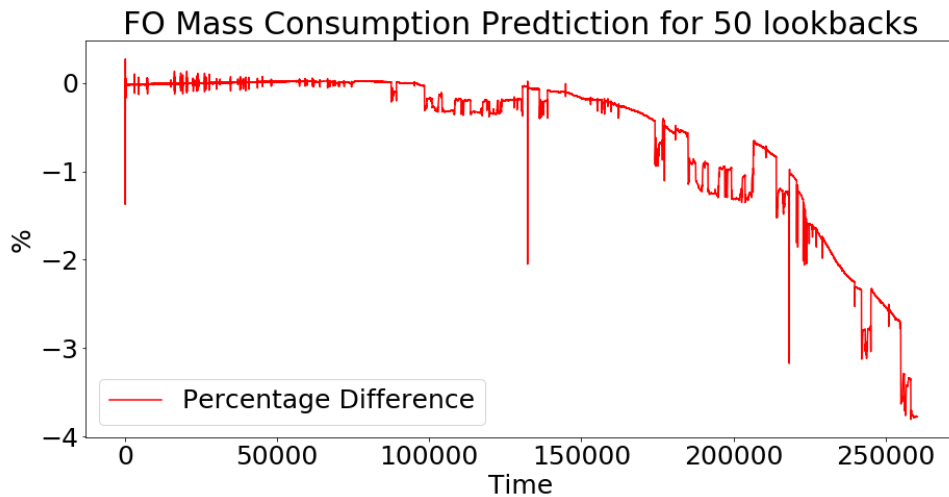


Figure 5.69: Results of the % difference of the *third* ship after 10 epochs for 50 lookback

### 5.3 Three variables: Fuel Oil Consumption, Fuel Oil Temperature and Main Engine RPM

Since the addition of only one extra variable did not improve the performance of the model one more variable was tested to see if it would achieve any improvement. The new variable was the ME rpm. This variable was chosen because there is a non linear connection between it and the FOC. In the figure 5.70 the new layout of the NN is presented.

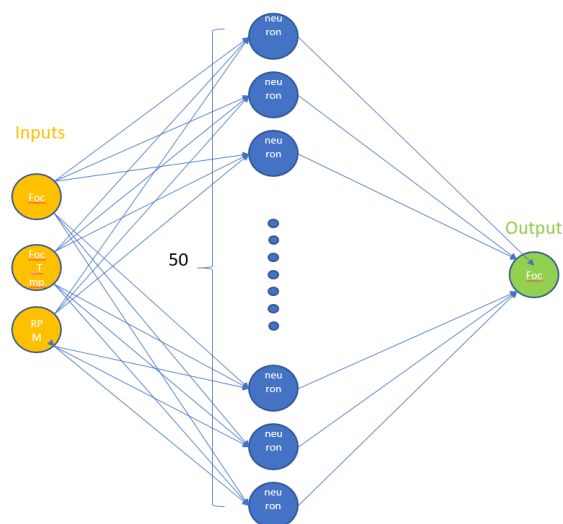


Figure 5.70: A NN with three inputs(FOC,FOC.tmp ,ME.rpm) one layer with 50 neuros and one output(FOC)

### 5.3.1 1 Lookback

The best results for one lookback using only one 50 neuron LSTM layer were produced when the model was trained for 5, 20 and 30 epochs for the first ship and for 5 and 15 epochs for the second and for 5 and 15 epochs for the third ship. So overall the best results were produced when the model was trained for 5 epochs.

In the figures below the following conclusions have been reached:

- First of all the model succeeded to train incredibly well for the first ship, with an error around 0%.
- Secondly, the model's performance on the second dataset was decent even that at the end it was around  $-2\%$ .
- Lastly, the model was able to generalize very well on the first half of the third dataset, with an error below  $-1\%$  but then the error increased to around  $-4\%$ .

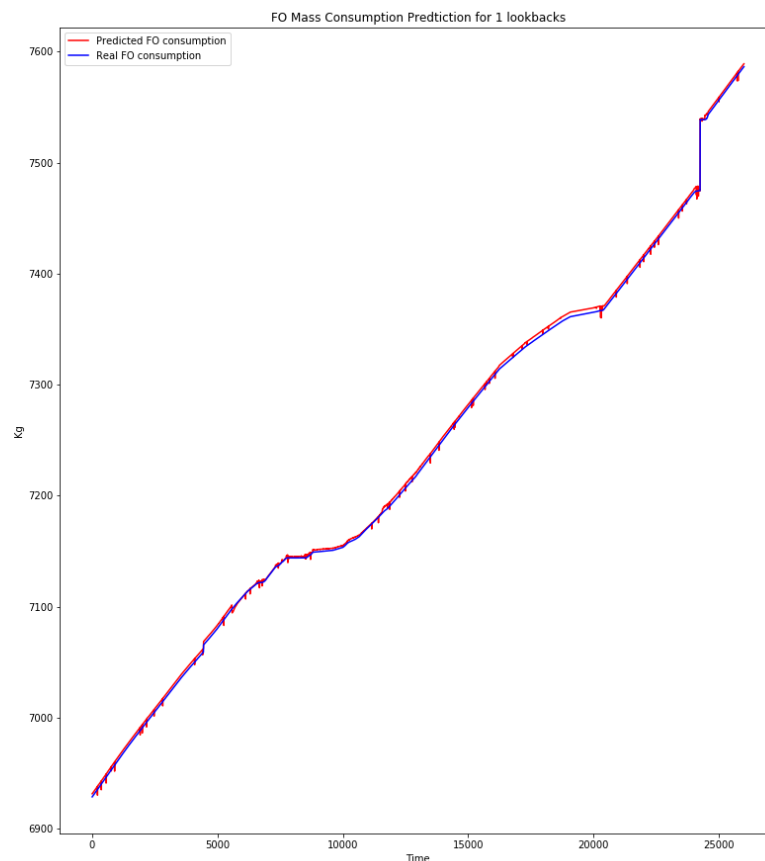


Figure 5.71: Results of the *first* ship after 5 epochs for 1 lookback

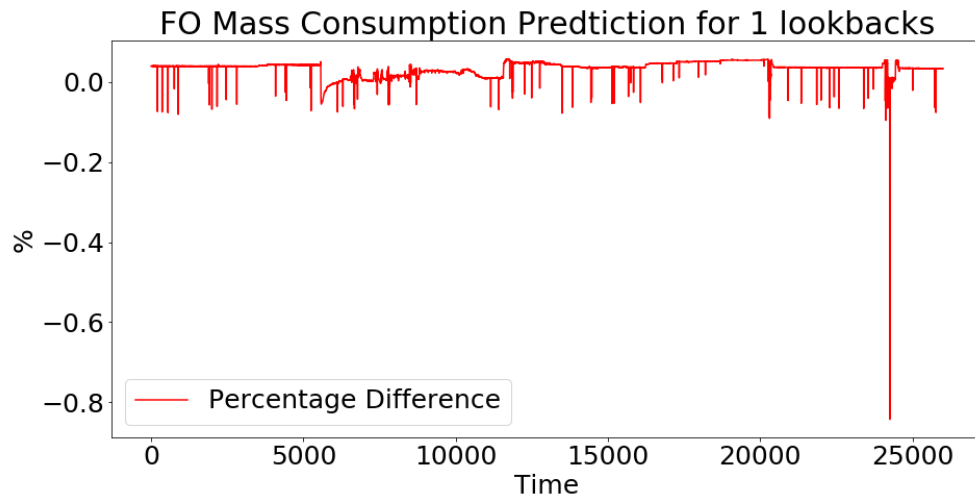


Figure 5.72: Results of the % difference of the *first* ship after 5 epochs for 1 lookback

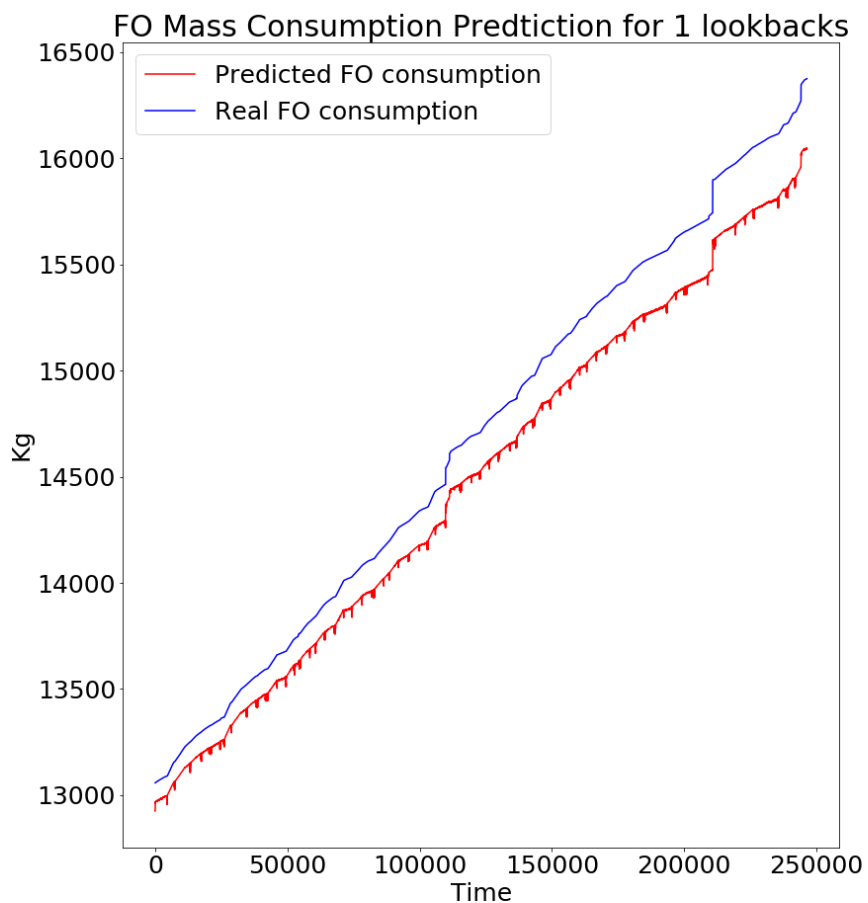


Figure 5.73: Results of the *first* ship after 5 epochs for 1 lookback

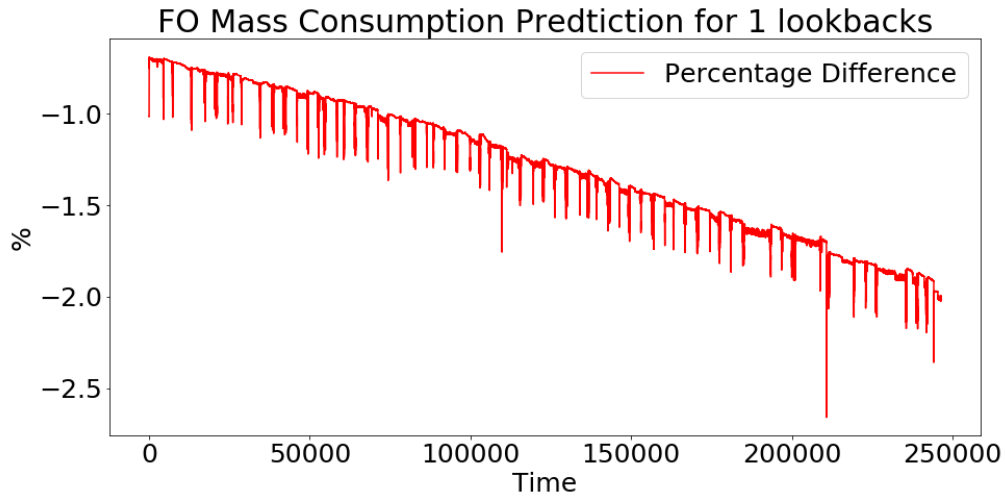


Figure 5.74: Results of the % difference of the *second* ship after 5 epochs for 1 lookback

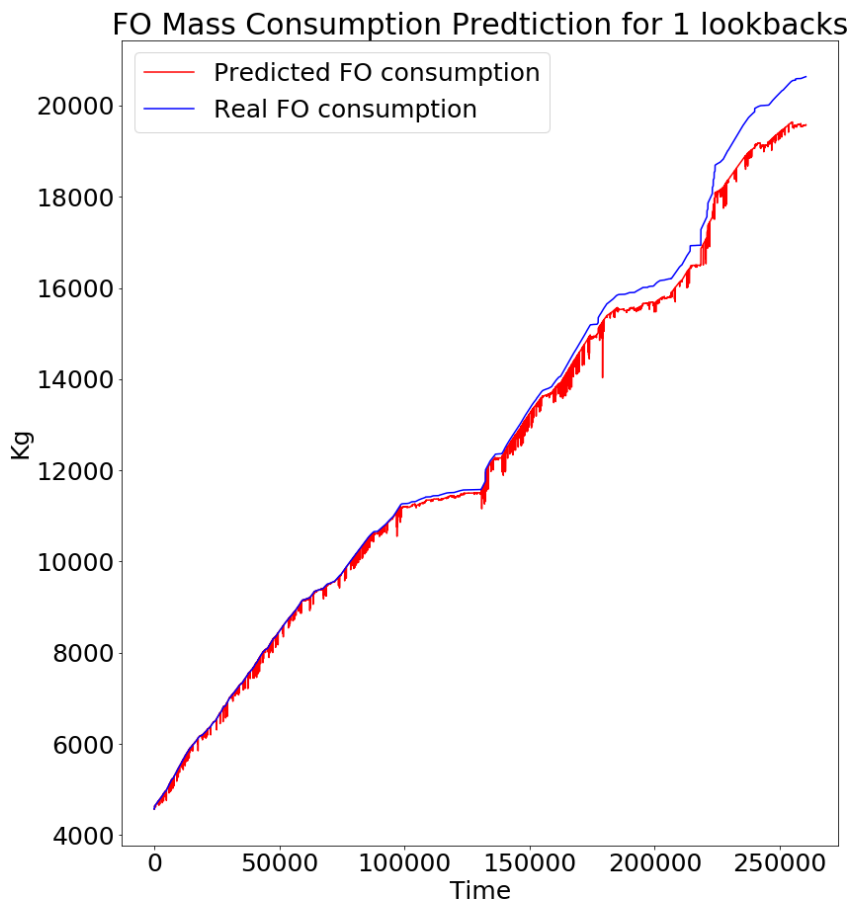


Figure 5.75: Results of the *third* ship after 5 epochs for 1 lookback

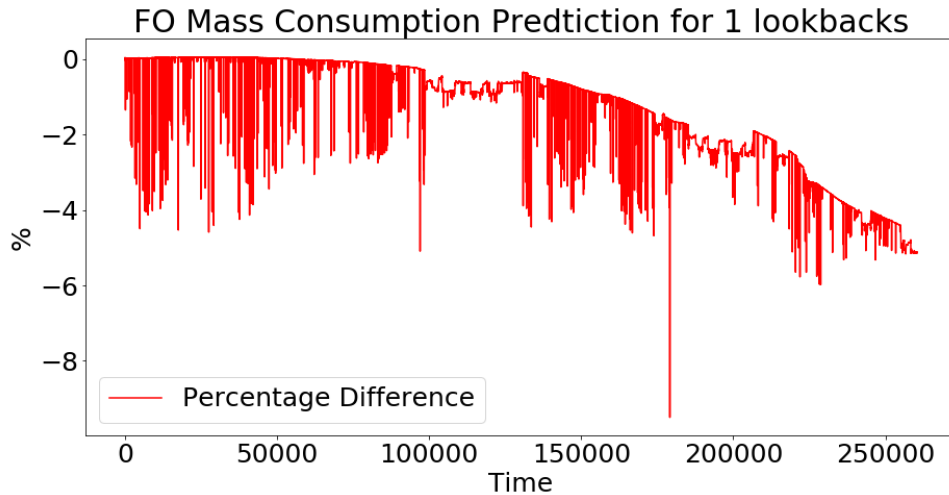


Figure 5.76: Results of the % difference of the *third* ship after 5 epochs for 1 lookback

### 5.3.2 5 Lookbacks

The best results for 5 lookbacks using only one 50 neuron LSTM layer were produced when the model was trained for 15, 20 and 30 epochs for the first ship and for 20 and 30 epochs for the second and for 15 and 30 epochs for the third ship. So overall the best results were produced when the model was trained for 30 epochs.

From the figures below the following conclusions have been reached:

- First of all, as in the 1 lookback the model succeeded to train incredibly well for the first ship, with an error around 0%.
- However, it failed to generalize so well for the second ship(sister ship) with the error starting from 3% and ending around -6%.
- On the third dataset however, the model was able to generalize on the first half but then failed on the second half with the error ending being -12%.



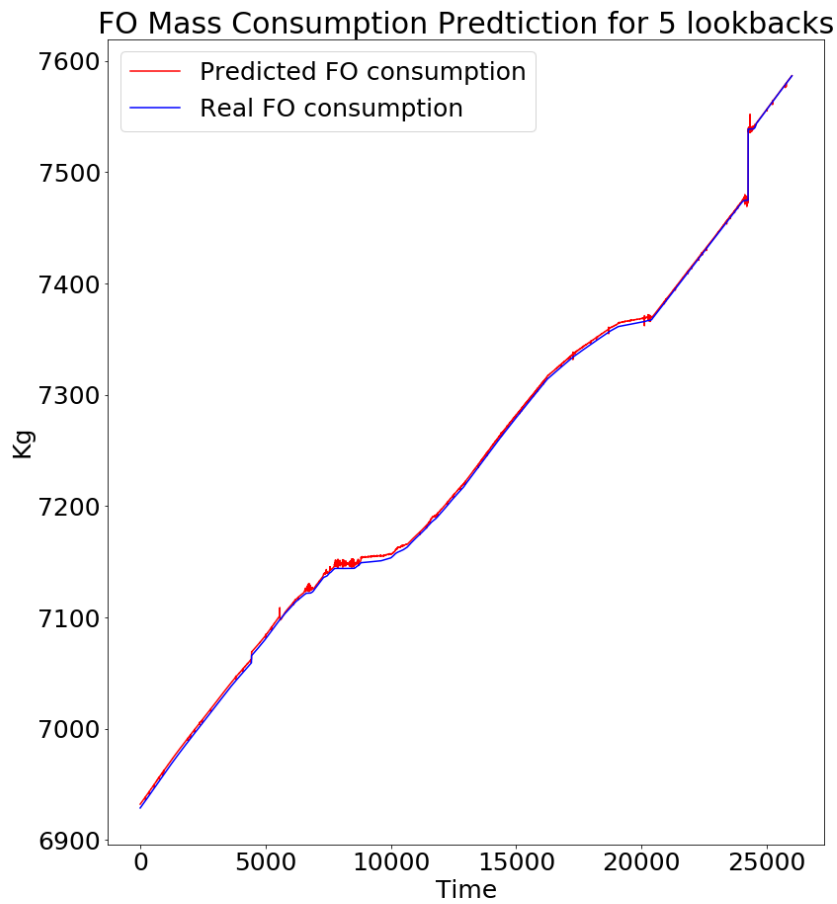


Figure 5.77: Results of the *first* ship after 30 epochs for 5 lookback

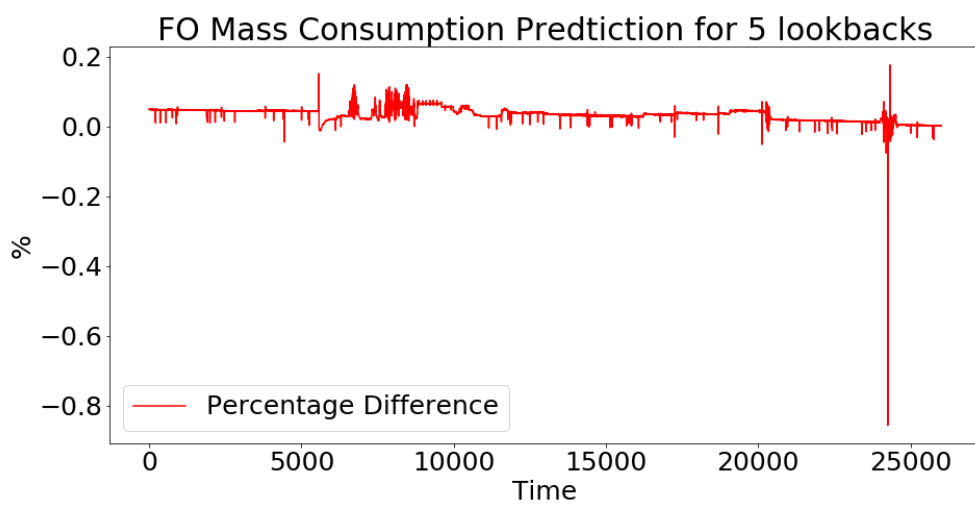


Figure 5.78: Results of the % difference of the *first* ship after 30 epochs for 5 lookback

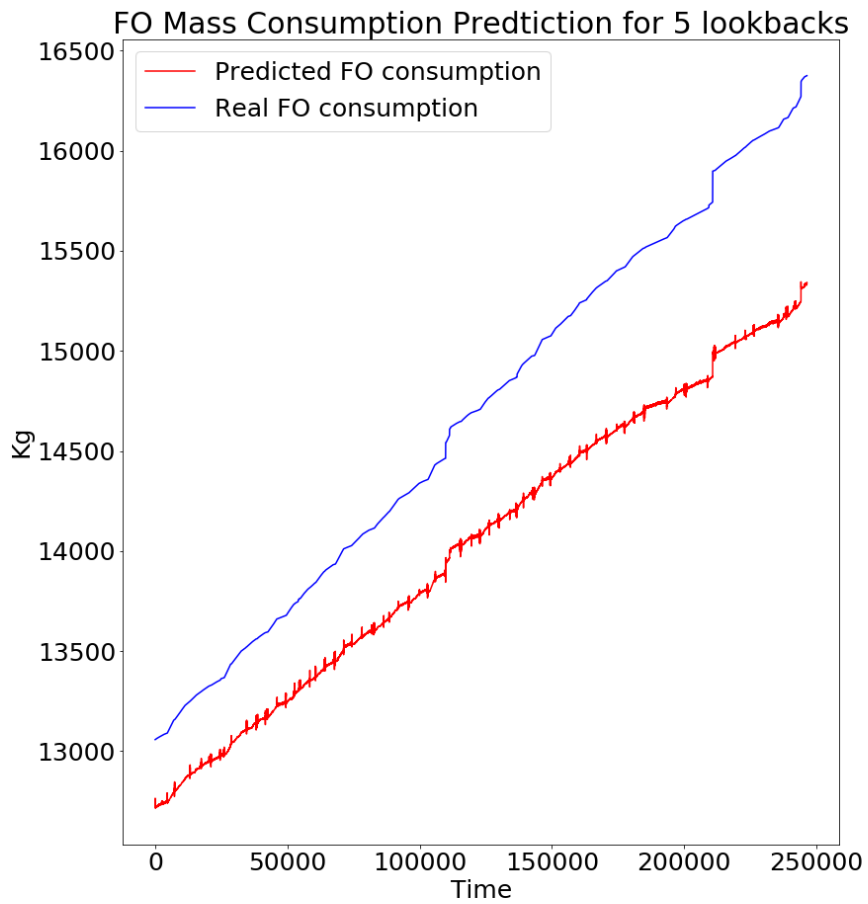


Figure 5.79: Results of the *second* ship after 30 epochs for 5 lookback

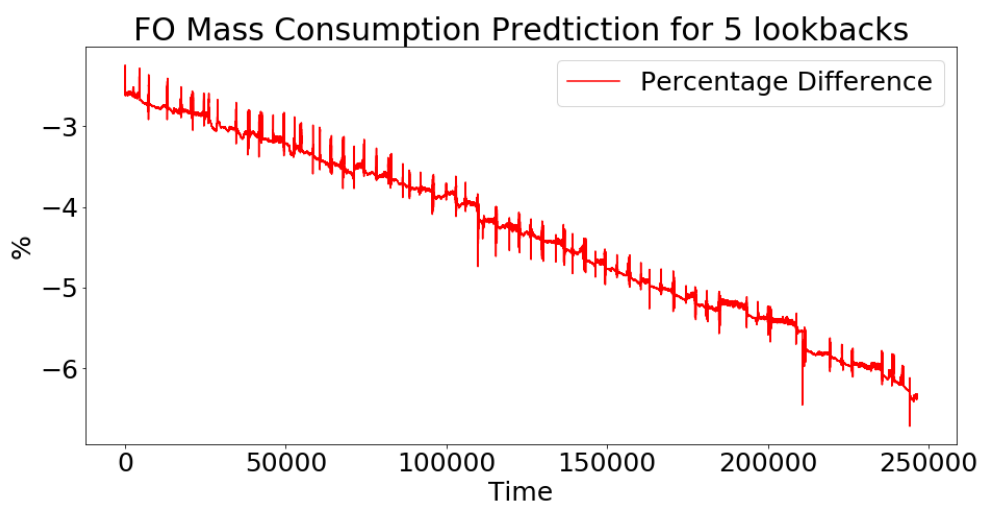


Figure 5.80: Results of the % difference of the *second* ship after 30 epochs for 5 lookback

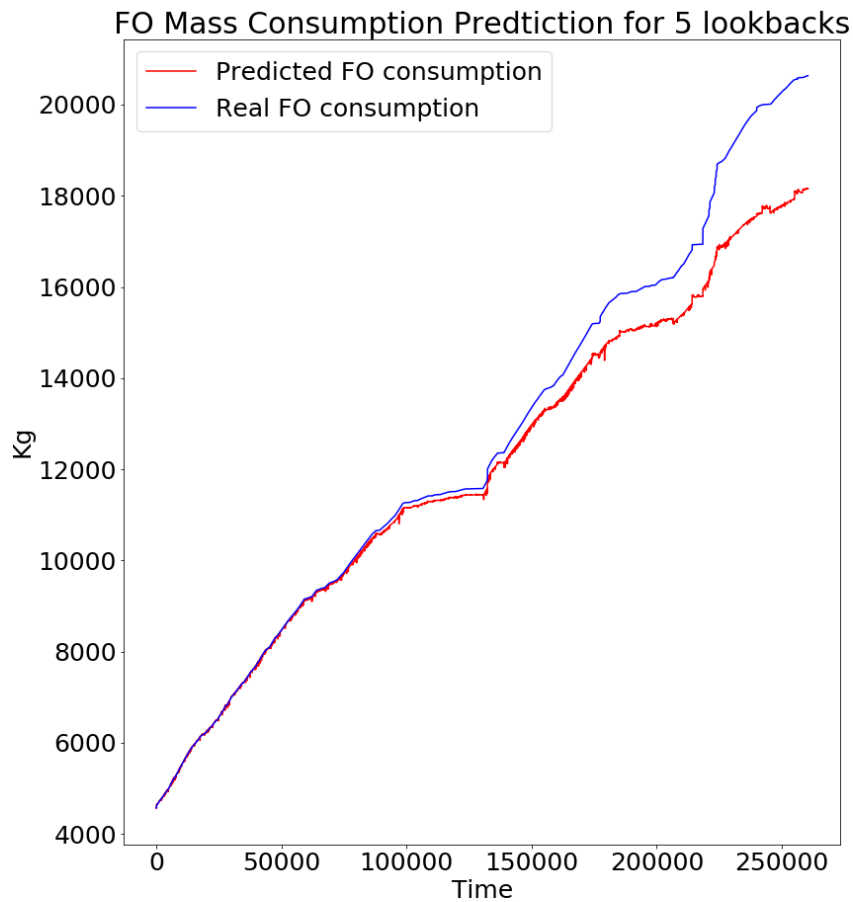


Figure 5.81: Results of the *third* ship after 30 epochs for 5 lookback

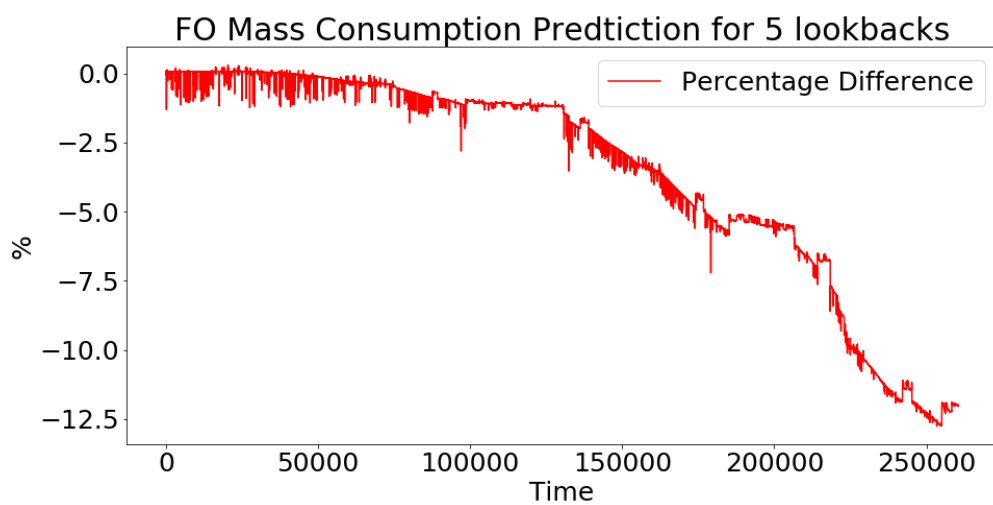


Figure 5.82: Results of the % difference of the *third* ship after 30 epochs for 5 lookback

### 5.3.3 10 Lookbacks

The best results for 10 lookbacks using only one 50 neuron LSTM layer were produced when the model was trained for 10, 30, 50 and 50 epochs for the first ship and for 15, 30 and 50 epochs for the second and third ship, with 50 having the best. So overall the best results were produced when the model was trained for 50 epochs.

From the figures below the following conclusions have been reached:

- Once again the model was able to train very well for the first ship, with an error around 0%.
- The model's results for the second and the third ship were satisfactory, since it appears the model was able to generalize but not as well as for 1 lookback.

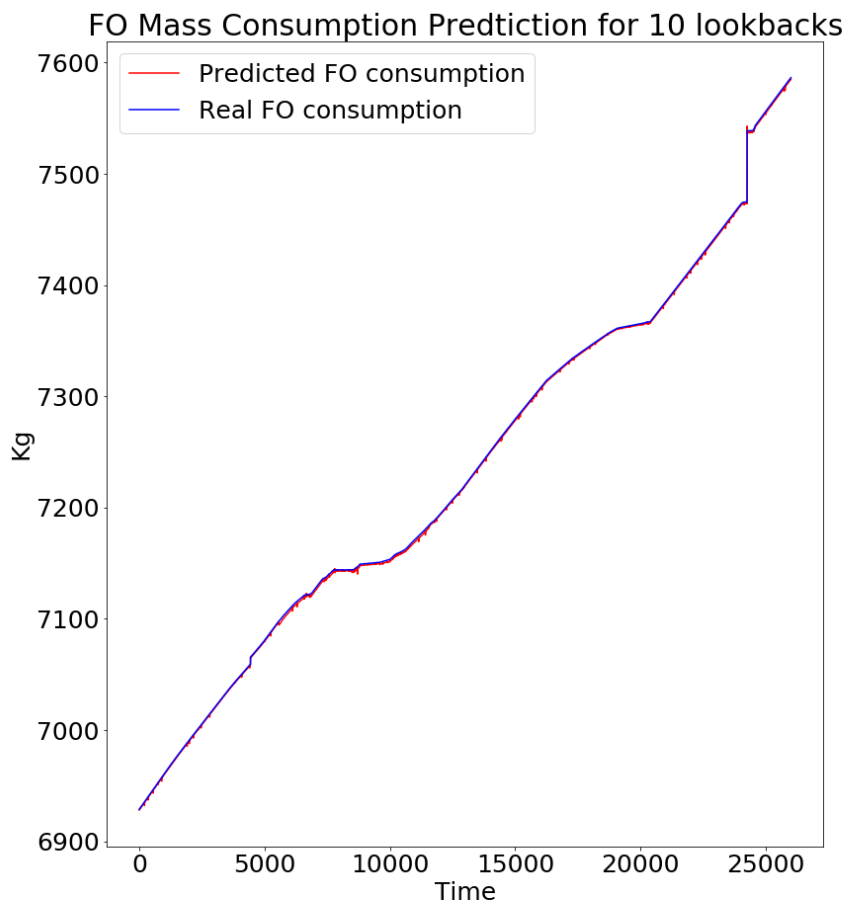


Figure 5.83: Results of the *first* ship after 50 epochs for 10 lookback

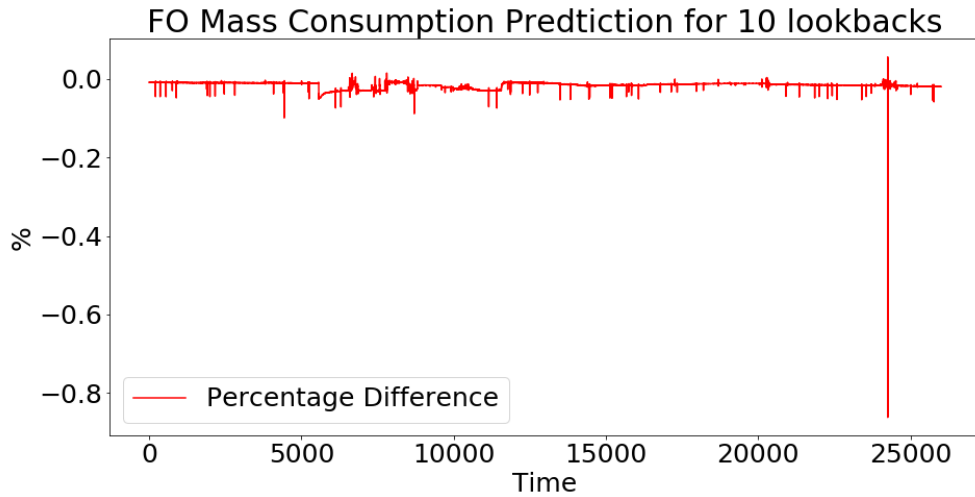


Figure 5.84: Results of the % difference of the *first* ship after 50 epochs for 10 lookback

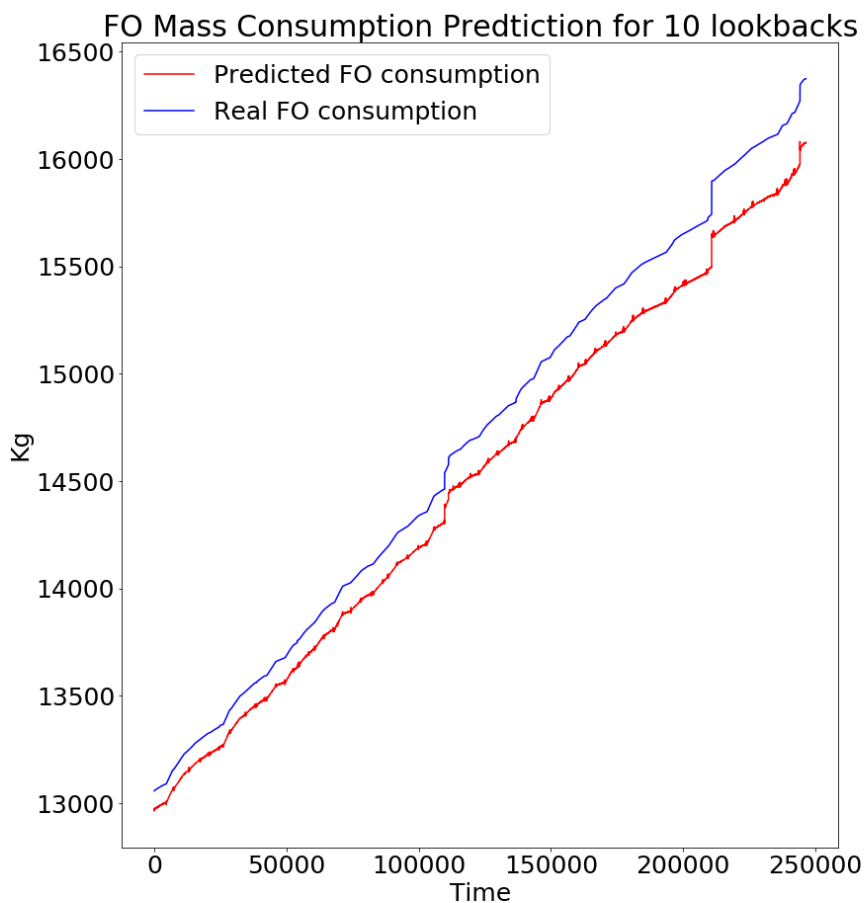


Figure 5.85: Results of the *second* ship after 50 epochs for 10 lookback

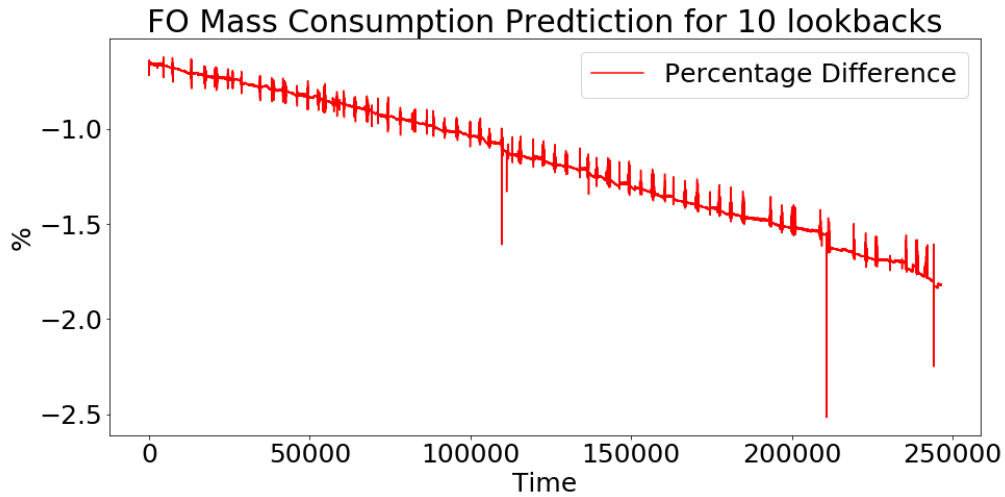


Figure 5.86: Results of the % difference of the *second* ship after 50 epochs for 10 lookback

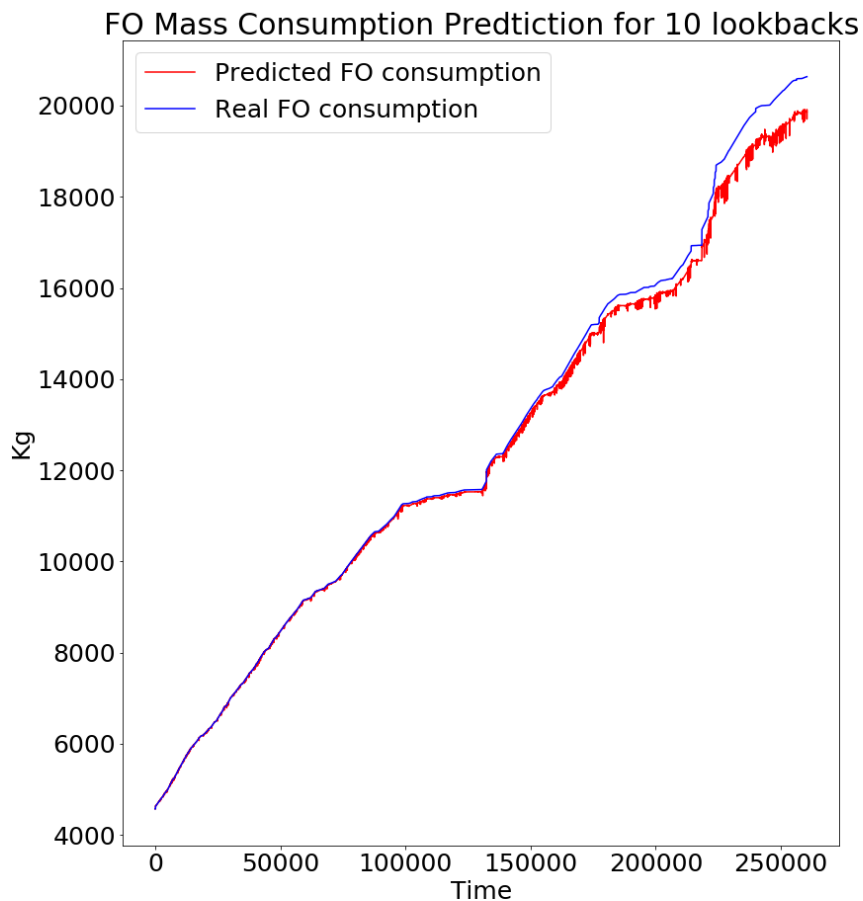


Figure 5.87: Results of the *third* ship after 50 epochs for 10 lookback

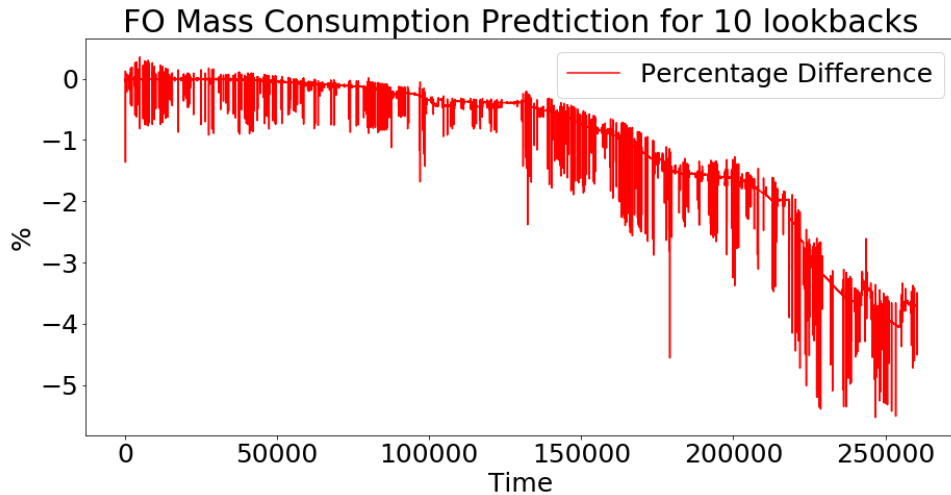


Figure 5.88: Results of the % difference of the *third* ship after 50 epochs for 10 lookback

#### 5.3.4 15 Lookbacks

The best results for 10 lookbacks using only one 50 neuron LSTM layer were produced when the model was trained for 10, 20, 30 and 50 epochs for the first ship and for 5 and 15 epochs for the second and for 10 and 15 epochs for the third ship. So overall the best results were produced when the model was trained for 10 epochs.

From the figures below the following conclusions have been reached:

- Once again the model was able to train very well for the first ship, with an error around 0%.
- On the second dataset, the model was able to generalize pretty well with an error between  $-0.5\%$  and  $-1\%$ .
- Lastly, the model was able to generalize very well on the first half of the third dataset, with an error below  $-1\%$  but then the error increased to around  $-2.5\%$ .

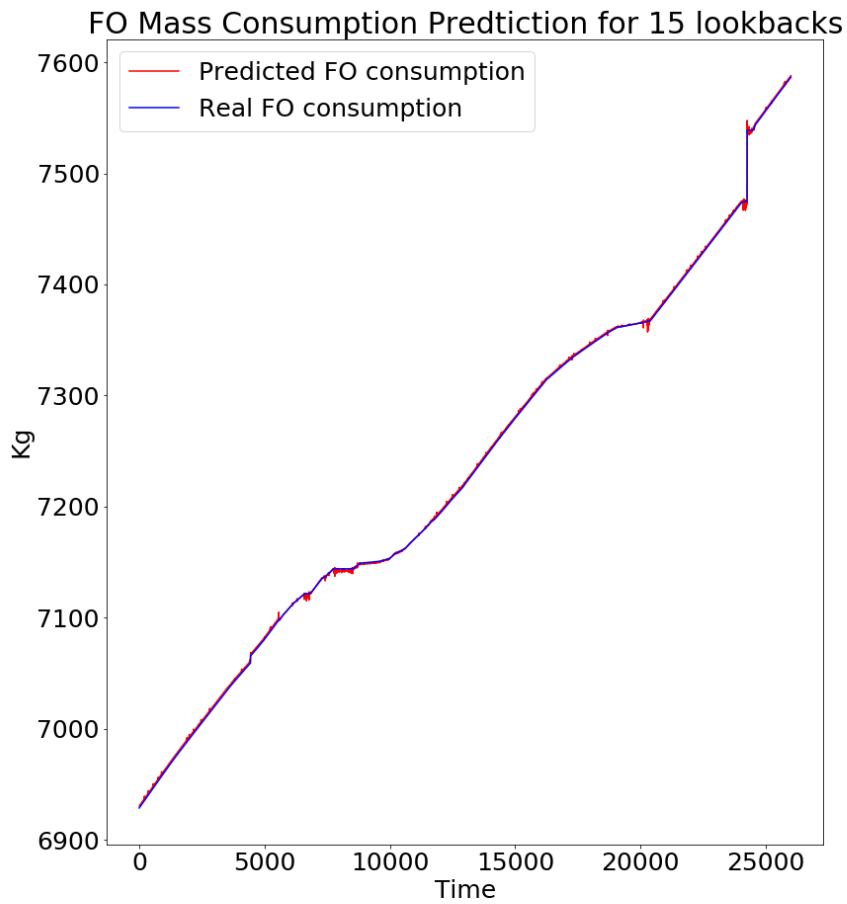


Figure 5.89: Results of the *first* ship after 10 epochs for 15 lookback

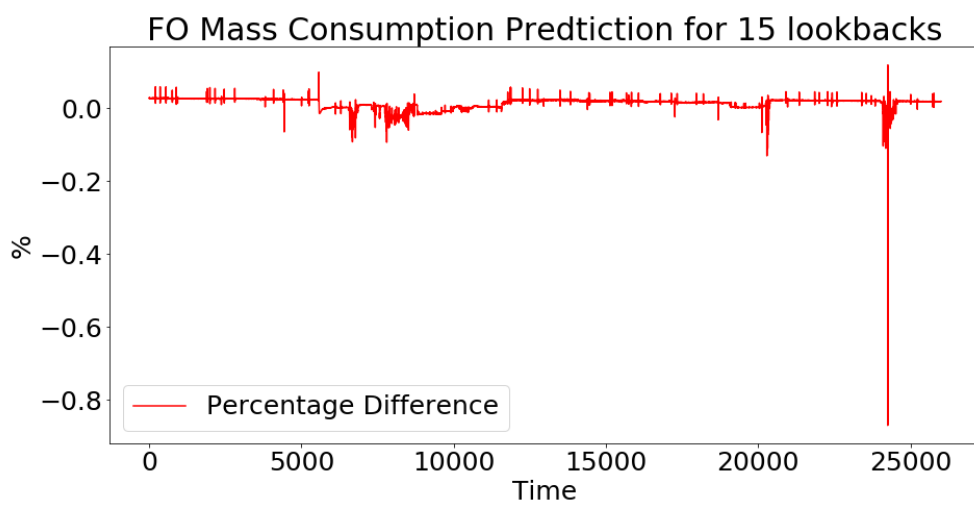


Figure 5.90: Results of the % difference of the *first* ship after 10 epochs for 15 lookback



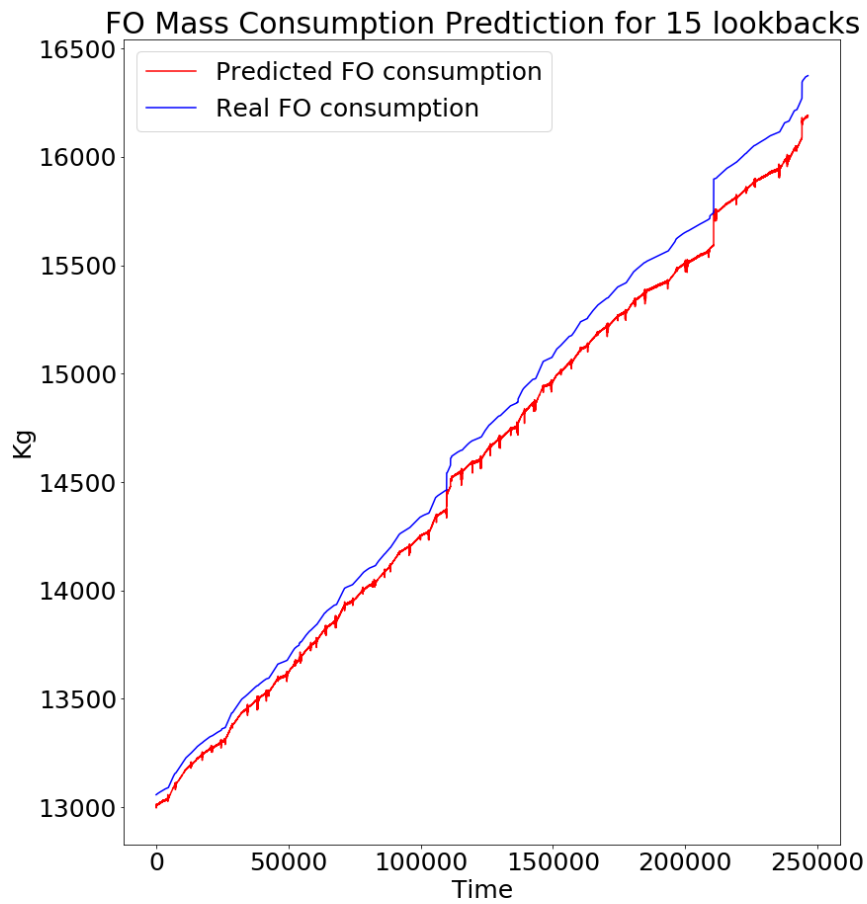


Figure 5.91: Results of the *second* ship after 10 epochs for 15 lookback

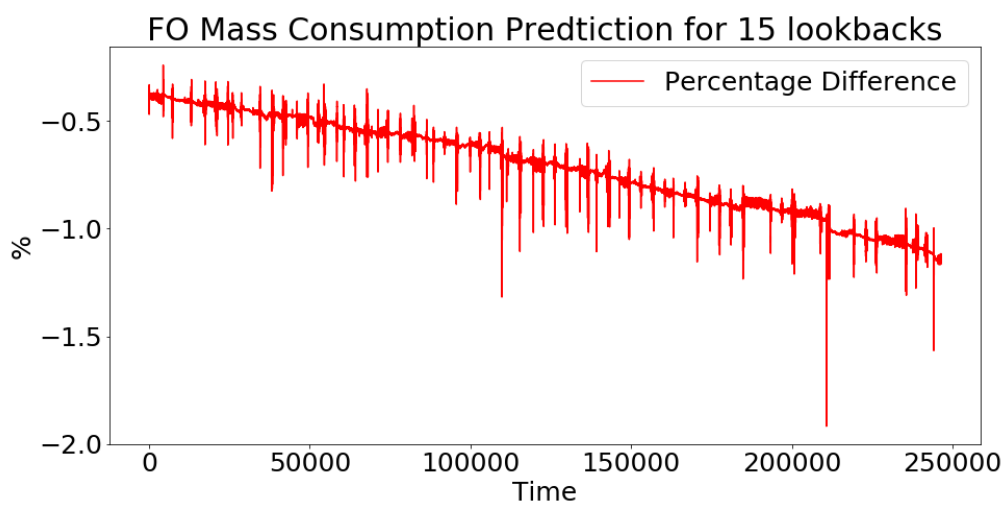


Figure 5.92: Results of the % difference of the *second* ship after 10 epochs for 15 lookback

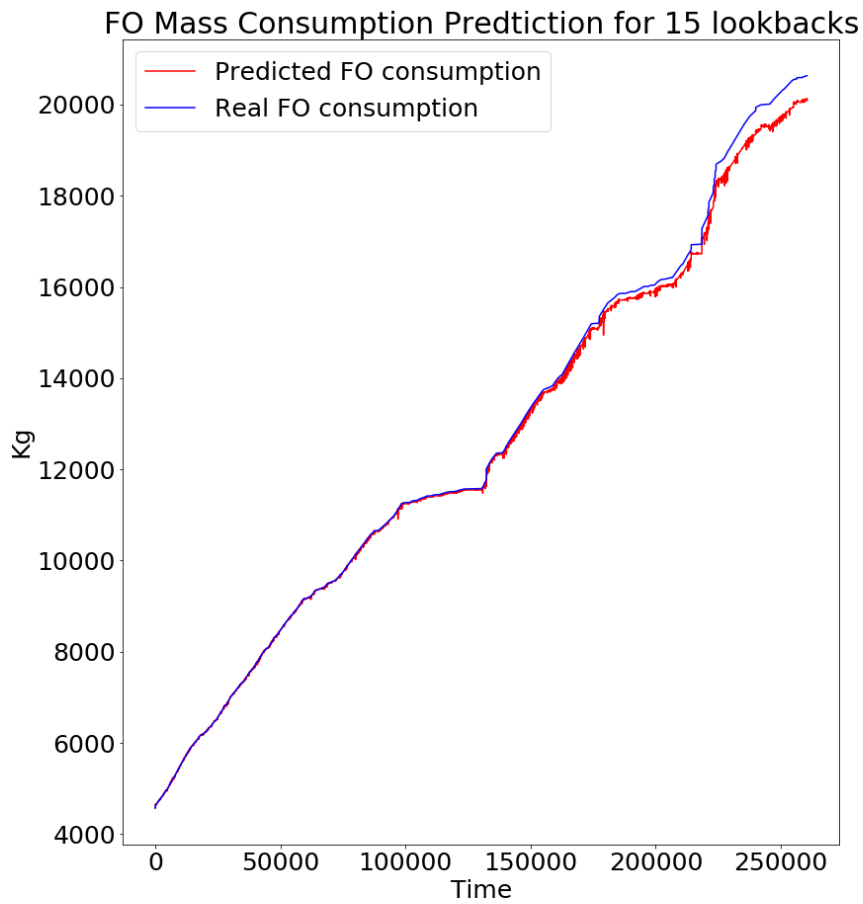


Figure 5.93: Results of the *third* ship after 10 epochs for 15 lookback

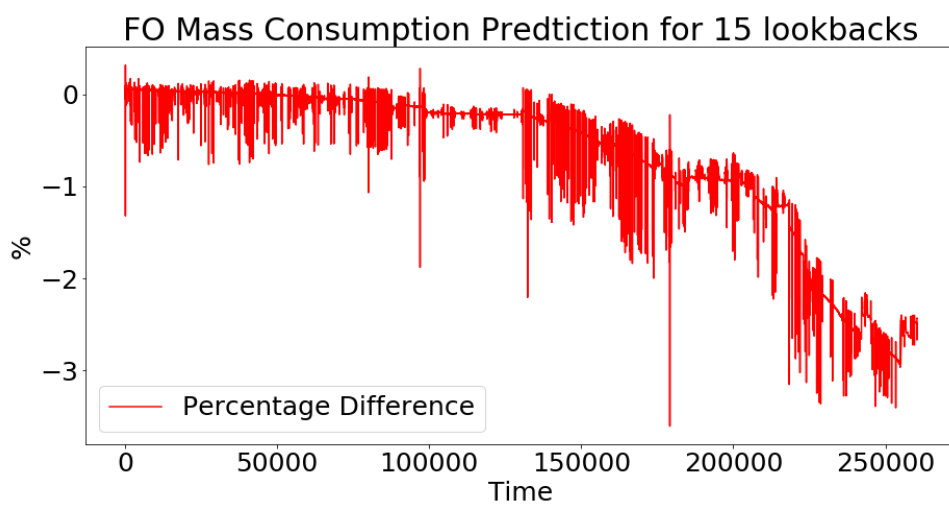


Figure 5.94: Results of the % difference of the *third* ship after 10 epochs for 15 lookback

### 5.3.5 20 Lookbacks

The best results for 20 lookbacks using only one 50 neuron LSTM layer were produced when the model was trained for 20, 30 and 50 epochs for the first ship and for 20 and 30 epochs for the second and for 30 epochs for the third ship. So overall the best results were produced when the model was trained for 30 epochs.

From the figures below the following conclusions have been reached:

- Once again the model succeeded to train incredibly well for the first ship, with an error almost 0%.
- Moreover, the model achieve an incredible performance on the second ship's dataset with an error approximately 0.2%, even though as it can be seen from the plot it overshooted a little bit in the last part of the dataset.
- The most impressive result though, was that on the third ship's dataset, where the error was between  $-0.5\%$  and  $0.5\%$  despite the oscillation.

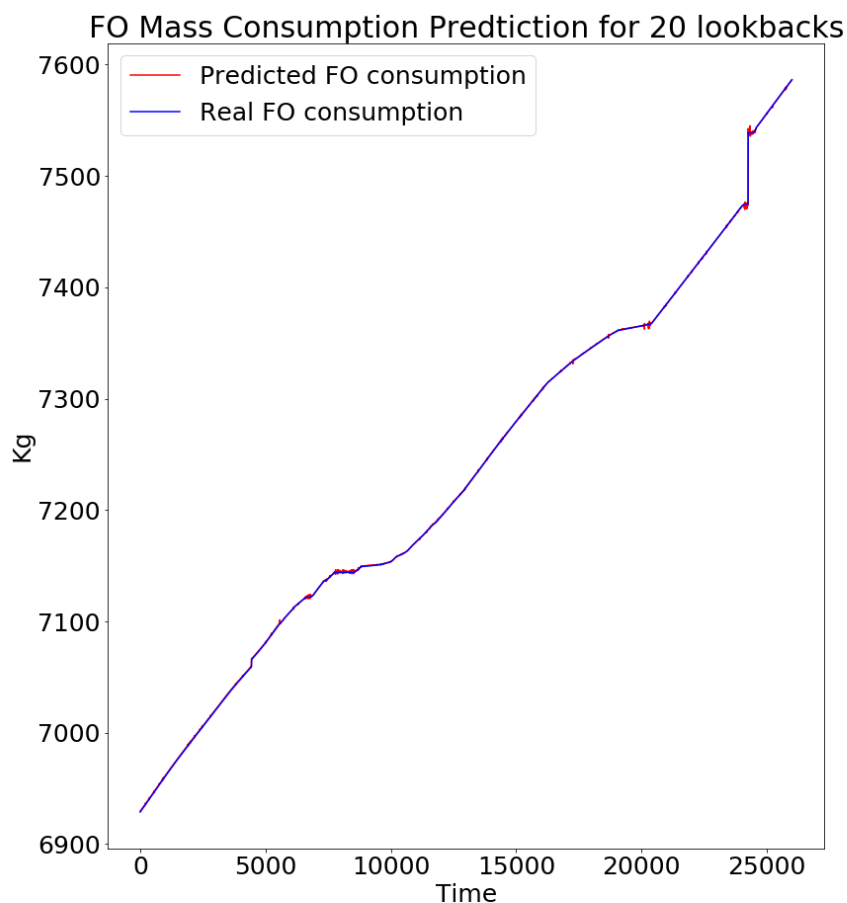


Figure 5.95: Results of the *first* ship after 30 epochs for 20 lookback

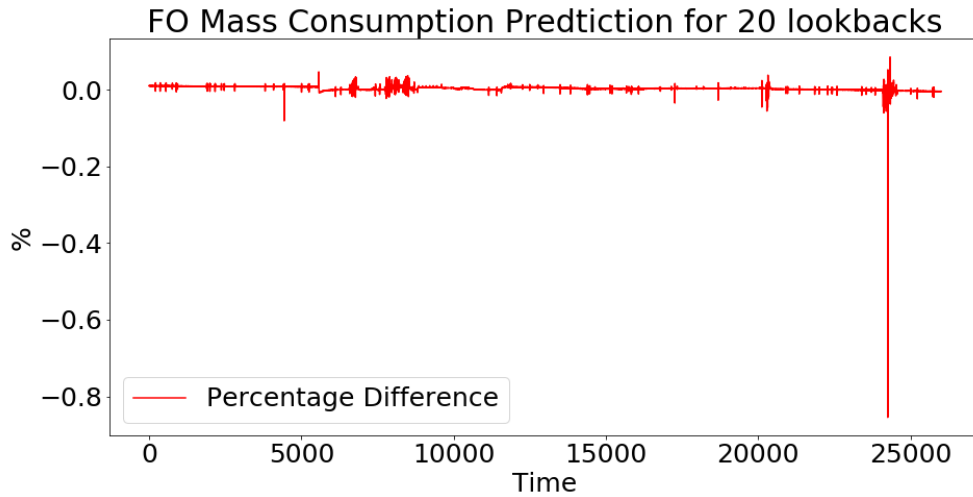


Figure 5.96: Results of the % difference of the *first* ship after 30 epochs for 20 lookback

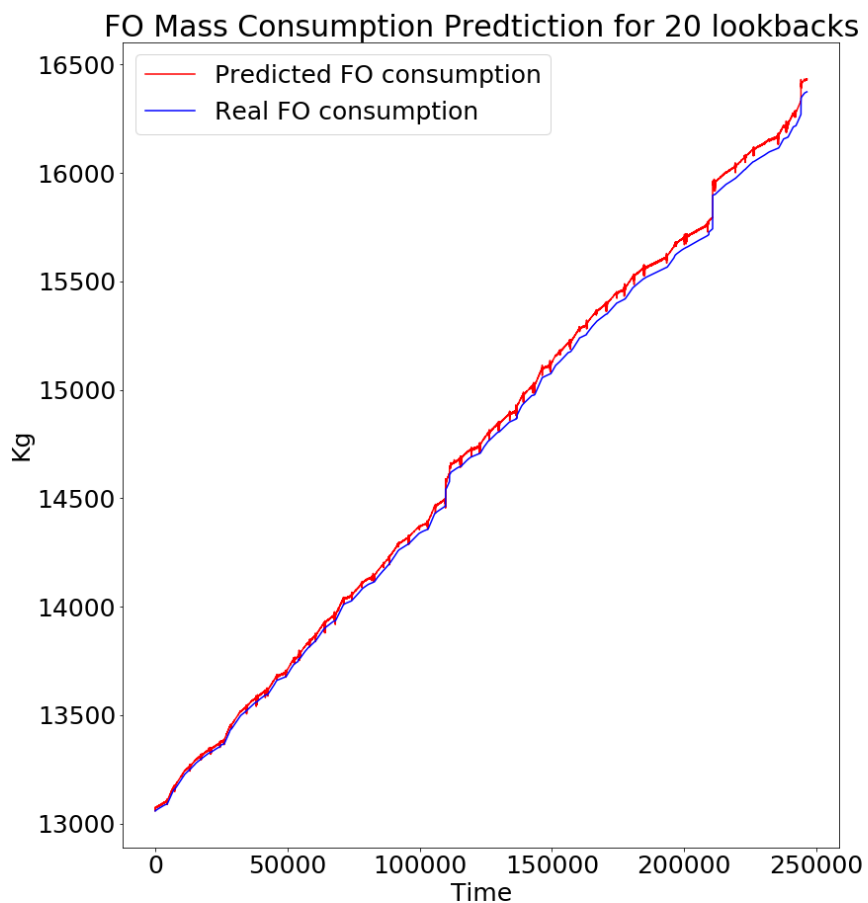


Figure 5.97: Results of the *second* ship after 30 epochs for 20 lookback

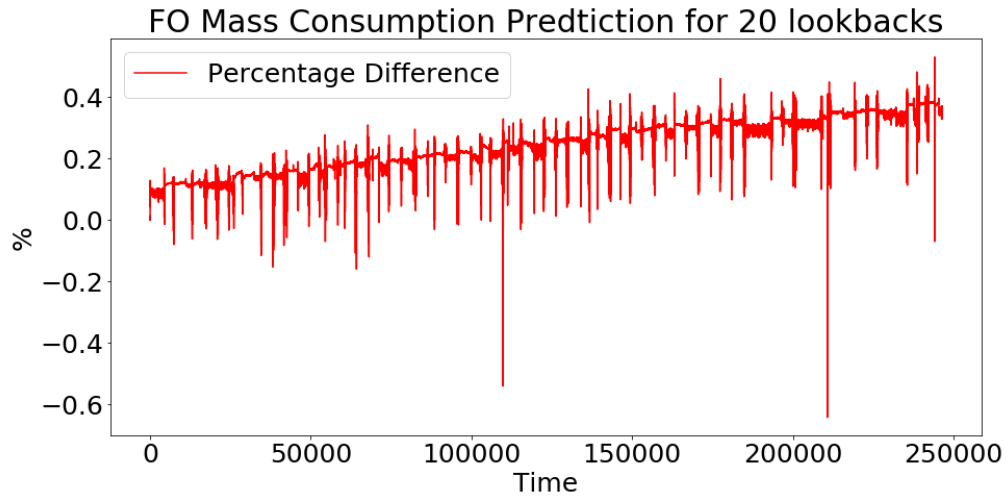


Figure 5.98: Results of the % difference of the *second* ship after 30 epochs for 20 lookback

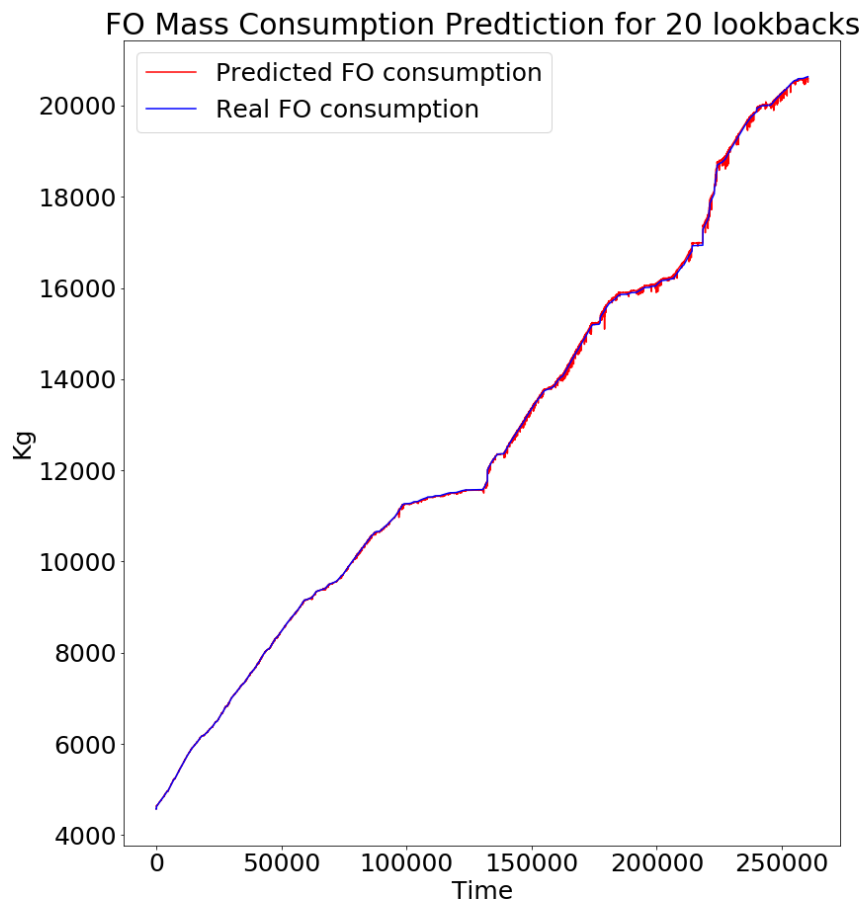


Figure 5.99: Results of the *third* ship after 30 epochs for 20 lookback

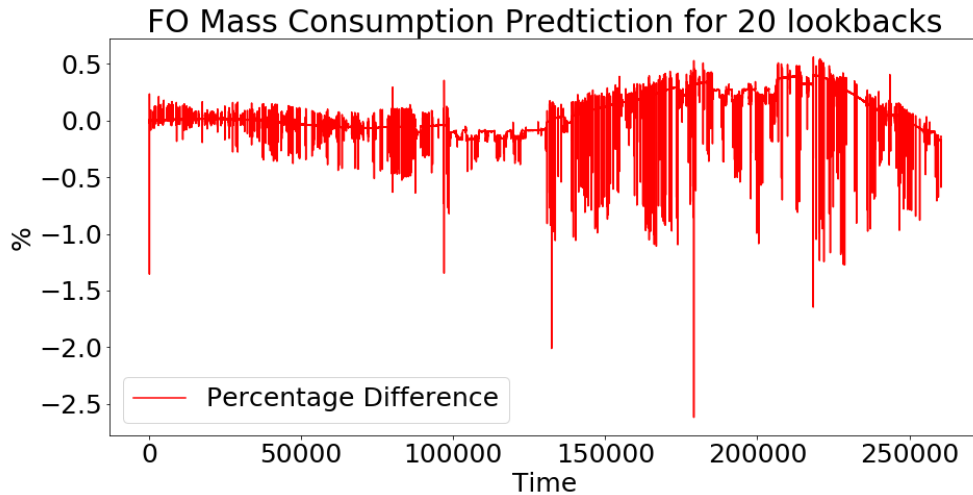


Figure 5.100: Results of the % difference of the *third* ship after 30 epochs for 20 lookback

### 5.3.6 50 Lookbacks

The best results for 50 lookbacks using only one 50 neuron LSTM layer were produced when the model was trained for 20, 30 and 50 epochs for the first ship and for 50 epochs for the second and third ship. So overall the best results were produced when the model was trained for 50 epochs.

From the figures below the following conclusions have been reached:

- Once again the model was able to train very well for the first ship, with an error around 0%.
- On the second dataset, the model was able to generalize pretty well with an error between 0% and -0.5%.
- Lastly, the model was able to generalize very well on the first half of the third dataset, with an error below -1% but then the error increased to around -2.5%.

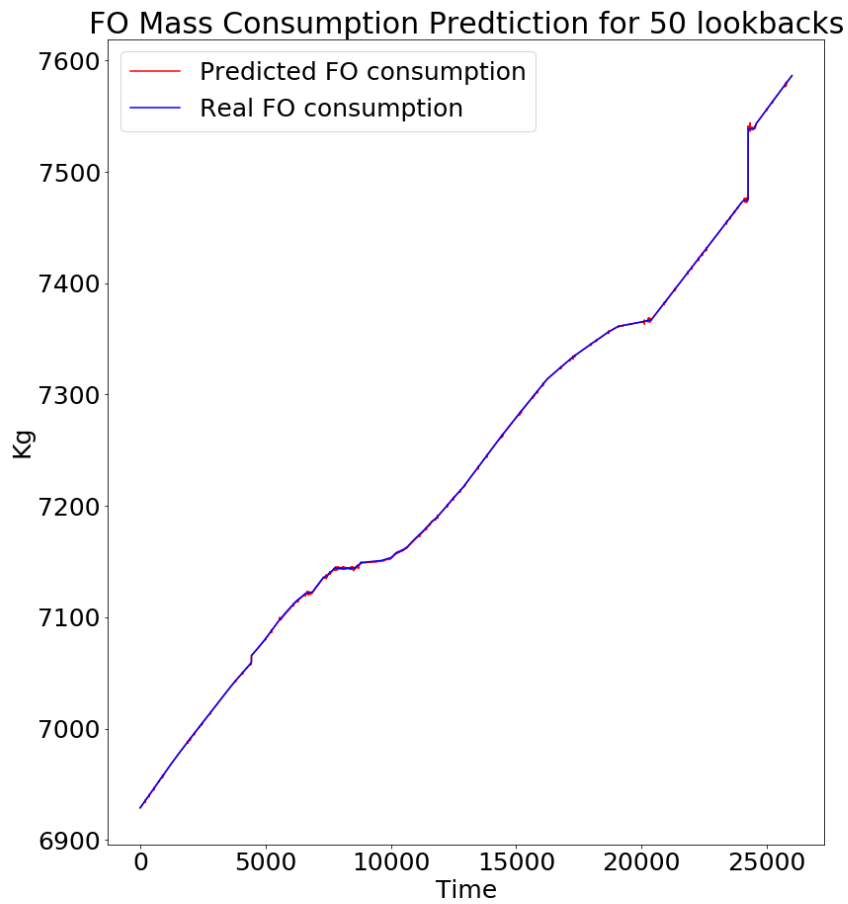


Figure 5.101: Results of the *first* ship after 50 epochs for 50 lookback

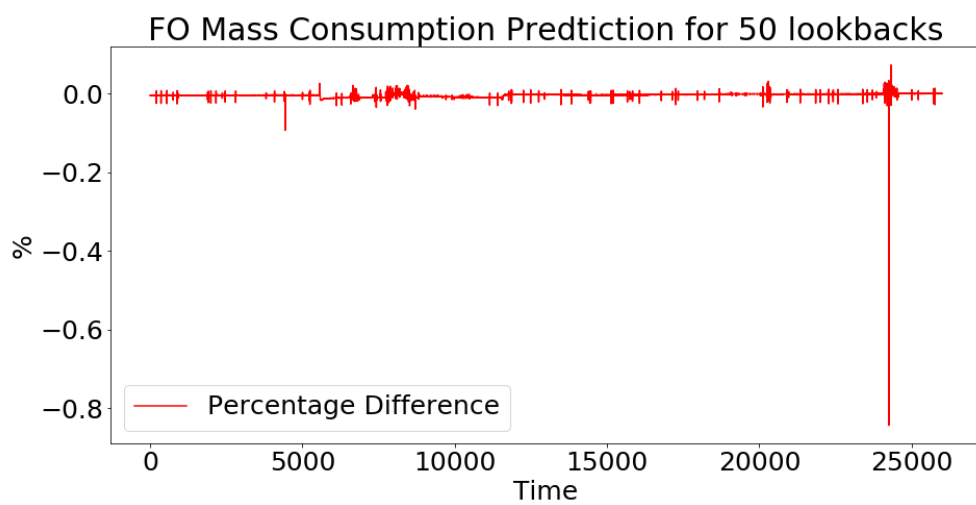


Figure 5.102: Results of the % difference of the *first* ship after 50 epochs for 50 lookback

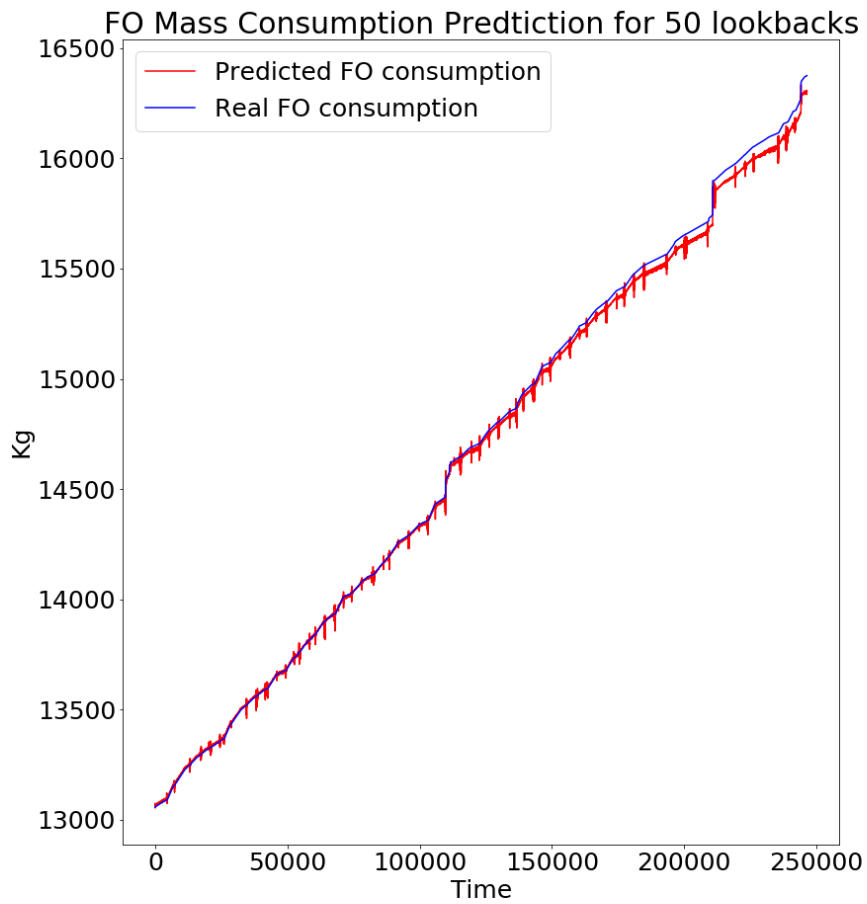


Figure 5.103: Results of the *second* ship after 50 epochs for 50 lookback

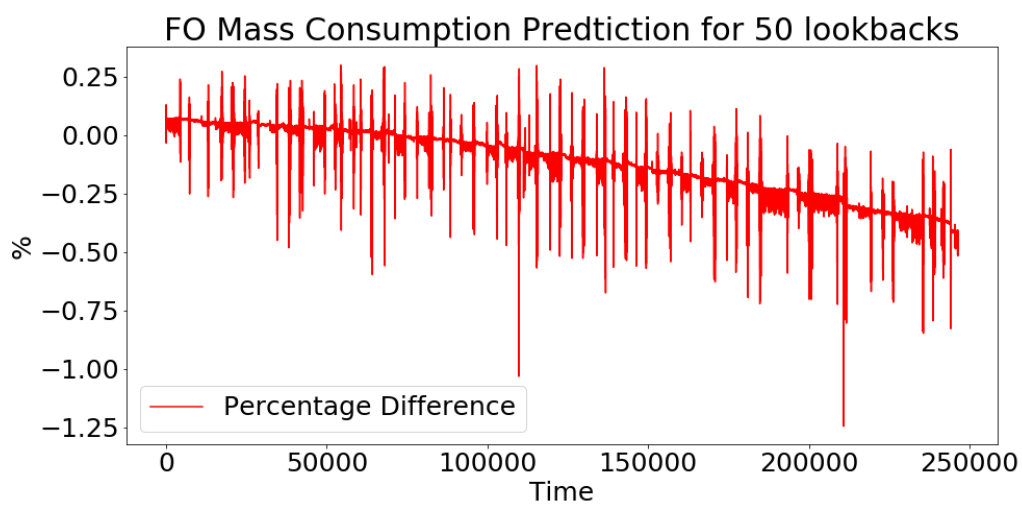


Figure 5.104: Results of the % difference of the *second* ship after 50 epochs for 50 lookback



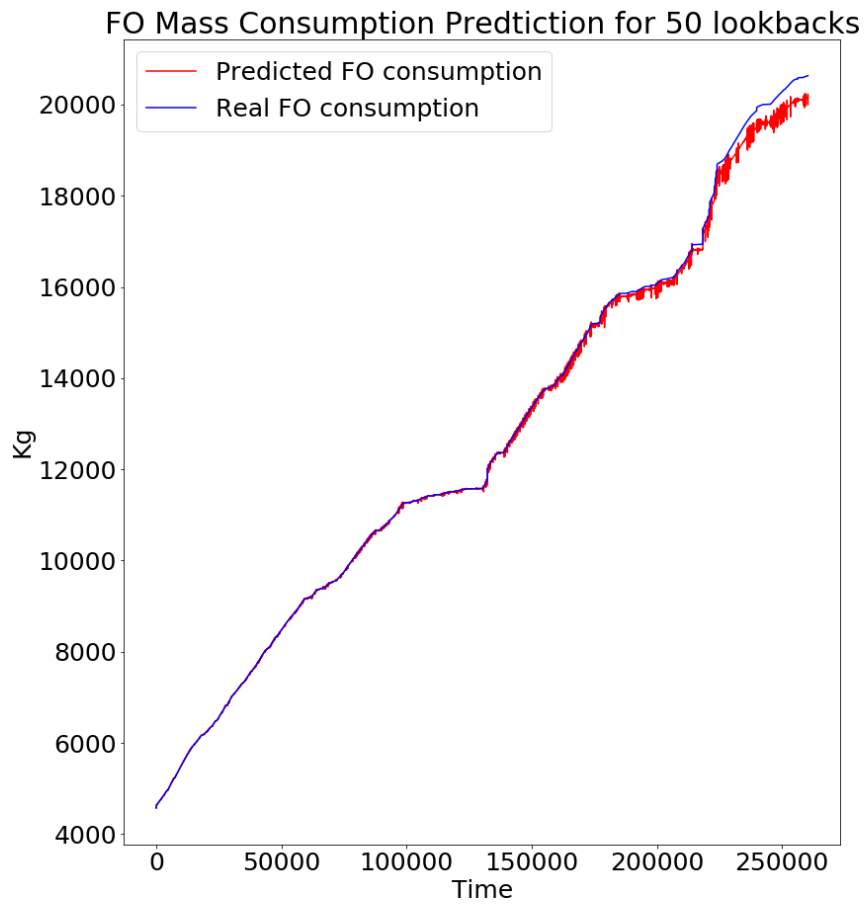


Figure 5.105: Results of the *third* ship after 50 epochs for 50 lookback

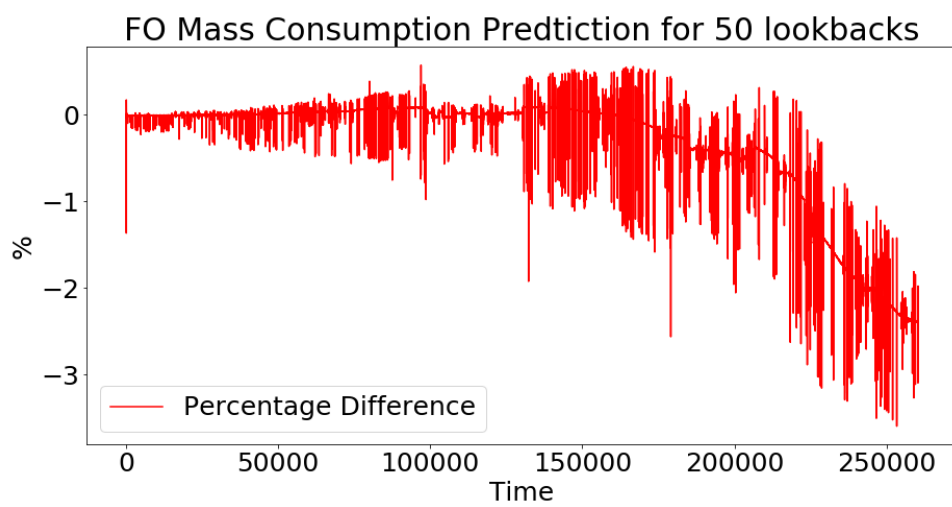


Figure 5.106: Results of the % difference of the *third* ship after 50 epochs for 50 lookback



## Chapter 6

# Conclusions and Future Work

### 6.1 Conclusions

From the overall results that were presented above the following conclusions can be drawn:

- First of all, it is possible to implement a neural network and predict the future value of the Fuel Oil Consumption of a ship with an accuracy above 99,5%.
- Moreover, this model can also predict fairly well the Fuel Oil Consumption of a sister ship most of the time.
- It is also possible to predict the Fuel Oil Consumption of a ship that has the same type as the one from which the training data was collected.
- Last but not least, it seems that when more variables are used for training and the bigger the timestep - lookback is the result tend to improve.

### 6.2 Next steps

In this, last, part of the thesis the future work will be discussed. From the previous paragraphs it is clear that the implementation of neural networks can increase the performance of the common predictive methods. It is sure that the NN will be used in the near future, especially, with the rapid development of the Performance department in most of the maritime companies.

As it is clear from the result of this thesis, there is plenty of room for improvement. First of all, it is very interesting to analyze the results using more variables, such as the ship's and wind direction, the sea state etc.. Moreover, a different architecture, like increasing the layers, increasing or decreasing the number of neurons per layer and experimenting with the hyper parameters, might bear more fruits. Last but not least, it is of great interest to examine the possibility of creating a model that can predict the best route for the ship's trip, with a specific loading condition, and optimizing the route on board using the changing data of the sea state and weather conditions as well as the routes of other ships.



# Bibliography

- [1] F. Ahlgren and M. Thern, “Auto machine learning for predicting ship fuel consumption,” in *ECOS 2018-the 31st International Conference on Efficiency, Cost, Optimization, Simulation and Environmental Impact of Energy Systems, 17-21 June, 2018, Guimarães*, 2018.
- [2] T. Anan, H. Higuchi, and N. Hamada, “New artificial intelligence technology improving fuel efficiency and reducing co2 emissions of ships through use of operational big data,” *Fujitsu Sci. Tech. J.*, vol. 53, pp. 23–28, 2017.
- [3] B. P. Pedersen and J. Larsen, “Gaussian process regression for vessel performance monitoring,” in *12th International conference on computer and IT applications in the maritime industries (COMPIT 13)*, 2013.
- [4] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O’Reilly Media, 2019.
- [5] E. B. Beşikçi, O. Arslan, O. Turan, and A. I. Ölçer, “An artificial neural network based decision support system for energy efficient ship operations,” *Computers & Operations Research*, vol. 66, pp. 393–401, 2016.
- [6] M. Chaal, “Ship operational performance modelling for voyage optimization through fuel consumption minimization,” 2018.
- [7] M. Jeon, Y. Noh, Y. Shin, O.-K. Lim, I. Lee, and D. Cho, “Prediction of ship fuel consumption by using an artificial neural network,” *Journal of Mechanical Science and Technology*, vol. 32, no. 12, pp. 5785–5796, 2018.
- [8] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An introduction to statistical learning*, vol. 112. Springer, 2013.
- [9] B. P. Pedersen and J. Larsen, “Prediction of full-scale propulsion power using artificial neural networks,” in *Proceedings of the 8th international conference on computer and IT applications in the maritime industries (COMPIT’09), Budapest, Hungary May*, pp. 10–12, 2009.
- [10] B. Yoo and J. Kim, “Probabilistic modeling of ship powering performance using full-scale operational data,” *Applied Ocean Research*, vol. 82, pp. 1–9, 2019.
- [11] A. Coraddu, L. Oneto, F. Baldi, F. Cipollini, M. Atlar, and S. Savio, “Data-driven ship digital twin for estimating the speed loss caused by the marine fouling,” *Ocean Engineering*, vol. 186, p. 106063, 2019.
- [12] D. Kim, S. Lee, and J. Lee, “Data-driven prediction of vessel propulsion power using support vector regression with onboard measurement and ocean data,” *Sensors*, vol. 20, no. 6, p. 1588, 2020.

- 
- [13] F. Ahlgren, M. E. Mondejar, and M. Thern, “Predicting dynamic fuel oil consumption on ships with automated machine learning,” *Energy Procedia*, vol. 158, pp. 6126–6131, 2019.
- [14] A. Coraddu, L. Oneto, F. Baldi, and D. Anguita, “Vessels fuel consumption: A data analytics perspective to sustainability,” in *Soft computing for sustainability science*, pp. 11–48, Springer, 2018.
- [15] Q. Liang, H. A. Tvette, and H. W. Brinks, “Prediction of vessel propulsion power using machine learning on ais data, ship performance measurements and weather data,” in *Journal of Physics: Conference Series*, vol. 1357, p. 012038, IOP Publishing, 2019.
- [16] M. Abebe, Y. Shin, Y. Noh, S. Lee, and I. Lee, “Machine learning approaches for ship speed prediction towards energy efficient shipping,” *Applied Sciences*, vol. 10, no. 7, p. 2325, 2020.
- [17] M. P. Deisenroth, A. A. Faisal, and C. S. Ong, *Mathematics for machine learning*. Cambridge University Press, 2020.