Czech University of Life Sciences Prague

Faculty of Economics and Management

Informatics

Master's Thesis

Viability of Siemens Industrial Edge to Deploy AI Into Automation Processes.

Pankaj Mishra

CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

DIPLOMA THESIS ASSIGNMENT

Pankaj Mishra, B.Tech.

Informatics

Thesis title

VIABILITY OF SIEMENS INDUSTRIAL EDGE TO DEPLOY AI INTO AUTOMATION PROCESSES.

Objectives of thesis

• Study and understand the viability of Industrial Edge, a SIEMENS solution intended for IoT, to deploy AI methods in automation.

• Identify the requirements and critical parameters to deploy AI methods into automation with Predictive maintenance.

• Define potential architecture(s) to deploy AI methods in Siemens Industrial Edge.

• Build the architecture(s) to evaluate the critical parameters identified.

• Evaluate and discuss the viability and conditions where Industrial Edge is suitable for AI, if any.

Methodology

The project can be framed into applied research as it studies the viability to deploy a new technology of an existing technological solution of SIEMENS for IoT in Industry 4.0.

The methodology to achieve such evaluation can be divided in four major blocks:

• Build knowledge in AI for Automation and SIEMENS Industrial Edge:

This phase covers the study of the state-of-the-art, AI architectures and the features of the product where this new technology is to be deployed: Industrial Edge.

• Identify the critical points and requirements:

In this stage, the requirements for Predictive maintenance in an industrial environment are to be set. The real time data will be collected directly from viking masek (A Siemens packaging machine) and analysis will be done on the data.

Internal information of AI-based solutions within the company can be used. The main goal is to identify which are the requirements and critical factors of the process (i.e.: operation cycle, process frequency, type of synchronization...) that determine whether Edge is suitable to deploy AI.

• Experiments and results:

Define, construct, and carry various tests to find out the critical parameters defined. Test's like (Correlation distribution, PCA, ICA etc) will be performed on the collected data and based on that a neural network will be proposed. Such definition should only focus on the components that are essential to evaluate predictive maintenance and the deployment of AI.

• Evaluation and discussion of the results:

Evaluate the pipelines proposed and discuss their viability. This discussion must also characterize which critical parameters comes out best for predictive maintenance.It also includes deploying the architecture proposed on Industrial Edge and carry some experiments to be able to evaluate and discuss about it.

The proposed extent of the thesis

30-40

Keywords

Industry 4.0, Artificial Intelligence, Industrial Edge, Automation, Predictive maintenance

---

Recommended information sources

GÉRON, A. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems.* Beijing ; Boston ; Farnham ; Sevastopol ; Tokyo: O'Reilly, 2019. ISBN 978-1-4920-3264-9.

CHEN, D Y. *Pandas for everyone : Python data analysis.* Boston: Addison-Wesley, 2018. ISBN 9780134546933.

CHOLLET, F. *Deep learning with Python.* Shelter Island: Manning, 2021. ISBN 978-1-61729-686-4.

LEVITT, J. *Complete guide to preventive and predictive maintenance.* New York: Industrial Press, 2011. ISBN 978-0-8311-3441-9.

Susto, G. A., Schirru, A., Pampuri, S., McLoone, S. & Beghi, A. Machine learning for predictive maintenance: A multiple classifier approach. IEEE Transactions on Industrial Informatics 11, 812–820 (2015).

Expected date of thesis defence

2022/23 SS – FEM

The Diploma Thesis Supervisor

Ing. Josef Pavlíček, Ph.D.

Supervising department

Department of Information Engineering

Electronic approval: 31. 10. 2022

Ing. Martin Pelikán, Ph.D.

Head of department

Electronic approval: 28. 11. 2022

doc. Ing. Tomáš Šubrt, Ph.D.

Dean

Prague on 25. 03. 2023

Acknowledgement

I would like to thank my thesis supervisor Ing. Josef Pavlíček, Ph.D. for his approval and support throughout the course of this research.

I would like to extend my deepest gratitude to my thesis advisor, Ing. Martin Čejka, for his unwavering support, patience, and advice during this investigation. His knowledge, experience, and helpful critique were crucial in forming this argument.

My sincere appreciation goes out to my Siemens colleague Ing. José Luis Samper Escudero, Ph.D. for their unwavering support, encouragement, and understanding during the challenging times of this research.

I would like to acknowledge the staff and faculty of Czech University of Life Sciences Prague, who have provided me with a stimulating and enriching academic environment. Their dedication to teaching and research has been truly inspiring.

I would like to also thank my parents, for their unwavering support, encouragement, and love throughout my academic journey. Their sacrifices and belief in me have been the driving force behind my success.

Finally, I would like to express my sincere gratitude to all the participants who took part in my study. Without their willingness to share their experiences and perspectives, this research would not have been possible.

Non-Disclosure Agreement

This report on my master's thesis contains confidential data of Siemens AG. The distribution of the contents of this thesis in whole or in part as well as making of copies or transcripts - also in digital form - is only permitted with the written consent of the responsible person of the Siemens Application Centre DI FA PMA APC 3. Within the scope of the practical project at the Czech University of Life Sciences Prague, only persons involved in the examination are authorized to view the work without the above-mentioned written consent.

VIABILITY OF SIEMENS INDUSTRIAL EDGE TO DEPLOY AI INTO AUTOMATION PROCESSES

Abstracts

The study explores the potential of Siemens Industrial Edge to integrate artificial intelligence (AI) methods into automation processes for Vertical Form Fill Seal (VFFS) machines. The focus is on the deployment of a Convolutional Neural Networks (CNN) for production predictive purposes. The research analyses the benefits of utilizing Siemens Industrial Edge in terms of predicting the quality of packaging belonging to different class. The study also discusses the challenges of implementing AI into automation processes and the importance of data accuracy in predictive maintenance. The findings suggest that deploying Siemens Industrial Edge with AI for predictive maintenance can significantly improve the performance and efficiency of VFFS machines. Overall, the research provides insights into the viability of Siemens Industrial Edge in deploying AI for automation processes and highlights the potential benefits of using Machine learning for predictive maintenance in industrial settings.

Keywords:

Siemens, Industrial Edge, Artificial Intelligence, Automation, Predictive maintenance, Industry 4.0, Viability, Industry 4.0, Edge computing, Edge devices, Real-time analytics, Data processing, Machine learning, Industrial IoT, Smart manufacturing, Operational efficiency.

ŽIVOTNOST SPOLEČNOSTI SIEMENS INDUSTRIAL EDGE PŘI NAVÁDĚNÍ AI DO PROCESŮ AUTOMATIZACE

Abstrakt

Studie zkoumá potenciál Siemens Industrial Edge integrovat metody umělé inteligence (AI) do automatizačních procesů pro stroje Vertical Form Fill Seal (VFFS). Důraz je kladen na nasazení konvolučních neuronových sítí (CNN) pro účely predikce výroby. Výzkum analyzuje výhody využití Siemens Industrial Edge z hlediska predikce kvality obalů patřících do jiné třídy. Studie také pojednává o problémech implementace AI do automatizačních procesů a důležitosti přesnosti dat v prediktivní údržbě. Zjištění naznačují, že nasazení Siemens Industrial Edge s AI pro prediktivní údržbu může výrazně zlepšit výkon a efektivitu strojů VFFS. Celkově výzkum poskytuje vhled do životaschopnosti Siemens Industrial Edge při nasazování AI pro automatizační procesy a zdůrazňuje potenciální výhody používání strojového učení pro prediktivní údržbu v průmyslových prostředích.

Klíčová slova:

Siemens, Industrial Edge, Umělá inteligence, Automatizace, Prediktivní údržba, Průmysl 4.0, Životaschopnost, Průmysl 4.0, Edge computing, Edge zařízení, Analytika v reálném čase, Zpracování dat, Strojové učení, Průmyslový IoT, Chytrá výroba, Provozní efektivita.

1 Content

Introduction

Manufacturing companies are challenged to succeed in dynamic international markets requesting high-quality products, flexibility, on-time delivery, and a reasonable cost structure [1]. A typical example of this is a Vertical Form Seal machine (VFFS) automated packaging machine used in various industries for packaging wide range of products including food, pharmaceutical, beverage etc [2].

In Packaging machines, the quality of the packaging is a big challenge to deal with the current pace of on-time delivery and standard quality. Standard quality is the need of the hour in the current market. Viking Masek is one of the latest VFFS machines which come in different size and configurations [3]. Viking Masek utilization of cutting-edge technologies in their packaging equipment to boost effectiveness, productivity, and quality is referred to as having an "industrial edge" in this context.

HMI is utilized in Viking Masek packaging machines to offer a graphical user interface that enables operators to communicate with the machine and manage its operations. Typically, the HMI system comprises of a touch screen display that shows operator the details about the machine's state, the settings that are in effect, and any alerts or warnings.

Figure 1 : Human Machine Interface (HMI)

Above Figure 1 shows HMI interface with inputs. However, current HMI doesn't show the packaging quality and doesn't test limitations of Industrial edge to deploy Artificial

Intelligence. With the increasing development of machine learning (ML), prediction models are becoming more and more established in the field. Looking from the business perspective, an AI model to predict the packaging quality and show the visualization on HMI will make the operator aware of the packaging standards and will reduce the loss and time. It will enhance the packaging procedures by enabling real-time data processing.

Thus, our manuscript aims to set up an ML-based model for the prediction of quality of packaging classified into multi classes. With the achieved results, the paper provides three main contributions:

- Collecting the data from the machine directly and analysing using matrix and graphs in python to extract the dependent features.

- Proposing the model and discussing various Hyperparameters techniques to train the machine learning model.

- Finally, Test the viability of industrial edge to deploy automation process by deploying on the industrial edge to predict the packaging class with a user interface for the operator.

Our paper is structured as follows in brief. Section 2 is divided into various sub sections Section. Section 2.2 talks about data processing. Section 2.3 first discuss about the current neural networks with different algorithms. Section 2.4 discusses about choosing CNN for the problem. Section 2.5 talks little bit about Siemens Automation Portfolio. Section 2.6 defines the problem statement and Section 2.6.1 talks in detail about its implementation. Finally, Section 3 shows the result and discussion and Section 4 talks about the final concluded result.

Objectives and Methodology

1.1 Objectives

The overall objectives of the project include:

The initial goal is to research and comprehends Industrial Edge, a Siemens IoT solution, and its potential. The usage of this approach to implement AI techniques in automation is the focus. The goal is to assess the solution's capacity to assist AI in automated activities.

The second goal is to determine the prerequisites and crucial variables needed to implement AI techniques in automation for predictive maintenance. This entails gathering information from machines and examining crucial elements. The objective is to identify the crucial variables that are essential for the effective implementation of AI techniques for predictive maintenance.

The third goal is to identify various architectures that Siemens Industrial Edge may utilize to implement AI techniques. To determine the class of the packing, this analysis also looks at various potential multiclass classification-capable architectures. The goal is to identify the best architecture for using AI techniques in the Siemens Industrial Edge environment.

Building the architecture(s) mentioned in the third aim to assess the critical parameters is the fourth goal. To determine the suggested parameters, this entails finding and training the neural network using hyperparameter optimization. The goal is to make sure that the created architecture can enable AI in automation operations successfully.

The goal is to assess and discuss the viability and circumstances in which Industrial Edge is appropriate for AI. This entails deploying AI on the edge using the right architecture, testing the outcomes, and drawing conclusions. The goal is to ascertain whether Industrial Edge is a workable solution for implementing AI techniques in automation and whether there are any requirements that must be satisfied for its implementation.

1.2 Methodology

The project focuses on investigating the potential use of an existing technological solution from siemens to support the adoption of a new IoT technology in Industry 4.0. The study falls under the category of applied research, which tries to address current issues and develop workable answers.

Building expertise in siemens Industrial Edge and AI for Automation is the initial part of the project. This stage seeks to research Industrial Edge's most recent AI architectures and features, which is the product where the new technology will be showcased.
Understanding the needs for predictive maintenance in an industrial setting and creating a successful AI-based solution require this knowledge.

Critical points and requirements for predictive maintenance in an industrial setting will be determined in the second step. This entails gathering real-time data from a Siemens packaging machine demo dubbed Viking Masek. The internal data of the company's AI-based solutions can be used to determine the needs and important aspects of the process, such as the cycle of operations, the frequency of the process, and the type of synchronization. This phase's goal is to identify the conditions and process elements that are crucial in determining whether Edge is a good candidate for AI deployment.

The critical parameters that were identified in stage two are discovered in the third stage of the project through the definition, construction, and execution of numerous tests. With the collected data, for instance, tests like Correlation distribution, PCA, and ICA will be run. Based on the findings, a neural network will be suggested. Only those elements of this neural network that are crucial for assessing predictive maintenance and the application of AI should be the focus of its definition.

The planned pipelines will be assessed, and their viability will be reviewed in the project's fourth and last stage. The best critical parameters for predictive maintenance will also be described in this discussion. The proposed architecture will also be put into use on Industrial Edge, and tests will be run to determine and discuss its practicality. These tests' findings will be used to the planned architecture to improve efficiency.

To investigate the potential of a new technology for IoT in Industry 4.0, the research will ultimately employ an existing technological solution from siemens. The project's four main steps include developing expertise in siemens Industrial Edge and AI for Automation, identifying critical areas and demands, carrying out experiments, and evaluating suggested pipelines and architecture.

2 Literature Review

2.1.1 Literature survey

The history of using machine learning (ML) and artificial intelligence (AI) in predictive maintenance of vertical form-fill-seal (VFFS) machines dates to the early 2000s. In recent years, the use of convolutional neural networks (CNNs) has emerged as a popular choice for predictive maintenance of VFFS machines. CNNs have been used to analyse data from various sensors installed on the machines to detect anomalies and predict the remaining useful life (RUL) of the machine components.

One of the earliest studies that used AI for predictive maintenance of VFFS machines was done in 2001 by Rutenbar et al [4]. They used an ML technique called support vector machine (SVM) to classify the machine health status as normal or faulty. Later, in 2012, Li et al. used a combination of principal component analysis (PCA) and decision tree analysis (DTA) to predict the RUL of a VFFS machine [5].

In recent years, there has been a surge of research on using CNNs for predictive maintenance of VFFS machines. For instance, in 2018, Kou et al. used a CNN to classify the health status of the machines based on vibration signals. They achieved a classification accuracy of 96% [6]. Similarly, in 2020 Jia et al. used a deep learning model that combined a CNN and long short-term memory (LSTM) networks to predict the RUL of VFFS machines based on acoustic signals [7].

The deployment of AI and CNN models for predictive maintenance in VFFS machines has been a subject of interest in recent years. By deploying the models on the edge, the computational load is reduced, and the response time is improved. In 2021, Chen et al. proposed an edge-based predictive maintenance framework that uses a CNN to analyse vibration signals and predict the RUL of VFFS machines. The framework was tested on a real-world dataset and achieved a prediction accuracy of 90% [8].

Also in 2021, Zhang et al. proposed a method for the deployment of an AI model on the industrial edge for predictive maintenance of VFFS machines [9]. The authors used a CNN model to predict the remaining useful life of the machine and demonstrated the effectiveness of their proposed approach in reducing maintenance costs.

For example, in a study by Jin et al. in 2021, a CNN-based method was proposed for detecting defects in vertical form fill seal (VFFS) packages. The authors demonstrated the effectiveness of their proposed method on a real-world dataset, achieving an accuracy of 98.86% in defect detection [10].

In a study by Xie et al. in 2021, a CNN-based method was proposed for predictive maintenance in the VFFS packaging process [11]. The authors deployed the model on an industrial edge device and achieved an accuracy of 96.8% in predicting packaging defects. The deployment of the model on the industrial edge device reduced the communication latency and improved the response time, making it suitable for real-time predictive maintenance.

The use of AI, specifically CNNs, for predictive maintenance of VFFS machines has a long history, with recent advances focusing on deploying the models on the industrial edge. With the ever-increasing demand for efficient and cost-effective maintenance practices, it is expected that the use of AI in VFFS predictive maintenance will continue to grow.

2.2 Data processing

Data Processing refers to manipulation and transformation of data to provide useful information. It involves data cleaning, integration, transformation, analysis, and visualization [12].

Data Processing is an essential component of both data science and data analysis. Data Science uses statistical and computational methods to extract insights from data. It works with large and complex datasets and provides variety of tools and techniques to extract meaningful patterns and relationships. It involves use of machine learning, artificial intelligence, and other advanced techniques to build effective models with can identify patterns and trends in data.

Data Analyses focus on working with data using statistical method to analyse and extract insights. It often works smaller datasets like databases, spreadsheet, and statistical software.
These are very important steps in analysing data because both provides essential tools to process. Some commonly used tools include:

1. Programming languages: Programming languages like Python and R are the two of the most popular languages which offers range of libraries and tools for data processing and analysis.

2. Statistical Software: Software includes SAS, SPPS are commonly used for statistical analyses.

3. Machine Learning Framework: Tools like scikit-learn, TensorFlow are used to build predictive models which performs advanced machine leaning task.

4. Data Visualization: Tools like matplotlib, seaborn are used to visualize data to extract useful information's.

In general, all these tools are the building blocks combined and work together to build a predictive machine learning model which can perform high level task efficiently. Later, data will be analysed using all these tools and a predictive model will be proposed.

2.3 Neural Network

Neural networks are set of algorithms, inspired, and modelled after a human brain, that are designed to learn and recognize patters [13].

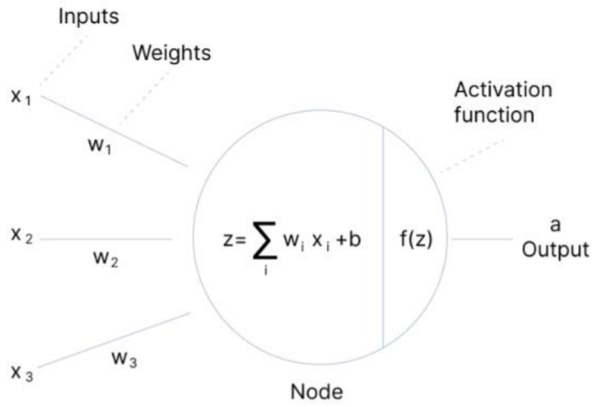

Figure 2: Artificial Neuron

Above is the visualization of the basic structure of a neuron or a node. It is a basic computational unit that receives one or more inputs and perform calculation to produce an output. In short, it is a place where mathematical computation take place closely related to a human brain. It consists of mainly their main components: inputs, weights, and an activation function as shown. They are the building blocks of a fully connected neural network.

Neural Network compromises of layers made of neurons or nodes. The number of layers may vary based on the data complexity and data dimensions. Each layer output is simultaneously and subsequently layer input.

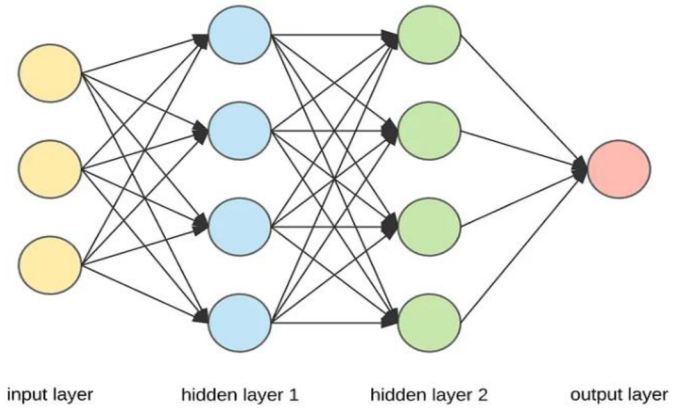

Figure 3:Nodes and layers of Neural Network

As seen from above visualization every circle represents a node in their respective layers. It is constructed from three types of layers.

1. Input Layer: The three yellow circles shown, acts as an input for the neural network in the form of $Z = w_0 + w_1x_1 + w_2x_2 + ...+ w_nx_n$
   Here w_0 is also called Bias or intercept represented as b in the figure above. The weights(w) are multiplied to the inputs(x), mathematical calculations are performed, and output of this layer is acting as an input for the hidden layers.

2. Hidden Layers: Its acts and intermediate layers between input and output layers represented in blue and green circles where all the permutation takes place. No of hidden layers may vary based on the hyper-parameter tuning that provides best accuracy avoiding underfitting and overfitting.

3. Output Layer: It produces the output for the network. The number of nodes in this layer may vary based on regression (predicts continuous quantity) or classification (predicts discreate class labels) problems which will be discussed later.

Neural network is also having activation function as shown above represented by f(z). This decide whether if the neuron will be activated or not. In short, it decides whether the neuron output is important in the process of prediction or not.

More specifically, in and activation function f(z) there is a threshold parameter that determine if a neuron should produce an output or not, or the neuron output should switch from being inactive (0) to active (1) or not. Let's understand with an example.

Let's take a simple neuron two input and threshold of 0.5 having following specifications.

input1=0.6, weight1=0.3, input1=0.4, weight2= 0.5
weighted sum= (input1 * weight1) + (input2 *weight2) = 0.47

The output of the neuron is determined by passing weighted sum through activation function. Let's take sigmoid as an example:
Sigmoid is represented by.

Sigmoid(x)=1/(1+exp(-x)), produces an output between 0 and 1.

Output = sigmoid (0.47) =0.615

Since, the output is greater than threshold of 0.5, the neuron will fire and produce and output of 1.
The choice of threshold and activation function plays a significant impact on its performance and ability to learn complex in input data.

Typically, activation functions are of two types [14]:

1. Linear Activation Function: As the name suggests, the function is a line or linear. This makes the output of the function not to be confined between any range between (-infinity to infinity).

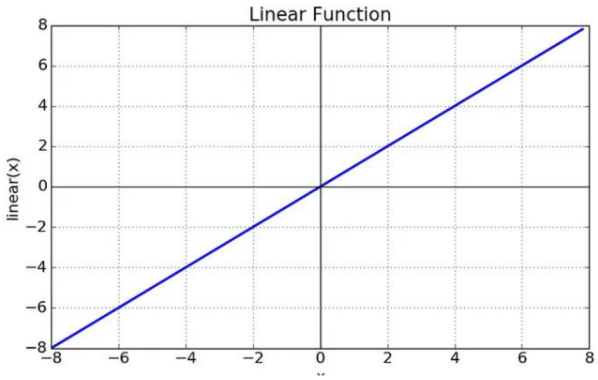

Figure 4: Linear Activation Function

2. Non-linear Activation Function: These functions are the most used activation functions which can adapt with the variety of data and differentiate between the outputs.

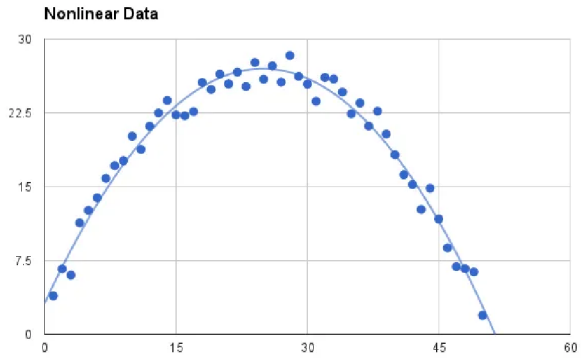

Figure 5: Non-Linear Activation Function

There are many non-linear functions like sigmoid, tanh, relu etc which are divided on the bases of the range. Below is the visualization representing those functions in brief.

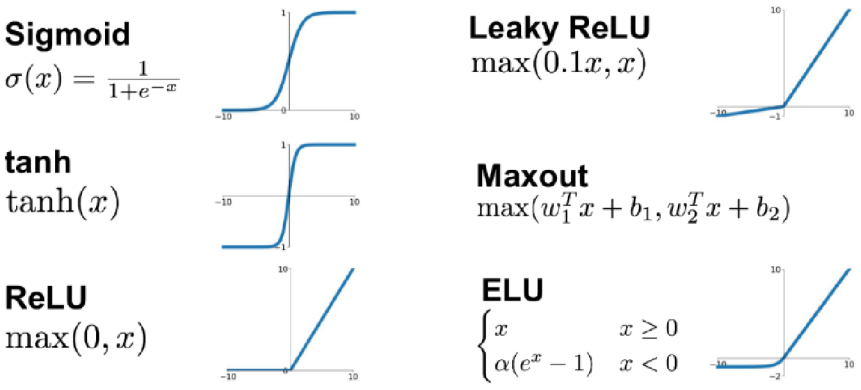

Figure 6: Other Activation Functions

These functions are selected based on different types of neural network needs to be trained. There are many types of neural network. Let's have a look.

2.3.1 Types of Neural Networks

The history of neural network dates to 1940s when first neural network model was proposed. However, the network was relatively simple with only few layers of neurons and no ability to learn from data. It was in 1980s when development of backpropagation algorithm came into existence which allow neural network to learn from data and adjust weights and biases to improve the performance. However, backpropagation networks were limited in their ability to process images and other form of spatial data. To address this problem, the first Convolutional neural network (CNN) was developed in 1990s.

There are many types of neural network available. They are classified depending on their structure, no of neurons, layers, activation function etc [15]. Let's look at some of them.

1. Recurrent Neural Network: Recurrent Neural Network (RNN) solves short-term memory problem due to vanishing gradient when working with long data sequence. RNNs create a cycle or a loop where the output of a particular layer is fed back as an input again to predict output of the layer.

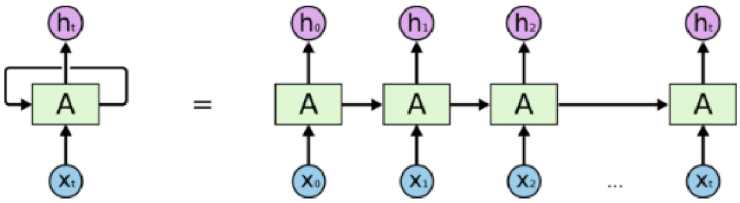

Figure 7:Recurrent Architecture

Consider the figure above, at X1 time the inputs are X0 which is the previous state as well as input of X1 which is the current state. This creates a cycle or a loop inside the network.

The main advantage of this network is it also stores previous input state in the short-term memory not only the current state. This make RNN to consider previous state as well as current state.

The major drawback of RNN is it requires more training data to learn effectively which makes it slower. It is also not suited for classification task where input data is not in sequence. It doesn't process sequential data using RELU as an activation function.

2. Feed Forward Neural Network: It is one of the simpler types of Feed-Forward neural network (FFNNs). It conveys information only in one direction through its node until it reaches the output node. It doesn't have short term memory, so it doesn't create loop like RNNs. It only follows one- way propagation.

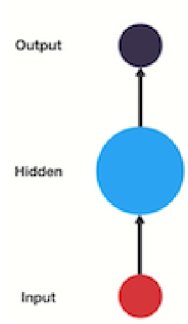

Figure 8:Feed-Forward Neural Network

Hidden layers may be present or not, but input and output layers are present. Based on this, it is classified as a single layered or multi layered feed forward neural network.

The main advantage of FFNNs is it is fast, speedy, easy to design and maintain.
The main disadvantage of FFNNs is it cannot be used for deep learning due to lack of dense layer and back propagation.

3. Multilayer perceptron: Multilayer perceptron's has input and output layers with many hidden layers inside it. They are also called Feedforward algorithms because inputs are multiplied with weighted sum and subjected to activation function, like a perceptron. In sort, each layer is feeding the result of their computation to the next layer. Hence, all goes through the hidden layer to the output layer at last.

If the algorithm computes only the weighted sum of the input(x) and weights(w) in each neuron and propagates through the output layer and stopped, it will not be able to learn the weights that minimize the cost function. In short, if it computes only one iteration then it will not learn which will result in zero learning. Therefore, backpropagation come in handy and plays a big role in training the neural network.

Backpropagation is a leaning mechanism that allows multilayer perceptron's to adjust the weights which helps in reducing the cost function. It is used in supervised learning.

Supervised learning is a well 'labelled' which means some data in dataset is already targeted with correct answers which helps the neural network to predict the outcomes for unforeseen data Semi- supervised and self- supervised leaning where it relies on partially or "unlabelled" data to learn patterns and relationship in data.

In semi-supervised learning a small amount of labelled data is used along with unlabelled data to train the model. Model can use information in unlabelled data to learn more generalizable features to improve the performance on labelled data. It is used in case when labelling data is expensive and time consuming.

In case of self-supervised learning a model is trained to predict certain features of data without explicit labels. For example, in image processing, a model may be trained to predict location of the cropped-out portion of the image.
They are particularly useful in cases where labelled data is insufficient and expensive to obtain.

Below is the visualization of backpropagation within neural network.

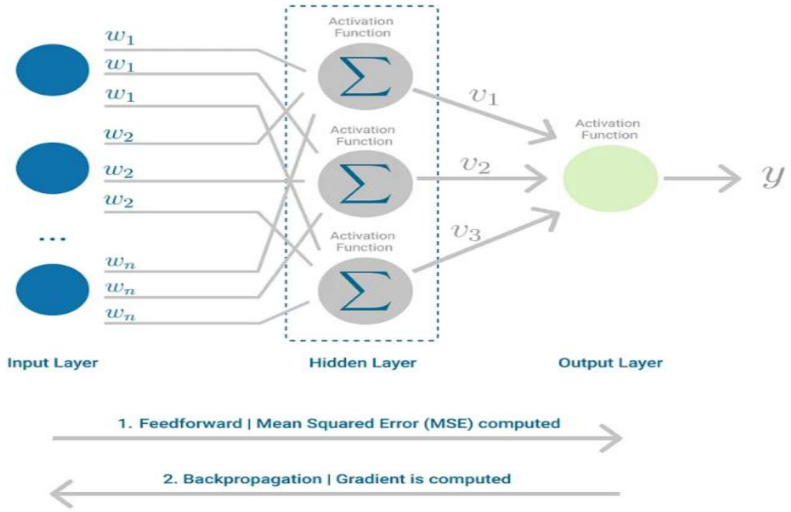

Figure 9: Backpropagation

When the weighted sum is forwarded through all the layers, gradient of MSE as shown above is calculated across all inputs and output pairs. Then to propagate it back, the weights of the first hidden layer are updated with gradient value and the weights are propagated it back to the starting point of the network. This process is repeated based on no of epochs which increases the model accuracy. This is how backpropagation works inside the neural network.

The main disadvantage of neural network is it is comparatively slow because it depends on number of hidden layers.

2.3.2 Regression and Classification Algorithm

Regression and Classification are the most popular machine leaning algorithms which follow supervised learning algorithm (Labelled datasets) [16]. Regression and Classification can be better understood by a figure shown below.

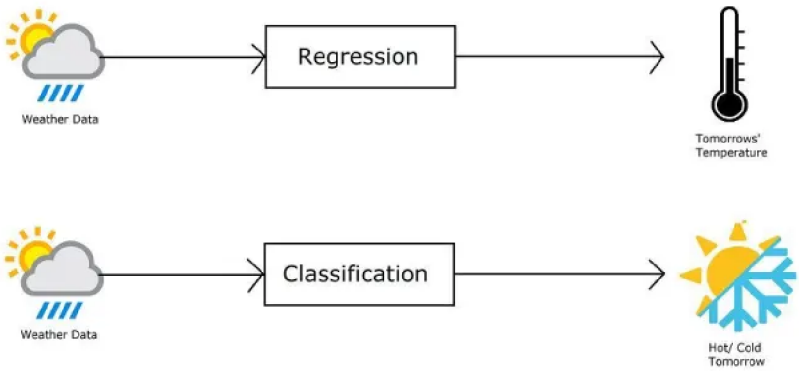

Figure 10: Regression and Classification

Regression is used for continuous value such as speed, weight, temperature etc. As shown in the diagram if you want to predict what will be the temperature for tomorrow using temperature datasets, regression algorithms are used. The model using regression will learn from the existing temperature dataset to predict the temperature for future dates. It predicts a single output value using training dataset.

There are many types of regression models like linear Regression, Polynomial Regression, Logistics Regression, lasso Regression etc. The most popular ones are linear Regression and Logistics Regression.

1. Linear Regression: It is used to predict the value of a variable based on another variable. The variable to be predicted is called dependent variable and variable used to predict the value is called independent variables.

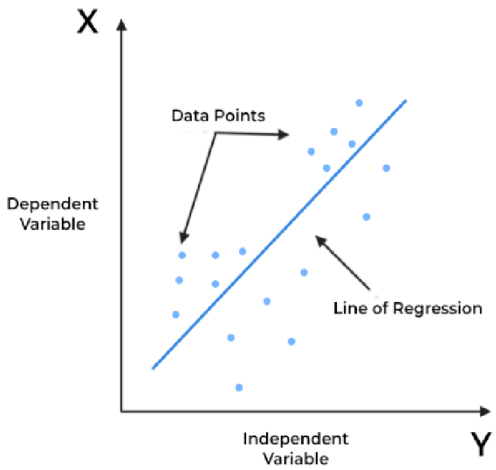

Figure 11: Linear Regression

2. Logistic Regression: This algorithm is used to find the probability of an event to occur based on the dataset. It uses independent variables from the dataset to predict the probability of the dependent variable which is between 0 and 1.

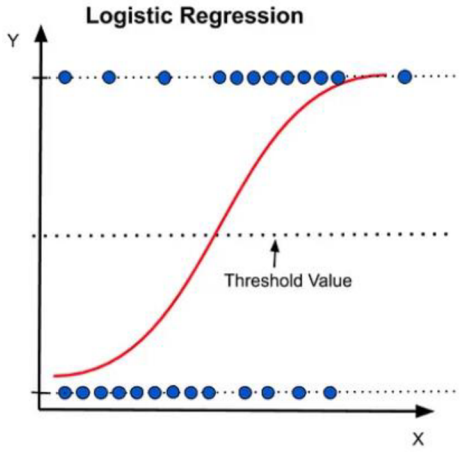

Figure 12: Logistic Regression

While classification algorithms are used to predict discrete values or classify if it is going to be hot or cold, or will it rain or not. If the output is between two distinct classes, then it called Binary classification. If the output is between more than two classes, then it is called multiclass classification. Below is the visualization of binary and multiclass classification.

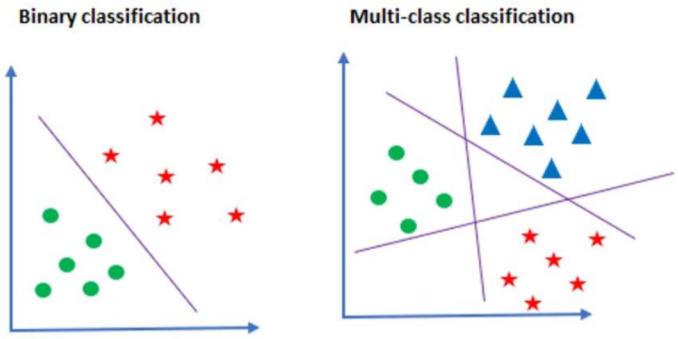

Figure 13: Binary and Multiclass Classification

The different colour shapes represent different classes. Binary classification has two classes, so the output predicted will be between these two classes. While multiclass classification has three different colour shapes representing three different classes, so the output will be between three different classes.

2.4 Convolutional neural networks

Convolutional neural network (CNN) is a Deep learning algorithm and a Feed- forward neural network that takes inputs as image, multivariate /univariate time series [17].

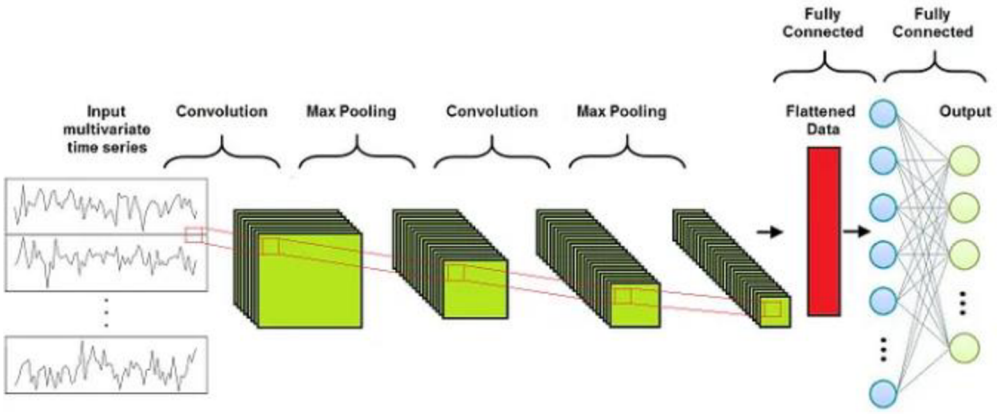

Figure 14:CNN Model Architecture

Univariate timeseries means only one variable over time. For example, torque collected from a machine every millisecond. Every millisecond it will have one-dimension value. Multivariate timeseries means multiple variables over time. For example, torque, temperature, speed etc every millisecond. It has more than one dimensional value.

CNN is mainly composed of three different layers:

1. Convolutional layer
2. Pooling Layer
3. Fully Connected Layer

Convolutional and Pooling layer can be altered before the output reaches to Fully connected layer.

1. Convolutional Layer: It is a building block of this type of network. It performs convolution of an input series of feature maps with a filter matrix to get a different series of feature of maps. The main purpose of this layer is to extract the high-level features. This layer has set of filters that are fixed size matrices applied to submatrix of the input with it same size.

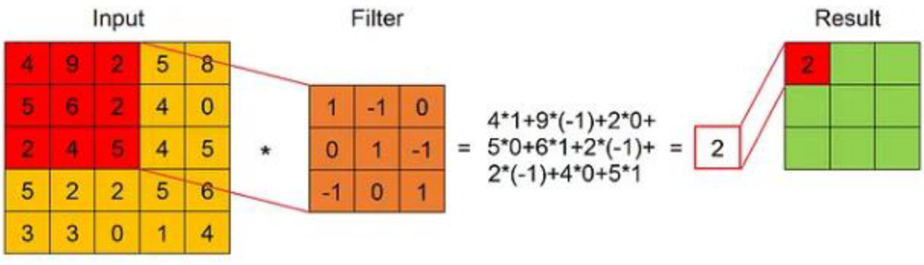

Figure 15: CNN Filter Logic

As seen from above fig, sum of the product of every element of the filter is placed in the same position of sub matrix. The result is shown below.

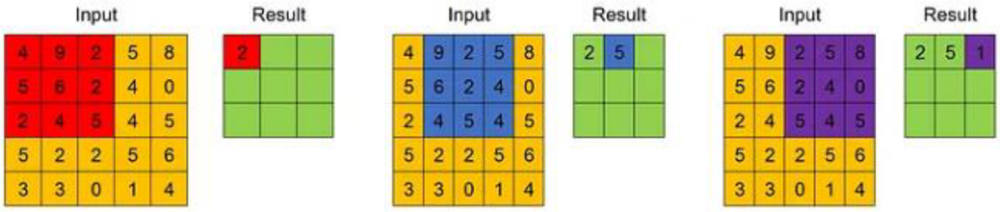

Figure 16:CNN Filter Result

Two other important parameters that must be chosen are Stride and Padding.

Stride: It controls how many units the filter will shift around one input feature map. Below is the figure representing an example of stride.

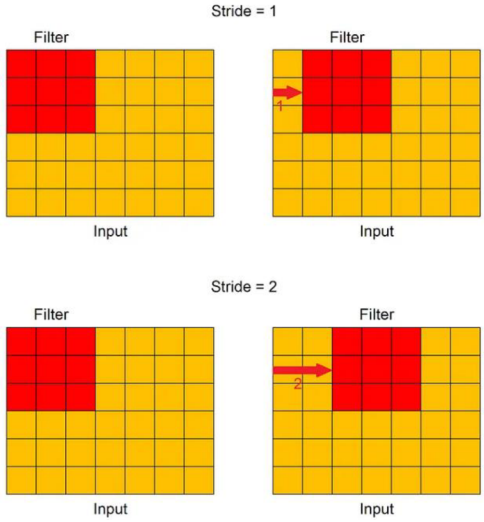

Figure 17:CNN Stride

Padding:

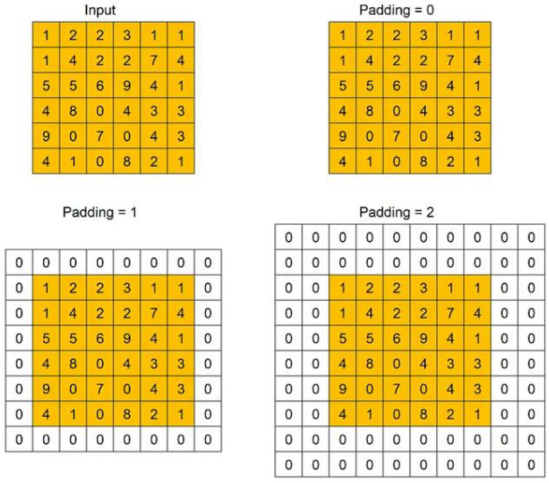

Figure 18:CNN Padding

Padding tells how many extra rows and columns to be added outside an input feature map. It is done before applying convolutional filter. It is usually filled with dummy values usually with 0 as shown in the figure.

Padding is applied because after applying convolutional filter to just an input feature map, the size decreases. So, if, many filters are applied to all input feature map the size become too small. Applying padding add extra rows and columns we preserve the original size. There are two types of Padding:

a. Valid Padding: When the size of the feature map is smaller after applying convolution filter then the size of input feature map.

b. Same Padding: When the size of the feature map is equal or greater after applying convolution filter then the size of input feature map.

2. Pooling Layer: The main purpose of the layer is reducing the dimension of feature map as much as possible. It is useful for extracting important and dominant features where the input feature map is different from output series of feature maps.

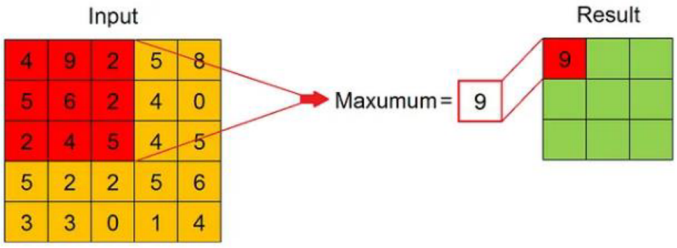

Figure 19:CNN Pooling Layer

There are two types pooling:

a. Max Pooling
b. Average Pooling

Max Pooling: It takes the maximum value within each region from a feature map to form a smaller output feature map. For example, a 2*2 max pooling applied to 4*4 input feature map. For each region, the maximum value is computed, and the result is 2*2 output feature map, where each element is the maximum value within 2*2 output feature map.

Average Pooling: It takes the average value within each region from a feature map to form a smaller output feature map. For example, a 2*2 max pooling applied to 4*4 input feature map. For each region, the average value is computed, and the result is 2*2 output feature map, where each element is the average value within 2*2 output feature map.

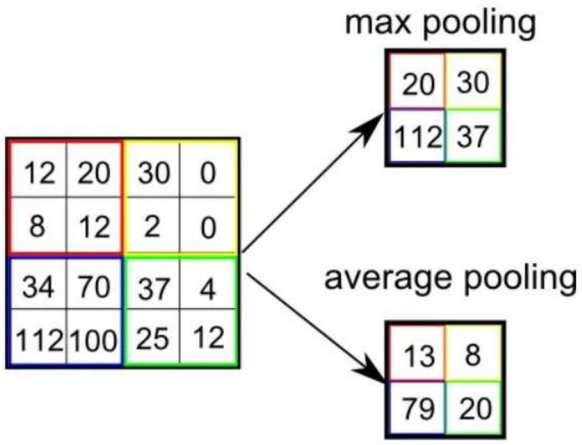

Figure 20:Max pooling and Average Pooling

The main objective is to reduce the size of the feature maps for faster computation because it reduces the number of parameters maintaining the maximum features.

3. Fully Connected Layer: Its main goal is to learn non-linear combinations represented by output of convolutional layer and pooling layer as show in the figure above. It is usually implemented using a multi-layer perceptron.

All the feature maps are flattened as shown in the above fig before representing as a final input to multi-layer perceptron. Multi-layer perceptron final output is equal to the number of classes specified.

The main advantages are backpropagation which is applied during the training based on no of epochs mentioned. Hence after backpropagation, the model will be able to extract dominant features and will be able to classify them.

Here is the working code of 1-D Convolutional Neural Network.

```
# 1D-CNN    //coded by Pankaj
def make_model(input_shape):
    input_layer = keras.layers.Input(input_shape)

    conv1 = keras.layers.Conv1D(filters=64, kernel_size=3, padding="same")(input_layer)
    conv1 = keras.layers.BatchNormalization()(conv1)
    conv1 = keras.layers.ReLU()(conv1)

    conv2 = keras.layers.Conv1D(filters=64, kernel_size=3, padding="same")(conv1)
    conv2 = keras.layers.BatchNormalization()(conv2)
    conv2 = keras.layers.ReLU()(conv2)

    conv3 = keras.layers.Conv1D(filters=64, kernel_size=3, padding="same")(conv2)
    conv3 = keras.layers.BatchNormalization()(conv3)
    conv3 = keras.layers.ReLU()(conv3)

    gap = keras.layers.GlobalAveragePooling1D()(conv3)
    output_layer = keras.layers.Dense(num_classes, activation="softmax")(gap)

    return keras.models.Model(inputs=input_layer, outputs=output_layer)
```

Figure 21:CNN Implementation

The above code shows three 1-D CNN layer with its own activation function RELU and batch normalization. Batch Normalization in every layer helps to train neural network much faster through normalization of the layers by re-centring and re-scaling the data in every output node making mean as 0 and standard deviation as 1.

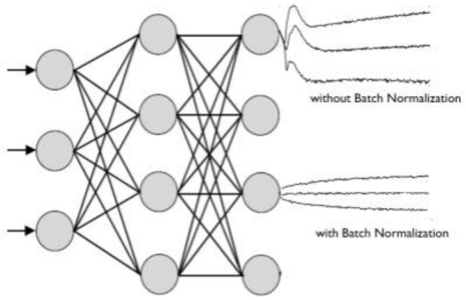

Figure 22:Node output with and without Batch Normalization

The output layer has Dense layer with num_classes which is 3 in our case with SoftMax has an activation function. This layer is responsible for the prediction of output in terms of probability between each class. The class with highest probability is generally consider the final output of the neural network.

Over the years Neural networks have improved. CNN have seen recent improvements over the years like more computationally efficient and accurate [18]. Neural networks like Generative Adversarial Network (GAN) are getting popular because is it used to generate new data that is like the training data [19].

2.4.1 Signal processing

The essential functions of signal processing include signal analysis, interpretation, and manipulation. To achieve the required shaping of the signal at the output, the fundamental nature of the signal is modified. The representation, transformation, and manipulation of signals and the information they carry are at the centre of its focus. Analog signal processing and digital signal processing are the two categories into which signal processing can be divided. [20].

Analog Signal processing: Signals are processed by analog circuits in Analog Signal processing, which change the signal in a variety of ways. Signal mixing, filtering, and other functions can be carried out by analog circuits. Applications including audio amplifiers, radio receivers, and power supply frequently make use of Analog Signal processing [21].

Digital signal processing: Signals are digitally transformed in DSP before being processed with algorithms that can carry out a variety of tasks, including filtering, modulation, and demodulation. Typically, DSP application-specific hardware and software are used for this. Several industries, including telecommunications, audio processing, image processing, and control systems, heavily rely on DSP [22].

Signal processing in VFFS Machines: A programmable logic controller (PLC) is used in the procedure to manage the different phases of the machine's functioning [23]. Signal processing, which entails converting electrical impulses into useful data that may be utilized to control the machine, is a crucial component of VFFS machines.

On VFFS machines, signal processing happens in stages. Initially, analog-to-digital converters are used to transform the analog signals from sensors, such as photoelectric sensors, load cells, and temperature sensors, into digital signals (ADCs). The PLC then applies algorithms like filtering, amplification, and modulation to the digital signals to process them. Following signal processing, the filling, sealing, and cutting operations of the VFFS machine are controlled using the processed signals. [24].

Wavelet analysis, Fourier transformations, and digital signal processing are just a few of the methods employed in VFFS machines for signal processing. These methods assist in locating patterns and trends in the signals, which can then be used to enhance the functionality of the machine [25].

2.5 Siemens Automation Portfolio

Siemens AG is a multinational Conglomerate company based in Germany and is one of the largest electrical engineering companies in the world. Its automation portfolio includes wide range of products, systems and solutions designed to optimize processes which increases efficiency in the industry [26]. Some of the key offerings in siemens automation portfolio include:

1. Siemens Simatic's: A family of (programmable logic controller) PLC's acts as a main brain to control and automate industrial processes [23].

2. Siemens totally integrated automation (TIA): It is an integrated hardware and software platform that includes PLS's, drives and software for program development. It can also provide simulations.

3. Siemens Sinamics Drives: It has range of AC and DC drives which is used to control and regulate speed of electric motors.

4. Siemens Networking Solutions: It is a family of networking component including switches, routers, and other wireless access points for automation applications.

5. Siemens Build Technologies: This portfolio includes automation of buildings which includes security, heating, ventilation, air conditioning, fire safety and other building energy managements.

6. Siemens Industries Software: A portfolio of software tools for product lifecycle management simulation, design, and engineering.

These are the just a few examples of many offerings in siemens automation portfolio. Siemens also provides automation solutions for industry. Let's have a look them.

2.5.1 Demo Packaging Machine: Vertical Form Fill Seal

Siemens provides automation solutions for Industry. Preventive and predictive maintenance can leverage machine producers and users. Thus, to illustrate that, a vertical packaging machine has been used as demo case, labelled as Viking Masek. It is a VFFS (Vertical Form Fill Seal) packaging machine.

VFFS is a type of packaging machine used in the food and beverages, pharmaceutical and other industries. It is a machine that forms a bag from a roll of flat material, fills it with a product, and seals the open end to create a finished package [27].

Figure 23: Viking Masek machine

VFFS packaging machine and one of the most complex machines and can run much faster up to 300 bpm and provide global packaging technologies which helps to hit the packaging number much faster with minimum risk, hence increase in revenue.

Above is the visualization of Viking Masek. Let's have a look at them briefly in order.

1. Food Fillers from Top: This is the essential step in packing process where wide range of products like food, beverage, cosmetic and other consumer goods are filled in the machine typically controlled by programmable logic controller (PLC) and HMI interface.

2. HMI and Edge: This section is the most important component of the machine comprising of a hardware human touch screen interface called HMI (Human machine Interface) and an 'INDUSTRIAL EDGE' run-time software environment. The operator can enter or set all parameters and packaging configuration using HMI. The machine has different state but mainly two of them are 'start' and 'stop' state. The start state let the machine in starting state by reading all the configuration set by the operator using HMI and the stop state let the machine to terminate the start state.
   This industrial edge is an open software platform that allow user with secure, scalable deployment and the execution of apps. It uses protocols like S7, SLMP, OPC-UA etc to communicate with industrial assets. HMI an Edge will be discussed in more details later.

3. Packets for Packaging: This section of machine provides the packets for packaging for the different products. Packets for Packaging depends on specific machines and types of material used. Some common types of packets produced are stand-up pouches (popular for food items to protect from moisture and lights), Three -side sealed bags (packaging products like tea, coffee spices) etc.

4. Packaging output: As seen, this section of machine act as a gateway for the finished product. The packaged product glides through and is collected from the end.

5. Sealband1 and sealband2: It refers to the band or strip of material (usually plastic) used to seal the bags which is formed. The seal band is created by heating the film and compressing between two rollers, which creates a strong and reliable seal. This is helpful in sealing the bags and keep the product inside.

6. Cross Seal Arm: This part of the machine is responsible for creating a cross seal. Cross seal consists of a heated jaw which melts the plastic and applies pressure to create a seal. This helps to cuts the packed product to be processed to the next stage.

Off course, it has many other components to look upon, but only few and important components have been outlined here.

2.5.2 Siemens Industrial Edge

Siemens industrial Edge is one of a portfolio of product, services and solutions that provides industrial automation and enhance digital capabilities for the company. It is designed to help a company move towards digitalization by providing them with tools and technologies needed to collect, analyse, and act on data generated by industrial equipment's and processes. This also includes edge computing devices, software platforms and cloud-based services which is used to implement industry 4.0 strategies and take advantage of the internet of things (IOT). This drive efficiency, improve productivity and enhance overall performance of industrial operations.

Siemens Industrial edge is designed to be highly scalable and flexible which allow companies to start small and expand their capabilities overtime based on their need and requirements change. This portfolio also provides robust security features which helps to protect sensitive industrial edge, making it more reliable and secure over time.

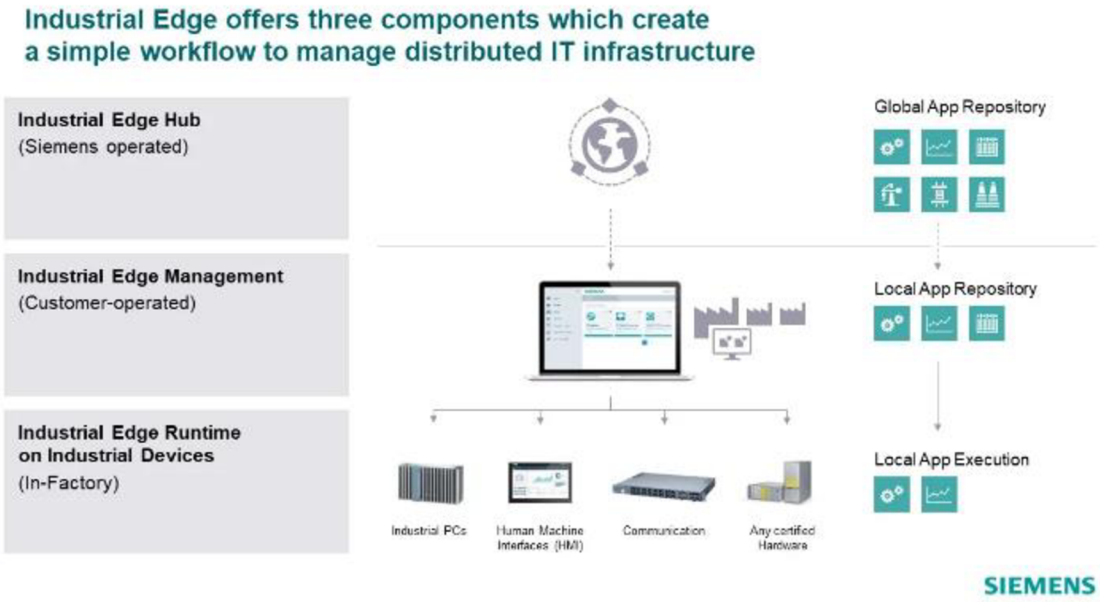

Figure 24: Siemens industrial edge workflow

Above is the image which represents the workflow of industrial edge [28]. Let's look at the components briefly.

A. Industrial-Edge hub: It is a component of siemens industrial edge portfolio which provide global app repository. It is equipped with powerful hardware and software capabilities like high performance data, such as sensors, motors, and other software platforms for data processing. It is used to collect and perform real-time data processing and make decision based on the results generated by real time Industrial equipment's.

B. Industrial-Edge Management: It is responsible for managing and monitoring edge computing devices that are deployed. These includes devices like industrial edge hub which are used to collect, analyse, and act on the data generated by the industrial machine processes. Siemens edge management also provides cloud-based management platforms which involves in deployment, configuration and monitoring the device performance and status.

C. Industrial-Edge Runtime: The siemens Industrial edge runtime provides a platform for running applications and environment for industrial edge hub where the apps are run, and process data generated by industrial equipment in real-time. This component also offers a human touch interface which is referred as HMI (Human Machine Interface) which control's industrial equipment. Industrial edge runtime from Siemens supports OPC UA protocol which is widely used in industrial environments to exchange data.

These three main components work together to manage distributed IT infrastructure.

Practical Part

2.6 Problem Statement

Predictive maintenance technique helps to determine when the maintenance should be performed. This approach promises cost and time saving.
Predictive maintenance is a technique used to detect anomalies and possible defects in equipment or in operation which can be fixed before any failure [29].

Thus, to illustrate that, a Packaging machine has been used as demo case, see Section 2.5.1. The operator can enter or set all parameters and packaging configuration using HMI (Human Machine Interface). The machine has different state but mainly two of them are 'start' and 'stop' state. The start state let the machine in starting state by reading all the configuration set by the operator using HMI and the stop state let the machine to terminate the start state. Basically, it controls machine.

HMI has all the required functionalities needed by the operator, like set the number of packets per min, for example 30bags/min,40bags/min,50bags/min etc but still it cannot let the operators know if the packing of the product is of required industrial standards! The operator is not aware of the faulty packet size, package sealing etc, hence result in loss of time and revenue.

To solve these problems, Siemens team came up with a visualization which let the operators know all the re-quired details, if the packaging is of set standards, or if the packaging can be approved and to test the flexibility of Industrial Edge Environment. A designing of AI (Artificial Intelligence) neural network was proposed which can be deployed on Siemens Industrial Edge environment and can be altered and monitor.

The neural network will specifically focus on predictive maintenance allowing the operator to check on packet's quality. The quality will be categorised into different classes given below:

Classes	Packaging
0	Valid
1	Improvable/Acceptable
2	Invalid/Rejected

Table 1: Packaging Classes

Based on the classes in which the packet belongs, the packaging quality will be determined, and the operator will be aware of the packaging quality and standards.

Of course, the initial proposed solution put out cannot be fixed at first and may be somewhat altered in response to new difficulties encountered during implementation.

2.6.1 Implementation

Based on the problem statement, siemens team came up with an implementation which is divided into following steps. Let's look at them.

2.6.2 Collecting data

The first and foremost step was to collect data from the machine. The machine works on OPC (Open Platform Communication) UA (Unified Architecture) protocol [30]. Python code was written using opcua library which helps to connect to the machine and collect the data.

The first part was to traverse through the node where each signal is located using the python code.

```
#This is for reading the SIMATIC (Viking MASEK) node path

Pankaj
def getNode(i):
    switcher = {
        0: firstNode,
        1: secondNode,
        2: thirdNode,
        3: fourthNode,
        4: fifthNode
    }
    returnNode = switcher.get(i)
    return returnNode()


Pankaj
def firstNode():
    return 'Objects'


Pankaj
def secondNode():
    return 'PLC_1'


Pankaj
def thirdNode():
    return 'DataBlocksGlobal'


Pankaj
def fourthNode():
    return 'DataToAI'


Pankaj
def fifthNode():
    return 'dataValue'
```

Figure 25:Node Path Traverse

The second part was to connect and to the machine and read the signals. The following code was used to connect to machine and log the signals.

```python
from opcua import ua

from ai.constant import nodepath


Pankaj
def getParentNodes(node, client, variables, count):
    allVariables = variables
    for childId in node.get_children():
        ch = client.get_node(childId)
        if ch.get_node_class() == ua.NodeClass.Object:
            if ch.get_browse_name().to_string().split(':')[1].__eq__(nodepath.getNode(count)):
                if len(ch.get_children()) > 0:
                    getParentNodes(ch, client, allVariables, count=count + 1)
                    break
        elif ch.get_node_class() == ua.NodeClass.Variable:
            if ch.get_browse_name().to_string().split(':')[1].__eq__(nodepath.getNode(count)):
                getChildrenNodes(ch, client, variables)
                break
            else:
                getParentNodes(ch, client, allVariables, count)

    return allVariables


Pankaj
def getChildrenNodes(node, client, variables):
    for childId in node.get_children():
        try:
            ch = client.get_node(childId)
            variables[ch] = ch.get_browse_name().to_string()
        except ua.uaerrors._auto.BadWaitingForInitialData:
            pass

    return variables
```

Figure 26: OPC UA code to connect to Packaging Machine

Above code was used to gather the signals when machine was running. Data was collected for 30 bags/ min, 40 bags/ min and 50 bags/ min.

1	
Sample	1564
X(ms) [25.11.2022 14:31:23 869 UTC]	1669386690124.686777
AxCrossSeal.ActualPosition(°)	-0.999
AxFoilFeed1.ActualPosition(mm)	22815.199
AxFoilFeed2.ActualPosition(mm)	22815.199
AxSealBand1.ActualPosition(mm)	162.792
AxSealBand2.ActualPosition(mm)	162.792
AxLinearAxis.ActualPosition(°)	-14.889
AxCrossSeal.StatusTorqueData.ActualTorque(Nm)	-14.7080100205494
AxFoilFeed1.StatusTorqueData.ActualTorque(Nm)	0
AxFoilFeed2.StatusTorqueData.ActualTorque(Nm)	0.000858764571603388
AxSealBand1.StatusTorqueData.ActualTorque(Nm)	0.000224609364522621
AxSealBand2.StatusTorqueData.ActualTorque(Nm)	-0.000898437458090484
AxLinearAxis.StatusTorqueData.ActualTorque(Nm)	0
instLFFS_VFFSMachineMasterPos.actMachineSpeed	30
2	
Sample	1565
X(ms) [25.11.2022 14:31:23 869 UTC]	1669386690129.538285
AxCrossSeal.ActualPosition(°)	-0.999
AxFoilFeed1.ActualPosition(mm)	22815.599
AxFoilFeed2.ActualPosition(mm)	22815.599
AxSealBand1.ActualPosition(mm)	163.192
AxSealBand2.ActualPosition(mm)	163.192
AxLinearAxis.ActualPosition(°)	-14.689
AxCrossSeal.StatusTorqueData.ActualTorque(Nm)	-14.7080100205494
AxFoilFeed1.StatusTorqueData.ActualTorque(Nm)	0
AxFoilFeed2.StatusTorqueData.ActualTorque(Nm)	0.000858764571603388
AxSealBand1.StatusTorqueData.ActualTorque(Nm)	0.000224609364522621
AxSealBand2.StatusTorqueData.ActualTorque(Nm)	-0.000898437458090484
AxLinearAxis.StatusTorqueData.ActualTorque(Nm)	0
instLFFS_VFFSMachineMasterPos.actMachineSpeed	30

Figure 27:Real-Time Collected Data

Above is the original data collected from the machine. Off course, this is some of data shown here not the full version. The main idea here is to show the list of signals recorded. These signals will be analysed further to find the dependent signals which will best fit the problem statements.

2.6.3 Data analysis

After data collection, Pearson Correlation was performed on collected data. Pearson Correlation indicates a number which measures strength and direction between two variables. It ranges between -1 to +1, with +1 indicates perfectly highly positively related and -1 indicates perfectly highly negatively related [31].

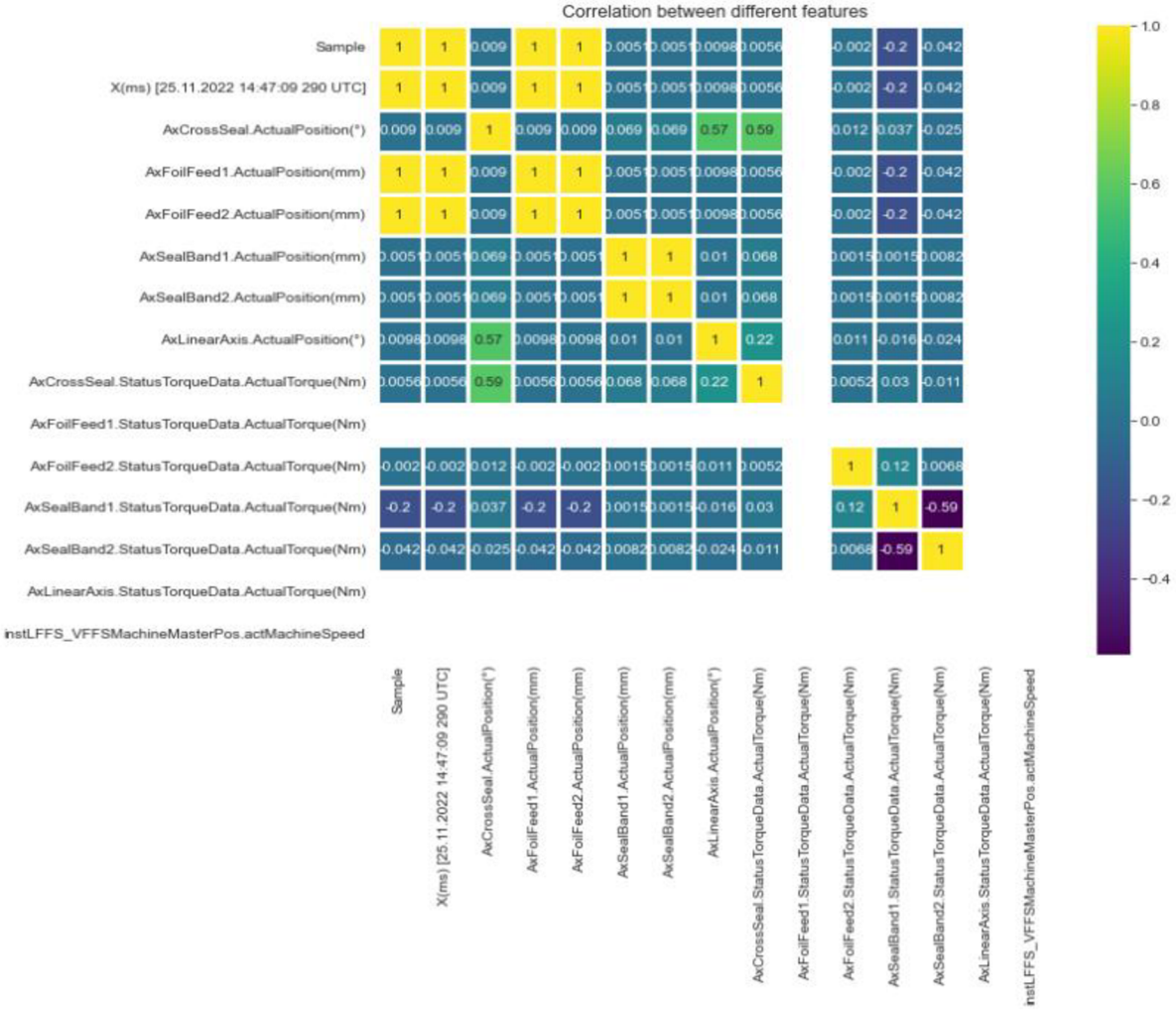

Figure 28: Correlation Matrix

Above is the correlation matrix which shows more than 50% positive correlation between CrossSeal.ActualPosition, LinearAxis. ActualPosition and CrossSeal.ActualPosition and CrossSeal.Status. TorqueDataActualTorque (NM) signals.

It was accepted that these signals will be highly positively correlated because these signals play an important role in packaging and can be realized visually.

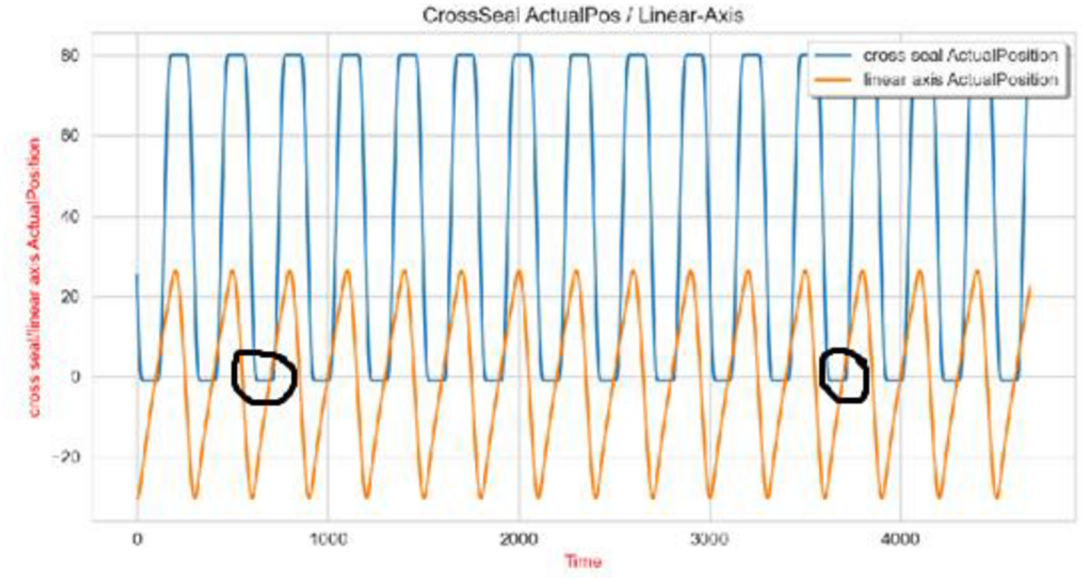

Figure 29: Cross Seal vs LinearAxis Actual Position

From the graph, it is clearly seen that as the cross seal actual position increases and reaches to the maximum value, the linear axis actual position also reaches too maximum and vice versa. Linear axis is more of a machinal system combined with cross seal as a set with operates together and both are directly proportional to each other. So, this was not the focused area which can determine the packaging quality.

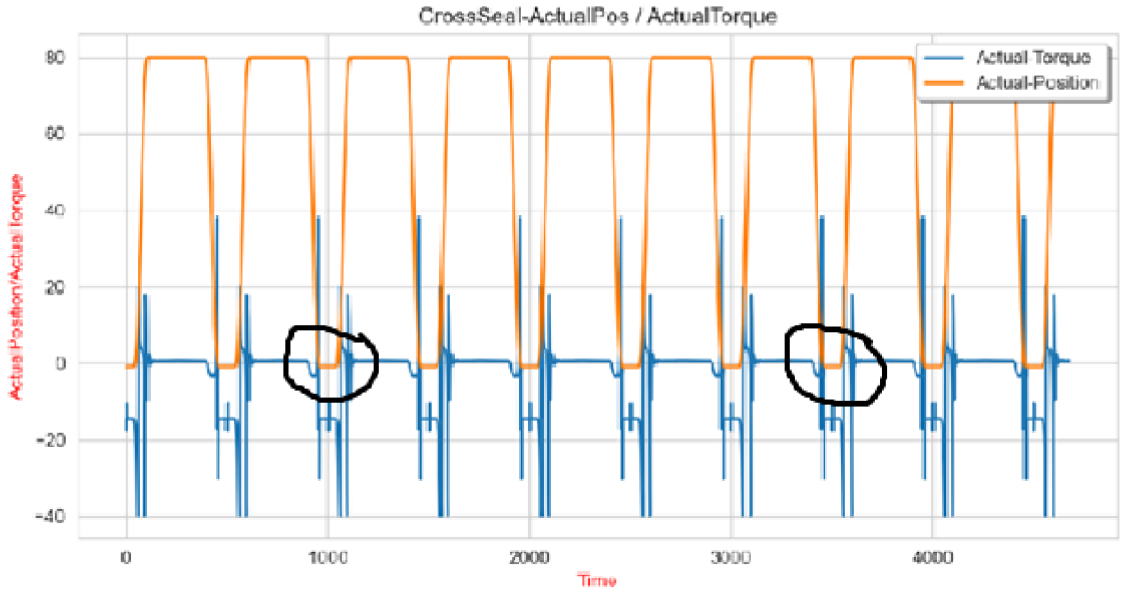

Figure 30:Cross Seal vs Torque Actual Position

From the above visualization, it can be clearly seen that as the cross-seal arm comes closer to cut the packet marked in circle as shown above, there is a torque generation which is negative in nature. As the cross-seal arm reaches to maximum distance the torque

generated is almost zero marked in circles. So, the idea was suggested to give more emphasis on the torque signal which can determine if the packaging is of set standards. In short, while cutting the packet, if the torque signal is measured and monitored, it can predict the packaging quality.

Off course, the correlation matrix will not convey all the required details between two linear dependent variables so other signals were analysed in detail too.

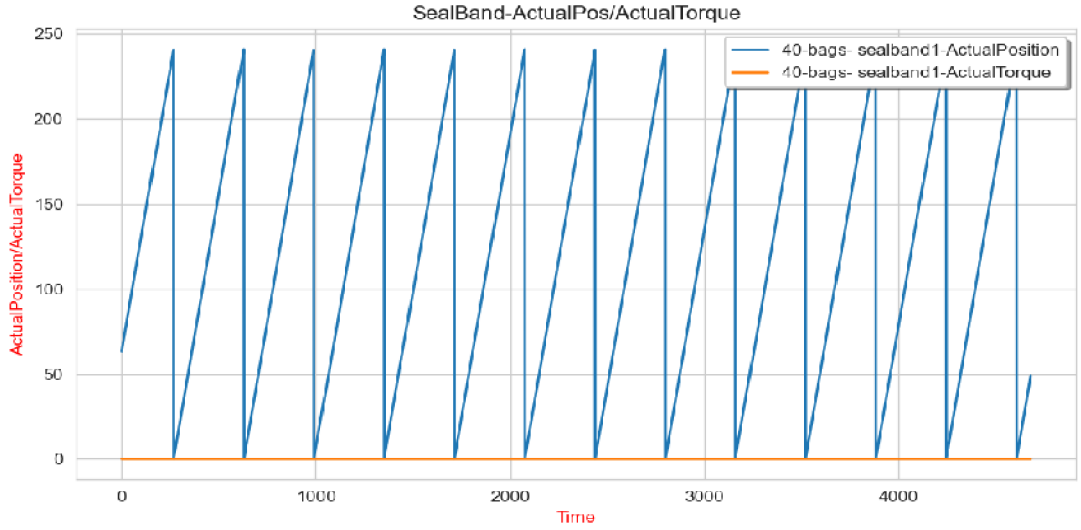

Figure 31: Seal Band vs Torque Actual Position

For example, the above visualization shows seal bands has no effect on actual torque, so this will have no impact in creating a model. Likewise, different analysis was performed, and results were analysed.

So, more emphasis was given on cross seal and torque actual position. Cross seal position was divided in t0 to t3 as one cycle. The idea was to get the torque between t0 to t3 and analyse it more. It can be seen from the visualization presented below.

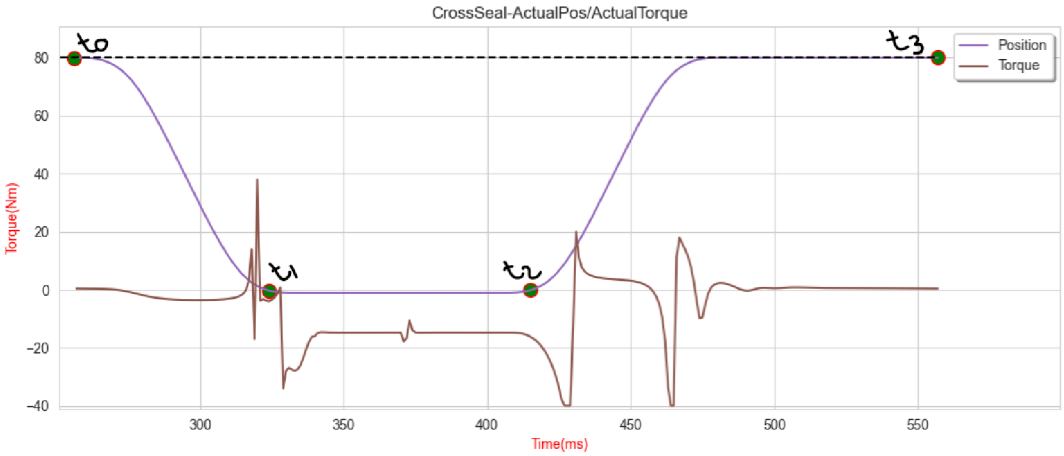

Figure 32: Cross Seal vs Torque Time Interval

The code to generate t0 to t3 cycles.

```python
#Step 2: Get first t0- t3  points
def get_Initial_t0_points():

    actual_arm_position=readData()[1]

    for firstMaxPos, elem in enumerate(actual_arm_position):  # first it should get max actual_arm_position
        if elem == np.max(actual_arm_position):
            flag = True
            break
    if flag:
        for x, elem in enumerate(actual_arm_position[firstMaxPos:]):
            if elem < np.max(actual_arm_position):
                FirstInitCut = x + firstMaxPos
                break

        for idx, elem in enumerate(actual_arm_position[FirstInitCut:]):
            if elem == np.max(actual_arm_position):
                break

        for y, elem in enumerate(actual_arm_position[idx + FirstInitCut:]):
            if elem < np.max(actual_arm_position):
                FirstEndCut = y + idx + FirstInitCut
                break
    else:
        print("ERROR:: No Fist max position Found!")
    return [FirstInitCut + 1, FirstEndCut]
```

Figure 33: Initial t0 to t3 cycles

```python
# finding all the remaining t1,t2,t3,t4 (X,Y)=(Time,Position) By Pankaj
def get_All_t0_t3_points(initCut, initEnd, lastRowNum, cycle, cycle_t0_t3_points, cycleLength):
    if lastRowNum >= initEnd:
        print("CYCLE::", cycle + 1)
        time_extracted = Time[initCut:initEnd]
        actual_arm_position_extracted = actual_arm_position[initCut - 1:initEnd]
        torque_data_extracted = torque_data[initCut:initEnd]
        k = np.max(actual_arm_position_extracted)
        t0 = t3 = 0
        for idx, elem in enumerate(actual_arm_position_extracted):
            if elem < k:
                t0 = initCut + idx
                break
        for idx, elem in enumerate(actual_arm_position_extracted[::-1]):
            if elem == k:
                t3 = initEnd - idx
                break
        initCut = t3 + 1
        time_extracted = Time[t0:t3]
        actual_arm_position_extracted = actual_arm_position[t0:t3]
        torque_data_extracted = torque_data[t0:t3]
        t1 = t2 = 0  #for finding t1 and t2
        for idx, elem in enumerate(actual_arm_position_extracted):
            if elem < 0:
                t1 = t0 + idx + 1
                break
        count = 0;
        element = 0
        for idx, elem in enumerate(actual_arm_position_extracted):
            if elem < 0:
                count = count + 1
                element = elem
        t2 = t1 + count - 1
        cycle_t0_t3_points.append(str(t0) + ":" + str(t1) + ":" + str(t2) + ":" + str(t3))
        get_All_t0_t3_points(initCut, initCut + cycleLength + 1, lastRowNum, cycle + 1, cycle_t0_t3_points, cycleLength)
```

Figure 34:All t0 to t3 cycles

Below is the table explaining the different intervals from t0 to t3.

No	Interval	Description
1	t0	Both the arm is at maximum distance with each other.
2	t1	Both the arms are coming closer in contact and cutting the packets.
3	t2	After cutting, both the arm is going away from each other
4	t3	Both the arm is at maximum distance with each other again.

Table 2 : Time Interval Description

2.6.4 Generating signals

After data analysis was done on the collected data it was observed that the cross-seal bands and torque are highly positively correlated with each other. The general idea proposed was to measure the torque generated while packaging to identify the fault. If there is a slight variance in the torque value while cutting the packet, the model can predict the acceptability of the packet.

But the main challenge was the collected torque which was almost accurate all the time which cannot be used to train the neural network because of the inability of the neural network to differentiate between the incorrect and correct torque generated. Hence, some of the torque was needed to modify between the interval t0 to t3 cycle with some fake torque data for the neural network to differentiate.

```
#Random Modification ( code by Pankaj )
def rand_modification_dataset(dataset_arrange_in_samples, cycle_t0_t3_points):
    total_modified_signal = np.array([], dtype=float)
    perturbation_sample = np.array([], dtype=float)
    label_sample = np.array([], dtype=int)
    classes_type = np.array([], dtype=int)

    for index, sample_of_one_cycle in enumerate(dataset_arrange_in_samples):  # get 1 cycle from dataset

            percentage_modified_samples = 0.1  # 10% modification
            #Decide if this cycle is to be modified
            if random.randrange(0,11) / 10 >= 1 - percentage_modified_samples:  # X > (1-0.1), X is a number between 0 and 1

                perturbation =np.random.choice(range(0, 5))  # Get magnitude to modify a sample
                sample_of_one_cycle = np.round(sample_of_one_cycle * (1 + perturbation / 100), decimals=5)
                perturbation_sample = np.append(perturbation_sample,np.full((sample_of_one_cycle.size, 1),perturbation / 100))
                label_sample = np.append(label_sample,np.full((sample_of_one_cycle.size, 1), perturbation))
                classes_type = np.append(classes_type, np.full((sample_of_one_cycle.size, 1),classes_types(perturbation)))
            else:
                perturbation_sample = np.append(perturbation_sample, np.full((sample_of_one_cycle.size, 1), 0))
                label_sample = np.append(label_sample, np.full((sample_of_one_cycle.size, 1), 0))
                classes_type = np.append(classes_type, np.full((sample_of_one_cycle.size, 1), 0))
            total_modified_signal = np.append(total_modified_signal, sample_of_one_cycle)
            neural_network_dataset = pd.DataFrame(
                {'Modified_signal': total_modified_signal, ' Perturbation_sample': perturbation_sample,
                 'Label_sample': label_sample, 'Classes': classes_type})

    return neural_network_dataset
```

Figure 35: Random Modification of Torque

Above is the code which applies 10% modification of the torque between t0 to t3 cycle randomly. The main logic is show below:

	Torque	Modified Torque	Perturbation Sample	Label sample	Classes
Sample 1		1.01	1.01	1	0
	2	2.02	1.01	1	
	3	3.03	1.01	1	
	4	4.04	1.01	1	
	---	---	---	---	
	---	---	---	---	
	---		1.01	---	
	8	8.08	1.01	1	
Sample 2	9	9.18	1.02	2	1
	10	10.2	1.02	2	
	11	11.22	1.02	2	
	---	---	---	---	
	---	---	---	---	
	---	---	---	---	
	14	14.28	1.02	2	
	15	15.53	1.02	2	
Sample 3	16	16.64	1.04	4	2
	17	17.68	1.04	4	
	18	18.72	1.04	4	
	--	---	---	---	
	---	---	---	---	
	---	---	---	---	
	21	21.84	1.04	4	
	22	22.88	1.04	4	

Table 3 :Modified Torque and Class Definitions

Calculations:

Label sample= random (0,5)
Perturbation sample= 1+random (0,5)/100
Modified Torque = Torque * Perturbation sample

Cases:

Table 4 :Cases Calculation's

Case	Percentage	Perturbation sample	Classes	Packaging
1	0->1.5%	0 <=Perturbation <= 1	0	Valid
2	1.5 - 3.5%	2 <=Perturbation <= 3	1	Improvable/Acceptable
3	>3.5%	Perturbation >= 4	2	Invalid/Rejected

```python
#class generation ( code by Pankaj )
def classes_types(perturbation_value):
    if 0 <= perturbation_value <= 1:
        return 0;
    if 2 <= perturbation_value <= 3:
        return 1;
    if perturbation_value >= 4:
        return 2;
```

Figure 36: Python Class Definition

The above tables show the perturbation and cases calculations. Based on these calculations a graph was generated to show the original and modified torque.

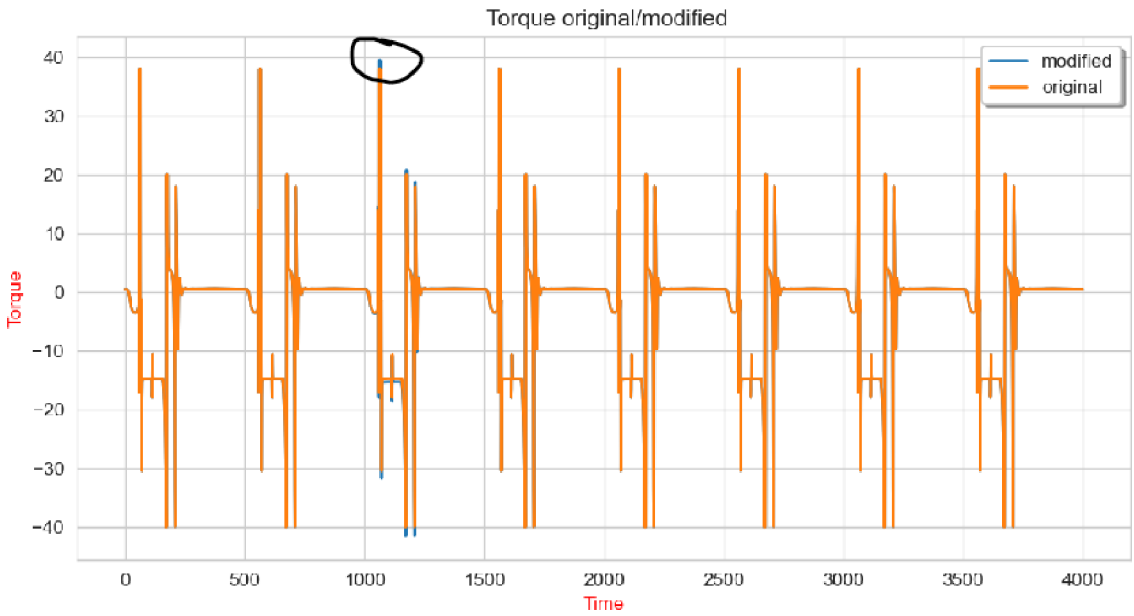

Figure 37:Modified vs Original Torque

The black section highlighted here shows the modified torque of a cycle. The modification applied here is completely random 10% of the total torque. These data will be used to train the neural network.

2.6.5 Suggesting a model

After creating the dataset, the next idea was to create a basic CNN univariate time series multi class classification architecture [32].

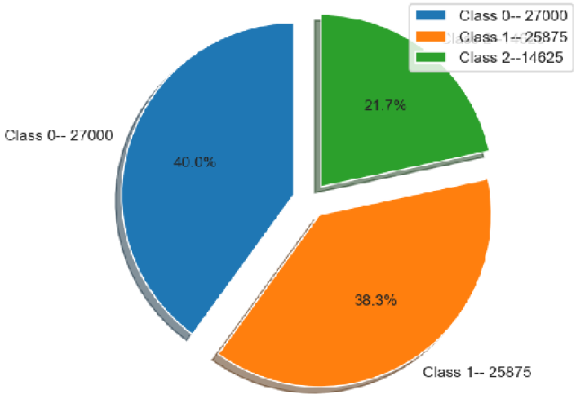

Figure 38: Dataset Classification

A total of 66.67% was used for training the neural network and 33.33% for testing.
As seen from above, dataset is divided into different class percentage. This percentage was proposed based on trying different combinations.

Modified_signal	Perturbation_sample	Label_sample	Classes
0.52807	0.04	4	2
0.52934	0.04	4	2
0.52807	0.04	4	2
0.52301	0.04	4	2
0.51541	0.04	4	2
0.50274	0.04	4	2
0.48501	0.04	4	2
0.46222	0.04	4	2
0.43056	0.04	4	2
0.38877	0.04	4	2
0.33052	0.04	4	2
0.25454	0.04	4	2
0.1545	0.04	4	2
0.02913	0.04	4	2
-0.12537	0.04	4	2
-0.30646	0.04	4	2
-0.51541	0.04	4	2
-0.74968	0.04	4	2
-1.00422	0.04	4	2
-1.27016	0.04	4	2
-1.53736	0.04	4	2
-1.79316	0.04	4	2
-2.0325	0.04	4	2
-2.25032	0.04	4	2
-2.4466	0.04	4	2
-2.62263	0.04	4	2
-2.77839	0.04	4	2

Figure 39: Real Dataset Overview

Real dataset generated is shown above. Off course, It not the full dataset but an overview. Modified _signal is the modified torque highlighted here. Modified torque will only be used to train the neural network.

Both CNN and LSTM are popular neural networks architecture for time series data. However, CNN came out to be better in many cases.

The first task was to build the basic CNN Model for univariate time series.

```
def make_model(input_shape):
    input_layer = keras.layers.Input(input_shape)

    conv1 = keras.layers.Conv1D(filters=64, kernel_size=3, padding="same")(input_layer)
    conv1 = keras.layers.BatchNormalization()(conv1)
    conv1 = keras.layers.ReLU()(conv1)

    conv2 = keras.layers.Conv1D(filters=64, kernel_size=3, padding="same")(conv1)
    conv2 = keras.layers.BatchNormalization()(conv2)
    conv2 = keras.layers.ReLU()(conv2)

    conv3 = keras.layers.Conv1D(filters=64, kernel_size=3, padding="same")(conv2)
    conv3 = keras.layers.BatchNormalization()(conv3)
    conv3 = keras.layers.ReLU()(conv3)

    gap = keras.layers.GlobalAveragePooling1D()(conv3)

    output_layer = keras.layers.Dense(num_classes, activation="softmax")(gap)

    return keras.models.Model(inputs=input_layer, outputs=output_layer)


model = make_model(input_shape=x_train.shape[1:])
keras.utils.plot_model(model, show_shapes=True)
```

Figure 40: Build the Model

```
epochs = 500
batch_size = 32

callbacks = [
    keras.callbacks.ModelCheckpoint(
        "best_model.h5", save_best_only=True, monitor="val_loss"
    ),
    keras.callbacks.ReduceLROnPlateau(
        monitor="val_loss", factor=0.5, patience=20, min_lr=0.0001
    ),
    keras.callbacks.EarlyStopping(monitor="val_loss", patience=50, verbose=1),
]
model.compile(
    optimizer="adam",
    loss="sparse_categorical_crossentropy",
    metrics=["sparse_categorical_accuracy"],
)
history = model.fit(
    x_train,
    y_train,
    batch_size=batch_size,
    epochs=epochs,
    callbacks=callbacks,
    validation_split=0.2,
    verbose=1,
)
```

Figure 41:Train the Model

This were the default Hyperparameters to tune a 1D-CNN network. Later the model was tuned using Hyperparameter Tuning with the Keras Tuner and TensorFlow.

2.6.6 Optimizing the model

Optimizing the model means finding the best combination of Hyperparameters to train the neural network to get maximum accuracy. Hyperparameters Tuning can be challenging because the best combination of Hyperparameters may differ on every tuning. Hyperparameters Tuning was done using TensorFlow and Keras Tuner [33].

First approach was to tune it using TensorFlow as shown.

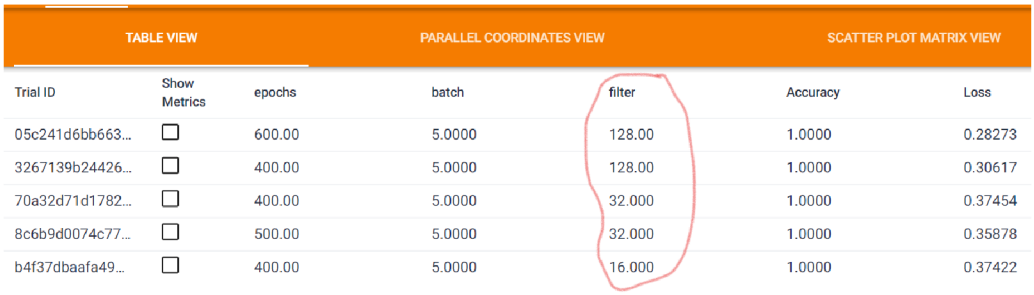

TABLE VIEW | PARALLEL COORDINATES VIEW | SCATTER PLOT MATRIX VIEW

Trial ID	Show Metrics	epochs	batch	filter	Accuracy	Loss
05c241d6bb663...	☐	600.00	5.0000	128.00	1.0000	0.28273
3267139b24426...	☐	400.00	5.0000	128.00	1.0000	0.30617
70a32d71d1782...	☐	400.00	5.0000	32.000	1.0000	0.37454
8c6b9d0074c77...	☐	500.00	5.0000	32.000	1.0000	0.35878
b4f37dbaafa49...	☐	400.00	5.0000	16.000	1.0000	0.37422

Figure 42: Hyperparameters Tuning Tensor Board

The Tensor Board gave a general idea about epochs, batch, filters etc. Based on this information Keras Tuner was configured to find the best fit.

```
Trial 725 Complete [00h 02m 49s]
val_accuracy: 0.8999999761581421

Best val_accuracy So Far: 1.0
Total elapsed time: 06h 25m 46s
INFO:tensorflow:Oracle triggered exit
Number of conv blocks: 3
filters_0: 96
filters_1: 176
filters_2: 256
learning_rate: 0.00013156408327740765
```

Figure 44: Keras Tuner Hyperparameters 1

```
Trial 725 Complete [00h 07m 44s]
val_accuracy: 1.0

Best val_accuracy So Far: 1.0
Total elapsed time: 05h 25m 31s
INFO:tensorflow:Oracle triggered exit
Number of conv blocks: 3
filters_0: 128
filters_1: 112
filters_2: 80
learning_rate: 0.00012180019936545399
```

Figure 43: Keras Tuner Hyperparameters 2

As seen from above, on running Keras Tuner multiple time, it gave different Hyperparameters. Keras Tuner Hyperparameters 2 got an accuracy of 100% with other parameters.
These parameters were used to construct the neural network as shown below.

```python
#1D-CNN-    //coded by Pankaj
def make_model(input_shape):
    input_layer = keras.layers.Input(input_shape)

    conv1 = keras.layers.Conv1D(filters=128, kernel_size=3, padding="same")(input_layer)
    conv1 = keras.layers.BatchNormalization()(conv1)
    conv1 = keras.layers.ReLU()(conv1)

    conv2 = keras.layers.Conv1D(filters=112, kernel_size=3, padding="same")(conv1)
    conv2 = keras.layers.BatchNormalization()(conv2)
    conv2 = keras.layers.ReLU()(conv2)

    conv3 = keras.layers.Conv1D(filters=80, kernel_size=3, padding="same")(conv2)
    conv3 = keras.layers.BatchNormalization()(conv3)
    conv3 = keras.layers.ReLU()(conv3)

    gap = keras.layers.GlobalAveragePooling1D()(conv3)
    output_layer = keras.layers.Dense(num_classes, activation="softmax")(gap)

    return keras.models.Model(inputs=input_layer, outputs=output_layer)
```

Figure 45: 1D- CNN Model

```python
#compiling the model
model = make_model(input_shape=(None,1))
model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.00012180019936545399),
    loss="sparse_categorical_crossentropy",
    metrics=["accuracy"],
)
```

Figure 46: Compiling 1D- CNN Model

```python
#training the model
epochs=[550]
batch_size =[5]
for batch in batch_size:
    for epoch in epochs:
        history = model.fit(
            x_train,
            y_train,
            batch_size=batch,
            epochs=epoch,
            callbacks=callbacks_list,
            validation_split=0.5,
            verbose=1,
        )
```

Figure 47: Training 1D- CNN Model

As seen, recommended parameters were used and the model was allowed to train on training data. It took some time to get trained and later testing was performed.

```
#Test the model on 33.33% Data
model = keras.models.load_model("model.h5")
test_loss, test_acc = model.evaluate(x_test, y_test)
print("Test accuracy", test_acc)
print("Test loss", test_loss)
```

Figure 48: Testing 1D-CNN

The model was saved with a name 'model.h5' and it was tested on 33.33% of testing data. The result of this operation is shown below.

```
2/2 [==============================] - 0s 21ms/step - loss: 0.4935 - accuracy: 1.0000
Test accuracy 1.0
Test loss 0.49349263310432434
END
```

Figure 49: Testing Result

As seen, the accuracy of 100% was achieved with minimum loss as possible. Off course, this is not the limit, a better set of Hyperparameters can be defined which will have less computational architecture. There is always a room for improvement.

2.6.7 Deployment

After testing was done, the model was deployed on Edge. To deploy anything on edge, an architecture must be followed.

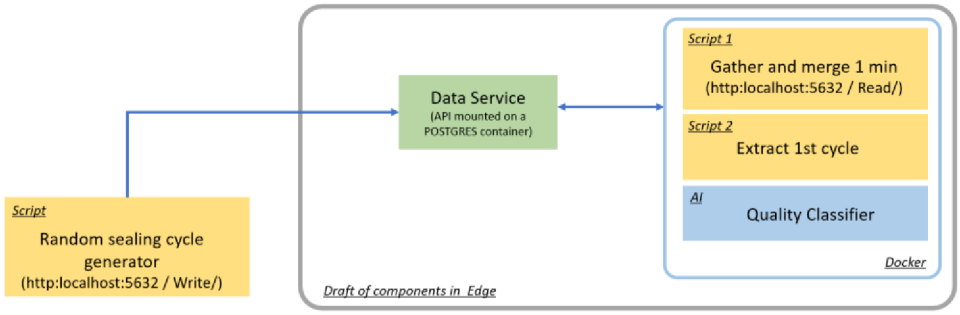

Figure 50: Deployment Process

Deployment Process includes three sections.

1. Random Sealing cycle: This part includes generating data with modification as discussed, every 200ms from the machine.

2. Data Service: This service is the part of Edge environment. It takes data from the machine every 200ms from the machine and saves it in Database. It provides Rest Api's to consume the services [34].

```
[
  {
    "variableId": "4d97c334269e4e51a2fa53ed8d4eb376",
    "values": [
      {
        "timestamp": "2022-11-25T15:31:30.686800Z",
        "value": -0.999
      },
      {
        "timestamp": "2022-11-25T15:31:30.538000Z",
        "value": -0.999
      }
    ]
  },
  {
    "variableId": "88dac46ea6a34a5d89a33d426af572d5",
    "values": [
      {
        "timestamp": "2022-11-25T15:31:30.686800Z",
        "value": -14.7080100205494
      },
      {
        "timestamp": "2022-11-25T15:31:30.538000Z",
        "value": -14.7080100205494
      }
    ]
  }
]
```


Figure 51: JSON Structure

Data service accepts data from the machine in JSON format shown above. The first variableId represents the cross Seal Actual Position and the second represents modified Torque.

3. Docker: Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly [35]. This service includes three sub services.

    a. Gather Data: It includes reading from Data service as soon as data is available to data service. A python code was written to gather data.

    b. Extract Cycle: This part includes constructing the cycle (t0-t3) of modified torque coming from Data Service. Json received shown above from Data Service is used to extract and create cycle which is to be fed to neural network to predict classification.

    c. Quality Classifier: Next, the cycle is passed to the neural network to predict the class it belongs or if the packaging is of set standards.

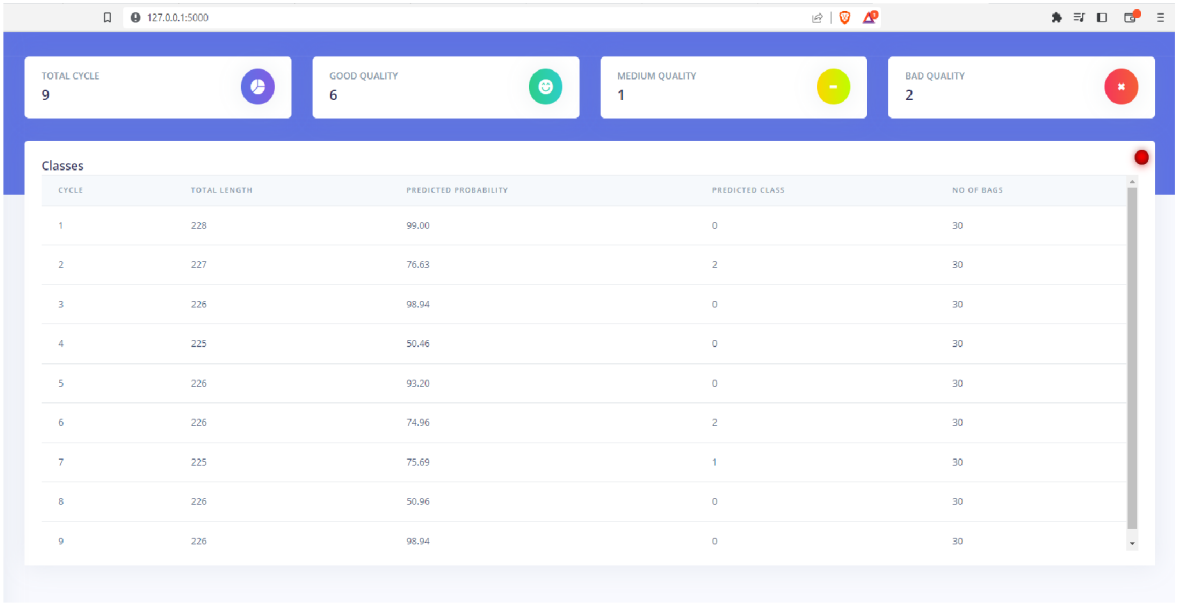

Figure 52: Final Output on Edge

The final output is shown above. The output shows no of good (Class 0), medium (Class 1), and bad (Class 2) quality packaging in total cycle. The table shows the predicted classes with predicted probability by the trained neural network. The total cycle represents no of bags per min.

3 Results and Discussion

3.1 Result

The result can be better Interpreted using Confusion matrix. The confusion Matrix gives a comparison between actual and predicted values. It is used for the optimization of machine learning models. The confusion matrix is a N x N matrix, where N is the number of classes or outputs [36]

```
from sklearn.metrics import confusion_matrix
import seaborn as sb
import matplotlib.pyplot as plt
#Testing the model using X_test and storing the output in y_pred
y_pred = model.predict(x_test)
y_pred=np.argmax(y_pred, axis=1)

# Creating  a confusion matrix,which compares the y_test and y_pred
plt.figure(figsize=(10,6))
fx=sb.heatmap(confusion_matrix(y_test,y_pred), annot=True,cmap="GnBu")
fx.set_title('Confusion Matrix \n');
fx.set_xlabel('\n Predicted Values\n')
fx.set_ylabel('Actual Values\n');
fx.xaxis.set_ticklabels(["valid","Improvable","Rejected"])
fx.yaxis.set_ticklabels(["valid","Improvable","Rejected"])
plt.show()
```

Figure 53: Confusion Matrix Code

Above is the code to generate confusion matrix using python in build matplotlib and seaborn library.

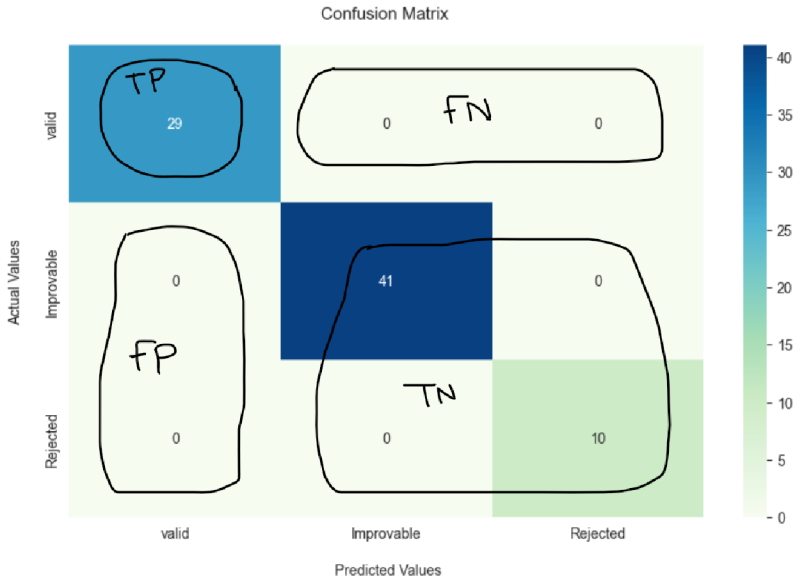

Figure 54: Confusion Matrix

Above figure is representation based on class ‘valid’. Let’s calculate TP, FN, FP, TN values for class ‘valid’.

TP: True Positive means both the actual and Predicted value are Positive. So, the TP value is 29 (cell 0).

FN: False Negative means predicted value is negative, but actual value is negative. FN will be (0+0) =0 (cell 2+cell 3).

FP: False Positive means predicted value is positive, but the actual value is negative. FP will be (0+0) =0 (cell 4+cell 7).

TN: True Negative means both the actual and predicted values are negative. TN will be (41+0+0+10) =51 (cell 5+cell 6 + cell 8+cell 9).

Similarly, TP, FN, FP, TN values for class 'Improvable' will be:

TP: 41 (cell 0)
FN: 0+0=0 (cell 4+cell 6)
FP: 0+0=0 (cell 2+cell 8)
TN: 29+0+0+10=39 (cell 1+cell 3+ cell 7+cell 9)

Below is the classification Report generated using sklearn metrics.

Classification Report

	precision	recall	f1-score	support
valid	1.00	1.00	1.00	29
Improvable	1.00	1.00	1.00	41
Rejected	1.00	1.00	1.00	10
accuracy			1.00	80
macro avg	1.00	1.00	1.00	80
weighted avg	1.00	1.00	1.00	80

Figure 55: Classification Report

Precision: It measures out of all predicted positives how many are positive.
Recall: It measures how many positive records are predicted correctly.
F1-score: It is mean of precision and recall.
Accuracy: It measures how accurate is the model prediction.

Hence, Confusion metrics and Classification report gives a valuable information about the model accuracy and other valuable info which can be used to improve the model.

3.2 Discussion

The outcome of this assignment has provided a neural network model and tested viability of siemens industrial edge. Neural network was created by gathering the real time data from the machine. The data was analysed using various techniques to extract the important and dependent features. The dataset was prepared, and CNN Model was chosen to perform multiclass classification. The model was trained and tested to classify the validity of packaging. In the end, a suitable deployment architecture was created to deploy, and the trained model was then deployed and tested on siemens Industrial edge.

Nearly 33.33% equivalent to nearly 150-180 cycles was tested with an accuracy of 100%. Of course, more testing of different or more cycles in future will decrease the accuracy with unexpected new scenarios. Machine learning played a big part in the outcome because initially, the operator had no way to detect the quality of the packaging because of the lack of visual configuration on HMI.

A better strategy played a very vital role in achieving the goals which was set. Off course, there are another various element which can be improved to achieve results in least possible ways in future.

- Alternate and a better neural network selection can be helpful to train the model in less time. Neural networks like Long Short-Term Memory (LSTM) which is the popular variant of RNN shown in Figure 7 and General Adversarial Networks (GAN) can be tried.
- Better selection of dataset can play an important role. Other positive correlation like LinearAxis. ActualPosition can also be analysed further to check its dependencies on CrossSeal.Status. TorqueDataActualTorque (NM) as shown in Figure 28.
- A running and working simulation can be set up to collect data instead from the real time data as it will save time.
- Deployment on siemens industrial edge can be streamlined in future making it much easier to deploy AI.

These measurements can be taken to improve the process which can lead to design a neural networks and deployments in more convenient ways in future. The world is moving to AI and implementing AI on machines can bring various benefits such as increased efficiency, improved safety, and enhanced productivity. Implementing AI on machine such as Packaging machine could reduce a lot of manual work which results in less labour and cost reduction.

Predicting the next values based on historical data to prevent a defected product can be next big step which can be thought about a will be quite interesting to accomplish because future of AI in such machines are immense because it can bring various benefits such as increased efficiency, improved safety, and enhanced productivity. It can solve complex problems and generate insights that were previously impossible with traditional computing methods. However, there are also potential risk and challenges associated with AI in machines which includes job displacement and possibilities of unintended consequences. Therefore, it is important to address these issues while implementing AI in machines.

4 Conclusion

We analysed and discussed the problem and proposed a visualization for the operator to check the quality of packaging and test its viability on industrial edge. We started with collecting the data from the demo machine using python OPC UA library and analysed the

data by creating correlation matrix and graphs using python Matplotlib with Jupiter notebook.

After analysing the data, we identified the dependent signals and applied 10% modification in the signal. We also categorized the signals into its respective packaging classes of valid, improvable, and rejected quality. A total of 66.67% was used for training the neural network and 33.33% for testing.

We opted for one dimensional Convolutional Neural Network (CNN) for univariate time series multi-class classification and used various hyper parameter tuning techniques using TensorFlow and Keras Tuner to optimize the neural network for better performance. After repeating the process several times, we found the best fit hyper tuning parameters for the 1D-CNN with an accuracy of 100%.

Finally, we followed the deployment architecture shown in Figure 50, which included creating a docker image and successfully deployed our trained neural network on industrial edge and interpreted the result. We were able to test our neural network on the industrial edge. The deployed model on industrial edge was able to predict the signals belonging to different classes of good, medium, and bad quality packaging as shown in Figure 52.

We should end this article with a note of advice, though. However, compelling it may sound, there is usually no-one-size-fits all solution to select and train a neural network. It may vary. So, we must use intelligence when choosing the simplest answers for the problems we aim to tackle.

5 References

[1] Burggräf, P. *et al.* (2017) "Cost-benefit analysis for Disruption Prevention in low-volume assembly," *Production Engineering*, 11(3), pp. 331–342. Available at: https://doi.org/10.1007/s11740-017-0735-6.

[2] Miller, S. (2022) *What is vertical form filling machines? Viking Masek*. Available at: https://vikingmasek.com/packaging-machine-resources/packaging-machine-blog/a-guide-to-vertical-form-fill-seal-machines (Accessed: March 27, 2023).

[3] *Packaging machine manufacturer* (2023) *Viking Masek*. Available at: https://vikingmasek.com/ (Accessed: March 27, 2023).

[4] Rutenbar, M. W., Chandrasekaran, S., & Jiang, W. (2001). Machine health assessment using support vector machines. IEEE Transactions on Semiconductor Manufacturing, 14, 175-185.

[5] Li, J., Ding, F., Li, Z., Li, J., & Liu, Z. (2012). The application of principal component analysis and decision tree analysis in the remaining useful life prediction of a packaging machine. Expert Systems with Applications, 39, 12687-12694.

[6] Kuo, R. J., Hsieh, Y. C., & Hsieh, C. Y. (2018). Using a convolutional neural network for machine health monitoring and fault diagnosis. Journal of Manufacturing Systems, 49, 126-139.

[7] Jia, L., Dong, F., Zhang, L., Wei, X., & Zhao, J. (2020). A deep learning model based on CNN-LSTM for predicting the remaining useful life of packaging machines. Journal of Intelligent Manufacturing, 31, 1337-1352.

[8] Chen, C., Tang, Y., Lu, H., & Xiong, H. (2021). Edge-based predictive maintenance for VFFS machines using a deep convolutional neural network. IEEE Access, 9, 34695-34706.

[9] Zhang, H., Wang, L., and Zhu, X. (2021). Deep learning for VFFS packaging quality inspection: a comprehensive review. Journal of Intelligent Manufacturing, 32, 1353-1371.

[10] Jin, H., Li, J., Li, X., Li, H., & Li, J. (2021). Detection of Defects in VFFS Packages Using Convolutional Neural Networks. IEEE Access, 9, 45180-45188.

[11] Xie, C., Guo, X., Wang, Y., Huang, X., & Zhang, Q. (2021). Application of convolutional neural network and industrial edge computing in VFFS packaging process for predictive maintenance. Journal of Manufacturing Systems, 61, 72-82.

[12] CARTER, Roger. Data Processing. *Information Technology*. 1991. P. 192–217. DOI 10.1016/b978-0-7506-0141-2.50014-x.

[13] SYDENHAM, P. H., THORN, Richard and ABRAHAM, Ajith. 129: Artificial Neural Networks. In: *Handbook of Measuring System Design* [online]. Chichester, England: Wiley, 2005. Available from: http://wsc10.softcomputing.net/ann_chapter.pdf.

[14] *Activation functions in neural networks [12 types & use cases]* (no date) *V7*. Available at: https://www.v7labs.com/blog/neural-networks-activation-functions (Accessed: March 29, 2023).

[15] WRITER, Anukrati MehtaA creative. A comprehensive guide to types of Neural Networks. *Digital Vidya* [online]. 28 April 2022. [Accessed 29 March 2023]. Available from: https://www.digitalvidya.com/blog/types-of-neural-networks/

[16] Why is it called "logistic regression" and not "logistic ... - turbofuture. [online]. [Accessed 29 March 2023]. Available from: https://turbofuture.com/industrial/Why-Logistic-Regression-Why-not-Logistic-Classification

[17] SAHA, Sumit. A comprehensive guide to Convolutional Neural Networks - the eli5 way. *Medium* [online]. 16 November 2022. [Accessed 29 March 2023]. Available from: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

[18] GU, Jiuxiang, WANG, Zhenhua, KUEN, Jason, MA, Lianyang, SHAHROUDY, Amir, SHUAI, Bing, LIU, Ting, WANG, Xingxing, WANG, Li, WANG, Gang, CAI, Jianfei and CHEN, Tsuhan. Recent advances in Convolutional Neural Networks. *arXiv.org* [online]. 19 October 2017. [Accessed 12 March 2023]. Available from: https://arxiv.org/abs/1512.07108.

[19] CRESWELL, Antonia, WHITE, Tom, DUMOULIN, Vincent, ARULKUMARAN, Kai, SENGUPTA, Biswa and BHARATH, Anil A. Generative Adversarial Networks: An overview. *IEEE Signal Processing Magazine*. 2018. Vol. 35, no. 1p. 53–65. DOI 10.1109/msp.2017.2765202.

[20] Oppenheim, A. V., & Schafer, R. W. (2010). Discrete-time signal processing. Pearson Education India.

[21] Gray, P. R., Hurst, P. J., Lewis, S. H., & Meyer, R. G. (2001). Analysis and design of analog integrated circuits. John Wiley & Sons.

[22] Proakis, J. G., & Manolakis, D. G. (2006). Digital signal processing: principles, algorithms, and applications. Pearson Education India.

[23] Kamel, K. and Kamel, E. (2014) Programmable logic controllers: Industrial control. New York: McGraw-Hill Education.

[24] Xu, H., Wang, H., & Zhang, Y. (2019). Signal processing for VFFS machines using PLC. In 2019 3rd International Conference on Intelligent Transportation Engineering (ICITE) (pp. 176-179). IEEE.

[25] Tandon, A., & Goyal, R. K. (2018). Signal processing techniques for VFFS machines: A review. Journal of Food Engineering, 219, 32-41.

[26] Industrial Automation portfolio (no date) siemens.com Global Website. Available at: https://new.siemens.com/global/en/products/automation.html (Accessed: March 27, 2023).

[27] Miller, S. (2022) What is vertical form filling machines? Viking Masek. Available at: https://vikingmasek.com/packaging-machine-resources/packaging-machine-blog/a-guide-to-vertical-form-fill-seal-machines (Accessed: March 27, 2023).

[28] Industrial edge – edge computing for industry 4.0 (no date) siemens.com Global Website. Available at: https://www.siemens.com/global/en/products/automation/topic-areas/industrial-edge.html (Accessed: March 27, 2023).

[29] Girdhar, P. and Scheffer, C. (2004) "Predictive maintenance techniques," Practical Machinery Vibration Analysis and Predictive Maintenance, pp. 11–28. Available at: https://doi.org/10.1016/b978-075066275-8/50002-3.

[30] MAHNKE, Wolfgang, LEITNER, Stefan-Helmut and DAMM, Matthias. OPC Unified Architecture. Scholars Portal, 2009.

[31] Understanding correlations and correlation matrix. Muthukrishnan [online]. 7 May 2021. [Accessed 29 March 2023]. Available from: https://muthu.co/understanding-correlations-and-correlation-matrix/

[32] Song, K., Wang, N. and Wang, H. (2020) "A metric learning-based univariate time series classification method," Information, 11(6), p. 288. Available at: https://doi.org/10.3390/info11060288.

[33] NEWMAN, Luke. Hyperparameter tuning with Kera tuner and TensorFlow. Medium [online]. 27 August 2021. [Accessed 12 March 2023]. Available from: https://towardsdatascience.com/hyperparameter-tuning-with-kerastuner-and-tensorflow-c4a4d690b31a#:~:text=The%20process%20of%20searching%20for,units%20in%20a%20dense%20layer

[34] Data Service for Industrial Edge V1. [online]. [Accessed 12 March 2023]. Available from: https://support.industry.siemens.com/cs/attachments/109781417/EdgeApp_DataServicedeDE_de-DE.pdf

[35] Docker Overview. Docker Documentation [online]. 28 March 2023. [Accessed 29 March 2023]. Available from: https://docs.docker.com/get-started/overview/

[36] BHARATHI. Confusion matrix for multi-class classification: Latest Guide 2023. Analytics Vidhya [online]. 10 March 2023. [Accessed 12 March 2023]. Available from: https://www.analyticsvidhya.com/blog/2021/06/confusion-matrix-for-multi-class-classification/

6 List of pictures and tables

6.1 List of pictures

6.2 List of tables

Appendix

An appendix exists for this thesis; however, it contains confidential information that cannot be disclosed. As a result, it has been omitted from this version of the thesis.