DETERMINATION OF REMAINING USEFUL LIFE AND FAULT DIAGNOSTICS OF SOLENOID VALVE IN OXYGEN INJECTION SYSTEM

A PROJECT REPORT

Submitted by

SHIVANI R (RA2011026010060)

SINDHU KALEESWARAN (RA2011026010082)

VAISHALI V (RA2011026010100)

CHEREDDY SOWMYA SRI (RA2011026010113)

Under the guidance of

Dr. A. Jackulin Mahariba

(Assistant Professor, Department of Computational Intelligence)

in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

w/s in

ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING

of

FACULTY OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTATIONAL INTELLIGENCE

COLLEGE OF ENGINEERING AND TECHNOLOGY

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

KATTANKULATHUR- 603 203

MAY 2024

Department of Computational Intelligence
SRM Institute of Science & Technology

Own Work* Declaration Form

This sheet must be filled in (each box ticked to show that the condition has been met). It must be signed and dated along with your student registration number and included with all assignments you submit – work will not be marked unless this is done.

To be completed by the student for all assessments

Degree/ Course : **B. Tech. CSE AIML**

Student Name : **Vaishali V, Sindhu K , Sowmya C, Shivani R**

Registration Number : **RA2011026010100, RA2011026010082 , RA2011026010113 , RA201102601060**

Title of Work : **Determination of Remaining Useful Life and Fault Diagnostics of Solenoid Valve in Oxygen Injection System**

We hereby certify that this assessment compiles with the University's Rules and Regulations relating to Academic misconduct and plagiarism**, as listed in the University Website, Regulations, and the Education Committee guidelines.

We confirm that all the work contained in this assessment is our own except where indicated, and that We have met the following conditions:

- Clearly referenced / listed all sources as appropriate
- Referenced and put in inverted commas all quoted text (from books, web, etc)
- Given the sources of all pictures, data etc. that are not my own
- Not made any use of the report(s) or essay(s) of any other student(s) either past or present
- Acknowledged in appropriate places any help that I have received from others (e.g. fellow students, technicians, statisticians, external sources)
- Compiled with any other plagiarism criteria specified in the Course handbook / University website

We understand that any false claim for this work will be penalized in accordance with the University policies and regulations.

DECLARATION:
We are aware of and understand the University's policy on Academic misconduct and plagiarism and we certify that this assessment is our own work, except where indicated by referring, and that we have followed the good academic practices noted above.

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY
KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that 18CSP109L project report titled **"Determination of Remaining Useful Life and Fault Diagnostics of Solenoid Valve in Oxygen Injection System "** is the bonafide work of " **Shivani R (RA2011026010060), SindhuKaleeswaran (RA2011026010082) , Vaishali V (RA2011026010100)** & **Chereddy Sowmya Sri (RA2011026010113)**" who carried out the project work under my supervision. Certified further, that to the best of my knowledge the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Signature of the GUIDE
Dr. A.Jackulin Mahariba
Assistant Professor
Dept. of Computational Intelligence
SRM IST

Signature of the HOD
Dr. R.Annie Uthra
Head of the Department
Dept. of Computational Intelligence
SRM IST

EXAMINER I

EXAMINER II

ACKNOWLEDGEMENT

First and foremost, we wish to express our sincere gratitude and grateful acknowledgement to the Management of **SRM INSTITUTE OF SCIENCE AND TECHNOLOGY,** for providing us the opportunity to do this project. We express our heartfelt and sincere thanks to respected **Dr. C. Muthamizhchelvan, Vice Chancellor and Prof.T.V.Copal, Dean (CET)**, for their encouragement towards the growth of research work and providing vital resources in the university.

We take this opportunity to express sincere thanks and deep sense of gratitude to beloved **Dr. Annie Uthra, Professor and Head of the Department & Dr. Karthik S, Academic Advisor &Associate Professor** for their encouragement as well as inputs towards the completion of the project.

We would like to thank our project guide **Dr. A. Jackulin Mahariba**, **Assistant Professor** for her support, knowledge, and valuable inputs throughout our project. We wish to place our heartfelt thanks to our project evaluators **Dr. R. Annie Uthra, Professor and Head of the Department, Dr. AK. Reshmy, Assistant Professor, Dr. AL Amutha , Assistant Professor , Dr. JV Vidhya , Assistant Professor and Mrs K . Suganya Varshini , Assistant Professor(Jr)** who have groomed our project through their technical suggestions and feedback. We are deeply grateful for their help, valuable guidance, suggestions and support.

Finally, we express our very profound gratitude to our parents and friends , for providing us with unfailing support and continuous encouragement throughout the process of researching and writing this thesis. This accomplishment would not have been possible without them.

(CHEREDDY SOWMYA SRI)

(VAISHALI V)

(SHIVANI R)

(SINDHU KALEESWARAN)

ABSTRACT

The life support system (LSS) within a submarine ensures crew safety and sustenance during underwater missions. The Oxygen Injection System (OIS) within the LSS is crucial for supplying breathable air to the crew through various oxygen valves. Our model consists of two main components: firstly, a cohesive fault diagnostic for the oxygen valves, using a multimeter and pressure gauge to diagnose and monitor the functioning of two-way internally piloted solenoid valves in the OIS. This is implemented through MATLAB Simulink &Python for various time intervals in the operational range. Secondly, the estimation of the Remaining Useful Life (RUL) of the OIS is achieved through a Bayesian Convolutional Neural Network (BCNN) model & Kullback Leibler (KL) Divergence by using the data obtained after fault analysis. This comprehensive approach considers various ambient parameters & human parameters to estimate the RUL in hours & mins. Therefore, the OIS and LSS are integrated which then ensures smooth functioning of the submarine that allows detection of faults in a timely fashion, avoid leakage in O_2 valves and provides time to time data on the RUL to the crew members.

Keywords- Bayesian Convolutional Neural Network , Fault Identification , Kullback Leibler, Life support system, , Oxygen injection system, Remaining useful life

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

NOTATION	ABBREVIATION
BCNN	**Bayesian Convolutional Neural Network**
CNN	**Convolutional Neural Network**
CUI	**Controller User Interface**
ELBO	**Evidence Lower Bound**
EOL	**End of Life**
KL	**Kullback Leibler**
LSS	**Life Support System**
MAE	**Mean Absolute Error**
MC Dropout	**Monte Carlo Dropout**
MCMC	**Markov Chain Monte Carlo**
ML	**Machine Learning**
MSE	**Mean Square Error**
NC	**Normally Closed**
NO	**Normally Open**
OIS	**Oxygen Injection System**
ORM	**Oxygen Redundancy Management**
PID	**Proportional Integral Derivative**
Py	**Python**
ReLu	**Rectified Linear Unit**
RUL	**Remaining Useful Life**
SV	**Solenoid Valves**
TL	**Transfer Learning**
VI	**Variational Inference**

CHAPTER 1

INTRODUCTION

1.1 GENERAL

The life support system of the submarine is a complicated network designed to provide its crew with an acceptable living environment while they are underwater. It includes systems to maintain the flow of air, regulate temperature, pressure and humidity. The Oxygen injection system in LSS is an important component which provides breathable oxygen to crew members throughout the mission. The oxygen valves must be without fault and in working conditions for the entire period of time to ensure this.

This project has two modules: Firstly, fault identification and diagnosis of the solenoid valves in OIS where the working of a two way internally piloted solenoid valve is checked to provide the right amount of oxygen to the crew. The solenoid valve works on the principle of electromagnetism and hence the opening & closing of these valves are set to be checked using a multimeter which detects the voltage/current supply provided to the valve. Also, a pressure gauge is introduced on both sides of the valves to detect the in/out flow of pressure which identifies the leakage in oxygen valves. Thus, the diagnosis of solenoid valves is essential for proper supply of oxygen.

Secondly, remaining useful life (RUL) is the length of time a system is expected to function/last before it needs to be replaced. It refers to the remaining operational time of the system before it reaches a predefined failure threshold, in addition to the operational duration of the system before it becomes non-functional or requires maintenance. These signal helps to activate the twin LSS that increases the reliability of the mission. The RUL of LSS is calculated through the various parameters such as Temperature, Humidity, Absolute pressure, O_2/CO_2concentration, body temperature, SPO_2, Blood Pressure and Heart rate. The result is calculated in hours by using different mathematical equations and Bayesian Convolutional Neural Network algorithm. This approach helps in timely interventions in a mission, preventive maintenance actions and efficient resource allocation by ensuring the OIS operates at its optimal performance level.

Overall, fault identification and diagnosis techniques & RUL estimation directly contribute to safeguarding lives, enhancing operational efficiency, and survival of mankind inside the submarine.

1.2 MOTIVATION

The primary motivation is on ensuring the safety and well-being of the submarine crew by maintaining a reliable oxygen injection system for breathable air. Detecting and addressing any issues with the system promptly is crucial to prevent disruptions that could endanger the crew. Additionally, minimizing failures in the oxygen system is essential to avoid hazards like increased cabin pressure and hypoxia among crew members, which could threaten both individuals and submarine operations. Swift detection and diagnosis enable proactive measures to reduce the risk of system failure. Moreover, understanding the remaining useful life of resources is vital for optimizing resource usage, minimizing downtime, and improving productivity and efficiency during underwater missions. The motivation is explained below:

I. ***Safety and survival of the crew:*** The primary concern is the safety and survival of the submarine crew, which is ensured by the oxygen injection system that provides breathable air throughout the mission. To avoid interruptions in oxygen supply which might jeopardize crew well-being or safety, it is essential that time faults are detected and identified as soon as possible.

II. ***Minimizing critical failures:*** Oxygen injection system failures could lead to critical oxygen valve failure, resulting in increased cabin pressure, increasing carbon dioxide concentrations and members of the crew becoming hypoxic. This would be not only a threat to the crew members, but also for submarine operations. Rapid detection and diagnosis allow preventive measures to be taken in order to mitigate the risk of system failure.

III. ***Estimation of Remaining Useful Life:*** It is of great importance for the cabin members to be aware of the remaining useful life during their underwater mission to optimize their resource utilization, reduce their downtime by enhancing their productivity and mission efficiency.

1.3 LITERATURE SURVEY

Haifeng Guo et al.[1] argued that the detection of fault in solenoid valves would be based on vibration signal measurement by way of Labview SignalExpress system where a vibration sensor was placed at the end of an solenoid valve and vibrations were collected therefrom. The degradation of valve field is detected by the changes in the vibrational signal waveform and his model enhances the ability to resist the interference and provides

information on valve functioning with condition-based maintenance operation,

Wenhao Guo et.al [2] suggested a fault diagnosis method in solenoid valves by building Multi-Kernel Support Vector Machine model with multiple kernel learning weights in which the current signal of the six modes of solenoid valve is calculated & its characteristics is extracted through the Empirical mode decomposition giving an accuracy rate of 98.9%. His findings suggested that SVM has advantages over a tree-based classification since it improves the classification accuracy.

Soo-Ho Jo et.al [3] described the fault detection for coil burnout in solenoid valves using the dynamic thermal loading mechanism. His work was directed towards the discovery of an equivalent current model based on Kirchhoff's voltage law and building a predictive regression model to find temperature relationship with electricity.

Utah Michael Ngbede et.al [4] claimed that detection of the solenoid valve fault condition can be done using artificial intelligence methods to process the coil's signature. In order to extract the coil signature characteristics in form of wavelets, which were aimed at optimizing valve predictive maintenance, a convolutional neural network was used.

V. A. Tsachouridis et.al [5] study examines the operation of a pneumatic diaphragm driven by two PWM solenoid valves, starting with an examination of their characteristics and leading to development of simplified mathematical models. A PWM control law is then formulated, accounting for valve effects and the asymmetrical flow characteristics during charging and discharging. The study focuses on optimization of key parameters such as frequency response and stiffen factors, which demonstrate improvements in performance.

Stoyan Stoyanov et. al [6] suggested that Vacuum solenoid valves are important for regulating processes and operating various throttle valves. These valves are managed by electronic control units (ECUs) using constant frequency and pulse width modulation signals, but their performance can change, influencing actuator control and, as a result, the engine's overall operation and environmental performance. He stated that the lack of feedback mechanisms made diagnosing difficult. His works concluded that understanding these dependencies is essential for reducing environmental impact by ensuring the proper functioning of vacuum solenoid valves.

Said Amrane et .al [7] stated that to monitor the coil resistance of electromagnetic solenoid valves, supervised machine learning has been proposed. The EMS valve shall be coupled to a photonic fiber squeezer, which serves as the force sensor. The Simulink model helps to define the initial response of an Open Loop system and achieve its transfer function, according to EMS parameters like coil resistance.

Tran Dinh Son et. al[8] In order to control pneumatic actuators, this paper summarizes the use of on/off of solenoid valves in comparison with Pneumatic Proportional and Service Valves. The study re-evaluates a controller introduced by Truong in 2020, which utilized four on-off solenoid valves. Seven different operating modes of the four on-off solenoid valves, combined with Pulse Width Modulation (PWM), are explored in the research. Works showed that the modified controller is working with frequencies less than or equal to 0.1 Hz.

Yu et.al [9] introduced an optimization method for the structural parameters of the proportional solenoid. He validated their 3D finite element method simulation model by conducting a force-displacement characteristic experiment on the proportional solenoid. The study parameters, such as displacement, armature length & arc radius the experimental results showed an increase of 20.1% in the average electromagnetic field.

He LI et.al[10] investigated the behavior of solenoid valves used for ship engine rooms with a focus on current changes during valve opening and closing. The current data collection shall be carried out with a special CAN node specifically configured for the management of pneumatic valves. To determine the true state of a solenoid valve and identify common faults, an algorithm with low computational complexity shall be executed on that node's MCU. In his tests, the solenoid valve control node developed has proved to be precise and fast in detecting faults.

Seungjin Yoo et.al [11] studies the Failures of hydraulic solenoid valves, which can result in system outages and safety hazards. A data-driven approach that employs voltage and current signals from both normal and damaged valve samples was used to diagnose these issues. To identify fault types, the system employs hypersphere-based clustering and a convolutional autoencoder to classify the data into discrete groups, such as normal and defective. Even with white noise included, the model was able to classify measurement data with 98% accuracy rate across seven different fault categories.

Jianfeng Li et.al[12] model helps explain solenoid valve failure mechanisms using temperature and stress distribution. His findings suggest a strong correlation between coil thermal expansion and solenoid valve malfunctions. The melting of insulating layers due to thermal expansion at high temperatures can result in coil shorting, reduced resistance, and eventually valve failure. According to his experiment, solenoid valve failure is a progressive process characterized by changes in coil resistance and temperature and these predictions were validated.

Boseong Seo et.al[13] In order to detect solenoid valve faults, it was suggested to analytically model the sensor signals, including electrical current, input and output pressure.

In order to quantify the state of solenoid valves with various failure modes, a health index was used.

Jerónimo Andrés Auzmendi et.al[14] Research suggests a simple way of detecting the movement of valves through solenoids by means of sensors which can be generated from plungers inside them. The plunger's signal makes it possible to determine with a minimal systematic error of under 2 mS when the diaphragm or pinch solenoid valves open and close. The error rate of the experiments to trials is reduced to a minimum of 0.2 ms when this Systematic Error has been rectified. It makes it easier to monitor solenoid valve operation with this method.

Torsten Brune et.al [15] paper focuses on the detection of switching points in solenoid valves through a signal-based approach that exploits the derivative of measured quantities and predefined thresholds. The derivative is calculated using Euler's initial order approximation and the signals are prefiltered via an analog filter. Training of the pressure and switch points relationship is based on a neural network. The study also explores the potential use of the valve as an intelligent pressure sensor, indicating its versatility and potential applications.

Kyoungkwan Ahn et.al[16] suggested that by introducing a new modified pulse-width modulation (MPWM) valve pulsing algorithm, costly servo valves can be replaced with inexpensive on/off solenoid valves. He implemented a continuous controller including position, velocity, and acceleration feedback. In addition, the pneumatic actuator's switching algorithm uses a learning vector quantization neural network, LVQNN, to classify external loads. The effectiveness of this algorithm to control different external loads with a smooth switching has been demonstrated in experiments, indicating its applicability and flexibility.

Tom Lefebvre et.al [17] introduced a new prognostic tool for mechatronic systems condition monitoring, emphasizing the enhancement of prophetic delicacy for criteria like Remaining Useful Continuance(RUL) and State of Health(SOH). Employing Hidden Markov Models(HMMs), the approach identifies declination dynamics from completely instrumented dimension sets gathered in exploration settings and applies virtual seeing to estimate real- time amounts from limited functional measures. The optimal idle state dimension aligns with the number of features in utmost cases. The paper envisions unborn work expanding into soothsaying operations.

Baoping Cai et.al [18] details the design and manufacturing of a subsea solenoid stopcock prototype for subsea blowout preventers which employs deterministic and

probabilistic thermal waves. It also electromagnetic signals through ANSYS software. The impact of uncertainties on material properties, physical characteristics and applied voltage is investigated in this research. The results highlight the importance of probabilistic analysis to optimize subsea solenoid stopcock design for enhanced reliability in blowback preventers, demonstrating that specific design parameters have a significant influential impact on the maximum level of temperature and electromagnetic power that is exerted.

Ganjour Mazaev et.al [19] study and analysis method introduced a methodology utilizing Bayesian Convolutional Neural Networks (BCNNs) for direct prediction of Remaining Useful Life (RUL) in solenoid valves (SV) based on current signatures. The proposed BCNN approach significantly enhances predictive accuracy, reducing RUL MeanAbsolute Error by approximately 40% compared to previous methods involving Deep Learning and feature-based techniques. In order to achieve further improvements, it is possible to incorporate an important physical feature of the electromechanical model into a hybrid design resulting in a 20 % lower RUL MAEs. The BCNN architecture provides well-calibrated uncertainty estimations, crucial for reliable prognostic decision-making. In order to demonstrate the superiority of hybrid models for early RUL predictions, insights into improved performance can be obtained by means of occlusion, a feature correlation method. The Bayesian nature of the BCNN, utilizing Mean-Field Variational Inference architecture, ensures well-calibrated predictive uncertainty, facilitating trustworthy prognostic decisions.

Yakup Genc et.al [20] explains for data-driven remaining usable life (RUL) prediction, as an end-to-end combined autoencoder-regressor architecture that combines long-short term memory networks and convolutional neural networks. Using the C-MAPSS and PHME20 datasets, the model—which was optimized using genetic algorithms—shows competitive performance in RUL prediction. Furthermore, a new defect detection-based RUL labelling technique is presented that exhibits performance that is either on par with or better than linear labelling. According to the study, non-linear adjustments can boost model performance by up to 50% even when there is just one flaw in the sensor data. Improving architectural stability and fault detection-based RUL labelling will be the primary objective of future research.

Jinsong Yang et.al [21] proposed to overcome the problems of external uncertainties and real-time monitoring, this research proposes a novel approach for estimating the remaining usable life (RUL) of rolling bearings. An enhanced dropout technique utilizing nonparametric kernel density and a fusion measure for runtime are presented in this approach, which is based on long-short term memory (LSTM) with uncertainty

characterization. Probabilistic distribution and precise RUL point estimation are demonstrated via validation of the dataset. The method's practical significance is indicated by its robust performance, particularly during stable deterioration. Moreover, by assessing and choosing efficient degrading features, a novel evaluation index exhibits higher performance over random-feature-trained LSTM models in terms of accuracy and stability.

Tarek Berghout et.al [22] predicts the Remaining Useful Life (RUL) of critical systems, in particular, this study highlights the importance of Prognosis and Health Management (PHM) in real-time monitoring and maintenance planning. In addition to offering a methodical approach for choosing and building ML models based on run-to-failure data, it analyses contemporary Machine Learning (ML) technologies for RUL prediction. The research highlights potential avenues for improving data quality and RUL model reconstruction, presents limitations and findings on the uses of ML models, and highlights chances for further development.

Daniel Azevedo et.al [23] predicts that the Prognostics and Health Management (PHM) system may now be simulated online thanks to a tool that this study presents. This tool allows researchers to experiment with different machine learning scenarios to estimate the RUL (Remaining Useful Life) of aviation systems. Dataset size, RUL prediction techniques, and outcome metrics can all be configured via the user-friendly interface. At present, it offers three cutting-edge machine learning approaches; in the future, it might be improved with more techniques and datasets for wider applicability, as well as hyperparameters for more comprehensive model customization. With its ability to conduct tests remotely at no expense, the tool is a useful tool for testing and investigating PHM systems.

Zhao-Hua Liu et.al [24] accurately predicted the residual life of rolling bearings, this study presents the E-LSTM approach, which combines elastic net with LSTM. Higher stability with better RUL forecasting performance are achieved by the E-LSTM algorithm, which tackles the overfitting problem by the addition of an elastic net regularization phenomenon to the LSTM structure. Subsequent investigations are intended to improve rolling bearing RUL prediction using E-LSTM in terms of overall efficiency and computational speed.

Fuqiang Sun [25] et.al found in contrast to the radial basis function neural network method, this research presents a Bayesian least-squares support vector machine (LS-SVM) method that shows superior accuracy and reliability for predicting the remaining useful life (RUL) of a microwave component. Point and interval estimates of RUL are provided, with a focus on the Bayesian LS-SVM framework. To improve accuracy and adaptability in

diverse scenarios, future research endeavours should focus on refining model parameters and investigating appropriate kernel functions for LS-SVM modelling.

James Carroll et.al [26] represents the artificial neural networks that are found to be the most accurate method for predicting gearbox failure in wind turbines through machine learning. Up to one month's notice of impending failure can be predicted using SCADA data; five to six months' notice can be obtained using vibration data. Vibration data yields 100% accuracy, while SCADA yields 72.5% to 75% accuracy for two-class neural networks. Multiclass neural networks are promising, particularly when used with vibration data. For more precise forecasts, future work might concentrate on improving algorithms and adding more SCADA inputs.

Yuhuang Zheng et.al [27] framed about a bearing remaining useful life (RUL) prediction technique for tracking bearing degradation is presented in this study. The method makes use of a unique health indicator that is taken from horizontal vibration signals using Hilbert-Huang entropy. RUL is predicted by a linear degradation model with bootstrapped sample thresholds. Effective degradation monitoring and RUL prediction are demonstrated by the experimental results. In order to address anomalies in degradation trends and achieve more accurate RUL prediction and robust wear-out period detection, future research will test the approach on more experimental datasets.

Joseph McGhee et.al [28] paper describes the use of artificial neural nets for modelling pneumatic actuators, which are used in actuator fault detection strategies. Typical failure rates data for the control elements of a process in both pure and dirty service have been incorporated into an overview classification of faults involving Process Valves and Actuators. The fundamental introduction to fault detection and isolation clearly indicates that the system of actuator fault detection is an important element in this philosophy, together with a system for detecting process or instrument faults. The experiments to date demonstrate the usefulness of ANNs as an accurate measurement tool for torque measurements from pneumatic actuators. The aim is to reduce the amount of duplicated information in a number of inputs, which allows for the reduction of unnecessary calculations and the promotion of learning. Even if the torque measurement in the above pneumatic actuator is just an indication that ANNs are applicable, it shows their potential for other more difficult problems such as measurements on spring return solenoids.

G. Belforte et.al [29] stated that to detect the connection between static and dynamic characteristics of pneumatic valves their operational cycles and conditions of operation are considered. Some variations have occurred in the setup of Cetop and ANSl due to analysis

and testing which enable dynamic measurements with greater accuracy, thereby providing more complete information on the performance that can be achieved by a solenoid valve. Measurements were taken to assess the validity of the suggested method and their results are presented. In order to determine the number of statically and dynamic valve performance, a test method referred to in this document may be applied. Dynamic analysis made it possible to build a new test bench capable of performing more precise dynamic analyses on corn ponents than can be performed under existing standards. The test results have shown that there is no correlation in valve characteristics with the number of operational cyclesrecorded by the valves to be checked, as regards tightness and changeover pressure. It seems to be the dynamic parameters that are of most importance. This applies especiallyto the highest operating frequency which is reduced after a lifetime test.

Lei Wang et.al [30] cohesively denoted that the ability of adaptive feature selection hassled to a distinguished performance in the use of network prediction methods based on residual useful life RULs. A Bayesian large kernel attention network (BLKAN) is proposed to carry out RUL prediction and quantification uncertainty. To allow RUL predictions to be accurate, BLKAN provides for quantification of uncertainty, long- range correlation and channel adaptability in the attention mechanism with a view to efficiently devaluing features. In order to reduce the parameters and computational costs, large kernel Bayesian convolutions that are used for generating attention weights in BLKAN will be decomposed into three simple components. At last, variational inference is introduced to inference probability distributions of the parameters of BLKAN and learn uncertainty-aware attention. As a result of the experiments performed at two sets of datasets, it has shown that BLKAN is more than capable of quantifying uncertainty in RUL predictions and also outpaced baseline comparisons on numerous occasions. The correlation between the degradation patterns and characteristics is shown in the visualization of focus weights.

1.4 OBJECTIVES

I. To diagnose and analyze the faults within the valves of OIS

II. To provide breathable oxygen supply and flow control inside OIS using ORM(Oxygen Redundancy Management) during unfortunate situations.

III. To calculate the Remaining useful life of OIS in LSS.

IV. To establish the Integration with LSS and ensure smooth functioning of OIS using Machine learning models.

1.5 REQUIREMENTS FOR FAULT DIAGNOSIS AND RUL

There are various requirements crucial for effectively computing the Remaining Useful Life (RUL) and identifying faults in solenoid valves. These encompass functional requirements, specifying what the system should do, non-functional requirements, detailing qualities such as performance and reliability, as well as hardware and software specifications necessary for system operation. All these components collectively influence the implementation process of the model, ensuring its accuracy, reliability, and functionality in predicting RUL and diagnosing faults in solenoid valves, which are integral for maintaining the optimal performance of systems and model.

1.5.1 Requirements for fault identification and diagnosis of solenoid valves

For fault identification in solenoid valves, functional requirements involve using NumPy and Matplotlib for mathematical operations and generating graphs. Non-functional requirements prioritize accuracy, scalability, real-time performance, robustness, and adaptability. Hardware requirements include a processor, memory, power supply, multimeter, and pressure gauge. Software needs comprise MATLAB Simulink, Collab, Python 3, and Windows 10.

1.5.1.1 Functional requirements for fault identification

The functional requirements for fault identification include NumPy and Matplotlib majorly for generating graphs and identifying the faults.

I. ***NumPy:*** Used to process the mathematical operations for multi-dimensional arrays and large dataset.

II. ***Matplotlib:*** Used to generate various graphs for the given inputs and other plots according to the dataset.

1.5.1.2 Non-functional requirements for fault identification:

The non-functional requirements for fault identification typically encompass aspects related to system performance:

I. ***Accuracy:*** The system should exhibit high precision in fault detection to maintain operational reliability. It must have high level of accuracy to avoid false positives or negatives in prediction, ensuring genuine faults are identified.

II. ***Scalability:*** It must effectively accommodate fluctuations in the quantity of solenoid valves without compromising efficiency.
III. ***Real-time Performance:*** Fault identification should occur promptly, ensuring responses are timely and appropriate.
IV. ***Robustness:*** The system should demonstrate resilience against external factorssuch as noise, disturbances, and variations in operational settings.
V. ***Adaptability:*** It should possess the capability to adjust to alterations in both the system's configuration and its operating environment.

1.5.1.3 Hardware requirements for fault identification:

Hardware requirements for fault identification of life support system typically involve the components necessary to collect, process and analyze sensor data integrating with the overall system, namely:

I. ***Processor:*** A processor is essential for processing sensor data and executing fault identification algorithms for a mitigate decision making. The choice of microcontroller depends on factors like computational needs, real-time processing abilities, and power consumption constraints. Sufficient processing power and memory is required to execute complex algorithm in real-time.
II. ***Memory:*** Adequate memory, encompassing both RAM and non-volatile memory like flash memory, is necessary for storing sensor data, program code, and any interim outcomes during fault identification.
III. ***Power Supply:*** A stable and dependable power supply is crucial to ensure uninterrupted operation of the fault identification system.
IV. ***Multimeter:*** It is a device utilized for gauging the voltage, electric current, and resistance within the solenoid valve.
V. ***Pressure Guage:*** An instrument used to measure and display the pressure of a fluid of oxygen in the solenoid valve.

1.5.1.4 Software requirements for fault identification:

The software requirements for fault identification encompass the programs, algorithms, interfaces and tools necessary to process sensor data and detect faults :

I. ***MATLAB Simulink :*** It provides a graphical environment for modeling and simulating dynamic systems. It allows you to create block diagrams representing the components and interactions of a system, making it an ideal platform for modeling the behavior of solenoid valves and associated components within the

LSS.

II. ***Python Collab :*** platform that provides powerful tools for simulation, analysis, and algorithm development for a particular time interval and report analysis is robust.

1.5.2 Requirements for Estimation of RUL:

The requirements for estimating the RUL encompass various aspects including functionality, non-functionality, hardware, and software.Functionally, it involves utilizing tools such as Keras for neural network setup, NumPy for numerical data processing, Matplotlib for data visualization, Scikit-learn for machine learning tasks, and TensorFlow for model creation and assessment.Non-functionally, the system must prioritize speed to ensure timely RUL estimates, tackle utilization challenges, guarantee reliability in predictions, and ensure compatibility across different frameworks.Hardware requirements include a capable processor, sufficient hard disk space (at least 8 GB), and a minimum of 1 GB RAM to support real-time performance.Regarding software, platforms like Google Colab and tools like Jupyter Notebook aid in data management and model development, all running within a Windows 10 environment.

1.5.2.1 Functional requirements for estimation of RUL

The functional requirements estimation of RUL include keras, NumPy, Matplotlib, Ski Learn and Tensor flow majorly for generating data into image and predicting the RUL in terms of hours:

I. ***Keras:*** Keras is an open-source neural network library that offers a user-friendly and intuitive interface for defining and configuring neural network layers. It is used for image preprocessing to load the images and to convert them into arrays for image loading, while a Sequential model with Conv2D, MaxPooling2D, Flatten, and Dense layers is being defined for building a convolutional neural network (CNN) for various tasks. The development and implementation of CNN architectures become streamlined and accessible by using keras.

II. ***NumPy:*** To process numerical data present in the dataset. It provides powerful array operations, mathematical functions, and tools for handling large datasets and complex computations. Its array-based approach enables faster execution and easier implementation of algorithms.

III. ***Matplotlib:*** For visualizing data in the form of trained model accuracy and loss, as well as creating histogram plots of neural network layers. It offers a wide range of plotting functions, including the ability to generate various types of plots,

histograms, and 3D models.

IV. ***Ski Learn:*** It is one of the most widely used open-source machine learning libraries in Python. It is built on NumPy, SciPy, and Matplotlib to provide simple and efficient data analysis tools.

V. ***TensorFlow:*** To build, train and compile the model and find its losses and accuracies for each of the trained model.

1.5.2.2 Non-Functional requirements for estimation of RUL

The non-functional requirements for RUL typically encompass aspects related to system performance:

I. ***Speed:*** The system must be capable of processing the provided data and generating RUL estimates in a timely manner. This is crucial for real-time applications where quick decisions based on RUL predictions are necessary. Efficient algorithms and optimized code implementation are key factors in achieving the required speed for RUL estimation.

II. ***Utilization:*** It is used to address the challenges associated with identifying and reducing the risks produced by safe and danger conditions in fields practical applications, technologies, and methods.

III. ***Reliability:*** The rate of failures in predicting RUL should be minimized.

IV. ***Portability:*** It is believed to be quite simple to implement in any framework.

1.5.2.3 Hardware Requirements for Estimation of RUL

Hardware requirements for RUL typically involve the components necessary to collect, process and predict duration integrating with the overall system, namely:

I. ***Processor:*** The choice of processor, such as an i3 or higher, depends on several factors related to the RUL estimation system. A capable processor is essential for handling the computational load involved in processing large datasets and running complex algorithms for RUL prediction. The processor should be compatible with the software frameworks and libraries used for RUL estimation.

II. ***Hard Disk:*** 8 GB. It is sufficient to run the ml model.

III. ***Memory :*** 1GB RAM: It is crucial to have sufficient consideration of memory so that the memory capacity of a system can impacts its performance and abilityin real-time applications.

1.5.2.4 Software requirements for estimation of RUL

The software requirements for RUL encompass the programs, algorithms, interfaces and tools necessary to process data and predict in terms of hours:

I. ***Collab :*** The platform that is beneficial for handling large datasets and complex algorithms. It supports Python programming language, making it suitable for implementing RUL estimation algorithms using Python libraries such as NumPy, Pandas, TensorFlow, and Scikit-learn.

II. ***Jupyter notebook:*** The notebook allows for interactive code execution and documentation, facilitating the development and testing of RUL prediction models.

1.6 SCOPE OF THE PROJECT

The scope of the project encompasses several key areas that are crucial for the successful implementation of fault diagnosis and monitoring in the Oxygen Injection System (OIS) of a submarine's life support system :

I. ***System Understanding and Analysis*:** Comprehensive study of the OIS components, including oxygen sources, pressure regulators, flow control valves, sensors, emergency mechanisms, and their interdependencies. Analysis of the system's normal operating conditions, potential fault scenarios, and their impact on crew safety and mission objectives.

II. ***Fault Simulation and Testing:*** Creation of fault simulation scenarios to mimic various fault conditions within the oxygen injection system. Evaluation of the developed algorithms through simulation and testing to validate their effectiveness in identifying and diagnosing faults.

III. ***Diagnostic Model Creation:*** Development of diagnostic models that can determine the type and severity of faults based on sensor data patterns. Incorporation of historical data and machine learning techniques to enhance the accuracy of fault diagnosis.

IV. ***Real-time Monitoring Interface:*** Design of a user-friendly interface for real-time monitoring of oxygen system parameters and fault alerts. Integration of visualizations and alerts to inform operators about potential anomalies or faults.

1.7 ORGANIZATION OF THE REPORT

The report provides a comprehensive exploration into the critical aspects of fault identification and Remaining Useful Life (RUL) estimation in the Oxygen Injection System (OIS) of a submarine's life support system. Beginning with an elucidation of the system's pivotal role in ensuring crew safety and mission success, the report delves into the necessity of fault identification mechanisms and RUL estimation for optimal system performance. Motivated by the imperative to enhance system reliability and operational efficiency, the report conducts a thorough literature survey to gather insights and methodologies from existing research. The scope and requirements of the project are clearly delineated, paving the way for the detailed exploration of fault diagnostics, solenoid valve design, RUL systems, implementation processes, results analysis, and conclusive recommendations. Through structured chapters, each facet of the project is meticulously examined, culminating in a comprehensive overview and concluding statements that encapsulate the project's achievements and provide avenues for further enhancement.

Chapter 1 starts off with the basic idea of the importance of life support system in a submarine, need of fault identification oxygen injection system and estimation of RUL. Following that is the Motivation behind this project, suggesting the intent of this project is stated, after which the Literature Survey follows. Through the literature survey, all the ideas have been collected, thus paving the way to get the best possible outcome. The scope of the project and the requirements to carry out to design our model are stated clearly. This is followed by the objectives our project and the standards adhered during the project preparation. Lastly the scope of the project is briefed.

Chapter 2 provides a concise overview of fault diagnostics in solenoid valves, outlining a step-by-step identification process illustrated with a flowchart. It discusses the different characteristics and significance of fault diagnostics, and then delves into the conditions necessary for effective fault diagnosis.

Chapter 3 outlines the solenoid valve design, including components and necessary parts of a solenoid valve. It then delves into the operational principles of solenoid valves, covering various inherent principles, with a particular focus on the internally piloted principle that being used. The chapter proceeds to explore the different types of solenoid valves and the typical operations they entail.

Chapter 4 delves into the functionality and characteristics of RUL (Remaining Useful

Life) systems. It outlines different estimation models and discusses the dataset utilized for RUL estimation, incorporating various threshold ranges for both human and ambient parameters. Subsequently, it covers the loading and preprocessing of the generated data and hierarchical clustering. Following this, it elaborates on the employed ML model, specifically BCNN (Bayesian Convolutional Neural Network), elucidating its various layers. The chapter then addresses KL divergence and transfer learning, along with the mathematical equations utilized for RUL estimation, emphasizing Bayes' rule. Finally, it discusses ELBO (Evidence Lower Bound) and KL loss regularization.

Chapter 5 details the implementation of the fault diagnostics process and the estimation of remaining useful life (RUL). It commences by discussing the conditions tested in fault diagnostics utilizing MATLAB and Python. Operational value ranges for multimeter and pressure gauge, pivotal features in fault detection of solenoid valves, are presented in tabular format alongside a V-I graph. Subsequently, various supporting functions employed in MATLAB and Python for graph generation, distinguishing between fault identified and fault not identified, are provided. Following this, RUL prediction is performed based on the diagnostic results, elucidating various states such as healthy status, defective realm, and end of life for RUL estimation. The entire process of RUL estimation using Bayesian Convolutional Neural Network (BCNN), Kullback-Leibler (KL) divergence, and Transfer Learning is elaborated. The compilation and training of the BCNN model are executed using two performance metrics, namely Mean Absolute Error (MAE) and Mean Squared Error (MSE).

Chapter 6 provides clear insights of the various results obtained from the fault diagnostics as fault identified or not and estimation of remaining useful life in hours.

Chapter 7 offers an overview of our work and the concluding statements for our project with further recommendations.

1.8 SUMMARY

This chapter emphasizes on the basic features of the Life support system in a submarine and how oxygen injection plays a vital role in LSS. It talks about how fault identification for solenoid valves in OIS and the estimation of RUL is essential. The Motivation to do this project and various literature reviews referred to enhance the knowledge and works have been included in this chapter.

Furthermore, the project's goals and the issue it aims to address have been clearly articulated. The requirements and scope of the project has been given clearly. Finally, for the better and easy understanding the organization of report has been added which helps in giving idea of the entire report and work done for building the module by adhering to specific standards mentioned above. Hence the overview of the module has been described.

CHAPTER 2
FEATURES OF FAULT DIAGNOSTICS IN SOLENOID VALVE

2.1 FAULT DIAGNOSTICS IN SOLENOID VALVES

Fault Identification and Diagnosis in Oxygen Injection system can be done by finding the faults present in the oxygen supplying valves. Oxygen supply valves are essential for controlling the flow of oxygen to different compartments within the submarine. These valves are typically electronically controlled [1][3] and can be adjusted as needed to maintain the desired oxygen levels.

The oxygen supply valves used in the oxygen injection system are basically solenoid valves. A solenoid valve is an electromechanical device used to control the flow of fluids, such as liquids or gasses, through a pipe or tubing system. It works on the principle of electromagnetism.

The opening and the closing of the solenoid valves are monitored with the help of the multimeter [10][15] present by varying the voltage supply that is required accordingly. Opening and closing of the valves depends on the energized state and the current supply to the coil.

2.2 PROCESS OF FAULT TESTING

The solenoid valve voltage and current ratings according to the manufacturing data is checked. The solenoid valve port configuration is ensured which can be either normally closed (NC) or normally open (NO)[15]. The regulated air pressure with optimal conditions that is generated from the oxygen generator should be ensured with help of a pressure gauge that is present besides thesolenoid valve in OIS.

The air supply port from the oxygen generator is connected to the solenoid valve inlet port and the current supply is connected to the valve terminals. Once power supply is initiated, With the help of pressure gauge, gauge pressure is monitored [15][16] to confirm that the valve is energized (the pressure should increase). At de-energized state NO valve is open and NC valve is closed and pressure is decreased as given in Fig 2.2.

Now with the help of the multimeter, resistance testing and voltage testing is done to ensure that the current flow inside the valve is regulated that provides smooth opening and closing of valve. If the current supply is the given operational range, then it energized and flow of airacross the valve is regulated. The flow ends in the outlet port according to Fig 2.1.

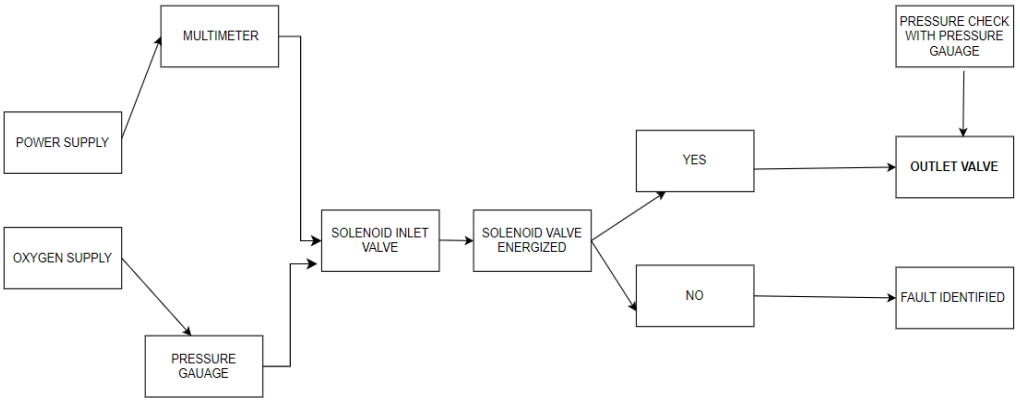


Fig 2.1 Solenoid Valve Testing

The pressure is checked again in the outlet port to ensure there is no leakage of oxygen. Thus, if the given current supply is in the operational range and the pressure is correct, valve would be energized, opening, and closing of valves occur and no faults will be detected.

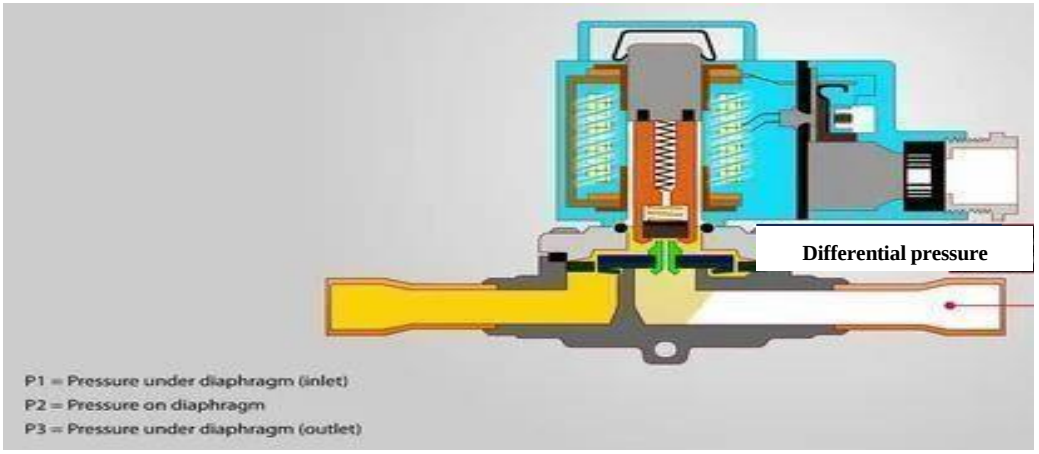


Fig 2.2 Solenoid Valve with Pressure Variation (Adopted from[31])

2.3 FEATURES OF FAULT DIAGNOSTICS IN SOLENOID VALVES

The features of fault diagnostics in solenoid valves encompass a range of functionalities aimed at comprehensive system monitoring, fault detection, and data visualization :

I. ***Random Data Generation****:* The module includes functionality to generate randomdata for different parameters such as power supply status, oxygen supply status, and solenoid valve status. This synthetic data is crucial for testing the fault detection and diagnostic algorithms under various scenarios.

II. ***System Status Monitoring:*** It monitors the status of various system components such as the power supply, oxygen supply, solenoid valve, multimeter, pressure gauge, outlet valve, and detects faults based on their states. By tracking these components, the system can detect deviations from normal operation that indicate potential faults.

III. ***Fault Detection:*** It has logic to detect faults such as low pressure, solenoid valve not energized, or power/oxygen supply being off. These fault detection mechanisms are essential for proactive maintenance and timely response to system anomalies.

IV. ***Data Visualization:*** The module includes capabilities for data visualization usingmatplotlib, specifically for plotting the system status over time using a line graph. This visualization aids in understanding trends, patterns, and anomalies in the system's behavior, facilitating effective fault analysis.

V. ***Threshold Checking:*** It checks the pressure against a defined threshold and takesactions (like turning on the outlet valve) based on the pressure level. When a parameter crosses a threshold, the module triggers actions such as activating the outlet valve. This threshold checking mechanism ensures timely intervention in critical situations.

VI. ***Message Display:*** It displays messages indicating system status, fault detection, and power/oxygen supply status. These messages serve as alerts for system operators, enabling prompt response to detected issues.

VII. ***Time-Based Operation:*** The code operates in a time-based manner, iterating overa range of time intervals (60 seconds in this case) and collecting data for each interval. This time-based operation facilitates data collection, analysis, and monitoring over extended periods, enhancing system visibility and diagnostic accuracy.

2.4 CONDITIONS FOR FAULT DIAGNOSTICS

The four fault identification conditions and statements are:

I. ***Condition 1 - Valve opened,*** **No fault is identified**: In this condition, the solenoid valve is open, and no faults are detected in its operation. It signifies that the valve is functioning correctly, allowing the controlled flow of oxygen into the submarine's environment without any issues or abnormalities. This condition ensures that the oxygen supply remains uninterrupted, supporting the life support system within the submarine.

II. ***Condition 2 - Valve opened,*** **Fault Identified –** Low pressure is exerted: When this condition occurs, the solenoid valve is open, but a fault is detected due to low pressure exerted in the system. Low pressure can indicate a potential risk to the oxygen supply, leading to inadequate oxygen levels within the submarine's sphere.

III. ***Condition 3 - Valve closed,*** **Fault Identified –** Valve not energized: In this scenario, the solenoid valve is closed, but a fault is detected because the valve is not energized. A non-energized valve means that the mechanism responsible for opening and closing the valve is not functioning correctly, leading to a disruption in the oxygen supply process.

IV. ***Condition 4 - Valve closed:*** **Fault Identified –** Power supply or oxygen supply is off: This condition occurs when the solenoid valve is closed, and a fault is detected due to the power supply or oxygen supply being off. A loss of power supply or oxygen supply can severely impact the submarine's life support system, risking the crew's safety and well-being.

2.5 SUMMARY

This chapter gives a glimpse of fault diagnostics in general and then talks about the process on fault testing for solenoid valves in OIS. Next, various features of fault diagnostics of solenoid valves with respect to our objective is elucidated. Finally, the fourmajor conditions involved in fault identification of solenoid valves is briefed.

CHAPTER 3

SOLENOID VALVES

3.1 SOLENOID VALVES- INTRODUCTION

The Solenoid valves are essential hydraulic devices utilized for regulating fluid flow, whether it's liquids or gases, through pipes or tubes. They operate on the principles of electromagnetism and are the primary control elements in fluidics. Solenoid valves come in different types, employing mechanisms like linear action, plunger-type actuators, or pivoted-armature actuators. Their operation is influenced by factors such as the applied current's characteristics, the strength of the magnetic field, the mechanism for fluid regulation, and the type of fluid being controlled.

3.2 Components of solenoid valves

The main components of a solenoid valve include the coil, plunger, coil windings, diaphragm, valve body, core spring, orifice, inlet & outlet port.

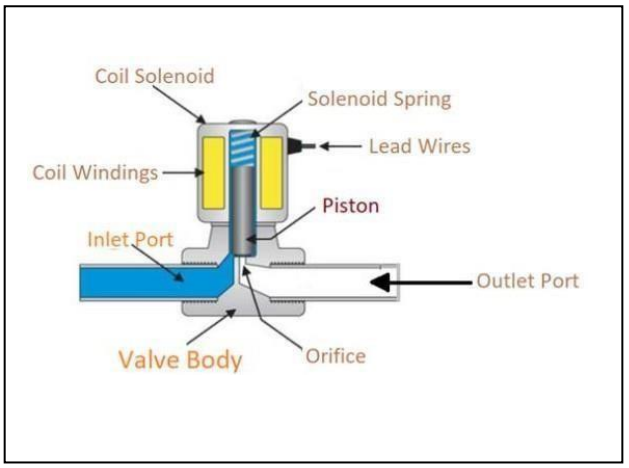


Fig 3.1 Solenoid valve diagram (Adopted from [35])

3.2.1 Coil

The solenoid coil, consisting of insulated copper wire embedded in a core tube and capable of generating magnetic fields [32] when currents pass through it, is an essential part of the solenoid valve. For the purposes of protecting these coils, they are sealed by materials such as epoxy. They are designed to deliver the required magnet field strength for valve operation in their specific shapes and sizes, which have a variety of rolling patterns and wire

thicknesses. These circuits have two terminals for external connections to ensure compatibility with electrical supplies and control systems, which are rated according to the voltage and current levels. The magnetic force produced is determined by the number of wires turns and the gauge[34] , which influences the movement and operation of the valve. A continuous operation without overheating is defined by the coil's duty cycle, which is essential for frequently used applications. To open and close the valve quickly in response to electrical changes, which are beneficial for accurate fluid flow control, it is important to have a rapid response time. Different voltage options, e.g., 12V, 24V, 110V, 220V, are available for solenoid valve coils.

3.2.2 Plunger

The core of the solenoid, also known as the armature or plunger, is a moveable component of the device. It is a soft magnetic metal, which means it is easily magnetized and demagnetized in low magnetic fields. When the coil is energized, it creates a magnetic field that attracts the core, opening or closing the valve.

3.2.3 Core spring

The core spring in a solenoid valve functions[32-34] to return the core to its original position once the magnetic field is deactivated. Depending on valve operation, it differs from one design and configuration to another. It is usually made out of durable, flexible materials such as stainless steel and able to withstand environmental conditions and mechanical stresses. In order to meet special applications and valve requirements, these springs are often helically spun in such a way that they generate the necessary amount of force for closing the core tube. The core spring is a crucial part of the mechanically return mechanism, responding to the current in the coil and allowing the solenoid valve to open bymovement of the inner tube. The spring force closes the core tube as soon as the current hasstopped and prevents fluid flow. Critical to response time and reliability, the spring's force and stiffness are calibrated to counteract magnetic attraction when energized and then returnthe core tube to its closed position when the magnetic field weakens. The spring size and characteristics must match that of the solenoid valve system as a whole, e.g. its main tube and coil being configured according to the required application and flow rate.

3.2.4 Diaphragm

The diaphragm of the solenoid valve is a flexible barrier that separates the solenoid assembly from the liquid, to be able to concentrate and regulate its pressure. They are flexible and thin, responding to changes in temperature or magnetic field from solenoid coils, which could be strengthened with layers of fabric for strength. Diaphragms of circular or oval shape are fitted in a tightly packed valve body that features a moving center regulating fluid flow. When the solenoid coil is activated, it moves the core tube, pushing the diaphragm to flex and opening a passage for the flow of fluid. When the diaphragm is deenergized, it reverts to its normal resting position and closes the passage so that fluid does not flow. Diaphragms are essential for maintaining the tight seal, which may be supplemented by other sealing rings or gaskets to prevent fluid leakage when closed. Special fluid, temperature and pressure conditions must be taken into consideration in the material, size and design of the diaphragm. In order to ensure compatibility and durability, in particular during frequent open close cycles, there are different materials for each of the fluids. The diaphragm's flexibility and responsiveness, which is crucial for ensuring optimal valve performance, allow a swift and effective monitoring of flow in response to changes in gravitation field or pressure.

3.2.5 Valve Body

The solenoid valve body serves as the main enclosure for crucial internal components like the diaphragm, disc, seat, and ports, essential for fluid control. The body[34] usually has a cylindrical or blocklike shape, houses inlet and outlet ports made out of materials such as brass, stainless steel, aluminium chosen on the basis of its fluid type, temperature, pressure environment. The installation of the pipeline is made easier with twisted or flanged connections. In addition to its primary function of protecting itself from internal elements, it also plays an important role by housing the passages and chambers for regulating fluid flow. Various valve configurations (2-way, 3-way, 4-way) dictate the body's design and port arrangement, influencing fluid pathways. Featuring sealing surfaces where movable components create tight seals when closed, precise alignment and design of these surfaces are crucial. In order to ensure the integrity of valves, especially in challenging conditions, a body is intended to be a buffer from external factors. In order to achieve optimum valve performance, proper alignment with other components such as bonnets, coils and seals is essential. The material and design of the body, resistant to wear, corrosion, or deformation, guarantees that a valve shall be reliable and operational for a longer time due to its durability and ability to withstand prolonged stresses, pressures, temperature variations.

3.2.6 Orifice

An aperture built into the diaphragm or valve body allows line pressure to be used to operate the valve in indirect or semi-direct acting solenoid valves. Orifices are positioned strategically inside the fluid route and are typically made of brass, stainless steel, or non-corrosive alloys. Their selection is dependent on the fluid compatibility and wear resistance. The design [34] of these orifices may vary, in particular with a small hole appearing between the valve body or related component sometimes involving control elements such as needle valves for flow rate adjustment. They play an essential role in regulating the controlled quantity of fluids to pass through the main valve when it is shut, mitigate system pressure and prevent problems such as water shaking or cavitation which are caused by a sudden change in flow. The orifice minimizes sudden spikes in the pressure and shields the system from damage or noise to ensure a stable, consistent force by bleeding out part of the fluid. An option to adjust the orifices for individual pressure relief depending on specific application needs can be provided by some solenoid valve designs. In order to ensure functionality under a variety of environment conditions, the orifices and their enclosures must be designed in such a way as to protect from external contaminants.

3.2.7 Inlet Port/Outlet Port

The inlet and outlet ports of a solenoid valve serve as entry and exit points for the fluid or gas it controls. The inlet port connects to the fluid supply, allowing the medium to enter the valve, while the outlet port releases the fluid after it has been regulated by the valve. Proper alignment and connection of these ports are vital for the efficient operation of the solenoid valve, ensuring accurate control of fluid flow in various industrial and commercial applications, such as irrigation systems, pneumatic machinery, and HVAC systems.

3.2.8 Seal

The solenoid valve's seat functions as a closing orifice that interfaces the disc, which ensures leakage free seals [9] when valves are closed. There may be no seat at all depending onthe valve design. Constructed from corrosion and erosion-resistant materials like stainless steel, brass, Teflon (PTFE), or elastomers, the seat material is chosen based on fluid compatibility and environmental conditions, critical for resisting wear, corrosion, and chemical damage. The seat, which prevents the flow of fluids through the valve and ensures a secured seal, shall be made from a sealed surface that is in alignment with the movement device or diaphragm when it is closed. The sealing action shall involve pressure on the movable element, which may be assisted by another seal ring or gasket to ensure a robust.

3.3 WORKING OF SOLENOID VALVE:

At the center of the solenoid is an electromagnetic inductive coil connecting to a plunger, or iron core. It can be either ordinarily closed (NC) or normally open (NO) while it is at rest. Normally closed valves should be shut and normally open valves opened during the deenergized state.

I. ***Energizing the Coil:*** An electric current shall be supplied to the electromagnet for controlling the solenoid valve. The coils generate a magnetic field when they're switched on.

II. ***Attracting the Plunger:*** The magnetic field draws plungers or pistons toward the center of the coil. There is a tendency of plunger movements to be at odds with the force of springs or any mechanical resistance.

III. ***Opening or Closing the Valve:*** The inner valve is operated, either by lifting or decreasing the sealing mechanism, as the plunger moves. This action can enable or preventing fluid to flow from the valve.

IV. ***De-energizing the Coil:*** The magnetic field dissipates and any mechanical force, e.g., a spring, returns the plunger's resting position after removal or interruption of electrical power.

3.4 PRINCIPLE OF SOLENOID VALVE:

The solenoid valves shall be described according to their mode of operation, as well as the number and direction of the flow paths. The three types of direct acting which can beused are: Direct Acting, Internal Piloting and Semi Directed.

3.4.1 Direct Acting

Direct operating solenoid valve has simple working principle [32][33], when it is normally closed for lack of power, the plunger blocks the orifice with a seal on the valves. This closure is the result of a spring. When you apply electricity to a coil, it creates waves that draw the plunger up and overcomes spring's force. As orifice size increases, the static pressures are increased in direct acting solenoid valves. The solenoid action needs to be increased in order for the static pressure increase to occur; therefore, a larger magnet field is required. It's opening the orifice and allowing a flow of media. The normal open valves contain the same components, but they're working in a different way. The minimum

operating pressure and flow rates are entirely dependent on the solenoid valve's orifice diameter and magnetic strength. Direct solenoid valves are normally employed in cases where the flow rate is relatively low according to Fig 3.2. For use from 0 bar up to maximum authorized pressure,the Direct Positioned Solenoid Valves are not required to apply at least a minimal operatingpressure and an equalized pressure.

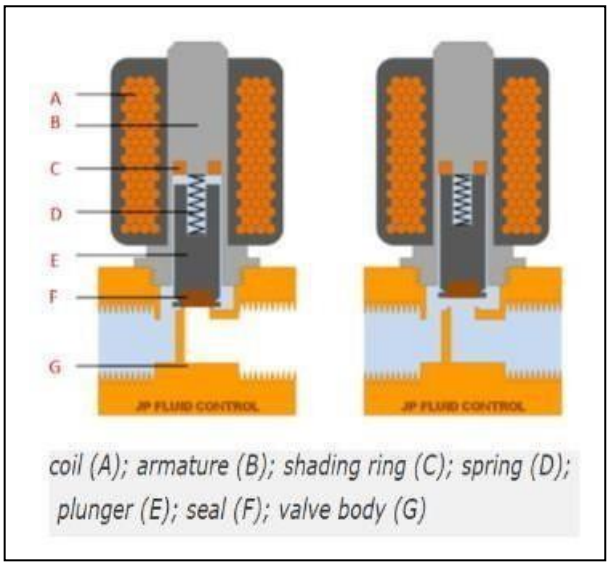


Fig 3.2 Direct Acting(Adopted from[32])

3.4.2 Indirect Acting

Indirectly operated solenoid valves, that are also known as servo valves or pilot valve and operate with a pressure differential between the medium in front of the outlet port from which the valve is opening and closing. Accordingly, it is typical that they require a minimal differential pressure of about 0.5 bar. They used for a high flow rate and high pressure. A rubber sleeve, also called a diaphragm, separates the inlet and outlet ports. In order to allow the medium to pass into the upper part of the chamber from the intake, there is a small hole in the membrane. When the fluid becomes clogged, it generates a shutdown force due to its larger displacement area at the top of the diaphragm. When the valve is opened, the core opens the orifice, releasing pressure from the top of the diaphragm. Then the valve will open as a result of pressure on the line. The inlet pressureunder the membrane and support spring above it shall ensure that the valve is closed for a normally closed indirect actuator solenoid valve. A small channel according to Fig 3.3shall connect the chamber beneath the membrane to a port at low pressure. The pilot orifice opens as soon as the solenoid is activated, causing a decrease in pressure on the

membrane. The membrane shall be raised, allowing the medium to pass out of the inlet port into the outlet port due to differences in pressure on both sides of the membrane. There are identical components in a standard open valve, but it works the other way. The extra pressure chamber above the membrane acts like an amplifier, which means a small solenoid will still be capable of controlling very high flow rates. For media flows in one direction only indirect solenoid valves shall be used.

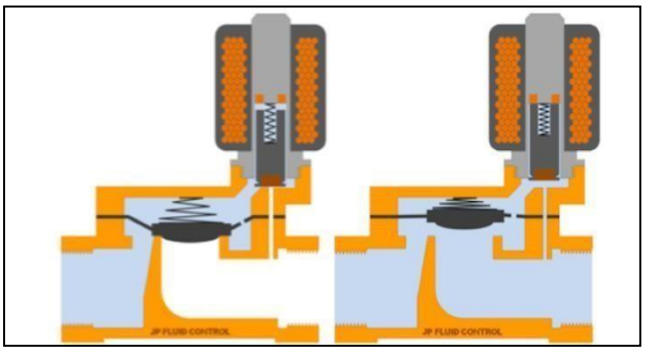


Fig 3.3 Indirect Acting (Adopted from[32])

3.4.3 Semi Direct Acting

The properties of the direct and indirect valves are combined in a semidirect solenoid valve. They can work at 0 bars, yet they can also handle great flow rates. They are similar to an indirect valve, but they also have a movable membrane on each side with small openings and pressure chambers. There is another difference: the solenoid plunger is in direct contact with the membrane. The membrane shall be directly pushed through the opening of the valve by lifting the plunger. In parallel, a plunger that has a slightly larger diameter than the membrane's first orifice opens another one. That will cause the pressure in the chamber on top of the membrane to decrease. Consequently, not just the plunger itself but also the difference in pressure can lift the membrane. The combination will result in a valve that operates from zero bars and is capable of controlling flow rates very high.

3.5 SOLENOID VALVE TYPES

Solenoid valves can either be two way or three ways.

3.5.1 Two-way solenoid valves

There are usually only one outlet and an outlet port for two-way solenoid valves which is given in Fig 3.4. This type of solenoid valves, used for blocking or allowing fluid flow, has a single upstream and one downstream port. The solenoid valve may

be configured to be normally open and normallyclosed; normal state is the state in which the solenoid valve is deenergized. A normally open valve opens when de-energized and closes when energized. When deenergized and opened when energized, the normally closed valve sshall close.

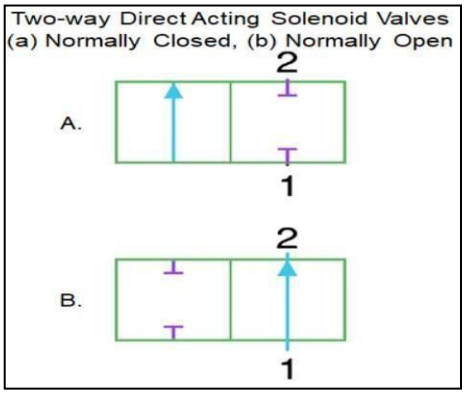


Fig 3.4 Two- way solenoid valve (Adopted from[34])

3.5.2 Three-way Solenoid Valves

There are usually only one outlet and an outlet port for two-way solenoid valves which is given in Fig 3.4. This type of solenoid valves, used for blocking or allowing fluid flow, has a single upstream and one downstream port. The solenoid valve may be configured to be normally open and normallyclosed; normal state is the state in which the solenoid valve is deenergized. A normally open valve opens when de-energized and closes when energized. When deenergized and opened when energized, the normally closed valve sshall close.

For a normally closed three-way valve, when the valve is de-energized, fluid doesn't flow from the inlet port to the outlet port, while the exhaust port is open. When the generator is turned on, an outlet port shall open and connect to an exhaust port.On the other hand, it is possible to select flow direction from one port to another through a single function.

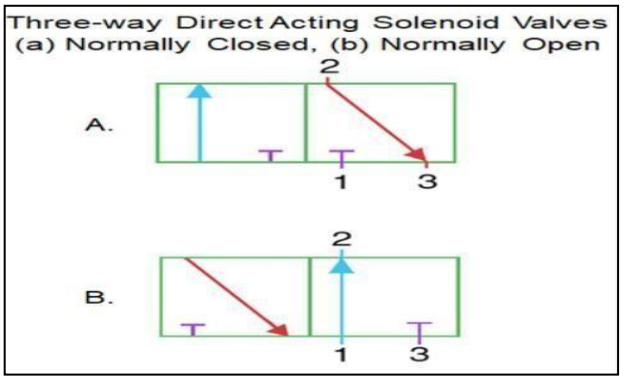


Fig 3.5 Three-way solenoid valves (Adopted from [34])

3.6 SOLENOID VALVE OPERATIONS

Solenoid valves can operate either in normally open or normally close or bistable state.

3.6.1 Normally Closed solenoid valve

At de-energized state the valve is closed and that means no flow of liquid/gas. When current is supplied to the coil [32], an electromagnetic field is induced in the coil which will force the piston to move upwards overcoming the spring force. This makes the Plunger to act as a return mechanism and the spring then pushes the plunger back to its resting position, which is often the default position of the valve depending on the valve's design given in Fig 3.6

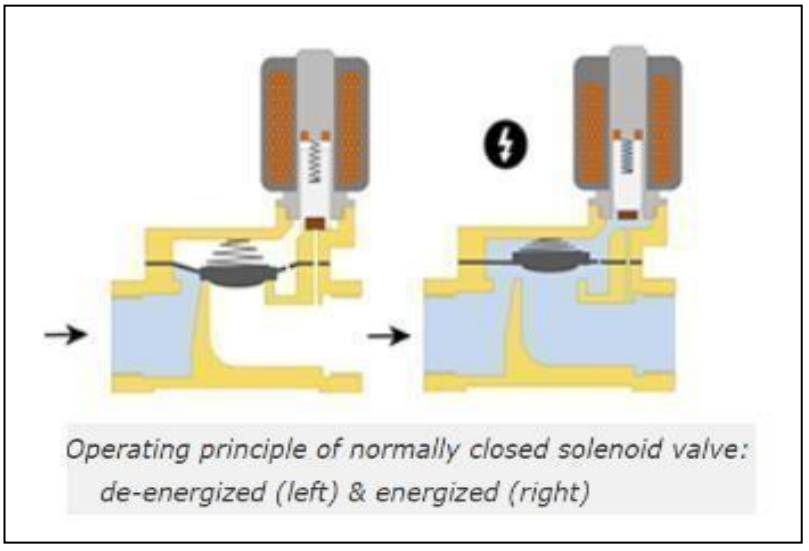


Fig 3.6. Normally closed solenoid valve (Adopted from[32])

3.6.2 Normally Open Solenoid Valve

At de-energized state the valve is open. That means the flow is continuous. An electromagnetic field is incorporated and initiated in the coil when there is a current supply which will force the plunger to move down and helps to overcome the spring force to intrude and stop the flow.

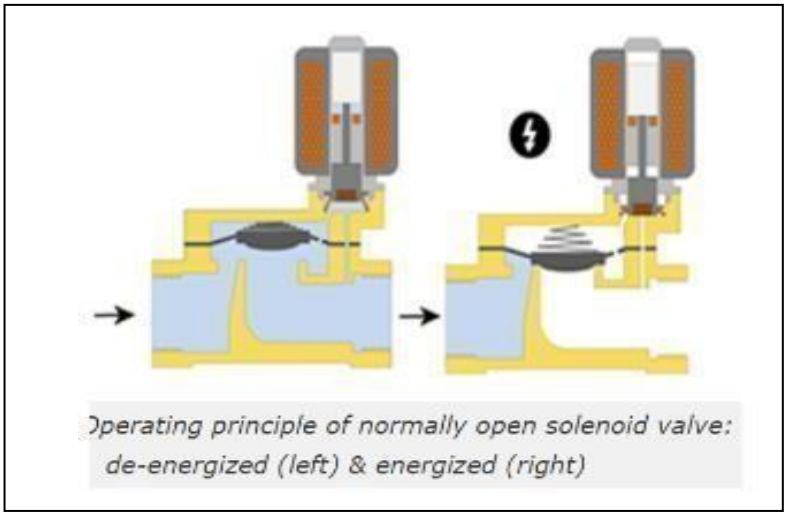


Fig 3.7 Normally open solenoid valve (Adopted from[32])

3.6.3 Bi-stable solenoid valve:

A momentary power supply is utilized to initiate switching between the bistable solenoid valves. The valve is going to stay with no power in that position. When it remains to maintain in the current position even when there is no power used, the valve doesn't open or shutdown. This is done by using permanent magnets instead of springs.

3.7 SUMMARY

The chapter starts with the introduction of solenoid valves and elucidates about the components of solenoid valves. The working is clearly explained and the entire process of opening and closing of valves is given. Further, the principle of solenoid valves i.e. direct/indirect/semi acting are given a clarity upon that is followed by the solenoid valvetypes. Later, the different operations performed by solenoid valves is briefed.

CHAPTER 4

WORKING & FEATURES OF RUL

4.1 REMANINIG USEFUL LIFE

Remaining Useful Life (RUL) is the length of time a machine is likely to operate before it requires repair or replacement. It helps to identify the total working hours of the machine, operating efficiency, and unplanned downfall duration. Finding out the remaining useful life for a LSS is very important so that the crew members know about the remaining operational time therefore increasing the reliability of the mission.

4.2 RUL ESTIMATION MODELS

The RUL estimation models gives a training method that uses historical data and this helps to predict or estimate the remaining useful life. The term life in this context denotes the operational lifespan of a machine, determined based on the specific metric used to measure the system's longevity. Likewise, value evolution based on usage, distance travelled, number of cycles or other quantity which refers to the length of life may be seen as time evolution. These models are useful when we have historical data and information such as:

1. Run-to-failure histories of machines like the one that must be diagnosed.
2. The known threshold value of one condition indicator that indicates failure regime in near end.

The following process is generally used for the use of RUL estimation models [39]:

1. The correct RUL estimation model should be selected based on the specific data and knowledge available.
2. The corresponding model object must be created and configured. Use the historical data that we have to build an estimation model.
3. Estimate RULs for the test component by reference to data from similar typesof tests as the historical data.

There are 3 different models for estimating the RUL (Fig 4.1) i.e., Similarity models, Degradation models and Survival models.

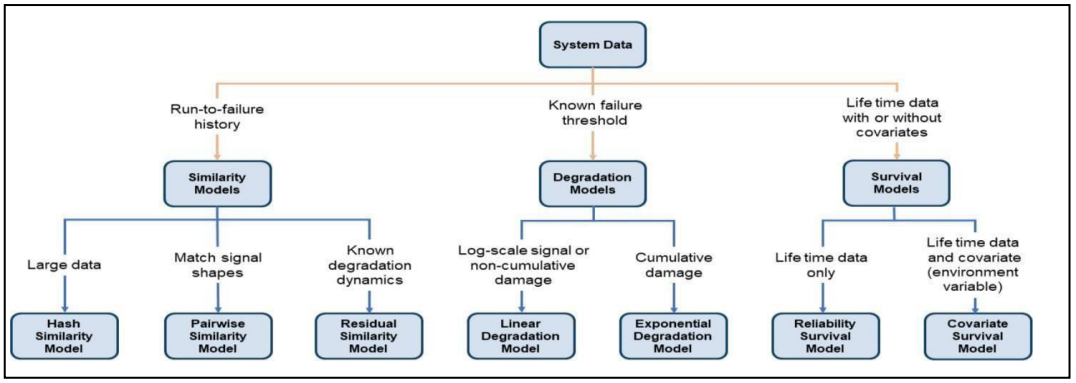


Fig 4.1 RUL Estimation Models (Adopted from [33])

4.2.1 Degradation model

For predicting the future condition, degradability models rely on past behaviour. Given the degradation profiles of the ensemble, that type of RUL calculation corresponds to an exponential or cumulation model for a condition indicator. Following to this it uses the degradation profile of the test component to find the time remaining by statistical means unless and until the indicator reaches the given threshold value. Initialize the model using historical information about health of a group of similarly configured components such as multiple machines manufactured to the same specification, having created the degradation model object. Use the right fit to do so. Then using predict RUL, forecasting the remaining useful life of similar components is possible. Only a single condition indicator can be used in the degradation models. The types of degradation model types are as follows:

I. Linear degradation model

A linear decay model describes degradation behavior as a linear stochastic process that has an offset term. When the system is not subject to cumulative deterioration,linear degradation models are appropriate**.**

II. Exponential degradation model

Exponential decay model describes degradation characteristics as exponential stochasticprocesses that have an offset term. For a test component that has accumulated damage, empirical degradation models are useful.

4.2.2 Survival model

The method used in survival model is based on statistics that incorporates the mode of time interval data for the survival analysis [39]. It is useful when there are no available complete run-to-failure histories, but instead have all data related to life span. For instance, one might know the number of miles every engine in an ensemble has been running before it needs to be repaired or how much time each machine on your band had operated up until its failure. Then using the reliability Survival Model in this case is optimal. The probability distribution of failures is estimated in this model, considering the historic information on failure time intervals for a fleet of similar components. Life expectancy, along with some other variable data that correspond to the RUL. Information such as the provider of the component, the production systems in which it is applied or a manufacturing batch are included in covariances, also referred to as Explanation Variables. Use a covariate survival model in this case. Survival model is a proportional hazard survival model that considers of the life spans and various covariates for computing the survival probability of the given test component. The types of survival model are as follows:

I. Reliability survival model

These are used to predict a system's or component's RUL or time to failure using historical data and features. These models are frequently used in predictive maintenance applications to estimate the probability distribution of failure times and to optimize maintenance plans.

II. *Covariate survival model*

This used in survival analysis to understand the relationship between the time until an event occurs and several predictor variables, or covariates.

4.2.3 Similarity model

Similarity models rely on known behavior of similar machines from historical databases to predict RUL for a test machine. These models represent a correlation of the trend in test data or condition indicators with similar information that is obtained from similar or different similar systems. For the following reasons, similarity models are useful:

1. There will be run to failure data which are collected from similar components.
2. Run-to-failure data refers to data that begins during healthy operation and ends when the machine gets close to failure or requires maintenance.

There are three types of similarity model which are as:

1. ***Hashed-feature model***

The program estimates the hashed features such as weights and stores them in the similarity model while the model fits on a hashSimilarityModel object[39]. The program finds the hashed features while running the predictRUL with data from a test component, and then compares the output to the values in the historical hashed feature table.

2. ***Pairwise model***

Pairwise Similarity Model, is used for identifying the components with the highest correlation between their historical deterioration trajectories and the test component's, pairwise similarity estimation establishes RUL. In comparison to the hash similarity model, pairwise similarity estimates can yield superior results by accounting for the degradation profile's evolution over time.

3. ***Residual similarity model***

Residual-based estimation process that the architecture of the one model is used as a starting point, and the final layer(s) are replaced or fine-tuned for the new task., such as ARMA models. Next, it calculates the residuals between the test component's data and the data predicted by the ensemble models.

4.3 DATASET CONSIDERED FOR ESTIMATION OF RUL

Upon the identification of a fault in the solenoid valve, an automated trigger initiates the data collection[38] process for estimating the RUL. This fault event serves as a crucial marker to activate sensors for the continuous monitoring of both ambient and human parameters. Sensors are installed to capture ambient parameters such as Temperature (Celsius), Humidity(RH%), Absolute pressure(hPa), Oxygen concentration (%) at regular intervals following the fault detection. Wearable sensors are used to monitor human parameters such as Body Temperature (Celsius), SpO2 (%), Blood Pressure(mmHg), Heart Rate(bpm). The data collected after fault identification is stored and used for processing further.

4.4 THRESHOLD RANGE OF PARAMETERS

This section provides an overview of the parameters used for estimating of RUL. Each parameter is explained with its description in detail. Additionally, Table 4.1 specifies the desired operating ranges for each parameter, ensuring accurate and reliable predictions.

Table 4.1 Parameter description and its operating ranges

S.No	Parameter	Description	Operating Range
1	Personal Sphere Diameter	The personal sphere is the sphere in which the crew members live in during the mission.	21 meters
2	Oxygen Levels	The oxygen levels must be maintained regularly inside sphere soas for the crew members to carry out the mission successfully and sustain themselves.	O2 during rest 26 L/hr O2 during work 45 L/hr O2 in average 35.5 L/hr
3	Humidity	Monitoring and controlling humidity levels are critical to maintain optimal conditions for the system and the crew.	79% RH - 105% RH
4	Ambient Pressure	It plays a critical role as it directly impacts the structural integrity and performance of the submarine's hull and systems.	110 Pascal
5	Body Temperature	Maintaining body temperature in submarines is essential for ensuring crew performance and safety during extended underwater missions.	98.6 Fahrenheit
6	Blood Pressure	Maintaining body pressure in submarines ensure the safety of the crew, and optimize the operational lifespan.	60 mmHg -120 mmHg

4.5 LOADING AND PRE-PROCESSING OF GENERATED DATASET

In the initial phase of the analysis, image data, which represents various parameters is mapped to their respective RUL values for training. The dataset is partitioned into training,

testing and validation subsets by allocating 80% of the data for training, 10% for testing and 10% for validation to ensure a balanced distribution of samples across the sets.

By associating each image with a distinct RUL value (Fig 4.2), the predictive model gains insights into the system's longevity efficiency. Subsequently, a custom function has been implemented to load and preprocess the images from the provided paths. This function sequentially iterates through each image path, loads the image, and resizes it to a consistent dimension of 100x100 pixels. To standardize the pixel values and facilitate model convergence, we normalized the image arrays by dividing each pixel intensity value by 255.0, ensuring a range between 0 and 1.

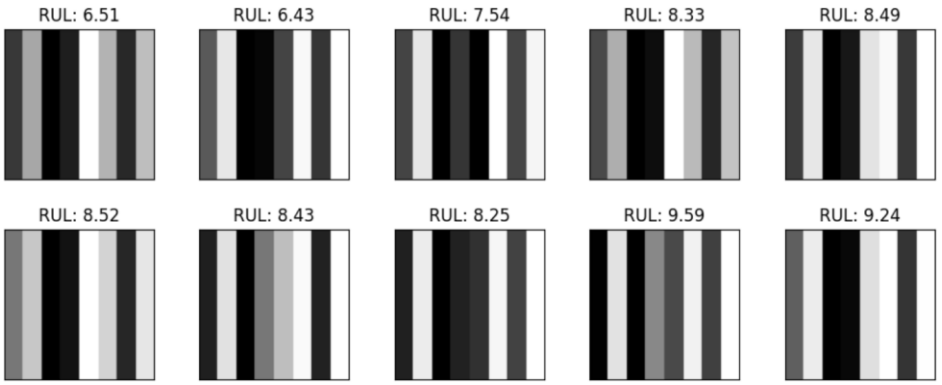


Fig 4.2 Random sample of 10 images

To further enhance the computational efficiency and reduce the complexity of the image data, RGB images is converted to grayscale. This transformation combines the colour channels of each image using weighted average coefficients to produce grayscale representations, effectively reducing the dimensionality of the data while preserving essential features and patterns.

4.6 HIERARCHAL CLUSTERING

Hierarchical clustering is employed as a powerful unsupervised machine learning technique to categorize the operational states of a machine into distinct classes: Healthy State, Defective Realm, and End of Life of the sphere. The hierarchical clustering algorithm initially treats each data point as an individual cluster and iteratively merges them based on their similarity or distance metrics until a hierarchical tree-like structure, known as a dendrogram (Fig 4.3), is formed.

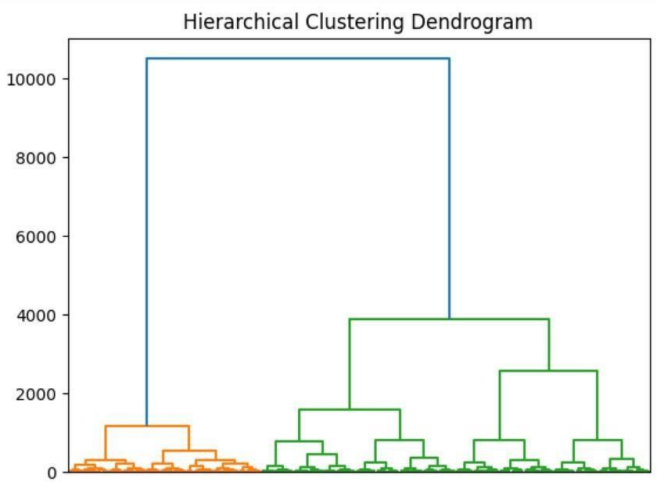


Fig 4.3 Representation of merged clusters

By setting the number of clusters to three, the clustering algorithm classifies the machine's operational data into these predefined categories (Fig 4.4), providing insights into its classifications.

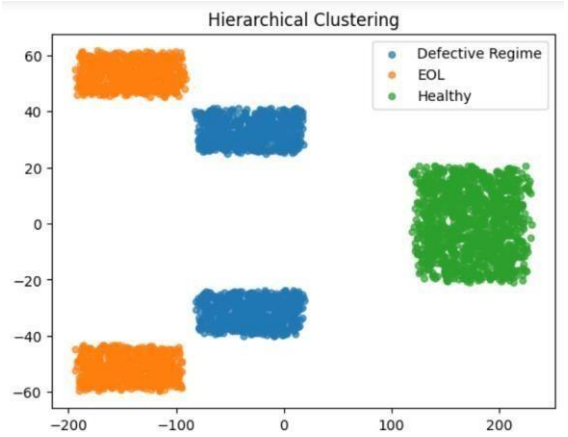


Fig 4.4 Hierarchical Clustering of Classes

The Healthy State class represents the optimal operational condition where the LSS functions efficiently without any issues, while the Defective Realm class signifies the presence of anomalies or minor faults requiring attention, maintenance, or corrective actions to prevent further deterioration. This is the crucial state of determining RUL and to decide whether the mission is to be continued or terminated. Lastly, the End-of-Life class indicates critical wear and tear, severe damage, or imminent failure, signalling the need for immediate intervention, replacement, or decommissioning the mission and returning to the surface.

4.7 BAYESIAN CONVOLUTIONAL NEURAL NETWORK (BCNN)

In this project, we use BCNN with regression tasks for the prediction of RUL. A BCNN[19][22] is a type of neural network that incorporates an inference based on statistical analysis for the purpose of determining uncertainty. Point estimates of their predictions or parameters are often provided by traditional neural networks such as a feedforward or convolutional neural network.

In traditional neural networks, weights are point estimates learned from the training data. However, these point estimates do not capture the uncertainty associated with the learned weights. BNN address this limitation by treating the weights as probability distributions rather than fixed values. When integrating Bayesian principles into CNN, we get BCNN. In BCNNs, both the fully connected layers and convolutional layers can be Bayesian, allowing for uncertainty quantification at multiple levels of the network architecture. By contrast, a Bayesian neural network could quantify uncertainty in prediction and model parameters. They do this by having network weights or parameters treated as random variables rather than fixed values. Those random variables will be assigned probability distributions, most commonly a Gaussian distribution, representing uncertainty in the weights.

4.7.1 BCNN with Regression Model

In this project, we are employing residual and hashed similarity-based transfer learning approach to predict the RUL of LSS. The transfer learning is used for acquiring the previously trained model's (pre-trained model) knowledge and applying it to a new(target) BCNN model, for a similar task, with minimal modifications to the existing model's architecture. It allows the learned features from the pre-trained model to effectively transfer them to the new RUL prediction task.

Regression is used to estimate continuous variables, for calculating RUL of the system. The Bayesian CNN, which is employed in regression tasks, captures complex relationships and patterns in the data to make accurate and reliable predictions. Despite the incorporation of Bayesian VI for uncertainty estimation, the fundamental architecture of the Bayesian CNN remains largely unchanged. The only modification that was made to the pre-trained model is the replacement of its final dense layer with a Bayesian VI layer. This addition enables that the knowledge and information learnt from the pre-trained task are

applied to the target task. By doing so, we enhance the model's predictive accuracy and reliability.

4.7.2 Convolutional Layer

This layer performs a 2D convolution on the input image data, extracting features through a set of 32 learnable filters (Fig 4.5) with a size of 3x3 pixels each. The ReLU activation function introduces non-linearity to the model, helping it capture complex patterns and relationships within how the RUL is interlinked.

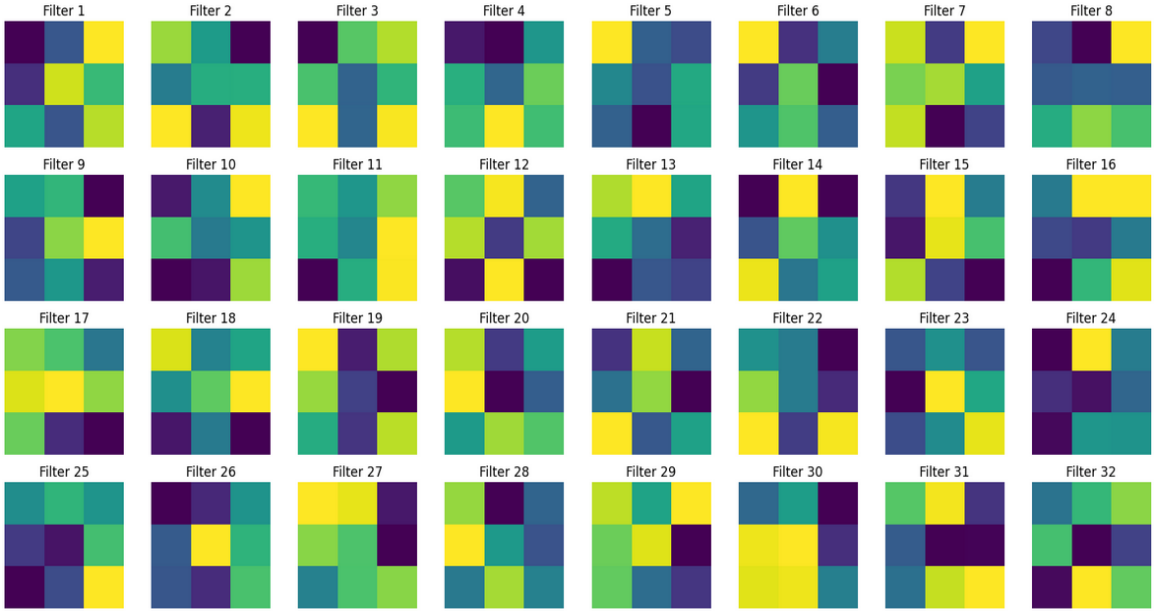


Fig 4.5 Thirty-two learnable filters (3x3 pixels)

4.7.3 Flatten Layer

It converts the 2D feature maps produced by the convolutional layer into a 1D vector. This flattening operation enables the subsequent dense layers to process the extracted features as a single continuous input.

4.7.4 Pooling & Dropout Layer

In Bayesian pooling, uncertainty estimates are propagated through pooling layers by considering distributions over pooling operations. Setting dropout rate at 0.5, it serves as a regularization technique to prevent overfitting by randomly setting 50% of the input units to 0 during each training iteration. This dropout mechanism encourages the networkto learn more robust and generalized features.

4.7.5 Dense Layer

This densely connected layer consists of 128 neurons and applies the ReLU activation function to make sure the model does not predict negative values. It performs a weighted sum of the input features from the flattened layer, enabling the model to learn complex representations and higher-level features from the extracted image data.

4.8 TRANSFER LEARNING

The knowledge obtained from the pre-trained model improves the Bayesian CNN's (Fig 4.6) performance and robustness on the new task by transferring and adapting previously learned features and patterns. The pre-trained CNN (Bayesian CNN) is initially trained on a specific dataset using the Mean Absolute Error (MAE) loss function. To optimize transfer learning process, the model is partially frozen by setting the trainable attribute to false for all but the last four layers. This ensures the learned features in the previous layers are retained and not changed during subsequent training. This transfer learning approach optimizes the model to generalize better to the new prediction, resulting in enhanced accuracy and reliability.

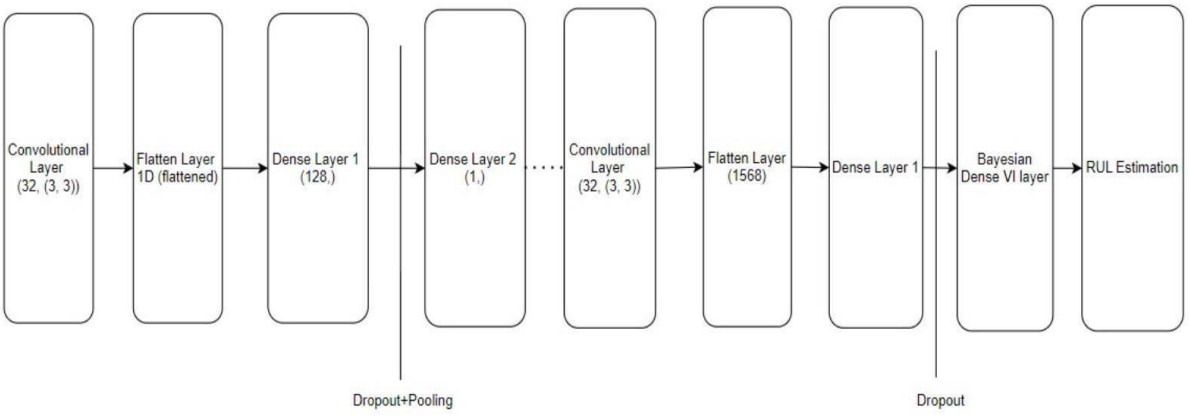


Fig 4.6 Architecture of pre-trained BCNN model

4.9 KULLBACK-LEIBLER(KL)

The KL divergence is a measure of how one probability distribution diverges froma second probability distribution. It quantifies the difference between two probability distributions, P and Q. The KL divergence is not symmetric and is always non-negative. It reaches zero if and only if P and Q are identical. In the context of reliability, the KL divergence can be used, for calculating, RUL using BCNN [41] for set of parameters

considered in this research work.

$$\boldsymbol{\pi(\omega|D) = \frac{p(\omega)p(D|\omega)}{\int p(\omega)p(D|\omega)d\omega}} \quad \textbf{.....Eqn 4.1}$$

The $\boldsymbol{\rho(m|D) = \rho(m)p(D|m)}$, is the evidence or the marginal likelihood. The joint distribution is defined by latent variables p(ω) and choice of model/likelihood p(D| ω). The KL divergence between two probability distributions P(x) and Q(x) is given by:

$$\mathbf{D_{KL}(P||Q) = \int (x) \log(\frac{P(x)}{Q(x)}) dx} \textbf{.....Eqn 4.2}$$

Where, define variables

ω: The weight associated with the BCNN parameters.

D: Dataset of that are labelled with corresponding RUL values for their respective images with their parameters.

p(ω|D): Posterior distribution of weights for the given data.

In BCNNs, the posterior distribution over weights q(ω |D) (where D denotes the RUL of the LSS) is approximated by a variational distribution q(ω). This distribution is then compared to a predefined prior distribution p(ω) using KL divergence. The KL divergence regularization term is applied to penalize the deviation of the approximate posterior distribution from the prior distribution, thus promoting better generalization and preventing overfitting.

4.9.1 Prior Distribution (P(ω))

The weights are sampled from a Gaussian distribution with mean and variance log(1 + exp(rho)). This is a common approach in variational inference for BNNs, where the mean and variance parameters are learned during training, and the KL divergence regularization term ensures that the learned distribution stays close to the prior distribution. If the weights are consistent with an ordinary distribution, it is a common choice to use Gaussian before.

4.9.2 Likelihood Function (p(D|ω))

In Gaussian likelihood, the probability function includes calculating probability of observed data based on predicted mean and a standard deviation learnt. The likelihood function is combined with the prior distribution over the parameters and the posterior distribution to compute the posterior distribution over the given data, using Baye's rule:

$$p(\omega|D) = \frac{p(D|w).p(\omega)}{p(D)} \text{.....Eqn 4.3}$$

Where the likelihood function p(D|ω) is implicitly represented by the mean absolute error loss function used to measure the discrepancy between the model's predictions and the true labels.

4.10 BAYE'S RULE WITH KULLBACK LEIBLER

The Baye's rule over Posterior distribution is defined as below

$$P(Z|X = D) = \frac{P(X = D|Z).P(Z)}{P(X=D)} \text{ .....Eqn 4.4}$$

Where, P(Z|X=D) is the posterior,

X=D|Z is the likelihood of images

P(X=D|Z) is the joint probability

P(Z)) is the prior of latent variable

P(X=D) is the marginal data

The problem arises when P(X=D) gets intractable due to various reasons such as no defined closed form that ultimately leads to no mean, median, or sampling. To encounter such problem, surrogate posterior(q(Z)) or approximate posterior is used that is as much as good as the existing posterior.

To evaluate the fit of the surrogate distribution to the true distribution, the Kullback-Leibler (KL) divergence is used as a metric for measuring the distance between two distributions[40]. The optimization problem is to find the surrogate distribution, q*, that minimizes the KL divergence between q* and the true distribution P(Z|X=D) given the observed data D. It is impossible to find a distribution that completely overlaps each other that is, it is impossible to find zero KL divergence. So, we find best fitting surrogate distribution. And that KL divergence is calculated as the expectation of the logarithm of q* over the distribution P(Z|X=D).

$$KL(q(z)||p(z|x)) = \int q(z) log \frac{q(z)}{p(Z|X)} dz = -\int q(z) \log \frac{p(z|x)}{q(z)} dz \text{ .....Eqn 4.5}$$

Evidence Lower Bound(ELBO) is the KL divergence between surrogate and true posterior. Evidence is the logarithm of the marginal probability and ultimately the probability is always between 0 and 1. Therefore, the ELBO is smaller than the evidence.

The final goal in VI is to find the surrogate posterior (q) that minimizes the KL divergence between q and the true posterior, which is equivalent to maximizing the ELBO. The ELBO is an important finding in variational inference as it provides a computable lower bound on the log evidence, which is often intractable. The ELBO is always smaller than the evidence, but it can be equal to the evidence if and only if the surrogate posterioris the true posterior.

$$(\boldsymbol{q}(\boldsymbol{z})||\boldsymbol{p}(\boldsymbol{z}|\boldsymbol{x})) = \in_{\boldsymbol{q}} [\log \boldsymbol{q}(\boldsymbol{z}) - \log \boldsymbol{p}(\boldsymbol{x},\boldsymbol{z})] + \log \boldsymbol{p}(\boldsymbol{x}) \text{.....Eqn 4.6}$$

In ELBO, if lower bound is equal to the evidence, the true posterior has been found. However, this is not typically achieved in most cases in variational inference. Instead, the goal is to find the best fit, which is the minimum of the Kullback-Leibler (KL) divergenceor the maximum of the lower bound. So, we rearrange the equation (Eqn 4.7) for the ELBOto show that the ELBO of q is the negative KL divergence plus the logarithm of the marginal or the evidence.

$$[\log \boldsymbol{p}(\boldsymbol{x},\boldsymbol{z}) - \log \boldsymbol{q}(\boldsymbol{z})] = \log \boldsymbol{p}(\boldsymbol{x}) - \boldsymbol{KL}(\boldsymbol{q}(\boldsymbol{z})||\boldsymbol{p}(\boldsymbol{z}|\boldsymbol{x})) \text{.....Eqn 4.7}$$

The formula for ELBO is rearranged as below:

$$\boldsymbol{ELBO}(\boldsymbol{q}) = \underset{\boldsymbol{q}}{\boldsymbol{E}} [\log \boldsymbol{p}(\boldsymbol{x},\boldsymbol{z}) - \log \boldsymbol{q}(\boldsymbol{z})] = \underset{\boldsymbol{q}}{\boldsymbol{E}} [\log \frac{(\boldsymbol{x},\boldsymbol{z})}{\boldsymbol{q}(\boldsymbol{z})}] \text{.....Eqn 4.8}$$

In conclusion, VI is a technique used to approximate complex posterior distributions in Bayesian inference. A variational distribution is optimized to approximate the true posterior distribution for better optimization of surrogate posterior. ELBO is a key component of variational inference. It serves as an objective function to optimize the parameters of the variational distribution.

4.11 EVIDENCE LOWER BOUND (ELBO)

ELBO is defined as the sum of the expected log likelihood of the data under the variational distribution and the negative KL divergence between the variational distribution and the true posterior distribution. The above Eqn 4.7 can be rearranged as below [40]:

$$(\boldsymbol{q}(\boldsymbol{Z})) = -\boldsymbol{K}(\boldsymbol{q}(\boldsymbol{Z})||\boldsymbol{p}(\boldsymbol{Z}|\boldsymbol{X} = \boldsymbol{D})) + \log (\boldsymbol{p}(\boldsymbol{X} + \boldsymbol{D})) \text{.....Eqn 4.9}$$

The goal is to maximize the ELBO with respect to the parameters of the variational distribution. Maximizing the ELBO encourages the variational distribution to be close to the true posterior distribution while maximizing the likelihood of the observed data.

Maximizing the ELBO is equivalent to minimizing the KL divergence between the variational distribution and the true posterior distribution. The below pictures compare the ELBO plots of two distributions (Fig 4.7). The blue line denotes the true posterior and red line denoted Surrogate. As discussed, the maximum the ELBO, the closer the distributions are.

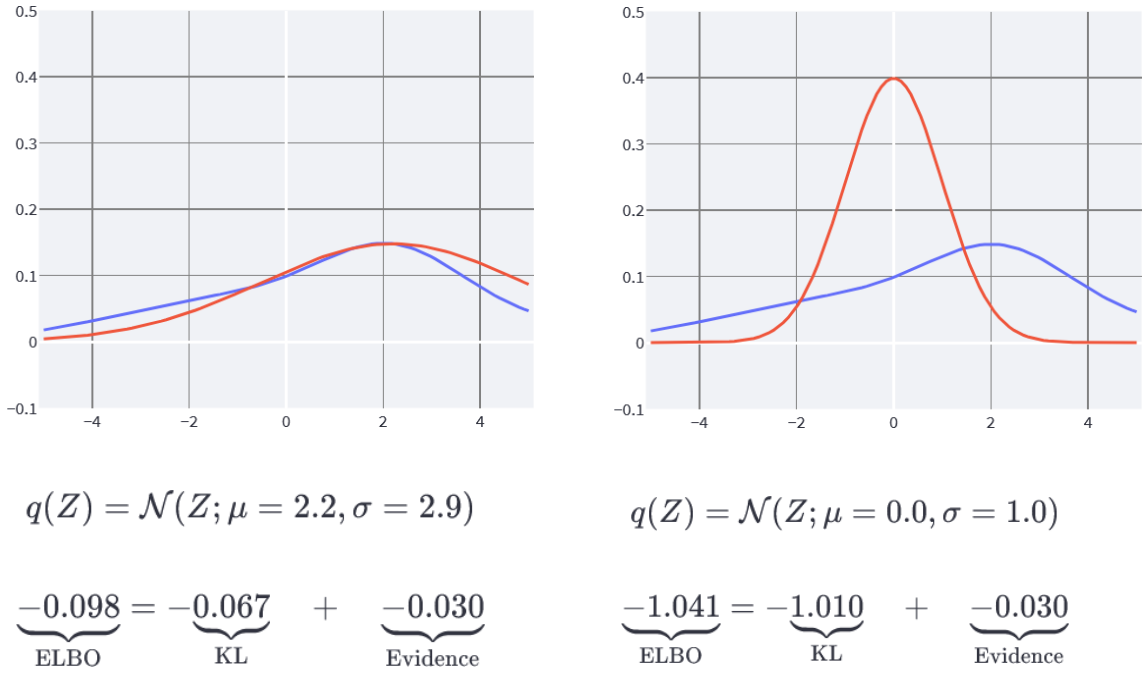


Fig 4.7 ELBO plot (Adopted from [43])

4.12 KL IN LOSS REGULARIZATION

In this project, KL divergence is used to regularize the custom loss function. It uses regularization approaches to reduce overfitting and increase model generalization. ELBO can be broken down into two terms: data likelihood and KL divergence between the approximate posterior and the prior. The KL loss regularization serves as a penalty term in the loss function to ensure that the learnt posterior distribution does not diverge significantly from the prior distribution. By doing so, the model is encouraged to find a posterior that strikes a balance between fitting the data accurately and maintaining simplicity by aligning it with the prior distribution. The KL loss regularization and the KL divergence with ELBO are related as they both serve to constrain the learned posterior distribution of the model parameters to be close to a given prior distribution. Thisconstraint is crucial for ensuring the robustness and generalization capability of Bayesian Neural Networks by preventing overfitting and promoting model simplicity.

4.13 SUMMARY

This chapter starts with the introduction of RUL and the three types of RUL estimation models available for training. The dataset is discussed along with their threshold ranges for each of the parameters. The loading and preprocessing of dataset and hierarchical clustering is explained. Furthermore, the transfer learning and its necessary BCNN architectures are discussed in detail. Lastly, the importance of KL is and its incorporation is briefed.

CHAPTER 5
IMPLEMENTATION

5.1 INTRODUCTION TO IMPLEMENTATION OF FAULT DIAGNOSTICS

Fault diagnostics are carried out through a complementary software simulation conducted using Python and MATLAB Simulink. The focus lies on identifying faults within the solenoid valves of the oxygen injection system, assessing the flow of oxygen and pressure. A multimeter and a pressure gauge are utilized to check the flow and pressure, respectively. The data and readings obtained from these instruments serve as inputs for generating graphs in Simulink or Python. The implementation involves four primary conditions:

I. ***When the valve is open:*** No fault is identified.
II. ***When the valve is open:*** A fault is identified due to low pressure.
III. ***When the valve is closed:*** A fault is identified as the valve remains unenergized.
IV. ***When the valve is closed:*** A fault is identified due to a power supply or oxygen supply being turned off.

5.2 FAULT TESTING IN SOLENOID VALVE

The fault testing in the solenoid valve can we done with the help of a multimeter. Multimeter is an instrument used to measure multiple parameters such as voltage, current and resistance. Basically, the solenoid valve will be activated by the electrical current that is supplied to it and its opening or closing shall be regulated. This multimeter that displays the voltage is used to determine the correct amount of current supply. A multimeter is set up that measures voltage and then gives a resistance value. After that, the multimeter probes shall be connected to the solenoid valve's two electrical terminals and shall be checked if the reading on the multimeter is within the specified range of the solenoid valve through the monitor. The V-I graph (Fig 5.1) shows the drop and increase in voltage .This can therefore be used to determine the current drop and increase in solenoid valves that result in valve regulation, as well as oxygen supply.

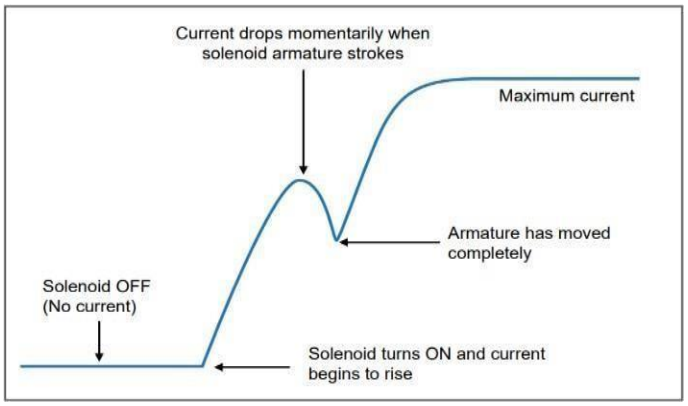


Fig 5.1 V-R Graph for Fault Testing

5.2.1 Operational Values for Fault Testing in Solenoid Valve

Common DC voltages used in solenoid valves are 6, 12, 24, 120 and 240 volts, while common AC current at 50 Hz is 110 to 220 volts according to Table 1. The circuit will close and the solenoid valve will be activated when the solenoid is connected to the power source. The diameter of the pipe, orifice size, operating pressure and current supply is considered to calculate the power supply using the formula P = I^2R and displayed in Watts(W). The power consumed by each valve is generally different because of the size of the coil and also the valve stem which is opened and closed. With the help of a multimeter, two kinds of testing shall be carried out on solenoid valves:

I. ***Resistance Testing (ohm):*** A resistance test ensures that the circuit within the solenoid valve is stable operating condition.

II. ***Voltage Testing:*** A voltage test ensures that an electric solenoid is receiving or functioning in accordance with the correct level of electrical current supplied by a power source.

Table 5.1 Operational Values of V&I

Current Flow	Voltage
Direct Current	6V to 24V DC
Alternating Current	110V to 230V AC

5.2.2 Supporting functions in python

Python's supporting functions encompass diverse functionalities, such as generating random integers with random.randrange(a, b), creating arrays using np.arange(60), and plotting line graphs with plt.plot(x, y). Additional functions like plt.title(), plt.ylabel(), plt.xlabel(), plt.xlim(), and plt.ylim() aid in customizing plot attributes such as titles, axis labels, and limits, contributing to comprehensive data visualization capabilities.

I. ***random.randrange(a, b):*** This function is from the `random` library and generates a random integer between `a` (inclusive) and `b` (exclusive).

II. ***np.arange(60):*** This function is from the `numpy` library (`np` alias) and generates an array of values from 0 to 59.

III. ***plt.plot(x, y):*** This function is from the `matplotlib.pyplot` library (`plt` alias) and is used to plot a line graph with the x-axis values from the array `x` and the y-axisvalues from the array `y`.

IV. ***plt.title("System Status for Different Input Combinations"):*** This function sets the title of the plot to "System Status for Different Input Combinations".

V. ***plt.ylabel('Status'):*** This function sets the label for the y-axis to "Status".

VI. ***plt.xlabel('Time(seconds)'):*** This function sets the label for the x-axis to "Time(seconds)".

VII. ***plt.xlim(0,60):*** This function sets the limits for the x-axis from 0 to 60.

VIII. ***plt.ylim(0,5):*** This function sets the limits for the y-axis from 0 to 5.

5.2.3 Supporting functions in MATLAB

MATLAB's supporting functions include zeros() for creating zero-filled matrices, false() for generating arrays of logical false values, and dec2bin() for converting decimal numbers to binary. Additionally, functions like plot(), xlabel(), ylabel(), title(), ylim(), yticks(), grid on, and text() aid in customizing plots, setting axes labels, limits, ticks, enabling grid lines, and adding annotations for improved visualization and analysis of data.

I. ***zeros:*** The zeros function creates a 2D matrix of zeros to store input values. In our code, inputs = zeros(9, 3) creates a 9x3 matrix to store binary input combinations.

II. ***false:*** The false function creates an array of logical false values. In our code, outputs= false(9, 1) initializes an array to store the output status as logical values.

III. ***dec2bin:*** The dec2bin function converts decimal numbers to binary format. It is used to convert the iteration value i to a binary representation in the loop.

IV. ***'0':*** The single quotes around the character 0 ('0') are used to create a string

containing the character 0. This is subtracted from the binary representation to convert binary digits to numeric values.

V. ***plot:*** The plot function is used to create the line plot. It takes the input combinations (0 to 8) on the x-axis and the system status (true/false) on the y-axis.

VI. ***X label & Y label, title:*** These functions set the labels for the x-axis, y-axis, and the title of the plot, respectively.

VII. ***Ylim & Yticks:*** These functions set the limits and ticks on the y-axis. In our code, ylim([-0.1 1.1]) sets the y-axis limits from -0.1 to 1.1, and yticks([0 1]) sets the y-axis ticks to 0 and 1.

VIII. ***grid on:*** This function turns on the grid lines on the plot for better visualization.

IX. ***text:*** The text function adds text annotations to the plot. In our code, it adds annotations at specific points on the graph to indicate fault detection messages.

5.2.4 Experimental results using Python

The Python code is a simulation over a one-minute period, generating random data for various parameters such as power supply, oxygen supply, and solenoid valve status. Firstly, the code utilizes the random module to generate random binary values (0 or 1) for the power supply, oxygen supply, and solenoid inlet valve. These valuessimulate the on/off states of these components, crucial for the system's operation. The loop iterates 60 times, corresponding to each second of a one-minute duration, capturing the system's behavior over time. Within each iteration, the code checks the status of the power supply and oxygen supply. If both are on 1, it proceeds to check the status of the solenoid inlet valve.The status of these components determines whether the system proceeds with further checks and actions. The code simulates the use of instruments like a multimeter and a pressure gauge.If the power supply and oxygen supply are on, it assumes these instruments are active. The pressure threshold is set at 10 bar, defining a critical level for system operation. If the pressure exceeds this threshold, the outlet valve is turned on, indicating normal system operation. Conversely, if the pressure is below the threshold, a low-pressure fault is detected, and the code flags this as a fault condition. Additionally, the code checks the status of the solenoid inlet valve. If this valve is not energized while the power supply and oxygen supply are on, it indicates a fault condition related to the solenoid valve. Throughout these checks, the code updates a list (y value) with binary values (0 or 1) representing fault detection status for each second of the simulation. A value of 0 indicates no fault detected, while a value of 1 indicates a fault was detected during that second. The

resulting graph plots (Fig 5.2 & Fig 5.3) these fault detection statuses againsttime, with the x-axis representing time in seconds (from 0 to 59) and the y-axis representing the fault detection status (0 for no fault, 1 for fault). Each point on the graph corresponds to a second of the simulation, showing how the system's fault detection status changes over time. The graph's title, "Fault detection analysis," reflects the varying combinations of input parameters (power supply, oxygen supply, solenoid valve status) that influence the system's behavior and fault detection. The axis labels, "Time(seconds)" for the x-axis and "Status" for the y-axis, provide context for interpreting the graph's data.

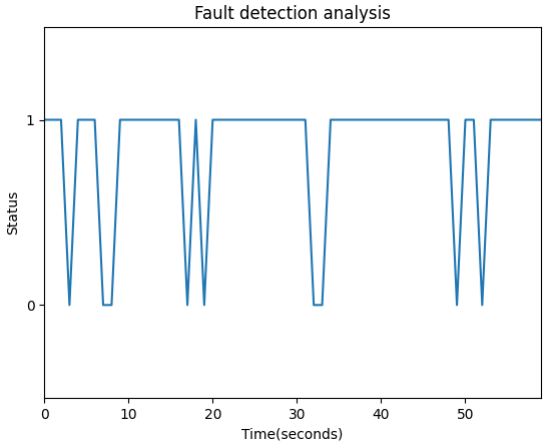


Fig 5.2 Fault Detection Graph using Py (60sec)

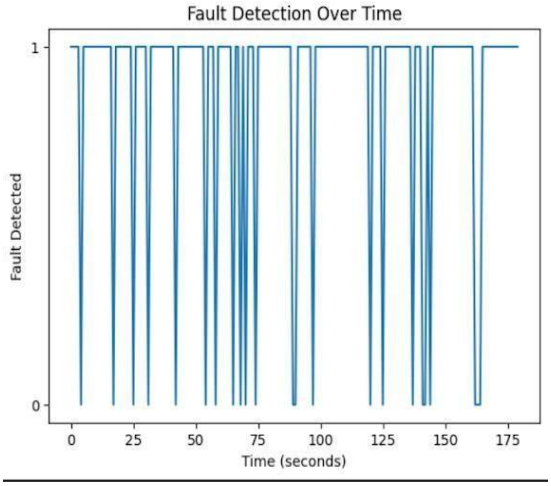


Fig 5.3 Fault Detection Graph using Py(180sec)

5.2.5 Experimental results using MATLAB

The graph generated by the MATLAB code (Fig 5.4) displays the system's status (fault detection) against different input combinations. The x-axis of the graph represents the input combinations, ranging from 0 to 8, where each number corresponds to a unique combination of power supply, oxygen supply, and solenoid valve status. The y-axis represents the system's status, specificallywhether a fault is detected or not. The graph uses blue circles connected by lines ('bo-') to plot the system's status for each input combination. A blue circle represents a detected fault, while a blank space between circles indicates no fault detected. The graph's title is "Fault Detection Graph," indicating its purpose of visualizing fault detection based on input conditions. The x-axis label is "Time (seconds)," which in this context represents different input combinations rather than actual time. The y-axis label is "System Status," denoting whether the system is functioning correctly (no fault) or has encountered a fault. The y-axis limits are set between -0.1 and 1.1, ensuring that only two distinct values (0 and 1) are displayed, corresponding to the absence or presence of a fault. This setup aligns with your request to show only binary system status on the graph without in-between values. Overall, the graph effectively visualizes how different input combinations impact the system's status, highlighting fault detection scenarios and normal operation instances. It provides a clear and concise representation of the system's behavior under varying conditions, aiding in understanding fault patterns and system performance.

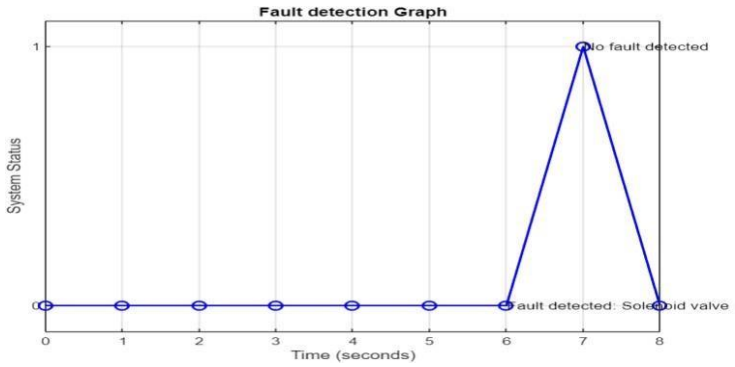


Fig 5.4 Fault Detection Graph using MATLAB (8sec)

5.3 INTRODUCTION OF REMAINING USEFUL LIFE

RUL is the number of hours a machine would operate before it requires repair or replacement. It helps to identify the total working hours of the machine, operating efficiency, and unplanned downfall duration. Finding out the remaining useful life for a LSS is very important so that in case of any system failures, the appropriate decision would be taken further whether to continue the mission or terminate them thus increasing the reliability of the mission. The parameters considered for the dataset are of two types i.e. Ambient& Human parameters. For Healthy State, Defective Realm, and End of Life of the sphere values should be in a specified range.

5.3.1 Description of Healthy State

In the healthy state, the system operates optimally, demonstrating robust performance and efficiency, with parameter values consistently within predefined ranges. These stable parameters ensure smooth mission operations and reliable execution. Below are the defined parameters for the healthy state of the system:

I. ***Body Pressure –*** The body pressure for healthy state should be maintained between 60 mmHg to 120 mmHg.

II. ***Body Temperature –*** Temperature of human body should be maintained regularly between 35.60° C and 37.10° C

III. ***SpO_2 -*** Oxygen saturation is percent of oxygen carried by blood. For a healthy individual, the normal SpO2 should be between 95% to 100%.

IV. ***Heart Rate -*** The normal heart rate for adults is typically between 60 and 100 bpm.

V. Absolute Pressure - The total pressure exerted by a fluid should be between 1 to 100 Pa for healthy state.

VI. ***Absolute Temperature -*** Temperature measured on an absolute scale where zero represents the absence of thermal energy. Such temperature is preferred to maintain between 29.330° C to 33.520° C

VII. ***Humidity -*** Ideally, the relative humidity inside a submarine should be maintained between 79% to 87.67%

VIII. ***Oxygen Levels -*** The oxygen levels have to be maintained regularly inside sphere so as for the crew members to carry out the mission successfully and sustain themselves. O_2 needed per person during the operation is averaged out to be 26.9L/hr to 32.9 L/hr.

5.3.2 Description of Defective Realm

When the system enters the defective realm, it shows deflections in its parameters. These deviations act as early warnings for possible faults or worsening conditions, prompting actions to safeguard the mission. Estimating the RUL becomes crucial here as it determines whether to proceed with the mission or retreat, ensuring safety and mission success.

I. ***Body Pressure –*** The upper limit of bp is between 130 mmHg to 140 mmHg and lower limit is about 40 to 50 mmHg.

II. ***Body Temperature –*** The upper boundary temperature is between 38° C and 40° C and lower boundary is about 33° C to 35° C.

III. ***SpO_2 -*** The percent of oxygen carried by blood at moderate risk is between 105% to 115% and 85% to 95%.

IV. ***Heart Rate -*** On the borderline, the heart rate for adults is typically between 105 to 120 bpm and 40 to 55 bpm.

V. ***Absolute Pressure -*** The total pressure exerted by a fluid should fall within the range of 200 to 300 for defective realm.

VI. ***Absolute Temperature -*** Absolute temperature in this case is usually between 34.5°C to 36.5° C and 25° C to 28° C.

VII. ***Humidity –*** In defective state, the relative humidity inside a submarine is between 89% to 94% and 73 to 77%.

VIII. ***Oxygen Levels -*** The oxygen levels have to be maintained inside the sphere will be increased to 34 L/hr to 38 L/hr or decreases to 21 to 25L/hr.

5.3.3 Description of End of Life (EOL)

The end of life (EOL) is when the machine reaches the end of its working life. At this stage, it shows significant wear and its performance drops below acceptable levels. When the EOL is reached, it usually means stopping the mission and returning to the surface for repair or replacement. This ensures safety and reliability for future operations.

I. ***Body Pressure –*** At the very danger, the body pressure increases to 140 to 200 mmHg or decreases to 10 to 30mmHg.

II. ***Body Temperature –*** Temperature of human body would be between 41° C to 45°C or 27° C to 31° C.

III. ***SpO_2*** – The amount of SpO2 is between 120% to 140% or 60% to 80%.

IV. ***Heart Rate –*** When EOL is approached, according to ambient changes, the abnormal heart rate usually ranges between 125 to 140bpm or 30 to 35 bpm.

V. ***Absolute Pressure*** **-** The total pressure increases as depth increases, so the absolute pressure is between 300 to 400 Pa.

VI. ***Absolute Temperature*** **–** The temperate usually fluctuates in depth and is between 20° C to 24° C or 37.5° C to 40° C.

VII. ***Humidity*** **–** The humidity inside the sphere is between 96 to 105% and 65 to 71%.

VIII. ***Oxygen Levels*** **–** At the danger state, with fault in SV, the oxygen might fluctuate at the rates of 40 to 45 L/hr and 15 to 25 L/hr at EOL.

5.4 IMPLEMENTATION OF RUL

In this implementation (Fig 5.5), we make use of two powerful methodologies: BayesianCNNs and transfer learning. The flowchart of RUL computation is given below.

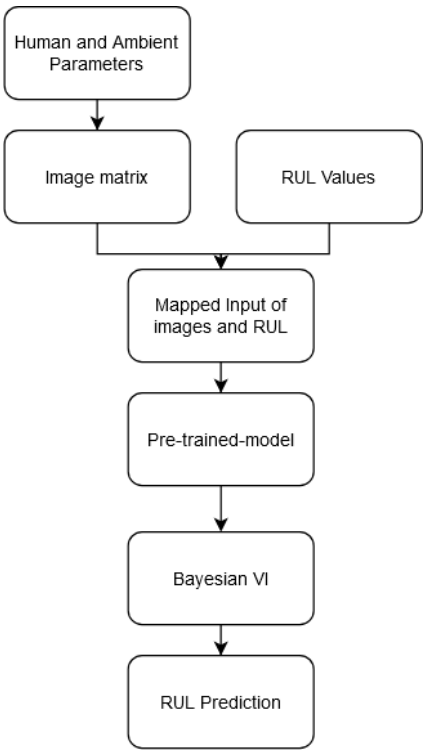


Fig 5.5 Flowchart for computing RUL

We aim to build and use the pre-trained Bayesian CNN to a new task by fine-tuning its parameters using Bayesian Variational Inference, making use of the learned features from the original task. The following steps are incorporated in BCNN:

I. Parameters to image matrix

Initially, the data is read into a dataframe, which contains human and ambient parameter values. Subsequently, for each row in the dataframe, an intensity plot is generated using matplotlib. This plot represents the intensity with the x-axis which denotes the row number and the y-axis that denotes a fixed range of the intensity for respective values. The generated plots are saved as a PNG image file with a unique index identifier (Fig 5.6).

```
import pandas as pd
from sklearn.model_selection import train_test_split


data = pd.read_excel(r"C:\Users\Sindhu\mapping_output.xlsx")

data_test = pd.read_excel(r"C:\Users\Sindhu\mapping_output_test.xlsx")
```

Fig 5.6 File path for BCNN &pretrained model

II. Mapping images and RUL

Each of these images are then mapped with their respective RUL values by iterating over the images and their corresponding RUL values respectively. This iteration is done by synchronizing time-based image filenames with RUL data.

III. Pre-Processing the data

For each image, the image is being loaded and resized to a specified target size of 100 x 100 pixels (Fig 5.7), and then converts it into a numerical array representation using NumPy. This array is then normalized by dividing the pixel values by 225 to make the pixel values range in between 0 and 1 for faster data processing. To further speed up the computations, these images are being transformed to gray-scale images for efficient and speedy processing.

```
def load_and_preprocess_images(image_paths):
    images = []
    for path in image_paths:
        img = load_img(path, target_size=(100,100))
        img_array = img_to_array(img) / 255.0
        images.append(img_array)
    return np.array(images)
```

Fig 5.7 Loading and preprocessing images

IV. Bayesian CNN Architecture - Pre-trained model

This BCNN model processes and analyzes the image data for regression tasks. The architecture consists of convolutional, dense, dropout, and output layers, in which each of these layers is used in extracting features and making predictions. For the foremost convolutional later, with number of kernels being set as 32, that has 3 x3 window size along with ReLU activation function, it serves to extract features from input images. Then we use flatten layer to transforms the multi-dimensional output from the previous convolutional layer into a 1-D array, preparing the data for input into the further dense layers present in the model. Then this pre-trained model has a dense layer of 128 units and ReLU activation. Then dropout regularization technique is used with dropout rate about 0.5 so that 50% of the units are set to 0 to prevent over fitting. The final layer is the output layer with a single output, that gives out the final regression prediction. This is the pre-trained model architecture (Fig 5.8).

```
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import load_img, img_to_array

class BayesianCNN(tf.keras.Model):
    def __init__(self):
        super(BayesianCNN, self).__init__()
        self.conv1 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu', groups=1)
        self.flatten = tf.keras.layers.Flatten()
        self.dense1 = tf.keras.layers.Dense(128, activation='relu')
        self.dropout = tf.keras.layers.Dropout(0.5)
        self.dense2 = tf.keras.layers.Dense(1)

    def call(self, inputs):
        x = self.conv1(inputs)
        x = self.flatten(x)
        x = self.dense1(x)
        x = self.dropout(x)
        return self.dense2(x)
```

Fig 5.8 Architecture for BCNN model

5.4.1 Incorporating Bayesian Dense VI Layer into a BCNN

This section will discuss how to incorporate a Bayesian Dense VI[42] layer into a BCNN. This is done by the layer that introduces uncertainty into the model's weights by representing them as probabilistic distributions instead of fixed values. Within this layer, two sets of parameters, 'mean' and 'rho'(Fig 5.9), are initialized. The 'mean' represents the mean values of the weights, while 'rho' captures the uncertainty or variance associated with these weights.

During the forward pass, the model adds a bit of randomness (ε or epsilon) to its weights. This randomness comes from a normal distribution, which is like a bell- shaped curve of possible values. This allows the model to estimate the uncertainty in predictions and confidence level. `

```
class BayesianDenseVI(Layer):
    def __init__(self, units, activation='relu'):
        super(BayesianDenseVI, self).__init__()
        self.units = units
        self.activation = tf.keras.activations.get(activation)
        self.batch_norm = BatchNormalization()

    def build(self, input_shape):
        self.mean = self.add_weight(shape=(input_shape[-1], self.units),
                                    initializer='random_normal',
                                    trainable=True,
                                    name='mean')
        self.rho = self.add_weight(shape=(input_shape[-1], self.units),
                                   initializer='zeros',
                                   trainable=True,
                                   name='rho')

    def call(self, inputs):
        epsilon = tf.random.normal(shape=tf.shape(self.mean))
        weights = self.mean + tf.math.log(1 + tf.exp(self.rho)) * epsilon
        x = tf.matmul(inputs, weights)
        x = self.batch_norm(x)
```

Fig 5.9 Architecture for Target model

To leverage this Bayesian approach effectively, load a pre-trained BCNN model which is previously saved using the 'load_model' function (Fig 5.10) from TensorFlow Keras. This pre-trained model represents a previously trained Bayesian CNN model that has learnt features and representations captured during the original training process, to improve computational efficiency.

```
import tensorflow as tf

loaded_model = tf.keras.models.load_model('bcnn_model_save')
```

Fig 5.10 Loading saved BCNN model

Freezing specific layers ensures they remain unaltered during subsequent training on new data. This helps to retain the knowledge learnt by the pre-trained model and prevents the forgetfulness of the model, where the model forgets previously learned features while learning new ones. Replace the last layer of the pre-trained model with a Bayesian Dense VI layer and this integration introduces Bayesian uncertainty estimation into the pre-trained model. Model uncertainty through the computation of both prior and posterior distributions

over layer weights using variational inference. Consequently, define a function to calculate the KL divergence regularization term for the Bayesian Dense VI layer (Fig 5.11). This term enables the learned weights to approximate a prior distribution using Gaussian. Additionally, define a custom loss function that combines the original loss which is the mean absolute error between the true and predicted values, with the KL divergence regularization term with the model (Fig 5.12). The regularization term penalizes deviations of learned weights from the prior distribution.

```
def kl_divergence_regularizer(model):
    kl_loss = sum(model.losses)
    return kl_loss
```

Fig 5.11 Function to calculate KL divergence

```
def custom_loss(y_true, y_pred):
    original_loss = tf.keras.losses.mean_absolute_error(y_true, y_pred)
    kl_loss = kl_divergence_regularizer(pre_trained_model)
    return original_loss + kl_loss
```

Fig 5.12 Function to define custom loss

Then, compile the target model using the Adam optimizer with an even lower learning rate of 0.0001 for fine-tuning. Use the custom loss function and monitor MAE loss metrics. Fit the target model on the new training data and train for up to 50 epochs with early stopping to prevent overfitting (Fig 5.13). This monitors the performance on a validation dataset and stops the training process if the performance fails to improve for a specified number of consecutive epochs. BCNN with VI provide not only predictions but also quantify the associated uncertainty, therefore enhancing the robustness and reliability of the predictions.

```
early_stopping = tf.keras.callbacks.EarlyStopping(patience=5, restore_best_weights=True)

# Train the model
history_transfer_learning = target_model.fit(X_new_processed, y_new,
                                             batch_size=16,
                                             epochs=10,
                                             validation_data=(X_new_processed, y_new),
                                             callbacks=[early_stopping])
```

Fig 5.13 Trained model using early stopping

After, training the target model, the saved model can be used for prediction of RUL (Fig 5.14). The predicted RUL value is then rounded to one decimal place and converted from a decimal representation to a time format in hours and minutes.

```
input_image = np.expand_dims(img_array, axis=0)

prediction = pre_trained_model.predict(input_image)

pred = round(prediction[0][0], 1)

def convert_decimal_to_time(decimal_value):
    hours = int(decimal_value)
    minutes = int((decimal_value - hours) * 60)
    return f"{hours} hrs and {minutes:02d} mins"

time_string = convert_decimal_to_time(pred)

print(f"RUL : {time_string} left")
```

Fig 5.14 RUL final prediction

5.4.2 Compilation and Training of the BCNN model

Compilation refers to the configuration of the learning process. When a model is compiled, the loss function, the optimizer, and the metrics that the model should use during training are specified. Training is the process of fitting the model to the training data. During training, the model adjusts its weights based on the optimization algorithm and the loss function to minimize the difference between the predicted RUL and the actual RUL. The performance metrics to assess the BCNN model is given below:

A. ***Mean Square Error (MSE):*** It is a common metric used in regression problems to assess the performance of a predictive model. Mathematically, MSE is expressed as the average of the squared differences between predicted and actual values. Here's the formula [43]:

$$\boldsymbol{MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \quad \textbf{.....Eqn 5.1}$$

Where:

n is the number of samples in the dataset.

y_i is the true value of the target variable for the i-th sample.

$\hat{y}$ is the predicted value of the target variable for the i-th sample.

The evaluation of two metrics namely MSE with Stochastic CNN and MSE with BCNN are compared below:

- ***MSE with stochastic CNN:***

  In a Stochastic CNN with Monte Carlo Dropout (MCDropout), MSE was initially used as a loss function to train the model. The stochastic nature of MCDropout introduces randomness, enabling the model to estimate uncertainty and optimize towards minimizing the MSE. However, the high variability in the loss made it less suitable. A visual representation(Fig 5.15) of an epoch in the regression neural network training process, demonstrating the progression of MSE loss over multiple iterations. To address this, another approach is employed to capture better uncertainty and optimize the model's performance.

```
Epoch 1/10
180/180 [==============================] - 41s 215ms/step - loss: 5831778171044510543380 48.0000 - accuracy: 0.0014
Epoch 2/10
180/180 [==============================] - 38s 211ms/step - loss: 4047512704388547215 36.0000 - accuracy: 0.0014
Epoch 3/10
180/180 [==============================] - 39s 215ms/step - loss: 280915479910940672.0000 - accuracy: 0.0014
Epoch 4/10
180/180 [==============================] - 39s 217ms/step - loss: 194967725146112.0000 - accuracy: 0.0014
Epoch 5/10
180/180 [==============================] - 39s 217ms/step - loss: 135316201472.0000 - accuracy: 0.0014
Epoch 6/10
180/180 [==============================] - 39s 219ms/step - loss: 93915216.0000 - accuracy: 0.0014
Epoch 7/10
180/180 [==============================] - 39s 217ms/step - loss: 65180.2461 - accuracy: 0.0014
Epoch 8/10
180/180 [==============================] - 40s 219ms/step - loss: 51.8937 - accuracy: 0.0014
Epoch 9/10
180/180 [==============================] - 39s 215ms/step - loss: 6.8616 - accuracy: 0.0014
Epoch 10/10
180/180 [==============================] - 38s 213ms/step - loss: 6.8313 - accuracy: 0.0014
```

Fig 5.15 Training Epoch for stochastic CNN +MSE

- ***MSE with BCNN:***

  Continuing from the use of a Stochastic CNN with MSE, Bayesian CNN with MSE (Fig 5.16) was employed to better handle the uncertainty estimation. While the loss improved, it did not decrease significantly. So that, we need for further optimization for exploring different loss functions to better leverage the Bayesian framework and improve the model's performance.

```
Epoch 1/10
144/144 [==============================] - 98s 299ms/step - loss: 25.1288 - mse: 25.1288 - val_loss: 3.2606 - val_mse: 3.2606
Epoch 2/10
144/144 [==============================] - 43s 298ms/step - loss: 6.3435 - mse: 6.3435 - val_loss: 2.5490 - val_mse: 2.5490
Epoch 3/10
144/144 [==============================] - 43s 297ms/step - loss: 6.0951 - mse: 6.0951 - val_loss: 2.6409 - val_mse: 2.6409
Epoch 4/10
144/144 [==============================] - 43s 297ms/step - loss: 6.0231 - mse: 6.0231 - val_loss: 1.8956 - val_mse: 1.8956
Epoch 5/10
144/144 [==============================] - 43s 297ms/step - loss: 6.0021 - mse: 6.0021 - val_loss: 2.5387 - val_mse: 2.5387
Epoch 6/10
144/144 [==============================] - 43s 298ms/step - loss: 5.7366 - mse: 5.7366 - val_loss: 2.1936 - val_mse: 2.1936
Epoch 7/10
144/144 [==============================] - 43s 301ms/step - loss: 5.5222 - mse: 5.5222 - val_loss: 2.4657 - val_mse: 2.4657
Epoch 8/10
144/144 [==============================] - 43s 298ms/step - loss: 5.8763 - mse: 5.8763 - val_loss: 2.7271 - val_mse: 2.7271
Epoch 9/10
144/144 [==============================] - 43s 297ms/step - loss: 5.5548 - mse: 5.5548 - val_loss: 1.9433 - val_mse: 1.9433
Epoch 10/10
144/144 [==============================] - 43s 296ms/step - loss: 5.4814 - mse: 5.4814 - val_loss: 2.4343 - val_mse: 2.4343
```

Fig 5.16 Training Epochs for BCNN +MSE

B. ***Mean Absolute Error (MAE):*** MAE is calculated by averaging the absolute differences between predicted and actual values. The formula is mentioned below[43]:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \quad \text{.....Eqn 5.2}$$

Where:

n is the number of samples in the dataset.

y_i is the true value of the target variable for the *i*-th sample.

$\hat{y}_i$ is the predicted value of the target variable for the *i*-th sample.

The evaluation of MAE metrics with BCNN is given below:

- ***BCNN with MAE:***

After experimenting with various approaches, the BCNN with MAE as the loss function proved to be the most effective (Fig 5.17), leading to minimal loss and superior performance in the task.

```
Epoch 2/9
144/144 [==============================] - 43s 298ms/step - loss: 1.9695 - mae: 1.9695 - val_loss: 1.3384 - val_mae: 1.3384
Epoch 3/9
144/144 [==============================] - 42s 295ms/step - loss: 1.8780 - mae: 1.8780 - val_loss: 1.1551 - val_mae: 1.1551
Epoch 4/9
144/144 [==============================] - 43s 296ms/step - loss: 1.7543 - mae: 1.7543 - val_loss: 1.2713 - val_mae: 1.2713
Epoch 5/9
144/144 [==============================] - 43s 296ms/step - loss: 1.7081 - mae: 1.7081 - val_loss: 1.2243 - val_mae: 1.2243
Epoch 6/9
144/144 [==============================] - 42s 291ms/step - loss: 1.6353 - mae: 1.6353 - val_loss: 1.1197 - val_mae: 1.1197
Epoch 7/9
144/144 [==============================] - 43s 297ms/step - loss: 1.5846 - mae: 1.5846 - val_loss: 1.0934 - val_mae: 1.0934
Epoch 8/9
144/144 [==============================] - 42s 295ms/step - loss: 1.4962 - mae: 1.4962 - val_loss: 1.1838 - val_mae: 1.1838
Epoch 9/9
144/144 [==============================] - 44s 304ms/step - loss: 1.5325 - mae: 1.5325 - val_loss: 1.1020 - val_mae: 1.1020
```

Fig 5.17 Training Epoch for BCNN +MAE

5.5 SUMMARY

This section discusses the implementation of fault identification procedures in solenoid valves using MATLAB Simulink and Python Collaboratory. Graphical outputs are produced using Python for durations of 60 and 180 seconds, showcasing parameters such as power supply, oxygen supply, and solenoid valve status. MATLAB Simulink generates graphs spanning 8 seconds, illustrating scenarios of fault detection alongside normal operational states under different solenoid valve conditions. Once these faults areidentified, data is obtained for RUL prediction. A Bayesian CNN model is employed for regression with 32 kernels, dense layers, and dropout regularization. Training uses Adam optimizer with MAE loss. A Bayesian Dense VI layer is added to the pre-trained model,which is fine-tuned with a custom loss combining MAE and KL divergence. The final model predicts the RUL in hours and minutes after training.

CHAPTER 6

RESULTS

6.1 RESULTS OF FAULT DIAGNOSTICS IN SOLENOID VALVES

The fault analysis of solenoid valve is done using MATLAB and python. In MATLAB, the result analysis involves defining input combinations and system statuses in binary format, stored in a 2D matrix and an array respectively. Through iterative processes, all possible combinations are generated and evaluated during simulation, determining the system's operational status based on conditions like power and oxygen supply. Subsequently, a customized graph is plotted, depicting input combinations against system status, with labels, title, limits, ticks, and grid for clarity. In this graph, a status of 0 represents a fault detected, while a status of 1 indicates the system is working without fault. Conversely, Python's analysis simulates system states using random numbers and tracks fault detection performance over time. By assessing conditions and updating fault detection variables, a line graph is generated with time on the x-axis and fault detection status on the y-axis, showcasing instances of fault detection throughout the simulation. Both approaches offer insights into system behavior and fault detection, utilizing distinct methodologies and visualization techniques for different time intervals.

6.1.1 Result analysis using MATLAB

The code defines input combinations as binary values representing power supply, oxygen supply, and solenoid inlet valve status. These combinations are stored in a 2D matrix, while the system's statuses, indicating whether it's working or not, are stored in an array.To generate input combinations, the code iterates from 0 to 8, representing all possible combinations in binary. It converts each iteration value to binary using the `dec2bin` function and subtracts '0' to get numeric values. This process ensures that all possible input combinations are generated for the system. During simulation, the code evaluates the behavior of the system for each input combination. It checks if the power supply and oxygen supply are ON and if the solenoid inlet valve is energized. Based on these conditions, it determines whether the system is working (outputs(i+1)= true) or not working (outputs(i+1) = false).After simulating the system for all input combinations, graph is plotted usingMATLAB's plotting functions. The x-axis of the graph represents the input combinations (0 to 8) while the y-axis represents the systems status where 0 for not working i.e. fault detected and 1 is working i.e. no fault detected. Furthermore, the plot (Fig

6.1) is customized with labels for the axes (`xlabel`, `ylabel`), a title (`title`), y-axis limits (`ylim`), y-axis ticks (`yticks`), and a grid (`grid on`).

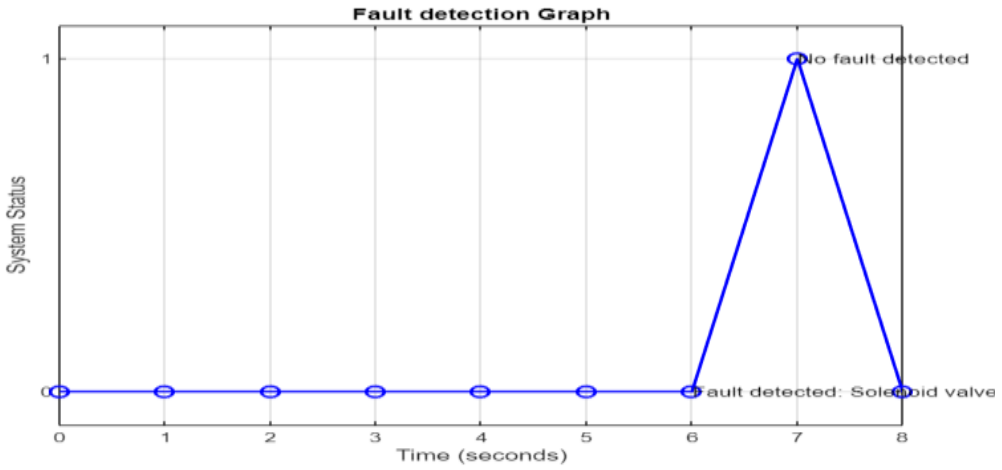


Fig 6.1 Fault Detection Graph using MATLAB(8sec)

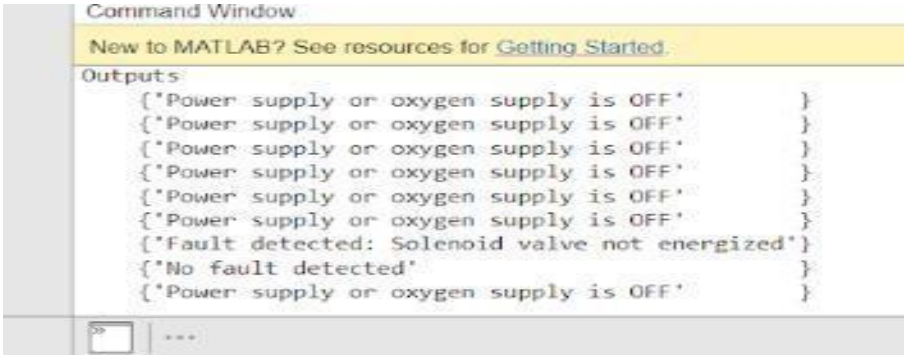


Fig 6.2 Fault Detection Output using MATLAB

6.1.2 Result analysis using python

The code simulates various system states such as power supply, oxygen supply, and solenoid inlet valve status using random numbers (random.randrange(0,2)). These states are crucial for determining the fault detection process within the system. The yvalue list collects the values of fault_detected over time (60/180 iterations) and this data is essential for analyzing the system's fault detection performance (Fig 6.3 & Fig 6.4) across multiple time points. If conditions are met (e.g., pressure is above a threshold), the code sets fault_detected to 0, indicating no fault. Otherwise, it sets fault_detected to 1, indicating a fault detected.The fault_detected variable keeps track of whether a fault has been detected during each iteration of the simulation. A line graph using Matplotlib is generated , where the x-axis represents time (60 seconds/180seconds) and the y-axis

represents the fault detection status (0 for no fault, 1 for fault detected). A horizontal line at 0 indicates no faults detected, while spikes at 1 indicate instances of fault detection during the simulation(Fig 6.6 & Fig 6.5).

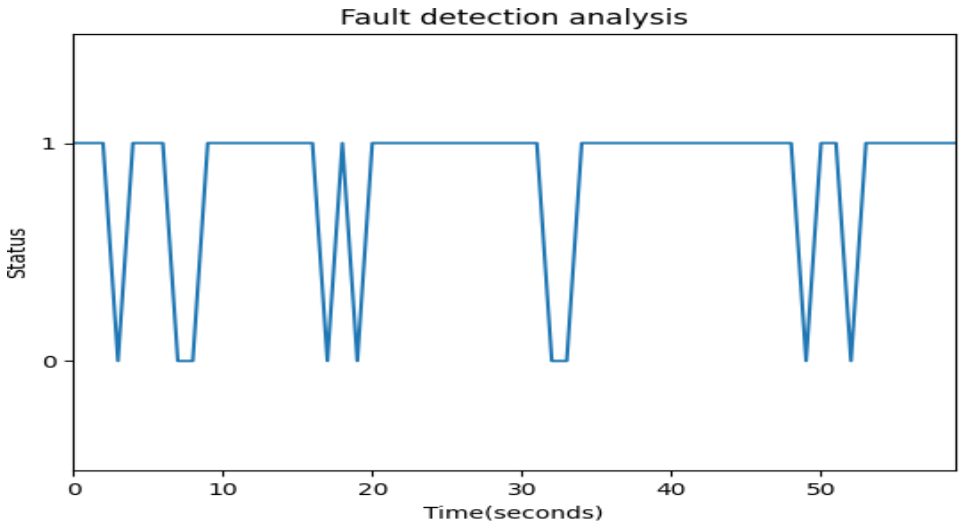


Fig 6.3 Fault Detection Graph using Py(60sec)

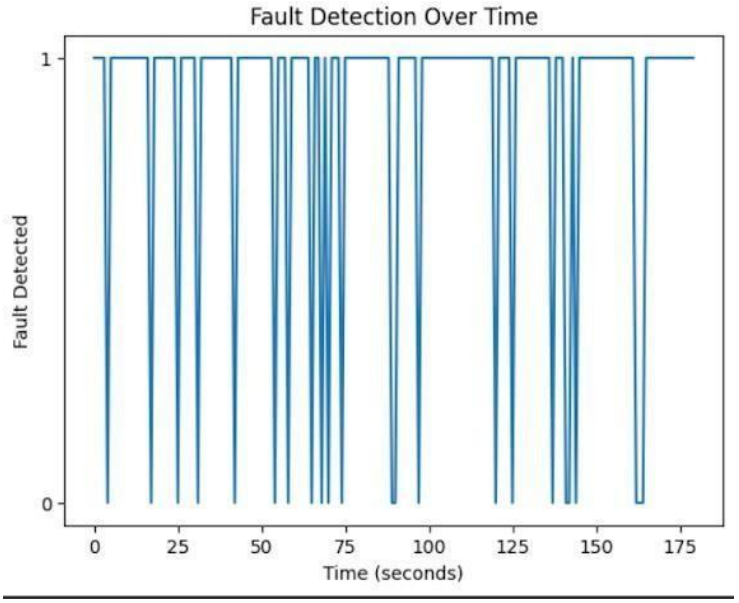


Fig 6.4 Fault Detection Graph using Py(180sec)

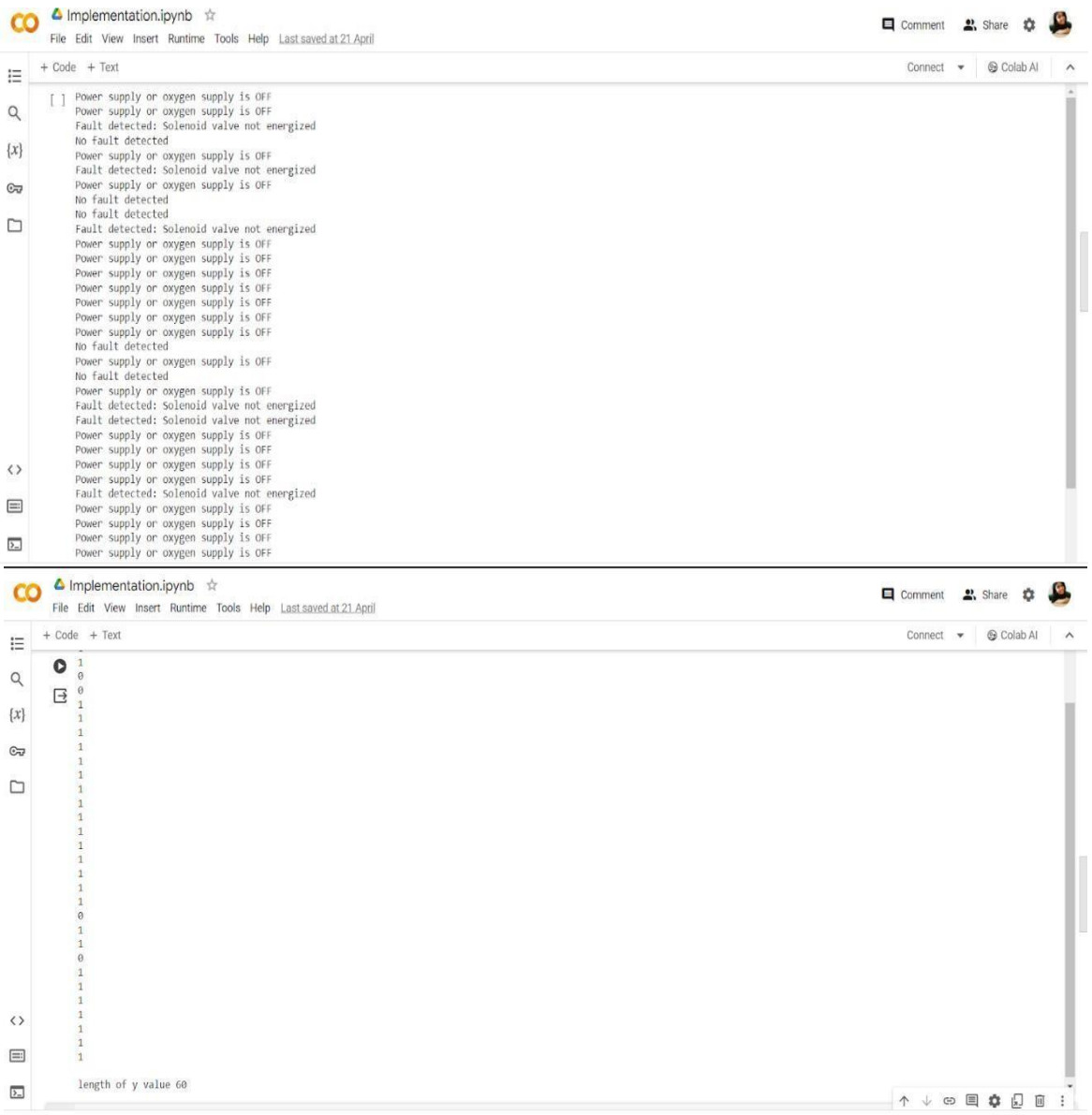


Fig 6.5 Output Screenshot using py (60sec)

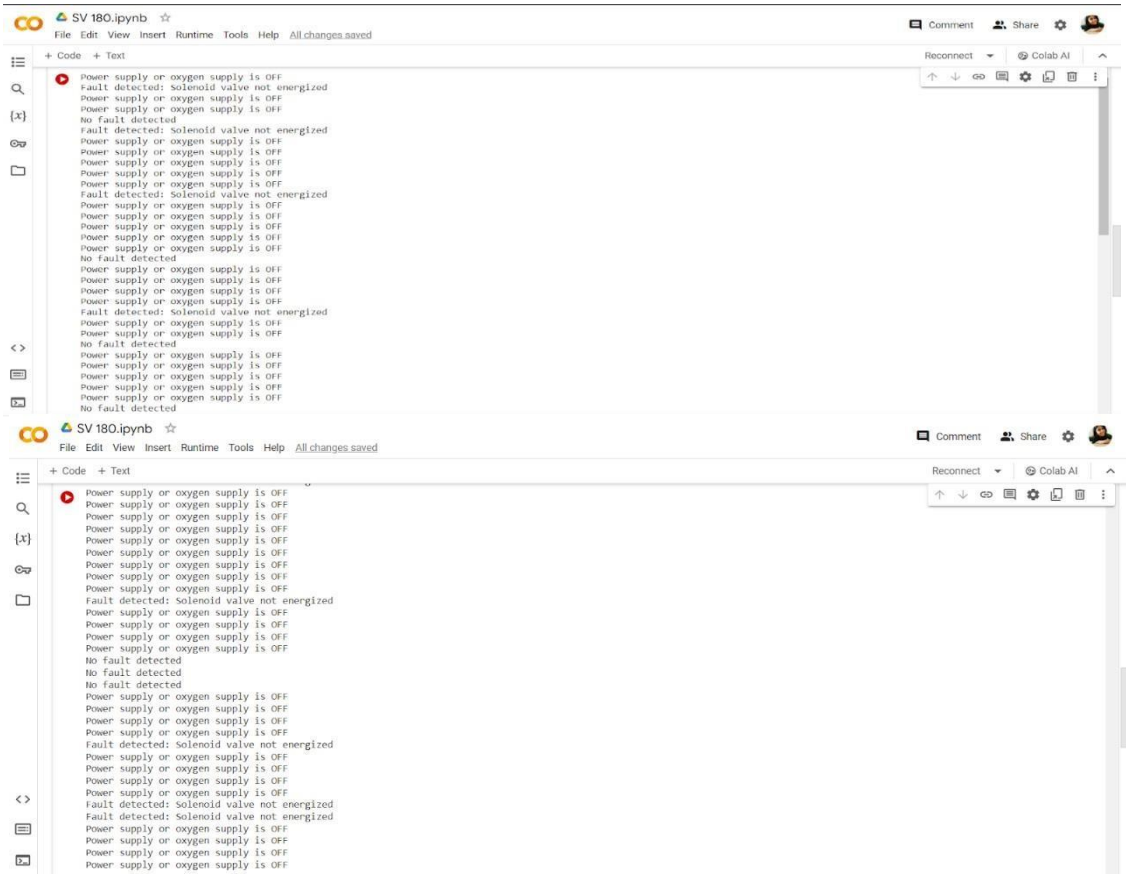


Fig 6.6 Output Screenshot using py (180sec)

6.2 RESULTS OF ESTIMATION OF RUL

The results obtained from the estimation of RUL provide valuable insights into the performance and behavior of the models employed in this study. Through various analyses and visualizations, we gain a comprehensive understanding of how each model learns, generalizes, and predicts RUL.

6.2.1 Loss Prediction

The loss graphs for both models provide useful information about their training methods and performance (Fig 6.7). The MAE loss for the Bayesian CNN model decreases steadily during the first epochs, indicating that the training data is being learned effectively. However, as the epochs progress, the validation MAE fluctuates indicating potential overfitting or problems with generalization. On the other hand, the transfer learning model with Bayesian VI layer shows a different pattern. The MAE loss exhibits an initial decrease (Fig 6.7) , similar to the pre trained model. Still, with the addition of the VI layer and fine-tuning of the pre-trained model, the model is able to better leverage the learned features, potentially leading to more stable validation MAE values than the pre trained model.

Furthermore, the inclusion of the Kullback-Leibler (KL) divergence loss component in the Transfer Learning model results in a regularization effect. This KL loss component aims to bring the Bayesian weights closer to a prior distribution, which improves the model's uncertainty quantification and robustness. In conclusion, while both models show a decrease in MAE loss over epochs, the Transfer Learning model's integration of the VI layer and KL divergence regularization shows a potentially more stable learning process with improved uncertainty quantification.

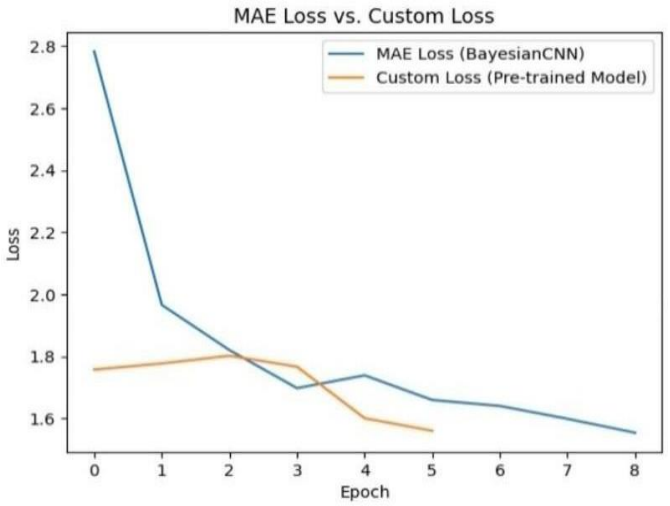


Fig 6.7 MAE vs Custom Loss

6.2.2 Histogram plots

The plot aims to visualize the distribution of weights within each layer of the Bayesian CNN model. Analyzing weight distributions can provide insights into weight sparsity, magnitude and potential issues of over fitting or underfitting. The x-axis weight values are the individual weight values associated with a specific parameter (Parameter 1 or Parameter 2) within a Bayesian convolutional neural network layer. These values provide insights into the distribution of weights within each layer of the neural network, which can be helpful for understanding the learning dynamics of the model. Parameter 1 refers to first set consisting of the kernel weights. These are the weights that are convolved with the input data to produce the output feature maps. Parameter 2 refer to different sets of weights associated with set consisting of the bias terms. Each filter typically has its own bias term. The density values on the histogram's y-axis represent the estimated probabilities of observing weight values in each bin on the x-axis.

They are numerical representations of the likelihood of observing different weight values that aid in visualizing the weight distribution within each layer of the neural network.

Each of these images (Fig 6.8) represent the layers present in BCNN, the first layer being convolution layer, second is the dense layer 1 and lastly is the dense layer 2. For each of these layers, the weight distribution is interpreted. In the first layer, the weight values range from -0.2 to 0.1, indicating that the weights in this layer are relatively small in magnitude.This suggests that the model is designed to have a certain level of sparsity or regularization to prevent overfitting. The density of the weight distribution ranges from 0 to 5, so that it implies that the weights are somewhat concentrated around specific values, rather than being uniformly spread out. The bias values range from 0.001 to 0.1so that the model is designed to have a low bias effect. The density of the bias distribution ranges from 0 to 175, which is significantly higher than the weight density. This implies that there might be a larger number of unique bias values or that certain bias values are highly repetitive compared to the weights. These are the inferences observes from the first layer of BCNN. The small magnitude of weights and biases suggests that the model employs regularization techniques to prevent overfitting. Likewise, the inferences are observed for each of these layers.

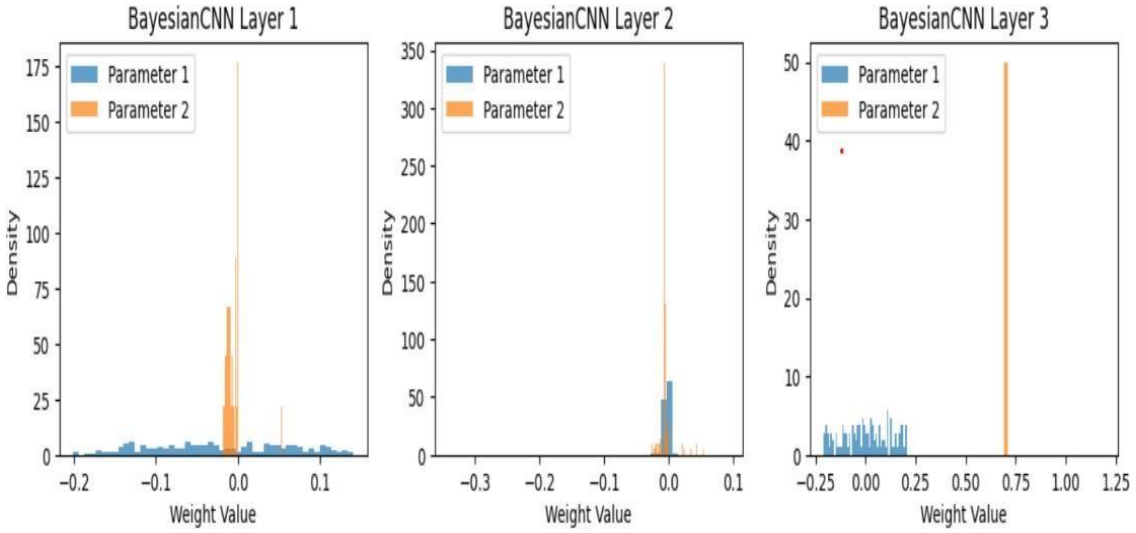


Fig 6.8 Visualisation of weights using Histogram

6.2.3 RUL Prediction

Once the training of target model is done, the saved model can be used for prediction of RUL (Fig 6.9). The model's predictions are processed, rounding the RUL value to one decimal place for precision. Subsequently, this value is transformed from a decimal representation into a time format in hours and minutes. This final prediction helps in critical insights into the lifespan of submarine. Such accurate forecasting is necessary for ensuring the reliability and enhancing safety of submarine and its crew members.

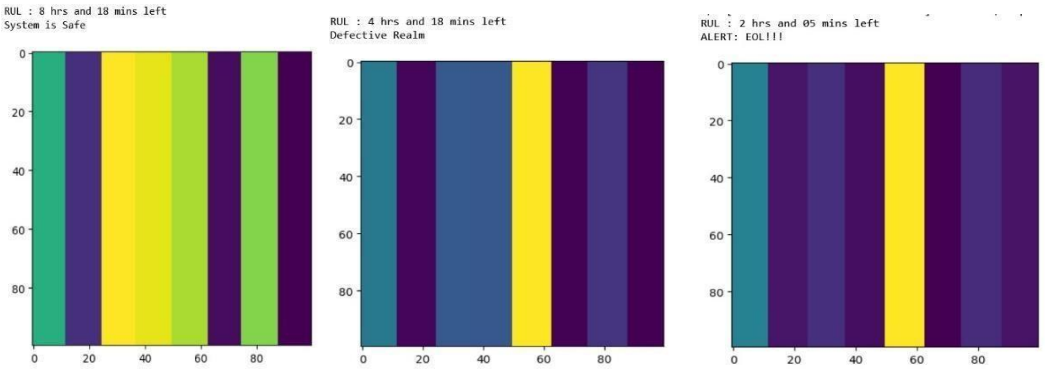


Fig 6.9 Predicted RUL in hours & mins

6.3 SUMMARY

Firstly, this chapter gives the results analyzed using MATLAB and python for Fault Identification in solenoid valves. It works on the status of 0/1. In python 0 represents as no fault detected and 1 for fault detected in solenoid valve. Likewise in MATLAB Simulink, 0 represents fault detected and 1 represents as no fault detected. The various output screenshots are explained. These data is taken as the input for the estimation of remaining useful life of the system where RUL is calculated in hours & mins based upon various human and ambient parameters. RUL estimation is done using BCNN and the performance metrics used to assess the pre-trained model is MAE and MSE.

CHAPTER 7

CONCLUSION AND FUTURE ENHANCEMENT

7.1 CONCLUSION

Fault diagnosis in solenoid valves of OIS is critical for submarine LSS, monitored through periodic statements based on four conditions: Valve opened - no fault, Valve opened - low pressure, Valve closed - not energized, Valve closed - power or oxygen is off. MATLAB Simulink and Python Collaboratory are utilized for fault diagnosis, with binary outputs indicating fault detection status as 0 & 1. The system provides status updates every second, aiding in fault diagnosis and decision-making for optimal oxygen supply. Analysis of system outputs reveals strong fault detection capabilities across various input scenarios, suggesting potential for improvement through enhanced algorithms and broader testing. Integration with real-time data acquisition systems and sensors could enhance simulation realism and practical applicability. Additionally, our implementation of RUL estimation for LSS incorporates BCNN and transfer learning. This integration is crucial for ensuring mission reliability in submarines, where component functionality is vital for crew safety and operational success. BCNN models and fine-tuning strategies improve predictive accuracy and computational efficiency. The incorporation of Bayesian Dense Variational Inference layers enables uncertainty estimation, essential for risk assessment. Comparison with Mean Squared Error (MSE)-based Stochastic Convolutional Neural Networks (CNN) demonstrates BCNN's stability and performance, particularly with Mean Absolute Error (MAE) as the loss function. Accurate estimation of RUL that is provided by the trained model in hours and minutes enhances overall dependability and contributes to mission success by facilitating real-time decision-making.

7.2 FUTURE ENHANCEMENT

Further enhancements for fault diagnostics in solenoid valves may include advanced anomaly detection techniques such as deep learning models for more accurate and robust detection of subtle deviations in system behaviour. Developing predictive maintenance algorithms based on machine learning can also be considered, allowing for proactive maintenance scheduling, and reducing downtime by predicting potential failures

before they occur. Implementing a comprehensive fault response system that includes automated decision-making based on predefined rules or machine learning algorithms can further streamline corrective actions and reduce manual intervention. Furthermore, leveraging cloud-based solutions for real-time data processing, predictive analytics, and collaborative decision-making can enhance scalability, flexibility, and accessibility of the fault detection and response system. Continuous monitoring and optimization of fault response strategies, including refining algorithms for automatic valve shutdown, alternate route activation, and emergency oxygen supply management, should also be part of the on-going enhancements to ensure optimal system performance and reliability. Also for the estimation of RUL ,there are several areas where enhancements could refine the RUL computation for LSS. Firstly, integrating an emergency signal feature that activates in response to urgent situations, combined with real-time anomaly detection algorithms, could accelerate response times, ensure timely interventions in case of any emergencies. Secondly, developing a predictive maintenance dashboard that consolidates RUL predictions, system and crew health metrics, could streamline decision-making processes for crew members. This user-friendly interface could notify the crew members with various human and ambient parameter levels by enabling them to make informed decisions about whether to continue with the mission or abort it. Thirdly, exploring advanced model architectures or ensemble methods may better capture complex relationships within the data, leading to improved predictive performance. Lastly, exploring the potential of integrating advanced sensor technologies, such as IoT-enabled devices, could provide more accurate data. This could offer deeper insights into the system's performance and degradation, further enhancing the precision and reliability of RUL predictions.

REFERENCES

[1] H. Guo, K. Wang, H. Cui, A. Xu and J. Jiang, "A Novel Method of Fault Detection for Solenoid Valves Based on Vibration Signal Measurement," 2016 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), Chengdu, China, 2016, pp. 870-873, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2016.179.

[2] Wenhao Guo, Jinjun Cheng, Yangbo Tan and Qiang Liu[1]Published under licence by IOP Publishing Ltd IOP Conference Series: Earth and Environmental Science, Volume 170, Issue 4Citation Wenhao Guo *et al* 2018 *IOP Conf. Ser.: Earth Environ. Sci.* 170 042134DOI 10.1088/1755-1315/170/4/042134

[3] Jo, Soo-Ho & Seo, Boseong & Oh, Hyunseok & Youn, Byeng Dong & Lee, Dongki. (2020). Model-Based Fault Detection Method for Coil Burnout in Solenoid Valves Subjected to Dynamic Thermal Loading. IEEE Access. 8. 77387-70400. 10.1109/ACCESS.2020.2986537.

[4] Fault State Detection in Solenoid Operated Valve based on Convolutional Neural Network using Coil Current Signature Utah Michael Ngbede, Adebena Oluwasegun, M.J. Choi, J.C. Jung* KEPCO International Nuclear Graduate School (KINGS), 658-91 Haemaji-ro, Seosaeng-myeon, Ulju-gun, Ulsan, 45014 Republic of Korea.

[5] Vassilios A. Tsachouridis, for correspondence, Nikos G. Tsagarakis, D. O. Caldwell, & Brain Robotics. (2011). Sampled Data Control of a Compliant Actuated Joint Using On/Off Solenoid Valves. Journal of Engineering Science and Technology Review, 4(1), 14–24. doi:10.25103/jestr.041.02

[6] Citation Stoyan Stoyanov and Veselin Mihaylov 2020 IOP Conf. Ser.: Mater. Sci. Eng. 1002 012033DOI 10.1088/1757-899X/1002/1/012033

[7] Said Amrane, Abdallah Zahidi, Mostafa Abouricha, Nawfel Azami, Naoual Nasser, & M. Errai. (2021). Machine Learning for Monitoring of the Solenoid Valves Coil Resistance Based on Optical Fiber Squeezer. Journal Européen Des Systèmes Automatisés, 54(5), 763–767. doi:10.18280/jesa.540511

[8] A Method for Improving Position Control Performances of a Pneumatic Cylinder Using On-Off Solenoid Valves. (2022). Jst: Smart Systems and Devices, 32(1), 34–41. doi:10.51316/jst.155.ssad.2022.32.1.5

[9] Ying Yu, S. D. Ke, & K D Jin. (2020). Structural Parameters Optimization for a Proportional Solenoid. International Journal of Simulation Modelling. doi:10.2507/ijsimm19-4-co18

[10] Research on Solenoid Valve Fault Diagnosis Algorithm Based on Coil Current

[11] TY - JOUR AU - Yoo, Seungjin AU - Jung, Joon AU - Lee, Jai-Kyung AU - Shin, Sang AU - Jang, Dal PY - 2023/08/18 SP - 7249 T1 - A Convolutional Autoencoder Based Fault Diagnosis Method for a Hydraulic Solenoid Valve Considering Unknown Faults VL - 23 DO - 10.3390/s23167249 JO - Sensors

[12] J. Li, M. Xiao, Y. Sun, G. Nie, Y. Chen and X. Tang, "Failure Mechanism Study of Direct Action Solenoid Valve Based on Thermal-Structure Finite Element Model," in IEEE Access, vol. 8, pp. 58357-58368, 2020, doi: 10.1109/ACCESS.2020.2982941.

[13] B. Seo, S.-H. Jo, H. Oh, and B. D. Youn, “Solenoid Valve Diagnosis for Railway Braking Systems with Embedded Sensor Signals and Physical Interpretation”, PHM_CONF, vol. 8, no. 1, Oct. 2016.

[14] Auzmendi JA, Moffatt L. Increasing the reliability of solution exchanges by monitoring solenoid valve actuation. J Neurosci Methods. 2010 Jan 15;185(2):280-3. doi: 10.1016/j.jneumeth.2009.10.002. Epub 2009 Oct 14. PMID: 19835912.

[15] Torsten Brune, Rolf Isermann,Model and Signal Based Fault Detection for On/Off Solenoid Valves,IFAC Proceedings Volumes,Volume 33, Issue 11,2000,Pages 423-428,ISSN 1474-6670,https://doi.org/10.1016/S1474-6670(17)37395-0.

[16] Kyoungkwan Ahn, Shinichi Yokota,Intelligent switching control of pneumatic actuator using on/off solenoid valves,Mechatronics,Volume 15, Issue 6,2005,Pages 683-702,ISSN 0957-4158,https://doi.org/10.1016/j.mechatronics.2005.01.001.

[17] Victor Vantilborgh, Tom Lefebvre, Kerem Eryilmaz, & Guillaume Crevecoeur. (2023). Data-Driven Virtual Sensing for Probabilistic Condition

Monitoring of Solenoid Valves. Ieee Transactions on Automation Science and Engineering, (99), 1–15. doi:10.1109/tase.2023.3287598

[18] Baoping Cai, Wen-Jun Li, Congkun Ren, Aibaibu Abulimiti, Xiaojie Tian, & Yanzhen Zhang. (2012). Probabilistic Thermal and Electromagnetic Analyses of Subsea Solenoid Valves for Subsea Blowout Preventers. Strojniški Vestnik, 58(11), 665–672. doi:10.5545/sv-jme.2012.681

[19] G. Mazaev, G. Crevecoeur and S. V. Hoecke, "Bayesian Convolutional Neural Networks for Remaining Useful Life Prognostics of Solenoid Valves With Uncertainty Estimations," in IEEE Transactions on Industrial Informatics, vol. 17, no. 12, pp. 8418-8428, Dec. 2021, doi: 10.1109/TII.2021.3078193.

[20] Kürşat İnce, Yakup Genc,Joint autoencoder-regressor deep neural network for remaining useful life prediction,Engineering Science and Technology, an International Journal,Volume 41,2023,101409,ISSN 2215-0986 https://doi.org/10.1016/j.jestch.2023.101409.

[21] Yang, Jinsong & Peng, Yizhen & Xie, Jingsong & Wang, Pengxi. (2022). Remaining Useful Life Prediction Method for Bearings Based on LSTM with Uncertainty Quantification. Sensors. 22. 4549. 10.3390/s22124549.

[22] A Systematic Guide for Predicting Remaining Useful Life with Machine Learning

[23] Daniel Camara Azevedo, Bernardete Ribeiro, & Alberto Cardoso. (2020). Prediction of the Remaining Useful Life of Aircraft Systems via Web Interface. International Journal of Online and Biomedical Engineering, 16, 23–32. doi:10.3991/ijoe.v16i04.11873

[24] Zhao-Hua Liu, Xudong Meng, Hua-Liang Wei, Long-Sheng Chen, Bi-Liang Lu, Zhenheng Wang, & Lei Chen. (2021). A Regularized LSTM Method for Predicting Remaining Useful Life of Rolling Bearings. International Journal of Automation and Computing, 18(4), 581–593. doi:10.1007/s11633-020-1276-6

[25] Fuqiang Sun, Xiang Li, Haitao Liao, & Xiankun Zhang. (2017). A Bayesian least-squares support vector machine method for predicting the remaining useful life of a microwave component. Advances in Mechanical Engineering, 9(1). doi:10.1177/1687814016685963

[26] Carroll James, Koukoura Sofia, McDonald Alasdair, Charalambous Anastasis, Weiss Stephan, & McArthur Stephen. (2019). Wind turbine gearbox

failure and remaining useful life prediction using machine learning techniques. Wind Energy, 22(3), 360–375. doi:10.1002/we.2290

[27] Yuhuang Zheng. (2019). Predicting Remaining Useful Life Based on Hilbert–Huang Entropy with Degradation Model. Journal of Electrical and Computer Engineering, 2019, 1–11. doi:10.1155/2019/3203959

[28] Joseph McGhee, Ian A. Henderson, Alistair Baird,Neural networks applied for the identification and fault diagnosis of process valves and actuators,Measurement,Volume 20, Issue 4,1997,Pages 267-275,ISSN 0263-2241,https://doi.org/10.1016/S0263-2241(97)00043-2.

[29] G. Belforte, M. Velardocchia,Fault Detection and Dynamic Behaviour of Pneumatic Valves,IFAC Proceedings Volumes,Volume 27, Issue 5,1994,Pages 441-446,ISSN 1474-6670,https://doi.org/10.1016/S1474-6670(17)48067-0.

[30] Lei, Wang & Cao, Hongrui & Ye, Zhisheng & Xu, Hao. (2023). Bayesian Large-kernel Attention Network for Bearing Remaining Useful Life Prediction and Uncertainty Quantification. Reliability Engineering & System Safety. 238. 109421. 10.1016/j.ress.2023.109421.

[31] https://images.app.goo.gl/BLYJT28dZQqAb6YT6

[32] https://tameson.com/pages/solenoid-valve-types

[33] https://www.bolasystems.com/help-advice/what-are-solenoid-valves

[34] https://www.iqsdirectory.com/articles/solenoid-valve.html

[35] https://www.controlandinstrumentation.com/valves/solenoid-valves.html

[36] https://in.mathworks.com/help/predmaint/ug/rul-estimation-using-rul estimator-models.html

[37] https://englishprobabilistic-machine-learningelbo-interactive--or5u7m.streamlit.app/

[38] A Novel Transfer Learning Approach in Remaining Useful Life Prediction for Incomplete Dataset January 2022 IEEE Transactions on Instrumentation and Measurement 71:1-1 DOI:10.1109/TIM.2022.3162283

[39] https://in.mathworks.com/help/predmaint/ug/rul-estimation-using-rul

[40] https://mpatacchiola.github.io/blog/2021/01/25/intro-variational-inference.html

[41] Bayesian Neural Networks: An Introduction and Survey 22 Jun 2020 https://doi.org/10.1007/978-3-030-42553-1_3

[42] Variational encoding approach for interpretable assessment of remaining useful life estimation Volume 222, June 2022, 108353 Reliability Engineering & System Safety https://doi.org/10.1016/j.ress.2022.108353

[43] https://towardsdatascience.com/understanding-the-3-most-common-loss-functions-for-machine-learning-regression-23e0ef3e14d3

APPENDIX A

SOURCE CODE AND SCREENSHOTS OF MODULES

Code & Output Screenshot for Fault Identification Using Python

```
            if (pressure >= pressure_threshold):
                # Turn on the outlet valve
                outlet_valve = 1
                # Display the message "No fault detected"
                0
                print('No fault detected')
            else:
                # Display the message "Fault detected: Low pressure"
                print('Fault detected: Low pressure')
                # Set the fault_detected variable to 1
                fault_detected = 1
        else:
            # Display the message "Fault detected: Solenoid valve not energized"
            print('Fault detected: Solenoid valve not energized')
            # Set the fault_detected variable to 1
            fault_detected = 1
    else:
        # Display the message "Power supply or oxygen supply is OFF"
        print('Power supply or oxygen supply is OFF')
        fault_detected=1
    yvalue.append(fault_detected)
print("\nvalue")

for i in yvalue:
    print(i)
print("\nlength of y value",len(yvalue))
```

```
import random
yvalue=[]
for i in range(0,60): # range(0,60) means 60 seconds
    power_supply = random.randrange(0,2) #1 for ON, 0 for OFF
    oxygen_supply = random.randrange(0,2) # 1 for ON, 0 for OFF
    solenoid_inlet_valve = random.randrange(0,2) # 0 for OFF, 1 for ON

    # 0 is no , 1 is yes (fault detected)

    multimeter = 1 # 0 for OFF, 1 for ON
    pressure_gauge = 1 # 0 for OFF, 1 for ON
    outlet_valve = 1 # 0 for OFF, 1 for ON
    fault_detected = 0 # 0 for NO, 1 for YES

    #Define the threshold for pressure check
    pressure_threshold = 10 # in bar

    # Check if the power supply and oxygen supply are ON
    if (power_supply == 1 & oxygen_supply == 1):
        # Turn on the multimeter
        multimeter = 1
        # Check if the solenoid inlet valve is energized
        if solenoid_inlet_valve == 1 :
            # Turn on the pressure gauge
            pressure_gauge = 1
            #Check the pressure with the pressure gaugedec
            # pressure = read_pressure();
            pressure = 15; # function that reads the pressure from the sensor
```

Implementation.ipynb

File Edit View Insert Runtime Tools Help Last saved at 21 April

Comment Share

+ Code + Text

Connect Colab AI

```
[ ] Power supply or oxygen supply is OFF
    Power supply or oxygen supply is OFF
    Fault detected: Solenoid valve not energized
    No fault detected
    Power supply or oxygen supply is OFF
    Fault detected: Solenoid valve not energized
    Power supply or oxygen supply is OFF
    No fault detected
    No fault detected
    Fault detected: Solenoid valve not energized
    Power supply or oxygen supply is OFF
    Power supply or oxygen supply is OFF
    Power supply or oxygen supply is OFF
    Power supply or oxygen supply is OFF
    Power supply or oxygen supply is OFF
    Power supply or oxygen supply is OFF
    Power supply or oxygen supply is OFF
    No fault detected
    Power supply or oxygen supply is OFF
    No fault detected
    Power supply or oxygen supply is OFF
    Fault detected: Solenoid valve not energized
    Fault detected: Solenoid valve not energized
    Power supply or oxygen supply is OFF
    Power supply or oxygen supply is OFF
    Power supply or oxygen supply is OFF
    Power supply or oxygen supply is OFF
    Fault detected: Solenoid valve not energized
    Power supply or oxygen supply is OFF
    Power supply or oxygen supply is OFF
    Power supply or oxygen supply is OFF
    Power supply or oxygen supply is OFF
```

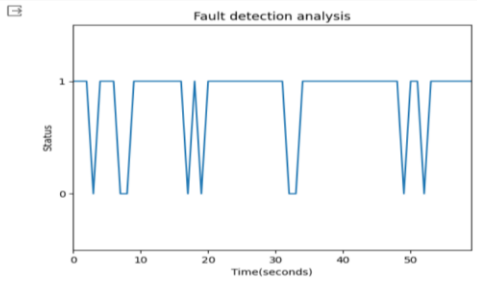
+ Code + Text

Colab AI

```
0
1
1
1
0
0
1
1
0
0
1
1
1
1
1
0
1
0
1
0
1
0
0
1
1
1
1
1

length of y value 60
```

Code & Output for Generating Graph Of Fault Identification Using Python (60 secs)

```
import numpy as np
import matplotlib.pyplot as plt

# Assuming yvalue is already defined from your previous code
y = np.array(yvalue)

plt.plot(y)  # Using y directly without x, assuming x is not needed for this modification

plt.title("Fault detection analysis")
plt.ylabel('Status')
plt.xlabel('Time(seconds)')

# Set y-axis ticks to only display 0 and 1
plt.yticks([0, 1])

plt.xlim(0, len(y) - 1)  # Adjusting the x-axis limit based on the length of y
plt.ylim(-0.5, 1.5)  # Setting y-axis limit to include only 0 and 1

plt.show()
```



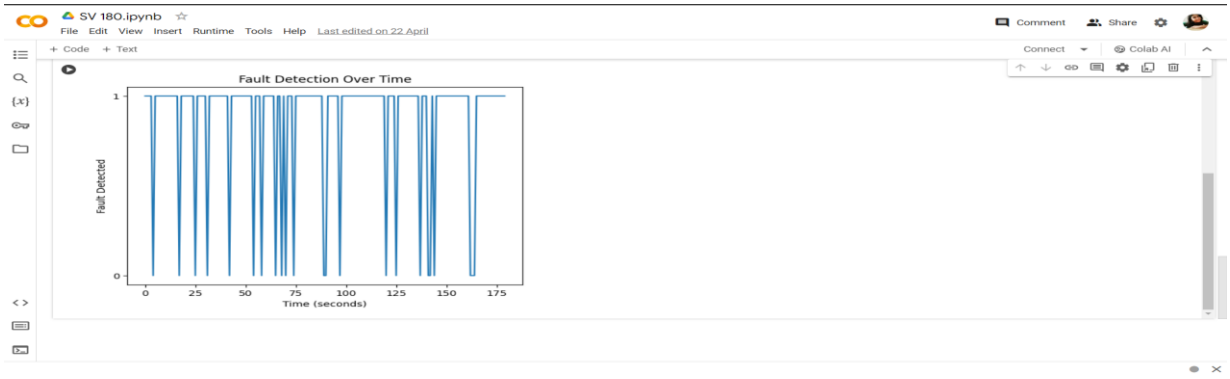
Code & Output for Generating Graph of Fault Identification Using Python (180 secs)

```
import random
import matplotlib.pyplot as plt

yvalue = []
for i in range(0, 180):  # Generate 180 outputs
    power_supply = random.randrange(0, 2)  # 1 for ON, 0 for OFF
    oxygen_supply = random.randrange(0, 2)  # 1 for ON, 0 for OFF
    solenoid_inlet_valve = random.randrange(0, 2)  # 0 for OFF, 1 for ON

    # 0 is no, 1 is yes (fault detected)

    multimeter = 1  # 0 for OFF, 1 for ON
    pressure_gauge = 1  # 0 for OFF, 1 for ON
    outlet_valve = 1  # 0 for OFF, 1 for ON
    fault_detected = 0  # 0 for NO, 1 for YES

    # Define the threshold for pressure check
    pressure_threshold = 10  # in bar

    # Check if the power supply and oxygen supply are ON
    if power_supply == 1 and oxygen_supply == 1:  # Use 'and' instead of '&'
        # Turn on the multimeter
        multimeter = 1
        # Check if the solenoid inlet valve is energized
        if solenoid_inlet_valve == 1:
            # Turn on the pressure gauge
            pressure_gauge = 1
            # Check the pressure with the pressure gauge
            # pressure = read_pressure();
```

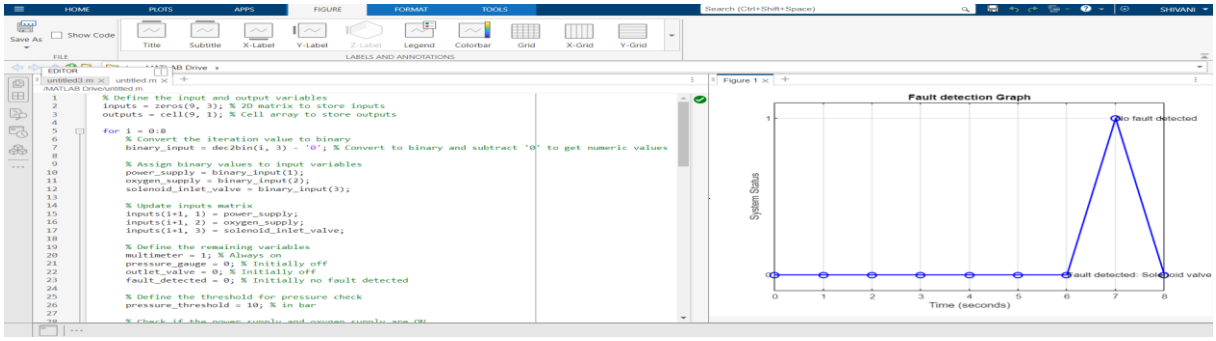
```
plt.xlim(0, len(y) - 1)  # Adjusting the x-axis limit based on the length of y
plt.ylim(-0.5, 1.5)  # Setting y-axis limit to include only 0 and 1


plt.show()
```

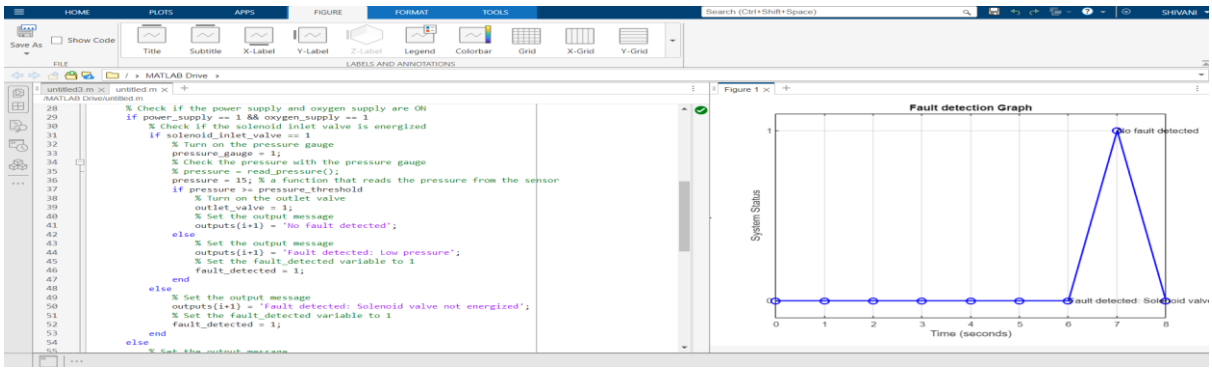
```
Power supply or oxygen supply is OFF
No fault detected
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
No fault detected
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Fault detected: Solenoid valve not energized
Fault detected: Solenoid valve not energized
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
No fault detected
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Fault detected: Solenoid valve not energized
Fault detected: Solenoid valve not energized
```

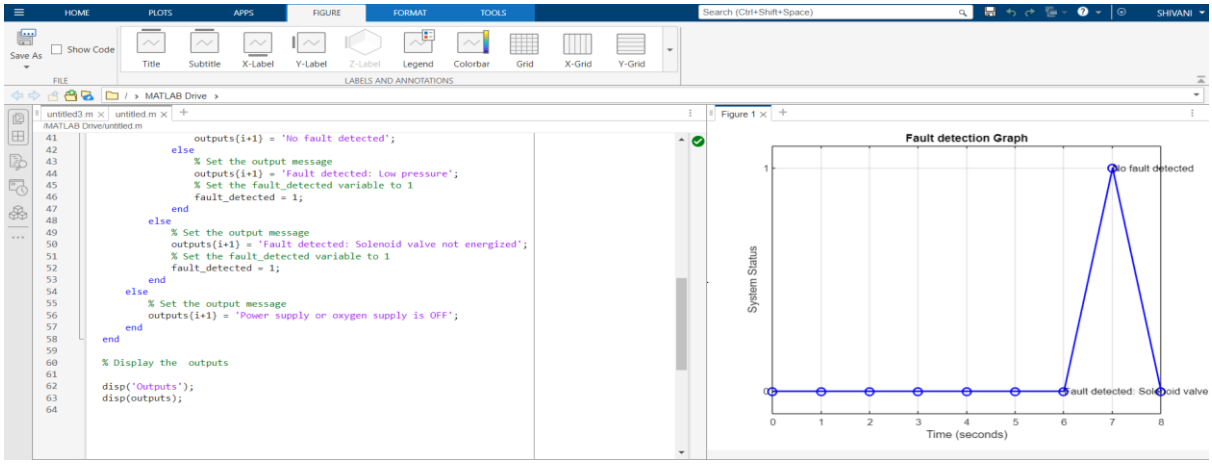
```
No fault detected
No fault detected
Power supply or oxygen supply is OFF
No fault detected
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Fault detected: Solenoid valve not energized
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Fault detected: Solenoid valve not energized
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
No fault detected
No fault detected
No fault detected
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
Fault detected: Solenoid valve not energized
Power supply or oxygen supply is OFF
Power supply or oxygen supply is OFF
```

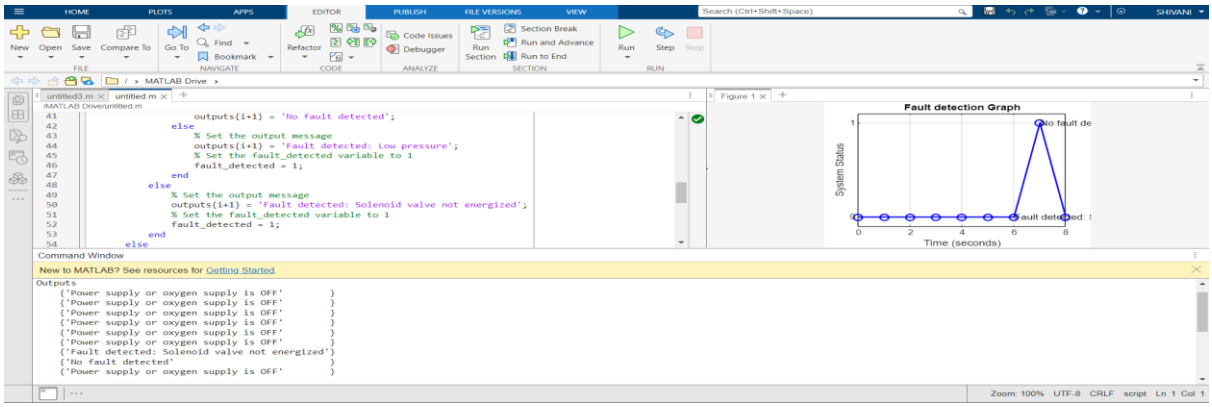


Code & Output for Generating Graph of Fault Identification Using MATLAB (8secs)









File path for BCNN and Target model

```
In [1]: import pandas as pd
        from sklearn.model_selection import train_test_split

        data = pd.read_excel(r"C:\Users\Sindhu\mapping_output.xlsx")
        data_test = pd.read_excel(r"C:\Users\Sindhu\mapping_output_test.xlsx")

In [2]: import numpy as np
        import tensorflow as tf
        from tensorflow.keras.preprocessing.image import load_img, img_to_array
        from tensorflow.keras.models import Sequential
        from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
        from sklearn.metrics import accuracy_score

        WARNING:tensorflow:From C:\Users\Sindhu\AppData\Roaming\Python\Python311\site-packages\keras\src\losses.py:2976: The name tf.lo
        sses.sparse_softmax_cross_entropy is deprecated. Please use tf.compat.v1.losses.sparse_softmax_cross_entropy instead.

In [3]: import numpy as np
        import pandas as pd
        import tensorflow as tf

        from keras.preprocessing.image import ImageDataGenerator, load_img
        from keras.utils import to_categorical
        from keras import optimizers
        from keras.optimizers import SGD, RMSprop, Adam
        from sklearn.model_selection import train_test_split
        from keras.utils import to_categorical
        from keras.callbacks import EarlyStopping

        import matplotlib.pyplot as plt
        import random

In [4]: X = data['Path']
        y = data['RUL']

In [5]: X_new = data_test['Path']
        y_new = data_test['RUL']

In [6]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

In [7]: def load_and_preprocess_images(image_paths):
            images = []
            for path in image_paths:
                img = load_img(path, target_size=(100,100))
                img_array = img_to_array(img) / 255.0
                images.append(img_array)
            return np.array(images)

In [8]: X_train_processed = load_and_preprocess_images(X_train)
        X_test_processed = load_and_preprocess_images(X_test)

In [9]: X_new_processed = load_and_preprocess_images(X_new)

In [10]: print("Training set:",
                "IMAGES (X):",X_train.shape[0],
                "LABELS(y):", y_train.shape[0] ,
                "\nTest set :" ,
                "IMAGES (X):", X_test.shape[0],
                "LABELS(y):", y_test.shape[0])

         Training set: IMAGES (X): 2880 LABELS(y): 2880
         Test set : IMAGES (X): 720 LABELS(y): 720

In [12]: print("y_train:", y_train)
         print("Length of y_train:", len(y_train))

         y_train: 3281    0.00
         2383    5.19
         2009    4.15
         2114    5.48
         1128    9.04
                 ...
         1130    9.04
         1294    5.17
         860     6.34
         3507    1.56
         3174    0.58
         Name: RUL, Length: 2880, dtype: float64
         Length of y_train: 2880

In [13]: # A random sample of 10 images from the dataset.
         plt.figure(figsize=(10, 10))
         for i in range(10,20):
             plt.subplot(5,5,i+1)
             plt.grid(False)
             plt.xticks([])
             plt.yticks([])
             plt.title('RUL: {:.2f}'.format(y_new.iloc[i]))
             plt.tight_layout(pad=2)
             plt.imshow(X_new_processed.squeeze()[i], cmap=plt.cm.binary)
         plt.show()
```

A Random Sample of 10 images from the dataset


```
In [14]: import cv2
         def check_image_color(image_path):
             image = cv2.imread(image_path)
             num_channels = image.shape[2]

             if num_channels == 1:
                 print("Grayscale image")
             elif num_channels == 3:
                 print("RGB image")
             else:
                 print("Unknown image type")
         check_image_color(r"C:\Users\Sindhu\3class_intensity\3281.png")

         RGB image
```

In [11]:
```
def rgb_to_grayscale(images):
    return np.dot(images[..., :3], [0.2989, 0.5870, 0.1140])

X_train_processed = load_and_preprocess_images(X_train)
X_train_processed = rgb_to_grayscale(X_train_processed)
```

In [12]:
```
def rgb_to_grayscale(images):
    return np.dot(images[..., :3], [0.2989, 0.5870, 0.1140])

X_new_processed = load_and_preprocess_images(X_new)
X_new_processed = rgb_to_grayscale(X_new_processed)
```

In [13]:
```
import numpy as np

X_train_processed = np.expand_dims(X_train_processed, axis=-1)
```

In [14]:
```
X_new_processed = np.expand_dims(X_new_processed, axis=-1)
```

In [19]:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import load_img, img_to_array
```

In [20]:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import load_img, img_to_array

class BayesianCNN(tf.keras.Model):
```

Architecture for Bayesian CNN

In [20]:
```
import numpy as np
import tensorflow as tf
from tensorflow.keras.preprocessing.image import load_img, img_to_array

class BayesianCNN(tf.keras.Model):
    def __init__(self):
        super(BayesianCNN, self).__init__()
        self.conv1 = tf.keras.layers.Conv2D(32, (3, 3), activation='relu', groups=1)
        self.flatten = tf.keras.layers.Flatten()
        self.dense1 = tf.keras.layers.Dense(128, activation='relu')
        self.dropout = tf.keras.layers.Dropout(0.5)
        self.dense2 = tf.keras.layers.Dense(1)

    def call(self, inputs):
        x = self.conv1(inputs)
        x = self.flatten(x)
        x = self.dense1(x)
        x = self.dropout(x)
        return self.dense2(x)
```

In [21]:
```
bayesian_cnn = BayesianCNN()
```

```
WARNING:tensorflow:From C:\Users\Sindhu\AppData\Roaming\Python\Python311\site-packages\keras\src\backend.py:873: The name tf.get_default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.
```

In [22]:
```
bayesian_cnn.compile(optimizer=Adam(lr=0.01), loss='mae', metrics=['mae'])
```

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.legacy.Adam.
```

In [23]:
```
history = bayesian_cnn.fit(X_train_processed, y_train, batch_size=16, epochs=9, validation_split=0.2)
```

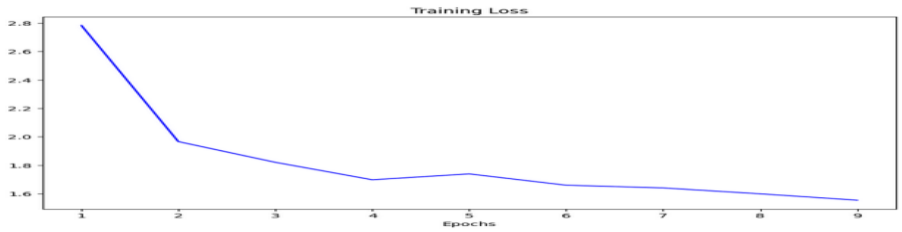
```
Epoch 1/9
WARNING:tensorflow:From C:\Users\Sindhu\AppData\Roaming\Python\Python311\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\Sindhu\AppData\Roaming\Python\Python311\site-packages\keras\src\utils\tf_utils.py:492: The name tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\Sindhu\AppData\Roaming\Python\Python311\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

WARNING:tensorflow:From C:\Users\Sindhu\AppData\Roaming\Python\Python311\site-packages\keras\src\engine\base_layer_utils.py:384: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions instead.

144/144 [==============================] - 48s 317ms/step - loss: 2.7826 - mae: 2.7826 - val_loss: 1.3961 - val_mae: 1.3961
Epoch 2/9
144/144 [==============================] - 42s 289ms/step - loss: 1.9661 - mae: 1.9661 - val_loss: 1.2481 - val_mae: 1.2481
Epoch 3/9
144/144 [==============================] - 42s 290ms/step - loss: 1.8206 - mae: 1.8206 - val_loss: 1.5669 - val_mae: 1.5669
Epoch 4/9
144/144 [==============================] - 40s 279ms/step - loss: 1.6980 - mae: 1.6980 - val_loss: 1.1466 - val_mae: 1.1466
Epoch 5/9
144/144 [==============================] - 41s 283ms/step - loss: 1.7393 - mae: 1.7393 - val_loss: 1.1578 - val_mae: 1.1578
Epoch 6/9
144/144 [==============================] - 43s 297ms/step - loss: 1.6598 - mae: 1.6598 - val_loss: 1.2904 - val_mae: 1.2904
Epoch 7/9
144/144 [==============================] - 42s 288ms/step - loss: 1.6407 - mae: 1.6407 - val_loss: 1.1586 - val_mae: 1.1586
```

Loss Graph for Bayesian CNN

In [33]:
```
import matplotlib.pyplot as plt
loss = history.history['loss']
mse = history.history['mae']
epochs = range(1, len(loss) + 1)
```

In [34]:
```
plt.figure(figsize=(10, 6))
plt.plot(epochs, loss, 'b', label='Loss')
plt.title('Training Loss')
plt.xlabel('Epochs')
```

Out[34]:
```
Text(0.5, 0, 'Epochs')
```



In [27]:
```
import numpy as np
from tensorflow.keras.preprocessing.image import load_img, img_to_array

image_path = r'C:\Users\Sindhu\3class_intensity\201.png'
img = load_img(image_path, target_size=(100, 100), color_mode='grayscale')
img_array = img_to_array(img) / 255.0

input_image = np.expand_dims(img_array, axis=0)

prediction = bayesian_cnn.predict(input_image)

print("RUL Predicted value:", prediction)
```

```
1/1 [==============================] - 1s 774ms/step
RUL Predicted value: [[7.5083895]]
```

In [28]:
```
print("X_train_processed shape:", X_train_processed.shape)
print("y_train_encoded shape:", y_train.shape)
```

```
X_train_processed shape: (2880, 100, 100, 1)
y_train_encoded shape: (2880,)
```

In [29]:
```
------------------------------------------------------------
'''bayesian_cnn.save_weights('bcnn_model_weights.h5')
```

Saved BCNN model

In [ ]:
```
'''bayesian_cnn.save('bcnn_model_save', save_format='tf')
```

In [15]:
```
import tensorflow as tf
loaded_model = tf.keras.models.load_model('bcnn_model_save')
```

```
WARNING:tensorflow:From C:\Users\Sindhu\AppData\Roaming\Python\Python311\site-packages\keras\src\saving\legacy\saved_model\loa
d.py:107: The name tf.gfile.Exists is deprecated. Please use tf.io.gfile.exists instead.

WARNING:tensorflow:From C:\Users\Sindhu\AppData\Roaming\Python\Python311\site-packages\keras\src\saving\legacy\saved_model\loa
d.py:1113: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_fun
ctions instead.
```

In [16]:
```
import tensorflow as tf
from tensorflow.keras.layers import Layer, BatchNormalization
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.regularizers import l2
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Load and preprocess your data
X_test_processed = load_and_preprocess_images(X_test)
X_test_processed = rgb_to_grayscale(X_test_processed)
X_test_processed = np.expand_dims(X_test_processed, axis=-1)
y_pred = loaded_model.predict(X_test_processed)
```

```
23/23 [==============================] - 3s 39ms/step
```

BayesianDenseVI layer for target model after applying transfer learning

In [17]:
```
class BayesianDenseVI(Layer):
    def __init__(self, units, activation='relu'):
        super(BayesianDenseVI, self).__init__()
        self.units = units
        self.activation = tf.keras.activations.get(activation)
        self.batch_norm = BatchNormalization()

    def build(self, input_shape):
        self.mean = self.add_weight(shape=(input_shape[-1], self.units),
                                    initializer='random_normal',
                                    trainable=True,
                                    name='mean')
        self.rho = self.add_weight(shape=(input_shape[-1], self.units),
                                   initializer='zeros',
                                   trainable=True,
                                   name='rho')

    def call(self, inputs):
        epsilon = tf.random.normal(shape=tf.shape(self.mean))
        weights = self.mean + tf.math.log(1 + tf.exp(self.rho)) * epsilon
        x = tf.matmul(inputs, weights)
        x = self.batch_norm(x)
        return self.activation(x)
```

In [18]:
```
pretrained_model_path = r'C:\Users\Sindhu\Major\bcnn_model_save'
pre_trained_model = tf.keras.models.load_model(pretrained_model_path)

for layer in pre_trained_model.layers[:-4]:
    layer.trainable = False

pre_trained_model.layers[-1] = BayesianDenseVI(units=1, activation='relu')

def kl_divergence_regularizer(model):
    kl_loss = sum(model.losses)
    return kl_loss

def custom_loss(y_true, y_pred):
    original_loss = tf.keras.losses.mean_absolute_error(y_true, y_pred)
    kl_loss = kl_divergence_regularizer(pre_trained_model)
    return original_loss + kl_loss
```

In [19]:
```
optimizer = Adam(lr=0.000001)  # Adjust learning rate
pre_trained_model.compile(optimizer=optimizer, loss=custom_loss, metrics=['mae'])
```

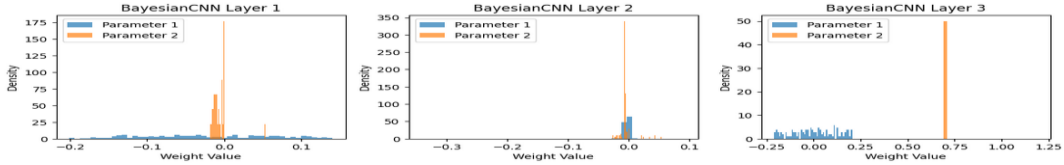
```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optim
izers.legacy.Adam.
```

In [20]:
```
early_stopping = tf.keras.callbacks.EarlyStopping(patience=5, restore_best_weights=True)

# Train the model
history_transfer_learning = pre_trained_model.fit(X_new_processed, y_new,
                                                  batch_size=16,
                                                  epochs=10,
                                                  validation_data=(X_new_processed, y_new),
                                                  callbacks=[early_stopping])
```

```
Epoch 1/10
WARNING:tensorflow:From C:\Users\Sindhu\AppData\Roaming\Python\Python311\site-packages\keras\src\utils\tf_utils.py:492: The nam
e tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\Sindhu\AppData\Roaming\Python\Python311\site-packages\keras\src\utils\tf_utils.py:492: The nam
e tf.ragged.RaggedTensorValue is deprecated. Please use tf.compat.v1.ragged.RaggedTensorValue instead.

WARNING:tensorflow:From C:\Users\Sindhu\AppData\Roaming\Python\Python311\site-packages\keras\src\engine\base_layer_utils.py:38
4: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions i
nstead.

WARNING:tensorflow:From C:\Users\Sindhu\AppData\Roaming\Python\Python311\site-packages\keras\src\engine\base_layer_utils.py:38
4: The name tf.executing_eagerly_outside_functions is deprecated. Please use tf.compat.v1.executing_eagerly_outside_functions i
```

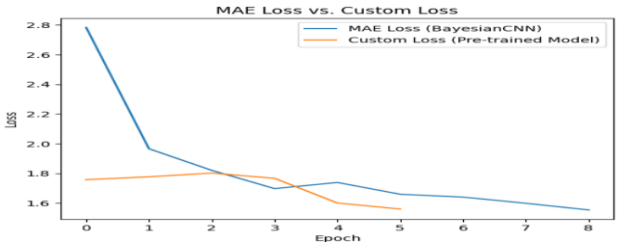
```
10/10 [==============================] - 11s 630ms/step - loss: 1.7474 - mae: 1.7474 - val_loss: 0.9905 - val_mae: 0.9905
Epoch 2/10
10/10 [==============================] - 3s 295ms/step - loss: 1.6840 - mae: 1.6840 - val_loss: 1.1697 - val_mae: 1.1697
Epoch 3/10
10/10 [==============================] - 3s 282ms/step - loss: 1.5774 - mae: 1.5774 - val_loss: 1.0902 - val_mae: 1.0902
Epoch 4/10
10/10 [==============================] - 3s 311ms/step - loss: 1.5171 - mae: 1.5171 - val_loss: 0.9185 - val_mae: 0.9185
Epoch 5/10
10/10 [==============================] - 3s 291ms/step - loss: 1.5583 - mae: 1.5583 - val_loss: 1.1404 - val_mae: 1.1404
Epoch 6/10
10/10 [==============================] - 3s 290ms/step - loss: 1.5852 - mae: 1.5852 - val_loss: 1.2222 - val_mae: 1.2222
Epoch 7/10
10/10 [==============================] - 3s 280ms/step - loss: 1.5816 - mae: 1.5816 - val_loss: 0.9680 - val_mae: 0.9680
Epoch 8/10
10/10 [==============================] - 3s 311ms/step - loss: 1.7548 - mae: 1.7548 - val_loss: 1.0051 - val_mae: 1.0051
Epoch 9/10
10/10 [==============================] - 3s 306ms/step - loss: 1.6444 - mae: 1.6444 - val_loss: 0.9805 - val_mae: 0.9805
```

In [37]:
```
import numpy as np
import matplotlib.pyplot as plt

weights_bayesian_cnn = []
for layer in bayesian_cnn.layers:
    if isinstance(layer, tf.keras.layers.Conv2D) or isinstance(layer, tf.keras.layers.Dense):
        weights_bayesian_cnn.append(layer.get_weights())

plt.figure(figsize=(12, 6))
for i, layer_weights in enumerate(weights_bayesian_cnn):
    plt.subplot(2, len(weights_bayesian_cnn), i + 1)
    plt.title(f'BayesianCNN Layer {i+1}')
    for j, weights in enumerate(layer_weights):
        plt.hist(weights.flatten(), bins=50, density=True, alpha=0.7, label=f'Parameter {j+1}')
    plt.xlabel('Weight Value')
    plt.ylabel('Density')
    plt.legend()
```

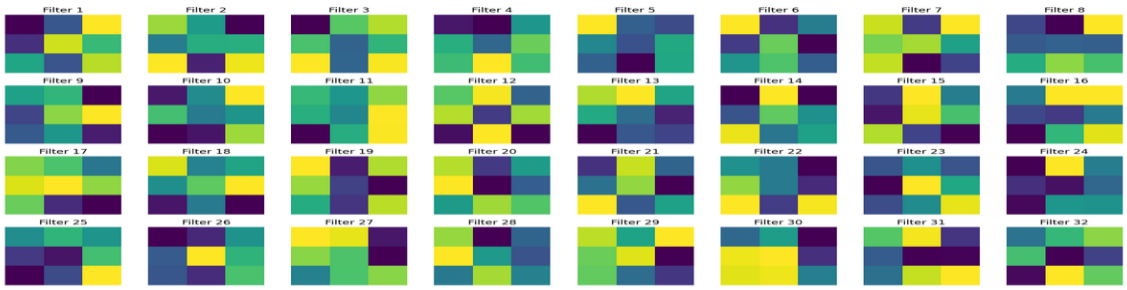


Graph for MAE vs Custom loss

```
In [31]: import matplotlib.pyplot as plt
plt.plot(history.history['mae'], label='MAE Loss (BayesianCNN)')
plt.plot(history_transfer_learning.history['loss'], label='Custom Loss (Pre-trained Model)')
plt.title('MAE Loss vs. Custom Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()
```




```
In [60]: import matplotlib.pyplot as plt

# Extract and visualize convolutional filters
filters, biases = bayesian_cnn.conv1.get_weights()
f_min, f_max = filters.min(), filters.max()
filters = (filters - f_min) / (f_max - f_min)

def feature_dropout_importance(model, X, y_true):
    baseline_loss, _ = model.evaluate(X, y_true, verbose=0)
    importance_scores = []
    for i in range(X.shape[1]):
        X_drop = X.copy()
        X_drop[:, i] = 0
        loss_drop, _ = model.evaluate(X_drop, y_true, verbose=0)
        importance_score = (baseline_loss - loss_drop) / baseline_loss
        importance_scores.append(importance_score)
    return importance_scores

n_filters = filters.shape[3]
rows = 4
cols = n_filters // rows

plt.figure(figsize=(cols * 2, rows * 2))

for i in range(n_filters):
    f = filters[:, :, 0, i]  # Accessing the first channel of the filter
    plt.subplot(rows, cols, i+1)
    plt.imshow(f, cmap='viridis')
    plt.title(f'Filter {i+1}')
    plt.axis('off')

plt.tight_layout()
plt.show()
```



Estimation of RUL

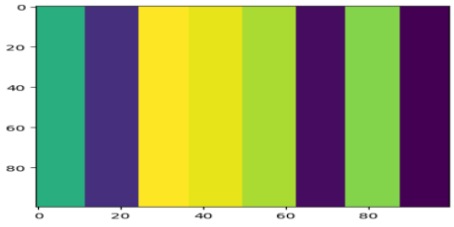
```
In [36]: import numpy as np
from tensorflow.keras.preprocessing.image import load_img, img_to_array
image_path = r'C:\Users\Sindhu\3class_intensity\1101.png'
img = load_img(image_path, target_size=(100, 100), color_mode='grayscale')
img_array = img_to_array(img) / 255.0
input_image = np.expand_dims(img_array, axis=0)
prediction = pre_trained_model.predict(input_image)
plt.imshow(img)
#print("RUL Predicted value:", prediction)
#print(f"RUL : {prediction[0][0]:.2f} hrs")
pred = round(prediction[0][0], 1)
#print(pred)
def convert_decimal_to_time(decimal_value):
    hours = int(decimal_value)
    minutes = int((decimal_value - hours) * 60)
    return f"{hours} hrs and {minutes:02d} mins"
time_string = convert_decimal_to_time(pred)

print(f"RUL : {time_string} left")

if(prediction>=6 and prediction<=10):
    print("System is Safe")
elif(prediction>=4 and prediction<6):
    print("Defective Realm")
elif(prediction<4):
    print("ALERT: EOL!!!")
else:
    print("Error Calculating")
```

```
1/1 [==============================] - 1s 1s/step
RUL : 7 hrs and 41 mins left
System is Safe
```

RUL in hours and minutes

```
    print("Defective Realm")
elif(prediction<4):
    print("ALERT: EOL!!!")
else:
    print("Error Calculating")
```

```
1/1 [==============================] - 1s 1s/step
RUL : 7 hrs and 41 mins left
System is Safe
```



APPENDIX B
PLAGIARISM REPORT

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY **(Deemed to be University u/s 3 of UGC Act, 1956)**		
Office of Controller of Examinations		
REPORT FOR PLAGIARISM CHECK ON THE DISSERTATION/PROJECT REPORTS FOR UG/PG PROGRAMMES **(To be attached in the dissertation/ project report)**		
1	Name of the Candidate **(IN BLOCK LETTERS)**	1. R SHIVANI 2. SINDHU KALEESWARAN 3. VAISHALI V 4. CHEREDDY SOWMYA SRI
2	Address of the Candidate	1. N408,Abode Valley,kakkan street,Kattankulathur,Chennai 2. 4/257,5th Cross Street,Perumal Nagar,Nanmangalam, Chennai. 3. G5 , E-block, Vandalur Park Residency,Urapakkam,Chennai 4. 10-87-3,Ravindra Nagar,Madanapalli,Chittoor district,Andhra pradesh.
3	Registration Number	1. RA2011026010060 2. RA2011026010082 3. RA2011026010100 4. RA2011026010113
4	Date of Birth	1. 26/08/2002 2. 27/10/2002 3. 20/06/2002 4. 14/11/2002
5	Department	Department of Computational Intelligence
6	Faculty	Faculty of Engineering and Technology
7	Title of the Dissertation/Project	Determination Of Remaining Useful Life And Fault Diagnostics Of Solenoid Valve In Oxygen Injection System
8	Whether the above project /dissertation is done by	Group : 4 members 1. R SHIVANI - RA2011026010060 2. SINDHU KALEESWARAN- RA2011026010082 3. VAISHALI V- RA2011026010100 4. CHEREDDY SOWMYA SRI- RA2011026010113
9	Name and address of the Supervisor / Guide	Dr. A Jackulin Mahariba Assistant professor, Department of Computational Intelligence **Mail ID:** jackulia@srmist.edu.in **Mobile Number:** 9841088361

10	Name and address of Co-Supervisor / Co- Guide (if any)	**Mail ID: -** **Mobile Number: -**
11	Software Used	Turnitin
12	Date of Verification	29/04/2024
13	**Plagiarism Details: (to attach the final report from the software)**	

Chapter	**Title of the Chapter**	**Percentage of similarity index (including self citation)**	**Percentage of similarity index (Excluding self-citation)**	**% of plagiarism after excluding Quotes, Bibliography, etc.,**
1	Determination Of Remaining Useful Life And Fault Diagnostics Of Solenoid Valve In Oxygen Injection System	8%	8%	8%
2				
3				
4				
5				
6				
7				
8				
Appendices		0%	0%	0%

We declare that the above information have been verified and found true to the best of our knowledge.

Signature of the Candidate	**Name & Signature of the Staff (Who uses the plagiarism check software)**
Name & Signature of the Supervisor/ Guide	**Name & Signature of the Co-Supervisor/Co-Guide**
Name & Signature of the HOD	

OIS project report

ORIGINALITY REPORT

8%	5%	3%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	www.iqsdirectory.com Internet Source	1%
2	Lei Wang, Hongrui Cao, Zhisheng Ye, Hao Xu. "Bayesian large-kernel attention network for bearing remaining useful life prediction and uncertainty quantification", Reliability Engineering & System Safety, 2023 Publication	1%
3	www.butlerandland.com Internet Source	1%
4	Submitted to Virginia Polytechnic Institute and State University Student Paper	<1%
5	deepai.org Internet Source	<1%
6	Submitted to University of Wales Swansea Student Paper	<1%
7	Submitted to The University of Manchester Student Paper	<1%

8	cs.brown.edu Internet Source	<1%
9	Nam-Ho Kim, Dawn An, Joo-Ho Choi. "Prognostics and Health Management of Engineering Systems", Springer Science and Business Media LLC, 2017 Publication	<1%
10	web.archive.org Internet Source	<1%
11	Submitted to Nanyang Technological University Student Paper	<1%
12	Submitted to University of Oxford Student Paper	<1%
13	kylo.tv Internet Source	<1%
14	Jerónimo Andrés Auzmendi, Luciano Moffatt. "Increasing the reliability of solution exchanges by monitoring solenoid valve actuation", Journal of Neuroscience Methods, 2010 Publication	<1%
15	Submitted to University of Glamorgan Student Paper	<1%
16	www.techscience.com Internet Source	<1%

17	Submitted to University of Stellenbosch, South Africa Student Paper	<1 %
18	Submitted to unistgallen-plagiat Student Paper	<1 %
19	Submitted to University of Hertfordshire Student Paper	<1 %
20	docplayer.net Internet Source	<1 %
21	Fernando Castano, Yarens J. Cruz, Alberto Villalonga, Rodolfo E. Haber. "Data-Driven Insights on Time-to-Failure of Electromechanical Manufacturing Devices: A Procedure and Case Study", IEEE Transactions on Industrial Informatics, 2023 Publication	<1 %
22	Submitted to Veermata Jijabai Technological Institute Student Paper	<1 %
23	cityofhenderson.com Internet Source	<1 %
24	jst.hust.edu.vn Internet Source	<1 %
25	www.arxiv-vanity.com Internet Source	<1 %

26	www.coursehero.com Internet Source	<1 %
27	www.cs.princeton.edu Internet Source	<1 %
28	www.phmsociety.org Internet Source	<1 %
29	arxiv.org Internet Source	<1 %
30	Submitted to Asia Pacific University College of Technology and Innovation (UCTI) Student Paper	<1 %
31	Submitted to Brunel University Student Paper	<1 %
32	Submitted to National Institute of Technology Karnataka Surathkal Student Paper	<1 %
33	dokumen.pub Internet Source	<1 %
34	ebin.pub Internet Source	<1 %
35	www.manmonthly.com.au Internet Source	<1 %
36	Submitted to Birkbeck College Student Paper	<1 %

37	www.scss.tcd.ie Internet Source	<1%
38	"Proceedings of the International Conference on Advanced Intelligent Systems and Informatics 2021", Springer Science and Business Media LLC, 2022 Publication	<1%
39	Submitted to Berlin School of Business and Innovation Student Paper	<1%
40	Submitted to Central Queensland University Student Paper	<1%
41	studymoose.com Internet Source	<1%
42	A. Louri, B. Weech, C. Neocleous. "A spanning multichannel linked hypercube: a gradually scalable optical interconnection network for massively parallel computing", IEEE Transactions on Parallel and Distributed Systems, 1998 Publication	<1%
43	A. Waibel. "Class phrase models for language modeling", Proceeding of Fourth International Conference on Spoken Language Processing ICSLP 96 ICSLP-96, 1996 Publication	<1%

44	Submitted to University of Salford Student Paper	<1 %
45	Submitted to M S Ramaiah University of Applied Sciences Student Paper	<1 %
46	Submitted to Queen Mary and Westfield College Student Paper	<1 %
47	Submitted to University of Glasgow Student Paper	<1 %
48	Submitted to University of Sunderland Student Paper	<1 %
49	biolexicon.asme.org Internet Source	<1 %
50	literature.rockwellautomation.com Internet Source	<1 %
51	www.industrial-electronics.com Internet Source	<1 %
52	Submitted to AlHussein Technical University Student Paper	<1 %
53	Submitted to Morgan State University Student Paper	<1 %
54	backend.orbit.dtu.dk Internet Source	<1 %

55	dev.to Internet Source	<1%
56	www.science.gov Internet Source	<1%
57	www.teses.usp.br Internet Source	<1%

Exclude quotes	On	Exclude matches	< 10 words
Exclude bibliography	On		