Fakultät IV

Naturewissenschaftlich-Technische Fakultät

Department Maschinenbau

Prof. Dr.- Ing. Peter Kraemer

Studienarbeit Report

Name: Guru Prasaath Pushparaj Matr. -Nr:1702033

Title:

Synthetic Data Generation for the extension of degradation index from run-to-failure tests to compensate for missing data

Submission date: 18.10.2024

Problem Statement:

One of the primary challenges encountered with run-to-failure measurements for any components is the lack of data about the failure event. This issue arises from the industry practice of replacing critical components before they fail, a preventive measure to control material costs and avoid unplanned downtime. As a result, only incomplete time series data for a component's entire lifetime is available, which can hinder the development of accurate remaining useful life (RUL) prediction models.

Proposed solution:

To implement Generative Adversarial Networks (GANs) / Recurrent Neural Networks (RNNs) or other time series data generation models to generate realistic degradation indices for the given component from the point of their replacement to the hypothetical failure point. These models should be able to learn the underlying patterns in the available patterns in the available degradation data and generate plausible future sequences, effectively filling the gaps and providing complete run-to-failure data. This approach can improve the performance of RUL prediction models by providing them with more comprehensive training data.

Abstract:

Predicting a components remaining useful life (RUL) is very crucial in Industries, where preventive maintenance to control material costs and lowering the unplanned downtime are important. The acquisition of data from a damaged or low-quality components are often a potential challenge and can lead to unbalanced time series datasets that do not capture entire degradation process. This particularly affect the data analysis via various machine learning (ML) algorithms, for which balanced data sets are the main prerequisites. There are several methods to solve this issue. The generative AI methods like Generative Adversarial Networks (GAN) and Variational Autoencoders (VAE) are proven to be an efficient method to replicate and produce realistic degradation trajectories. In this project a combination of VAE and GAN is used to produce realistic futuristic time series. The VAE learns the underlying patterns in the available degradation data and the GAN model generate the plausible future sequence providing the effective run to failure data. Here the Gated Rectified Unit (GRU) is used in the generator and the Convolution Neural Network (CNN) is used in the critic / discriminator. This method aims to improve the accuracy and reliability of RUL predictions, thereby contributing better to maintenance planning, etc.

Table of Contents

Page no

1. Introduction:

To cut many complex and intricate shapes from various materials like foam, rubber, plastics, etc an advanced Industrial tool called Contour cutting Machine is designed. They use technology like Computer Numerical Control (CNC) to follow the detailed pattern and contours, thus creating the custom parts with very high accuracy and minimum material waste. These machines are widely used in automotive, aerospace and packaging industries. Also, these machines enhance the production efficiency and product quality with exceptional precision. The common issue that has been faced by these machines are blade failure despite having some high advanced capabilities. This issue has a great impact on production and quality. The factors which can cause blade failure includes material hardness, improper blade selection, inadequate maintenance. Uneven abrasions, Excessive noise, Vibration and wear are the signs of blade failure.

To minimize the risk of failure of blades regular maintenance and proper blade selection has to be followed. By properly following the above precautions and methods, the manufacturers can minimize the downtime, increase the lifespan of the equipment and results in high quality output. Finding out the Remaining Useful Life (RUL) of the blade is one of the significant works to be done in machine tool prognostics. The failure data is collected from the machine using various sensors under different operating conditions. The features extracted from the sensors are processed in time as well as frequency domain. But there are several issues in this. One of the key issues is the lack of the data of the blade from the sensors due to the replacement or from a damaged sensor. This Studienarbeit deals to address the missing data issue along with providing one of the realistic solutions for the problem using one of the most prominent Deep learning (DL) techniques namely Generative Adversarial Network (GAN) and Variational Auto Encoders (VAE).

2. Experimental Setup:

For this studienarbeit study, Albrecht Bäumer's contour cutting machine was used with some modifications to facilitate failure experiments for different operating conditions. The machine is designed for efficient foam cutting process and precise processing. It operates with a band blade that is tensioned and driven by four pulleys. Each pulley is placed in each corner of the machine as shown in the figure. The pulleys which are crucial for maintaining the cutting accuracy and consistency of the foam cutting. The blade moves along the Y-Z axis as shown in the figure and the foam bed which moves in the X direction is given as fed for the machine. The twisting of the blade is given about Z axis. The material feed direction, shown by the arrow at the bottom centre, guides the material into the machine for cutting.

There are three groups of sensors which are positioned in different positions in the machine for their effective functioning. The sensor group 1 which is placed near the tension pulley i.e., the top left corner is used to detect any perturbations between the band blade and the tensioning pulley. The sensor group 1 contains the following sensors.

- Strain gauge – x 1
- Tri- axis acceleration sensor – x 1
- Laser distance sensor – x 1

- Limit switch – x 1

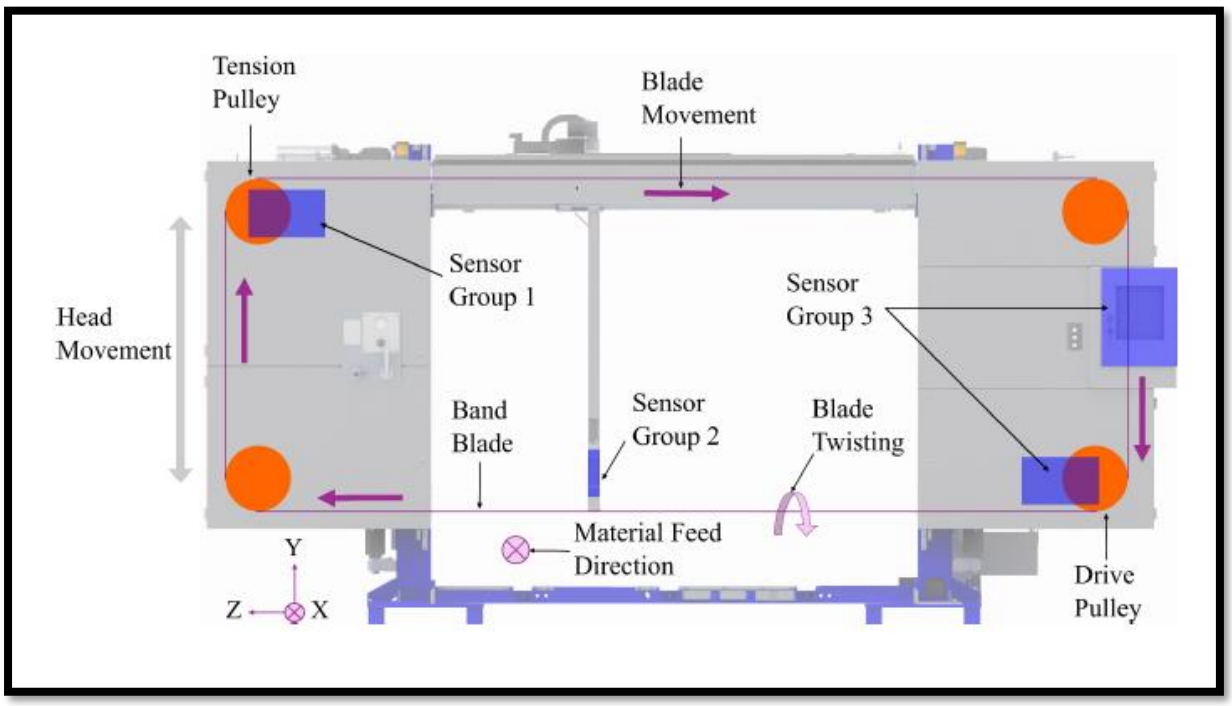

***Figure 1**: Description of Bäumer's Contour cutting machine*

The sensor group 2 which is placed near the feed bed, is used to have a track on the wear of the band blade throughout the testing process. This group has the minimum distance to the blade and the contact parts. So, this part of the sensor group is very crucial for data collection. The sensor group 2 has the following sensors.

- Acoustic emission sensor – x 1
- Tri- axis acceleration sensor – x 1

The sensor group 3, positioned between the pulleys in the right side of the machine. This group contains 2 sensors.

- Eddy current sensor – x 1
- Temperature sensor – x 1

The eddy current sensor detects any cracks present in the band blade and the other sensor notes the temperature fluctuations around the blade due to the rolling of the blade around the pulley at very high speed.

Some simplifications are made to the contour cutting machines due to the space constraint of the laboratory environment. As already said the foam bed is given as feed to the machine. Although it is a foam, it exerts a load on the band blade which tends to deflect the blade. Due to the constraint, the bed is compensated by the integration of a loading mechanism that contains adjustable screws and a plate placed on the support. This mechanism causes a simulation of the blade deflection that will be created by the real environment from a limit ranging from 0 mm to 15 mm. Because of the use of this, the feed movement of the assembly is omitted as they have no effect on the test. The tension pulley moves out completely when

there is a blade failure. There were multiple runs to failure tests conducted with different operating conditions. Some of them are varying load intensities, two different blade types, two different blade speeds and two blade twisting programs.

3. Data Analysis:

The tests were conducted till the blade fail and the sensors data were stored for further processing. One of the important steps in data analysis is feature extraction from the data collected from the sensors. The data collected are extracted into time domain and frequency domain features. The data from all the sensors are extracted into various time domain features namely Mean, maximum value, minimum value, Skewness, root mean square (RMS), kurtosis and crest factor. These time domain statistical measures help in understanding the central tendency, variability and the type of signal distribution over the period of time.

Apart from the time domain features, several frequency domain features are extracted from the acoustic emission sensor and two other acceleration sensors. The features that are extracted from these sensors are spectral mean, spectral root mean square, spectral variance, spectral standard deviation, spectral skewness, spectral kurtosis, spectral mean frequency and spectral median frequency. There are so many features that are extracted from the sensor data. Frequency domain features provide more insights into signal's frequency count which are essential for identifying the patterns and anomalies that can't be found from time domain features. Combining both domain features, it enables more accurate diagnostics and predictive maintenance in the industrial application.

Sensors	Time domain Features
• Eddy Current sensor x 2 • Laser distance sensor • Pulse Generator • Temperature sensor • Strain gauge • Accelerometer x2 • Acoustic emission sensor	Mean
	Maximum
	Minimum
	Skewness
	RMS
	Kurtosis
	Crest factor

***Table 1**: Time domain features*

So, there are 6 data from 2 accelerometers (x, y and z axis) and 7 other data from other sensors. So, there are totally **91** (13 *7) different time domain features extracted at the end.

Sensors	Frequency domain Features
• Accelerometer x2 • Acoustic emission sensor	Spectral Mean frequency
	Spectral variance
	Spectral Standard deviation
	Spectral Skewness
	Spectral RMS
	Spectral Kurtosis
	Spectral Median frequency
	Spectral Mean

***Table 2**: Frequency domain features*

There are 56 frequency domain features extracted from the three sensors data. In total there are **147** features.

3.1. Dataset Overview:

Totally there are 54 accelerated runs to failure tests were conducted with various different operating conditions which resulted in varied failure timings. Out of 54, we received 5 different dataset which has 3 different operating conditions and different blade type. The dataset with specific operating conditions and blade type is mentioned in the tabular below. The dataset received is not normalised.

Test Number	Blade speed (m/s)	Loading (mm)	Blade type	Twisting type	Failure timings
15	14	9.5	Toothless	360	2h 6m 25s
19	14	8.2	Toothless	360	4h 9m 25s
28	14	4.5	Toothed	360	7h 17m 46s
30	10	4.5	Toothed	720	4h 51m 25s
46	14	9.5	Toothed	720	2h 13m 45s

***Table 3**: Dataset description*

4. Degradation Index calculation:

A system or a machinery's health and performance can be assessed using a degradation index which can be calculated from the extracted features. A linear degradation model can be to all the 147 features derived from the time and frequency domain. But including all those features will make the model more complicated and difficult to assess. The features have to be analysed over time to know its trend and correlation with the system's health. The features which show a consistent trend either it can be of increasing or decreasing type are to be chosen for calculating the degradation index.

Some of the metrics that are used in prognostics and health management applications to compare the feature's trend are

- Monotonicity
- Trendability
- Prognosability

For our study the metric **Trendability** is used to select features that can be used to calculate the degradation index. Trendability measures the extent to which the features display the same shape across the group of units. It is the similarity between all the damage trajectories of the population of units. This metric is devised for run to failure data. It shows how well a feature's trend correlate with time. Features with high trendability value shows a clear, continuous trend over time, making it an important indicator of the system's health.

The formally, the trendability of a feature is defined as

$$Trendability = \ min_{j,k}\left|corr(x_j, x_k)\right|, \qquad j,k = 1,2, \ldots . M$$

Where:

- M = number of units
- x_j = vector of measurements of a feature on unit j
- x_k = vector of measurements of a feature on unit k here its time
- $Corr$ = Pearson correlation coefficient

The metric ranges between the range [0,1] which is same as the other metrics. By selecting features with the high trendability value becomes more sensitive to actual changes to the condition of the system and are more robust. This makes sure that only the most relevant and predictive features are selected, thereby reducing the noise and increasing the accuracy of the degradation index.

For our study the features with the trendability index greater than or equal than 0.6 is selected for calculating the degradation index. The threshold for the degradation index is of random and selected from the iteration method. The value which shows the failure of the blade appropriate is selected. Below figure shows the calculated trendability values of features extracted from the dataset no:15. Out of 147 features only 8 was used since only they exhibit a clear trend over the period of time.

The information about the sensor features from the csv file is given below.

- Component 1 and 2 - Eddy current sensor
- Component 3 - Acoustic sensor
- Component 4 - Laser sensor
- Component 5 - Acceleration sensor 1 (5_1 -x axis, 5_2 -y axis, 5_3 – z axis)
- Component 6 - Pulse generator
- Component 7 - Temperature sensor
- Component 8 - Acceleration sensor 2
- Component 9 - Strain gauge

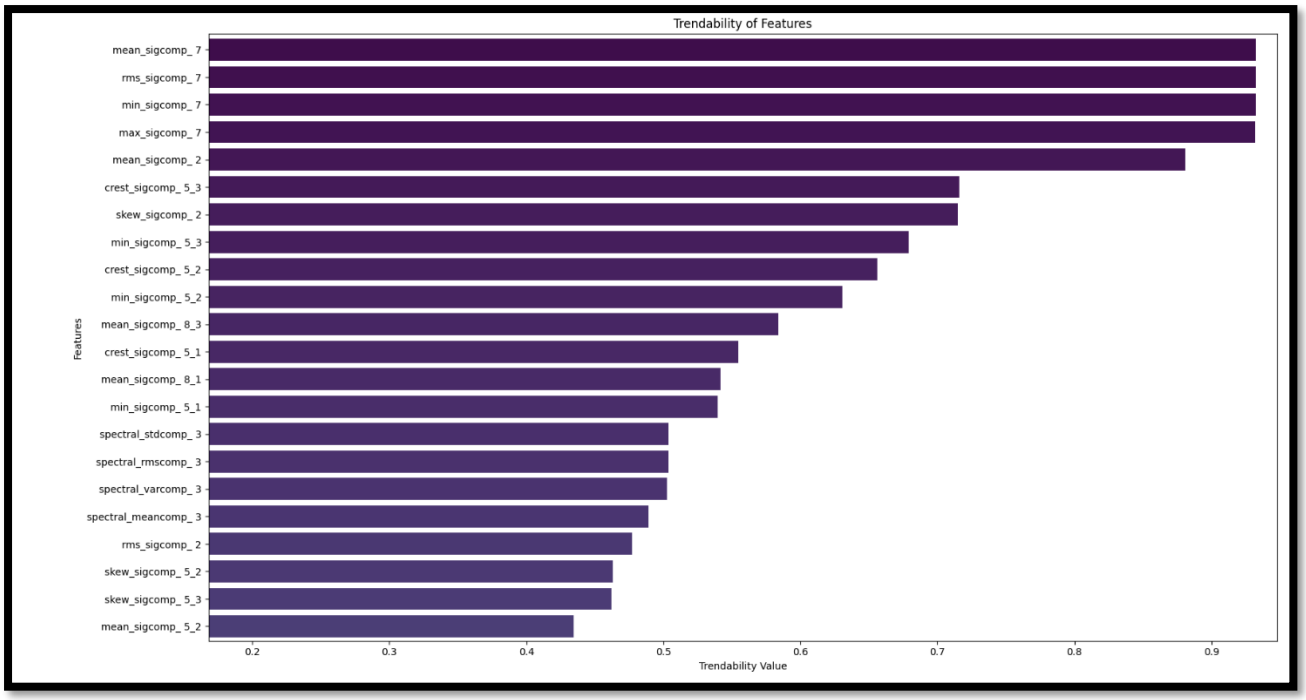

***Figure 2**: Trendability values of features*

Creating a linear degradation model to fuse the selected features involves a methodical approach to quantify the system's health using relevant features. This process is known as performance assessment as it is very essential for predicting the machinery's performance and calculating the remaining useful time (RUL). One of the common methods used is the logistic regression which transforms the multi-dimensional features into single degradation index or else called as Health Indicator (HI). These methodologies are used in models like ARMA (Autoregressive Moving Average) to predict the performance. But use of logistic regression has a drawback. It can distort the original degradation pattern if it follows an exponential trend.

To overcome this issue and preserve the original degradation pattern, Linear regression model is used instead of a logistic one. The linear regression model can be represented as

$$y = \propto + \sum_{i=1}^{N} \beta_i x_i + \varepsilon$$

Where $x = (x_1, x_2, \ldots\ldots, x_N)$ is the N dimensional feature vector,

Y is the health indicator or degradation index,

$(\alpha, \beta) = (\alpha, \beta_1, \beta_2, \ldots\ldots, \beta_N)$ are the N+1 model parameters and ε is the noise term.

Here α is kept as zero and β is the trendability value of the individual features. The linear regression doesn't have the range of [0, 1], while the logistic regression has the range of [0,1]. This doesn't change the effect of estimating the system's RUL. In our case the normalised degradation index 0 indicates the healthier machine or system and the value of 1 denotes the failure system or broken blade. In the figure below the degradation index of the feature 15 csv

file is given. Where the failure occurs at the 127th minute where the value of the degradation index is close to 1 or almost 1 and at the beginning its at 0. Since the data from the sensor has too much noise, a non-linear filter i.e., median filter is used to eliminate the noise.

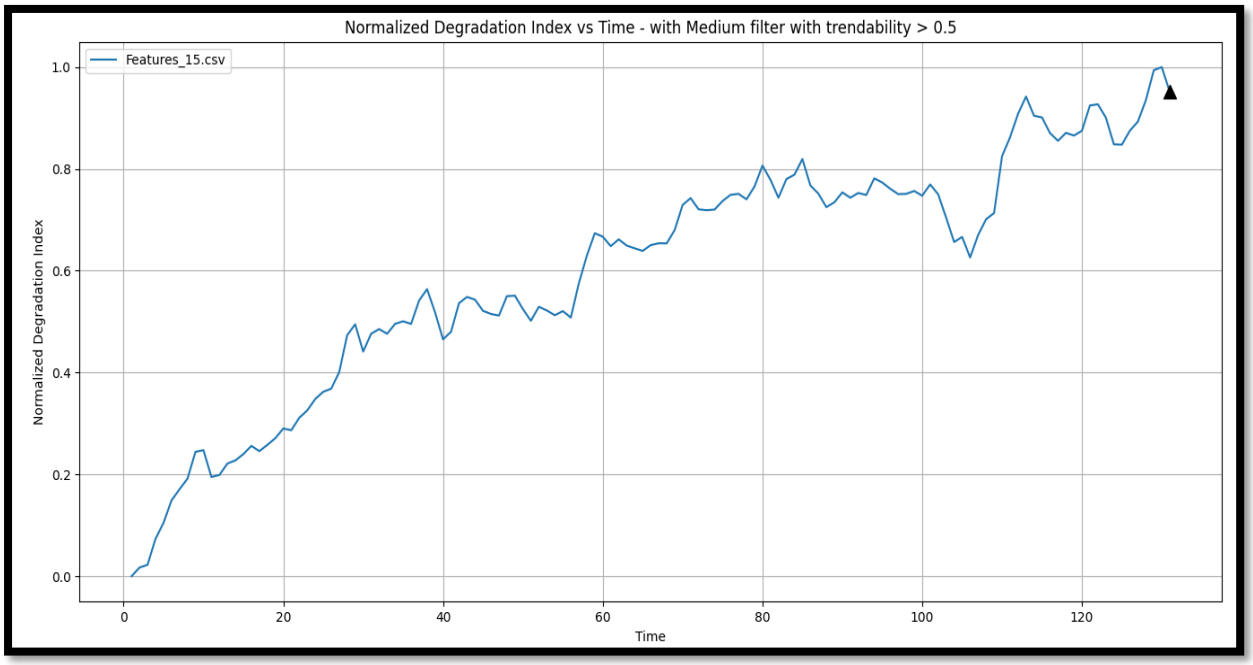

Figure 3: *Normalised degradation index*

5. Data Augmentation:

We have received only 5 dataset features. But these 5 won't be sufficient for the training of the neural network model. So, data augmentation has to be done with these features. One of the data augmentation techniques that can be used for the machinery datasets is grouping based on the operating conditions. There are 4 different operating conditions including the type of blade used, cutting speed, blade loading and the twisting program.

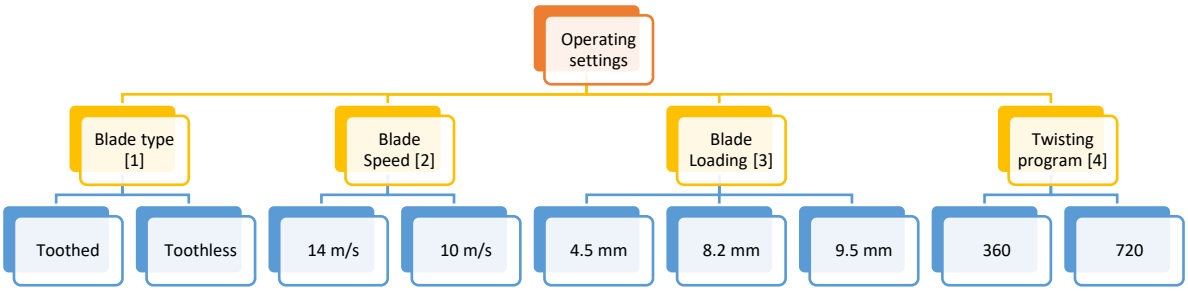

Figure 4: *Different operating conditions*

So, using the above different operating conditions we can group them using one operating condition to different combinations of operating conditions. So, there are sub categories in each operating conditions tend to attain a greater number of combinations. This results in 75 different groups of datasets. The following table illustrates the combination of operating conditions taken into consideration.

Group	Combination of Operating conditions
1	Operating condition 1
2	Operating condition 2
3	Operating condition 3
4	Operating condition 4
5	Operating condition 1 + 2
6	Operating condition 1 + 3
7	Operating condition 1 + 4
8	Operating condition 2 + 3
9	Operating condition 2 + 4
10	Operating condition 3 + 4
11	Operating condition 1 + 2 + 3
12	Operating condition 1 + 2 + 4
13	Operating condition 2 + 3 + 4
14	Operating condition 1 + 3 + 4
15	All Operating conditions

Table 4: *Combination of operating conditions*

After grouping the feature dataset according to the above combination of the operating conditions they have to be normalised. It is a technique used to adjust the data values to a common scale without distorting the difference in the range of values. One of the most common methods of normalization is min-max normalization. The formula for the normalization is

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Where:

- X is the original value,
- X' is the normalized value,
- X_{min} is the minimum value in the dataset,
- X_{max} is the maximum value in the dataset.

This method scales the original data values to a range between 0 and 1. After normalization the grouped dataset have to be regrouped to the original dataset feature numbers. Also, the duplicate dataset has to be deleted which might cause some discrepancies further. As a result,

the 75 features dataset is reduced to 25 different datasets from which the degradation index can be calculated as explained above, which can be used to train the GAN model.

6. Theoretical Background:

6.1. Generative Adversarial Network (GAN):

GANs are one of the most powerful machine learning models introduced by Ian Goodfellow and his colleagues in 2014. The architecture consists of two neural networks, the generator and the discriminator. The generator is aimed to generate data that looks as real as possible and the aim of the discriminator is to distinguish the examples as real or fake. Both the generator and the discriminator are trained through a process of adversarial training. This leads to a dynamic equilibrium game between the two neural network which compete against each other results in continuous improvement.

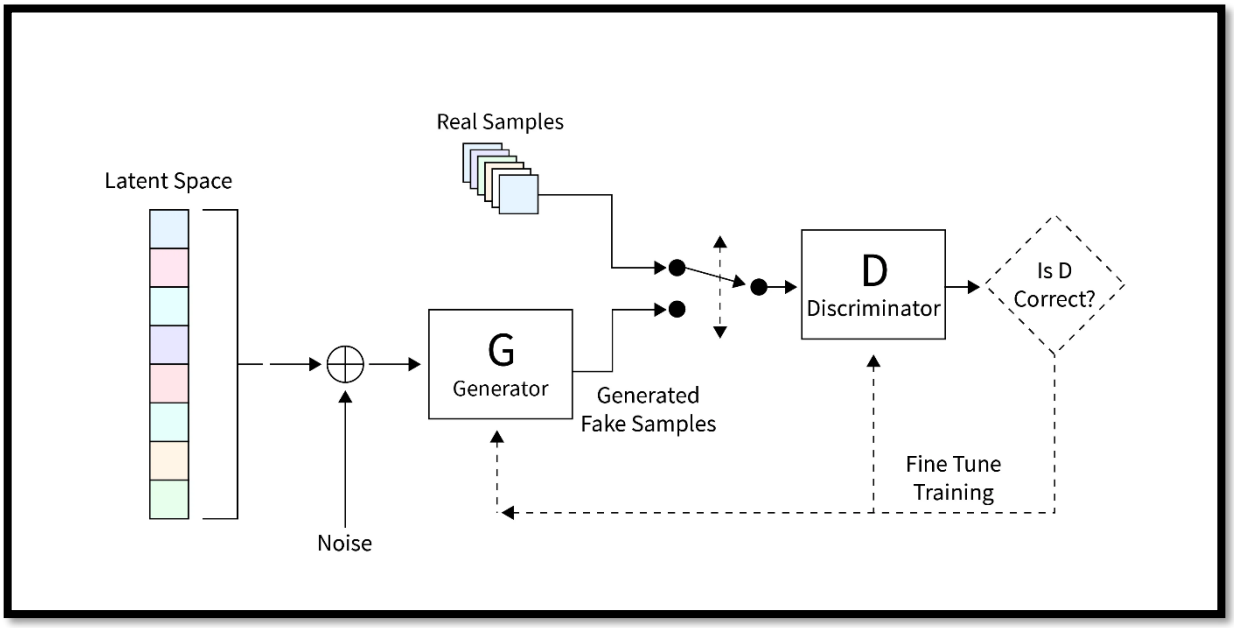

***Figure 5**: Architecture of General GAN*

The primary objective of GAN is to generate data that is indistinguishable from the real one, which make them useful for tasks where creating realistic data is crucial. The generator's role is to create fake samples that mimic the real data. It begins with a random noise and transforms it into a real modulated data through series of layers. Also, the generator's data has to fool the discriminator by classifying it as real. During training the generator learns to create increasingly convincing data by adjusting the parameters.

On the other hand, the discriminator acts as critic or judge that evaluates the authenticity of the data received from the real data as well as from the generator. It outputs a probability indicating the given sample as real or fake. The feedback from the discriminator helps to improve the generator. The training of GANs is iterative and adversarial. However, the generator and discriminator are trained at the same time, the generator improves its ability to create realistic data and the discriminator improves its capacity to detect fake data.

There are various types of GAN that have been proposed such as Conditional GAN (CGAN) which uses extra label information to improve the discriminator's accuracy. Other GAN like Wasserstein GAN (WGAN) which includes Wasserstein earth mover distance to the loss function. Also, WGAN with gradient penalty adds regularization to the loss function.

The loss function of the normal vanilla GAN discriminator is

$$-\frac{1}{m}\sum_{i=1}^{m}\log D\left(y^i\right) \; -\frac{1}{m}\sum_{i=1}^{m}(1-\log D(G\left(x^i\right)))$$

The loss function of the generator is

$$-\frac{1}{m}\sum_{i=1}^{m}\log D(G\left(x^i\right))$$

Where x is the input for the generator, y is the target from the real dataset, $G(x^i)$ is the generated fake target from the generator.

The discriminator of the normal Vanilla GAN is not as powerful enough when compared to the other types of GAN. This is due to the discriminators unstableness and they are tending to be slow. So, in order to overcome this a new type of GAN is proposed, which is called WGAN-GP. It uses Wasserstein distance to solve the above-mentioned issue. Earth mover distance is the minimum cost of transporting mass in converting the data distribution. Unlike vanilla GAN, WGAN-GP is without sigmoid function and gives result as a scaler instead of a probability.

The loss function of WGAN-GP discriminator is

$$-\frac{1}{m}[D\left(y^i\right)-D\left(G\left(x^i\right)\right)+\;\lambda(\left\|\nabla D_{y^i\sim x^i}\right\|_2-1)^2]$$

Where λ is the gradient penalty coefficient that penalises the discriminator to a higher extent if it fails to distinguish the data as real or fake.

The loss function of the generator is

$$-\frac{1}{m}\sum_{i=1}^{m}D(G\left(x^i\right))$$

6.2. Variational Autoencoders (VAE):

A variational Autoencoder (VAE) is a generative machine learning model that aims to learn a probabilistic mapping from an observed data to a latent dimension and back. It was introduced by Kingma and Welling in 2013. By reintroducing the probabilistic mapping, it differs from the traditional VAE. In VAE, the encoder maps the input data to a latent space

distribution which is most often a Gaussian distribution. The encoder produces parameters like mean and variance of the distribution instead of producing a single point in latent space.

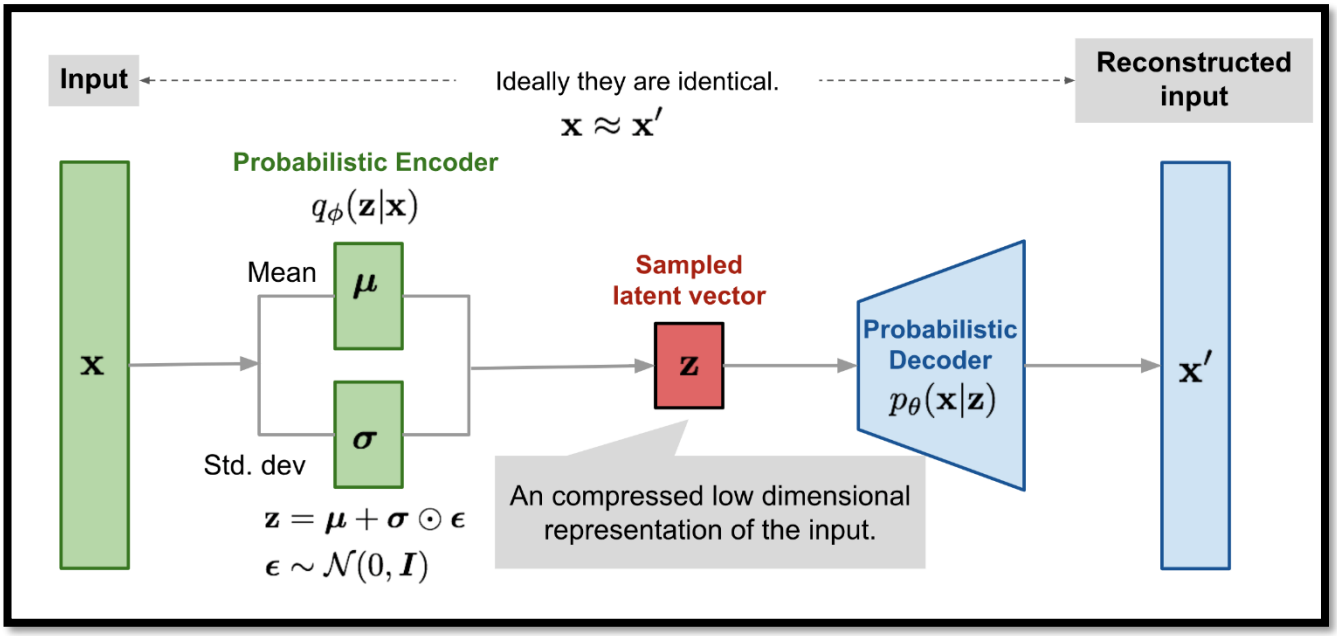

***Figure 6**: Architecture of a VAE*

The decoder network reconstructs the data from the sampled latent variables. This probabilistic approach to encoding the input allows VAEs learn more continuous and structured latent space representation. This is useful for generative modelling and new data synthesis.

The loss function of a VAE contains two main components, the reconstruction loss and the regularization loss. First the reconstruction loss measures how well the VAE is able to reconstruct the input data. Whereas the regularization loss whether the latent space is following a specified prior distribution which can be gaussian or normal distribution.

The overall loss function of a VAE can be written as

$$\mathcal{L} = \mathcal{L}_{reconstruction} + \mathcal{L}_{KL}$$

The reconstruction loss measures the difference between the original input and the output generated by the decoder. In other words, it measures how similar the reconstructed output is when compared to the actual input. It is represented as the negative log- likelihood of the reconstruction given the input. It is written as

$$\mathcal{L}_{reconstruction} = -\mathrm{E}_{q_{\phi(Z|X)}}[log\ p_\theta(X|Z)]$$

Where:

- X is the input data,
- Z is the latent variable,
- $q_{\phi(Z|X)}$ is the encoder,
- $p_{\theta(X|Z)}$ is the decoder.

The regularization loss is often termed as Kullback-Leibler (KL) divergence, measures how the encoder's distribution i.e., the learned probability distribution is diverging from the prior distribution. It can be written as

$$\mathcal{L}_{KL} = D_{KL}(q\_\emptyset\ (Z \mid X) \parallel p(z))$$

This term can be analytically written as

$$\mathcal{L}_{KL} = -\frac{1}{2}\sum_{j=1}^{d}(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2\ )$$

Where:

μ_j and σ_j are the mean and the standard deviation of the latent dimension Z for the j^{th} dimension.

Combining both the loss function, the total loss function of the VAE is given as

$$L_{Total} = -\mathrm{E}_{q_{\phi(Z|X)}}[log\ p_\theta(X|Z)] + -\frac{1}{2}\sum_{j=1}^{d}(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2\ )$$

VAEs have many advantages over traditional autoencoders. VAE allow for many generative modelling, creating new data points from the latent space distribution. This allows for continuous latent space representations which means interpolation can be done between different points in the latent space to generate new novel data points. Also, they are less prone to overfitting than the traditional ones. But VAE are often more difficult to train and they require more computational resources.

7. Architecture Implemented:

7.1. VAECGAN:

The VAECGAN (Variational Autoencoder Conditional Generative Adversarial Network) is one of the advanced deep learning models that combines the strength of both the Variational Autoencoders (VAE) and the Conditional Generative Adversarial Network (CGAN). These architectures are generally used for producing high quality, realistic data by using the abilities of VAE and adversarial training of GANs.

The VAE component consists of two main parts. One is the encoder and the other one is the decoder (generator). The encoder here is similar to that of a normal VAE. It receives the input data, in our study it's the multi time series degradation index data. The encoder aims to convert the high dimensional input time series into a lower dimensional latent space. Here we have decided to go with a latent space of dimension 10. It captures some of the essential features of the data. The encoder gives the output like mean and variance of data, which define a particular distribution from which the latent variables are sampled. Also, it makes sure that the latent space is represented in both meaningful and in a compact way.

The architecture and components of VAECGAN is given below.

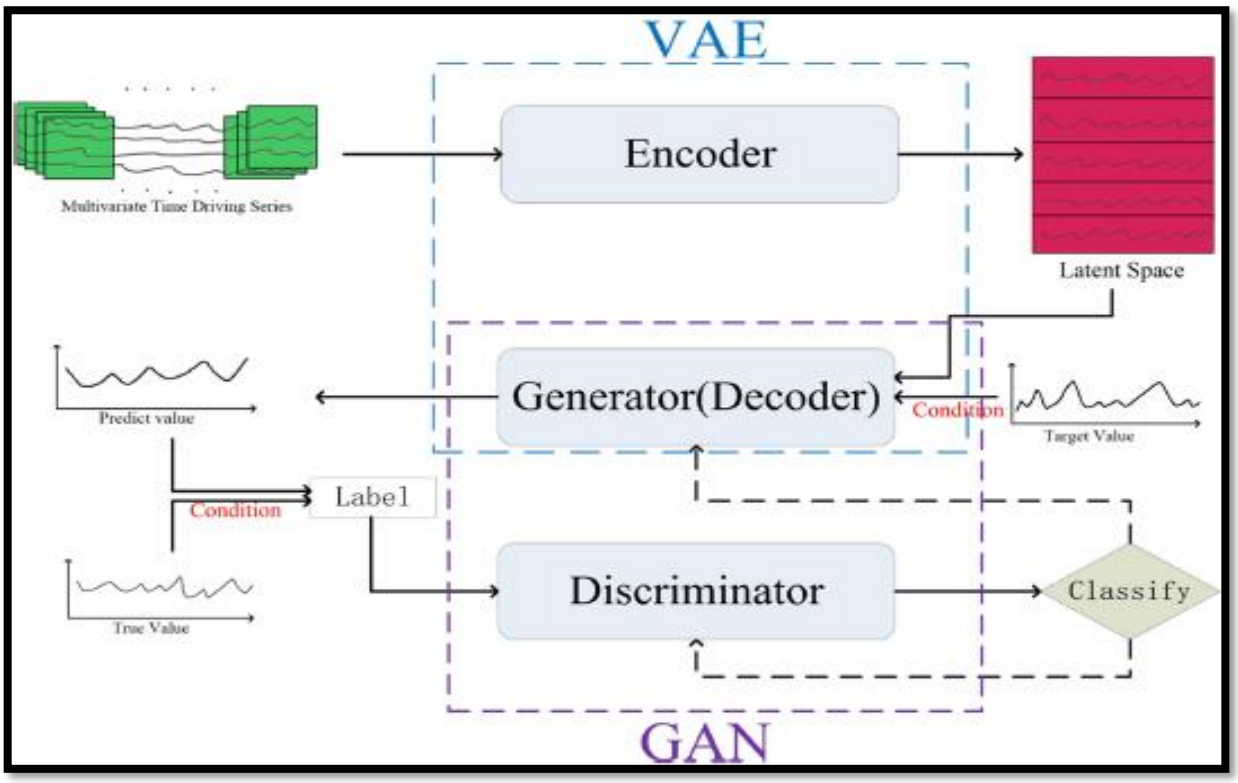

***Figure 7**: Architecture of VAECGAN*

After the data is encoded into the latent space, now the decoder that is the generator takes over. Its task is to reconstruct the original input data from the latent space. This decoder also some information such as labels and target values, which produces relevant and accurate reconstructions data. This one helps the model to generate data that is more realistic compared to the original input data.

The GAN component of the VAEGAN architecture comprises of the generator and the Critic. The VAE's decoder is same as the GAN's generator here. The critic which is another neural network tries to distinguish between the real and the fake data generator from the generator. It receives both the real and the generated data and then it classifies the input as fake or real, giving more feedback to the generator to improve its efficiency. The integration of VAE and GAN is where the architecture of VAECGAN lies.

The latent space produce by the encoder is fed into the generator. The generator tries to fool the discriminator by producing more realistic data. The discriminator's feedback is very important to improve the ability of generator to create realistic data.

7.2. Training and loss functions

The training of VAECGAN involves optimizing the two loss functions, the reconstruction loss and the adversarial loss. The reconstruction loss from the VAE ensures how the generated data is similar to the original data. It usually takes the form of a mean squared error (MSE) or sometime Binary cross-entropy (BCE).

The adversarial loss from the GAN, ensures how effective the discriminator in distinguishing the data as real or fake. This loss function improves the ability of the generator to produce fake data that is indistinguishable from the real data.

7.3. Sliding windows

Sliding window refers to a fixed subset of sequential data that will slide across the data at each step. The sequential data are processed in chunks, moving step by step to capture different portions of the dataset.

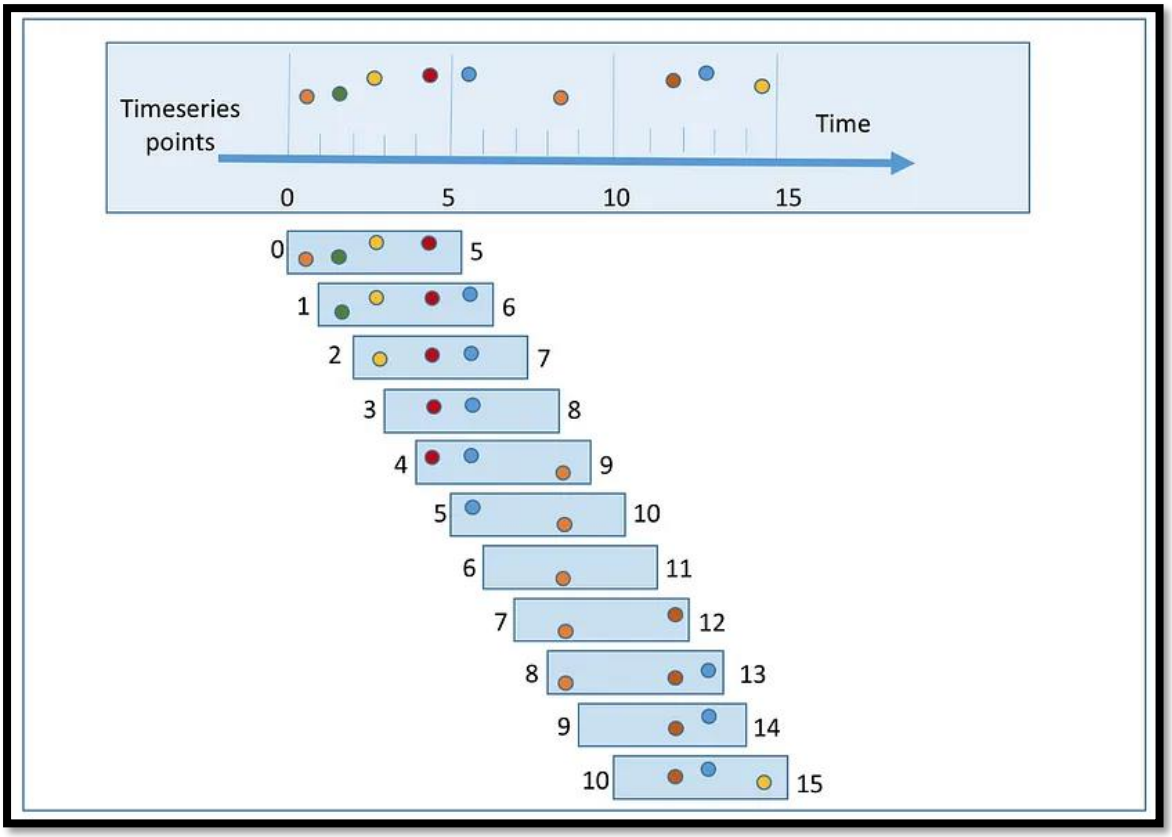

***Figure 8**: Sliding windows for Time series data*

A window of fixed size has to be defined first for creating a sliding window. This will extract a chunk of data at a time. This window slides over the sequential dataset by a certain strides or step size. After one chunk of data being processed, the window moves forward by a certain number of steps and process the next chunk of data. The windows can sometimes overlap or be discontinuous. Some part of the previous window's data will contain in the other window in case of overlapping. This method is used to create input output pairs, where the past data are used to predict the future data.

7.4. Structure of the Generator and Discriminator

The generator begins with an input layer that receives the multiple time series degradation indices and the latent space representation from the encoder which has the information of the data distribution. It includes multiple Gated Recurrent Unit (GRU) layers, which are well known for capturing the temporal dependencies in the time series. It is followed by the dense layers which process the data further. The final generator output is reshaped to match the format of the discriminator's input.

The discriminator begins with an input layer that receives the both real and fake datasets. It contains 1-Dimensional convolutional layers (Conv1D) to process the sequential data. After convolution the data is flattened and passed through a series of dense layers that tries to learn the discriminative features. The final layer of the discriminator provides the classification result whether it is real or fake.

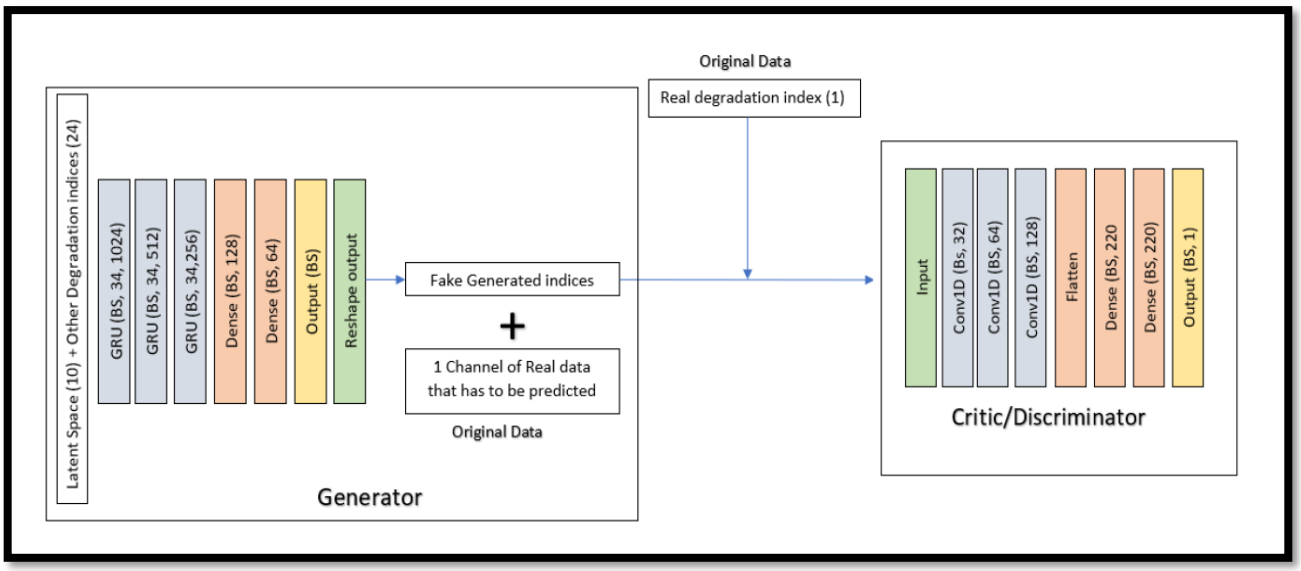

***Figure 9**: Structure of Generator and Discriminator*

The task of using the observed time series in the past to predict the unknown or remaining time series in the future. Larger the prediction steps, harder the problem. To overcome this issue, we use VAECGAN. The encoder of the VAE is to encode the data driving time series into a latent space and provide it to the generator, such that the latent space representation doesn't contain random noise and has a part of the data information of the driving series. The encoder processes the driving series and produces the latent space which can keep the relationship information of the series. In the discriminator the convolution layers are mainly used to extract the data features and to distinguish the real and generated data. The CGAN module can improve the capability of the VAE module to generate long term time series data. We code the exogenous sequence of the temporal data through the VAE instead of the noise as the generator's input. As the random noise input to the generator doesn't produce time series of good quality and it doesn't look realistic.

7.5. Concatenation of Fake with the real data:

The generated fake data from the generator of the GAN is concatenated with the part of the real data. The main idea of this is to have a meaningful way. By doing this, the model is designed to retain some part of the real information while introducing the synthetic generated part. In time series and some images where the continuity of the data matters, its crucial to have some part of the real data and generate the synthetic data for certain dimensions. This keeps part of the context from the real data making it more plausible and easier to align with reality.

Another reason is the conditional generation. The GAN's generator will be used to predict the future sequence or some of the missing features while preserving the real part of the

input as the conditioning information. If the generator is only responsible for generating a specific portion of the data, like the future sequence of the data or the missing portion, it can focus its capacity on generating the high-quality output for the specific part. If so, the real data remains unchanged, and ensures that the result remains close to the real actual data. By preserving the part of the real data, the generator tends to generate contextually consistent fake data. This helps the model to attain a balance between real and synthetic data ensuring the generated data fits smoothly into the real data, resulting in more realistic and useful outcomes.

8. Results:

The performance of the machine learning model is usually evaluated by many metrics. Here we have used the Root Mean Square Error (RMSE) as the indicator and it is defined as

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N}(x_i - \hat{x}_i)^2}{N}}$$

Where:

- N is the number of data points,
- X_i is the actual data,
- $\widehat{X}_i$ is the predicted data.

As already explained in the theoretical part, the VAE is trained using the 24 different time series degradation indices to attain the laten space representation of dimension 10. The following are the outline and hyperparameters of the VAE model.

- input data dimension = 24
- Number of hidden layers = 3
- Number of neurons in each layer = 400
- Output dimension = 10
- Learning rate = 0.00003
- Number of epochs = 500
- Optimizer used = ADAM
- Loss function = BCE (Binary Cross entropy) + KL divergence

Below is the loss function of the VAE while training it for over 500 epochs. The loss function seems to get saturate after 400 epochs.

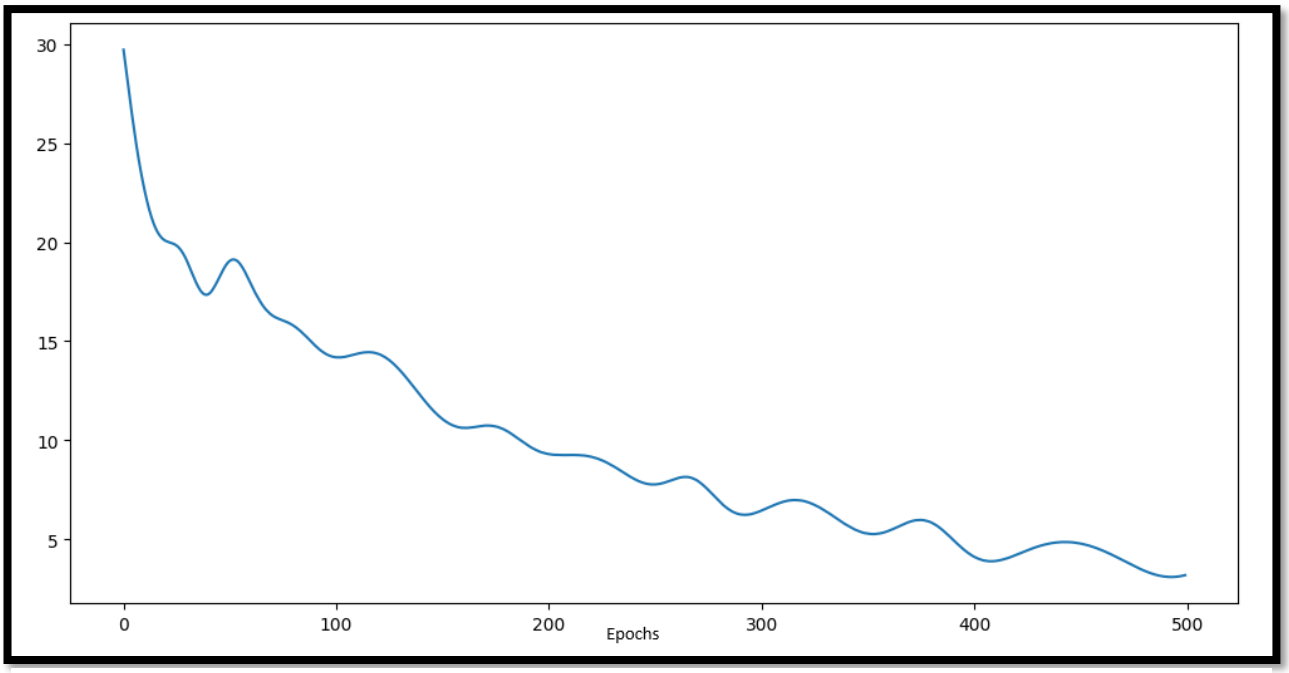

***Figure 10**: Loss vs Epochs in training the VAE*

After training the VAE the latent space is derived. The latent space along with the other degradation index is fed into the generator of the GAN. Before feeding the generator, the data is given to a sliding window to create chunks of data. A window size of 3 is chosen to get an effective output.

The discriminator is given both the real degradation index and the fake concatenated degradation index and its role is to distinguish it whether its fake or real. The purpose of the concatenation is already explained in the previous part. The data set is split into training and testing data. The training data is of size 60% and the test data has the remaining 40% with 20 datapoints of the training data to match the future sequence.

The optimizer and the hyperparameters used in the GAN are as follows,

- Batch size = 64,
- Learning rate of generator = 0.000115,
- Optimizer of generator = ADAM,
- Weight decay = 1e-3
- Learning rate of critic = 0.000115,
- Optimizer of critic = ADAM,
- Weight decay = 1e-3
- Critic iterations = 3,
- Lambda_gp = 10
- Number of epochs = 200

The GAN model is trained for 200 epochs. The blue solid line is the generator loss and the black solid line is the critic loss. After 150^{th} iterations the minmax game of the GAN tend to attain an equilibrium state. The discriminator loss is maximum at the beginning and got reduced after training several epochs. The adversarial training of the Generator and the discriminator is shown below.

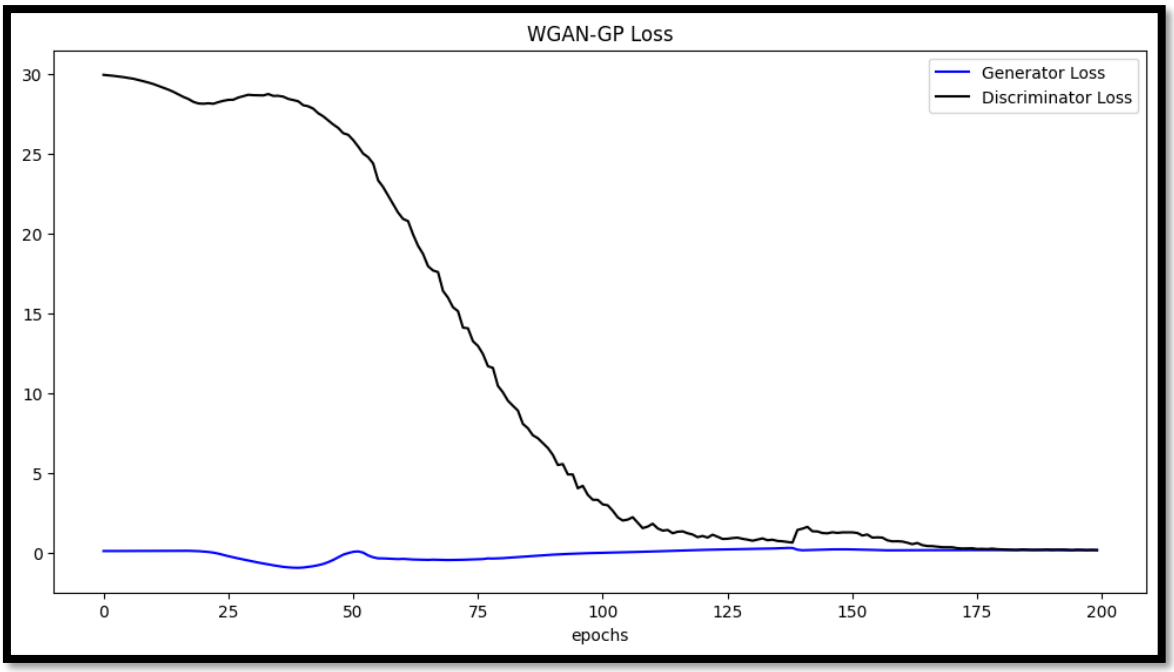

***Figure 11**: Training of GAN*

There are totally 25 different time series degradation indices. Here the feature **"1**9_group_14_360.csv**"** dataset has to be predicted for its future sequence. It is attained after grouping and normalization. Then the linear degradation index is calculated. The prediction of the training dataset from the GAN is shown below.

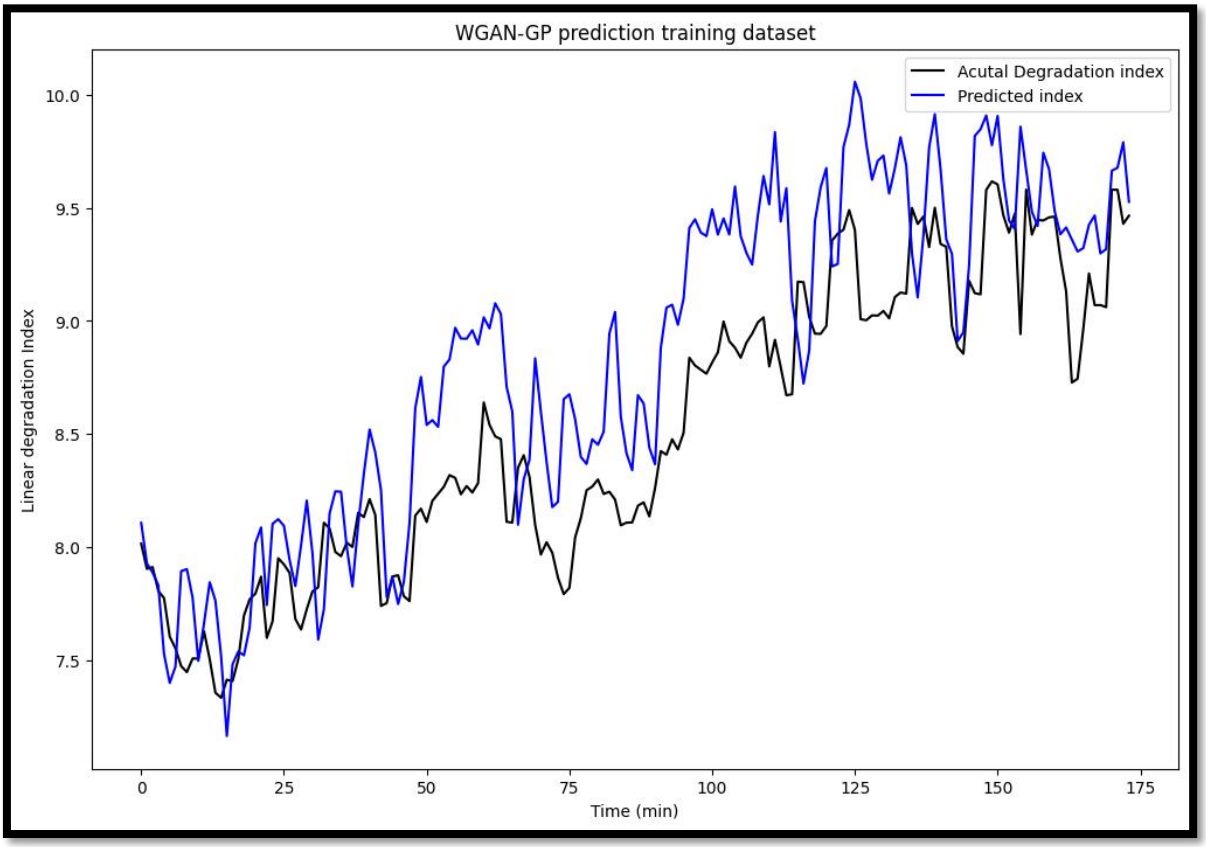

***Figure 12:** Training dataset prediction*

The root means squared error (RMSE) of the WGAN-GP for the training dataset is calculated as 0.435776.

The prediction of the test data set using the WGAN-GP is given below.

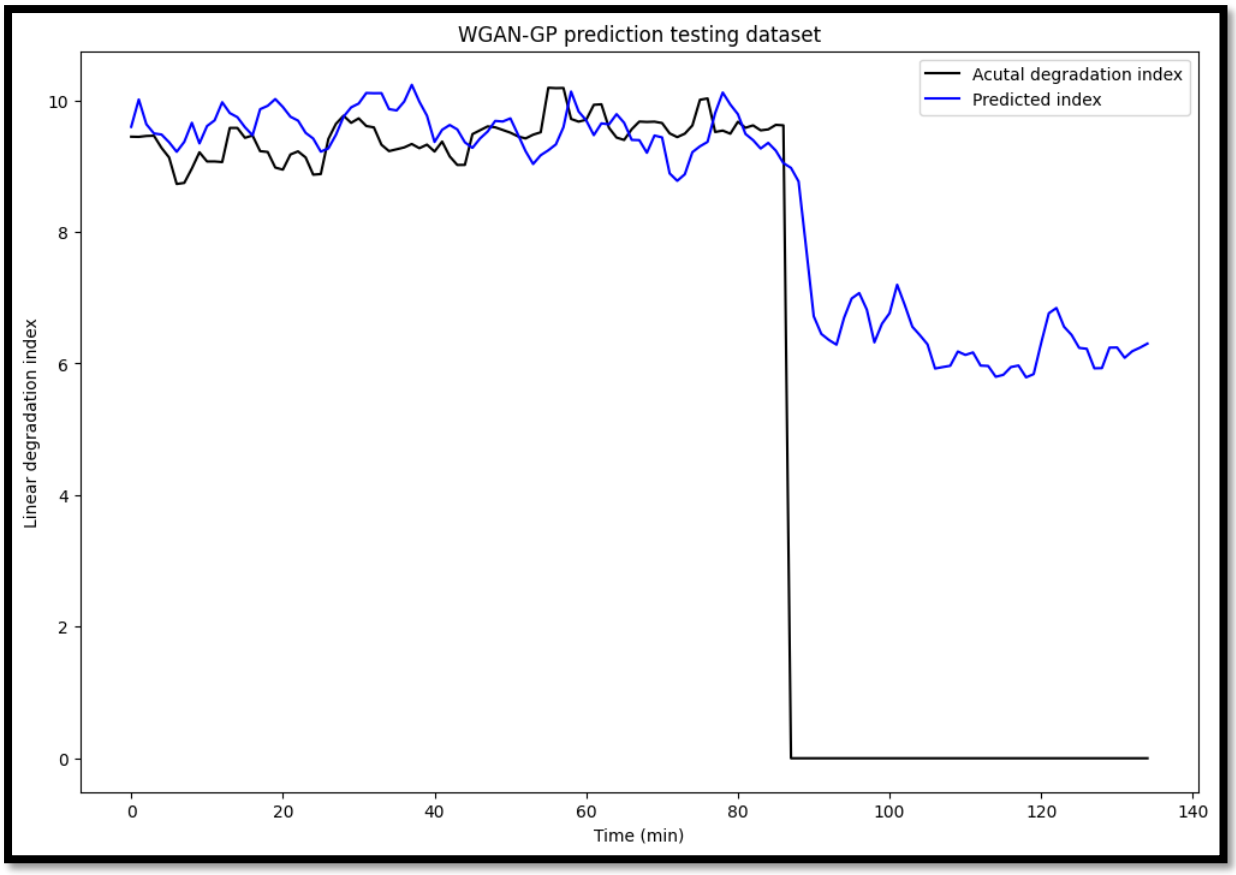

Figure 13: *Test dataset prediction*

The time stamp starts from zero but it's actually the continuity of the training dataset with 20 data set prior. The test data time starts from 155 min (175 min - 20). The sharp serge in the actual degradation index at the time (88 min i.e., 155+ 88 = 243 min) indicates that the band blade has been broken. Eventually the predicted index also shows some sharp or maximum gradient after the 88th min. The model somehow actually predicts the failure, also it generates synthetic data that is more realistic. The RMSE of the WGAN-GP for the test data is calculated as 3.88705.

9. Summary:

Generation of synthetic time series data for the degradation index was successfully carried out in this project. The implementation of the WGAN with Gradient Penalty along with VAE produces more realistic time series rather than other methods. Earlier methods like Signature GAN, Time GAN, RITS and several other GAN were tried, but they never tend to produce realistic time series data. Here the concatenation of the fake data with the real data makes it possible to look like a realistic and promising one. The work done is actually a mimic of stock market future trend prediction. But before going it straight further pre-processing has

to be done knowing the domain specifications. There are still many research teams proposing reinforcement learning for the time series prediction.

References:

1. *Z.Liang, J.Gao, H.Jiang, Xu Gao, Z.Gao, R.Wang, (2018) "A similarity based method for remaining useful life prediction based on operational reliability", Xi'an Jiaotong University , china.*
2. *Zhang, K., Zhong, G., Dong, J., Wang, S., & Wang, Y. (2019). Stock market prediction based on generative adversarial network. Procedia computer science, 147, 400-406.*
3. *Chen Chen, HungChun Lin, "Stock price prediction using Generative Adversarial Networks" MIT. Access: https://github.com/ChickenBenny/Stock-prediction-with-GAN-and-WGAN/tree/main.*
4. *Banushev, B., (2020). Using the latest advancements in AI to predict stock market movements. Access: https://github.com/borisbanushev/stockpredictionai.*
5. *Xiang Yin, Yanni Han, Zhen Xu, Jie Liu, (2021), "VAECGAN: a generating framework for long-term prediction in multivariate time series", Institute of information Engineering, Chinese Academy of Sciences, Beijing, China.*
6. *Zhou, X., Pan, Z., Hu, G., Tang, S., & Zhao, C. (2018). Stock market prediction on high-frequency data using generative adversarial nets. Mathematical Problems in Engineering, 2018.*
7. *Z.Liang, J.Gao, H.Jiang, Xu Gao, Z.Gao, R.Wang, (2018), "A Degradation Degree Considered Method for Remaining Useful Life Prediction Based on Similarity", ", Xi'an Jiaotong University , china.*
8. *Marcia L.Baptisa, Kai Goebel, Elsa M.P. Henriques, (2022), "Relation between Prognostics Predictor Evaluation Metrics and Local Interpretability SHAP Values"*
9. *Furkan Luleci, F.Necati Catbas, Onur Avci, (2022), "Generative Adversarial Networks for Data Generation in Structural Health Monitoring", United States.*
10. *Jun Dai, Jun Wang, Weiguo Huang, Juanjuan Shi, Zhongkui Zhu, "Machinery Health Monitoring Based on Unsupervised Feature Learning via Generative Adversarial Networks", Soochow University, Suzhou, Jiangsu, China.*
11. *Tianyi Wang, Jianbo Yu, David Siegel and Jay Lee, (2008), "A Similarity-Based Prognostics Approach for Remaining Useful Life Estimation of Engineered Systems", Department of Mechanical Engineering, University of Cincinnati, Cincinnati, China.*
12. *W. Fabian, K. Timo, B. Moritz, K. Markus, L. Marcus, (2022), "Generation of synthetic data with low-dimensional features for condition monitoring utilizing Generative Adversarial Networks", University of Applied Sciences, Aalen, Germany.*
13. *Jinsung Yoon, Daniel Jarrett, Mihaela Van der Schaar, (2019), "Time-series Generative Adversarial Networks", University of California, Los Angeles, USA.*