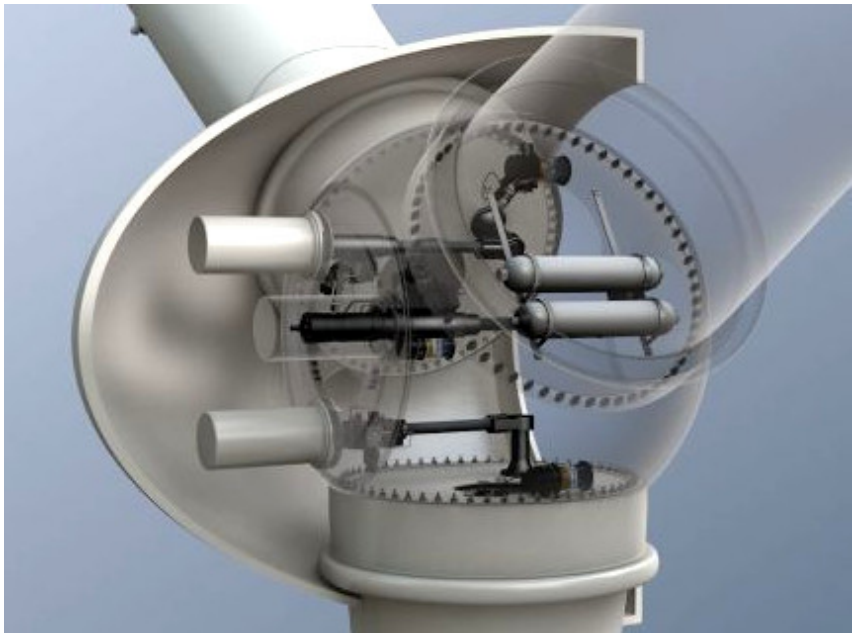
Master's Thesis

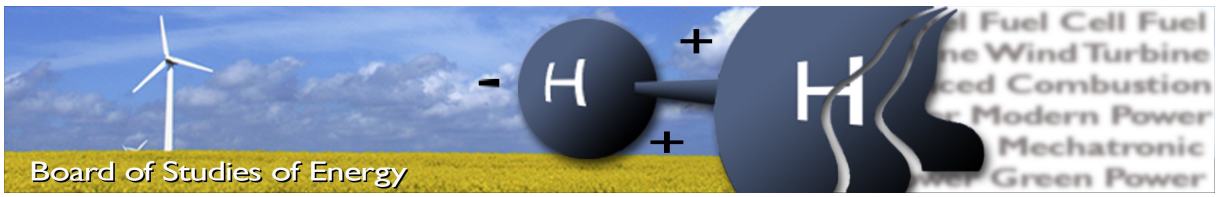
A Neural Network Approach for Gas Leakage Detection in Fluid Power Accumulators of Wind Turbines



MCE4-1029
Denis Bartz Rafaeli Neto

Aalborg University
Department of Energy Technology
June 2, 2023

LaTeXwas used for typesetting this document using VS Code LaTeX Workshop extension. Figures were created using diagram.io and plots using Matlab or matplotlib Python library. Simulink was used for all simulations and TensorFlow/Keras for neural network development.



Title:	A Neural Network Approach for Gas Leakage Detection in Fluid Power Accumulators of Wind Turbines
Semester:	10th
Semester theme:	Master's Thesis in Mechatronic Control Engineering
Project period:	01.02.2023 to 01.06.2023
ECTS:	30
Supervisor:	Henrik C. Pedersen
Project group:	MCE4-1029

Denis Bartz Rafaeli Neto

Denis Bartz Rafaeli Neto

Pages, total:	50
Appendix:	2
Supplements:	0

SYNOPSIS:

This thesis proposes a neural network-based approach for gas leakage detection in accumulators of offshore wind turbines, aiming to improve maintenance strategies and minimize downtime. The research begins with the development and validation of an accurate accumulator model using data that captures various operational scenarios. The model is validated using experimental data from the Hydraulics Laboratory at AAU. Subsequently, a Fully Convolutional Network (FCN) model is developed for gas leakage detection. It is designed to classify a group of input signals and determine the corresponding gas pre-charge pressure. The FCN model is trained using time series data and evaluated for its performance. The study explores the impact of input variables, sliding window size, hyperparameters, and sensor utilization on the performance of the neural network. Experimental results show that incorporating oil pressure, along with oil and ambient temperature signals in the neural network model achieves an accuracy of 95% when classifying the pre-charge pressure. Adding thermocouples to the accumulator's surface significantly enhances the neural network performance, reaching 100% accuracy.

By accepting the request from the fellow student who uploads the study group's project report in Digital Exam System, you confirm that all group members have participated in the project work, and thereby all members are collectively liable for the contents of the report. Furthermore, all group members confirm that the report does not include plagiarism.

PREFACE

Developed by the MCE4-1029 group, the presented thesis focus on finding a new solution for gas leakage detection in piston accumulators using neural networks. The achieved results are specific for the used case scenarios, serving as proof of concept and base for future development.

Reader's Guide

Citation style was chosen after the IEEE standards. The sources used in this report are listed in page 51. They are numbered in order of appearance in the text, marked within square brackets, e.g. [1]. The reference applied before the period of a sentence is attributed to only that sentence. When used after a period, it is attributed to the section before.

Matrices and vectors are denoted bold, e.g. $\boldsymbol{\varphi_1}$, $\mathbf{M}$.

A list of abbreviations is shown in page vii.

Aalborg University, June 2, 2023

Denis Bartz Rafaeli Neto
<dneto21@student.aau.dk>

SUMMARY

Having a good maintenance strategy is of paramount importance for the future development of Offshore Wind Turbine (OWT). Pitch system's accumulators are critical components that pose challenges in fault detection. To address this issue, this thesis proposes a neural network-based approach for gas leakage detection in accumulators of offshore wind turbines.

The first step involves developing and validating an accurate accumulator model. Data is generated by varying the gas pre-charge pressure, load flow intensity, and oil and ambient temperature to capture different operational scenarios. The model's accuracy is evaluated by comparing simulated and experimental data, with a reasonably good estimation of piston position and some discrepancies in wall temperatures.

Subsequently, a Fully Convolutional Network (FCN) model is developed for gas leakage detection. The FCN model is designed to classify a group of input signals and determine the corresponding gas pre-charge pressure. It is trained using time series data and evaluated for its performance.

The FCN model demonstrates good performance in classifying gas pre-charge pressure. The choice of input variables and the size of the sliding window significantly affect the model's performance, with a window size of 3906 samples achieving a balance between accuracy and training time.

Optimizing hyperparameters, such as the number and size of filters, is crucial for improving model performance. By considering only three available sensors (oil pressure, oil temperature, and ambient temperature), the model achieves 95% accuracy and a loss of 0.1634. Further improvements are observed by including the temperature of the accumulator surface, reaching 100% accuracy and a loss of 0.0085 when three thermocouples are included.

Overall, the experiments demonstrate the potential of neural networks for predicting gas pre-charge pressure in wind turbines. Careful selection of input variables, optimization of hyperparameters, and consideration of data characteristics are essential for achieving optimal performance. The incorporation of additional sensor data improves model accuracy, highlighting the effectiveness of the neural network approach with a limited number of sensors.

CONTENTS

NOMENCLATURE

Abbreviations

CNN Convolutional Neural Network

DL Deep Learning

FDD Fault Detection and Diagnosis

FNN Fully Convolutional Networks

LCOE Levelized Cost of Energy

LSTM Long Short-Term Memory

ML Machine Learning

MLP Multi Layer Perceptrons

NN Neural Network

OWT Offshore Wind Turbine

ResNet Residual Network

SAEKF State Augmented Extended Kalman Filter

TSC Time-Series Classification

TSR Time-Series Regression

1 INTRODUCTION

Offshore wind turbine (OWT) sites are subject to unpredictable and erratic wind load variations. While the increased loading cycle are beneficial for increasing power output, they also can cause turbine components to fatigue and wear out more quickly. Moreover, being situated in the ocean makes OWTs highly susceptible to corrosion and erosion, which leads to a necessity of continuous and frequent maintenance of the system.[1]

The Levelized Cost of Energy (LCOE) of OWT is still considerably higher compared to onshore wind turbines, and one factor contributing to this is the high maintenance costs [2]. According to Ren et al. [1], the operation and maintenance costs for wind turbines represent about 23% of the total investment for offshore and 5% for onshore. Hence, having a good maintenance strategy is of paramount importance for the future development of OWT.

Offshore installations present numerous maintenance challenges, which highlights the importance of these operations. For instance, accessibility is compromised by the distance from the port and the increased risk of inclement weather. The longer it takes to reach the site, the more turbine downtime, thus increasing revenue losses and decreasing potential energy production. [1, 2]

Pitch system faults are one of the main causes of turbine downtime. These correspond to about 13% of all failures in OWT [3]. This system is responsible for adjusting the output power by changing the blade pitch angle. It uses the aerodynamics of the blade to slow down or speed up the turbine's rotation. Above all, it is part of the safety system in case of emergency, serving as a rotational brake.

In order to increase safety, each blade is designed with an independent pitch system. These can be hydraulic or electrical systems. The former utilizes pumps, valves, accumulators and cylinders to rotate the blade, and the latter uses batteries and electrical motors. A comparison of the reliability of these two systems is provided by Walgern et al. [4], and results show that both have similar failure rates, but varying according to power output and manufacturer. Fluid power pitch systems will be the focus of this thesis, as they are commonly used on offshore turbines and are preferred over electrical systems [2].

It is shown in [2, 3, 5] how faults are distributed in fluid power pitch systems. Accumulators account for about 10.5% of all pitch system failures, making it the third most common failure. In the two main pitch operations — normal operation and emergency shutdown — different accumulators are utilized. During normal operation accumulators are used to support the pump actuation and improve the supply system. For emergency situations accumulators are used as hydraulic energy storage, supplying power to the pitch system so that it can brake and stop the turbine if necessary.

The importance of accumulators in the reliability of OWT is clear. The next section provides a more in-depth review of accumulator failures and troubleshooting strategies.

2 PROBLEM ANALYSIS

This chapter aims to provide a clear understanding of accumulator failures and how to measure and troubleshoot them. A review of relevant fault detection methods is made, including both classical and neural networks-based methods. Furthermore, a detailed description of the problem statement is presented, along with a discussion of the limitations encountered throughout the course of this thesis.

2.1 System and Fault Definition

As stated in the chapter 1, accumulators are utilized in pitch systems for both normal and emergency operations. Liniger [5] provided a comprehensive hydraulic diagram and system description. However, for the purpose of this thesis, a simplified version from [6] is presented in figure 1, depicting an accumulator that functions for both operations.

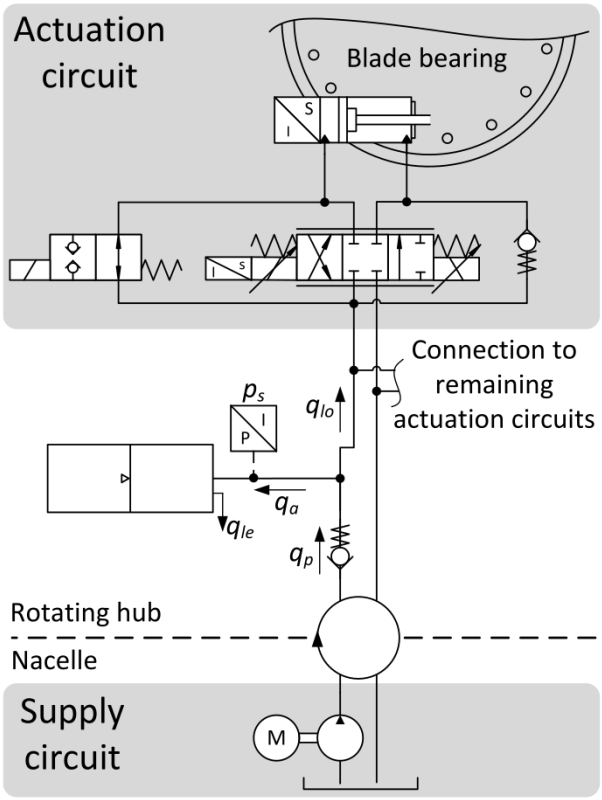


Figure 1: Hydraulic schematics of a fluid power pitch system from Liniger et al. [6]

There is one pitch system for each blade, with independent supply circuit located in the stationary Nacelle. The actuation circuit, connected to the blade, is situated within the rotation hub. To control the blade pitch angle during normal operation, the actuation circuit is enabled by the proportional valve. In the event of emergency, the safety valve opens, linking the accumulator to both cylinder chambers, which extend the cylinder and position the blade at a 90°angle to the wind. [6]

2.1.1 *Accumulators*

As mentioned before, accumulators serve as energy storage devices that collect liquid under pressure and release it as needed. A spring, weight, or compressed gas is used to counterbalance the fluid pressure within a given pressure range. Gas accumulators are preferred for wind turbines due to their reliability, efficiency, and ability to withstand extreme environmental conditions. For the purposes of this thesis, piston accumulators will be considered.

Piston accumulators provide flexibility in terms of mounting position and size, with a higher power density, better overall performance, and greater robustness against extreme temperatures. They also have lower seal permeability, making them a better choice for hydraulic systems that use oil or special fluids. A significant advantage of piston accumulators is that their seals wear slowly, making maintenance and intervention easier to manage. [7]

The main parts of the piston accumulator are shown in figure 2.

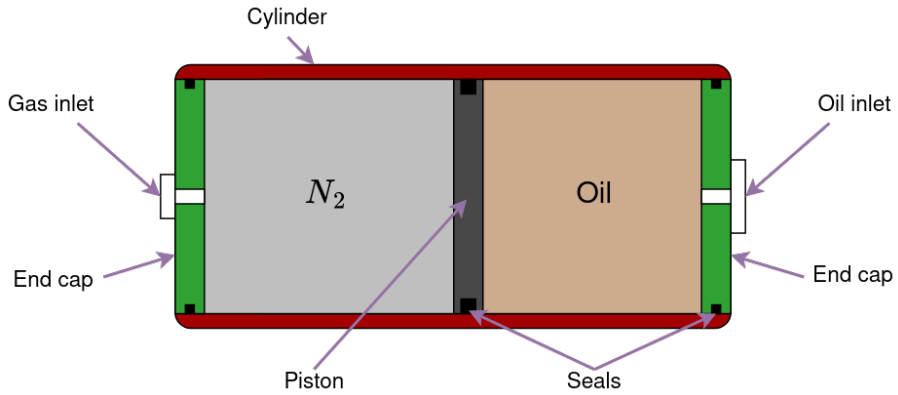


Figure 2: Piston accumulator internal parts.

In this type of accumulator, gas and oil are separated by a piston, a movable part that is in constant contact with the inner wall of the accumulator. A seal must be placed here to maintain a clear separation between the fluids. In addition, the end caps are threaded and all have O-rings to prevent external leakage. In the figure 2, the seals are represented by black squares.

2.1.2 *Pre-charge Pressure*

The gas pre-charge pressure is an important parameter for this type of accumulator, as it directly affects the energy storage capacity. The correct way to pre-charge the accumulator is when the fluid chamber is empty, i.e. the system is stopped and no fluid is being supplied. This can happen during installation or maintenance procedures. A dry and inert gas such nitrogen is filled into the accumulator and the pressure is measured, thereby the pre-charge pressure is acquired. By knowing the gas pressure and temperature, and accumulator volume, it is possible to calculate the initial amount of gas contained in the accumulator. Later it can be compared in a condition monitoring system to detect gas leakage. These calculations are going to be discussed further in section 3.4. [2, 6, 7]

Of course, setting up the correct pre-charge pressure is crucial. Excessive pre-charge pressure could cause the piston to hit the oil side end cap, thereby decreasing the accumulator's output power and ultimately damaging the piston or the seals. Furthermore, if this pressure is higher than the maximum pressure of the fluid system, no more fluid can be admitted into the accumulator. On the other hand, when there is an insufficient pre-charge pressure, the piston may hit the end cap of the gas side and damage may also occur. [6, 7]

Measuring the pre-charge pressure is a common way to know the amount of gas in the accumulator, however, when in OWT normal operation, the gas pressure and temperature sensors are omitted due to safety and costs reasons. A way around the situation would be measuring the piston position, because it gives the current gas chamber volume information and in combination with the oil pressure, the amount of gas could be calculated. Nevertheless, adding piston position sensors would also increase the risk of failures. [6, 8]

2.1.3 Gas Leakage

Leakage generally comes from small breaches in the seals, caused by improper installation, wear over time, fluid contamination or damage due to external circumstances. It is defined as the escape of fluid or gas from a container. There are two main types of leakage: fluid leakage into the gas chamber or gas leakage. [7]

- When fluid leaks to the gas chamber the available space for the gas decreases. As the amount of gas remains the same the pressure rises.
- When gas is leaking into the fluid chamber or the external environment, the accumulator pre-charge pressure decreases, i.e., the charging capacity decreases. Thus, the amount of gas decreases, but the pressure is maintained, resulting in a decrease in volume. Finally, gas leakage is associated with a loss of gas mass.

According to [2], the primary mode of failure for accumulators is gas leakage. Due to the gas's low viscosity, gas leakage through sealing cracks is facilitated. Moreover, slow seal wear also makes it challenging to detect leaks, as it develops over a long period of time. This is significant when modeling the accumulator behavior because the simulation time frame may not be long enough to account for system changes such as drop in pre-charge pressure or reduction in gas mass. [8] Therefore, for this thesis, leakage will not be modeled.

2.2 Maintenance Methods

Proactive maintenance is preferred over corrective maintenance as it is done before the actual failure and, therefore, the turbine downtime is minimized. The goal is to prevent minor faults from turning into major failures. Preventive, predictive and condition-based are types of proactive maintenance. [1]

- **Preventive** refers to schedule maintenance that could be planed based on parameters such as weather data, component age and energy production plan.

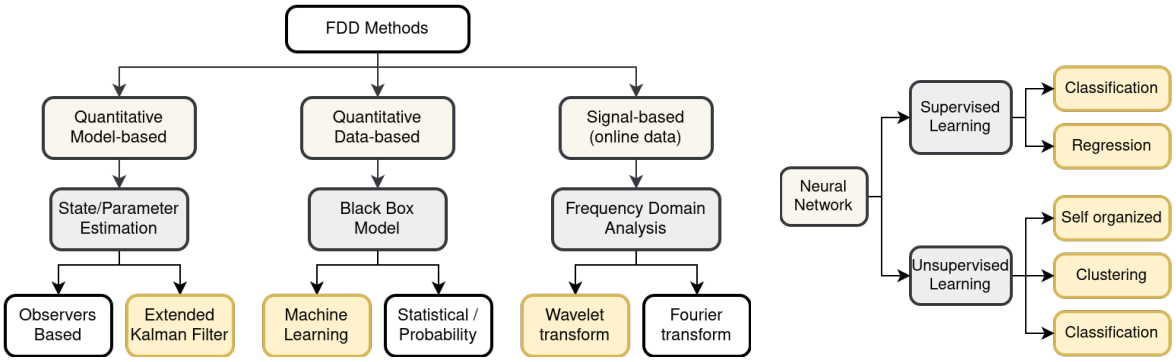
- **Condition-based** uses state and deterioration information of components to prevent major failures. Sensors provide data for condition monitoring systems, health diagnosis and fault analysis systems.
- **Predictive** is a more advanced approach that utilizes of condition-based methods combined with virtual models to predict when the failure will happen. This method, although more complex, effectively reduces turbine downtime.

While preventive maintenance can be beneficial, it can become costly when not optimized and made efficient. Condition-based and predictive maintenance methods are valuable in minimizing maintenance costs and increasing wind turbine reliability. However, the addition of sensors to these systems can increase complexity and cost, as well as the risk of sensor failures. [1] Condition monitoring in conjunction with planned maintenance are key to reduce costs with operation of a OWT [5].

2.3 Fault Detection and Diagnosis

Fault detection and diagnosis (FDD) methods are often used as part of maintenance strategies to identify and address problems in a timely manner, before they escalate into more significant issues. FDD is a high-level process that consists of four steps. The first step is to detect the fault in the system, followed by diagnosing the cause of the fault. Once these are done, the next step is to evaluate the impact of the fault and finally the fault is addressed. [9]

Several types of FDD classification and framework exist, including model-based, signal-based, and data-based methods. A comprehensive analysis of these classifications can be found in [9, 10]. The selection of an appropriate FDD method depends on several considerations, such as the complexity of the system, available data, the type and severity of the faults, the available computational resources, and desired level of accuracy and robustness. To provide a better context for the methods presented in the next section, a simplified FDD classification diagram is shown in figure 3a. The neural network methods are also commonly divided regarding the learning approach, as shown in figure 3b.



(a) Common FDD classification. **(b)** Common NN methods.

Figure 3: Classification showing some of the most common FDD and NN methods.

Classical FDD methods, which are addressed in section 2.3.1, have been widely developed and extensively researched, although there are limited publications on gas leakage detection on accumulators [2]. These methods include signal processing techniques, statistical methods, and model-based approaches. Signal processing techniques analyze system-generated signals and identify abnormalities or changes that could indicate a fault. Statistical methods rely on statistical models to analyze data and detect anomalies. Model-based approaches utilize mathematical models to simulate system behavior under various operating conditions and identify faults based on deviations from expected behavior. [10]

Neural networks (NN) have emerged as a promising approach in FDD due to their ability to effectively handle complex and nonlinear systems. Their advantages include parallel data processing, generalization capability, and ease of implementation [11]. NNs can be trained to detect and classify faults in real-time by learning from data generated by the system. They are commonly employed in tasks such as regression, classification, and clustering, which involve quantitative output, discrete classification, and automatic categorization based on similarities, respectively [12]. The section 2.3.2 presents previous works in the field of FDD using NN and serves as a foundation for the testing of various neural network approaches in this thesis.

2.3.1 Classical FDD Methods

A comprehensive research was made by Asmussen, Liniger, and Pedersen [2] on gas leakage detection methods in pitch systems. Most approaches mentioned are only effective during charge or discharge periods, not during continuous operation. However, from this study, two methods are capable of continuous operation: the first is a signal-based leakage detection method that utilizes multiresolution signal decomposition based on wavelets, while the second method is a State Augmented Extended Kalman Filter (SAEKF) approach.

The proposed signal-based gas leakage detection method in [6] use multiresolution signal decomposition based on wavelets. Detail coefficient 9 of the supply pressure signal was found to be sensitive to gas leakage faults. The method was able to isolate several levels of pre-charge pressure during nominal operating conditions of a wind turbine, and was also shown to be robust in a multi-fault environment. The experimental results showed good correlation to the simulations and indicated that the method was effective.

The paper [8] proposes a State Augmented Extended Kalman Filter (SAEKF) based approach for gas leakage detection by estimating the pre-charge pressure of a hydraulic piston accumulator using temperature measurements on the accumulator surface as inputs. The SAEKF model estimates the number of moles of nitrogen gas in the accumulator, which is used to determine the pre-charge pressure. The estimation was associated with an offset up to 10%, which can be attributed to uncertainties in heat transfer coefficients and model assumptions. Despite the offsets, the SAEKF was shown to be suitable for detecting changes in the pre-charge pressure over time.

2.3.2 *Neural Network FDD Methods*

From Asmussen, Liniger, and Pedersen [2] review, only one gas leakage detection method was based in neural networks, so further research was made within this topic. Not many articles were found in gas leakage detection in accumulators, however FDD using neural networks has been shown good results for similar applications.

Jin et al. [13] presents a fault detection and identification scheme for diagnosing piston seal wear and subsequent internal leakage in hydraulic cylinders. The proposed method uses wavelet transform as a feature extractor to transform the raw oil pressure data into a feature vector consisting of wavelet packet subband energy, energy entropy, energy variance, and root mean square of the wavelet detailed coefficient d4. This feature vector feeds into the wavelet neural network serving as a pattern recognizer for automatically classifying the fault patterns. The study demonstrates that the proposed scheme is capable of effectively detecting and classifying the piston seal wear with excellent accuracy.

Chen et al. [14] proposes a deep learning (DL) based method to detect and classify wind turbine imbalance faults caused by ice accretion. The method combines long short-term memory (LSTM) with an attention mechanism to improve the learning ability and convergence rate. It uses voltage and current signals, as well as wind speed and torque. The simulation results show that the proposed method is feasible for wind turbine blade imbalance detection, with the highest accuracy of 100

Kopbayev et al. [15] presents a neural network model to detect gas leakage in natural gas storage facility. The interesting part in this study is that it transforms the acquired signals into images, which later are used in the model. The feature extraction is made by the pre-trained convolutional neural network (CNN) GoogLeNet, and classification is achieved by a bidirectional LSTM layer network. The proposed model demonstrated promising results for early and accurate leak detection without requiring fine-tuning of layer configuration and other training parameters.

Specifically for wind turbines, in a more general review, Helbing and Ritter [16] discusses the use of neural networks for fault detection and classification. Studies have shown that neural networks are capable of detecting early stage incipient faults in various components of turbines. Unsupervised deep learning methods using historical production data have shown better performance than shallow multi-layer neural networks. However, supervised deep learning approaches are more effective for fault diagnosis, although obtaining labeled data for training is challenging due to issues such as label accuracy and class imbalance.

CNNs and LSTM networks are fundamental components of more complex neural network architectures and both have been extensively studied in various application domains. As mentioned previously, they have shown promising potential as alternatives to classical FDD methods. A detailed exploration of these and others NN methods employed in this thesis is provided in chapter 4.

2.4 Methodology

The approach for solving the leakage detection problem is described below. The steps required to carry out this thesis are: state-of-art analysis, accumulator model, data generation, neural network model, documentation of results and discussion.

State-of-art Analysis Before delving into the modeling process, it is crucial to gather knowledge from established and proven methods. As previously mentioned, the use of neural networks to detect accumulator faults are not extensively researched. Therefore, this step aims to gather insights from various scenarios and sources of information.

Accumulator Model At first an accumulator model will be developed, based on the research conducted by Liniger et al. [6] and Asmussen et al. [8]. This model is important to provide flexibility in changing parameters and conditions that would otherwise be difficult with experimental setup. It serves as a useful testing and experimentation tool for the neural network methods used in this thesis. Once the model has been validated using experimental data from the laboratory, multiple data frames will be generated, incorporating variations in pre-charge pressure and load.

Neural Network Model After reviewing basic concepts of machine learning and neural networks, the work starts by setting up the required software and testing basic models as a way of practicing and understanding the methods. The state-of-art analysis is useful as foundation on the development of the methods chosen. Several methods may be tested against the generated data from the accumulator model, however only some of the most interesting ones are going to be presented in this thesis.

Discussion Among the results discussed are: the validation of the accumulator model and potential improvements, a comparison of the tested neural network methods and recommendations for their deployment. The report will be written as the previous steps progress, and it will also cover general improvements and outline future work.

2.5 Problem Statement

Considering the possibility of identifying gas leakage through monitoring the pre-charge pressure, and taking into account the existing pre-charge estimation methods outlined in section 2.3.1, as well as the neural network methods presented in section 2.3.2, the primary objective of this thesis is to tackle the following problem statement:

"How can a neural network be developed and utilized to assist in the investigation of gas leakage from fluid power accumulators of wind turbines?"

2.6 Limitations

Some limitations emerged during the development of this thesis and are better explained below.

- Only the accumulator is considered, the rest of the pitch system and wind turbine are simplified as a "load" signal.
- Sensors cannot be added to the system, therefore the goal is to develop the neural network with fewer variables and scale up as needed.
- Due to time constraints, the neural network model is not applied in real hardware.
- The amount experimental data provided by the laboratory is not enough to train the neural network, therefore it is only used for accumulator model validation.
- Hardware used for training the neural network is a personal computer with an Intel® Core™ i7-8750H CPU and Nvidia GTX 1050 Ti GPU.
- Due to familiarity, the neural network will be programmed using TensorFlow/Keras and other python libraries.

3 ACCUMULATOR MODEL

The works of Liniger et al. [6] and Asmussen et al. [8] are the foundation for the model presented in this section. In order to simulate the accumulator real-life behavior, several equations are taken into account. A mechanical model estimates the piston position using Newton's second law of motion. For the gas pressure and temperature, an equation of state and a balance of energy are utilized. Finally, the oil pressure is derived from the continuity flow equation. Some important considerations taken are:

1. As mentioned in section 2.1.3, the gas leakage is not explicitly modeled, hence it is considered that the amount of gas is constant ($m_g = constant$) during simulation time.
2. The gas pressure and temperature distribution are uniform throughout the chamber.
3. Oil pressure and temperature are uniform throughout the hoses and the accumulator oil chamber.
4. The accumulator dimensions, the flow reference direction are based on figure 4.

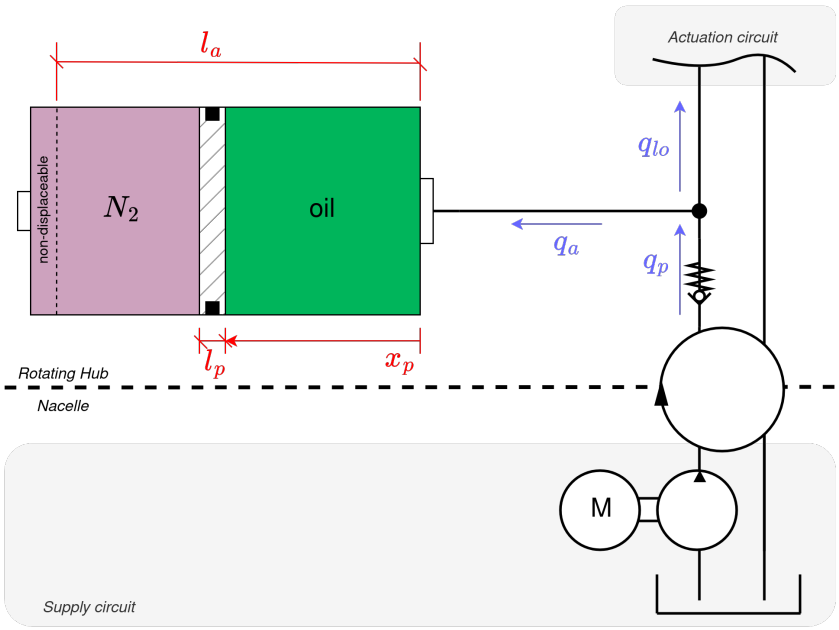


Figure 4: Piston accumulator diagram for model and simulation.

3.1 System Input/Output

As can be seen in figure 4 the system input flow, q_p, is being supplied by a fixed-displacement pump. For simplification purpose the pitch system load is represented by an output flow q_{lo}. The difference between both is the flow entering or leaving the accumulator oil chamber.

$$q_a = q_p - q_{lo} \tag{1}$$

The accumulator flow q_a is positive when charging and negative when discharging. The pump is activated if the oil pressure drops below a lower limit $p_{o,min}$, and is deactivated when it rises above an upper limit $p_{o,max}$. The accumulator hence works within this threshold.

The load flow in this thesis is modeled based on the work of Liniger et al. [6] and load profiles provided by the AAU laboratory. The mentioned work simulates a controlled pitch system connected to FAST (Fatigue, Aerodynamics, Structures and Turbulence). This program allows different configurations of wind speed, turbulence and ambient temperature, which can be used for simulation with Matlab/Simulink. Due to the wind stochastic nature, the load variations on the pitch system are also random. Therefore, for this thesis, the load is generated by a combination of stochastic signals and band-limited white noise. The load validation is presented in section 3.6. Base pitch system parameters are provided in table 1.

Table 1: Base parameters for the pitch system simulation.

Notation	Description	Value	Unit
q_p	Pump flow	20	(L/min)
q_{lo}	Load flow	$\approx 0 \rightarrow 20$	(L/min)
q_a	Accumulator flow	-	(L/min)
$p_{o,max}$	Oil pressure upper limit	190	(bar)
$p_{o,min}$	Oil pressure lower limit	165	(bar)

3.2 Oil Model

As the system input is the pump flow, it is necessary to know the oil pressure, p_o, which is considered uniform along the hoses and the whole accumulator oil chamber. The continuity flow equation (2) is applied.

$$Q_{in} - Q_{out} = \frac{dV}{dt} + \frac{V}{\beta}\frac{dp}{dt} \tag{2}$$

The change in oil pressure $\dot{p}_o$ is acquired using equation (3), which later is used in the accumulator mechanical model [8].

$$\dot{p}_o = (q_a - \dot{V}_o) \cdot \frac{\beta}{V_{o,0} + V_o} \tag{3}$$

Where $\dot{V}_o$ is the time derivative of the oil volume, $V_{o,0}$ is initial oil volume, V_o is the current oil volume, and β is the oil Bulk Modulus.

3.3 Mechanical Model

The piston is included in the model so that the volume change of both gas and oil chambers take into account the piston dynamics. This is achieved by using Newton's second law of motion [8]:

$$\ddot{x}_p m_p = (p_o - p_g)A_p - F_{fric} \tag{4}$$

Again, p_o is the oil pressure and p_g is the gas pressure, x_p is the piston position, A_p is the piston area, m_p is the piston mass. The friction F_{fric} is modeled as in equation (5) [8].

$$F_{fric} = B_{fric}\dot{x}_p + C_{fric}sign(\dot{x}_p) \tag{5}$$

Where B_{fric} is the viscous friction and C_{fric} is the Coulomb friction. The constants used in this model are presented in table 2.

Table 2: Accumulator mechanical model parameters [8, 17].

Notation	Description	Value	Unit
A_p	Piston area	2.54×10^{-2}	(m^2)
l_p	Piston length	0.11	(m)
m_p	Piston mass	3.97	(kg)
B	Viscous friction coefficient	5×10^3	$(\frac{N \cdot s}{m})$
C	Coulomb friction coefficient	2×10^3	(N)
$V_{a,0}$	Non-displaceable accumulator volume	1.4×10^{-3}	(m^3)
l_a	Accumulator length	0.983	(m)

With the piston position determined, and considering the accumulator dimensions shown in figure 4, it becomes straightforward to calculate the volumes of the oil and gas chambers:

$$V_o = x_p \cdot A_p \tag{6}$$

$$V_g = V_a - V_o - V_{piston} + V_{a,0} \tag{7}$$

Where the accumulator volume V_a is defined as equation (8).

$$V_a = l_a \cdot A_p = 25 \text{ L} \tag{8}$$

3.4 Gas Model

Pre-charge pressure, as mentioned in section 2.1.2, can be used to estimate the quantity of gas present in the accumulator, i.e., the mass of the gas. Given that leakage, for this thesis, is defined as a loss of gas mass, by monitoring the gas pressure along with the gas temperature, it is possible to determine whether leakage has occurred.

3.4.1 Gas Pressure

When pressure changes in the accumulator, the resulting gas force on the piston behaves non-linearly, the gas stiffness increases with the pressure [18]. As the gas used is nitrogen, it is common to assume that it is an ideal gas and adiabatic, however in applications where steel accumulator are used, the gas behavior is best described by the Benedict-Webb-Rubin (BWR) gas state equation (9) [6, 8].

$$p_g = \frac{RT_g}{v_g} + \frac{B_0 RT_g - A_0 - \frac{C_0}{T_g^2}}{v_g^2} + \frac{bRT_g - a}{v_g^3} + \frac{a\alpha}{v_g^6} + \frac{c\left(1 + \frac{\gamma}{v_g^2}\right) e^{-\frac{\gamma}{v_g^2}}}{v_g^3 T_g^2} \tag{9}$$

Where v_g is specific molar volume, m_g is the gas mass, and $A_0, B_0, C_0, a, b, c, \alpha, \gamma$ and R are the gas constants for nitrogen that are shown in table 3.

Table 3: Accumulator gas model parameters [8].

Notation	Description	Value	Unit
R	Gas constant	8.31	(N)
A_0	BWR constant	0.11	$(-)$
B_0	BWR constant	4.07×10^{-5}	$(-)$
C_0	BWR constant	816.58	$(-)$
a	BWR constant	2.54×10^{-6}	$(-)$
b	BWR constant	2.33×10^{-9}	$(-)$
c	BWR constant	7.38×10^{-2}	$(-)$
α	BWR constant	1.27×10^{-13}	$(-)$
γ	BWR constant	5.3×10^{-9}	$(-)$

The gas specific molar volume v_g is calculated by equation (10).

$$v_g = \frac{V_g}{n_{N2}} \quad \left[\frac{m^3}{mol}\right] \tag{10}$$

Here n_{N2} is the number of moles of nitrogen, which is another measure for gas leakage. As mentioned in section 2.1.3, the gas leakage is not explicitly modeled, so it assumed that n_{N2} is constant. In order to determine this value, calculations are made prior to running the simulation, and after the initial conditions have been set. Since gas filling occurs only when the accumulator is completely discharged of fluid, the initial gas volume $V_{g,0}$ is known. By measuring the gas pre-charge pressure and the gas temperature, in combination with equation (9), the gas specific volume can be found. This equation's input are now p_g and T_g and the output is v_g. Equation (10) allows for isolation of n_{N2}, which finally determines the initial number of moles. The initial gas mass can be calculated by equation (11).

$$m_g = n_{N2} \cdot M_{N2} \tag{11}$$

3.4.2 Gas Temperature

Due to the lack of sensors, it is challenging to gather data on the gas temperature of accumulators in OWT. This might be as a result of complexity or cost savings from IO signal modules. Either way, this thesis utilizes two method of estimating gas temperature and a comparison is made in section 3.6.

Method 1: A simpler way of calculating the gas temperature is done in Liniger et al. [6], which uses Otis and Pourmovahed [19] energy balance deduction in conjunction with an approximated thermal time constant τ made by Rotthäuser [20]. Assuming the gas temperature to be uniform across the accumulator chamber, the time derivative of the gas temperature is expressed by equation (12).

$$\dot{T}_g = \frac{T_a - T_g}{\tau} - \frac{\dot{v}_g T_g}{c_{N2} M_{N2}} \left(\frac{\partial p_g}{\partial T_g}\right) \tag{12}$$

The approximated τ is calculated by equation (13).

$$\tau \approx 0.3 \cdot 10^{-5} \cdot p_{pc} \cdot V_a^{0.33} + 86.2 \cdot V_a^{0.49} \tag{13}$$

Where p_{pc} is the gas pre-charge pressure and V_a the accumulator total volume.

Method 2: It has been demonstrated in the work of Asmussen et al. [8] that accumulator surface temperature measurement yields an accurate gas temperature estimation. It utilizes the energy balance deduction from Pourmovahed and Otis [21] to determine the change in temperature, which is presented in equation (14).

$$\dot{T}_g = -\frac{\dot{Q}_s}{c_{N2} M_{N2} n_{N2}} - \frac{\dot{v}_g T_g}{c_{N2} M_{N2}} \left(\frac{\partial p_g}{\partial T_g} \right) \tag{14}$$

The total heat flow from the gas to all of its interacting elements is represented here by $\dot{Q}_s$. By using this method, it is necessary to create a model of the accumulator's thermal behavior that incorporates the heat exchange between the accumulator elements. This thermal model is shown in section 3.5.

For both methods, the time derivative of the gas specific molar volume is found by equation (15), and the partial derivative of gas pressure with respect of gas temperature by equation (16).

$$\dot{v}_g = \frac{\dot{V}_g}{n_{N2}} = -\frac{\dot{V}_o}{n_{N2}} = -\frac{\dot{x}_p \cdot A_p}{n_{N2}} \tag{15}$$

$$\frac{\partial p_g}{\partial T_g} = \frac{R}{v_g} + \frac{\left(\frac{2C_0}{T_g^3} \right) + RB_0}{v_g^2} + \frac{bR}{v_g^3} - \frac{2c \left(\frac{\gamma}{v_g^2 + 1} \right) e^{\frac{-\gamma}{v_g^2}}}{T_g^3 v_g^3} \tag{16}$$

3.5 Thermal Model

The main goal of this model is to find the heat flow from the gas to its interfacing elements and later find the gas temperature with equation (14). The elements are: the gas end cap, the piston and the accumulator wall, which is divided in 8 parts with equal dimensions. The accumulator and its elements are depicted in figure 5.

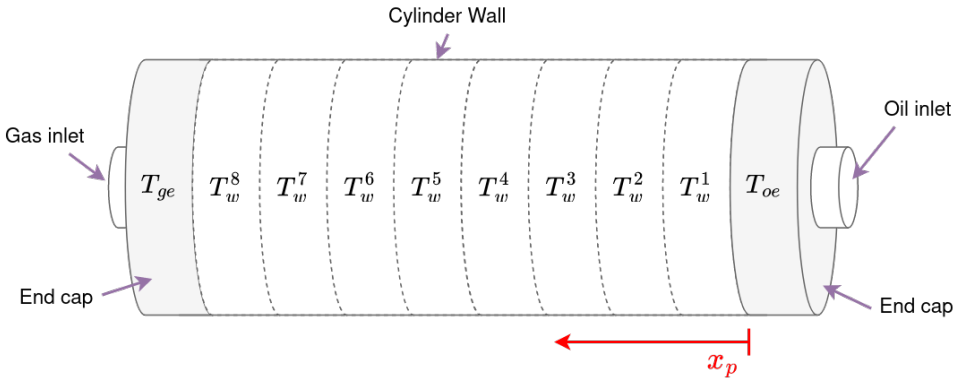


Figure 5: Accumulator elements for thermal model.

The oil end cap is always in contact with the oil, so it is considered that both are in temperature equilibrium ($T_{oe} = T_o$). The equations shown in this section are from Asmussen et al. [8]. The total heat flow from the gas to its interfacing elements is expressed by equation (17).

$$\dot{Q}_s = \left(\sum_{i=1}^{8} \dot{Q}_{gw}^{[i]} \right) + \dot{Q}_{gp} + \dot{Q}_{ge} \tag{17}$$

Where $\dot{Q}_{gw}$ is the heat flow from the gas to the wall, $\dot{Q}_{ge}$ is the heat flow to the end cap and $\dot{Q}_{gp}$ is to the piston, which are elaborated in the following subsections. Ambient, end cap, gas and oil are denoted by a, e, g, o respectively.

3.5.1 The Gas End Cap

The general heat flux equation can be used to determine the gas side end cap temperature T_e. The end cap exchanges heat with the gas and the surrounding ambient, therefore their heat flow are included in the equation.

$$\dot{T}_e = \frac{\dot{Q}_{ge} + \dot{Q}_{ae}}{m_e c_{steel}} \tag{18}$$

Where m_e is the end cap mass and c_{steel} the specific hear capacity for steel. The convective heat transfer from the end cap to the gas and to the ambient are:

$$\dot{Q}_{ge} = \alpha_g A_{eg} \left(T_g - T_e \right) \tag{19}$$

$$\dot{Q}_{ae} = \alpha_a A_{ea} \left(T_a - T_e \right) \tag{20}$$

Here, α_g and α_a are the convective heat transfer coefficients and A_{eg} and A_{ea} are the surface area of contact between the respective elements.

3.5.2 The Piston

The piston is treated similarly to the end cap, but now the elements in contact are the gas and the oil. The heat transfer to the wall element is considered negligible. The equations are:

$$\dot{T}_p = \frac{\dot{Q}_{gp} + \dot{Q}_{op}}{m_p c_{steel}} \tag{21}$$

$$\dot{Q}_{gp} = \alpha_g A_{pg} \left(T_g - T_p \right) \tag{22}$$

$$\dot{Q}_{op} = \alpha_o A_p \left(T_o - T_p \right) \tag{23}$$

3.5.3 The Wall

Similar procedures are applied to the wall elements, however, in this case, it is necessary to consider not only the convection between the wall elements and the gas, oil, and ambient, but also the heat conduction between the wall elements. The total wall heat flow is expressed in equation (24). The index i denotes each wall element ($i_{max} = 8$).

$$\dot{T}_w^{[i]} = \dot{T}_{convective}^{[i]} + \dot{T}_{conductive}^{[i]} \tag{24}$$

Convective As previously stated, the convective heat transfer from each wall element to the surroundings must be considered. It is important to note that the heat exchange from the accumulator wall and the piston is disregarded, since it is significantly smaller.

$$\dot{T}^{[i]}_{convective} = \frac{\dot{Q}^{[i]}_{aw} + \dot{Q}^{[i]}_{ow} + \dot{Q}^{[i]}_{gw}}{\rho_{steel} c_{steel} (r^2_{os} - r^2_{is}) \pi d_x} \tag{25}$$

The thickness of the wall element is now taken into account, r_{os} and r_{is} are the accumulator's inner and outer surface radius. Here d_x is introduced, which is the length of one wall element. ρ_{steel} is the density of steel. The convective heat flows are expressed by:

$$\dot{Q}^{[i]}_{ow} = \alpha_o A_{ow} \left(T_o - \dot{T}^{[i]}_w \right) \xi^{[i]}_o \tag{26}$$

$$\dot{Q}^{[i]}_{gw} = \alpha_g A_{gw} \left(T_g - \dot{T}^{[i]}_w \right) \xi^{[i]}_g \tag{27}$$

$$\dot{Q}^{[i]}_{aw} = \alpha_a A_{aw} \left(T_a - \dot{T}^{[i]}_w \right) \tag{28}$$

As the piston is moving back and forth, the heat flow for given wall element varies. At certain times, the wall comes into contact with the gas, while at other times it is in contact with the oil. To account for this dynamic behavior, two terms are introduced: ξ_o and ξ_g. These terms represent the ratio of contact with the oil and gas, respectively, for each wall element. For the oil the ratio is defined as equation (29), and for the gas, equation (30):

$$\xi^{[i]}_o = \begin{cases} 1 & \text{for} \quad x_p > id_x \\ \dfrac{x_p - (i-1)d_x}{d_x} & \text{for} \quad id_x \leq x_p \leq id_x \\ 0 & \text{for} \quad x_p < id_x \end{cases} \tag{29}$$

$$\xi^{[i]}_g = \begin{cases} 0 & \text{for} \quad x_p + l_p > id_x \\ 1 - \dfrac{x_p - l_p + id_x}{d_x} & \text{for} \quad (i-1)d_x \leq x_p + l_p \leq id_x \\ 1 & \text{for} \quad x_p + l_p < id_x \end{cases} \tag{30}$$

Where l_p is the length of the piston.

Conductive The conductive heat flow for each wall element is considered solely in the axial direction, and a uniform temperature distribution is assumed within the element. The total heat flow is derived from the Fourier's law of heat conduction, where the heat flow is proportional to the surface area and the thermal conductivity λ_{steel}. Therefore, after a series of deductions, it is finally defined as:

$$\dot{T}^{[i]}_{conductive} = \frac{\lambda_{steel}}{\rho_{steel} \cdot c_{steel}} \left(\frac{T^{[i-1]}_w - 2T^{[i]}_w + T^{[i+1]}_w}{d^2_x} \right) \tag{31}$$

It is important to note that there is a slight variation in the equation for the first and last wall elements. Specifically, when calculating the temperature change caused by conduction in the first wall element, the term $T^{[i-1]}_w$ is replaced with the temperature of the oil side end cap T_o. Similarly, for the last element, the term $T^{[i+1]}_w$ is replaced with T_e, which denotes the temperature of the gas side end cap.

The equations are then:

$$\dot{T}^1_{conductive} = \frac{\lambda_{steel}}{\rho_{steel} \cdot c_{steel}} \left(\frac{T_o - 2T^1_w + T^2_w}{d_x^2} \right) \tag{32}$$

$$\dot{T}^8_{conductive} = \frac{\lambda_{steel}}{\rho_{steel} \cdot c_{steel}} \left(\frac{T^7_w - 2T^8_w + T_e}{d_x^2} \right) \tag{33}$$

The constants used in the thermal model are presented in table 4.

Table 4: Parameter values used in the thermal model [8].

Notation	Description	Value	Unit
M_{N2}	Molar mass of nitrogen	2.8×10^{-2}	$\left(\frac{kg}{mol}\right)$
c_{N2}	Specific heat capacity of nitrogen	1040	$\left(\frac{J}{kg \cdot K}\right)$
m_e	End-cap mass	28.45	(kg)
c_{steel}	Specific heat capacity of steel	490	$\left(\frac{J}{kg \cdot K}\right)$
ρ_{steel}	Density of steel	7800	$\left(\frac{kg}{m^3}\right)$
λ_{steel}	Density of steel	58	$\left(\frac{W}{m \cdot K}\right)$
α_a	Heat transfer coefficient (air to steel)	35	$\left(\frac{W}{m^2 \cdot K}\right)$
α_o	Heat transfer coefficient (oil to steel)	140	$\left(\frac{W}{m^2 \cdot K}\right)$
α_g	Heat transfer coefficient (gas to steel)	65	$\left(\frac{W}{m^2 \cdot K}\right)$
A_{eg}	Area between end-cap and gas	2.54×10^{-2}	(m^2)
A_{ea}	Area between end-cap and air	9.18×10^{-2}	(m^2)
A_{pg}	Area between piston and gas	2.71×10^{-2}	(m^2)
r_{is}	Inner radius of accumulator	9×10^{-2}	(m)
r_{os}	Outer radius of accumulator	11×10^{-2}	(m)
n	Number of wall elements	8	$(-)$
d_x	Length of wall element	0.123	(m)

[1] α_a, α_o, and α_g are tuning parameters that were found during the model validation.

3.6 Model Validation

Validating the model is important to ensure that the accumulator system is accurately represented. A more accurate model means that the neural network can be trained using simulation data, and it will still perform greatly in the real system. This not only saves time but also reduces costs associated with implementing the neural network-based FDD method. The validation process includes assessing the load model, comparing the two thermal models discussed in section 3.4.2, and evaluating the overall system behavior by comparing it with experimental data.

3.6.1 *Load Model Validation*

As mentioned in section 3.1, the load flow q_{lo}, which is a model's input, is generated during simulation. Load profiles based on real pitch system data were provided by the laboratory. In order to validate the load generation, a visual comparison is made. The achieved load stochastic behavior is presented in figure 6.

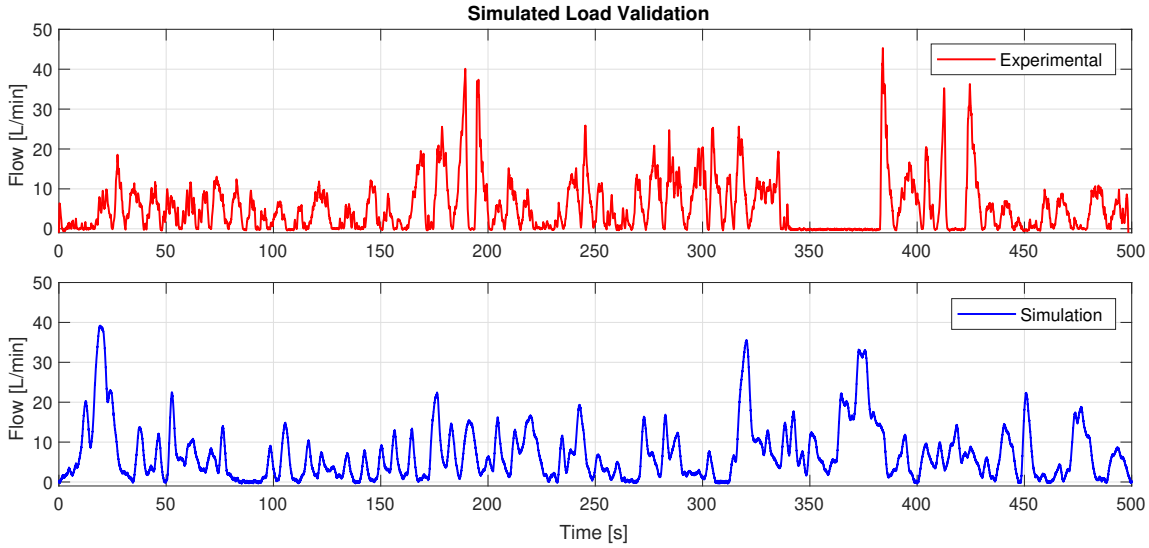


Figure 6: Simulated and experimental load.

While the simulation model may not be an exact match, it effectively captures the load behavior observed in the experimental data. The model successfully replicates the stochastic nature of the load, including high peaks that can reach up to 40 L/min. However, in order to enhance the time performance of the simulation, certain high-frequency components were suppressed. Overall, the simulation model provides a satisfactory representation of the load based on the experimental data.

3.6.2 *Thermal Model Comparison*

Two thermal models have been presented in section 3.4.2 and 3.5. The first model is simpler and quicker to implement, relying on rough estimations of the thermal time constant τ. The second model, while more precise, requires the installation of 8 thermocouple sensors on the surface of the accumulator. In figure 7, a comparison is shown between the estimated gas temperature obtained from both thermal models.

Thermal model 2 provides a more detailed understanding of the temperature distribution within the accumulator. Since the oil and gas exhibit different heat transfer coefficients, the temperature measured on the accumulator surface will change as the piston moves. Hence, this method offers a smart alternative to measuring the piston position. Additionally, obtaining more temperature signals is beneficial for training the neural network as it allows for a more accurate representation of the real system by incorporating additional input variables.

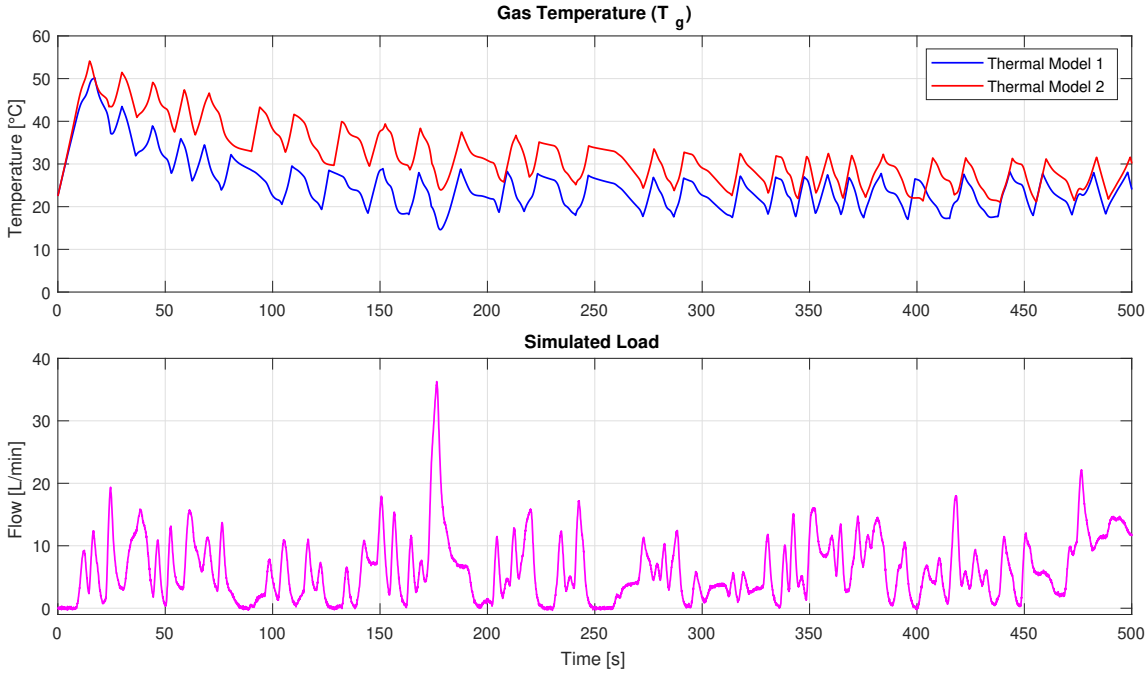


Figure 7: Accumulator thermal model comparison using gas pre-charge pressure of 130 bar. The first method uses the thermal time constant τ, and the second utilizes 8 surface temperature measurement to estimate the gas temperature.

Although both methods exhibit a similar overall trend, the first model has a faster response compared to the second model. After the first charging stage, the gas temperature goes up to similar value, about 50°C and start decreasing as the load cycle continues. The first method then settles in a permanent condition faster than the second method. The behavior of the models is influenced by the chosen values of τ, α_a, α_o, and α_g, and further tuning could enhance and approximate the result.

3.6.3 Accumulator Model Validation

A test bench, as shown in figure 8, resembling the hydraulic components and circuit of the presented model is available in the Hydraulics Laboratory at Aalborg University. For validation purposes, the system is equipped with various sensors, including thermocouples for measuring wall and end cap temperatures, piston position, supply pressure, and supply and load flow. The data used for this validation was collected by students of the 8th semester and provided by the professor Jesper Liniger from the department of Energy Technology. It includes temperature measurements from all eight wall elements, with additional measurements at the bottom of elements 6 and 8. It also contains a range of pre-charge pressures and different load conditions.

For the model validation, a specific dataset consisting of a pre-charge pressure of 175 bar, zero load flow, and a constant supply pressure of 195 bar was chosen. This dataset is unique within the provided data as it represents a scenario with zero load and a measurement duration of 1000 seconds.

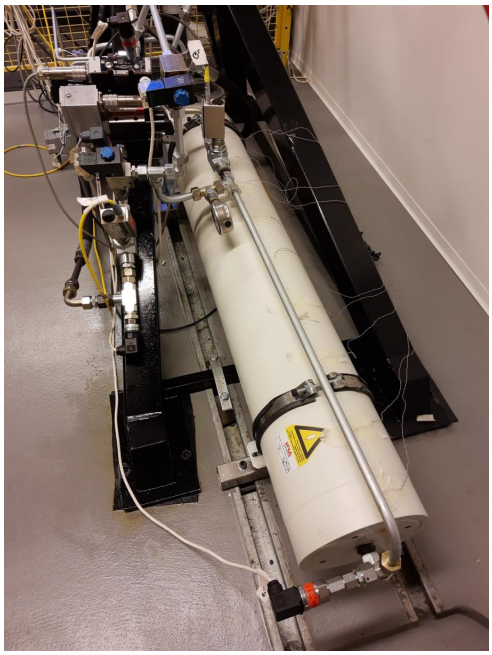


Figure 8: Test bench available in the laboratory [8].

To initiate the validation process, the initial conditions of the model need to be set as closely as possible to the conditions present during the collection of experimental data. The chosen values for this validation were based on the report [17] authored by the students who conducted the experiments. It should be noted that while the provided data is valuable, certain conditions were unknown due to incomplete documentation of the tests or limitations of the experimental system. For instance, supply pressure has an offset value hard-coded, and load flow is affected by internal leakage in the valve, leading to non-zero flow when closed [17]. The initial conditions of the model are set according to table 5.

Table 5: Model initial condition for validation.

Parameter	Description	Value	Unit
p_{pc}	Gas pre-charge pressure	175	(bar)
T_a	Ambient temperature	22.5	$(°C)$
T_o	Oil temperature	40.5	$(°C)$
x_{p0}	Piston position	0	(m)
T_{w0}	Wall temperature	22.5	$(°C)$
T_{ge}	Gas side end cap temperature	22.5	$(°C)$

After setting the initial conditions, several calculations are performed. The initial oil and gas chamber volume are found to be $V_{o0} = 0$ and $V_{g0} = 23.6$ liters, respectively. The initial specific molar volume of the gas is found using the BWR equation (9), followed by the calculation of the number of nitrogen moles, $n_{N2} = 163.6$ moles, that will remain constant during simulation. Gas mass is also calculated for verification purposes, $m_g = 4.6$ kilos.

Piston Position In order to verify the piston position, the measured supply pressure p_o is fed into the mechanical model (section 3.3), and the estimated piston position is recorded. The chosen gas thermal model is the method 2 (section 3.5), which utilizes the temperature of the eight elements to determine the gas temperature. The results are shown in figure 9. The piston position value from both simulation and experimental data is plot, as well as the residual, which represents the difference between them. The bottom graph shows the experimental oil pressure. It can be observed that the piston position residual remains within the range of ± 5 millimeter, with a noticeable offset at the steady-state regime.

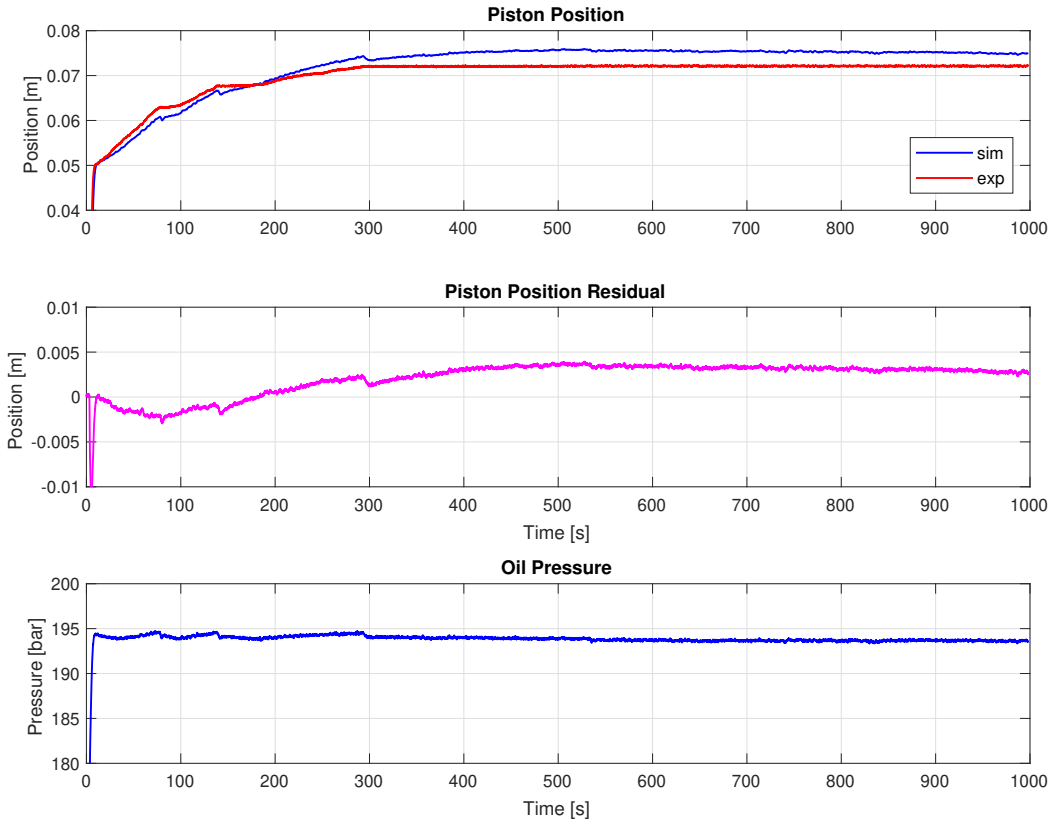


Figure 9: Simulated and experimental piston position with constant pressure supply, zero load flow and gas pre-charge pressure of 175 bar.

Gas Temperature In order to evaluate the thermal model, the wall temperatures of elements 1, 6 and 8 are compared. The results are presented in figure 10. The top graph displays the piston position and dashed lines indicating the wall elements. In this particular test, the piston is positioned within the first element. The bottom three graphs shows a comparison between the experimental and simulated temperatures. For both wall element 6 and 8, both the top and bottom temperatures are shown.

Observing the bottom graphs, it becomes apparent that the accumulator top and bottom temperatures are not equal, indicating natural convective heat transfer within the fluid. This observation contradicts the assumption of uniform temperature distribution. However, the simulated temperature resides between the top and bottom temperatures, serving as an intermediate temperature that provides a reasonable approximation.

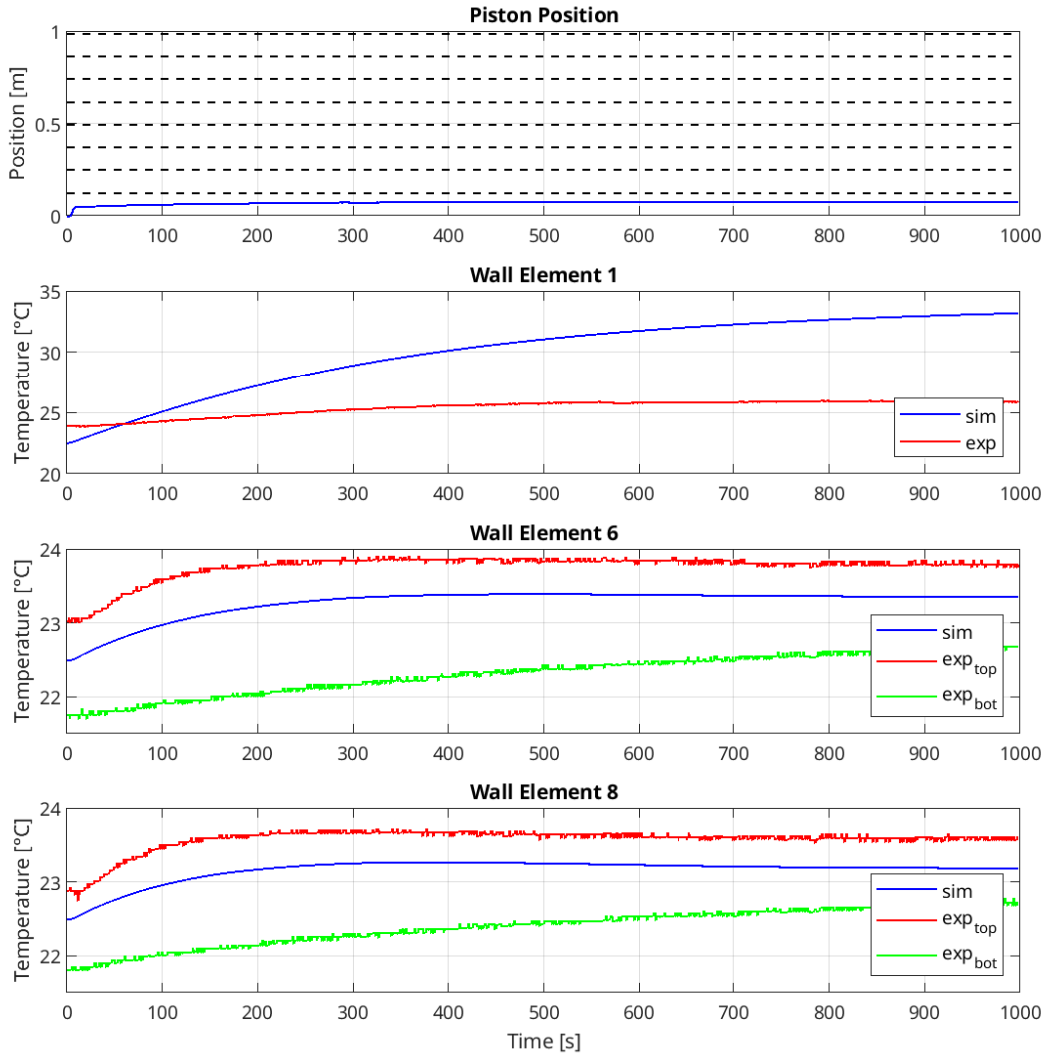


Figure 10: Simulated and experimental wall temperature with constant pressure supply, zero load flow and gas pre-charge pressure of 175 bar.

Deviations can be observed in wall element 1, which can be attributed to inaccurate assumptions regarding the uniform distribution and constant temperature of the oil, as well as the assumption that the oil end cap has the same temperature as the oil, which is 40.5°C. These deviations suggest that the high temperature of the oil may affect the temperature of the accumulator wall differently than what is observed in the experimental measurements. Another possible reason for the deviations could be inaccurate documentation of the experimental tests, as it is unclear which tests were conducted accurately or if any errors were made during the measurements.

Overall, the model can be considered validated with minor deviations. The simulated and experimental piston position showed reasonable agreement, and, although some discrepancies, the thermal model showed reasonable approximation. To further enhance the accuracy of the model, fine-tuning and refinement, along with more comprehensive and detailed experimental data, would be beneficial in practical applications.

3.7 Data Generation for Neural Network

After understanding the limitations of the model, the next step is to generate data for training the NN. Here, no experimental data is used, which means that p_o is estimated as described in section 3.2 and the load is automatically generated as explained in section 3.1. The system now simulates the behavior of pitch system when working in normal operation. The simulation data is then saved in a *.mat* file using a sampling rate of 12.5 Hz.

A simulation time of 3000 seconds is selected, offering a significant timeframe for data manipulation and preprocessing before training. The simulation process is performed across five gas pre-charge pressure levels, four ambient temperature levels, and four oil temperature levels. Furthermore, the load generator seed is altered, and the process is repeated five additional times while maintaining constant initial conditions. The total number of simulations therefore is 400. This approach ensures a diverse and comprehensive dataset for training and evaluation. The variations and their respective parameter is condensed in table 6.

Table 6: Variation parameter levels.

Parameter	Variation List
load seed	23342, 23343, 23344, 23345, 23348
p_{pc}	70, 90, 110, 130, 150 bar
T_a	18, 22, 26, 30 ° C
T_o	20.5, 30.5, 40.5, 50.5 ° C

To provide a visual representation of the generated data and aid in its comprehension, several graphs are presented below. In figure 11, the supply and load flow as well as the accumulator flow throughout the entire operation cycle is presented.

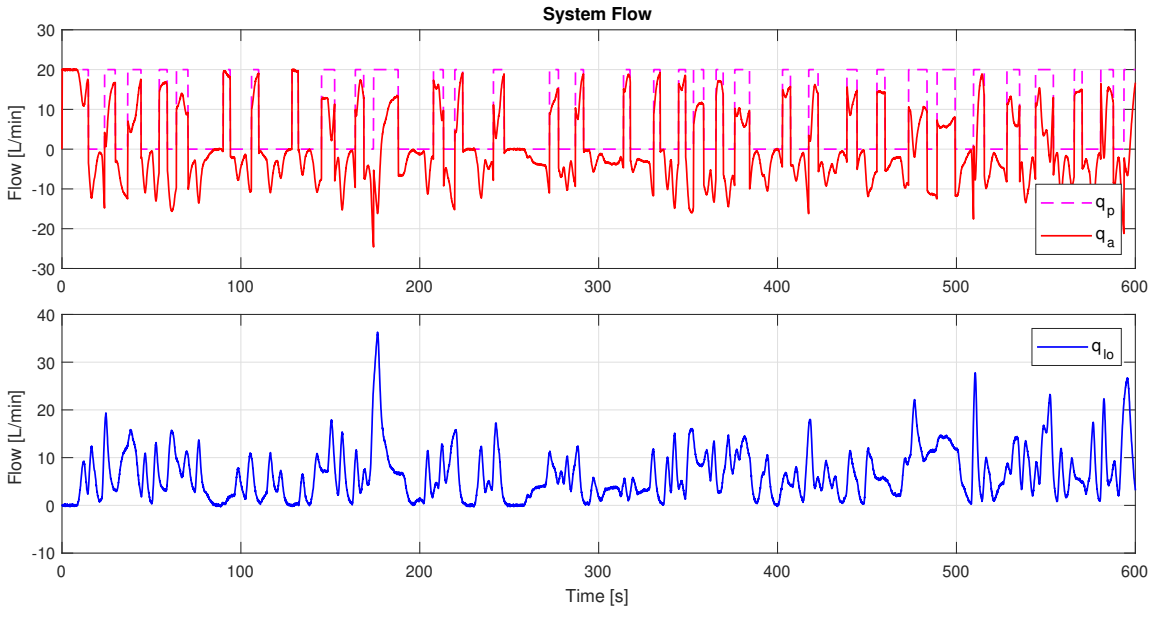


Figure 11: System flow at a pre-charge pressure of 130 bar. The load is automatically generated. The flow is considered positive when entering the accumulator.

By observing this figure, the flow behavior resulting from the activation and deactivation cycle of the fixed-displacement can be perceived. The constant flow rate of 20 L/min is activated whenever the oil pressure falls below 165 bar and deactivated when it rises to 190 bar. The gas pre-charge pressure is set to 130 bar.

With the model, it is also possible to know the gas and oil pressure, volume, and temperature, which behavior during normal operation is depicted in figure 12. It is noteworthy that the gas temperature reaches a stable state close to the ambient temperature after approximately 400 to 500 seconds. Thus, the system remains in a steady-state condition for a duration of approximately 2500 seconds.

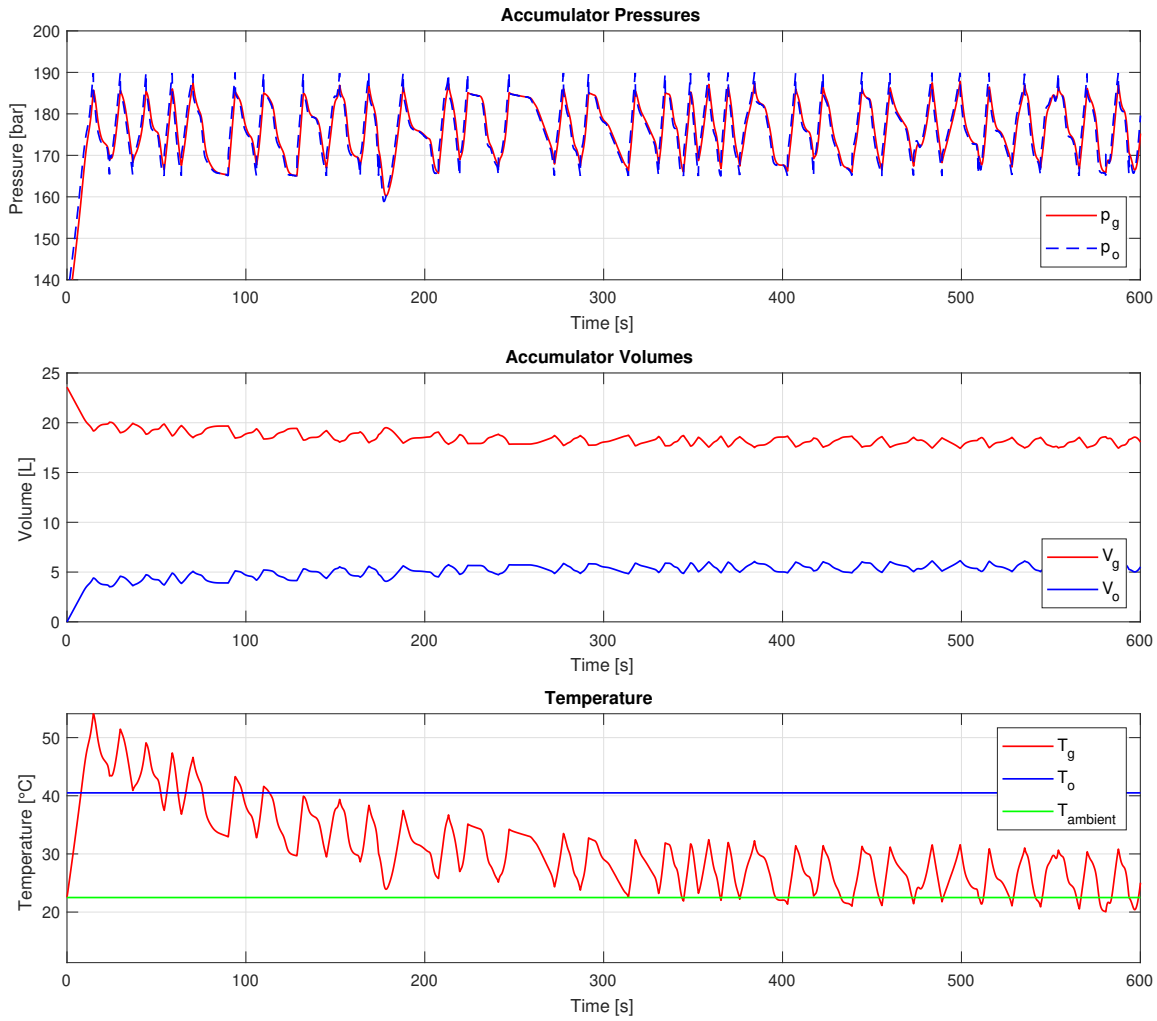


Figure 12: System pressure, volume and temperature at 130 pre-charge pressure.

In the figure, it is possible to observe pressure drops below the 165 bar limit, which aligns with the findings in [6]. These drops typically happen when there are sudden load changes, and the accumulator lacks sufficient charge to compensate for them. Moreover, the delay in pump activation intensify this process. During periods of high pitch activity, this behavior could lead to incorrect shutdowns, as the pump is deactivated and the accumulator fails to adequately support the load.

The figure 13 provides a comparison between the pressure drops of all five levels of gas pre-charge pressure. It is evident that as the gas pre-charge pressure decreases, the magnitude of the pressure drop becomes more pronounced.

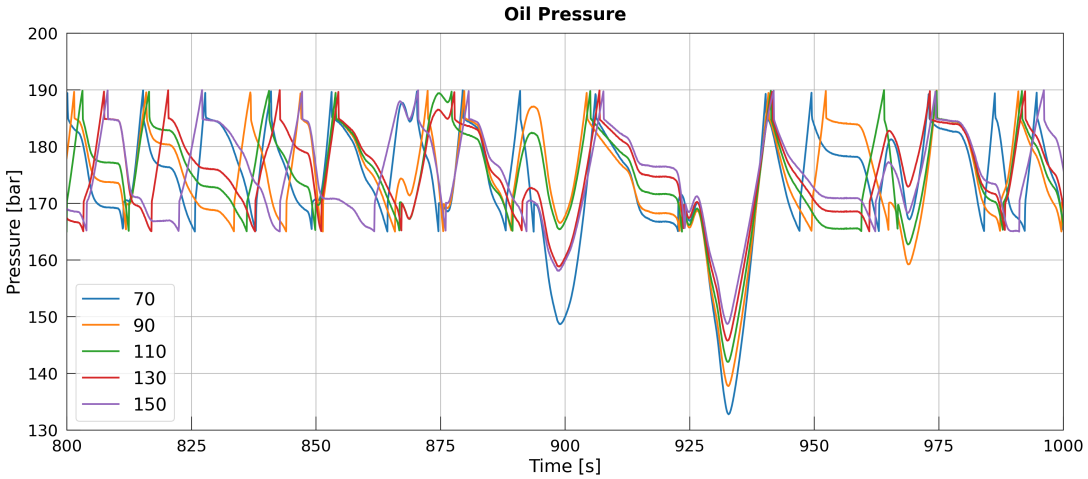


Figure 13: Pressure drop comparison for all five pre-charge levels.

In addition to the information shown previously, all the eight wall temperatures are provided. The figure 14 shows a comparison between the 8th wall element temperature for the five gas pre-charge pressures.

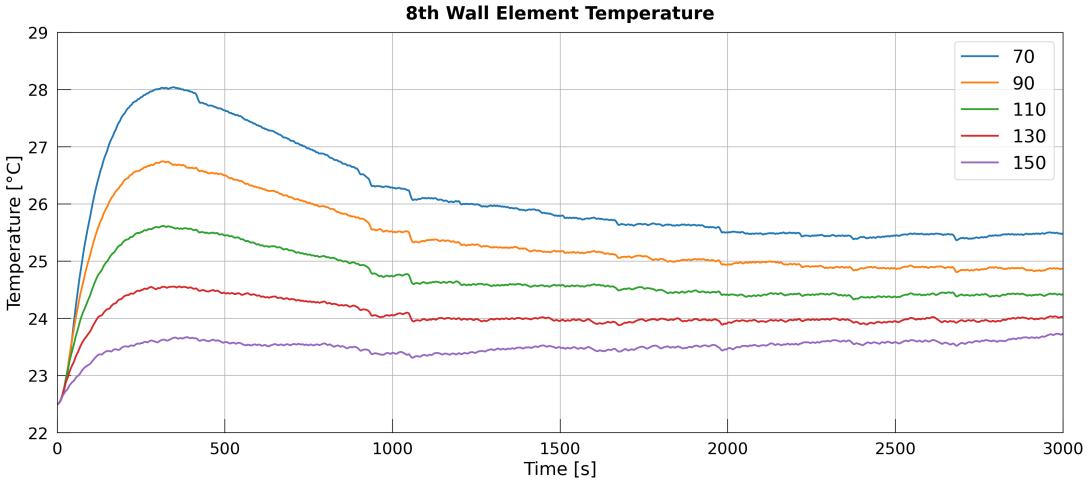


Figure 14: Eight wall element temperature comparison.

The distinct separation between these observed data behaviors suggests a promising opportunity for successful neural network training. In the next chapter 4, the methodology of constructing an effective neural network is presented, as well the methods utilized to capture and analyze these patterns. Furthermore, the results of the neural network training will be presented and discussed in chapter 5.

4 NEURAL NETWORK MODEL

This chapter provides details on modeling of a neural network for gas leakage detection. The modeling process is described in section 4.1 and includes the important characteristics of NNs and how to treat the generated data. The specific neural network architectures used in this study are presented and thoroughly discussed in section 4.2. Subsequently, the training and validation procedures are described in section 4.3. The results obtained from these trained models are presented and analyzed in the following chapter 5. Before delving into the following sections, it is important to provide a clear understanding of key definitions and terminology commonly used in the field of machine learning. A comprehensive overview of these concepts can be found in appendix A.

4.1 Design of Neural Networks

4.1.1 Semi-physical Modeling

NNs offer exceptional capabilities in modeling nonlinear systems due to their property of parsimonious approximation. This means they provide accurate results with minimal parameters. Compared to conventional methods like polynomial approximation, NNs demonstrate a linear increase, instead of exponential, in the number of variables as parameters grow. This efficiency is particularly advantageous for models with numerous parameters, allowing NN to achieve comparable accuracy using fewer parameters or input data. This attribute becomes even more valuable in models involving more than two inputs. [22]

NNs are often classified as black-box models because they can learn complex relationships between inputs and outputs without explicit knowledge of the underlying phenomena. In contrast, knowledge-based models, also known as white-box models, incorporate theoretical knowledge to predict and explain a phenomenon. These models are mathematical representations based on fundamental equations from relevant fields like physics or chemistry. They offer transparency by explicitly representing the underlying principles and relationships, providing interpretability and insights into the decision-making process. [22]

Semi-physical models, or grey-box models, are a balance between black-box and white-box modeling approaches. They combine empirical data with domain knowledge or theoretical understanding. Including measurement data offers an advantage as the knowledge-based model may not capture all phenomena accurately. And by utilizing a knowledge-model, the semi-physical model typically requires less experimental data to reliably estimate its parameters compared to a black-box model. [22]

The design of a semi-physical neural model typically involves three steps: [22]

- **Step 1:** Construction of a discrete-time knowledge-based model.
- **Step 2:** Training the NN model using the results obtained from the knowledge-based model to determine appropriate initial parameter values.

- **Step 3:** Training the NN model using experimental data to improve its performance and accuracy for real-world applications.

The type of modeling pursued in this thesis is now evident. The utilization of accumulator model developed in Matlab as the foundation for training the NN signifies a grey-modeling approach. However, as stated in section 2.6, the available experimental data is limited and will not be utilized for additional training and testing of the NN model.

4.1.2 Design Procedure

The process of working with data and designing neural networks consists of multiple steps, which, based on [22, 23], are outlined below.

1. **Define and Prepare Problem:** This step focuses on understanding the problem, identifying relevant inputs, and gathering all the necessary data for training and testing. It involves identifying the questions the data should answer.
2. **Summarize and Understand Data:** Here, the collected data is analyzed and summarized to gain insights and identify trends. Descriptive statistics and visualizations can be used to assist in the investigation.
3. **Process and Prepare Data:** In this step, the data is processed and manipulated to make it suitable for analysis. Tasks such as cleaning the data, selecting relevant features, and transforming the data are performed.
4. **Evaluate Algorithms:** Neural network algorithms are evaluated to find the best ones that can effectively exploit the data structure. The complexity of the model, including the number of hidden neurons, and the parameters to minimize the cost function are determined. Evaluating the generalization ability of the network is also performed.
5. **Improve Results:** Features and models are selected and optimized to enhance the accuracy and performance of the predictions. This step involves parameter tuning for individual algorithms and ensemble techniques for combining multiple models.

While not all tasks within these steps may be utilized in this study, they provide a base framework for further development. The ultimate goal is to create a model that can accurately predict unseen data. Once achieved, the model can be deployed in a production environment. The following subsections aim to clarify the type of data that the neural network is handling, serving as an introduction and establishing the basis for the chosen methods in this study.

4.1.3 Understanding the Data

The data generated by the accumulator model comprises signals that exhibit changes in value over time. These sequences of data points, indexed by time, are commonly referred to as time series data. Mathematically, a univariate time series X can be defined as an ordered collection of T pairs consisting of measurements x and timestamps t. [24]

$$X = \{(x_1, t_1), (x_2, t_2), \ldots, (x_T, t_T)\} \tag{34}$$

Therefore, each of the signals provided by the accumulator simulation, such as p_o, T_w, q_{lo}, and so on, corresponds to a distinct univariate time series X, also known as features in the data series. According to the definition available in [25], the combination of M distinct univariate time series X is called multivariate time series, and can be expressed as:

$$\boldsymbol{X} = \left[X^1, X^2, \ldots, X^M\right] \tag{35}$$

Consequently, in the case of this thesis, the concatenation of all features X in the generated data produces the final multivariate vector $\boldsymbol{X}$, as shown below.

$$\boldsymbol{X} = \left[q_p, q_{lo}, q_{ac}, p_g, T_g, V_g, p_o, V_o, x_p, dx_p, T_a, T_{ge}, T_{w1}, \ldots, T_{w8}\right] \tag{36}$$

As previously mentioned, the primary objective of NN is to estimate the pre-charge pressure based on the available measurement signals. During the data generation process, the actual pre-charge pressure is known and recorded as the correct output of the NN, denoted as Y. This output is also commonly referred to as the ground truth value.

$$Y = 70, 90, 110, 130 \text{ or } 150 \text{ bar} \tag{37}$$

In this context, a dataset D is defined as a collection of N different pairs $(\boldsymbol{X}, Y)$ [25], where N represents the number of simulations conducted. As discussed in section 3.7, these simulations vary in terms of load cycle, oil and ambient temperature, and pre-charge pressure. Therefore, the dataset D consists of input signals $\boldsymbol{X}$ and their corresponding ground truth outputs Y.

$$D = \{(\boldsymbol{X}_1, Y_1), (\boldsymbol{X}_2, Y_2), \ldots, (\boldsymbol{X}_N, Y_N)\} \tag{38}$$

The dataset D is used for training and evaluating the neural network's performance and effectiveness in estimating the pre-charge pressure. The importance of dataset diversity allows the neural network to learn and generalize from different scenarios, enhancing its ability to handle various simulation conditions.

4.1.4 Neural Network for Time Series

NNs have brought about significant advancements in computer vision and natural language processing. Particularly in the area of image recognition and machine translation, respectively. It is important to highlight that these applications share the characteristic of dealing with sequential data, similar to time series data. The community of NN for time-series problems are much smaller, but with valuable research developed. [25]

There are two approaches to address the pre-charge estimation problem: treating it as a regression problem or as a classification problem. Classification and regression are supervised learning tasks that aim to establish the relationship between a target variable and a set of time series data. The main difference between Time-Series Regression (TSR) and Time-Series Classification (TSC) lies in their prediction outcomes. TSC predicts the probability of an item belonging to a specific class, while TSR focuses on estimating a continuous value. Examples of TSC include detecting anomalies in sensor data and recognizing human activities, while examples of TSR include predicting stock prices and forecasting energy consumption. [22, 24]

Several popular architectures are commonly used to address these types of problems, including Multi-Layer Perceptrons (MLP), Fully Convolutional Networks (FCN), Long Short-Term Memory (LSTM), and Residual Networks (ResNet). These architectures have been extensively studied and applied in the field of time series analysis and classification [24–30]. While most of these methods are typically applied for classification tasks, they can also be adapted for regression problems by making simple modifications to the output layer and selecting an appropriate loss function. [24]

In multi-class classification tasks, the output layer of a neural network often incorporates the *SoftMax* activation function, which is shown in appendix A. This activation function ensures that the probabilities for all classes sum up to 100% for a given input. By mapping the outputs of the preceding layer to the range of 0-1, it provides the probabilities of an example belonging to each class. In such cases, a cross-entropy loss function is commonly used during training. For regression problems, the output layer typically employs a linear activation function like *ReLU* or a non-linear activation function with a single value output such as sigmoid. Training for regression tasks often involves the mean square error or mean absolute error loss function [22, 24]. The training process and the mechanisms of cost functions will be further discussed in section 4.3.

As the adaptability of NNs for regression problems is a quite trivial task, for this thesis, only classification architectures are considered. It can be said that if the time series data transitions from one class to another, it is likely that gas leakage has occurred. The figure 15 shows a general approach on how to treat the time series data in a classification problem.

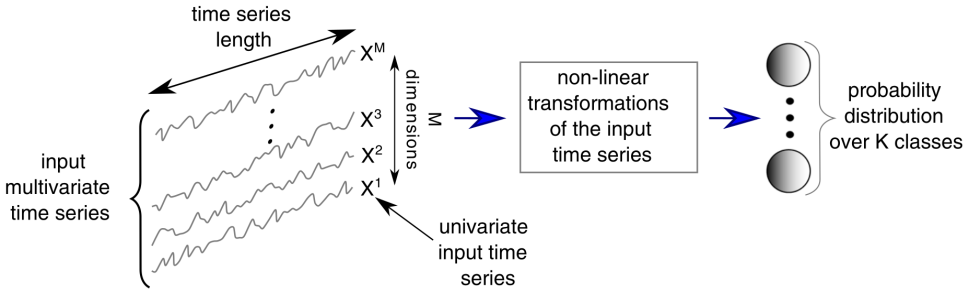


Figure 15: Time series neural network framework [25].

The input to the neural network consists of a multivariate time series, which includes signals such as p_o, T_w, T_a, and T_o. The neural network performs non-linear transformations on this data to classify it into the correct class based on the ground truth Y. The output of the neural network is a probability distribution across all classes. For example, if the output indicates a 60% probability for the pre-charge pressure to belong to the class "110 bar," it can be represented as a vector:

$$\hat{y} = [0.1, 0.1, 0.6, 0.1, 0.1]$$

Further explanation is provided below. Some methods mentioned in this section is presented in the next section 4.2. The training process is described in depth in section 4.3. The data preprocessing techniques applied are discussed in section 4.4.

4.2 The Model

This section provides an overview of the chosen neural network architecture for gas leakage detection. The work of Wang, Yan, and Oates [26] introduced three deep learning models, namely MLP, FCN, and ResNet, which served as strong baselines for TSC problems. Building upon this, Fawaz et al. [25] conducted an extensive study involving these three models along with six others, evaluating their performance on 85 univariate and 12 multivariate TSC datasets from the UCR repository [27]. These studies have served as a foundation and guided the selection of an appropriate method for this thesis, narrowing down the search for an effective approach.

Other studies that have demonstrated promising results include those by Pham [28], Xia, Huang, and Wang [29], and Sainath et al. [30]. These studies employed a combination of LSTM networks with CNN. Additionally, complex models like Inception-ResNet models, such as InceptionV4 [31] and GoogLeNet [32], have shown excellent performance in various tasks and offer potential for achieving high accuracy in gas leakage detection.

4.2.1 Promising NN Architectures

MPL is a basic and traditional deep learning model consisting of fully connected layers, meaning that each activation function is connected to all the activation functions of the previous and following layers. While they are easy to implement, MLPs are not well-suited for time series data, as each timestamp has its own weight and the temporal information is lost. [24]

FCN is based on convolutional neural networks (CNNs), which is detailed in appendix B. It is designed to keep the length of a time series unchanged throughout the convolutions. Differently then CNN, it does not contain any local pooling layers, which prevents overfitting. In addition, the last layer, witch generally is a fully connected layer, is replaced with a Global Average Pooling (GAP) layer, which reduces the spatial dimensions and further greatly reduces the number of parameters in the NN. An advantage of this architecture is that a class activation map (CAM) can be used to visualize the parts of the input data that have the most significant impact on the training and performance of the NN. [24, 26]

ResNet is a deep architecture known for its depth. It consists of three residual blocks followed by a Global Average Pooling (GAP) layer and a SoftMax classifier. Each residual block resembles the FCN architecture. [24] ResNet models have demonstrated excellent performance on TSC tasks and are a promising option for solving the gas leakage problem.

LSTM network is an improved version of Recurrent Neural Networks (RNN), developed to solve the problem of vanishing gradients while remembering information for long periods of time. It incorporates three gates, controlled by sigmoid activations, to decide which information is added, kept, or discarded. This architecture is particularly well-suited for capturing long-term dependencies and sequential patterns in time series data, making it a valuable tool in this context. [33]

The studies mentioned at the beginning of this section have shown that MLPs do not perform as reliably as other methods for TSC tasks, therefore they will not be considered for this thesis. While ResNet has shown excellent performance, its implementation and understanding are more complex, making it impractical within the given time constraints. Both FCN and LSTM are relatively simple to implement and have demonstrated good performance in other TSC applications. Both the FCN and LSTM models have shown good performance in various TSC applications and are relatively simple to implement. However, in order to conduct an extensive set of experiments using the provided dataset, this thesis focuses solely on the development and testing of the FCN model.

4.2.2 Classification using FCN

The FCN model used in this study is inspired by the FCN model introduced by Wang, Yan, and Oates [26] and incorporates modifications based on the Keras documentation [34]. As mentioned in the previous section, the FCN is essentially a CNN, but without local pooling after each convolution and with a GAP instead of fully connected layer. The SoftMax activation function is still applied for classification. The architecture of the FCN model designed for TSC tasks is illustrated in 16.

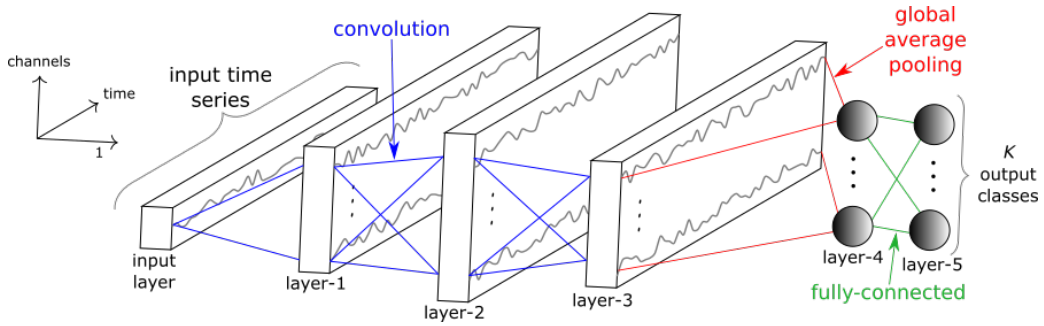


Figure 16: FCN architecture for time series classification [25].

Input Layer The input layer receives the multivariate time series which then is passed through three blocks of convolution. According to Wang, Yan, and Oates [26], the FCN is designed to minimize the need for extensive data preprocessing. However, as shown is chapter 5, the performance of the NN will be evaluated using different input data sizes. Moreover, by omitting local pooling layers, the length of the time series remain unchanged during the convolutional operations. [23]

Convolution Block The convolution block in the FCN architecture consists of three layers: a convolutional layer, a batch normalization layer, and a ReLU activation layer. The convolutional layer applies a sliding window (filter/kernel) over the time series to detect patterns by computing the dot product between the filter weights and the input values. The batch normalization layer is included to accelerate convergence and enhance generalization performance. Finally, the ReLU activation layer introduces non-linearity to the output of the batch normalization layer, allowing the network to learn complex relationships in the data. [23, 25]

The mathematical representation of this convolution block is shown below.

$$y = W \otimes x + b \tag{39}$$

$$s = BatchNorm(y) \tag{40}$$

$$h = ReLU(s) \tag{41}$$

Where $\otimes$ is the convolution operator. There are three hyperparameters that can be used to configure the convolution filters: [23–25]

- The *Filter Size* defines the length of the filter. For example, setting it to '3' results in applying a moving average with a sliding window of length '3' time stamps.
- The *Stride* determines the number of time stamps to move forward while performing convolution.
- The *Padding* parameter is used to preserve the length of the input data when applying a filter or sliding window. It involves adding zeros to the borders of the time series before performing the convolution.

Average global pooling The GAP layer after the third convolutional block computes the average across the time dimension, resulting in a single value as the output. This average global pooling operation reduces the dimensionality of the time series data and allows for capturing the overall presence of important features. The Class Activation Map (CAM) technique can be applied to identify the specific regions in the raw data that contribute to the predicted labels. However, due to time constraints, the implementation of the CAM is not included in this study. [23–25]

Output Layer The SoftMax layer, explained in section 4.1.4 and appendix A, produces the final output of the model. This layer applies an activation function that transforms the input into a vector of probabilities. The sum of these probabilities is always equal to 100%. Each element in the vector corresponds to a specific class. For instance, considering the classes defined in equation (37), if the output vector is $\hat{y} = [0.93, 0.07, 0, 0, 0]$, it means that there is a 93% probability that the input belongs to the "70 bar" class and a 7% probability that it belongs to the "90 bar" class. It is important to note that, as the model does not approach the gas pre-charge pressure as a regression problem, the output does not provide a single real number representing the pressure value.

The hyperparameters used in the FCN model, as provided by Wang, Yan, and Oates [26], include 128 filters of length 8 for the first convolutional layer, 256 filters of length 5 for the second, and 128 filters of length 3 for the third. To maintain the original length of the time series, the convolutions utilize a stride of 1 and zero padding.

During the development and testing of the FCN model, issues related to memory consumption and crashes were encountered. It was observed that the provided hyperparameters resulted in approximately 130,000 trainable parameters. This high parameter count may have posed challenges in terms computational resources, especially considering the hardware limitations. Nevertheless, the research paper does not mention how these hyperparameters were determined. Therefore, an alternative variation of the model is employed.

To address the complexity of training and deploying deep learning models, hyperparameter tuning can be employed. The Keras documentation [34] suggests the use of KerasTuner [35], a tool that automatically searches for optimal hyperparameters for a given dataset. The FCN model discussed earlier can also be found in the Keras documentation. However, in their approach, they used KerasTuner to explore different sets of hyperparameters. Through this tool, they determined that using 64 filters with a kernel size of 3 for all three convolutional layers was the optimal configuration.

Although the tuning process was not conducted for the dataset presented in this thesis, the number of trainable parameters was reduced to approximately 30,000. Through experimentation, it was discovered that the new set of hyperparameters yielded improved performance, resulting in a significantly faster training process.

Table 7 provides an example of the layers used in the FCN model, along with the corresponding parameter counts. As shown in the "Input Shape" column, this example assumes 6 input variables and a time step of 3906 samples. The "None" value in the table indicates that there is no specific limitation on the quantity of input data, allowing for flexibility in the model's input size.

Table 7: Fully Convolutional Network output from compiling the model in TensorFLow.

Layer	**Input Shape**	**Parameters**
Input Layer	(None, 3906, 6)	0
Convolutional 1D	(None, 3906, 64)	1216
Batch Normalization	(None, 3906, 64)	256
ReLU	(None, 3906, 64)	0
Convolutional 1D	(None, 3906, 64)	12352
Batch Normalization	(None, 3906, 64)	256
ReLU	(None, 3906, 64)	0
Convolutional 1D	(None, 3906, 64)	12352
Batch Normalization	(None, 3906, 64)	256
ReLU	(None, 3906, 64)	0
Global Average Pooling 1D	(None, 64)	0
Dense (SoftMax)	(None, 5)	325

[1] Total params: 27,013
[2] Trainable params: 26,629
[3] Non-trainable params: 384

4.3 The Training Process

The two use cases of a newly created neural network model are: learning and prediction. During the learning phase, the model's weights and biases are iteratively updated until the predicted output converges with the desired output. Both supervised and unsupervised training methods are feasible, although only supervised learning will be implemented in this thesis, since the ground truth value Y is known for all simulations.

Prior to initiating the training process, it is crucial to split the available dataset into three subsets: training, validation, and testing sets. The training set, containing the majority of the data, is employed to train the neural network. The validation set, a small portion of the data, is employed during training to evaluate the model performance. Finally, the testing set is used in the evaluation of the trained model, where the performance on new and unseen data is measured. [23]

The most common algorithm responsible to perform the training is called *backpropagation*. It calculates the gradient of the cost function with respect to the weights and biases of the network, and uses this gradient to update the network's parameters to minimize the cost function. [22] The complete learning process is summarized in the figure 17 and will be subject of discussion in this section. [23]

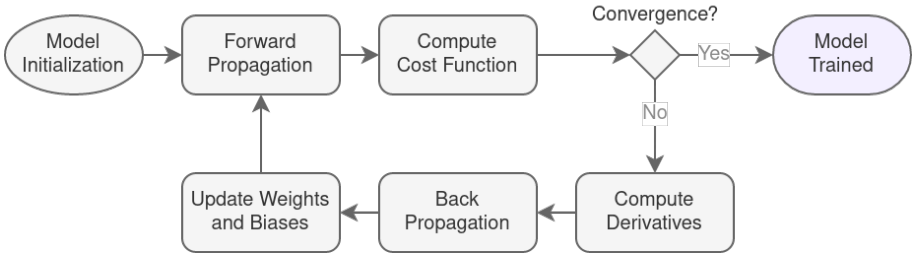


Figure 17: Backpropagation process for training the neural network.

4.3.1 *Model Initialization*

Model initialization is an essential step in training a neural network. The weights and biases need to be initialized with some initial values before the training process starts. It is common practice to initialize weights and biases with small random values that are close to zero. This is because initializing all weights with the same value can lead to symmetry breaking, which in turn reduces the capacity of the network. Also, initializing weights with large values can cause saturation of the activation function, leading to a very slow learning process. [22]

4.3.2 *Forward Propagation*

This stage involves passing the input values through the neural network and computing the corresponding output. It is named as such due to the sequential flow of computation from the input layer to the output layer. The activation function of each neuron generates an output, which serves as the input to the subsequent layer or the output of the final layer.

The predicted neuron output $\hat{y}$ is found by, generally, equation (42). [22]

$$\hat{\boldsymbol{y}}(x, w, b) = f\left(b + \sum_{j=1}^{m} w_j \cdot x_j\right) \tag{42}$$

Where f is the activation function, x_j are the input variables, w_j are the weights and b is the bias. The total number of input variables is represented by m, and j is the index in the variables list.

4.3.3 *Cost Function Calculation*

The cost function C_i, which can also be referred to as the residual or loss function, provides a way to evaluate how well the calculated output of a neural network matches the correct known output for a specific training example i. In simpler terms, it measures the degree of similarity between the predicted output $\hat{y}_i$ and the actual output y_i for that particular example. Expressed in equation (43), the cost function C is the average of all loss function values across all training examples n. [12]

$$C = \frac{1}{n} \sum_{i=1}^{n} C_i \tag{43}$$

Loss functions are divided into two categories: regression losses and probabilistic losses. For regression problems, a common way to calculate the loss is using the *least square error* method presented in equation (44). [23]

$$C_i = \frac{1}{2} \left(y_i - \hat{y}_i\right)^2 \tag{44}$$

In classification tasks, cross-entropy techniques are employed. The *binary cross-entropy* is utilized when predicting values between two classes, while the *categorical cross-entropy*, as defined in equation (45), is used for multi-class classification problems. [33]

$$C_i = \sum_{k=1}^{K} y_{i,k} \log\left(\hat{y}_{i,k}\right) \tag{45}$$

Where K is the number of classes, $y_{i,k}$ is the true value for the i-th example and $\hat{y}_{i,k}$ is the predicted probability for whether the i-th example belongs to the k-th class. When the predicted output of the network closely matches the true output, the cost function approaches zero. Therefore, minimizing the cost function is desired. The optimal value for weights and biases provides a minimum cost function, thus, the training process is an optimization problem.

Generally the cost function is computed for both training and validation datasets on every iteration. Their values can be plotted against the iteration step to assist in the performance evaluation of the NN. Training usually stops when the validation error starts to increase after an initial decrease, indicating overfitting of the model. Additionally, other loss functions, such as *accuracy*, *root mean squared error*, can be employed during the training and later on the testing of the NN. These are called *Metrics*, and provide additional performance measures that are used to evaluate the model's accuracy and effectiveness. It is important to note that these metrics do not directly impact the parameter updating. [23, 34]

4.3.4 *Backpropagation*

The backpropagation method is divided into three parts: computation of partial derivatives, computation of the optimization algorithm, and update of weight and biases. The following equations and explanation are derived from [12, 23].

Backpropagation starts by computing the derivative of the cost function with respect to the output of the last layer. This derivative is then propagated backwards through the network, layer by layer, using the chain rule of differentiation. In this way, the derivative of the cost function with respect to each weight and bias in the network can be obtained. Consider the system presented in figure 18.

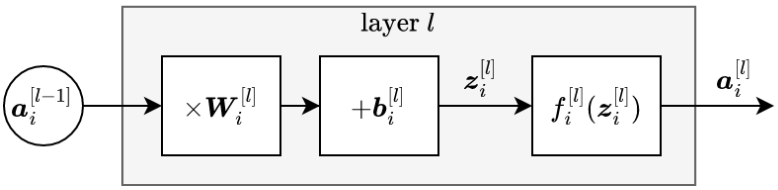


Figure 18: Forward pass of layer l.

The variables in bold represent the multiple neurons in the layer. For a layer l the input $\boldsymbol{a}^{[l-1]}$ comes from the previous layer $l-1$ output. Then, it is multiplied by the weights $\boldsymbol{W}$ and summed by bias $\boldsymbol{b}$. The result $\boldsymbol{z}_i^{[l]}$, called weighted input, is passed to the activation function f. Note that, for the last layer L, $\boldsymbol{a}_i^{[L]}$ is equivalent to $\hat{\boldsymbol{y}}_i$. Therefore, the derivative of the cost function C with respect to $\boldsymbol{W}$ and $\boldsymbol{b}$ are found using the chain rule. The subscript i is suppressed for readability.

$$\frac{\partial C}{\partial \boldsymbol{W}^{[l]}} = \frac{\partial C}{\partial \boldsymbol{z}^{[l]}} \cdot \frac{\partial \boldsymbol{z}^{[l]}}{\partial \boldsymbol{W}^{[l]}} = \boldsymbol{\delta}^{[l]} \cdot \boldsymbol{a}^{[l-1]} \tag{46}$$

$$\frac{\partial C}{\partial \boldsymbol{b}^{[l]}} = \frac{\partial C}{\partial \boldsymbol{z}^{[l]}} \cdot \frac{\partial \boldsymbol{z}^{[l]}}{\partial \boldsymbol{b}^{[l]}} = \boldsymbol{\delta}^{[l]} \tag{47}$$

Here, the variable $\boldsymbol{\delta}^{[l]}$ is introduced, which describes the sensitivity of the cost function toward a change in the neuron's weighted input $\boldsymbol{z}^{[l]}$. It also denotes a vector of errors associated with layer l. The derivative of the weighted input $\boldsymbol{z}^{[l]}$ in respect to the weight $\boldsymbol{W}^{[l]}$ is proven to be equal $\boldsymbol{a}^{[l-1]}$ and the derivative in respect to the bias $\boldsymbol{b}^{[l]}$ is proven to be equal 1 in [12].

Now, in order to find the value for the error $\boldsymbol{\delta}^{[l]}$, the following expression is used: [12, 23]

$$\boldsymbol{\delta}^{[l]} = \boldsymbol{W}^{[l+1]} \cdot \boldsymbol{\delta}^{[l+1]} \cdot f'(\boldsymbol{z}^{[l]}) \tag{48}$$

By knowing the quantities $\boldsymbol{\delta}$, the derivative of the cost function C in respect to the parameters $\boldsymbol{W}$ and $\boldsymbol{b}$ are found. By using these derivatives, an optimization algorithm can update the weights and biases to minimize the cost function and improve the network's performance. In the next subsection, further explanation is given on how such optimization algorithms are employed.

4.3.5 *Updating Parameters*

There are various optimization algorithms available to train neural networks. The choice depends on various factors, including the size of the data, the type of neural network, and the desired accuracy. Examples of popular optimization algorithms include Gradient Descent, Adam, and Stochastic Gradient Descent.

Gradient descent is an iterative method that is employed to update the model parameters by following the direction of the steepest descent of the cost function. It takes big steps when it is far away from the minimum, and short steps when close to it. The previous calculated partial derivatives correspond to the slope of the cost function. The gradient decent equation is shown below: [12]

$$\boldsymbol{W}_{t+1} = \boldsymbol{W}_t - \alpha \cdot \frac{\partial C}{\partial \boldsymbol{W}} \tag{49}$$

The symbol t represents the index of the epoch iteration and α denotes the learning rate, which regulates the size of the step taken in the optimization process. It is worth noting that a large learning rate can hinder convergence of the algorithm, while a small one can result in slow convergence. Additionally, computing all partial derivatives for the entire training set can incur high computational costs. [12]

Adam (Adaptive Moment Estimation), proposed originally by Kingma and Ba [36], is widely used and efficient stochastic optimization algorithm that incorporates momentum and adaptive learning rates. It is an adaptive method that adjust the learning rate for each parameter automatically during training, which leads to faster convergence and better generalization performance. It is particularly useful when dealing with large datasets and complex models. The update equation described below are based on [33, 36]:

$$\theta_{t+1} = \theta_t - \alpha \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon} \tag{50}$$

Where θ_t are the parameters at iteration t, α is the learning rate, and ϵ is a small value added for numerical stability. The variables $\hat{m}_t$ and $\hat{v}_t$ are the first and second bias-corrected moment estimates of the gradients, and are calculated as following:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{51}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{52}$$

The bias correction is necessary because the moments are initialized as zero, which can cause a bias towards zero in the early iterations. The first and second moment are:

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{53}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{54}$$

Where g_t is the gradient at time t. β_1 and β_2 are hyperparameters that control the exponential decay rates of the moving averages of the gradient and the squared gradient, respectively. The aforementioned hyperparameters typically are $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\alpha = 0.001$, and $\epsilon = 10^{-8}$.

4.4 Data Preprocessing

Data preprocessing is an essential step in preparing the data for training the neural network models. It involves transforming the raw data into a format that is suitable for the learning algorithms and maximizing the model's effectiveness. In section 4.1.3, where the initial analysis of the data is presented, it is established that the time series data is considered as the NN's input $\boldsymbol{X}$, while the corresponding gas pre-charge pressure serves as the output Y. This section discusses the data preprocessing utilized to treat these inputs and output, which begins by applying a sliding window technique for time series data, then scaling the input and output with the *Robust Scaler* and *One-Hot Encoder*, respectively.

4.4.1 Sliding Window

As mentioned in section 3.7, the simulation data covers a time range of 3000 seconds. However, it is important to note that this range may not be optimal for training certain types of NN architectures. Long sequences of data can cause difficulties for NNs in capturing temporal dependencies and patterns effectively. Furthermore, using long sequences as input can lead to increased memory consumption and potential crashes during training.

Unlike traditional CNNs, which have limitations in retaining past data and may face challenges with large input vectors, the FCN method is well-suited for this particular application. One of its advantages is that it does not alter the length of the time series during the convolution process, allowing for more effective modeling of temporal dependencies. [23, 33] Although FCN is built specifically for TSC tasks, in order to assist in the challenge of capturing temporal dependencies and patterns, a sliding window method is employed.

This technique is commonly used in prediction and forecasting problems [37–39], as it enables the creation of labels based on the input data, thereby transforming it into a supervised learning problem. Similarly, CNNs make use of kernels or filters that also slide across the input data to extract features and capture spatial information. [23, 33]

The sliding window process involves sliding a fixed-size window over the time series, extracting subsets of data step by step to create input-output pairs. It reshapes the data by reducing the input time range and increasing the number of examples. An illustration of this process is presented in figure 19. This visual representation helps to illustrate how the sliding window captures different subsets of the dataset as it moves along the time series.

It is crucial to carefully determine the window size and step size to capture relevant patterns without excessive data overlap or information loss. In chapter 5, various window sizes are tested to evaluate their impact on the performance of the NN. To ensure consistency and remove the transient regime, the first 500 seconds of the dataset are excluded, resulting in a dataset with 2500 seconds $\times 12.5$ Hz $= 31251$ samples for each time series. The window sizes used for the experiments are determined as outlined in table 8. The step sizes for sliding the window are set to one-fifth of the window size.

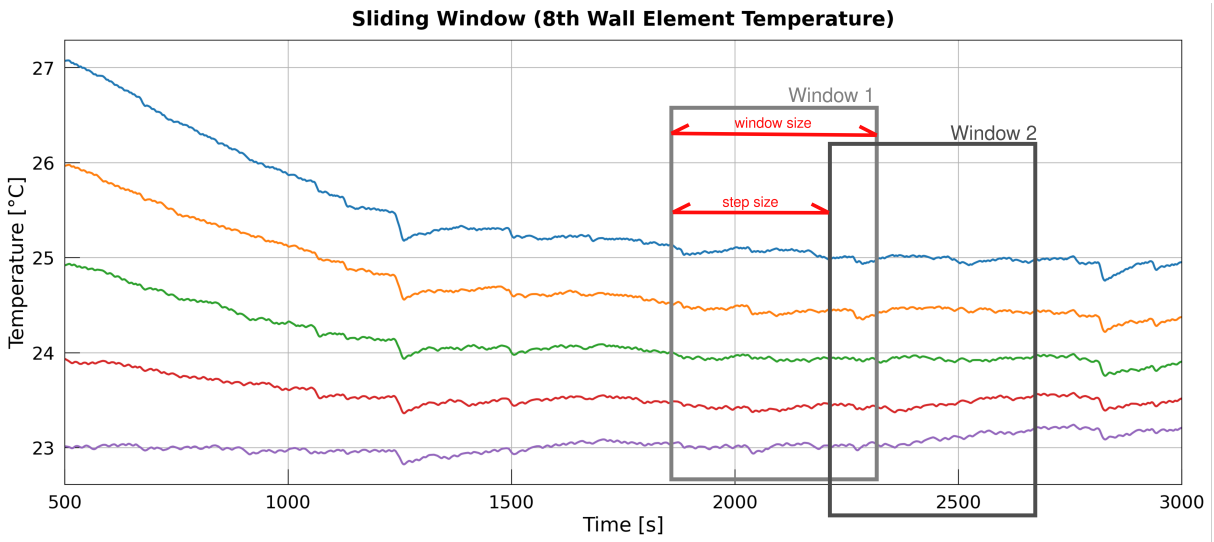


Figure 19: Representation of the sliding window in the T_{w8} time series.

Table 8: Window and step sizes used to train the neural network.

Window Size		Step Size	
(seconds)	**(samples)**	**(seconds)**	**(samples)**
$2500/2 = 1250$ s	15625	250 s	3125
$2500/4 = 625$ s	7812	125 s	1562
$2500/8 = 312.5$ s	3906	62.5 s	781
$2500/16 = 156.25$ s	1953	31.25 s	390

4.4.2 Robust Scaler

When preparing a dataset for machine learning, it is often necessary to standardize the data to ensure consistent ranges and distributions. A scaler is often used to scale numerical input features, making them comparable in terms of range and distribution. While the typical approach involves removing the mean and scaling to unit variance, this method can be sensitive to outliers and yield suboptimal results. [23, 40]

One preprocessing technique that addresses these concerns is the Robust Scaler. It is a type of input scaler that eliminates the median and scales the data based on the interquartile range (IQR). The IQR represents the range between the 25th and 75th percentiles, which helps capture the central tendency and variability of the data while being less influenced by extreme values. [40]

An example demonstrating the impact of scaling the temperature of the 8th wall element using the Robust Scaler is depicted in figure 20. Similarly, other input features like p_o, q_{lo}, and others are also scaled using the same procedure. This ensures that all input variables are adjusted to a consistent range, facilitating accurate analysis and modeling while diminishing the change of exploding gradients.

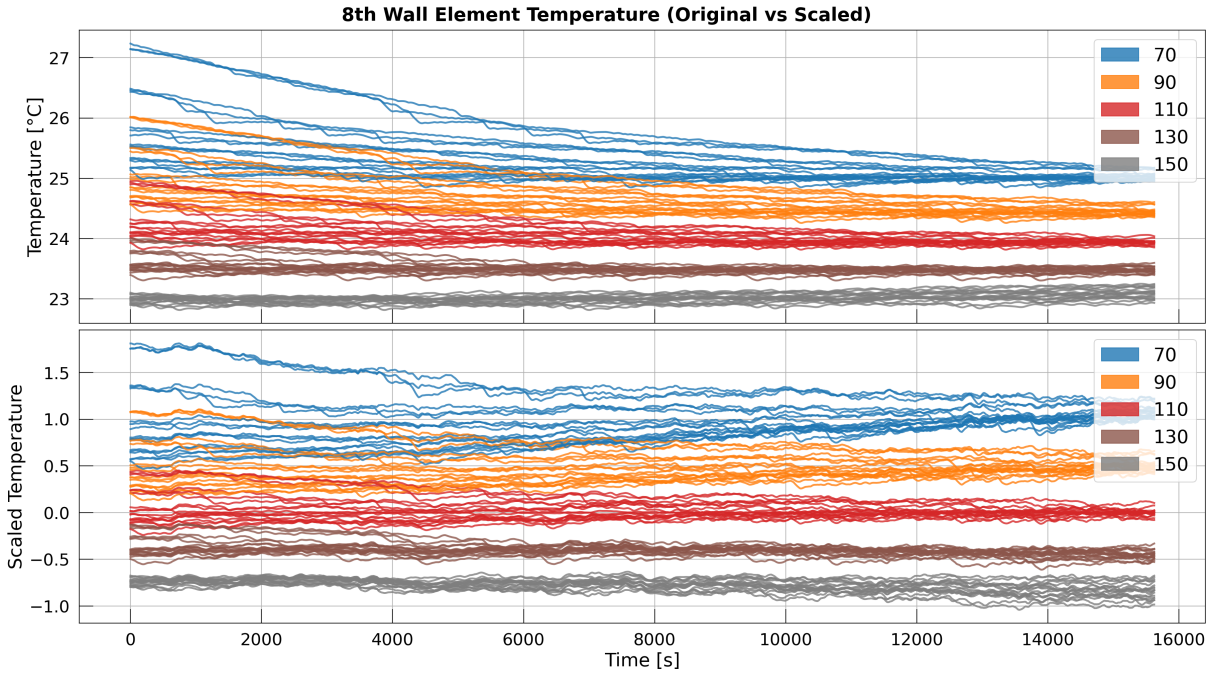


Figure 20: Original and scaled sequences of data for T_{w8}.

4.4.3 *One-Hot Encoding*

In section 4.1.3 it was shown that the output variable Y of the neural network can take on one of five values: 70, 90, 110, 130, or 150, which represent different gas pre-charge pressure levels. Although these categories, also called labels, are numerical values, they cannot be treated as such, because they are merely names or identifiers. Therefore, to make them compatible with the NN layers, it is necessary to convert these labels into a suitable format.

One common method is to convert them into boolean values, which can represent binary values (0 or 1) or probabilities (0-100%). Since the output layer of classification models utilizes the *SoftMax* as activation function, the One-Hot Encoder technique is particularly useful and required when dealing with multiple classes. [23]

The One-Hot Encoder transforms categorical labels into a vector representation. This approach involves creating a new dummy feature for each unique value in the categorical label column. By doing so, the neural network model can effectively learn and predict categorical outputs by treating them as separate binary classification problems. [23] For instance, the vector representation for each gas pre-charge pressure level is as follows:

$$70 = [1,0,0,0,0]$$
$$90 = [0,1,0,0,0]$$
$$110 = [0,0,1,0,0]$$
$$130 = [0,0,0,1,0]$$
$$150 = [0,0,0,0,1]$$

5 EXPERIMENTATION AND RESULTS

The current chapter presents the experimental approach and outcomes of the thesis. It provides a detailed description of the experimentation conducted and discusses the results obtained from training the neural network model described in section 4.2. Various scenarios and variations are explored to analyze the implications and significance of the findings.

5.1 Experiment Setup

As explained in section 4.4, the data preprocessing includes reshaping the input variables X using a sliding window and scaling the variables to a smaller range using the Robust Scaler. In addition, the true output Y is one-hot encoded in order to the SoftMax function to work. The experimentation involves training the NN with different input shapes, determined by the size of the sliding window (refer to table 8) and the number of input variables. Additionally, variations are made in the variables T_a and T_o, testing both fixed and varied temperatures.

5.1.1 Tests Description

The tests are conducted as follows:

Test 1: Evaluate the performance of the NN using minimal input signals, such as only using data from one sensor (e.g., p_o or T_{w8}).

Test 2: Utilize both p_o and T_{w8} as input variables to assess the impact of the input shape on network training. The input is represented as a three-dimensional matrix with shape consisting of (1) number of examples or number of sequences, (2) time step range or number of samples, and (3) number of input variables. For example, if input shape is equal $(540, 3906, 2)$, it means 540 sequences of 3906 timestamps, with 2 input variables (features). It is important no remind that 3906 samples are equal to $3906/12.5$ Hz $= 312.5$ seconds.

Test 3: The objective is to exclude the accumulator surface temperature, as it is not typically available in wind turbines. Instead, the sensors p_o, T_a, and T_o that are already present are utilized in this test.

Test 4: In an attempt to maximize performance, the input for this test includes both T_o and T_a temperatures along with p_o and T_{w8}.

Test 5: To further enhance the performance of the neural networks, additional wall temperature sensors are included. This means the system's p_o, T_{w1}, T_{w6}, T_{w8}, T_a, and T_o, are incorporated as input variables.

Test 6: Now some hyperparameters related to the FCN model are changed to match the configuration of the original paper. This involves changing quantity of filters to (128,256,128) instead of (60,60,60), and filter size to (8,5,3) instead of (3,3,3).

5.1.2 *Training Description*

The chosen loss function for all the tested models is the categorical cross entropy, which measures the average difference between the predicted output $\hat{y}$ and the ground truth Y across all classes. In addition, the categorical accuracy metric is used to evaluate the classification performance, which calculates the frequency with which the predicted output matches the ground truth.

The optimization algorithm used in training is "Adam". The models are trained for a limit of 500 epochs, however some models have converged earlier. The training is stopped utilizing an *EarlyStopping* function, which is built-in in TensorFlow/Keras. This function automatically stops training if the loss error remains unchanged for more than 20 epochs. Another function utilized is the *ReduceLROnPlateau* which reduces the learning rate by a factor of 0.5 when the validation loss is lower than 0.00001 for more than 10 epochs.

To address memory consumption issues and prevent model crashes, the batch size is adjusted to 32. This means that the input data is fed to the NN in batches of 32 sequences. The input sequences are shuffled before training. This introduces randomness in the input order, meaning the NN will be trained with time sequences of unspecified order. An illustration of the coding of these hyperparameters is shown in figure 21.

```
epochs = 500
batch_size = 32

callbacks = [
    keras.callbacks.ModelCheckpoint(
        MODEL_FILE, save_best_only=True, monitor="val_loss"
    ),
    keras.callbacks.ReduceLROnPlateau(
        monitor="val_loss", factor=0.5, patience=10, min_lr=0.00001
    ),
    keras.callbacks.EarlyStopping(monitor="val_loss", patience=20, verbose=2),
]
model.compile(
    optimizer="adam",
    loss="categorical_crossentropy",
    metrics=["categorical_accuracy"],
)
```

Figure 21: Illustration of the hyperparameters being setting up in the TensorFlow/Keras program.

5.1.3 *Analysis Limitations*

The robustness of the neural network model to extreme variations in system states, sensor noise, and outliers is not examined in this thesis. However, alternative approaches to testing these cases can be considered. One approach is to generate additional simulation data with specific variations in the input variables or to deploy the trained model in the same simulation environment as the accumulator model. In this latter case, it is important to note that the neural network requires a certain amount of time before it can start predicting the pre-charge pressure, which corresponds to the size of the sliding window as specified in table 8.

Each time a NN is trained from scratch, the resulting weights and biases will have different values. This stochastic nature of the training means that the performance of the model can vary without any specific pattern. Evaluating the performance of these NN algorithms ideally involves conducting statistical analysis and comparing measures such as the t-score. Several benchmarking methods for NN exist, as demonstrated in studies like [25, 26]. However, due to time constraints, a comprehensive statistical analysis was not conducted in this thesis. Instead, the average results of two to three training runs were considered, including metrics such as accuracy and loss error, which are presented in table 9.

5.2 Training Results

In order to validate the NN model performance, both the loss and accuracy metrics are analyzed for the validation and test sets. The validation set is used to monitor the training progress and determine when the model has reached its optimal performance. The test set, on the other hand, is used to evaluate the final performance of the trained model on unseen data. It provides a measure of how well the model generalizes to new and unseen examples.

During the training process, the validation loss and accuracy values monitored and recorded for each epoch. Plotting these values over time helps visualize the model's learning progress. It is also crucial to monitor the accuracy of the training dataset to detect overfitting, which occurs when there is a notable difference between the accuracy obtained on the training dataset and that on the validation/test dataset.

In the case of the test 5, which utilize a moving window of 3906 samples and 6 variables ($p_o, T_{w1}, T_{w6}, T_{w8}, T_a, T_o$) as input, it exhibits the best performance among the tested models. The training process of this model is depicted in figure 22, showcasing its accuracy over the training epochs.

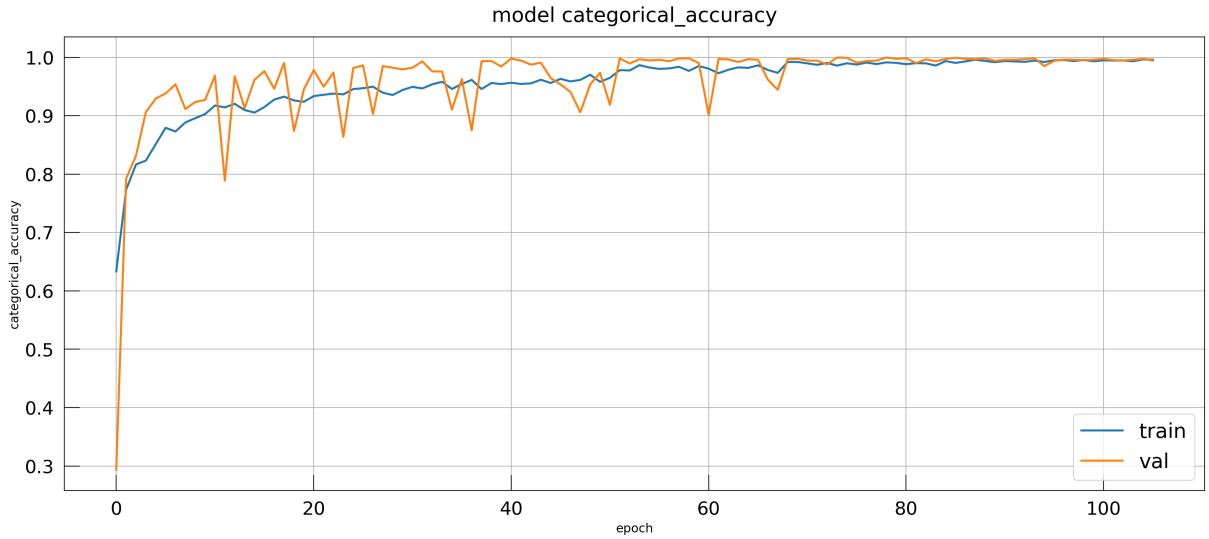


Figure 22: Accuracy of both training and validation datasets when utilizing 6 input variables and a moving window if 3906 samples.

Additionally, a confusion matrix, as in figure 23, can be generated to evaluate the model's performance in predicting each class. The confusion matrix shows the number of correct and incorrect predictions for each class, allowing for a deeper understanding of which classes the neural network is more likely to confuse or misclassify.

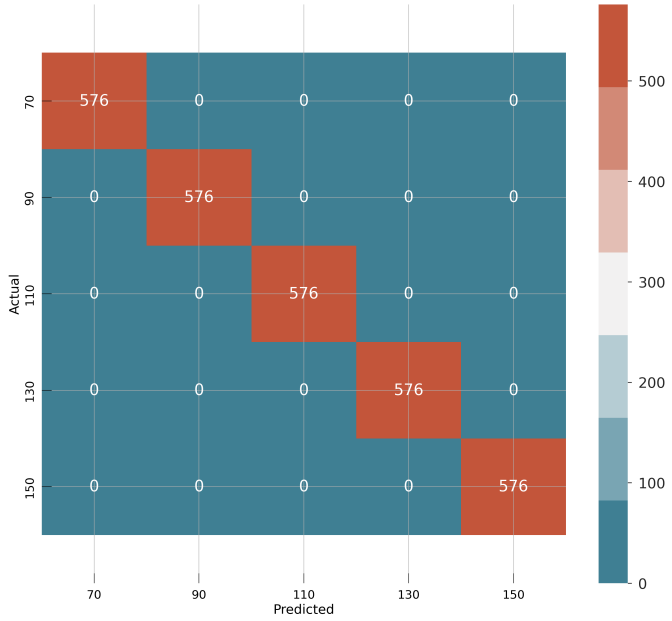


Figure 23: Confusion matrix when utilizing 6 input variables and a moving window if 3906 samples.

While the confusion matrix may not provide significant insights when the accuracy is at 100%, it becomes valuable when the accuracy is slightly lower. For instance, with an accuracy of 82% in the case of Test 2, which applies a sliding window of 15625 samples and 2 input variables (p_o, T_{w8}), the confusion matrix, figure 24, offers valuable information.

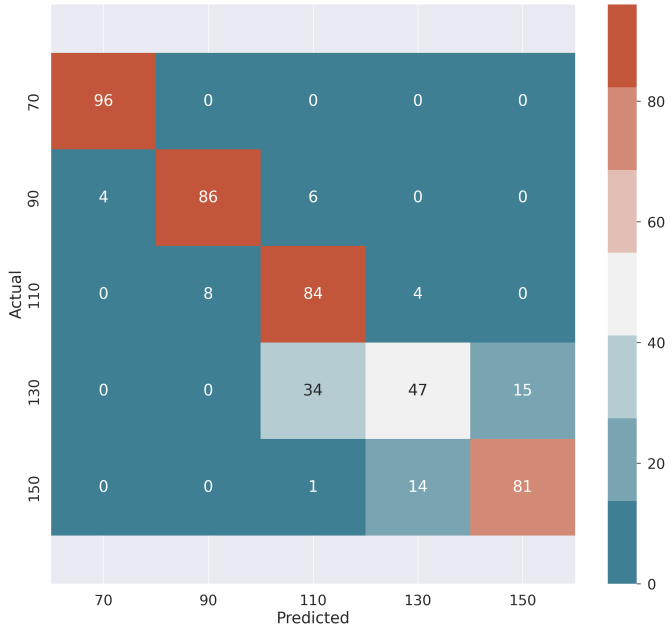


Figure 24: Confusion matrix when utilizing 2 input variables and a moving window if 15625 samples.

For the gas pre-charge pressure class "130 bar", the confusion matrix reveals that 47 examples were correctly predicted, while 34 were mistakenly classified as "110 bar" and 15 as "150 bar". This information can be utilized to reevaluate the data preprocessing or data generation approaches and focus on localized solutions to improve the model's performance.

5.3 Test Validation Results

The experiments, described in section 5.1.1, and their corresponding outcomes are presented in table 9. Additional analysis and interpretation are provided below.

Table 9: Performed tests and neural network performance.

Test Nº	Input Size	Input Variables	T_a,T_o	Test Accuracy	Test Loss	Epochs Used	Training Time
1	(540, 3906, 1)	p_o	Fixed	0.23	1.5871	51	1.8
1	(540,3906,1)	T_{w8}	Fixed	0.89	0.2748	70	1.3
1	(8640, 3906, 1)	p_o	Varied	**0.94**	**0.2200**	85	27.0
1	(8640, 3906, 1)	T_{w8}	Varied	0.50	1.2309	44	14.1
2	(1140,1953,2)	p_o, T_{w8}	Fixed	0.93	0.1904	204	3.5
2	(540,3906,2)	p_o, T_{w8}	Fixed	**0.91**	**0.2889**	64	1.2
2	(240,7812,2)	p_o, T_{w8}	Fixed	0.56	1.3676	25	0.5
2	(90,15625,2)	p_o, T_{w8}	Fixed	0.60	1.2356	42	0.7
2	(18240, 1953, 2)	p_o, T_{w8}	Varied	0.90	0.2391	69	24.2
2	(8640, 3906, 2)	p_o, T_{w8}	Varied	**0.96**	**0.1644**	98	31.1
2	(3840, 7812, 2)	p_o, T_{w8}	Varied	0.92	0.3037	100	28.4
2	(1440, 15625, 2)	p_o, T_{w8}	Varied	0.82	0.9149	43	8.9
3	(8640, 3906, 3)	p_o, T_a, T_o	Varied	**0.95**	**0.1634**	124	39.5
3	(1440, 15625, 3)	p_o, T_a, T_o	Varied	0.59	1.0448	53	11.4
4	(540,3906,4)	p_o, T_{w8}, T_a, T_o	Fixed	**0.91**	**0.2867**	60	1.1
4	(90,15625,4)	p_o, T_{w8}, T_a, T_o	Fixed	0.36	1.2768	42	42.3
4	(8640,3906,4)	p_o, T_{w8}, T_a, T_o	Varied	**0.96**	**0.1209**	101	33.8
4	(1440,15625,4)	p_o, T_{w8}, T_a, T_o	Varied	0.81	0.3719	121	25.7
5	(540,3906,6)	$p_o, T_{w1}, T_{w6}, T_{w8}, T_a, T_o$	Fixed	1.00	0.0008	83	1.8
5	(90, 15625, 6)	$p_o, T_{w1}, T_{w6}, T_{w8}, T_a, T_o$	Fixed	1.00	0.0014	474	8.0
5	(8640, 3906, 6)	$p_o, T_{w1}, T_{w6}, T_{w8}, T_a, T_o$	Varied	**1.00**	**0.0085**	106	35.0
5	(1440,15625,6)	$p_o, T_{w1}, T_{w6}, T_{w8}, T_a, T_o$	Varied	1.00	0.0167	187	40.6
6	(540, 3906, 4)	p_o, T_{w8}, T_a, T_o	Fixed	0.95	0.1490	260	17.5
6	(8640,3906,4)	p_o, T_{w8}, T_a, T_o	Varied	**0.98**	**0.0973**	60	70.4

[1] Input Shape: (# of sequences, # of samples, # of variables)

[2] Numbers in bold present more interesting results.

5.3.1 Sliding Window Size

The impact of the sliding window size on the performance of the neural network is evident from Test 2. Different window sizes were investigated, namely 1953, 3906, 7812, and 15625 samples, which correspond to 156, 312, 625, and 1250 seconds of simulation, respectively.

It was found that splitting the data with a window size of **3906** achieved the best performance. This window size strikes a balance between high accuracy and low training time. Smaller window sizes may yield higher accuracy, but they require more time to train, while larger window sizes result in poorer accuracy. This finding holds true for both fixed and varied values of T_a and T_o.

It is important to note that the hardware used for training may have influenced this result. Furthermore, with a shorter window width, there are more sequences available for training the data. This increased number of sequences can have an impact on the performance and learning capabilities of the neural network.

5.3.2 Minimum Sensor Utilization

According to Test 1, utilizing the oil pressure signal p_o with a sliding window of 3906 samples achieves an accuracy of 94% for predicting the gas pre-charge pressure. However, the loss function value of 0.2200 indicates that this NN model may not be very robust and may not generalize well to different data.

Test 3 shows that incorporating the oil and ambient temperature signals into the input of the neural network improves the accuracy to 95% and reduces the loss to 0.1634. This is a significant finding, as it confirms that no additional sensors need to be installed on the wind turbine. The FCN model can accurately predict the pre-charge pressure by considering only three available sensors.

In addition, the NN model performance can be further enhanced by including the temperature of the accumulator surface into the NN input. Test 4 that adding a single thermocouple, such as T_{w8}, leads to an accuracy increase to 96% and a decrease in loss to 0.1209. The performance continues to improve when three thermocouples (T_{w1}, T_{w6}, T_{w8}) are included, achieving a perfect accuracy of 100% with a loss of 0.0085. This is the lowest loss achieved for the dataset where T_a and T_o are varied.

5.3.3 Fixed vs Varied Oil and Ambient Temperature

Upon analyzing all the tests conducted for both fixed and varied oil and ambient temperature, two conclusions can be drawn. First, when assuming fixed temperature, there are fewer variations in the input data, making it easier for the NN to converge. Second, as a consequence of filtering the input data, there is fewer data available for the training. The results demonstrate varying performance across the tests, with no clear pattern emerging. The tests that performed better can be attributed to the presence of fewer data variation, while the tests that performed worse can be attributed to a lower amount of available data.

5.3.4 *Chosen Hyperparameters vs Original*

The final test performed, Test 6, used the original hyperparameters from Wang, Yan, and Oates [26], which consist of (128,256,128) filters of size (8,5,3). This test utilized a sliding window of 3906 samples and included four input variables (p_o, T_{w8}, T_a, T_o), which is the same input configuration as Test 4.

The original hyperparameters proved to be beneficial for the NN model, achieving higher accuracy of 98%, compared to 96% in Test 4, and lower loss of 0.0973, compared to 0.1209. However, it is worth noting that, as mentioned in section 4.2, using the original hyperparameters resulted in issues with memory consumption and crashes. Test 6 took 70 minutes to train, while Test 4 took 34 minutes. Although the alternative hyperparameters did not outperform the original ones, they were sufficient for the purposes of this study, considering the context and constraints.

6 CONCLUSION

In order to ensure the reliable operation of offshore wind turbines, a robust maintenance strategy is essential. In this thesis, pitch systems, specifically the role of accumulators in the wind turbine downtime is discussed. Gas leakage in accumulators presents a significant challenge, particularly due to limitations in sensor installation. To solve this issue, an approach utilizing neural networks for gas leakage detection is proposed.

The first step was to develop and validate an accurate accumulator model. In order to capture different operational scenarios, data was generated by varying the gas pre-charge pressure, load flow intensity, and oil and ambient temperature. Subsequently, a fully connected neural network (FCN) was developed, trained, and evaluated for gas leakage detection.

Accumulator Model Validation The validation process of the accumulator model provided valuable insights into their accuracy and performance. The comparison of piston position between the simulated and experimental data showed that the residual remained within an acceptable range of $\pm$ 5mm, indicating a reasonably good estimation of the piston position.

The evaluation of wall temperatures in different elements revealed some discrepancies, particularly in wall element 1, which can be attributed to the assumptions made regarding the distribution and constancy of oil temperature, as well as the assumption of the oil end cap having the same temperature as the oil. It is also possible that inaccuracies in the documentation of experimental tests may have contributed to these deviations. Overall, these findings highlight the importance of careful consideration of model assumptions and the need for accurate and well-documented experimental data for effective validation of accumulator system models.

Neural Network Performance The developed FCN model, inspired by previous studies, demonstrated good performance in the task of classifying gas pre-charge pressure. The choice of input variables and the size of the sliding window significantly impact the performance of the neural network model. It was found that a window size of 3906 samples achieved the best balance between accuracy and training time.

The choice of hyperparameters, such as the number of filters and their sizes, plays a crucial role in model performance. Fine-tuning these hyperparameters can lead to improved accuracy and reduced loss. However, it is important to consider the trade-off between performance and computational resources, as using larger filters and more filters can increase training time and memory consumption.

In terms of sensor utilization, incorporating the oil and ambient temperature signals into the input of the neural network improved the accuracy and reduced the loss. The model achieved 95% accuracy and a loss of 0.1634 by considering only three available sensors (oil pressure, oil temperature, and ambient temperature). Adding the temperature of the accumulator surface further enhanced the performance, reaching 100% accuracy and a loss of 0.0085 when three thermocouples were included.

Overall, the experiments demonstrate the potential of neural networks for predicting gas pre-charge pressure in wind turbines. The results highlight the importance of carefully selecting input variables, optimizing hyperparameters, and considering the specific characteristics of the data to achieve the best possible performance. Incorporating additional sensor data, such as oil and ambient temperature, improves the accuracy of the model. The neural network can effectively predict the gas pre-charge pressure using a limited number of sensors.

7 FUTURE WORK

There are several potential opportunities for further research and development in the field of gas leakage detection in accumulators using neural network-based method. The following areas can be explored:

Expanded Data Generation: In order to capture a wider range of operational scenarios, future work can involve generating model data using different pump and load flow, and different oil and ambient temperatures. Additionally, incorporating sensor noise and disturbances can further evaluate the robustness of the neural network and enhance its adaptability to real-world conditions.

Residual Neural Network (ResNet): ResNet architectures have shown excellent performance in various tasks such computer vision and TSC. Implementing a residual neural network (ResNet) for gas leakage detection could be beneficial, especially when the performance of the current fully connected network (FCN) using only one sensor is not great.

Wavelet Analysis and Pre-trained CNN: Wavelet analysis can be utilized to transform wavelet signals into images, which can then be fed into a pre-trained convolutional neural network (CNN), such as Inception. This approach can potentially improve the detection accuracy by utilizing the power of pre-trained models and their ability to extract meaningful features from complex data.

Class Activation Map (CAM): The integration of Class Activation Map (CAM) can provide valuable insights into the regions of interest and contribute to interpretability of the neural network model. By visualizing the important features and areas that contribute to the gas leakage detection decision, CAM can aid in understanding the underlying patterns and further improve the detection system.

BIBLIOGRAPHY

[1] Zhengru Ren et al. "Offshore wind turbine operations and maintenance: A state-of-the-art review". In: *Renewable and Sustainable Energy Reviews* 144 (2021), p. 110886. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2021.110886. URL: https://www.sciencedirect.com/science/article/pii/S1364032121001805.

[2] Magnus F. Asmussen, Jesper Liniger, and Henrik C. Pedersen. "Fault Detection and Diagnosis Methods for Fluid Power Pitch System Components—A Review". In: *Energies* 14.5 (2021). ISSN: 1996-1073. DOI: 10.3390/en14051305. URL: https://www.mdpi.com/1996-1073/14/5/1305.

[3] James Carroll, Alasdair McDonald, and David Mcmillan. "Failure rate, repair time and unscheduled O&M cost analysis of offshore wind turbines". In: *Wind Energy* 19 (Aug. 2015). DOI: 10.1002/we.1887.

[4] Julia Walgern et al. "Reliability of electrical and hydraulic pitch systems in wind turbines based on field-data analysis". In: *Energy Reports* 9 (2023), pp. 3273–3281. ISSN: 2352-4847. DOI: https://doi.org/10.1016/j.egyr.2023.02.007. URL: https://www.sciencedirect.com/science/article/pii/S235248472300149X.

[5] Jesper Liniger. "Design of Reliable Fluid Power Pitch Systems for Wind Turbines". English. PhD supervisor: Prof. Henrik C. Pedersen, Aalborg University Assistant PhD supervisor: Assoc. Prof. Mohsen Soltani, Aalborg University. PhD thesis. 2018. DOI: 10.5278/VBN.PHD.ENG.00040.

[6] Jesper Liniger et al. "Signal-Based Gas Leakage Detection for Fluid Power Accumulators in Wind Turbines". In: *Energies* 10.3 (2017). ISSN: 1996-1073. DOI: 10.3390/en10030331. URL: https://www.mdpi.com/1996-1073/10/3/331.

[7] *Hydropneumatic Accumulators and Components*. General Catalogue. Epe Italiana s.r.l. 2019. URL: https://www.epeitaliana.it/wp-content/uploads/2020/06/Catalogo-Generale-EPE-2019.pdf.

[8] Magnus F. Asmussen et al. "Pre-Charge Pressure Estimation of a Hydraulic Accumulator Using Surface Temperature Measurements". In: *Wind* 2.4 (2022), pp. 784–800. ISSN: 2674-032X. DOI: 10.3390/wind2040041. URL: https://www.mdpi.com/2674-032X/2/4/41.

[9] Kamilla Heimar Andersen, Simon Pommerencke Melgaard, and Daniel Leiria. *Working document: Summary of Existing FDD Frameworks for Building Systems*. English. Apr. 2023.

[10] Simon P. Melgaard et al. "Fault Detection and Diagnosis Encyclopedia for Building Systems: A Systematic Review". In: *Energies* 15.12 (2022). ISSN: 1996-1073. DOI: 10.3390/en15124366. URL: https://www.mdpi.com/1996-1073/15/12/4366.

[11] Marcin Mrugalski. "Introduction". In: *Advanced Neural Network-Based Computational Schemes for Robust Fault Diagnosis*. Cham: Springer International Publishing, 2014, pp. 1–7. ISBN: 978-3-319-01547-7. DOI: 10.1007/978-3-319-01547-7_1. URL: https://doi.org/10.1007/978-3-319-01547-7_1.

[12] Stefan Kollmannsberger et al. *Deep Learning in Computational Mechanics. An Introductory Course*. Cham: Springer International Publishing, 2021. ISBN: 978-3-030-76587-3. DOI: 10.1007/978-3-030-76587-3. URL: https://doi.org/10.1007/978-3-030-76587-3.

[13] Yao Jin et al. "Fault Diagnosis of Hydraulic Seal Wear and Internal Leakage Using Wavelets and Wavelet Neural Network". In: *IEEE Transactions on Instrumentation and Measurement* 68.4 (2019), pp. 1026–1034. DOI: 10.1109/TIM.2018.2863418.

[14] Jianjun Chen et al. "An Imbalance Fault Detection Algorithm for Variable-Speed Wind Turbines: A Deep Learning Approach". In: *Energies* 12.14 (2019). ISSN: 1996-1073. DOI: 10.3390/en12142764. URL: https://www.mdpi.com/1996-1073/12/14/2764.

[15] Alibek Kopbayev et al. "Gas leakage detection using spatial and temporal neural network model". In: *Process Safety and Environmental Protection* 160 (2022), pp. 968–975. ISSN: 0957-5820. DOI: https://doi.org/10.1016/j.psep.2022.03.002. URL: https://www.sciencedirect.com/science/article/pii/S0957582022001999.

[16] Georg Helbing and Matthias Ritter. "Deep Learning for fault detection in wind turbines". In: *Renewable and Sustainable Energy Reviews* 98 (2018), pp. 189–198. ISSN: 1364-0321. DOI: https://doi.org/10.1016/j.rser.2018.09.012. URL: https://www.sciencedirect.com/science/article/pii/S1364032118306610.

[17] Lasse Bonde Hansen et al. *Gas leakage detection for fluid power accumulator. Accumulator fault detection using neural networks*. Aalborg University, 8th Semester Report. 2022.

[18] Anders Hedegaard Hansen. *Fluid Power Systems. A Lecture Note in Modelling, Analysis and Control*. Springer International Publishing, 2023. ISBN: 978-3-031-15088-3. DOI: 10.1007/978-3-031-15089-0. URL: https://doi.org/10.1007/978-3-031-15089-0.

[19] D. R. Otis and A. Pourmovahed. "An Algorithm for Computing Nonflow Gas Processes in Gas Springs and Hydropneumatic Accumulators". In: *Journal of Dynamic Systems, Measurement, and Control* 107.1 (Mar. 1985), pp. 93–96. ISSN: 0022-0434. DOI: 10.1115/1.3140714. eprint: https://asmedigitalcollection.asme.org/dynamicsystems/article-pdf/107/1/93/5492491/93_1.pdf. URL: https://doi.org/10.1115/1.3140714.

[20] Siegfried Rotthäuser. "Verfahren zur Berechnung und Untersuchung hydropneumatischer Speicher". Aachen, Techn. Hochsch., Diss., 1993. PhD thesis. Aachen, 1993, VII, 133 S. : Ill., graph. Darst. URL: https://publications.rwth-aachen.de/record/75573.

[21] A. Pourmovahed and D. R. Otis. "An Experimental Thermal Time-Constant Correlation for Hydraulic Accumulators". In: *Journal of Dynamic Systems, Measurement, and Control* 112.1 (Mar. 1990), pp. 116–121. ISSN: 0022-0434. DOI: 10.1115/1.2894128. eprint: https://asmedigitalcollection.asme.org/dynamicsystems/article-pdf/112/1/116/5557217/116_1.pdf. URL: https://doi.org/10.1115/1.2894128.

[22] G. Dreyfus. *Neural Networks. Methodology and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. ISBN: 978-3-540-28847-3. DOI: 10.1007/3-540-28847-3. URL: https://doi.org/10.1007/3-540-28847-3.

[23] Hisham El-Amir and Mahmoud Hamdy. *Deep Learning Pipeline. Building a Deep Learning Model with TensorFlow*. Berkeley, CA: Apress, 2020. ISBN: 978-1-4842-5349-6. DOI: 10.1007/978-1-4842-5349-6. URL: https://doi.org/10.1007/978-1-4842-5349-6.

[24] Navid Mohammadi Foumani et al. *Deep Learning for Time Series Classification and Extrinsic Regression: A Current Survey*. 2023. arXiv: 2302.02515 [cs.LG]. URL: https://doi.org/10.48550/arXiv.2302.02515.

[25] Hassan Ismail Fawaz et al. "Deep learning for time series classification: a review". In: *Data Mining and Knowledge Discovery* 33 (2019), pp. 917–963. ISSN: 1573-756X. DOI: 10.1007/s10618-019-00619-1. URL: https://doi.org/10.1007/s10618-019-00619-1.

[26] Zhiguang Wang, Weizhong Yan, and Tim Oates. "Time series classification from scratch with deep neural networks: A strong baseline". In: *2017 International Joint Conference on Neural Networks (IJCNN)*. 2017, pp. 1578–1585. DOI: 10.1109/IJCNN.2017.7966039.

[27] Anthony Bagnall et al. *The UEA & UCR Time Series Classification Repository*. [Online; accessed 18-May-2023]. 2023. URL: https://www.timeseriesclassification.com.

[28] Tuan D Pham. "Time-frequency time-space LSTM for robust classification of physiological signals". In: *Scientific Reports* 11 (2021), pp. 2045–2322. DOI: 10.1038/s41598-021-86432-7. URL: https://doi.org/10.1038/s41598-021-86432-7.

[29] Kun Xia, Jianguang Huang, and Hanyu Wang. "LSTM-CNN Architecture for Human Activity Recognition". In: *IEEE Access* 8 (2020), pp. 56855–56866. DOI: 10.1109/ACCESS.2020.2982225.

[30] Tara N. Sainath et al. "Convolutional, Long Short-Term Memory, fully connected Deep Neural Networks". In: *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2015, pp. 4580–4584. DOI: 10.1109/ICASSP.2015.7178838.

[31] Christian Szegedy et al. *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*. 2016. arXiv: 1602.07261 [cs.CV].

[32] Christian Szegedy et al. *Going Deeper with Convolutions*. 2014. arXiv: 1409.4842 [cs.CV].

[33] Ovidiu Calin. *Deep Learning Architectures. A Mathematical Approach*. Cham: Springer International Publishing, 2020. ISBN: 978-3-030-36721-3. DOI: 10.1007/978-3-030-36721-3. URL: https://doi.org/10.1007/978-3-030-36721-3.

[34] François Chollet et al. *Keras*. https://keras.io. 2015.

[35] Tom O'Malley et al. *KerasTuner*. https://github.com/keras-team/keras-tuner. 2019.

[36] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. 2017. arXiv: 1412.6980 [cs.LG].

[37] HS Hota, Richa Handa, and Akhilesh Kumar Shrivas. "Time series data prediction using sliding window based RBF neural network". In: *International Journal of Computational Intelligence Research* 13.5 (2017), pp. 1145–1156.

[38] Pedro M. Ferreira and AntÓnio E. Ruano. "Online Sliding-Window Methods for Process Model Adaptation". In: *IEEE Transactions on Instrumentation and Measurement* 58.9 (2009), pp. 3012–3020. DOI: 10.1109/TIM.2009.2016818.

[39] Hanan A. Saeed et al. "Online fault monitoring based on deep neural network & sliding window technique". In: *Progress in Nuclear Energy* 121 (2020), p. 103236. ISSN: 0149-1970. DOI: https://doi.org/10.1016/j.pnucene.2019.103236. URL: https://www.sciencedirect.com/science/article/pii/S0149197019303427.

[40] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

A NEURAL NETWORK REVIEW

In order to support the methods presented in section 4, some used frequently terms in the field of machine learning are defined below. The information contained in this section is derived from Kollmannsberger et al. [12], Dreyfus [22], El-Amir and Hamdy [23], and Calin [33].

A.1 Basic Definitions

Machine Learning Machine learning is a subset of artificial intelligence that uses algorithms and statistical models to learn and improve from experience, without being explicitly programmed. One important machine learning method is the so-called *Neural Network*, that is going to be properly defined below. The term deep learning refers to a neural network which are composed of several interconnected layers and involves the use of large datasets and complex models.

Neuron A neuron is a nonlinear function that is parameterized and bounded. The input $\boldsymbol{X}$ is called "variables" and the output value $\boldsymbol{y}$ is often referred as "prediction". A typical neuron is presented in figure 25.

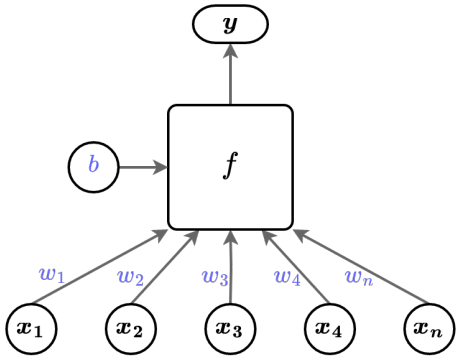


Figure 25: A neuron representation.

Neural Networks Neurons can be combined into complex networks, often built in layers. The input layer, although called layer, is not a neuron, since it is not a function. Hidden layers are nested abstract functions of the inputs. The output layer provides the final computation. Neural networks can be divided into two types: feedforward networks and recurrent networks.

Feedforward Neural Networks The information goes from inputs to outputs. There is no cyclic path. Some examples are: Multilayer Perceptrons (MLP), Convolutional Neural Network (CNN) and Autoencoder.

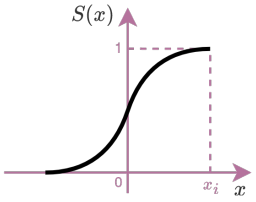
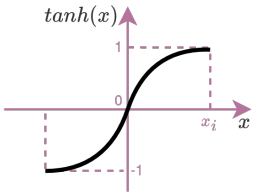
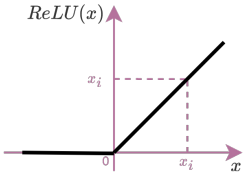
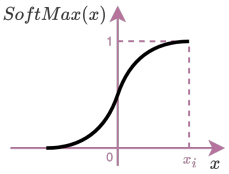
Recurrent Neural Networks Likewise known as Feedback Neural Network. They contain cyclic paths leading back to a starting neuron. However, this causes a neuron to be function of itself, which is not possible, so in this case, time is explicitly taken into account, as a neuron can be function of its previous values. This neural network therefore consists of nonlinear discrete-time recurrent equations. Examples are: Long-short Term Memory (LSTM) and Gated Recurrent Unit (GRU).

Activation Function The neuron function f is called activation function. There are different types of activation functions that are used depending on the problem being solved. Equations and graph representation of these functions is presented in table 10.

To introduce non-linearity in the neural networks, generally a sigmoid (or logistic) function, a hyperbolic tangent (tanh) function, or a Rectified Linear Unit (ReLU) function are used in hidden layers. They are useful for logistic regression or classification algorithms, allowing the modeling of complex relationships between inputs and outputs.

The SoftMax function is often used in output layers for multi-class classification. It is utilized due to three key properties it possesses. Firstly, it ensures that the sum of probabilities is always equal to 1. Secondly, the function is differentiable, which is beneficial for gradient-based optimization algorithms. Lastly, it is an extension of logistic regression to handle multinomial cases. [25]

Table 10: Examples of activation functions.

Activation Function	Graph Representation
$S(x) = \frac{1}{1+e^{-x}}$	
$tanh(x) = \frac{2}{1+e^{-2x}} - 1$	
$ReLU(x) = max(0, x)$	
$SoftMax(x) = \frac{e^{x_i}}{\sum e^{x_i}}$	

Supervised Training The model is provided with both known input and the ground truth output. The goal is to adjust the weights and biases so that the model can make accurate predictions. Classification and regression problems are examples that utilize supervised learning.

Unsupervised Training The model is provided with an unlabeled dataset. The goal is to finds hidden patterns in the input data without any prior knowledge of the output. Clustering and dimensionality reduction are examples that utilize unsupervised learning.

Training Example A training example is a single input-output pair that is used to train a machine learning model. It consists of an input, which is typically a vector of feature values that describe some object or phenomenon, and an output, which is the label or value associated with that object or phenomenon.

A.2 Hyperparameter Definitions

Hyperparameters are parameters whose values are set before training the model and cannot be learned from the data. They have a significant impact on the performance of the model and their selection requires experimentation. The selection of these hyperparameters is often done using techniques such as grid search, random search, or Bayesian optimization. The most common discussed hyperparameters are presented below.

Learning Rate Determines the step size taken during the optimization process of a machine learning model. It determines the rate at which the weights of the model are updated with respect to the loss gradient. A large learning rate can result in unstable training, while a small learning rate can result in very slow convergence.

Batch Size Refers to the number of training examples utilized in one forward/backward pass. The model will process the whole batch of training examples at once before updating the weights. Larger batch sizes can lead to faster convergence but can also require more memory resources. There are three types:

1. In *full batch* the entire training dataset is fed to the model in one iteration. This means that the batch size is equal to the number of examples in the dataset.
2. For *mini-batch* a small subset of the training dataset is used in each iteration. The batch size is typically chosen to be a power of 2, and is less than the total number of examples in the dataset.
3. At last, the *stochastic batch* uses a single training example in each iteration. This means that the batch size is equal to 1.

Epoch One epoch means one iteration over the training dataset. It is the complete pass through, followed by the parameters update. Generally, the more epochs, the more times the model will learn and update its parameters, but this can also lead to overfitting if not carefully controlled.

B NEURAL NETWORK ARCHITECTURES

As a way of providing a reference knowledge for this thesis, the most basic and common neural network architectures are presented in this appendix. The information is derived from Kollmannsberger et al. [12], Dreyfus [22], El-Amir and Hamdy [23], and Calin [33].

B.1 Convolutional Neural Network

CNN is a specialized type of neural network architecture commonly used for analyzing grid-like or spatial data, such as images or time-series data with spatial structure. CNNs are particularly effective at extracting meaningful features from input data through the use of convolutional layers and pooling layers. A typical architecture for CNN is illustrated in figure 26. [23]

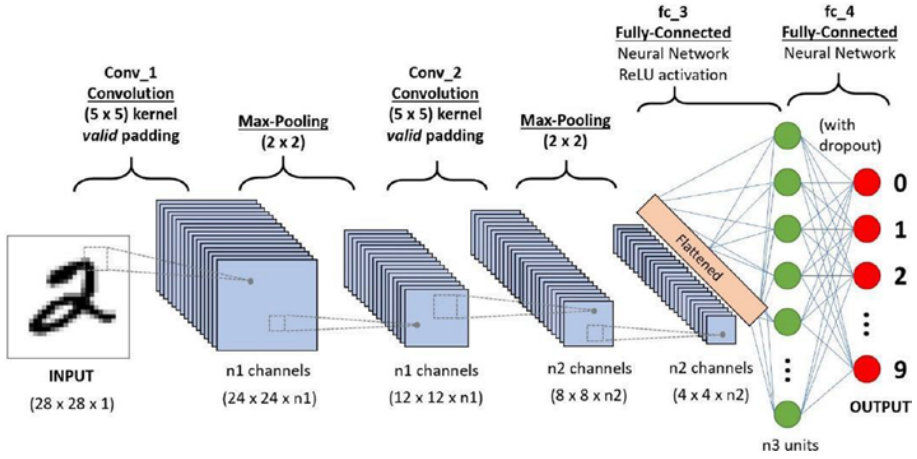


Figure 26: Simple CNN architecture utilized for image classification [23].

Convolutional Layers: The convolutional layers in a CNN apply a set of learnable filters (kernels) to the input data, allowing the network to capture local patterns or features present in the data. Each filter performs a convolution operation, which involves sliding the filter over the input data and computing element-wise multiplications and summations to produce feature maps.

Pooling Layers: Pooling layers are used to downsample the feature maps generated by convolutional layers, reducing their spatial dimensions while retaining important features. Common pooling operations include max pooling (selecting the maximum value within a region) and average pooling (taking the average value within a region).

Activation Functions: CNNs employ activation functions, such as the rectified linear unit (ReLU), to introduce non-linearity and enable the network to model complex relationships in the data.

Fully Connected Layers: After the convolutional and pooling layers, CNNs typically include one or more fully connected layers. Each neuron in one layer is connected to every neuron in the previous and the next layer. These layers perform high-level feature extraction and transformation, allowing the network to learn more abstract representations of the input data.

The combination of convolutional and pooling layers in CNNs enables them to automatically learn hierarchical representations of data, capturing both low-level and high-level features. This makes CNNs highly effective in tasks like image classification, object detection, and analyzing time-series data with spatial characteristics.