A Predictive Maintenance Framework for Remaining Useful Life Classification on IoT Enabled Devices using Neural Networks

MSc Research Project
Data Analytics

Raul Damian Sainz Calderon
Student ID: 19158696

School of Computing
National College of Ireland

Supervisor: Paul Stynes

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Raul Damian Sainz Calderon
Student ID:	19158696
Programme:	Data Analytics
Year:	2021
Module:	MSc Research Project
Supervisor:	Paul Stynes
Submission Due Date:	16/12/2021
Project Title:	A Predictive Maintenance Framework for Remaining Useful Life Classification on IoT Enabled Devices using Neural Networks
Word Count:	4,556
Page Count:	19

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Raul Sainz
Date:	31st January 2022

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

A Predictive Maintenance Framework for Remaining Useful Life Classification on IoT Enabled Devices using Neural Networks

Raul Damian Sainz Calderon
19158696

Abstract

The unprecedented technological advances in the fields of the Internet of Things (IoT), Artificial Intelligence (AI) are at the heart of the new industrial Revolution known as "Industry 4.0" which has pushed improvements in the manufacturing industry, especially in predictive maintenance research and technologies. With the simplistic premise of fixing a machine before it fails, companies and organizations can achieve great improvements in productivity by reducing downtime and machine failures. Remaining Useful Life (RUL) is a metric used to forecast the performance of the system to predict the time (cycles) left before the machine fails. However, the prognosis RUL can be a challenge, many researchers have used many model-based and data-driven models for trying to solve these issues. The proposed framework compares the performance of most widely used data-driven machine learning models LSTM and CNN versus the state of the art TCN to predict and classify if a machine will fail within a certain time range. The popular dataset CMAPSS from NASA is used for running the experiments of this research. The results show promising results for the proposed model TCN even with noisy and complex conditions datasets. Additionally, techniques for dealing with imbalanced data are compared throughout the experiments, the use of weights when training the models resulted in more stable training/accuracy plots and improvement of classification for imbalanced label classification.

1 Introduction

Downtime is one of the most significant contributors to production inefficiency in the manufacturing industry, the cost generated by it is a key factor for many decisions on planning, projects and budgets. A manufacturing system usually contains a series of machines or workstations and operations required to be performed sequentially. Thus, without a proper **maintenance program** to timely repair and avoid downtime, a single machine failure can potentially spread throughout the system and consequently halt the production of goods causing great economic losses.Chang et al. (2012)

Predictive Maintenance (PM) aims to monitor machine's conditions and sensors to actively determine maintenance schedules, strategies and budgets. According to Wang et al. (2019) researchers have defined the two most important perspectives for PM: *1) Remaining useful life (RUL)*, that consist of estimating the time until the next failure or end of useful life; *2) failure prediction*, aims to predict **the probability of a machine**

failing within a specified time frame. In this research, we will focus on the failure prediction problem but the probability will be based on the RUL index.

Existing methods for PM problems are grouped into three main categories: model-based, data-driven and hybrid. A model-based approach can be more accurate for complex systems, however, they require extensive knowledge about the systems. Because of the development of Industry 4.0 and the Internet of Things (IoT) many machines have been equipped with sensors for monitoring the operational behaviour and health conditions of machines, favouring the heavy adoption of data-driven models for this task because of their ability to model the degradation characteristics based on historical sensor data. Li et al. (2018)

1.1 Research Question and Objectives

The aim of this research is to perform a review of the most widely used neural networks algorithms like Long short-term Memory (LSTM) and Convolutional Neural Networks (CNN) that have shown good results when approaching RUL prediction problems and compare their performance against the state of the art model architecture for sequential data: Temporal Convolution Networks (TCN), but instead of approaching the problem as a regression we will first perform a binary classification based on a failure threshold to perform binary classification and thus, responding the following research question:

- **To what confidence level can we perform failure classification and prediction of a machine based on their underlying operational conditions and sensor data using Temporal convolution Networks?**

Furthermore, this paper also discusses some measures to improve the performance of the models when dealing whit imbalanced classification labels.

The major contribution of this research is the introduction of a classification framework for predicting the failure of a machine using neural networks.

Section 2 provides an overview of the literature on the most widely used data-driven models for RUL prediction. Section 3 shows the methodology followed for this research and the steps to replicate the experiments. In 5 and 6 and the implementation and evaluation process is described along with the experiment results. Finally in 7 the conclusions of this research are stated.

2 Related Work

An extensive literature review was performed to understand the context and limitations of the application of the most widely used machine learning models for preventive maintenance and to discover the state of the art model and their benefits for RUL prediction problems.

2.1 Preventive Maintenance

Industrial revolution (I4.0), Internet of Things (IoT) and the novel machine learning techniques has created the perfect environment for research and innovation for major improvements for preventive maintenance strategies. according to (Barlow; 2015) it is estimated that a comprehensive predictive maintenance program can decrease maintenance cost by up to 30%, cut downtime by 45% and reduce machine failures by up to 75%.

In maintenance, there are four categories, corrective, preventive and prescriptive. Predictive maintenance is the focus of this study because it uses time-based information and knowledge to detect a possible failure and consequently avoiding down times.

2.2 Remaining Useful Life (RUL)

Different machine learning models, especially neural network-based have been used to solve RUL problems. The advantage of neural networks over other legacy ML models is that they can model complex, non-linear and multi-dimensional systems without prior expertise and their ability to handle raw sensor data directly feed into them Li et al. (2018).

Since the data usually comes from sensors collecting readings at each point in time, the nature of the data is of time series, A time series is a sequence of vectors, $x(t), t = 0, 1, ...,$ where t stands for elapsed time and x is a series of discrete data points, equally spaced in time. Frank et al. (2001). Therefore, In order to accurately estimate RUL the models to be used need be able to capture sequential information from a multi-variable time series.

According to Wan et al. (2019), one of the key challenges for multivariate time series is non-linearity and aperiodicity of data originated by short-term and long-term dynamical behavior, because of this, legacy models such as auto-regressive integrated moving average (ARIMA) are prone to overfitting in high-dimensional scenarios like RUL. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have been successfully applied but recently the use of Temporal Convolution Network (TCN) has been gaining popularity Wan et al. (2019) for problems related to sequential modeling.

2.3 Long short-term Memory (LSTM)

LSTMs are a special type of recurrent neural network (RNN), designed to solve the problem of vanishing and exploding gradients, they are characterized by an architecture that includes long and short term states that change based on three control gates: forget gate, input gate and output gate Li et al. (2019). Consequently, the memory cell is able to preserve its state over long duration's, that is **learning long-term dependencies that may influence future predictions**. Sequence learning models such as Recurrent Neural Networks (RNNs) have problems for modeling sequence information with long-term dependencies in the data. To solve this issues a Long short-term Memory (LSTM) approach is proposed in Zheng et al. (2017) for RUL estimation because it can potentially take into consideration the full sensor sequence data and detect hidden patterns even under multiple operating conditions. Obtaining better accuracy results based on RMSE (1862) over other compared models like CNN, SVR and MLP.

LSTM algorithm is usually the preferred model for many tasks involving sequential signal processing, however, its **computational requirements are relatively high** when compared with similar neural network architectures Li et al. (2018). Moreover, as stated by Wang et al. (2019) **LSTM depends on strong assumptions** such as the mathematical mappings from current and prior sensor readings in regards to the RUL label are the same along the sequence but this assumption **can introduce biases.**

2.4 Convolutional Neural Networks (CNN)

The usage of convolutional neural network (CNN) for RUL prognosis was proposed in Li et al. (2018), using an sliding window time approach for feature extraction 2.6, this approach showed the effectiveness when applied to the C-MAPSS dataset for aero-engine A. Saxena , K. Goebel (2008) achieving high accuracy when compared to other architectures such as RNN and LSTM.

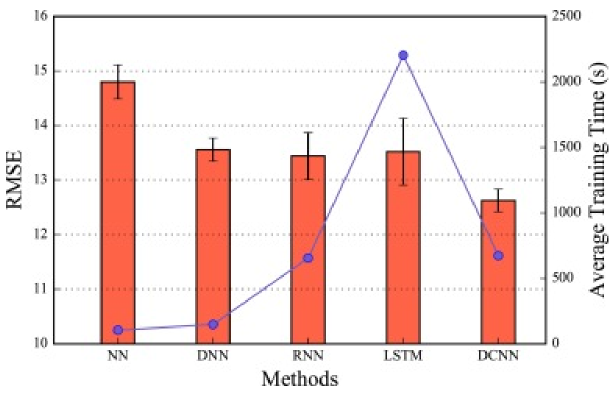


Figure 1: performance by different methods.

CNN are the most popular machine learning models for image data processing. Two key characteristics of CNNs are: spatially shared weights and spatial pooling Krishna and Baghaei (2019), however, recently CNNs have also been used for time series forecasting because of their capability of handling raw input data that makes them less dependent on prior knowledge. The first implementation of CNN for predicting RUL of jet engines was proposed in Babu et al. (2016), the model performed better than all the state-of-art papers at that time.

2.5 Temporal Convolution Networks (TCN)

(Koh et al.; 2021) proposed a novel approach for estimating RUL using deep convolutional neural networks with the premise that by exploiting the compositional locality of the time series, the invariant features can be extracted layer by layer. Real world time series problems such as RUL estimation and prognosis are plagued with issues like noisy, non-stationary, non-linear data. By using TCN architecture the author aimed to handle the highly variant time series, so that, shift-invariant features could be extracted layer by layer in order to boost the performance. The TCN's architecture seen in "Fig. 2" aims to solve the following issues: representation and distribution of a temporal context and the ability to learn from many layers.

TCN introduces the usage of causal convolutions Wan et al. (2019), in which an output at time t with elements from an earlier time in the previous layer, this layer is used to create one to one relationships in chronological order. The other component of the TCN are dilated convolutions, that contrary to the traditional convolutions don't require a pooling process in which some sequential information could be lost, instead , each convolution contains rich information for long-term tracking, making it ideal for long information dependence of sequences. Lea et al. (2017) performed various experiments using TCN that demonstrating their ability to capture complex patterns like time-delays, duration and compositions, outperforming other models like LSTM.

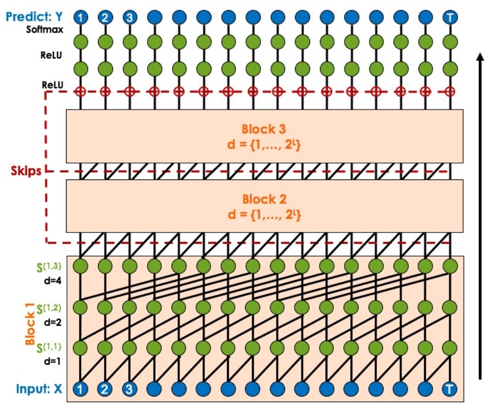


Figure 2: The TCN architecture is composed of a stack of deep delated convolutions to capture long range temporal patterns Lea et al. (2017)

2.6 Sliding window technique

Neural networks are used for time series forecasting using a sliding window approach as the standard method for performing time series prediction Frank et al. (2001). This method consist in using a set of N tuples as inputs and a single output as target value of the network "Fig. 3".

RUL estimation uses multi-variable time series data, better information can be obtained from a temporal sequence sampled at a single time step using a time window of size N_{tw} to collect high-dimensional feature vectors.

As reviewed by Li et al. (2018), the size window N_{tw} can have an effect on the performance of the models, the author performed a comparison between different time windows (1 to 60), finding that estimations were better as the window size increased but **no further improvement was shown after N_{tw} is above 30.**

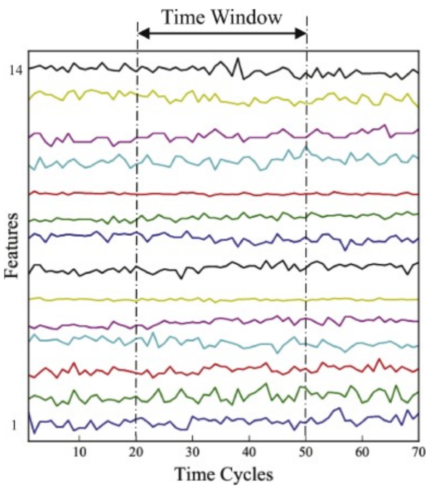


Figure 3: Training sample within a time window of length 30 Li et al. (2018).

This technique is used in the reviewed architectures LSTM Zheng et al. (2017) and TCN (Koh et al.; 2021).

2.7 Failure Threshold Estimation for Remaining Useful Life Classification

It is the scope of this research to approach the RUL prediction problem based on discrete-event as proposed by Li et al. (2019) by defining and labeling the samples into two classes making it a **binary classification problem**:

- **Healthy**: patterns and conditions in which a healthy condition remains after the same time interval
- **Failure**: patterns and conditions will lead to a failure after the defined time period.

The time interval t_{pred} is a fixed length RUL that needs to be defined as *the minimum time required for performing a maintenance action* (plan, schedule, prepare and execute) to avoid failure and operational disruption. The limitation of this concept is that t_{pred} must be defined depending on the characteristics of the organization systems, maintenance crew capacity and other requirements that will need to be considered as the optimal warning time to perform a maintenance operation. The author Li et al. (2019) approaches this issue by applying a Extreme learning machine (EML) with a single hidden layer because of the speed advantage for retraining the model, allowing the parameter to be defined dynamically based on the organization needs.

On the other hand, Chehade et al. (2017) states that most of the existing research on this failure threshold simply assumes that the t_{pred} or threshold is defined a priory and set to be constant and deterministic for all units. However, it points out that this can be problematic, to solve this, the authors propose to calculate and find random optimal threshold. The results showed that accuracy was improved when the number of observations was greater than or equal to 40 when the proposed method was used on the same C-MAPSS dataset used in this paper. A. Saxena , K. Goebel (2008).

2.8 Recurrence Plots

A recurrent plot (RP) is a technique of nonlinear data analysis. It is used to reveal the times when the phase space trajectory of the dynamical system visits roughly the same area in the phase space. RP's help to perform a visual inspection of higher dimensional space trajectories, providing hints about the evolution of these trajectories over time N et al. (n.d.).

RP's are a highly efficient and widely used tools for the analysis of time series data, they provide the possibility to estimate invariants and can be used to analyze non-stationary data because of their ability to **quantify hidden structures and patterns**. On their research Thiel et al. (2004), the authors performed experiments to demonstrate how much data is contained in this plots by trying to reconstruct the time series from the plots, they were able to successfully reconstruct them even from time series with stochastic components or corrupted by noise confirming their **ability to encode information**.

Recurrence plots have been used by Hsueh et al. (2019) to perform rotation machine condition monitoring in which the time series signals were converted into two dimensional images (2D) and feed into a neural network (CNN) to extract the key features and diagnose fault conditions by performing classification of the images, showing competitive performance.

2.9 Related Work Summary

As we can see, all three neural network architectures used for RUL prediction have been successfully used by researchers, each architecture has it's own benefits and drawbacks but the main point in common they all share is the ability to process raw non-linear and multi-dimensional features commonly found in IoT enabled machines with multiple sensors. This data-driven algorithms can be used to make estimations based on time series data but the most important characteristic is their ability to find patterns based on long term dependencies, therefore, it makes sens that the TCN architecture can perform better because of the dilation layers. However, this models require special techniques such as sliding windows and recurrence plots, that depend on parameters that can affect the performance of the models.

3 Methodology

The research methodology consist of five main stages shown on "Fig. 4"

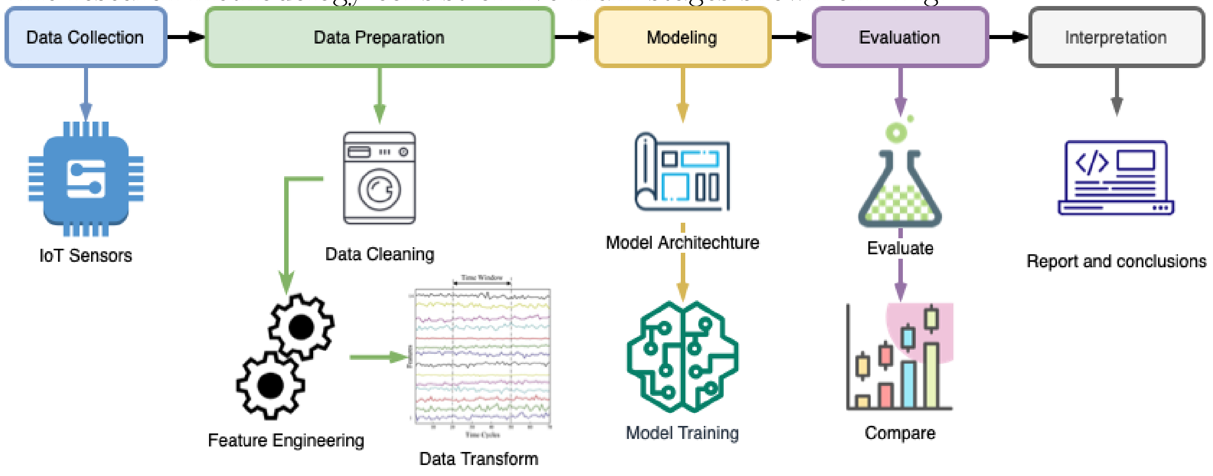


Figure 4: 5 stages methodology approach.

3.1 Data Collection

This stage involves acquire and collect set of data representing healthy and faulty conditions, perform exploratory visualizations analysis, clean and verify the quality of the data (remove outliers and feature selection).

3.1.1 Data Source

In this research, the proposed method is evaluated using the Li et al. (2019) the famous dataset Commercial Modular Aero-Propulsion System Simulation CMAPSS A. Saxena , K. Goebel (2008). The dataset contains a column to identify the engine (machine_id) and the cycle number that represents a time step of the life of the engine. The main features include: three operational settings along with 21 sensor data including low and high pressure compressor (LPC & HPC), fan, nozzle, high and low pressure turbine (HPT & LPT).

This dataset is divided into four different sub-sets simulating different combinations of operational conditions and fault modes along with sensor channels to represent fault evolution. The six different flight conditions include a range of values for the three operational conditions: Altitude (0 to 42K ft.), Mach Number (0 to 0.84) and TRA (20-100), these margins can change as the engine degradation progresses Saxena et al. (2008). Fault conditions can include HPC Degradation and Fan degradation.

Table 1: The C-MAPSS dataset

Sub-Set	**FD001**	**FD002**	**FD003**	**FD004**
Time series training set	100	260	100	249
Time series test set	100	259	100	24
Operating conditions	1	6	1	6
Fault conditions	1	1	2	2

FD001 contains sampled data from 100 engines operated under a single operating condition and only one degradation cause of engine's failure. On the contrary, FD004 is a more complex dataset since it includes a larger number of sample engines (249), each engine may be employed under 6 operating condition combinations from one cycle to another and the two types of fault conditions can occur.

The selection of FD001 and FD004 sub-sets for this research experiments is intended to compare the performance of the model between the simplest conditions of FD001 versus the more complex scenarios of FD004.

3.2 Data Preparation

3.2.1 Data Cleaning

The C-MAPSS dataset contains multiple engines measurements from 21 sensors, all columns contain numerical values, no NaN or missing values detected, but as seen on "Fig. 5" some features have constant outputs meaning that they do not provide any valuable information for the prediction, thus, they will be dropped from the data frame, the resulting dataset contains 16 features that will be used for the following processes.

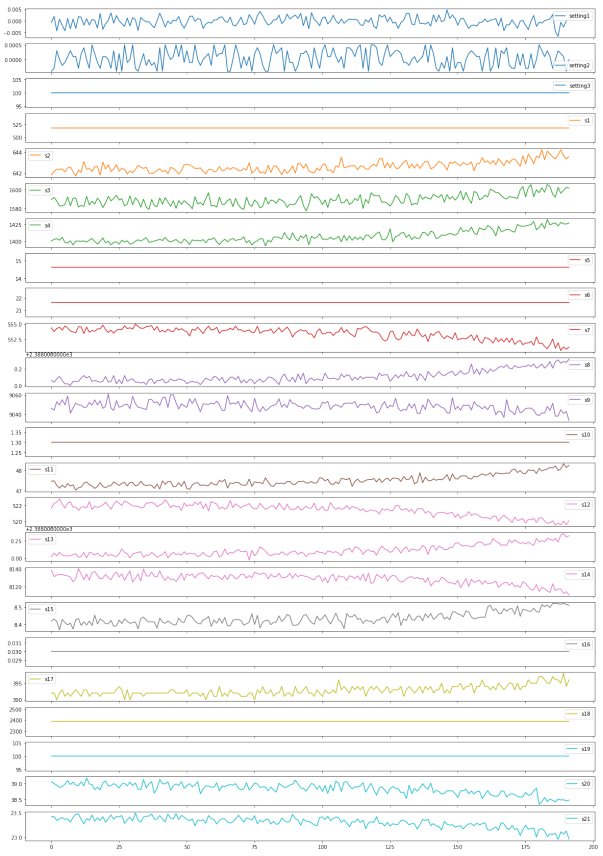


Figure 5: Plot of time series of machine Id 1

3.3 Feature Engineering

Each one of the selected features from sensors and settings data will be normalized used MinMax Scaler from sklearn library, each feature is scaled to a default range between zero and one.

3.4 Data Transformation

3.4.1 RUL Classification

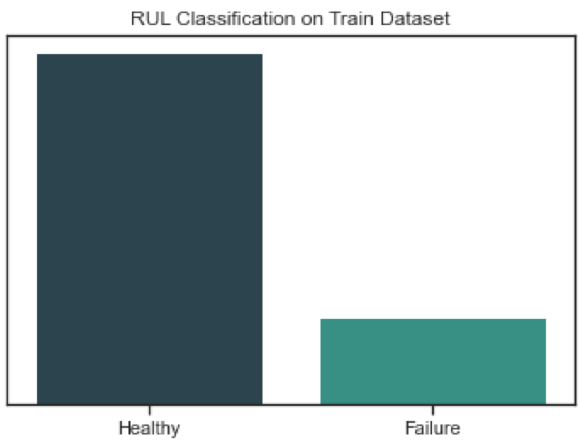


Figure 6: Training dataset FD001 has the following label distribution 80.1% of healthy (0) and 19.9% of Failure (1) samples.

Because of the binary classification approach proposed and as reviewed in section 2.7 a failure threshold will be applied to the RUL column of each engine to generate a new label to the dataset that indicates whether the sequence of a given machine cycle presents a "Failure" or "Healthy" condition, to generate this label we first need to calculate the remaining cycles of each machine at each time step, to do this, we add a new column named "rul" and calculate the value by grouping the dataset by the machine id and resting each cycle to the max number of cycles of each machine, for example, machine #1 has a maximum number of cycles of 192, meaning that there are the same number of data samples, we proceed to assign the current cycle minus the max number of cycles to estimate the number or remaining useful cycles of the machine at each step.

Once the column "rul" is created, we proceed to apply the threshold reviewed on 2.7 of to create a new binary column "rul" to categorize weather the condition is healthy (0) of failure (1). As we can see in "Fig. 6" the **distribution of the generated classes is imbalanced** and we will need to apply corrections in the training of the model to try to compensate for this condition using class weight's on the training process.

3.4.2 Time Sequence Processing

As covered in the methodology section and using the same method used by Li et al. (2018), the multi-variate time series is better approached by obtaining a temporal sequence using a time window 2.6 technique, as proposed by the author we will **use the same windows size of 30 cycles**.

3.4.3 Recurrence plots (RP)

CNN models are used for image classification, thus, we will need to transform the time series to images using the recurrence plots technique discussed in 2.8, for this we generate sequences (17,631 for training and 10,096 for testing using FD001) and store them into an new array, we can see a random example of the generated images in "Fig. 7"

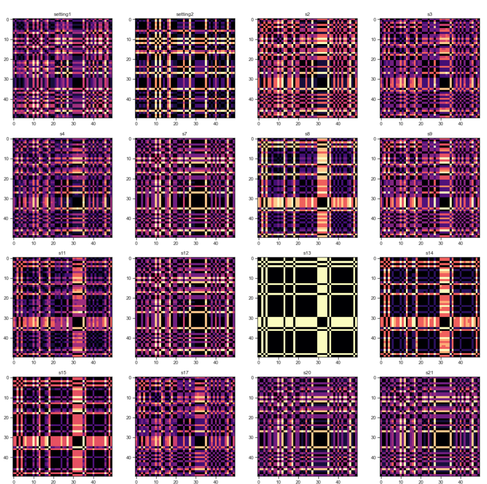


Figure 7: RPs contain rich information on spatio-temporal dimensions.

3.5 Modeling and training

For implementing the evaluated models in this research we will use TensorFlow & Keras as the framework to build and train the architectures .

For the LSTM model will use two stacked "LSTM layers" which is a pre-build layer, in first layer receives an input shape given by the pre-defined window-size of 50, and the number of selected features of the dataset (16), both layers configured to return sequences parameters that forces the layer to return the hidden state output for each input time step. Each layer is followed by a dropout layer of 20% to set the input units to 0 at the given rate, this is a regularization technique for helping reduce over-fitting. Final layer is a dense layer with activation function "softmax".

The CNN architecture uses two stacked "Conv2D" layer with a relu activation with a filter of 3x3 followed by a "MaxPooling2D" layer to choose the best features. A fully connected dense layer is then used to extract the relevant data and finally another dense layer with activation "softwax" is used to extract the output probabilities.

The TCN model architecture is created using Sequential model with a TCN layer that is part of the keras-tcn API published by Remy (2020) followed by a Dense Layer with "softmax" activation function. The model is compiled using "Adam" optimizer and "accuracy" as metric.

3.6 Training and Testing

All three models are executed through out a custom function "run_model" that trains and test the provided model with the provided data and labels, and calculates the evaluation metrics and plots. The training step of the function is configured with an early stopping function.

3.6.1 Early Stopping

All three models that are fed into the run_model function are configured with a callback function to stop training at the point when performance on a validation dataset has stopped improving.

4 Design Specification

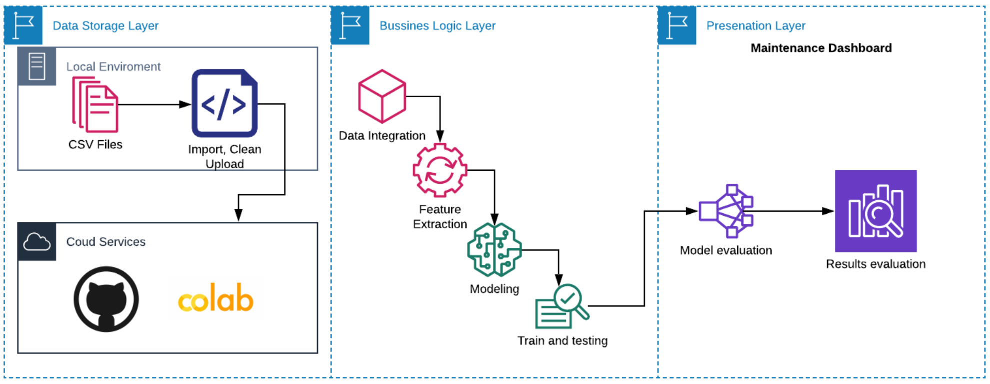


Figure 8: Tree-Tier tier design architecture, includes storage, business logic and presentation layers.

The proposed design considers using Github as version control of the source code and required files, a local version of the repository is used for local changes, and the early stages of the methodology, but google colab will be used for the training and testing stages because of the computing resources like GPU, Additionally the use of Google Colab will make it easier to run the notebook and replicate the experiments. Therefore, The process will be to produce local Jupyter notebooks, push any changes and open the notebooks in **Google Colab** for training and evaluation of results, the final notebook includes functionality to automatically detect the environment and set up the required variables and install any package dependencies.

5 Implementation

5.1 Resources and Equipment

The following tools and components were used for the implementation of this research, "Table. 2" describes required resources, software and services used during the project.

Table 2: Resources, Software and Services

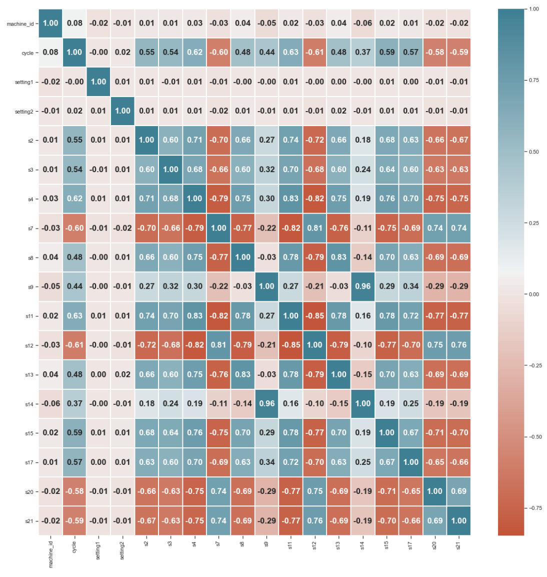
Category	Item	Description
Computing	RAM	8GB (16GB are recommended)
	Processor	64-bit multi-core processor (Intel i5 or superior)
	Storage	250+ GB of available space in hard disk
	Operating System	Ubuntu, macOS or Windows
Software	Python	Main Programming language.
	Tensorflow & Keras	Library to develop and train models.
	Anaconda	Distribution of to simplify package management.
	Jupyter Notebook	ML and data processing and modeling.
	VS code	Programming IDE.
Cloud Services	Google Colab	Run Notebooks for Neural Networks.
	Github	Code repository and version control.

5.2 Enviroment Setup

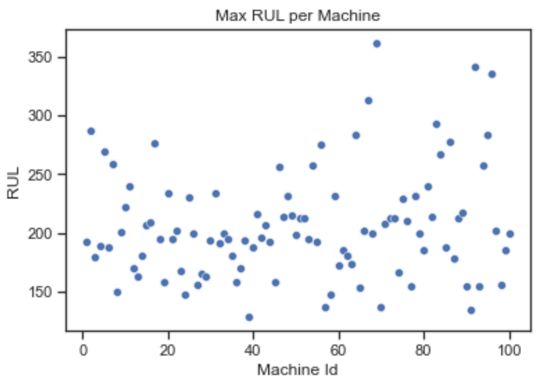
Automatic detection of the running environment is the first step of the notebook, this to make sure to configure the right variables and packages when running locally or in Google Colab.

5.3 Exploratory Analysis

We use the correlation heat-map to show pairwise correlations among the variables "Fig. 9a" it can be observed high positive and negative correlation between the sensor features. In "Fig. 9b" we can review the maximum number of cycles (RUL) per machine, the minimum value is 128 and the maximum is 362 with an average of 206.31.



(a) Correlation heatmap of features in FD001.



(b) Max RUL per machine in FD001.

5.4 Data Cleaning

After initial inspection no null or NaN values detected, but some sensor channels have constant values with no predictor value, remove the following columns: "setting3","s1", "s5", "s6", "s10", "s16","s18","s19"

5.5 Data splitting for training, validation and testing

The C-MAPSS dataset is already splitted into training and testing subsets, each sub files contains the following number of samples:

Table 3: The C-MAPSS dataset training and testing samples

Sub-Set	**FD001**	**FD004**
Training Samples	17,731	57,522
Testing Samples	100	248

Additional to the training and test dataset, a **validation split of 10%**, this parameter is used to hold back the corresponding number of samples of the training dataset from the training of the model and is then used to give an unbiased estimate of the skill of the final tuned model.

5.6 Data Wrangling

5.6.1 Sequence Generation

Using the sliding window technique to generate a sequence on both training and testing datasets by obtaining the last 30 cycles for each machine_id with the selected features, the resulting variable is a three-dimension NumPy array with 15,631 training samples, of 30 cycles and 16 features each, the testing variable contains 8,162 samples.

5.6.2 Feature Normalization

Firstly, the selected features are normalized using MinMaxScaler that scales and translates each feature individually into a range between zero and one. As stated by Li et al. (2018), no additional signal/feature processing such as kurtosis or skewness is required.

Then the normalized training and test datasets are transformed using the sliding window technique to create a NumPy array *X_train* and *X_test* variables that will be fed into LSTM and TCN, additionally, the variable *X_train_img* and *X_test_img* with recurrence plot images is generated for CNN.

5.7 Training and Evaluating

To run the experiments a custom function is implemented to train and test the models, this function takes the train and test data and labels, fits them into the model, the model is trained and the metrics and plots 5.8 are automatically generated and stored into results variable for later comparison of the models.

5.8 Methods of Evaluation

The criteria we will use as a metric for evaluating the performance of the models will be the following:

- **Confusion Matrix:** Plot describes how many times the target variable groups were properly identified by the model.
- **ROC Curve:** Plot showing the performance of a classification model.
- **AUC:** Measures the entire underneath the ROC curve.
- **Precision:** Ratio between the True Positives and all the Positives.
- **Recall*:** Ratio of total positive cases correctly classified.
- **Accuracy:** Number of correct predictions over all the predictions.
- **F1-Score*:** Is a metric which combines Precision and Recall.

*Because of the imbalanced class distribution of the dataset's RUL classification we will put higher weight on metrics that measure the negative/positive observations like the Precision and F1-Score metrics when comparing the model's performance.

6 Evaluation

Once the corresponding data has been formated and transformed to be feed into the neural networks, the following experiments are performed to evaluate and compare the performance of the three models with different settings and data.

In each one of the experiments the results are shown in a table format containing the model applied, the dataset used and the resulting metric in a percentage format including Accuracy, ROC_AUC, Precision, Recall and F1 score.

6.1 Experiment / Case Study 1

The baseline experiment consists on training and testing the three architectures LSTM, CNN and TCN with default values. All three models are processed by a custom function "run_model" that fits the training features and labels in the model, performs the prediction of the testing dataset and generates the discussed performance metrics 5.8 Three models are trained and fit using a default number of epochs of 25 with an early stopping callback function to stop the training once the model's performance has stopped improving. Using a validation split or 90:10 and default batch size of 200. The results can be seen in "Table. 4" and the evaluation plots on "Fig. 4" show an overall good performance of the TCN model, with higher accuracy and F1 score in the FD001 dataset.

Table 4: Experiment 1 Results (Default parameters)

Model	Dataset	Accuracy	ROC AUC	Precision	Recall	F1 Score
LSTM	FD001	96.36%	75.16%	97.95%	75.16%	82.48%
CNN	FD001	96.25%	89.86%	84.77%	89.86%	87.10%
TCN	FD001	**97.81%**	87.46%	95.81%	87.46%	**91.14%**
LSTM	FD004	96.87%	87.91%	82.65%	87.91%	85.05%
CNN	FD004	94.08%	50.00%	47.45%	50.00%	48.69%
TCN	FD004	96.39%	80.30%	81.63%	80.30%	80.95%

6.2 Experiment / Case Study 2

Because of the imbalanced class distribution detected on the data transformation stage 3.4 the second experiment consists of using the class_weight parameter in the fit process of the model to evaluate if that helps to improve the prediction rate of the unbalanced class.

The results can be seen in "Table. 5" and show that in all 3 models the use of the weight parameter did improve the performance of the models, it is also worth noticing that TCN outperformed again the other models in F1 score in both datasets.

6.3 Experiment / Case Study 3

Another way to improve the performance of the classification models for imbalanced data is to train the model using a larger than default batch size to ensure that each batch has a higher chance of containing positive samples. Experiment 3 consist on training the three models using a larger batch size, the results shown on "Table. 6", were we can see that the performance only showed a marginal improvement in the precision, as in experiment 2 TCN outperformed the other models.

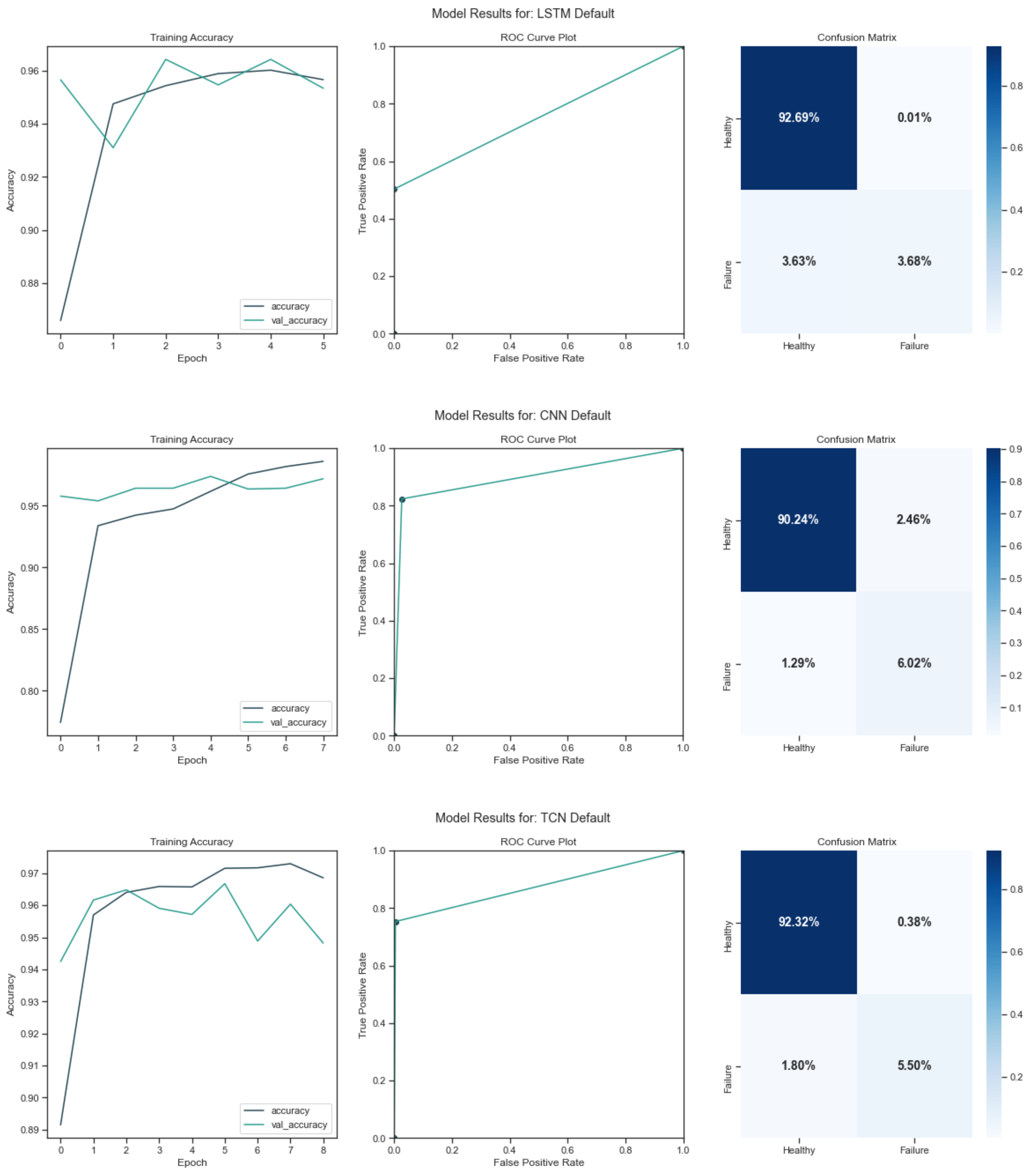


Figure 10: Experiment 1 on FD001 Performance Plots

Table 5: Experiment 2 Results (Using Class Weights)

Model	**Dataset**	**Accuracy**	**ROC AUC**	**Precision**	**Recall**	**F1 Score**
LSTM	FD001	95.56%	90.82%	82.00%	90.82%	85.74%
CNN	FD001	96.25%	89.47%	84.89%	89.47%	87.01%
TCN	FD001	**97.85%**	88.71%	**94.82%**	88.71%	**91.50%**
LSTM	FD004	96.31%	73.09%	83.92%	73.09%	77.34%
CNN	FD004	94.90%	50.00%	47.54%	50.00%	48.69%
TCN	FD004	95.69%	**84.83**%	77.20%	**84.83**%	**80.46**%

Table 6: Experiment 3 Results (Larger Batch Size)

Model	**Dataset**	**Accuracy**	**ROC AUC**	**Precision**	**Recall**	**F1 Score**
LSTM	FD001	96.86%	79.52%	96.66%	79.52%	85.86%
CNN	FD001	96.56%	89.49%	86.37%	89.49%	87.85%
TCN	FD001	**97.67%**	**86.99%**	**95.06%**	**86.99%**	**90.56%**
LSTM	FD004	95.76%	66.17%	81.68%	66.17%	71.10%
CNN	FD004	94.90%	50.00%	47.45%	50.00%	48.69%
TCN	FD004	95.75%	66.24%	81.59%	66.24%	77.11%

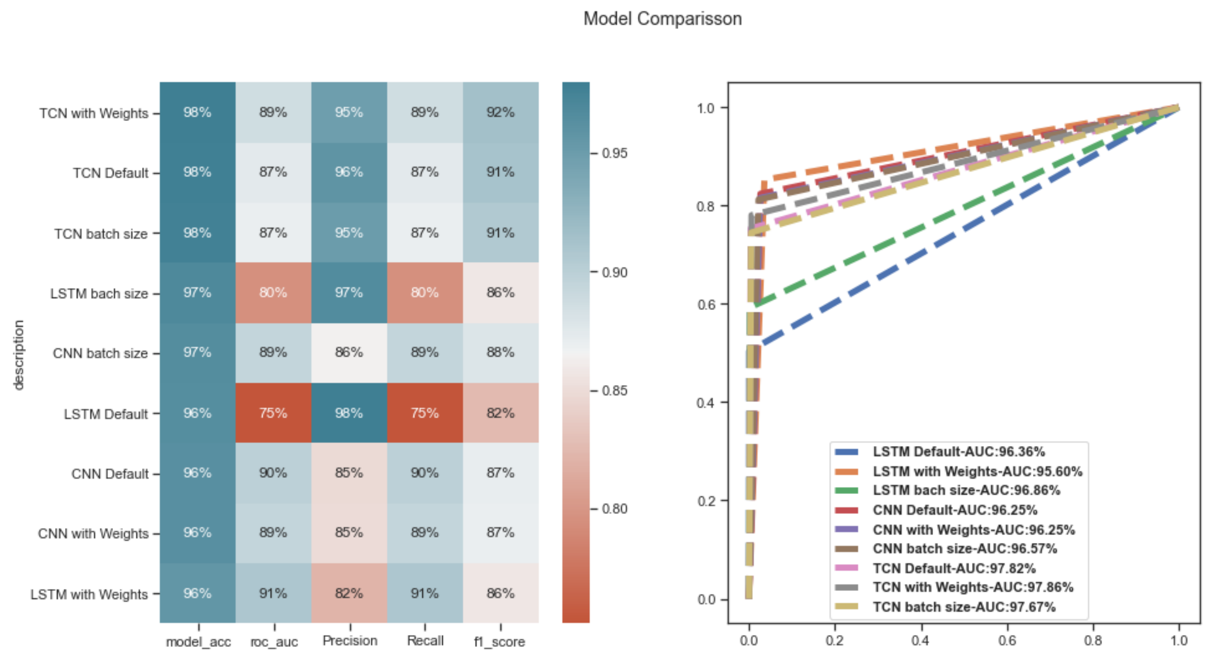
6.4 Discussion

After performing the 3 experiments on the datasets FD001 we use the comparison plots shown on “Fig. 11”, it is clear that **TCN outperformed LSTM and CNN**, especially considering the F1 score. It can also be noticed that the use of class weights was beneficial for increasing the recall performance of the model, this is especially important for imbalanced label distributions.

On the other hand, when comparing the results of the experiments using FD004 “Fig. 12”, which is a more complex data scenario with multiple conditions we can observe that performance was reduced in almost all the models, but it was especially noticeable for the CNN models in which almost all the performance metrics drop down drastically, however, TCN again showed a more consistent and improved performance among the three experiments, despite not being the best model just below LSTM.

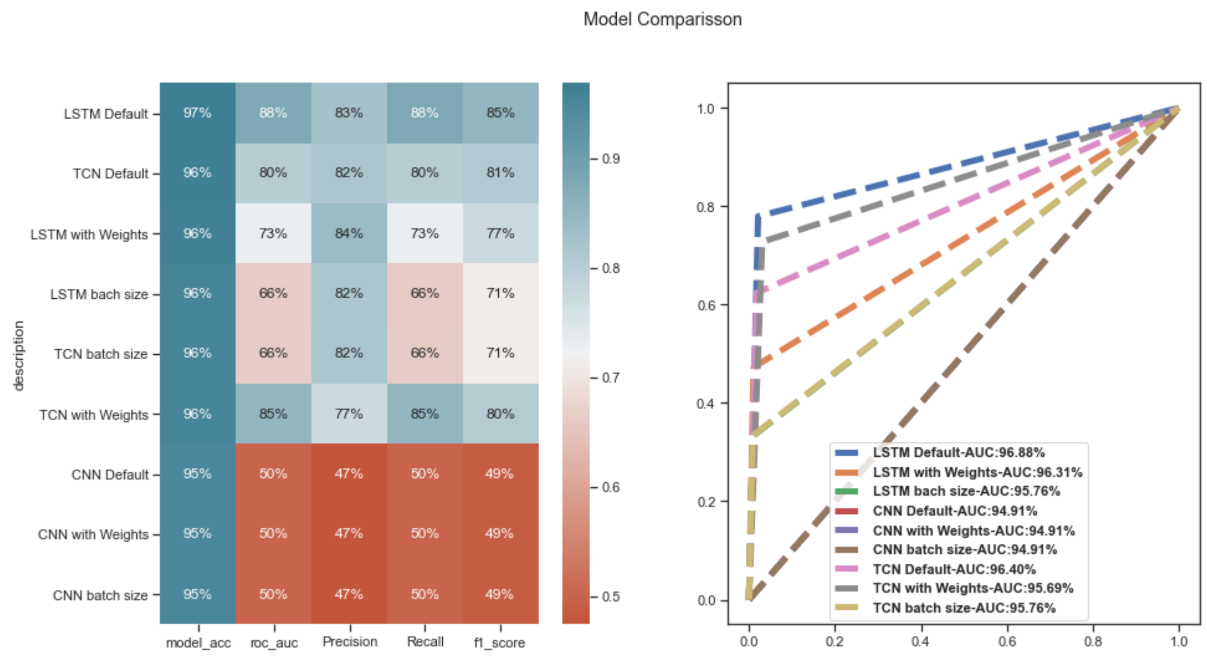
In the 3 experiments, TCN consistently outperformed the other compared models more notoriously under the less complex data in FD001, especially considering accuracy and F1 score, which was an important factor given the unbalanced state of the resulting classes, however, when given a more complex set of data FD004 TCN outperformed only the CNN model that shows overall very poor results, but LSTM did showed better results, this could imply that TCN is the best model only for certain operating conditions and is only able to perform more accurately for single fault scenarios, although it did not outperformed TSLM on FD004 dataset, the results are still very competitive and considered satisfactory in terms of accurately predict failure.

In practice this results show that both TCN and LSTM can be confidently applied for predicting failure of a machine with a given range, since both models show consistent good performance in all experiments.



description	model_acc	roc_auc	Precision	Recall	f1_score
TCN with Weights	98%	89%	95%	89%	92%
TCN Default	98%	87%	96%	87%	91%
TCN batch size	98%	87%	95%	87%	91%
LSTM bach size	97%	80%	97%	80%	86%
CNN batch size	97%	89%	86%	89%	88%
LSTM Default	96%	75%	98%	75%	82%
CNN Default	96%	90%	85%	90%	87%
CNN with Weights	96%	89%	85%	89%	87%
LSTM with Weights	96%	91%	82%	91%	86%

Figure 11: Heatmap plot is used to compare the performance metrics of all the experiments on FD001



description	model_acc	roc_auc	Precision	Recall	f1_score
LSTM Default	97%	88%	83%	88%	85%
TCN Default	96%	80%	82%	80%	81%
LSTM with Weights	96%	73%	84%	73%	77%
LSTM bach size	96%	66%	82%	66%	71%
TCN batch size	96%	66%	82%	66%	71%
TCN with Weights	96%	85%	77%	85%	80%
CNN Default	95%	50%	47%	50%	49%
CNN with Weights	95%	50%	47%	50%	49%
CNN batch size	95%	50%	47%	50%	49%

Figure 12: Heatmap plot is used to compare the performance metrics of all the experiments on FD004

7 Conclusion and Future Work

The aim of this research was to compare the state of the art deep neural networks architectures for the classification of Remaining Useful Life. This research proposes the usage of TCN as a better model for this task because of their dilated convolutions to capture long-range temporal patterns. The results demonstrate that TCN outperforms in almost all cases the other two models reviewed LSTM and CNN in the single fault scenarios, however, for the more complex data that includes multiple operating and fault conditions LSTM did outperformed TCN. Additionally, because of the imbalanced distribution of the resulting classification labels, additional experiments were performed, demonstrating that the usage of class weighs when training the model helps to increase the precision and F1 score of the model.

7.1 Future Work

The implemented machine learning algorithms implemented in this research LSTM, CNN and TCN all show excellent performance when compared with traditional prognosis algorithms for RUL prediction. All the techniques can work without any prior knowledge, allowing researchers to focus on the selection and tuning of models instead of feature engineering. However, these models heavily depend on supervised learning, meaning that they require large amounts of labelled data for training purposes. Thus, their prediction accuracy relies heavily on the quality of the data and their measurements. The usage of unsupervised learning models could help to extract high-level features from raw data automatically.

References

A. Saxena , K. Goebel (2008). Turbofan engine degradation simulation data set, `http://ti.arc.nasa.gov/project/prognostic-data-repository`. Accessed: 2018-12-06.

Babu, G. S., Zhao, P. and Li, X. (2016). Deep convolutional neural network based regression approach for estimation of remaining useful life, *DASFAA*.
URL: *https://link.springer.com/chapter/10.1007/978-3-319-32025-0*$_1$4

Barlow, M. (2015). *Predictive Maintenance*, : O'Reilly Media, Inc.
URL: *https://learning.oreilly.com/library/view/predictive-maintenance/9781492048183/*

Chang, Q., Liu, J., Xiao, G. and Biller, S. (2012). The costs of downtime incidents in serial multi-stage manufacturing systems, *Journal of Manufacturing Science and Engineering* **134**: 021016.

Chehade, A., Bonk, S. and Liu, K. (2017). Sensory-based failure threshold estimation for remaining useful life prediction, *IEEE Transactions on Reliability* **66**(3): 939–949.

Frank, R. J., Davey, N. and Hunt, S. P. (2001). Time series prediction and neural networks, *Journal of Intelligent and Robotic Systems* **31**(1): 91–103.
URL: *https://doi.org/10.1023/A:1012074215150*

Hsueh, Y., Ittangihala, V. R., Wu, W.-B., Chang, H.-C. and Kuo, C.-C. (2019). Condition monitor system for rotation machine by cnn with recurrence plot, *Energies* **12**(17).
URL: *https://www.mdpi.com/1996-1073/12/17/3221*

Koh, B. H. D., Lim, C. L. P., Rahimi, H., Woo, W. L. and Gao, B. (2021). Deep temporal convolution network for time series classification, *Sensors* **21**(2).
URL: *https://www.mdpi.com/1424-8220/21/2/603*

Krishna, M. and Baghaei, K. T. (2019). Recent approaches in prognostics: State of the art.

Lea, C., Flynn, M. D., Vidal, R., Reiter, A. and Hager, G. D. (2017). Temporal convolutional networks for action segmentation and detection, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Li, J., Li, X. and He, D. (2019). A directed acyclic graph network combined with cnn and lstm for remaining useful life prediction, *IEEE Access* **7**: 75464–75475.

Li, X., Ding, Q. and Sun, J.-Q. (2018). Remaining useful life estimation in prognostics using deep convolution neural networks, *Reliability Engineering & System Safety* **172**: 1–11.
URL: *https://www.sciencedirect.com/science/article/pii/S0951832017307779*

N, M., Romano, M. C., M, T. and J, K. (n.d.). Recurrence plots for the analysis of complex systems.
URL: *www.recurrence-plot.tk*

Remy, P. (2020). Temporal convolutional networks for keras, `https://github.com/philipperemy/keras-tcn`.

Saxena, A., Goebel, K., Simon, D. and Eklund, N. (2008). Damage propagation modeling for aircraft engine run-to-failure simulation, *2008 International Conference on Prognostics and Health Management*, pp. 1–9.

Thiel, M., Romano, M. C. and Kurths, J. (2004). How much information is contained in a recurrence plot?, *Physics Letters A* **330**(5): 343–349.
URL: *https://www.sciencedirect.com/science/article/pii/S0375960104009922*

Wan, R., Mei, S., Wang, J., Liu, M. and Yang, F. (2019). Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting, *Electronics* **8**(8).
URL: *https://www.mdpi.com/2079-9292/8/8/876*

Wang, Q., Zheng, S., Farahat, A., Serita, S. and Gupta, C. (2019). Remaining useful life estimation using functional data analysis, *2019 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 1–8.

Zheng, S., Ristovski, K., Farahat, A. and Gupta, C. (2017). Long short-term memory network for remaining useful life estimation, *2017 IEEE International Conference on Prognostics and Health Management (ICPHM)*, pp. 88–95.