Online Prognostics Strategies with Deep Learning Models for Fleets of Engineering Assets

Carel Johannes Louw

Submitted in Partial Fulfillment of the Requirements for the Degree
Master of Engineering (Mechanical Engineering)

Centre for Asset Integrity Management (C-AIM)
Department of Mechanical and Aeronautical Engineering
University of Pretoria

November 2018

Abstract

Dissertation Title: Online Prognostics Strategies with Deep Learning Models for Fleets of Engineering Assets

Author: CJ Louw

Student Number: 12032876

Supervisor: Prof PS Heyns

University: University of Pretoria

Department: Department of Mechanical and Aeronautical Engineering

Degree: Master of Engineering (Mechanical Engineering)

Date: November 2018

The accurate prediction of remaining useful life for fleets of engineering assets is an increasingly important task in prognostics and health management (PHM). This is because the accurate prediction of time to failure allows for improved planning, scheduling and decision-making of maintenance tasks for fleets of engineering assets. Accurate prediction of remaining useful life therefore has high potential to increase the reliability, availability, production output, profitability and safety, and to decrease downtime, unnecessary maintenance and operating costs for fleets of engineering assets.

This work proposes general and convenient prognostics strategies with data-driven deep learning models for online remaining useful life prediction for fleets of engineering assets, where historical run-to-failure condition monitoring measurements with trendable exponential degradation trajectories are available.

The modeling of long-term sequence information in condition monitoring measurements has previously been shown to be very challenging and crucial for effective data-driven prognostics. Long short-term memory (LSTM) and gated recurrent unit (GRU) recurrent neural network (RNN) deep learning models are currently the state-of-the-art sequence modeling techniques and can effectively model long-term sequence information. These gated recurrent neural networks have however to date not been comprehensively investigated and compared for data-driven prognostics for fleets of engineering assets.

In this work we investigate data sets which include a general asset degradation data set, turbofan engine degradation data set and turbofan engine degradation benchmarking data sets. The investigated data sets all simulate the exponential degradation trajectories and condition monitoring measurements for fleets of engineering assets that were run to failure, where each

asset had either univariate or multivariate condition monitoring sensor measurements performed at fixed time intervals over its lifetime.

The data sets investigated include training set examples and testing set examples. The training set examples represent historical (previously seen) engineering assets with condition monitoring measurements that were run to failure. The testing set examples represent future (completely unseen) general engineering assets with condition monitoring sensor measurements that were run to failure. The objective and challenge is therefore to propose a prognostics strategy and train a model on the condition monitoring measurements of the training set examples offline. The proposed prognostics strategy and trained model must then predict the remaining useful life from the condition monitoring measurements of the testing set examples fully online. The turbofan engine degradation benchmarking data sets allow for simple and effective prognostics performance comparisons between publications with different prognostics strategies and models.

The proposed general prognostics strategies for this work include a prognostics classification strategy and prognostics regression strategy.

The proposed prognostics classification strategy is to structure the remaining useful life modeling problem as a sequence-to-sequence classification deep learning problem, where the input sequence is the univariate or multivariate condition monitoring measurement time series and the target sequence is the remaining useful life classification time series. The remaining useful life classification time series for each individual training and testing set example consists of remaining useful life classes with different degradation levels that are based on its linearly decreasing remaining useful life time series with an applied threshold.

The prognostics regression strategy is to structure the remaining useful life modeling problem as a sequence-to-sequence regression deep learning problem, where the input sequence is the univariate or multivariate condition monitoring measurement time series and the target sequence is the remaining useful life regression time series. The remaining useful life regression time series for each individual training and testing set example consists of remaining useful life values that are based on its linearly decreasing remaining useful life time series with an applied threshold.

The sequence-to-sequence classification and regression deep learning models then learns and generalizes the mapping between the condition monitoring measurement time series and the remaining useful life classification and regression time series for all the training set examples offline. The trained sequence-to-sequence classification and regression deep learning models is then used to predict the remaining useful life classification and regression time series from the condition monitoring measurement time series for all the testing set examples fully online.

The proposed sequence-to-sequence deep learning classification and regression model architectures that are investigated and compared for the proposed prognostics classification and regression strategies include a feedforward neural network (FNN), simple recurrent neural

network (S-RNN), long short-term memory recurrent neural network (LSTM-RNN) and gated recurrent unit recurrent neural network (GRU-RNN).

The sequence-to-sequence deep learning classification and regression model architectures are trained on the training sets of the investigated data sets with the new and effective Adam algorithm. The deep learning models are also regularized with a combination of the early stopping, weight decay and dropout regularization techniques in order to reduce overfitting and improve generalization and prediction performance.

The prognostics classification and regression strategies were successfully applied on the investigated data sets with the FNN, S-RNN, LSTM-RNN and GRU-RNN classification and regression model architectures. The LSTM-RNN and GRU-RNN models drastically outperformed the FNN and S-RNN models as expected. The GRU-RNN models slightly outperformed the LSTM-RNN models and the S-RNN models significantly outperformed the FNN models on average.

The prognostics regression strategy and LSTM-RNN and GRU-RNN regression models achieved very competitive results when compared with other state-of-the-art publications on the turbofan engine degradation benchmarking data sets. The prognostics regression strategy and GRU-RNN regression model achieved a very competitive benchmarking score of 589 on the PHM08 turbofan degradation benchmarking data set.

Keywords: Prognostics and Health Management (PHM), Remaining Useful Life (RUL), Long Short-Term Memory (LSTM), Gated Recurrent Unit (GRU), Recurrent Neural Network (RNN), Deep Learning, Predictive Maintenance, Condition-Based Maintenance

Acknowledgements

The author would like to thank his supervisor Prof PS Heyns for his valuable advice, insight and time in completing this research. The author would like to acknowledge Dr A Saxena and Dr K Goebel for making available the turbofan engine degradation data sets that are investigated extensively in this research.

The author would also like to acknowledge the Eskom Power Plant Engineering Institute (EPPEI) for funding this research.

Abbreviations

C	Class
FNN	Feedforward Neural Network
GRU	Gated Recurrent Unit
GRU-RNN	Gated Recurrent Unit Recurrent Neural Network
LSTM	Long Short-Term Memory
LSTM-RNN	Long Short-Term Memory Recurrent Neural Network
PHM	Prognostics and Health Management
RUL	Remaining Useful Life
S	Sensor
S-RNN	Simple Recurrent Neural Network

Nomenclature

Roman Letters

a, b, c, d	Stochastic Exponential Degradation Model Parameters
AT	Applied Threshold
B_n	Bounds
C	Dimension of the Remaining Useful Life Class Target Vector
CE	Cross-Entropy
D	Dimension of Condition Monitoring Measurement Input Vector
f	Model Architecture
g	Gradient
H	Number of Selected Hidden Units
I	Length of the Output Vector from the Previous Layer
$\mathcal{L}$	Loss Function
$\mathcal{L}_{CE}$	Cross-Entropy Loss Function
$\mathcal{L}_{MSE}$	Mean Squared Error Loss Function
M	Variable Number of Time Steps for a Training or Testing Set Example
MAE	Mean Absolute Error
MSE	Mean Squared Error
N	Total Number of Training or Testing Examples
$\mathcal{N}$	Normal Distribution
P	Total Number of Time Steps for all the Training or Testing Set Examples Combined
r	Biased Second Moment Estimate
$\hat{r}$	Bias Corrected Second Moment Estimate
$\overline{RMSE}$	Benchmarking Root Mean Squared Error
s	Biased First Moment Estimate
$\hat{s}$	Bias Corrected First Moment Estimate
$\bar{S}$	Benchmarking Score
t	Time Step

TOF	Time of Failure
x	Training Set Input Example
$\hat{x}$	Testing Set Input Example
X	Training Set Input Examples
$\hat{X}$	Testing Set Input Examples
y_{CN}	Training Set Classification Target Example
y_{RN}	Training Set Regression Target Example
$\hat{y}_{CN}$	Testing Set Classification Target Example
$\hat{y}_{RN}$	Testing Set Regression Target Example
Y_{CN}	Training Set Classification Target Examples
Y_{RN}	Training Set Regression Target Examples
$\hat{Y}_{CN}$	Testing Set Classification Target Examples
$\hat{Y}_{RN}$	Testing Set Regression Target Examples

Greek Letters

δ	Numerical Stability Constant
$\Delta\theta$	Model Parameter Update
ϵ	Global Learning Rate
θ	Model Parameter
λ	Weight Decay Parameter
μ	Normal Distribution Mean
ρ_2	Exponential Decay Rate for the Second Moment Estimate
ρ_1	Exponential Decay Rate for the First Moment Estimate
σ	Normal Distribution Standard Deviation
τ	Dropout Fraction

Superscripts

C	Dimension of the Remaining Useful Life Class Target Vector
D	Dimension of Condition Monitoring Measurement Input Vector
f	Forget Gate
H	Number of Selected Hidden Units
(i)	Example
I	Length of the Output Vector from the Previous Layer
$[n]$	Current Layer
$[n-1]$	Previous Layer
(n)	Current Iteration
$(n+1)$	Next Iteration
o	Output Gate
(t)	Current Time Step
$(t-1)$	Previous Time Step
$(t+1)$	Next Time Step
r	Reset Gate
u	Update Gate

Subscripts

CE	Cross-Entropy
CN	Classifcation
MSE	Mean Squared Error
RN	Regression

Activation Functions

E	Softmax Activation Function
L	Linear Activation Function
S	Sigmoid Activation Function
T	Hyperbolic Tangent Activation Function

Model Parameters

b_i	Bias Vector
$R_{i,j}$	Recurrent Weight Matrix
$W_{i,j}$	Weight Matrix

Model Vectors

c_i	Cell State Vector
CM_i	Condition Monitoring Measurement Input Vector
f_i	Forget Gate Vector
h_i	Output Vector
o_i	Output Gate Vector
r_i	Reset Gate Vector
$RUL_{i,CN}$	Remaining Useful Life Class Target Vector
$RUL_{i,RN}$	Remaining Useful Life Value Target Vector
s_i	Candidate Cell State Vector
u_i	Update Gate Vector
z_i	Any Vector

Time Series

$CM(t)$	Univariate or Multivariate Condition Monitoring Measurement Time Series
$DT(t)$	Degradation Trajectory Time Series
$HI(t)$	Health Index Time Series
$RUL(t)$	Remaining Useful Life Time Series
$RUL_{CN}(t)$	Remaining Useful Life Classification Time Series
$RUL_{RN}(t)$	Remaining Useful Life Regression Time Series
$S1(t)$	Generated Univariate Condition Monitoring Sensor Measurement Time Series
t	Time [Cycles]

Contents

1 Introduction and Background

1.1 Background

The accurate prediction of remaining useful life for fleets of engineering assets is an increasingly important task in prognostics and health management (PHM). This is because the accurate prediction of time to failure allows for improved planning, scheduling and decision-making of maintenance tasks for fleets of engineering assets. Accurate prediction of remaining useful life therefore has high potential to increase the reliability, availability, production output, profitability and safety, and to decrease downtime, unnecessary maintenance and operating costs for fleets of engineering assets (Jardine, et al., 2006). These fleets of engineering assets can include military, mining, aeronautical, medical, automotive, manufacturing, aerospace and power plant machines and equipment, that degrade at a large scale and over long periods of time with condition monitoring measurements.

The remaining useful life predictions for a degrading general engineering asset are generally based on univariate or multivariate condition monitoring sensor measurements and a prognostics model (Lei, et al., 2018). The univariate or multivariate condition monitoring sensor measurements can include temperature, vibration, pressure, noise, stress, strain, efficiency or any other measurements that are accurate, convenient, safe, inexpensive and reliable (Lei, et al., 2018). The condition monitoring sensor measurements must however be trendable and representative of the true health of the degrading engineering asset for effective and accurate remaining useful life predictions. It is very important to point out that the condition monitoring sensor measurements in this work refers to the already extracted features (health indicators) from raw sensor measurements.

The prognostics model can generally be a physical model, data-driven model or hybrid model (Mishra, et al., 2014). Physical models generally require physical understanding of the degradation trajectories of the engineering assets and can be very useful when not enough run-to-failure data is available for data-driven models. The challenge with physical models however is that it can be very difficult, time consuming and expensive to understand and model the complex physics present in the degradation trajectories of engineering assets. Data-driven models generally require large amounts of run-to-failure data in order to model the degradation trajectories of the engineering assets, but can be very useful when the complex physics present in the degradation trajectories are too difficult, time consuming and expensive to understand. The challenge with data-driven models however is that the required run-to-failure data is not always available. Hybrid models combine physical models and data-driven models in order to overcome these challenges, but can be very challenging and expensive to implement properly (Mishra, et al., 2014).

There has recently been increased interest in applying pattern recognition and machine learning techniques (artificial intelligence) to numerous industries that require human intelligence and effort for the automation of complex and repetitive tasks at a large scale and over long periods of time. The main motivation for the increased interest in artificial intelligence is to reduce the cost and time required for complex and repetitive tasks that would traditionally be performed by humans.

The use of anomaly detection, diagnostics and prognostics applications in condition-based maintenance for fleets of engineering assets at a large scale and over long periods of time has also become more popular recently. Here the main motivation is increasing the reliability and reducing the operating costs for fleets of engineering assets. This increase in condition monitoring for fleets of engineering assets generally results in very large data sets that include trendable run-to-failure data, which can therefore motivate and justify data-driven prognostics models with pattern recognition and machine learning techniques.

The currently most popular pattern recognition and machine learning approach is known as deep learning (Goodfellow, et al., 2016). Deep learning models are neural networks that consist of multiple stacked layers with trainable parameters and nonlinear activation functions. This gives deep learning models the capacity and ability to learn very complex and nonlinear input to output mappings for very large data sets. The most popular tasks for supervised deep learning models are classification and regression. Deep learning classification models learn the mapping between inputs and associated output classes, where deep learning regression models learn the mapping between inputs and associated output values. The most popular deep learning model architectures include feedforward neural networks for general modeling, convolution neural networks for image modeling and recurrent neural networks sequence modeling (Chollet, 2018).

Deep learning models have recently achieved state-of-the-art performance on numerous complex modeling tasks including image classification, speech recognition, handwriting transcription, machine translation, autonomous driving, text-to-speech conversion, natural language processing, superhuman gaming, etc. Deep learning models also scale exceptionally well with very large data sets when compared with traditional pattern recognition and machine learning techniques (Chollet, 2018). Recurrent neural networks are however the most applicable and powerful deep learning models for data-driven prognostics, because condition monitoring measurements are generally continuous time series that contain important long-term sequence information.

Long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) and gated recurrent unit (GRU) (Cho, et al., 2014) recurrent neural networks are currently the state-of-the-art sequence modeling techniques and can effectively model important long-term sequence information (Goodfellow, et al., 2016). These gated recurrent neural networks have however to date not been comprehensively investigated for data-driven prognostics for fleets of engineering assets.

The recent increase in popularity of deep learning models is mainly due to algorithmic and software advances, increased availability of very large data sets, model benchmarking and increased computational resources in the form of graphics cards (Chollet, 2018).

1.2 Problem Statement

The problem statement for this work is to propose general and convenient prognostics strategies with data-driven deep learning models for online remaining useful life prediction for fleets of engineering assets, where historical run-to-failure condition monitoring measurements with trendable exponential degradation trajectories are available.

The proposed prognostics strategies with deep learning models have to automatically learn (model) the complex nonlinear mapping between the historical run-to-failure condition monitoring sensor measurements and the remaining useful life for the fleet of historical engineering assets. The trained deep learning models then have to predict the remaining useful life for the fleet of future engineering assets from their future condition monitoring sensor measurements fully online. This requires the deep learning models to learn (model) important long-term sequence information in the historical run-to-failure condition monitoring sensor measurement time series.

The implemented prognostics strategies and deep learning models have to be fully online for remaining useful life prediction at a large scale and over long periods of time for the fleet of engineering assets. This means that the prognostics strategies and trained deep learning models have to automatically predict the remaining useful life for the fleet of engineering assets live from their condition monitoring sensor measurements at each time step throughout their lifetime, without any additional online human intelligence and effort. The prognostics strategies with deep learning models therefore also have to automatically ignore healthy and light degradation condition monitoring sensor measurements for which the accurate remaining useful life is very difficult (impossible) and not important to model and predict. The fully online predicted remaining useful life could then be used for automated planning, scheduling and decision-making of maintenance tasks for the fleet of engineering assets if required.

The prognostics strategies with deep learning models have to be compatible with univariate or multivariate condition monitoring sensor measurements. This means that the prognostics strategies with deep learning models have to automatically weight the importance (trendability) of the different condition monitoring sensor measurements with respect to the remaining useful life for the fleet of engineering assets. The prognostics strategies with deep learning models also have to automatically ignore irrelevant condition monitoring sensor measurements with respect to the remaining useful life for the fleet of engineering assets.

The prognostics strategies with deep learning models have to be robust to noise in the condition monitoring sensor measurements of the engineering assets. This means that the condition monitoring sensor measurements did not require filtering and minimal preprocessing for remaining useful life modeling and prediction with the prognostics strategies with deep learning models.

The prognostics strategies with deep learning models have to be robust to different operating conditions and fault modes in the condition monitoring sensor measurements of the engineering assets. This means that the same trained deep learning model have to automatically predict and model the remaining useful life of the engineering assets with different operating conditions and fault modes, without having to train different models for different operating conditions and different fault modes.

The prognostics strategies with deep learning models have to be robust to variations in operating conditions, fault modes, process noise, manufacturing and assembly for the fleet of engineering assets. This requires the prognostics strategies with deep learning models to generalize very well between historical (previously seen) and future (completely unseen) engineering assets, when modeling and predicting their remaining useful life form their condition monitoring sensor measurements.

The prognostics strategies with deep learning models have to be general and applicable to future fleets of engineering assets with trendable exponential degradation and available historical run-to-failure condition monitoring measurements. The prognostics strategies with deep learning models did therefore not have to be specialized for a specific engineering problem or asset.

The prognostics strategies with deep learning models also have to be competitive with other publications on the turbofan engine degradation benchmarking data sets (Saxena & Goebel, 2008) for accurate remaining useful life prediction.

1.3 Literature Review

There are numerous comprehensive literature reviews on the broad field of prognostics. (Jardine, et al., 2006) provided a broad review on machinery diagnostics and prognostics implementing condition-based maintenance. (Lei, et al., 2018) more recently provided a broad systematic review on machinery prognostics from data acquisition to remaining useful life prediction. This literature review however specifically focuses on data-driven prognostics for fleets of engineering assets.

The turbofan engine degradation benchmarking data sets (Saxena & Goebel, 2008) are by far the most popular publically available data sets for data-driven prognostics research (Ramasso & Saxena, 2014). The turbofan degradation benchmarking data sets include five data sets namely the FD001, FD002, FD003, FD004 and PHM08 turbofan degradation benchmarking data set

respectively. The turbofan engine degradation benchmarking data sets each simulate the exponential degradation and condition monitoring measurements for a fleet of turbofan engines that were run to failure under different combinations of operating conditions and fault modes. Each individual turbofan engine example had 24 multivariate condition monitoring sensor measurements performed at fixed time intervals over its lifetime.

The turbofan engine degradation benchmarking data sets include training set examples and censored testing set examples. The training set examples represent historical (previously seen) turbofan engines that were run to failure with condition monitoring measurements. The censored testing set examples represent future (completely unseen) turbofan engines that were run to failure, but with condition monitoring sensor measurements that were censored for a random number of cycles before failure for each individual turbofan engine. With the idea of then predicting the censored number of cycles before failure (remaining useful life value) for each individual turbofan engine example in the testing set from its available (non-censored) condition monitoring measurements.

The objective and challenge for each turbofan engine degradation benchmarking data set is therefore to propose a prognostics strategy and train a model on the condition monitoring measurements of the training set examples. The prognostics strategy and trained model then have to predict the remaining useful life value for each individual turbofan engine example in the testing set from its available condition monitoring measurements.

The accuracy of the predicted remaining useful life values for each turbofan engine degradation benchmarking data set were then evaluated by calculating the benchmarking score and benchmarking root mean square error (Saxena, et al., 2008). This allowed for simple and effective prognostics performance comparisons between different publications with different prognostics strategies and models on the turbofan degradation benchmarking data sets.

(Ramasso & Saxena, 2014) performed an extensive literature review on previous prognostics strategies and models applied on the turbofan engine degradation benchmarking data sets.

An effective prognostics strategy for the turbofan engine degradation benchmarking data sets is to structure the remaining useful life modeling problem as a similarity-based matching problem. The condition monitoring measurements for the turbofan engines with known failure times are used to create a degradation trajectory library. The library can then be used for remaining useful life prediction of turbofan engines with similar condition monitoring measurements (Ramasso & Saxena, 2014).

(Wang, et al., 2008) and (Ramasso, 2014) both proposed similarity-based approaches that achieved very good results on the turbofan engine degradation benchmarking data sets. These similarity-based approaches were however outside the scope of this work, but are powerful approaches that can be investigated in future research.

An alternative prognostics strategy for the turbofan engine degradation benchmarking data sets is to structure the remaining useful life modeling problem as a sequence-to-sequence classification problem with different degradation classes for the turbofan engines as demonstrated by (Ramasso & Gouriveau, 2010) and (Ramasso & Gouriveau, 2014).

The most popular, straightforward and effective prognostics strategy however for the turbofan engine degradation benchmarking data sets is to structure the remaining useful life modeling problem as a sequence-to-sequence regression machine learning problem. The input sequence is the 24 multivariate condition monitoring measurement time series and the target sequence is a linearly decreasing remaining useful life time series with an applied threshold. The linearly decreasing remaining useful life time series with an applied threshold is also known as the knee (Heimes, 2008), kink (Lim, et al., 2014), elbow point (Rigamonti, et al., 2016) and piece-wise (Zheng, et al., 2017) approach.

The sequence-to-sequence regression machine learning model then learns and generalizes the mapping between the condition monitoring measurement time series and the linearly decreasing remaining useful life time series with an applied threshold, for all the training set examples offline. The trained sequence-to-sequence regression machine learning model is then be used to predict the linearly decreasing remaining useful life time series for the time steps below the applied threshold from the condition monitoring measurement time series, for all the testing set examples fully online. The big difference between publications on the turbofan engine degradation benchmarking data sets however was the type of regression machine learning model (technique) that was proposed, trained and applied.

(Lim, et al., 2014) proposed a switching Kalman filter neural network ensemble and compared it to an multi-layer perceptron feedforward neural networks and Kalman filter neural network ensemble. The switching Kalman filter neural network ensemble outperformed the multi-layer perceptron feedforward neural networks and Kalman filter neural network ensemble.

(Babu, et al., 2016) proposed a convolutional neural network and compared it to an multi-layer perceptron feedforward neural network, support vector regression and relevance vector regression. The convolutional neural network outperformed the multi-layer perceptron feedforward neural network, support vector regression and relevance vector regression on the turbofan engine degradation benchmarking data sets.

(Zhang, et al., 2017) proposed a multi-objective deep belief networks ensemble and compared it with eleven other popular regression machine learning models which include gradient boosting and random forests. The multi-objective deep belief networks ensemble outperformed the eleven other popular regression machine learning models.

(Li, et al., 2018) similarly proposed a deep convolutional neural network and compared it to an neural network, deep neural network, recurrent neural network and long short-term memory recurrent neural network. The deep convolutional neural network outperformed the neural network, deep neural network, recurrent neural network and long short-term memory recurrent neural network on the turbofan engine degradation benchmarking data sets (for this publication).

The problem with the applied regression models and strategies by (Babu, et al., 2016), (Zhang, et al., 2017) and (Li, et al., 2018) is that they all use a sliding window of condition monitoring measurements with a certain length as their input. The applied regression models can therefore only model the condition monitoring measurement in the relatively short sliding window and is therefore not able to model long-term sequence information. Another problem with the sliding window approach is that its length is another hyperparameter that needs to be tuned with trial and error (Li, et al., 2018). Another practical problem with the sliding window approach is that it requires waiting the length of the sliding window before a remaining useful life prediction can be made online. Recurrent neural networks can however model long-term sequence information, do not require a sliding window and generally outperform the above-mentioned approaches.

(Heimes, 2008) proposed a advanced recurrent neural network and compared it to an multi-layer perceptron feedforward neural network, where both models were trained with the extended Kalman filter algorithm. The advanced recurrent neural network drastically outperformed the multi-layer perceptron feedforward neural network. The advanced recurrent neural network was able to achieve second place in the IEEE 2008 PHM conference challenge problem. This is also known as the PHM08 turbofan degradation benchmarking data set (Saxena & Goebel, 2008).

(Rigamonti, et al., 2016) proposed a echo state network (recurrent neural network variant) that was trained with the differential evolution algorithm. The echo state network was found to be less computationally expensive than traditional recurrent neural networks. A strategy for determining the elbow point (applied threshold value) with a filtering approach was also proposed.

(Zheng, et al., 2017) and (Hsu & Jiang, 2018) both proposed long short-term memory recurrent neural networks that were trained with the RMSProp algorithm and both achieved very good results on the turbofan engine degradation benchmarking data sets.

It is therefore clear that the above-mentioned recurrent neural network regression models proposed by (Heimes, 2008), (Rigamonti, et al., 2016), (Zheng, et al., 2017) and (Hsu & Jiang, 2018) perform very well on the turbofan engine degradation benchmarking data sets. These recurrent neural networks could however be further improved upon with newer and better recurrent neural network architectures, training algorithms and regularization techniques that will be discussed and explained in chapter 3.

1.4 Research Scope

The research scope for this work is therefore to propose general and convenient prognostics strategies with data-driven deep learning models for online remaining useful life prediction for fleets of engineering assets, where historical run-to-failure condition monitoring measurements with trendable exponential degradation trajectories are available.

The modeling of long-term sequence information in condition monitoring measurements has previously been shown (Heimes, 2008) to be very challenging and crucial for effective data-driven prognostics. Long short-term memory (LSTM) (Hochreiter & Schmidhuber, 1997) and gated recurrent unit (GRU) (Cho, et al., 2014) recurrent neural network (RNN) deep learning models are currently the state-of-the-art sequence modeling techniques (Goodfellow, et al., 2016) (Chollet, 2018) and can effectively model long-term sequence information. These gated recurrent neural networks have however to date not been comprehensively investigated and compared for data-driven prognostics for fleets of engineering assets.

The data sets investigated in this work include a general asset degradation data set, turbofan engine degradation data set and turbofan engine degradation benchmarking data sets (Saxena & Goebel, 2008). The investigated data sets all simulate the exponential degradation trajectories and condition monitoring measurements for a fleet of engineering assets that were run to failure, where each general engineering asset had either univariate or multivariate condition monitoring sensor measurements performed at fixed time intervals over its lifetime. The univariate or multivariate condition monitoring measurements contained underlying asset health index information that was representative of the true asset health and was trendable with respect to the remaining useful life of the asset. It is very important to point out that the condition monitoring sensor measurements in this work refers to the already extracted features (health indicators) from raw sensor measurements. How these features (health indicators) are extracted from raw sensor measurements is however outside the scope of this research.

The data sets investigated include training set examples and testing set examples. The training set examples represent historical (previously seen) engineering assets with condition monitoring measurements that were run to failure. The testing set examples represent future (completely unseen) general engineering assets with condition monitoring sensor measurements that were run to failure. The objective and challenge is therefore to propose a prognostics strategy and train a model on the condition monitoring measurements of the training set examples offline. The proposed prognostics strategy and trained model must then predict the remaining useful life from the condition monitoring measurements of the testing set examples fully online. The turbofan engine degradation benchmarking data sets allow for simple and effective prognostics performance comparisons between publications with different prognostics strategies and models.

The proposed general prognostics strategies for this work include a prognostics classification strategy and prognostics regression strategy.

The proposed prognostics classification strategy is to structure the remaining useful life modeling problem as a sequence-to-sequence classification deep learning problem, where the input sequence is the univariate or multivariate condition monitoring measurement time series and the target sequence is the remaining useful life classification time series. The remaining useful life classification time series for each individual training and testing set example consists of remaining useful life classes with different degradation levels that are based on its linearly decreasing remaining useful life time series with an applied threshold. This is similar to the classification strategy proposed by (Ramasso & Gouriveau, 2010) and (Ramasso & Gouriveau, 2014).

The prognostics regression strategy is to structure the remaining useful life modeling problem as a sequence-to-sequence regression deep learning problem, where the input sequence is the univariate or multivariate condition monitoring measurement time series and the target sequence is the remaining useful life regression time series. The remaining useful life regression time series for each individual training and testing set example consists of remaining useful life values that are based on its linearly decreasing remaining useful life time series with an applied threshold. This is similar to the regression strategy proposed by (Heimes, 2008) and (Zheng, et al., 2017).

The sequence-to-sequence classification and regression deep learning models then learns and generalizes the mapping between the condition monitoring measurement time series and the remaining useful life classification and regression time series for all the training set examples offline. The trained sequence-to-sequence classification and regression deep learning models is then used to predict the remaining useful life classification and regression time series from the condition monitoring measurement time series for all the testing set examples fully online.

The proposed sequence-to-sequence deep learning classification and regression model architectures that are investigated and compared for the proposed prognostics classification and regression strategies include a feedforward neural network (FNN), simple recurrent neural network (S-RNN), long short-term memory recurrent neural network (LSTM-RNN) and gated recurrent unit recurrent neural network (GRU-RNN).

The sequence-to-sequence deep learning classification and regression model architectures are trained on the training sets of the investigated data sets with the new and effective Adam algorithm (Kingma & Ba, 2015). The deep learning models are also regularized with a combination of the early stopping, weight decay and dropout (Srivastava, et al., 2014) regularization techniques in order to reduce overfitting and improve generalization and prediction performance (Goodfellow, et al., 2016).

Practical considerations and recommendations for applying the general prognostics strategies with the data-driven deep learning models on future data sets are also provided. This includes the motivation for the applied threshold, how to select the applied threshold and the importance of a consistent definition of failure for effective remaining useful life prediction and modeling for fleets of engineering assets.

Fleets of engineering assets with available historical run-to-failure condition monitoring measurements and trendable degradation trajectories are the specific focus of this work. This is because fleets of engineering assets that degrade at a large scale and over long periods of time with condition monitoring measurements has recently become very popular in numerous industries and generally results in very large data sets that include trendable run-to-failure data. The data-driven deep learning models that are investigated in this work also require these large amounts of trendable historical run-to-failure data in order to accurately model the nonlinear degradation trajectories and to accurately predict the remaining useful life of future engineering assets. The main motivation for specifically predicting the remaining useful life for fleets of engineering assets is to increase the reliability and decrease the operating costs for an entire fleet of assets at a large scale. Deep learning (artificial intelligence) models are especially useful for the automation of the complex and repetitive tasks like condition monitoring pattern recognition and remaining useful life prediction for fleets of engineering assets at a large scale and over long periods of time.

The novelty and contribution of this research is that the state-of-the-art proposed LSTM-RNN and GRU-RNN deep learning models have to date not been comprehensively investigated and compared for data-driven prognostics for fleets of engineering assets. GRU-RNN deep learning models have especially never been investigated or applied for data-driven prognostics for fleets of engineering assets.

The deep learning models are regularized with a combination of the early stopping, weight decay and dropout regularization techniques. Regularization can drastically reduce overfitting and improve the generalization and prediction performance of deep learning models, but is however rarely discussed or applied in previous work. The deep learning models are also trained with the new and effective Adam algorithm, which is more popular, numerically efficient and robust when compared with other traditional training algorithms used in previous work.

The mathematical background of the investigated deep learning models is also provided and discussed in detail, which is generally omitted or very briefly discussed in previous work.

The author also generated a prognostics data set (general asset degradation data set) that is very representative of a fleet of general engineering assets that are run-to-failure, where previous work generally only focuses data sets provided by other authors like the turbofan engine degradation data sets.

The majority of previous work structure the remaining useful life modeling problem as a sequence-to-sequence regression machine learning problem. There has however been very limited previous work done that structure the remaining useful life modeling problem as a sequence-to-sequence classification machine learning problem, which can be more convenient in practice and has other advantages that will be discussed later. This work therefore investigates

and compares both these prognostics classification and regression strategies with state-of-the-art deep learning models.

The motivation and reason for the applied threshold is provided and a simple strategy to determine the value of the applied threshold for the proposed prognostics classification and regression strategies is also provided, which is generally not discussed clearly in previous work.

The proposed prognostics classification and regression strategies are very general, clear and simple and can therefore easily be applied to future fleets of general engineering assets with trendable historical run-to-failure condition monitoring measurements, where previous work generally only focuses on one specific asset. The author also provides practical considerations and recommendations for applying the proposed general prognostics strategies with the proposed deep learning models on future data sets, which is generally omitted in previous work.

The proposed prognostics strategies and deep learning models were implemented with the open-source (free) Tensorflow (Google Brain, 2016) and easy to use Keras (Chollet, 2015) application programming interfaces (APIs) in Python. The software used in previous work is rarely discussed or generally requires complex and expensive proprietary software.

It is obvious but important to point out that the abovementioned claims on the novelty and contribution of this research are all to the best of the author's knowledge.

1.5 Dissertation Overview

The chapters of this work have the following layout and logic: The second chapter provides background and describes the investigated prognostics data sets. The third chapter describes and motivates the proposed prognostics strategies and deep learning models. Chapter four presents the results of the proposed prognostics classification strategy and deep learning classification models applied on the investigated data sets. Chapter five presents the results of the proposed prognostics regression strategy and deep learning regression models applied on the investigated data sets. The last chapter discusses the conclusions and recommendations for the prognostics strategies, deep learning models and future work.

2 Prognostics Data Sets

2.1 Motivation for Simulated Data Sets

The biggest challenge in the development and benchmarking of data-driven prognostics strategies and models is the lack of complete run-to-failure condition monitoring sensor measurements for fleets of engineering assets. The reason for this is that the proper procurement of complete run-to-failure condition monitoring sensor measurements is generally a very expensive and time-consuming task. Therefore researchers that do procure complete run-to-failure condition monitoring sensor measurements understandably often withhold their data sets due to proprietary and competitive reasons (Saxena, et al., 2008). The selection of trendable condition monitoring sensor measurements that are representative of true asset health can also be a challenging, expensive and time-consuming task.

It can generally be assumed that the degradation trajectory (fault evolution) of mechanical and general engineering assets is exponential (Saxena, et al., 2008). The degradation trajectories for many physical models like the Arrhenius and Eyring chemical reaction models and the Coffin-Mason mechanical crack growth model are exponential. The degradation trajectories for many other mechanical crack growth models like the Paris-Erdogan, Foreman, Walker and McEvily power law models can also be interpreted as semi-exponential (Mishra, et al., 2014). These physical models therefore further justify the exponential degradation assumption. The data sets investigated in this work therefore all assume exponential degradation and simulate condition monitoring measurements for a fleet of engineering assets that are run to failure.

2.2 General Asset Degradation Data Set

The general asset degradation data set was generated by the author and simulate the exponential degradation trajectories and condition monitoring measurements for a fleet of general assets that were run to failure. Each individual general asset example had univariate condition monitoring sensor measurements at fixed time intervals (arbitrarily measured in cycles) over its lifetime.

The general asset degradation data set includes 80 training set examples and 20 testing set examples. The training set examples represent 80 historical (previously seen) general assets with condition monitoring measurements that were run to failure. The testing set examples represent 20 future (completely unseen) general assets with condition monitoring sensor measurements that were run to failure.

The objective and challenge for the general asset degradation data set is therefore to propose a prognostics strategy and train a model on the condition monitoring measurements of the training set examples offline. The prognostics strategy and trained model then have to predict the remaining useful life from the condition monitoring measurements of the testing set examples fully online.

The univariate condition monitoring sensor measurement time series $S1(t)$ was generated by adding noise to the health index time series $HI(t)$ for each individual general asset example. The health index time series $HI(t)$ consist of a healthy segment and a degradation segment, which represents the health of the general asset before and after a fault manifests respectively. The degradation segment was generated with a degradation trajectory time series $DT(t)$ and the healthy segment was a horizontal line of random length with its value equal to the initial value of the degradation trajectory time series.

The degradation trajectory time series $DT(t)$ for each individual general asset example was generated with a stochastic exponential degradation model as shown in equation (2-1). The stochastic exponential degradation model parameters a, b and c were randomly sampled from normal distributions $\mathcal{N}$ for each individual general asset example that was run to failure. The degradation trajectory time series values were all between one and zero, where zero represented general asset failure.

$$DT(t) = 1 - ae^{\left(\frac{t}{b}\right)} - c, \text{with } a = \mathcal{N}(\mu_a, \sigma_a), b = \mathcal{N}(\mu_b, \sigma_b) \text{ and } c = \mathcal{N}(\mu_c, \sigma_c) \quad (2\text{-}1)$$

The stochastic exponential degradation model parameter a randomly scales the degradation trajectories vertically. The stochastic exponential degradation model parameter b randomly scales the degradation trajectories horizontally. The stochastic exponential degradation model parameter c randomly adds an offset to the degradation trajectories. The stochastic degradation model parameters were therefore unique for each individual general asset example and made the degradation trajectories significantly more realistic and challenging to model.

The health index time series $HI(t)$ for each individual general asset example was then generated with its degradation trajectory time series $DT(t)$ as shown in equation (2-2). The random model parameter d was sampled from a normal distribution for each individual general asset example.

$$HI(t) = \begin{cases} DT(0) & \text{for } t < d, \text{with } d = \mathcal{N}(\mu_d, \sigma_d) \\ DT(t-d) & \text{for } t \geq d \end{cases} \quad (2\text{-}2)$$

The healthy segment of the health index time series was generated with the initial value of the degradation trajectory time series for d time steps. The degradation segment of the health index time series was generated with the shifted degradation trajectory time series of d time steps.

The univariate condition monitoring sensor measurement time series $S1(t)$ for each individual general asset example was then generated as shown in equation (2-3). Noise that was sampled from a normal distribution, was added to the health index time series $HI(t)$ at each time step.

$$S1(t) = HI(t) + \mathcal{N}(\mu_n, \sigma_n)(t) \tag{2-3}$$

The definition of failure for each individual general asset example was when its health index time series $HI(t)$ decreased to zero. The fixed time intervals between the condition monitoring sensor measurements and the time of failure for the general assets were arbitrarily measured in cycles.

The S1 condition monitoring measurements for five randomly selected general asset examples in general asset degradation data set are shown in Figure 2-1. The presented S1 condition monitoring measurements were normalized with min-max scaling between 0 and 1 for the entire training and testing set. This was done to effectively present the variation in condition monitoring measurements across all the training and testing set examples.

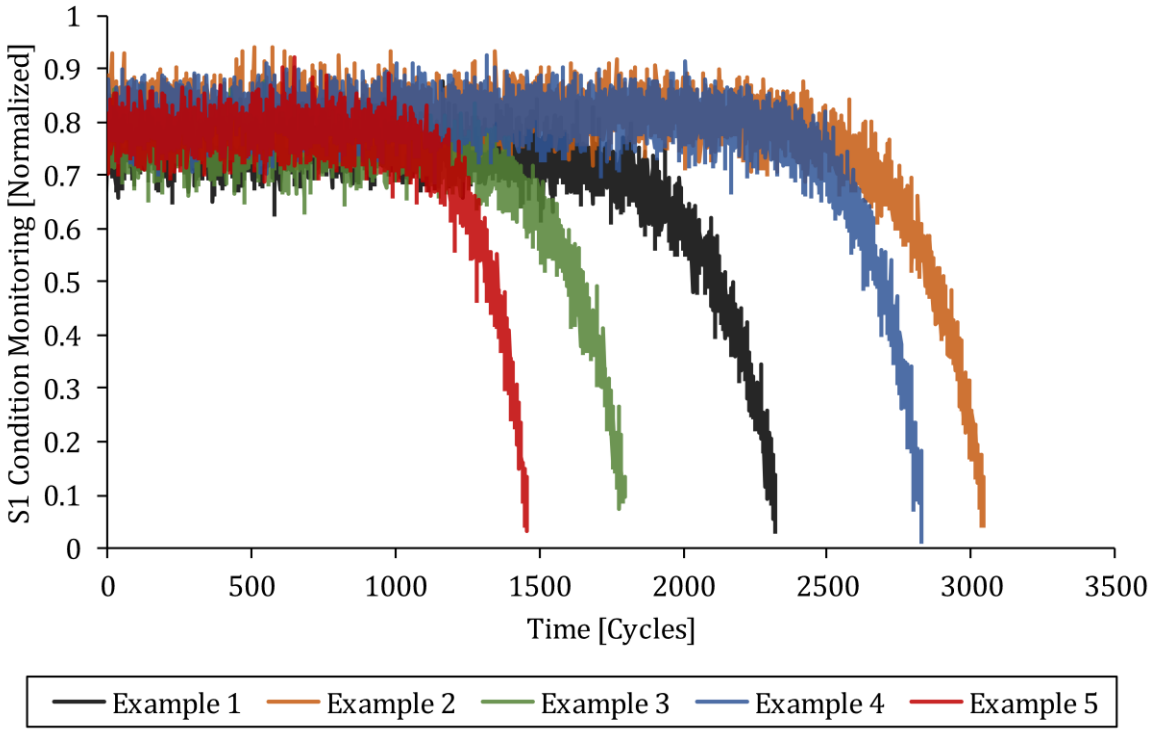

Figure 2-1: The S1 condition monitoring measurements for five randomly selected general asset examples in general asset degradation data set.

From Figure 2-1 it can be concluded that the S1 condition monitoring measurements is very noisy and that there is significant variance in the time of failure and degradation trajectories of the five randomly selected general asset examples. There is however definitely an observable trend in the S1 condition monitoring measurements of the five randomly selected general asset examples.

The mean, standard deviation, minimum and maximum statistics for the time of failure (measured in cycles) for the general asset examples in the training and testing set of the general asset degradation data set are shown in Table 2-1.

Table 2-1: The mean, standard deviation, minimum and maximum statistics for the time of failure (measured in cycles) of the general asset examples in the training and testing set of the general asset degradation data set.

Data Set	Mean	Standard Deviation	Minimum	Maximum
Training Set	2524.75	496.8644	1457	3434
Testing Set	2361.45	589.8072	1357	3198

From Table 2-1 it can be concluded that that there is significant variance in the time of failure for the general asset examples in the training and testing set of the general asset degradation data set. Remaining useful life prediction for a predictive maintenance or condition-based maintenance strategy would therefore be highly applicable and useful for this data set.

2.3 Turbofan Engine Degradation Data Set

The turbofan engine degradation data set simulate the exponential degradation trajectories and condition monitoring measurements for a fleet of turbofan engines that were run to failure. Each individual turbofan engine example had 24 multivariate condition monitoring sensor measurements at fixed time intervals (arbitrarily measured in cycles) over its lifetime. The turbofan engine degradation data set was based on the training set of the FD003 turbofan engine degradation benchmarking data set (Saxena & Goebel, 2008). This was because the testing set of the FD003 turbofan engine degradation benchmarking data set was censored and did contain full run-to-failure condition monitoring measurements for each individual turbofan engine. The turbofan engines in the FD003 turbofan engine degradation benchmarking data set are operated at one operating condition and have two fault modes namely high-pressure compressor degradation and fan degradation. The training set of the FD003 turbofan degradation benchmarking data set includes 100 examples that are split into 80 training set examples and 20 testing set examples and is referred to as the turbofan engine degradation data set for the rest of this work.

The turbofan engine degradation data set therefore include 80 training set examples and 20 testing set examples. The training set examples represent 80 historical (previously seen) turbofan engines with condition monitoring measurements that were run to failure. The testing set examples represent 20 future (completely unseen) turbofan engines with condition monitoring sensor measurements that were run to failure.

The objective and challenge for the turbofan engine degradation data set is therefore to propose a prognostics strategy and train a model on the condition monitoring measurements of the training set examples offline. The prognostics strategy and trained model then have to predict the remaining useful life from the condition monitoring measurements of the testing set examples fully online.

The turbofan engines that were run to failure each had 24 condition monitoring sensor measurements that were measured at fixed time intervals. The condition monitoring measurement description for each sensor in the turbofan engine degradation data set is shown in Table 2-2 (Saxena, et al., 2008). The condition monitoring measurements were also contaminated with sensor noise.

Table 2-2: The condition monitoring measurement description for each sensor in the turbofan engine degradation data set.

Sensor	**Condition Monitoring Measurement Description**
S1	Altitude
S2	Mach Number
S3	Throttle Resolver Angle
S4	Total Temperature At Fan Inlet
S5	Total Temperature At Low-Pressure Compressor Outlet
S6	Total Temperature At High-Pressure Compressor Outlet
S7	Total Temperature At Low-Pressure Turbine Outlet
S8	Pressure At Fan Inlet
S9	Total Pressure In Bypass-Duct
S10	Total Pressure At High-Pressure Compressor Outlet
S11	Physical Fan Speed
S12	Physical Core Speed
S13	Engine Pressure Ratio
S14	Static Pressure At High-Pressure Compressor Outlet
S15	Ratio of Fuel Flow To Static Pressure At High-Pressure Compressor Outlet
S16	Corrected Fan Speed
S17	Corrected Core Speed
S18	Bypass Ratio
S19	Burner Fuel-Air Ratio
S20	Bleed Enthalpy
S21	Demanded Fan Speed
S22	Demanded Corrected Fan Speed
S23	High-Pressure Turbine Coolant Bleed
S24	Low-Pressure Turbine Coolant Bleed

The units of the condition monitoring sensor measurements are not important for degradation modeling, but are given in (Saxena, et al., 2008). It is however only important that the units of the condition monitoring sensor measurements are consistent between the turbofan engine examples in the training and testing set. More information on how the multivariate condition monitoring measurements and exponential degradation trajectories for the fleet of turbofan engines were generated is also given in (Saxena, et al., 2008).

The health index time series for each individual turbofan engine example was however censored and not directly included in its 24 multivariate condition monitoring sensor measurements. The definition of failure for each individual turbofan engine example was when its censored health index time series decreased to zero. The fixed time intervals between the condition monitoring sensor measurements and the time of failure for the turbofan engines were arbitrarily measured in cycles.

The S10 condition monitoring measurements for five randomly selected turbofan engine examples with high-pressure compressor degradation in the turbofan engine degradation data set are shown in Figure 2-2. The presented S10 condition monitoring measurements were normalized with min-max scaling between 0 and 1 for the entire training and testing set. This was done to effectively present the variation in condition monitoring measurements across all the training and testing set examples.

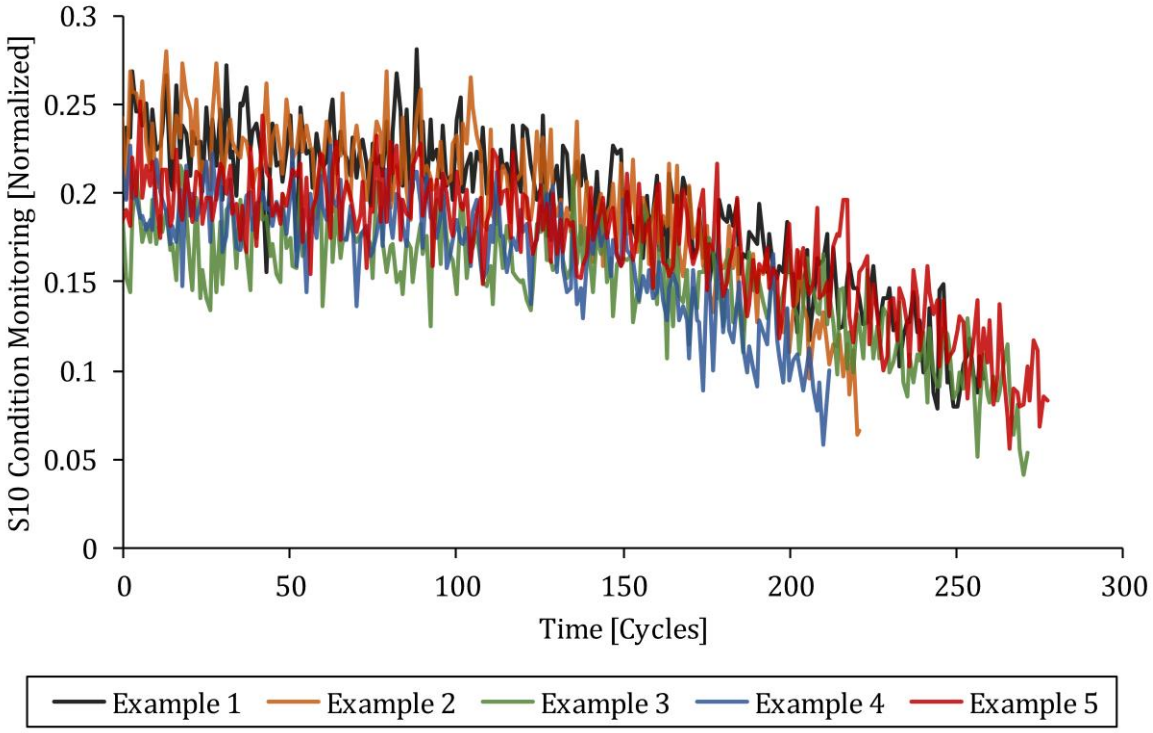

Figure 2-2: The S10 condition monitoring measurements for five randomly selected turbofan engine examples with high-pressure compressor degradation in the turbofan engine degradation data set.

The S10 condition monitoring measurements for five randomly selected turbofan engine examples with fan degradation in the turbofan engine degradation data set are shown in Figure 2-3.

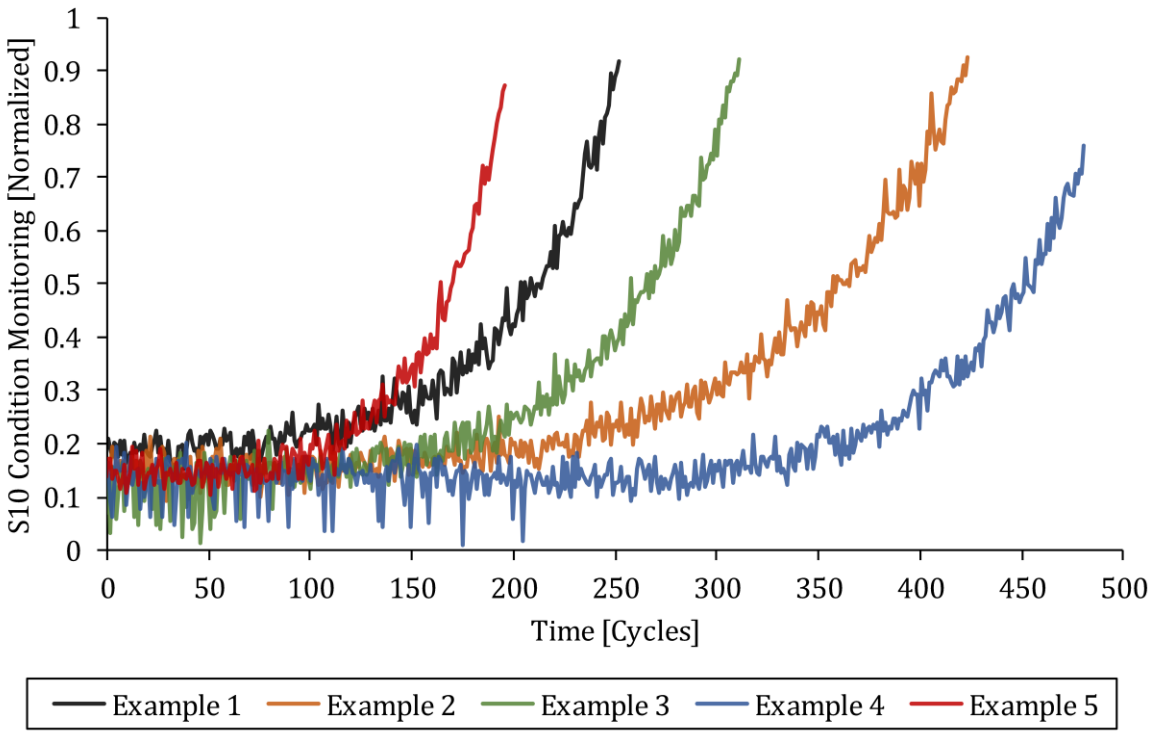

Figure 2-3: The S10 condition monitoring measurements for five randomly selected turbofan engine examples with fan degradation in the turbofan engine degradation data set.

From Figure 2-2 and Figure 2-3 it can be concluded that the S10 condition monitoring measurements are very noisy and that there is significant variance in the time of failure and degradation trajectories of the ten randomly selected turbofan engine examples. The S10 condition monitoring measurements for the turbofan engines with high-pressure compressor degradation decreased over time and the S10 condition monitoring measurements for the turbofan engines with fan degradation increased over time. There is however definitely an observable trend in the S10 condition monitoring measurements of the ten randomly selected turbofan engine examples.

The mean, standard deviation, minimum and maximum statistics for the time of failure (measured in cycles) for the turbofan engine examples in the training and testing set of the turbofan engine degradation data set are shown in Table 2-3.

Table 2-3: The mean, standard deviation, minimum and maximum statistics for the time of failure (measured in cycles) of the turbofan engine examples in the training and testing set of the turbofan engine degradation data set.

Data Set	Mean	Standard Deviation	Minimum	Maximum
Training Set	246.2375	84.0318	146	524
Testing Set	246.0500	93.6901	144	490

From Table 2-3 it can be concluded that that there is significant variance in the time of failure for the turbofan engine examples in the training and testing set of the turbofan engine degradation data set. Remaining useful life prediction for a predictive maintenance or condition-based maintenance strategy would therefore be highly applicable and useful for this data set.

The biggest differences and challenges between the general asset degradation data set and turbofan engine degradation data set are as follows: The general asset degradation data set only has univariate condition monitoring sensor measurements, where the turbofan engine degradation data set has 24 multivariate condition monitoring sensor measurements that are significantly more complex to model. The general asset degradation data set has significantly longer and noisy condition monitoring sensor measurement time series than the turbofan engine degradation data set that are significantly more complex to model. The general asset degradation data set has significant amounts of healthy condition monitoring sensor measurements that has to be ignored for remaining useful life modeling and prediction when compared to the turbofan engine degradation data set that has less amounts of healthy condition monitoring sensor measurements. The turbofan engine degradation data set has two different fault modes when compared to the general asset degradation data set that only has one fault mode.

The motivation for investigating both the general asset degradation data set and turbofan engine degradation data set is to demonstrate that the proposed prognostics strategies and deep learning models presented in chapter 3 are very general and can therefore be applied to future data sets.

2.4 Turbofan Engine Degradation Benchmarking Data Sets

The turbofan engine degradation benchmarking data sets (Saxena & Goebel, 2008) simulate the exponential degradation trajectories and condition monitoring measurements for a fleet of turbofan engines that were run to failure under different combinations of operating conditions and fault modes. Each individual turbofan engine example also had 24 multivariate condition monitoring sensor measurements at fixed time intervals (arbitrarily measured in cycles) over its lifetime (exactly similar to the turbofan engine degradation data set in the previous section).

The turbofan degradation benchmarking data sets include five data sets namely the FD001, FD002, FD003, FD004 and PHM08 turbofan degradation benchmarking data set respectively.

The FD001 turbofan degradation benchmarking data set includes 100 training set examples and 100 censored testing set examples. The turbofan engines in the FD001 turbofan degradation benchmarking data set are operated at one operating condition and have one fault mode namely high-pressure compressor degradation.

The FD002 turbofan degradation benchmarking data set includes 260 training set examples and 259 censored testing set examples. The turbofan engines in the FD002 turbofan degradation benchmarking data set are operated at six different operating conditions and have one fault mode namely high-pressure compressor degradation.

The FD003 turbofan degradation benchmarking data set includes 100 training set examples and 100 censored testing set examples. The turbofan engines in the FD003 turbofan degradation benchmarking data set are operated at one operating condition and have two different fault modes namely high-pressure compressor degradation and fan degradation.

The FD004 turbofan degradation benchmarking data set includes 248 training set examples and 249 censored testing set examples. The turbofan engines in the FD004 turbofan degradation benchmarking data set are operated at six different operating condition and have two different fault modes namely high-pressure compressor degradation and fan degradation.

The PHM08 turbofan degradation benchmarking data set includes 218 training set examples and 218 censored testing set examples. The turbofan engines in the PHM08 turbofan degradation benchmarking data set are operated at six different operating condition and have one fault mode namely high-pressure compressor degradation.

The training set examples represent historical (previously seen) turbofan engines that were run to failure with condition monitoring measurements. The censored testing set examples represent future (completely unseen) turbofan engines that were run to failure, but with condition monitoring sensor measurements that were censored for a random number of cycles before failure for each individual turbofan engine. With the idea of then predicting the censored number of cycles before failure (remaining useful life value) for each individual turbofan engine example in the testing set from its available (non-censored) condition monitoring measurements.

The objective and challenge for each turbofan engine degradation benchmarking data set is therefore to propose a prognostics strategy and train a model on the condition monitoring measurements of the training set examples. The prognostics strategy and trained model then have to predict the remaining useful life value for each individual turbofan engine example in the testing set from its available condition monitoring measurements.

The benchmarking score $\bar{S}$ for each turbofan degradation benchmarking data set is calculated with the predicted remaining useful life value for each individual turbofan engine example in the testing set as shown in equation (2-4) (Saxena, et al., 2008). $a^{(i)}$ is the difference between the predicted remaining useful life value $\hat{r}^{(i)}$ and actual remaining useful life value $r^{(i)}$ for each individual turbofan engine example in the testing set. N is the number of turbofan engines in the testing set. The benchmarking score therefore penalizes incorrect predictions exponentially, with late predictions that are penalized more than early predictions. A lower benchmarking score therefore indicates more accurately predicted remaining useful life values for the turbofan engines in the testing set.

$$\bar{S} = \sum_{i=1}^{N} \begin{cases} e^{-\left(\frac{a^{(i)}}{13}\right)} - 1, \text{for } a^{(i)} < 0, \text{with } a^{(i)} = \hat{r}^{(i)} - r^{(i)} \\ e^{\left(\frac{a^{(i)}}{10}\right)} - 1, \text{for } a^{(i)} \geq 0, \text{with } a^{(i)} = \hat{r}^{(i)} - r^{(i)} \end{cases} \tag{2-4}$$

The predicted remaining useful life values for the turbofan engine examples in the testing set of the PHM08 turbofan degradation benchmarking data set were uploaded to the NASA Prognostics Data Repository (NASA, 2018) website. The benchmarking score was then returned. The NASA Prognostics Data Repository website however only allowed for one attempt (submission) per day.

It is however important to point out that the author benchmarked each data-driven model only once on the testing set for each turbofan degradation benchmarking data set, in the spirit of the competition and ethics. Multiple submissions would allow each data-driven model to be tuned on the testing set and would therefore not be representative of real world conditions.

The benchmarking root mean squared error $\overline{RMSE}$ for each turbofan degradation benchmarking data set is calculated similarly with the predicted remaining useful life value for each individual turbofan engine example in the testing set as shown in equation (2-5). The benchmarking root mean squared error however only penalized incorrect predictions quadratically. The motivation for the benchmarking root mean squared error over the benchmarking score was that it penalized remaining useful value predictions that were very inaccurate less severely and was therefore a better indication of model performance on the entire testing set fleet on average. A lower benchmarking root mean squared error therefore indicated more accurately predicted remaining useful life values for the turbofan engines in the testing set on average.

$$\overline{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\hat{r}_i - r_i)^2} \tag{2-5}$$

The benchmarking score and benchmarking root mean squared error allowed for simple and effective prognostics performance comparisons between publications with different prognostics strategies and models on the turbofan degradation benchmarking data sets.

3 Prognostics Strategies and Deep Learning Models

3.1 Prognostics Strategies

This section explains the proposed prognostics classification and regression strategies and discuss their important practical considerations.

3.1.1 Remaining Useful Life Definition

The linearly decreasing remaining useful life time series $RUL(t)$ for each individual training and testing set example in each investigated data set is defined and calculated with its time of failure TOF as shown in equation (3-1).

$$RUL(t) = TOF - t \tag{3-1}$$

3.1.2 Prognostics Classification Strategy

The proposed prognostics classification strategy is to structure the remaining useful life modeling problem as a sequence-to-sequence classification deep learning problem. The input sequence is the univariate or multivariate condition monitoring measurement time series $CM(t)$ and the target sequence is the remaining useful life classification time series $RUL_{CN}(t)$. The remaining useful life classification time series for each individual training and testing set example consists of remaining useful life classes with different degradation levels that are based on its linearly decreasing remaining useful life time series with an applied threshold.

The remaining useful life classification time series $RUL_{CN}(t)$ for each individual training and testing set example in each investigated data set is defined and labeled as shown in equation (3-2). Different sections of the linearly decreasing remaining useful life time series $RUL(t)$ defined in equation (3-1) are given a number n of different class labels C_n with an applied threshold AT and different bounds B_n. It is important to point out that the applied threshold and different bounds are constant for all the training and testing set examples in each investigated data set.

$$RUL_{CN}(t) = \begin{cases} C_1, \text{for } RUL(t) > AT \\ C_2, \text{for } B_1 < RUL(t) \leq AT \\ C_3, \text{for } B_2 < RUL(t) \leq B_1 \\ C_4, \text{for } B_{n-2} < RUL(t) \leq B_2 \\ \vdots \\ C_n, \text{for } RUL(t) \leq B_{n-2} \end{cases} \tag{3-2}$$

The sequence-to-sequence classification deep learning model then learns and generalizes the mapping between the condition monitoring measurement time series and the remaining useful life classification time series for all the training set examples offline. The trained sequence-to-sequence classification deep learning model is then used to predict the remaining useful life classification time series from the condition monitoring measurement time series for all the testing set examples fully online. This is similar to the classification strategy proposed by (Ramasso & Gouriveau, 2010) and (Ramasso & Gouriveau, 2014).

3.1.3 Prognostics Regression Strategy

The proposed prognostics regression strategy is to structure the remaining useful life modeling problem as a sequence-to-sequence regression deep learning problem. The input sequence is the univariate or multivariate condition monitoring measurement time series $CM(t)$ and the target sequence is the remaining useful life regression time series $RUL_{RN}(t)$. The remaining useful life regression time series for each individual training and testing set example consists of remaining useful life values that are based on its linearly decreasing remaining useful life time series with an applied threshold.

The remaining useful life regression time series $RUL_{RN}(t)$ for each individual training and testing set example in each investigated data set is defined and labeled as shown in equation (3-3). The linearly decreasing remaining useful life time series $RUL(t)$ defined in equation (3-1) is given an applied threshold AT. It is important to point out that the applied threshold is constant for all the training and testing set examples in each investigated data set.

$$RUL_{RN}(t) = \begin{cases} RUL(t), \text{for } RUL(t) \leq AT \\ AT, \text{for } RUL(t) > AT \end{cases} \tag{3-3}$$

The sequence-to-sequence regression deep learning model then learns and generalizes the mapping between the condition monitoring measurement time series and the remaining useful life regression time series for all the training set examples offline. The trained sequence-to-sequence regression deep learning model is then used to predict the remaining useful life regression time series from the condition monitoring measurement time series for all the testing set examples fully online. This is similar to the regression strategy proposed by (Heimes, 2008), (Lim, et al., 2014), (Rigamonti, et al., 2016) and (Zheng, et al., 2017).

3.1.4 Motivation for Applied Threshold

The condition monitoring measurement time series for the training and testing set examples in the investigated data sets include healthy and light degradation condition monitoring measurements that generally have very little to no trend as shown in Figure 2-1, Figure 2-2 and Figure 2-3. It is therefore understandably very difficult for the sequence-to-sequence classification and regression deep learning models to learn and generalize the mapping between the healthy and light degradation condition monitoring measurement time series and the linearly decreasing remaining useful life time series with no applied threshold. The applied threshold therefore gave the healthy and light degradation condition monitoring measurements the same remaining useful life class or value target label. This implied that the healthy and light degradation condition monitoring measurements were ignored for remaining useful life class and value modeling. This made it significantly simpler and less confusing for the sequence-to-sequence classification and regression deep learning models to learn and generalize the mapping between the condition monitoring measurement time series and the remaining useful life classification and regression time series. The applied threshold also resulted in sequence-to-sequence classification and regression deep learning models that are more conservative, fully online, easier to train, generalize better and made more accurate predictions below the applied threshold, compared to when no threshold is applied. The value of the applied threshold is a hyperparameter for the prognostics classification and regression strategies that has to be tuned for each investigated data set. The strategy to determine the applied threshold value is proposed and explained in section 3.6.9.

3.1.5 Importance of the Definition of Failure

The objective of the sequence-to-sequence classification and regression deep learning models is to learn and generalize the mapping between the condition monitoring measurement time series and the remaining useful life classification and regression time series for the training and testing set examples in each investigated data set. The condition monitoring measurement time series contains underlying asset health index information and the remaining useful life classification and regression time series are based on the time of failure. It is therefore critical that the definition of failure is consistent with respect to the condition monitoring measurement time series and the underlying asset health index information between the training and testing set examples for each data set. The definition of failure was however consistent between training and testing set examples for each data set as discussed in chapter 2, but is important to point out when applying the prognostics regression and classification strategies on future data sets.

3.1.6 Comparison of Strategies

The prognostics regression strategy has the advantage that it can be more useful to know the exact predicted remaining useful life value for maintenance planning and scheduling, than the predicted remaining useful life class. The prognostics classification strategy however has the advantage that it can be more convenient to interpret the predicted remaining useful life class as a degradation severity level for online maintenance decision-making, than the predicted remaining useful life value. The prognostics classification strategy also has the advantage that it can be used for fault mode classification for maintenance decision-making and is a significantly simpler problem to model than the prognostics regression strategy. The prognostics regression and classification strategies therefore both have their merits and limitations depending on the predictive maintenance requirements.

3.2 Deep Learning Models

This section provides an overview of the proposed sequence-to-sequence deep learning model architectures that are investigated and compared for the prognostics classification and regression strategies. The provided deep learning mathematical background and rationale for the rest of this chapter is mainly based on the work of (Goodfellow, et al., 2016). The author also provides general deep learning modeling advice, conclusions and recommendations that are not always validated and motivated with results, as the focus of this work was on prognostics and not deep learning.

3.2.1 Background

The deep learning model architectures that are investigated and compared for the prognostics classification and regression strategies consist of multiple stacked layers with trainable parameters and nonlinear activation functions. This makes it possible for deep learning model architectures to learn and generalize the complex nonlinear mapping between the condition monitoring measurement time series and the remaining useful life classification and regression time series for the training and testing set examples in each data set. The model layers that are applied in the proposed model architectures include an input layer, fully-connected hidden layer, simple recurrent hidden layer, LSTM hidden layer, GRU hidden layer, regression output layer and classification output layer. The mathematical background and rationale for these model layers are provided in section 3.4. The activation functions that are applied in the layers of the proposed model architectures include the hyperbolic tangent activation function, sigmoid activation function, linear activation function and softmax activation function. The mathematical background and rationale for these model activation functions are provided in section 3.3.

3.2.2 Classification Model Architectures

The proposed classification model architectures that are investigated and compared for the prognostics classification strategy include a feedforward neural network (FNN), simple recurrent neural network (S-RNN), long short-term memory recurrent neural network (LSTM-RNN) and gated recurrent unit recurrent neural network (GRU-RNN). The mathematical background for the classification model architectures is provided in section 3.5.

The FNN classification model architecture consists of an input layer, three fully-connected hidden layers and classification output layer from input to output.

The S-RNN classification model architecture consists of an input layer, fully-connected hidden layer, simple recurrent hidden layer, fully-connected hidden layer and classification output layer from input to output.

The LSTM-RNN classification model architecture consists of an input layer, fully-connected hidden layer, LSTM hidden layer, fully-connected hidden layer and classification output layer from input to output.

The GRU-RNN classification model architecture consists of an input layer, fully-connected hidden layer, GRU hidden layer, fully-connected hidden layer and classification output layer from input to output.

3.2.3 Regression Model Architectures

The proposed regression model architectures that are investigated and compared for the prognostics regression strategy also include a feedforward neural network (FNN), simple recurrent neural network (S-RNN), long short-term memory recurrent neural network (LSTM-RNN) and gated recurrent unit recurrent neural network (GRU-RNN). The mathematical background for the regression model architectures is also provided in section 3.5.

The FNN, S-RNN, LSTM-RNN and GRU-RNN regression model architectures were exactly the same as the classification model architectures, except that the last classification output layer was a regression output layer for the regression model architectures.

3.2.4 Model Input and Target Vectors

The classification and regression deep learning models all have input vectors and associated target vectors for each time step. It is therefore important to clarify how the input condition monitoring measurement time series $CM(t)$, target remaining useful life classification time series $RUL_{CN}(t)$ and target remaining useful life regression time series $RUL_{RN}(t)$ were encoded as input vectors and associated target vectors for each time step.

The input condition monitoring measurement time series $CM(t)$ for each individual training and testing set example is encoded as a sequence of condition monitoring measurement input vectors $\left(CM_i^{(t)}, CM_i^{(t+1)}, CM_i^{(t+2)}, \dots, CM_i^{(t+M-1)}\right)$. The condition monitoring measurement input vector for the current time step $CM_i^{(t)} \in \mathbb{R}^{D\times 1}$ includes all the condition monitoring sensor measurements for the current time step. M is the variable number of time steps for each individual training and testing set example in each data set and D is the total number of condition monitoring sensor measurements.

The target remaining useful life classification time series $RUL_{CN}(t)$ for each individual training and testing set example is encoded as a sequence of remaining useful life class target vectors $\left(RUL_{i,CN}^{(t)}, RUL_{i,CN}^{(t+1)}, RUL_{i,CN}^{(t+2)}, \dots, RUL_{i,CN}^{(t+M-1)}\right)$. The remaining useful life class target vector for the current time step $RUL_{i,CN}^{(t)} \in \mathbb{R}^{C\times 1}$ is the categorically encoded class vector for the current time step. C is the number of possible target classes. Examples of categorically encoded class vectors for a hypothetical data set with four possible target classes are shown below:

$$RUL_{i,CN}^{(t)} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \text{for } C_1 \qquad RUL_{i,CN}^{(t)} = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \text{for } C_2 \qquad RUL_{i,CN}^{(t)} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \text{for } C_3 \qquad RUL_{i,CN}^{(t)} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}, \text{for } C_4$$

The target remaining useful life regression time series $RUL_{RN}(t)$ for each individual training and testing set example is encoded as a sequence of remaining useful life value target vectors $\left(RUL_{i,RN}^{(t)}, RUL_{i,RN}^{(t+1)}, RUL_{i,RN}^{(t+2)}, \dots, RUL_{i,RN}^{(t+M-1)}\right)$. The remaining useful life value target vector for the current time step $RUL_{i,RN}^{(t)} \in \mathbb{R}^{1\times 1}$ is the remaining useful life value for the current time step.

3.3 Model Activation Functions

This section describes the activation functions that are applied in the hidden and output layers of the proposed model architectures. The activation functions include the hyperbolic tangent activation function, sigmoid activation function, linear activation function and softmax activation function.

3.3.1 Hyperbolic Tangent Activation Function

The hyperbolic tangent activation function (Bishop, 2006) is used to introduce nonlinearity to the fully-connected, simple recurrent, LSTM and GRU hidden layers. The hyperbolic tangent activation function T is calculated element-wise for any vector z_i as shown in equation (3-4) and scales the vector elements between -1 and 1.

$$T(z_i) = \frac{e^{z_i} - e^{-z_i}}{e^{z_i} + e^{-z_i}} \tag{3-4}$$

3.3.2 Sigmoid Activation Function

The sigmoid activation function (Bishop, 2006) is used for gating operations in the LSTM and GRU hidden layers. The sigmoid activation function S is calculated element-wise for any vector z_i as shown in equation (3-5) and scales the vector elements between 0 and 1.

$$S(z_i) = \frac{1}{1 + e^{-z_i}} \tag{3-5}$$

3.3.3 Softmax Activation Function

The softmax activation function (Bishop, 2006) is used in the classification output layer of the classification model architectures. The softmax activation function E is calculated element-wise for any vector z_i as shown in equation (3-6). The softmax activation function applies an exponential operation to the vector elements and scales all vector elements such that their sum adds up to one. The length of the arbitrary vector z_i is equal to the number of target classes C for the classification strategy.

$$E(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{C} e^{z_j}} \tag{3-6}$$

3.3.4 Linear Activation Function

The linear activation function L is used in the regression output layer of the regression model architectures. The linear activation function L is calculated element-wise for any vector z_i as shown in equation (3-7) and applies a linear operation to the vector elements.

$$L(z_i) = z_i \tag{3-7}$$

3.4 Model Layers

This section describes the model layers that are applied in the model architectures presented in section 3.5. The applied layers include an input layer, fully-connected hidden layer, simple recurrent hidden layer, LSTM hidden layer, GRU hidden layer, regression output layer and classification output layer. The superscript $[n]$ in this section is used to distinguish between different stacked layers and their respective trainable parameters for the proposed model architectures presented in section 3.5.

3.4.1 Input Layer

The input layer is the first layer in all the model architectures. The output vector of the input layer does not have any trainable parameters, but its length is determined by the dimension of condition monitoring measurement input vector D.

The input vector of the input layer is the condition monitoring measurement input vector for the current time step $CM_i^{(t)} \in \mathbb{R}^{D\times 1}$. The output vector of the input layer for the current time step $h_i^{(t),[n]} \in \mathbb{R}^{D\times 1}$ is calculated as shown in equation (3-8).

$$h_i^{(t),[n]} = CM_i^{(t)} \tag{3-8}$$

3.4.2 Fully-Connected Hidden Layer

The fully-connected hidden layer (Bishop, 2006) is used in all the classification and regression model architectures. The fully-connected hidden layer is very simple and can model general nonlinear input to output mappings, but cannot model sequence information from previous time steps. The size of the trainable parameters for the fully-connected hidden layer is determined by the length of the output vector from the previous layer I and the number of selected hidden units H for the layer.

The input vector of the fully-connected hidden layer is the output vector from the previous layer for the current time step $h_j^{(t),[n-1]} \in \mathbb{R}^{I\times 1}$. The output vector of the fully-connected hidden layer for the current time step $h_i^{(t),[n]} \in \mathbb{R}^{H\times 1}$ is calculated element-wise with the hyperbolic tangent activation function T as shown in equation (3-9). The trainable parameters for the output vector of the fully-connected hidden layer is its weight matrix $W_{i,j}^{[n]} \in \mathbb{R}^{H\times I}$ and bias vector $b_i^{[n]} \in \mathbb{R}^{H\times 1}$.

$$h_i^{(t),[n]} = T\left(\sum_j W_{i,j}^{[n]} h_j^{(t),[n-1]} + b_i^{[n]}\right) \tag{3-9}$$

3.4.3 Simple Recurrent Hidden Layer

The simple recurrent hidden layer (Elman, 1990) is used in the simple recurrent neural network classification and regression model architectures. The simple recurrent hidden layer is very simple and can model general nonlinear input to output mappings and sequence information from previous time steps. The simple recurrent hidden layer does however suffer from the vanishing and exploding gradient problem (Hochreiter & Schmidhuber, 1997) during model training, which drastically limits its capacity to model long-term sequence information. The size of the trainable parameters for the simple recurrent hidden layer is determined by the length of the output vector from the previous layer I and the number of selected hidden units H for the layer.

The input vectors of the simple recurrent hidden layer is its output vector from the previous time step $h_j^{(t-1),[n]} \in \mathbb{R}^{H\times 1}$ and the output vector from the previous layer for the current time step $h_j^{(t),[n-1]} \in \mathbb{R}^{I\times 1}$. The output vector of the simple recurrent hidden layer for the current time step $h_i^{(t),[n]} \in \mathbb{R}^{H\times 1}$ is calculated element-wise with the hyperbolic tangent activation function T as shown in equation (3-10). The trainable parameters for the output vector of the simple recurrent hidden layer is its weight matrix $W_{i,j}^{[n]} \in \mathbb{R}^{H\times I}$, recurrent weight matrix $R_{i,j}^{[n]} \in \mathbb{R}^{H\times H}$ and bias vector $b_i^{[n]} \in \mathbb{R}^{H\times 1}$.

$$h_i^{(t),[n]} = T\left(\sum_j W_{i,j}^{[n]} h_j^{(t),[n-1]} + \sum_j R_{i,j}^{[n]} h_j^{(t-1),[n]} + b_i^{[n]}\right) \tag{3-10}$$

3.4.4 Long Short-Term Memory (LSTM) Hidden Layer

The long short-term memory (LSTM) hidden layer (Hochreiter & Schmidhuber, 1997) is used in the LSTM-RNN classification and regression model architectures. The LSTM hidden layer can model general nonlinear input to output mappings and long-term sequence information from previous time steps. The LSTM hidden layer introduced the concept of a cell state with an update gate, forget gate, candidate cell state and output gate to manage the vanishing and exploding gradient problem (Hochreiter & Schmidhuber, 1997) during model training, which drastically increased its capacity to model long-term sequence information. The LSTM hidden layer is however significantly more complex than the simple recurrent hidden layer. The size of the trainable parameters for the LSTM hidden layer is determined by the length of the output vector from the previous layer I and the number of selected hidden units H for the layer.

The input vectors of the LSTM hidden layer is its output vector from the previous time step $h_j^{(t-1),[n]} \in \mathbb{R}^{H\times 1}$, its cell state vector from the previous time step $c_i^{(t-1),[n]} \in \mathbb{R}^{H\times 1}$ and the output vector from the previous layer for the current time step $h_j^{(t),[n-1]} \in \mathbb{R}^{I\times 1}$.

The update gate vector of the LSTM hidden layer for the current time step $u_i^{(t),[n]} \in \mathbb{R}^{H\times 1}$ is calculated element-wise with the sigmoid activation function S as shown in equation (3-11). The trainable parameters for the update gate vector of the LSTM hidden layer is its update gate weight matrix $W_{i,j}^{u,[n]} \in \mathbb{R}^{H\times I}$, update gate recurrent weight matrix $R_{i,j}^{u,[n]} \in \mathbb{R}^{H\times H}$ and update gate bias vector $b_i^{u,[n]} \in \mathbb{R}^{H\times 1}$.

$$u_i^{(t),[n]} = S\left(\sum_j W_{i,j}^{u,[n]} h_j^{(t),[n-1]} + \sum_j R_{i,j}^{u,[n]} h_j^{(t-1),[n]} + b_i^{u,[n]}\right) \tag{3-11}$$

The forget gate vector of the LSTM hidden layer for the current time step $f_i^{(t),[n]} \in \mathbb{R}^{H\times 1}$ is calculated element-wise with the sigmoid activation function S as shown in equation (3-12). The trainable parameters for the forget gate vector of the LSTM hidden layer is its forget gate weight matrix $W_{i,j}^{f,[n]} \in \mathbb{R}^{H\times I}$, forget gate recurrent weight matrix $R_{i,j}^{f,[n]} \in \mathbb{R}^{H\times H}$ and forget gate bias vector $b_i^{f,[n]} \in \mathbb{R}^{H\times 1}$.

$$f_i^{(t),[n]} = S\left(\sum_j W_{i,j}^{f,[n]} h_j^{(t),[n-1]} + \sum_j R_{i,j}^{f,[n]} h_j^{(t-1),[n]} + b_i^{f,[n]}\right) \tag{3-12}$$

The candidate cell state vector of the LSTM hidden layer for the current time step $s_i^{(t),[n]} \in \mathbb{R}^{H\times 1}$ is calculated element-wise with the hyperbolic tangent activation function T as shown in equation (3-13). The trainable parameters for the candidate cell state vector of the LSTM hidden layer is its candidate cell state weight matrix $W_{i,j}^{s,[n]} \in \mathbb{R}^{H\times I}$, candidate cell state recurrent weight matrix $R_{i,j}^{s,[n]} \in \mathbb{R}^{H\times H}$ and candidate cell state bias vector $b_i^{s,[n]} \in \mathbb{R}^{H\times 1}$.

$$s_i^{(t),[n]} = T\left(\sum_j W_{i,j}^{s,[n]} h_j^{(t),[n-1]} + \sum_j R_{i,j}^{s,[n]} h_j^{(t-1),[n]} + b_i^{s,[n]}\right) \tag{3-13}$$

The cell state vector of the LSTM hidden layer for the current time step $c_i^{(t),[n]} \in \mathbb{R}^{H\times 1}$ is updated as shown in equation (3-14).

$$c_i^{(t),[n]} = u_i^{(t),[n]} s_i^{(t),[n]} + f_i^{(t),[n]} c_i^{(t-1),[n]} \tag{3-14}$$

The output gate vector of the LSTM hidden layer for the current time step $o_i^{(t),[n]} \in \mathbb{R}^{H\times 1}$ is calculated element-wise with the sigmoid activation function S as shown in equation (3-15). The trainable parameters for the output gate vector of the LSTM hidden layer is its output gate weight matrix $W_{i,j}^{o,[n]} \in \mathbb{R}^{H\times I}$, output gate recurrent weight matrix $R_{i,j}^{o,[n]} \in \mathbb{R}^{H\times H}$ and output gate bias vector $b_i^{o,[n]} \in \mathbb{R}^{H\times 1}$.

$$o_i^{(t),[n]} = S\left(\sum_j W_{i,j}^{o,[n]} h_j^{(t),[n-1]} + \sum_j R_{i,j}^{o,[n]} h_j^{(t-1),[n]} + b_i^{o,[n]}\right) \tag{3-15}$$

The output vector of the LSTM hidden layer for the current time step $h_i^{(t),[n]} \in \mathbb{R}^{H\times 1}$ is calculated element-wise with the hyperbolic tangent activation function T as shown in equation (3-16).

$$h_i^{(t),[n]} = o_i^{(t),[n]}\, T\left(c_i^{(t),[n]}\right) \tag{3-16}$$

3.4.5 Gated Recurrent Unit (GRU) Hidden Layer

The gated recurrent unit (GRU) hidden layer (Cho, et al., 2014) is used in the GRU-RNN classification and regression model architectures. The GRU hidden layer can model general nonlinear input to output mappings and long-term sequence information from previous time steps. The GRU hidden layer also introduced the concept of a cell state with an update gate, reset gate and candidate cell state to manage the vanishing and exploding gradient problem (Hochreiter & Schmidhuber, 1997) during model training, which drastically increased its capacity to model long-term sequence information. The GRU hidden layer is slightly less complex with fewer trainable parameters when compared to the LSTM hidden layer, since it does not have an output gate. The GRU hidden layer is therefore less prone to overfitting. The GRU hidden layer has also been shown to outperform and generalize better on smaller data sets when compared to the LSTM hidden layer (Chung, et al., 2014). The size of the trainable parameters for the GRU hidden layer is determined by the length of the output vector from the previous layer I and the number of selected hidden units H for the layer.

The input vectors of the GRU hidden layer is its output vector from the previous time step $h_j^{(t-1),[n]} \in \mathbb{R}^{H\times 1}$, its cell state vector from the previous time step $c_i^{(t-1),[n]} \in \mathbb{R}^{H\times 1}$ and the output vector from the previous layer for the current time step $h_j^{(t),[n-1]} \in \mathbb{R}^{I\times 1}$.

The update gate vector of the GRU hidden layer for the current time step $u_i^{(t),[n]} \in \mathbb{R}^{H\times 1}$ is calculated element-wise with the sigmoid activation function S as shown in equation (3-17). The trainable parameters for the update gate vector of the GRU hidden layer is its update gate weight matrix $W_{i,j}^{u,[n]} \in \mathbb{R}^{H\times I}$, update gate recurrent weight matrix $R_{i,j}^{u,[n]} \in \mathbb{R}^{H\times H}$ and update gate bias vector $b_i^{u,[n]} \in \mathbb{R}^{H\times 1}$.

$$u_i^{(t),[n]} = S\left(\sum_j W_{i,j}^{u,[n]} h_j^{(t),[n-1]} + \sum_j R_{i,j}^{u,[n]} h_j^{(t-1),[n]} + b_i^{u,[n]}\right) \tag{3-17}$$

The reset gate vector of the GRU hidden layer for the current time step $f_i^{(t),[n]} \in \mathbb{R}^{H\times 1}$ is calculated element-wise with the sigmoid activation function S as shown in equation (3-18). The trainable parameters for the reset gate vector of the GRU hidden layer is its reset gate weight matrix $W_{i,j}^{r,[n]} \in \mathbb{R}^{H\times I}$, reset gate recurrent weight matrix $R_{i,j}^{r,[n]} \in \mathbb{R}^{H\times H}$ and reset gate bias vector $b_i^{r,[n]} \in \mathbb{R}^{H\times 1}$.

$$r_i^{(t),[n]} = S\left(\sum_j W_{i,j}^{r,[n]} h_j^{(t),[n-1]} + \sum_j R_{i,j}^{r,[n]} h_j^{(t-1),[n]} + b_i^{r,[n]}\right) \tag{3-18}$$

The candidate cell state vector of the GRU hidden layer for the current time step $s_i^{(t),[n]} \in \mathbb{R}^{H\times 1}$ is calculated element-wise with the hyperbolic tangent activation function T as shown in equation (3-19). The trainable parameters for the candidate cell state vector of the GRU hidden layer is its candidate cell state weight matrix $W_{i,j}^{s,[n]} \in \mathbb{R}^{H\times I}$, candidate cell state recurrent weight matrix $R_{i,j}^{s,[n]} \in \mathbb{R}^{H\times H}$ and candidate cell state bias vector $b_i^{s,[n]} \in \mathbb{R}^{H\times 1}$.

$$s_i^{(t),[n]} = T\left(\sum_j W_{i,j}^{s,[n]} h_j^{(t),[n-1]} + \sum_j R_{i,j}^{s,[n]} r_j^{(t),[n]} h_j^{(t-1),[n]} + b_i^{s,[n]}\right) \tag{3-19}$$

The cell state vector of the GRU hidden layer for the current time step $c_i^{(t),[n]} \in \mathbb{R}^{H\times 1}$ is updated as shown in equation (3-20).

$$c_i^{(t),[n]} = u_i^{(t),[n]} s_i^{(t),[n]} + \left(1 - u_i^{(t),[n]}\right) c_i^{(t-1),[n]} \tag{3-20}$$

The output vector of the GRU hidden layer for the current time step $h_i^{(t),[n]} \in \mathbb{R}^{H\times 1}$ is calculated as shown in equation (3-21).

$$h_i^{(t),[n]} = c_i^{(t),[n]} \tag{3-21}$$

3.4.6 Classification Output Layer

The classification output layer is the last layer in all the classification model architectures. The size of the trainable parameters for the classification output layer is determined by the length of the output vector from the previous layer I and the dimension of the remaining useful life class target vector C.

The input vector of the classification output layer is the output vector from the previous layer for the current time step $h_j^{(t),[n-1]} \in \mathbb{R}^{I\times 1}$. The remaining useful life class target vector for the current time step $RUL_{i,CN}^{(t)} \in \mathbb{R}^{C\times 1}$ is calculated element-wise with the softmax activation function E as shown in equation (3-22). The trainable parameters for the remaining useful life class target vector of the classification output layer is its weight matrix $W_{i,j}^{[n]} \in \mathbb{R}^{C\times I}$ and bias vector $b_i^{[n]} \in \mathbb{R}^{C\times 1}$.

$$RUL_{i,CN}^{(t)} = E\left(\sum_j W_{i,j}^{[n]} h_j^{(t),[n-1]} + b_i^{[n]}\right) \tag{3-22}$$

3.4.7 Regression Output Layer

The regression output layer is the last layer in all the regression model architectures. The size of the trainable parameters for the regression output layer is determined by the length of the output vector from the previous layer I and the dimension of the remaining useful life value target vector, which is equal to 1.

The input vector of the regression output layer is the output vector from the previous layer for the current time step $h_j^{(t),[n-1]} \in \mathbb{R}^{I\times 1}$. The remaining useful life value target vector for the current time step $RUL_{i,RN}^{(t)} \in \mathbb{R}^{1\times 1}$ is calculated element-wise with the linear activation function L as shown in equation (3-23). The trainable parameters for the remaining useful life value target vector of the regression output layer is its weight matrix $W_{i,j}^{[n]} \in \mathbb{R}^{1\times I}$ and bias vector $b_i^{[n]} \in \mathbb{R}^{1\times 1}$.

$$RUL_{i,RN}^{(t)} = L\left(\sum_j W_{i,j}^{[n]} h_j^{(t),[n-1]} + b_i^{[n]}\right) \tag{3-23}$$

3.5 Model Architectures

This section describes the proposed classification and regression model architectures from section 3.2.2 and section 3.2.3 respectively. It presents how the trained classification model architectures calculate the remaining useful life class target vector for the current time step $RUL_{i,CN}^{(t)} \in \mathbb{R}^{C\times 1}$ from the condition monitoring measurement input vector for the current time step $CM_i^{(t)} \in \mathbb{R}^{D\times 1}$. This section also presents how the trained regression model architectures calculate the remaining useful life value target vector for the current time step $RUL_{i,RN}^{(t)} \in \mathbb{R}^{1\times 1}$ from the condition monitoring measurement input vector for the current time step $CM_i^{(t)} \in \mathbb{R}^{D\times 1}$. It is again important to point out that only the last layer is different between the classification and regression model architectures. The classification and regression model architectures were however trained completely independent from each other with different loss functions on the training set of each data set.

3.5.1 Feedforward Neural Network (FNN) Architectures

The first layer of the FNN regression and classification model architectures is an input layer and is calculated as shown in equation (3-24).

$$h_i^{(t),[1]} = CM_i^{(t)} \tag{3-24}$$

The second layer of the FNN regression and classification model architectures is an fully-connected hidden layer and is calculated as shown in equation (3-25).

$$h_i^{(t),[2]} = T\left(\sum_j W_{i,j}^{[2]} h_j^{(t),[1]} + b_i^{[2]}\right) \tag{3-25}$$

The third layer of the FNN regression and classification model architectures is another fully-connected hidden layer and is calculated as shown in equation (3-26).

$$h_i^{(t),[3]} = T\left(\sum_j W_{i,j}^{[3]} h_j^{(t),[2]} + b_i^{[3]}\right) \tag{3-26}$$

The fourth layer of the FNN regression and classification model architectures is another fully-connected hidden layer and is calculated as shown in equation (3-27).

$$h_i^{(t),[4]} = T\left(\sum_j W_{i,j}^{[4]} h_j^{(t),[3]} + b_i^{[4]}\right) \tag{3-27}$$

The fifth layer of the FNN classification model architecture is an classification output layer and is calculated as shown in equation (3-28).

$$RUL_{i,CN}^{(t)} = E\left(\sum_j W_{i,j}^{[5]} h_j^{(t),[4]} + b_i^{[5]}\right) \tag{3-28}$$

The fifth layer of the FNN regression model architecture is an regression output layer and is calculated as shown in equation (3-29).

$$RUL_{i,RN}^{(t)} = L\left(\sum_j W_{i,j}^{[5]} h_j^{(t),[4]} + b_i^{[5]}\right) \tag{3-29}$$

3.5.2 Simple Recurrent Neural Network (S-RNN) Architectures

The first layer of the S-RNN regression and classification model architectures is an input layer and is calculated as shown in equation (3-30).

$$h_i^{(t),[1]} = CM_i^{(t)} \tag{3-30}$$

The second layer of the S-RNN regression and classification model architectures is an fully-connected hidden layer and is calculated as shown in equation (3-31).

$$h_i^{(t),[2]} = T\left(\sum_j W_{i,j}^{[2]} h_j^{(t),[1]} + b_i^{[2]}\right) \tag{3-31}$$

The third layer of the S-RNN regression and classification model architectures is an simple recurrent hidden layer and is calculated as shown in equation (3-32).

$$h_i^{(t),[3]} = T\left(\sum_j W_{i,j}^{[3]} h_j^{(t),[2]} + \sum_j R_{i,j}^{[3]} h_j^{(t-1),[3]} + b_i^{[3]}\right) \tag{3-32}$$

The fourth layer of the S-RNN regression and classification model architectures is an fully-connected hidden layer and is calculated as shown in equation (3-33).

$$h_i^{(t),[4]} = T\left(\sum_j W_{i,j}^{[4]} h_j^{(t),[3]} + b_i^{[4]}\right) \tag{3-33}$$

The fifth layer of the S-RNN classification model architecture is an classification output layer and is calculated as shown in equation (3-34).

$$RUL_{i,CN}^{(t)} = E\left(\sum_j W_{i,j}^{[5]} h_j^{(t),[4]} + b_i^{[5]}\right) \tag{3-34}$$

The fifth layer of the S-RNN regression model architecture is an regression output layer and is calculated as shown in equation (3-35).

$$RUL_{i,RN}^{(t)} = L\left(\sum_j W_{i,j}^{[5]} h_j^{(t),[4]} + b_i^{[5]}\right) \tag{3-35}$$

3.5.3 Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) Architectures

The first layer of LSTM-RNN regression and classification model architectures is an input layer and is calculated as shown in equation (3-36).

$$h_i^{(t),[1]} = CM_i^{(t)} \tag{3-36}$$

The second layer of the LSTM-RNN regression and classification model architectures is an fully-connected hidden layer and is calculated as shown in equation (3-37).

$$h_i^{(t),[2]} = T\left(\sum_j W_{i,j}^{[2]} h_j^{(t),[1]} + b_i^{[2]}\right) \tag{3-37}$$

The third layer of the LSTM-RNN regression and classification model architectures is an LSTM hidden layer and is calculated as shown in equations (3-38)–(3-43).

$$u_i^{(t),[3]} = S\left(\sum_j W_{i,j}^{u,[3]} h_j^{(t),[2]} + \sum_j R_{i,j}^{u,[3]} h_j^{(t-1),[3]} + b_i^{u,[3]}\right) \tag{3-38}$$

$$f_i^{(t),[3]} = S\left(\sum_j W_{i,j}^{f,[3]} h_j^{(t),[2]} + \sum_j R_{i,j}^{f,[3]} h_j^{(t-1),[3]} + b_i^{f,[3]}\right) \tag{3-39}$$

$$s_i^{(t),[3]} = T\left(\sum_j W_{i,j}^{s,[3]} h_j^{(t),[2]} + \sum_j R_{i,j}^{s,[3]} h_j^{(t-1),[3]} + b_i^{s,[3]}\right) \tag{3-40}$$

$$c_i^{(t),[3]} = u_i^{(t),[3]} s_i^{(t),[3]} + f_i^{(t),[3]} c_i^{(t-1),[3]} \tag{3-41}$$

$$o_i^{(t),[3]} = S\left(\sum_j W_{i,j}^{o,[3]} h_j^{(t),[2]} + \sum_j R_{i,j}^{o,[3]} h_j^{(t-1),[3]} + b_i^{o,[3]}\right) \tag{3-42}$$

$$h_i^{(t),[3]} = o_i^{(t),[3]}\, T\left(c_i^{(t),[3]}\right) \tag{3-43}$$

The fourth layer of the LSTM-RNN regression and classification model architectures is an fully-connected hidden layer and is calculated as shown in equation (3-44).

$$h_i^{(t),[4]} = T\left(\sum_j W_{i,j}^{[4]} h_j^{(t),[3]} + b_i^{[4]}\right) \tag{3-44}$$

The fifth layer of the LSTM-RNN classification model architecture is an classification output layer and is calculated as shown in equation (3-45).

$$RUL_{i,CN}^{(t)} = E\left(\sum_j W_{i,j}^{[5]} h_j^{(t),[4]} + b_i^{[5]}\right) \tag{3-45}$$

The fifth layer of the LSTM-RNN regression model architecture is an regression output layer and is calculated as shown in equation (3-46).

$$RUL_{i,RN}^{(t)} = L\left(\sum_j W_{i,j}^{[5]} h_j^{(t),[4]} + b_i^{[5]}\right) \tag{3-46}$$

3.5.4 Gated Recurrent Unit Recurrent Neural Network (GRU-RNN) Architectures

The first layer of the GRU-RNN regression and classification model architectures is an input layer and is calculated as shown in equation (3-47).

$$h_i^{(t),[1]} = CM_i^{(t)} \tag{3-47}$$

The second layer of the GRU-RNN regression and classification model architectures is an fully-connected hidden layer and is calculated as shown in equation (3-48).

$$h_i^{(t),[2]} = T\left(\sum_j W_{i,j}^{[2]} h_j^{(t),[1]} + b_i^{[2]}\right) \tag{3-48}$$

The third layer of the GRU-RNN regression and classification model architectures is an GRU hidden layer and is calculated as shown in equations (3-49)–(3-53).

$$u_i^{(t),[3]} = S\left(\sum_j W_{i,j}^{u,[3]} h_j^{(t),[2]} + \sum_j R_{i,j}^{u,[3]} h_j^{(t-1),[3]} + b_i^{u,[3]}\right) \tag{3-49}$$

$$r_i^{(t),[3]} = S\left(\sum_j W_{i,j}^{r,[3]} h_j^{(t),[2]} + \sum_j R_{i,j}^{r,[3]} h_j^{(t-1),[3]} + b_i^{r,[3]}\right) \tag{3-50}$$

$$s_i^{(t),[3]} = T\left(\sum_j W_{i,j}^{s,[3]} h_j^{(t),[2]} + \sum_j R_{i,j}^{s,[3]} r_j^{(t),[3]} h_j^{(t-1),[3]} + b_i^{s,[3]}\right) \tag{3-51}$$

$$c_i^{(t),[3]} = u_i^{(t),[3]} s_i^{(t),[3]} + \left(1 - u_i^{(t),[3]}\right) c_i^{(t-1),[3]} \tag{3-52}$$

$$h_i^{(t),[3]} = c_i^{(t),[3]} \tag{3-53}$$

The fourth layer of the GRU-RNN regression and classification model architectures is an fully-connected hidden layer and is calculated as shown in equation (3-54).

$$h_i^{(t),[4]} = T\left(\sum_j W_{i,j}^{[4]} h_j^{(t),[3]} + b_i^{[4]}\right) \tag{3-54}$$

The fifth layer of the GRU-RNN classification model architecture is an classification output layer and is calculated as shown in equation (3-55).

$$RUL_{i,CN}^{(t)} = E\left(\sum_j W_{i,j}^{[5]} h_j^{(t),[4]} + b_i^{[5]}\right) \tag{3-55}$$

The fifth layer of the GRU-RNN regression model architecture is an regression output layer and is calculated as shown in equation (3-56).

$$RUL_{i,RN}^{(t)} = L\left(\sum_j W_{i,j}^{[5]} h_j^{(t),[4]} + b_i^{[5]}\right) \tag{3-56}$$

It was found that adding fully-connected hidden layers before and after the simple recurrent, LSTM and GRU hidden layers in the recurrent neural network architectures resulted in a significant increase in model training and testing performance, when compared to not adding fully-connected hidden layers before and after these layers. It was also found that stacking numerous LSTM or GRU hidden layers did not drastically improve the model testing performance for the data sets, and in some cases even reduced the model testing performance.

It is very important to point out that the LSTM-RNN and GRU-RNN architectures were the main focus of this work, as only these architectures were able to manage the vanishing and exploding gradient problem (Hochreiter & Schmidhuber, 1997) during model training with numerous gating operations. This drastically increased the capacity of the LSTM-RNN and GRU-RNN architectures to model long-term sequence information from previous time steps. The FNN architectures cannot model sequence information and the S-RNN architectures cannot model long-term sequence information. The FNN and S-RNN architectures were therefore only investigated to demonstrate the importance of modeling long-term sequence information in condition monitoring measurements for degradation modeling and effective data-driven prognostics.

3.6 Model Training and Testing

This section describes how the proposed model architectures were trained and tested on the investigated data sets.

3.6.1 Preprocessing

The training and testing set input feature values (condition monitoring measurements) for each data set were preprocessed with min-max normalization between -1 and 1, based on the minimum and maximum input feature values in the training set (Goodfellow, et al., 2016). Min-max normalization between -1 and 1 was found to significantly improve the model training and testing performance when compared to no normalization, standard normalization and min-max normalization between 0 and 1.

3.6.2 Model Parameter Initialization Strategies

The uniform random Glorot initialization strategy (Glorot & Bengio, 2010) is used to initialize the weight matrices for all the model architectures. The orthogonal random initialization strategy (Saxe, et al., 2014) with a gain of one is used to initialize the recurrent weight matrices for all the model architectures. The zeros initialization strategy (Goodfellow, et al., 2016) is used to initialize the bias vectors for all the model architectures. These model parameter initialization strategies are generally considered as best practice for the proposed model architectures (Chollet, 2015).

3.6.3 Training, Validation and Testing Sets

The investigated data sets each include a training set and testing set. The training set represents historical (previously seen) assets with condition monitoring sensor measurements that were run to failure and are used to train the classification and regression model architectures. The testing set represents future (completely unseen) assets with condition monitoring sensor measurements that were run to failure and are used to test the classification and regression model architectures. An individual training or testing set example refers to an individual asset.

The training set for each data set includes training set input examples $X = \{x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(N)}\}$ and corresponding training set classification target examples $Y_{CN} = \{y_{CN}^{(1)}, y_{CN}^{(2)}, y_{CN}^{(3)}, \dots, y_{CN}^{(N)}\}$ for the classification model architectures, and training set regression target examples $Y_{RN} = \{y_{RN}^{(1)}, y_{RN}^{(2)}, y_{RN}^{(3)}, \dots, y_{RN}^{(N)}\}$ for the regression model architectures. N is the number of training set examples. A training set input example $x^{(1)}$ consists of a sequence of condition monitoring measurement input vectors $\left(CM_i^{(t)}, CM_i^{(t+1)}, CM_i^{(t+2)}, \dots, CM_i^{(t+M-1)}\right)$. A corresponding training set classification target example $y_{CN}^{(1)}$ consists of a sequence of remaining useful life class target vectors $\left(RUL_{i,CN}^{(t)}, RUL_{i,CN}^{(t+1)}, RUL_{i,CN}^{(t+2)}, \dots, RUL_{i,CN}^{(t+M-1)}\right)$. A corresponding training set regression target example $y_{RN}^{(1)}$ consists of a sequence of remaining useful life value target vectors $\left(RUL_{i,RN}^{(t)}, RUL_{i,RN}^{(t+1)}, RUL_{i,RN}^{(t+2)}, \dots, RUL_{i,RN}^{(t+M-1)}\right)$. M is the variable number of time steps for the individual training set example.

Similar to the training set, the testing set for each data set includes testing set input examples $\hat{X} = \{\hat{x}^{(1)}, \hat{x}^{(2)}, \hat{x}^{(3)}, \dots, \hat{x}^{(N)}\}$ and corresponding testing set classification target examples $\hat{Y}_{CN} = \{\hat{y}_{CN}^{(1)}, \hat{y}_{CN}^{(2)}, \hat{y}_{CN}^{(3)}, \dots, \hat{y}_{CN}^{(N)}\}$ for the classification model architectures, and testing set regression target examples $\hat{Y}_{RN} = \{\hat{y}_{RN}^{(1)}, \hat{y}_{RN}^{(2)}, \hat{y}_{RN}^{(3)}, \dots, \hat{y}_{RN}^{(N)}\}$ for the regression model architectures. N is the number of testing set examples. A testing set input example $\hat{x}^{(1)}$ similarly consists of a sequence of condition monitoring measurement input vectors $\left(CM_i^{(t)}, CM_i^{(t+1)}, CM_i^{(t+2)}, \dots, CM_i^{(t+M-1)}\right)$. A corresponding testing set classification target example $\hat{y}_{CN}^{(1)}$ similarly consists of a sequence of remaining useful life class target vectors $\left(RUL_{i,CN}^{(t)}, RUL_{i,CN}^{(t+1)}, RUL_{i,CN}^{(t+2)}, \dots, RUL_{i,CN}^{(t+M-1)}\right)$. A corresponding testing set regression target example $\hat{y}_{RN}^{(1)}$ similarly consists of a sequence of remaining useful life value target vectors $\left(RUL_{i,RN}^{(t)}, RUL_{i,RN}^{(t+1)}, RUL_{i,RN}^{(t+2)}, \dots, RUL_{i,RN}^{(t+M-1)}\right)$. M is the variable number of time steps for the individual testing set example.

It is very important to point out that the testing set inputs and targets were completely unseen by the classification and regression model architectures during model training and were only used for model testing. The training set for each data set was however split into a new training and validation set with an 80% to 20% ratio. The validation set is then used as a trail testing set in order to tune the hyperparameters of the model architectures and prognostics strategies during model training, with the hope and assumption that prediction performance on the completely unseen testing set will be similar to that of the validation set. This is also known as simple hold-

out validation (Chollet, 2018). The validation set is also used for the early stopping regularization technique (Goodfellow, et al., 2016) that is explained in section 3.6.5.

3.6.4 Model Loss Functions

The recommended cross-entropy loss function is minimized in order to train the model parameters of classification model architectures on the training set of each data set (Bishop, 2006) (Goodfellow, et al., 2016). The cross-entropy loss function $\mathcal{L}_{CE}$ is defined as shown in equation (3-57) and is minimized in order to train the model parameters θ of a classification model architecture f on the training set input examples $X = \{x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(N)}\}$ and the corresponding training set classification target examples $Y_{CN} = \{y_{CN}^{(1)}, y_{CN}^{(2)}, y_{CN}^{(3)}, \dots, y_{CN}^{(N)}\}$ of a data set. N is the total number of training set examples, M is the variable number of time steps for each individual training set example, C is number of target classes and P is the total number of time steps for all the training set examples combined.

$$\mathcal{L}_{CE}(f(X,\theta), Y_{CN}) = -\frac{1}{P}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{C} y_{CN}^{(i,j,k)} \log\left(f(x,\theta)^{(i,j,k)}\right) \tag{3-57}$$

The recommended mean squared error loss function is minimized in order to train the model parameters of regression model architectures on the training set of each data set (Bishop, 2006) (Goodfellow, et al., 2016). The mean squared error loss function $\mathcal{L}_{MSE}$ is defined as shown in equation (3-58) and is minimized in order to train the model parameters θ of a regression model architecture f on the training set input examples $X = \{x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(N)}\}$ and the corresponding training set regression target examples $Y_{RN} = \{y_{RN}^{(1)}, y_{RN}^{(2)}, y_{RN}^{(3)}, \dots, y_{RN}^{(N)}\}$ of a data set. N is the total number of training set examples, M is the variable number of time steps for each individual training set example and P is the total number of time steps for all the training set examples combined.

$$\mathcal{L}_{MSE}(f(X,\theta), Y_{RN}) = \frac{1}{P}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(y_{RN}^{(i,j)} - f(x,\theta)^{(i,j)}\right)^2 \tag{3-58}$$

3.6.5 Model Training Algorithms

The popular, robust, new and effective Adam algorithm (Kingma & Ba, 2015) is used minimize the model loss functions and train the model parameters of the model architectures on the training set of each data set (Goodfellow, et al., 2016). The Adam (Adaptive Moments) algorithm is an adaptive learning rate optimization algorithm similar to the popular AdaGrad (Adaptive Gradient) algorithm (Duchi, et al., 2011) and RMSProp (Root Mean Square Propagation) algorithm (Hinton, et al., 2012). This means the Adam algorithm automatically adapts the learning rate for each

individual model parameter during model training, which drastically improves the model training and testing performance when compared with traditional algorithms. The recommended hyperparameters for adaptive learning rate optimization algorithms are also generally very robust for different data sets and model architectures and do generally not significantly benefit from further tuning. This makes them significantly more convenient to implement when compared with traditional algorithms that generally require extensive manual hyperparameter tuning.

The Adam algorithm is a Gradient Decent (Cauchy, 1847) based optimization algorithm. This means the Adam algorithm only uses gradient information in order to minimize the model loss functions and train the model parameters of the model architectures.

The Adam algorithm is also a mini-batch algorithm that can be used to train the model architectures on mini-batches of training set examples instead of the full-batch of training set examples, for increased computational efficiency. This is especially important when training the model architectures on very large future data sets. The model architectures were however trained on the full-batch of training set examples for the investigated data sets that were relatively small in a deep learning context.

The Adam algorithm is shown in equations (3-59)–(3-66) and can be interpreted as a combination between the very popular Momentum (Gradient Decent with Momentum) algorithm (Polyak, 1964) and RMSProp algorithm. The Adam algorithm incorporates the exponentially weighted accumulated gradients from previous iterations in order to calculate the biased first moment estimate. This is similar to the Momentum algorithm that also uses the exponentially weighted accumulated gradients from previous iterations in order to calculate the updates of the model parameters.

The Adam algorithm also incorporates the exponentially weighted accumulated squared gradients from previous iterations in order to calculate the biased second moment estimate. This is similar to the RMSProp algorithm that also uses the exponentially weighted accumulated squared gradients from previous iterations in order to calculate the updates of the model parameters. The Adam algorithm however also corrects the biased first and second moment estimates to account for their initialization at the origin (zero). The updates of the model parameters for the Adam algorithm are also calculated similarly to that of the RMSProp algorithm.

The Adam algorithm recommends the following constant hyperparameters for model training: global learning rate $\epsilon = 0.001$, exponential decay rate for the first moment estimate $\rho_1 = 0.9$, exponential decay rate for the second moment estimate $\rho_2 = 0.999$ and numerical stability constant $\delta = 10^{-8}$ (prevents zero division). These recommended constant parameters for model training were used to for all the model architectures and data sets.

The Adam algorithm requires the following parameters for initialization: initial model parameters $\theta^{(n)}$, initial biased first moment estimate $s^{(n)} = 0$, initial biased second moment estimate $r^{(n)} = 0$ and initial time step $t^{(n)} = 0$.

The Adam algorithm also requires the following for each training iteration: The model architecture f, model parameters θ, loss function $\mathcal{L}$, full-batch of training set example inputs $X = \{x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(N)}\}$ and corresponding targets $Y = \{y^{(1)}, y^{(2)}, y^{(3)}, \dots, y^{(N)}\}$, where N is the number of training set examples. Alternatively, mini-batches of training set examples for large data sets can also be used for each training iteration.

The Adam algorithm iteratively updates each individual model parameter θ for each training iteration n as follows:

The gradient $g^{(n)}$ is calculated as shown in equation (3-59). The partial derivative of the loss function with respect to the model parameter of the model architecture is derived, evaluated and averaged for all the training set examples (including all time steps).

$$g^{(n)} = \frac{1}{N}\sum_{i=1}^{N}\frac{\partial}{\partial\theta}\left(\mathcal{L}\left(f\left(x^{(i)}, \theta^{(n)}\right), y^{(i)}\right)\right) \tag{3-59}$$

The time step $t^{(n+1)}$ is updated as shown in equation (3-60).

$$t^{(n+1)} = t^{(n)} + 1 \tag{3-60}$$

The biased first moment estimate $s^{(n+1)}$ is updated as shown in equation (3-61).

$$s^{(n+1)} = \rho_1 s^{(n)} + (1 - \rho_1) g^{(n)} \tag{3-61}$$

The bias corrected first moment estimate $\hat{s}^{(n+1)}$ is calculated as shown in equation (3-62).

$$\hat{s}^{(n+1)} = \frac{s^{(n+1)}}{1 - \rho_1^{t^{(n+1)}}} \tag{3-62}$$

The biased second moment estimate $r^{(n+1)}$ is updated as shown in equation (3-63).

$$r^{(n+1)} = \rho_2 r^{(n)} + (1 - \rho_2) g^{(n)} g^{(n)} \tag{3-63}$$

The bias corrected second moment estimate $\hat{r}^{(n+1)}$ is calculated as shown in equation (3-64).

$$\hat{r}^{(n+1)} = \frac{r^{(n+1)}}{1 - \rho_2^{t^{(n+1)}}} \tag{3-64}$$

The model parameter update $\Delta\theta^{(n)}$ is calculated as shown in equation (3-65).

$$\Delta\theta^{(n)} = -\epsilon \frac{\hat{s}^{(n+1)}}{\sqrt{\hat{r}^{(n+1)}} + \delta} \tag{3-65}$$

The model parameter $\theta^{(n+1)}$ is then finally updated as shown in equation (3-66).

$$\theta^{(n+1)} = \theta^{(n)} + \Delta\theta^{(n)} \tag{3-66}$$

The Adam algorithm is then repeated for numerous training iterations until the stopping criterion is met. The stopping criterion was set to 10,000 training iterations for all the model architectures and data sets, as the deep learning models all converged long before 10,000 training iterations. It is important to point out that convergence in this context refers to when the value of the model loss function on the training set stops drastically decreasing and the model loss function on the validation set starts increasing.

The gradient calculations that are required in equation (3-59) of the Adam algorithm are very involved for the model architectures, especially for the recurrent neural networks that have to be unrolled over time (Goodfellow, et al., 2016). The model architectures were therefore trained on the data sets with the open-source (free) Tensorflow (Google Brain, 2016) and easy to use Keras (Chollet, 2015) application programming interfaces (APIs) in Python that could perform these required gradient calculations automatically with automatic differentiation (Baydin, et al., 2018).

The full-batch of training set and validation examples were post zero padded to be the same length of the longest training set or validation set example. This made it possible to parallelize the required gradient calculations, which drastically improved the training time of the deep learning models. The training time of 10,000 training iterations for the LSTM-RNN and GRU-RNN architectures were approximately 32 minutes for the general asset degradation data set and approximately 8 minutes for the turbofan engine degradation data set on a desktop PC with an Intel i7-6700 CPU, Nvidia GTX 1070 GPU and Samsung 850 EVO SSD.

The Adam algorithm was also compared with the popular Gradient Decent, Momentum, AdaGrad and RMSProp algorithms. The training set mean squared error loss function value for the GRU-RNN regression model architecture after each model training iteration with the Gradient Decent, Momentum, AdaGrad, RMSProp and Adam algorithms on the turbofan engine degradation data set are compared Figure 3-1. The validation set mean squared error loss function value for the GRU-RNN regression model architecture after each model training iteration with the Gradient Decent, Momentum, AdaGrad, RMSProp and Adam algorithms on the turbofan engine degradation data set are compared Figure 3-2. The following hyperparameters were used for the different model training algorithms: Gradient Decent with a manually tuned global learning rate $\epsilon = 0.0001$. Momentum with a manually tuned global learning rate $\epsilon = 0.0001$ and momentum parameter $\alpha = 0.9$. AdaGrad with a recommended global learning rate $\epsilon = 0.01$. RMSProp with a recommended global learning rate $\epsilon = 0.001$ and decay rate $\rho = 0.9$. Adam with a recommended global learning rate $\epsilon = 0.001$, exponential decay rate for the first moment estimate $\rho_1 = 0.9$, exponential decay rate for the second moment estimate $\rho_2 = 0.999$.

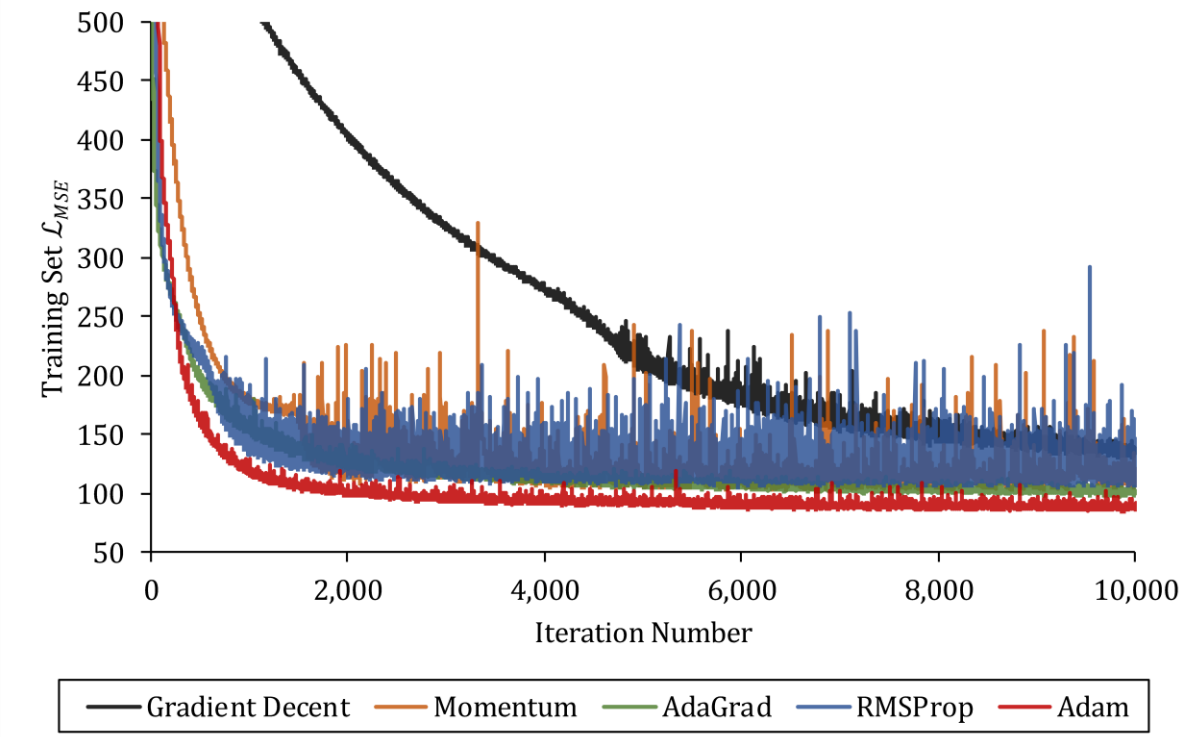

Figure 3-1: The training set mean squared error loss function value for the GRU-RNN regression model architecture after each model training iteration with the Gradient Decent, Momentum, AdaGrad, RMSProp and Adam algorithms on the turbofan engine degradation data set.

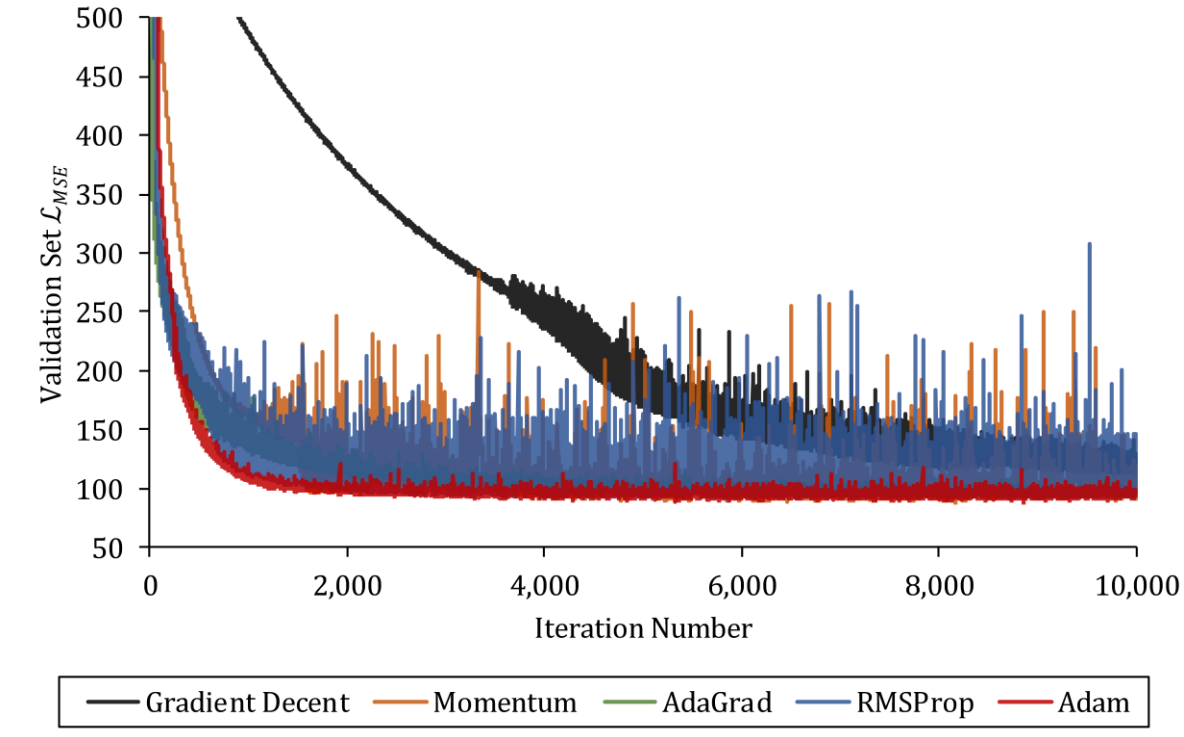

Figure 3-2: The validation set mean squared error loss function value for the GRU-RNN regression model architecture after each model training iteration with the Gradient Decent, Momentum, AdaGrad, RMSProp and Adam algorithms on the turbofan engine degradation data set.

From Figure 3-1 and Figure 3-2 it can be concluded that the Adam algorithm achieved the lowest mean squared error loss function values (best fit) for the GRU-RNN regression model architecture on the training set and validation set of the turbofan engine degradation data set. The Adam algorithm is also generally more stable, robust and converges in fewer training iterations when compared to the Gradient Decent, Momentum, AdaGrad and RMSProp algorithms. The Adam algorithm achieved similar results on all the other classification and regression model architectures and data sets. The recommended constant hyperparameters for the Adam algorithm were also found to be robust for all the data sets and model architectures and did not significantly benefit from further tuning.

3.6.6 Model Regularization Techniques

The biggest challenge for effectively training deep learning models that perform well on the completely unseen testing set is overfitting (Bishop, 2006). Overfitting is when a deep learning model can very accurately predict previously seen training set targets from the training set inputs, but cannot generalize and accurately predict the completely unseen testing set targets from the testing set inputs.

The model architectures were therefore regularized with a combination of the popular early stopping, weight decay and dropout regularization techniques in order to reduce overfitting.

The early stopping regularization technique (Goodfellow, et al., 2016) monitors the value of the model loss function on the training set and validation set after each model training iteration. The value of the model loss function on the training set will generally decrease with the number of model training iterations. The value of the model loss function on the validation set will however generally initially decrease and then start to increase with the number of model training iterations, as the model starts to overfit on the training set. The early stopping regularization technique then uses the setting of the model parameters when the value of the model loss function on the validation set was at a minimum, with the hope and assumption that the model loss function on the completely unseen testing set would also be at a minimum and would therefore result in a model with increased testing and prediction performance.

The weight decay regularization technique (also known as L^2 regularization, ridge regression and Tikhonov regularization) (Goodfellow, et al., 2016) adds the squared norm of a model parameter $\|\theta\|^2$ multiplied by a weight decay parameter λ to the model loss function during model training. This constrains and drives the model parameter θ towards the origin (zero) during model training and therefore reduces overfitting on the training set. The weight decay parameter λ is however also a hyperparameter that has to be tuned. The weight decay regularization technique was applied to all the model parameters of the model architectures. The weight decay parameter λ was however assumed to be the same for all the model parameters of a model architecture and

tuned on the validation set. The LSTM-RNN and GRU-RNN architectures were generally more prone to overfitting due to their increased sequence modeling capacity and therefore required higher weight decay parameters than the FNN and S-RNN architectures. The manually tuned and selected weight decay parameters λ for the data sets, prognostics strategies and model architectures is shown in Table 3-1.

Table 3-1: The manually tuned and selected weight decay parameters λ for the data sets, prognostics strategies and model architectures.

Data Set	Strategy	Architectures	λ
General Asset Degradation Data Set	Classification	FNN and S-RNN	0.000001
		LSTM-RNN and GRU-RNN	0.00001
	Regression	FNN and S-RNN	0.001
		LSTM-RNN and GRU-RNN	0.01
Turbofan Engine Degradation Data Set	Classification	FNN and S-RNN	0.0001
		LSTM-RNN and GRU-RNN	0.001
	Regression	FNN and S-RNN	0.1
		LSTM-RNN and GRU-RNN	1
Turbofan Engine Degradation Benchmarking Data Sets	Regression	FNN and S-RNN	0.1
		LSTM-RNN and GRU-RNN	1

The dropout regularization technique (Srivastava, et al., 2014) deactivates a random fraction τ of output vector elements for a hidden layer before a model training iteration and then reactivates the same random fraction of output vector elements after the model training iteration. This process is then repeated for each model training iteration. The model architecture is therefore slightly different during each model training iteration and can therefore be interpreted as a very efficient ensemble learning (model-averaging) technique. Traditional ensemble learning techniques generally require training numerous models independently on the training set and then averaging the predictions of the models on the testing set in order to improve model regularization and reduce overfitting. The dropout regularization technique therefore approximates these traditional ensemble learning techniques by changing the model architecture slightly during each training iteration, but is computationally drastically less expensive, since only one model needs to be trained. The dropout fraction τ for each hidden layer is however also a hyperparameter that has to be tuned. The dropout regularization technique was applied to all the hidden layers of the model architectures. The dropout fraction was assumed to be the same for all the hidden layers of a model architecture. The manually tuned and selected dropout fraction of $\tau = 0.2$ was applied to all the hidden layers of the model architectures.

The combination of these regularization strategies were found to drastically increase the prediction performance of all the model architectures on the validation and testing sets of the investigated data sets when compared with no regularization.

The prediction performance improvement of the early stopping, weight decay and dropout regularization techniques (and combinations thereof) over no regularization is demonstrated for the GRU-RNN regression model on the training set (including validation set) and testing set of the turbofan engine degradation data set. The mean squared error and mean absolute error between the target and GRU-RNN regression model predicted remaining useful life values 120 cycles before failure with different combinations of regularization techniques for all the training set and testing set examples is shown in Table 3-2.

Table 3-2: The mean squared error and mean absolute error between the target and GRU-RNN regression model predicted remaining useful life values 120 cycles before failure with different combinations of regularization techniques for all the training set and testing set examples.

Regularization Technique	**Training Set Mean Squared Error**	**Testing Set Mean Squared Error**	**Training Set Mean Absolute Error**	**Testing Set Mean Absolute Error**
No Regularization	3.6392	313.3261	1.0011	12.7686
Early Stopping	85.0043	143.3062	6.6942	8.7714
Dropout and Early Stopping	61.0528	110.3704	5.3521	7.3505
Weight Decay and Early Stopping	85.2189	79.0641	6.4801	6.3396
Dropout, Weight Decay and Early Stopping	62.7665	76.2654	5.5248	6.1194

From Table 3-2 it can be concluded that the combination of the early stopping, weight decay and dropout regularization techniques drastically improved the prediction performance of the GRU-RNN regression model on the completely unseen testing set of the turbofan engine degradation data set when compared with no regularization. The combination of the early stopping, weight decay and dropout regularization techniques achieved similar prediction performance improvements for all the other classification and regression model architectures and data sets.

3.6.7 Number of Selected Hidden Units

The number of selected hidden units H for the model parameters of the model architectures also had a significant influence on the model prediction performance on the validation and testing set of the investigated data sets. This is because the number of selected hidden units determines the size and capacity of the model parameters. The model architectures have limited modeling capacity when the number of selected hidden units is too low, but are more prone to overfitting when the number of selected hidden units is too high. The computational expense of training the model architectures also increased significantly as the number of selected hidden units increased. The number of selected hidden units was however manually tuned and selected as $H = 128$ units for all the hidden layers of the model architectures.

3.6.8 Model Testing

The prediction performance of the trained classification model architectures were compared by calculating and comparing the cross-entropy, accuracy and confusion matrix for the training set and testing set of the investigated data sets.

The cross-entropy CE is calculated for a trained classification model architecture f with trained model parameters θ on the testing set input examples $\hat{X} = \{\hat{x}^{(1)}, \hat{x}^{(2)}, \hat{x}^{(3)}, \dots, \hat{x}^{(N)}\}$ and corresponding testing set classification target examples $\hat{Y}_{CN} = \{\hat{y}_{CN}^{(1)}, \hat{y}_{CN}^{(2)}, \hat{y}_{CN}^{(3)}, \dots, \hat{y}_{CN}^{(N)}\}$ as shown in equation (3-67). N is the total number of testing set examples, M is the variable number of time steps for each individual testing set example, C is number of target classes and P is the total number of time steps for all the testing set examples combined. The cross-entropy is calculated similarly for the training set input examples $X = \{x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(N)}\}$ and corresponding training set classification target examples $Y_{CN} = \{y_{CN}^{(1)}, y_{CN}^{(2)}, y_{CN}^{(3)}, \dots, y_{CN}^{(N)}\}$.

$$CE\left(f\left(\hat{X}, \theta\right), \hat{Y}_{CN}\right) = -\frac{1}{P}\sum_{i=1}^{N}\sum_{j=1}^{M}\sum_{k=1}^{C} \hat{y}_{CN}^{(i,j,k)} \log\left(f(\hat{x}, \theta)^{(i,j,k)}\right) \tag{3-67}$$

The argmax function is used to convert the softmax predicted categorically encoded class vectors to predicted classes in order to calculate the accuracy and present the confusion matrix for the training set and testing set of each data set. Examples of the argmax function applied to the softmax predicted categorically encoded class vectors for a hypothetical data set with four possible target classes are shown below:

$$\text{argmax}\begin{bmatrix}0.94\\0.02\\0.01\\0.03\end{bmatrix} \rightarrow C_1 \qquad \text{argmax}\begin{bmatrix}0.04\\0.88\\0.02\\0.06\end{bmatrix} \rightarrow C_2 \qquad \text{argmax}\begin{bmatrix}0.02\\0.01\\0.94\\0.03\end{bmatrix} \rightarrow C_3 \qquad \text{argmax}\begin{bmatrix}0.04\\0.02\\0.06\\0.88\end{bmatrix} \rightarrow C_4$$

The accuracy is calculated for the training and testing set as the ratio between the total number of correctly predicted classes and the total number of correctly and incorrectly predicted classes. The confusion matrix presents the target classes versus the predicted classes for the number of correctly and incorrectly predicted classes (for each time step) in the training or testing set.

The prediction performance of the trained regression model architectures were compared by calculating and comparing the mean squared error and mean absolute error for the training set and testing set of the investigated data sets.

The mean squared error MSE is calculated for a trained regression model architecture f with trained model parameters θ on the testing set input examples $\hat{X} = \{\hat{x}^{(1)}, \hat{x}^{(2)}, \hat{x}^{(3)}, \dots, \hat{x}^{(N)}\}$ and corresponding testing set regression target examples $\hat{Y}_{RN} = \{\hat{y}_{RN}^{(1)}, \hat{y}_{RN}^{(2)}, \hat{y}_{RN}^{(3)}, \dots, \hat{y}_{RN}^{(N)}\}$ as shown in equation (3-68). N is the total number of testing set examples, M is the variable number of time steps for each individual testing set example and P is the total number of time steps for all the testing set examples combined.

$$MSE\left(f\left(\hat{X}, \theta\right), \hat{Y}_{RN}\right) = \frac{1}{P}\sum_{i=1}^{N}\sum_{j=1}^{M}\left(\hat{y}_{RN}^{(i,j)} - f(\hat{x}, \theta)^{(i,j)}\right)^2 \tag{3-68}$$

The mean absolute error MAE is calculated for a trained regression model architecture f with trained model parameters θ on the testing set input examples $\hat{X} = \{\hat{x}^{(1)}, \hat{x}^{(2)}, \hat{x}^{(3)}, \dots, \hat{x}^{(N)}\}$ and corresponding testing set regression target examples $\hat{Y}_{RN} = \{\hat{y}_{RN}^{(1)}, \hat{y}_{RN}^{(2)}, \hat{y}_{RN}^{(3)}, \dots, \hat{y}_{RN}^{(N)}\}$ as shown in equation (3-69).

$$MAE\left(f\left(\hat{X}, \theta\right), \hat{Y}_{RN}\right) = \frac{1}{P}\sum_{i=1}^{N}\sum_{j=1}^{M}\left|\hat{y}_{RN}^{(i,j)} - f(\hat{x}, \theta)^{(i,j)}\right| \tag{3-69}$$

The mean squared error and mean absolute error is calculated similarly for the training set input examples $X = \{x^{(1)}, x^{(2)}, x^{(3)}, \dots, x^{(N)}\}$ and corresponding training set regression target examples $Y_{RN} = \{y_{RN}^{(1)}, y_{RN}^{(2)}, y_{RN}^{(3)}, \dots, y_{RN}^{(N)}\}$.

It is obvious but important to point out that a higher accuracy and a lower cross-entropy, mean squared error and mean absolute error indicated better model testing and prediction performance.

3.6.9 Applied Threshold Selection Strategy

The value of the applied threshold for the prognostics classification and regression strategies is also a hyperparameter that has to be tuned for each data set. When the value of the applied threshold is too high the classification and regression models are difficult to train and do not generalize well between the training and validation set. However, when the value of the applied threshold is too low the classification and regression models are easier to train and generalize better, but are less useful and can only predict the remaining useful life classes and values close to failure. It is therefore important to tune the value of the applied threshold for each data set. The value of the applied threshold for each data set is manually tuned by training numerous classification and regression models with different applied thresholds. The cross-entropy (for classification models) or mean squared error (for regression models) is then recalculated for the time steps below the applied threshold for the numerous trained classification and regression models on the training and validation set. This is because the prediction performance for the time steps below the applied threshold is more important and representative of the model performance close to failure. The value of the applied threshold is then manually tuned and increased until the cross-entropy or mean squared error for the time steps below the applied threshold on the training and validation sets starts increasing drastically. The value of the applied threshold is also manually tuned and increased until the difference between the cross-entropy or mean squared error for the time steps below the applied threshold on training and validation sets starts diverging drastically. The applied threshold therefore improved the generalization ability of the classification and regression models between the training and validation sets, and hopefully the testing set.

4 Prognostics Classification Strategy Results

4.1 General Asset Degradation Data Set

This section presents the results of the prognostics classification strategy and classification deep learning model architectures applied on the general asset degradation data set.

4.1.1 Strategy and Model Description

The applied threshold for the prognostics classification strategy was tuned and selected as $AT = 600$ cycles for the general asset degradation data set. The selected remaining useful life definition and corresponding degradation level for the different classes of the prognostics classification strategy applied on the general asset degradation data set is shown in Table 4-1.

Table 4-1: The selected remaining useful life definition and corresponding degradation level for the different classes of the prognostics classification strategy applied on the general asset degradation data set.

Class	Degradation Level	Remaining Useful Life Definition
C1	Healthy and Light Degradation	RUL $>$ 600 Cycles
C2	Medium Degradation	400 Cycles $<$ RUL $\leq$ 600 Cycles
C3	Heavy Degradation	200 Cycles $<$ RUL $\leq$ 400 Cycles
C4	Extreme Degradation	RUL $\leq$ 200 Cycles

The FNN, S-RNN, LSTM-RNN and GRU-RNN classification model architectures were trained on the training set of the general asset degradation data set with the Adam algorithm and regularized with a combination of the early stopping, weight decay and dropout regularization techniques.

4.1.2 Training Set Examples

The objective of this section is to present and compare how accurately the trained FNN, S-RNN, LSTM-RNN and GRU-RNN classification models could predict the remaining useful life classes from the condition monitoring measurements for two randomly selected training set examples in the general asset degradation data set fully online. The two randomly selected training set examples represent two historical (previously seen) general assets that were run to failure.

It is important to point out that the trained FNN, S-RNN, LSTM-RNN and GRU-RNN classification models only used the condition monitoring measurements for the current time step (and previous time steps for the recurrent models) to predict the remaining useful life class for the current time step. The classification models therefore predicted the remaining useful life classes from the condition monitoring measurements fully online. The presented S1 condition monitoring measurements were normalized with min-max scaling between 0 and 1 for the entire training and testing set. This was done to effectively present the variation in condition monitoring measurements across all the training and testing set examples.

Training Set Example 1

The S1 condition monitoring measurements versus time for training set example 1 are shown in Figure 4-1. The target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes versus time for training set example 1 are shown in Figure 4-2, Figure 4-3, Figure 4-4 and Figure 4-5 respectively.

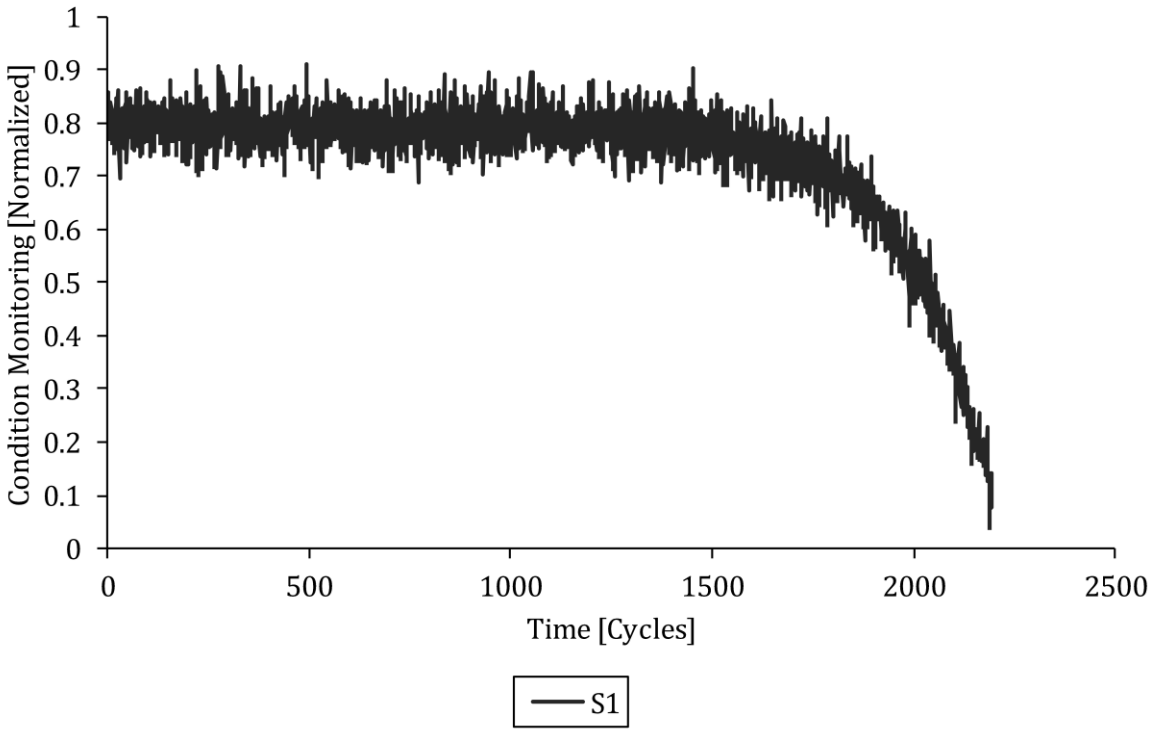

Figure 4-1: The S1 condition monitoring measurements versus time for training set example 1.

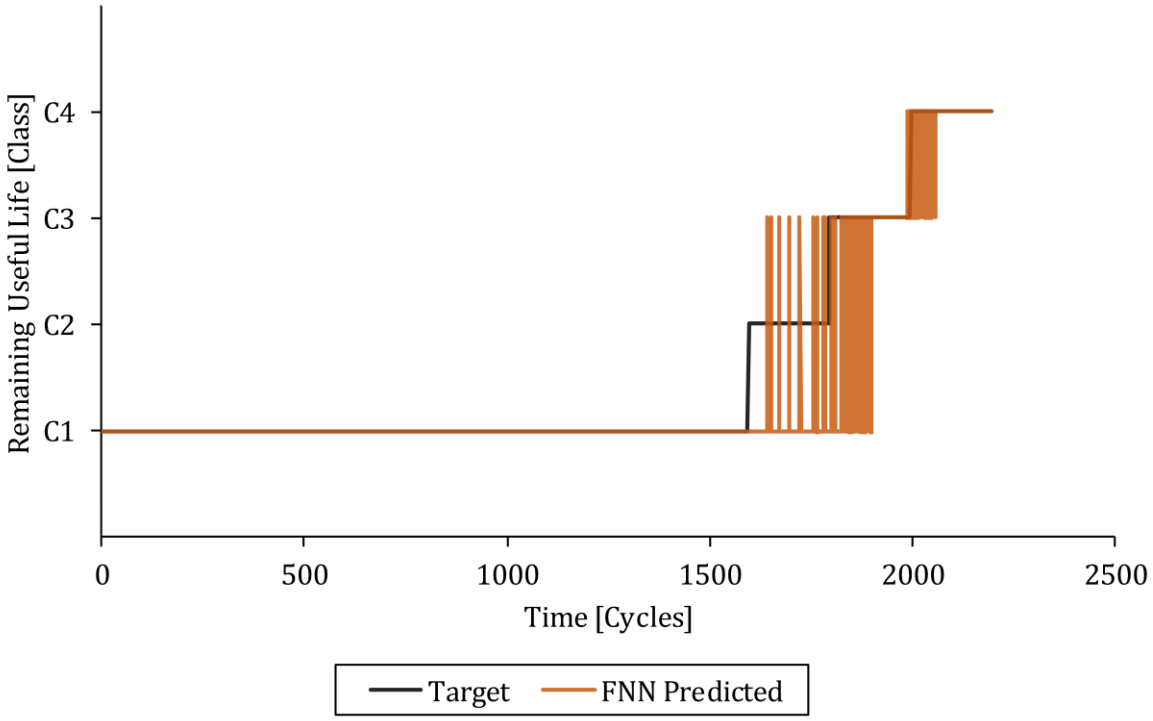

Figure 4-2: The target and FNN classification model predicted remaining useful life classes versus time for training set example 1.

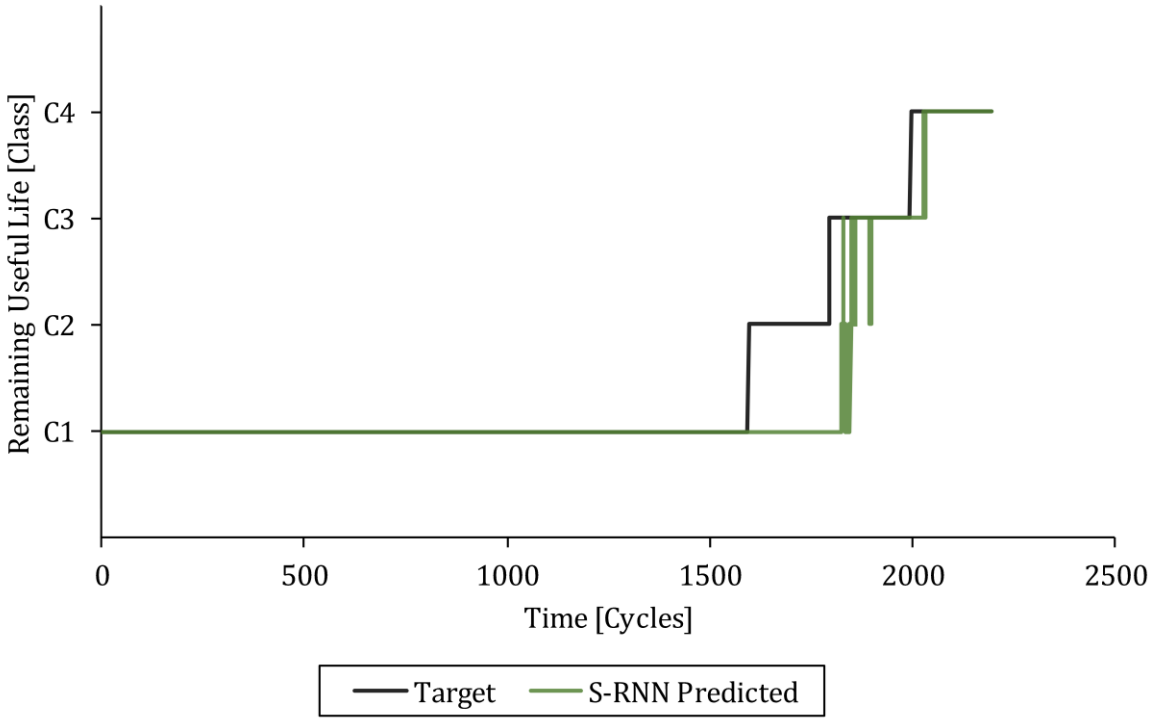

Figure 4-3: The target and S-RNN classification model predicted remaining useful life classes versus time for training set example 1.

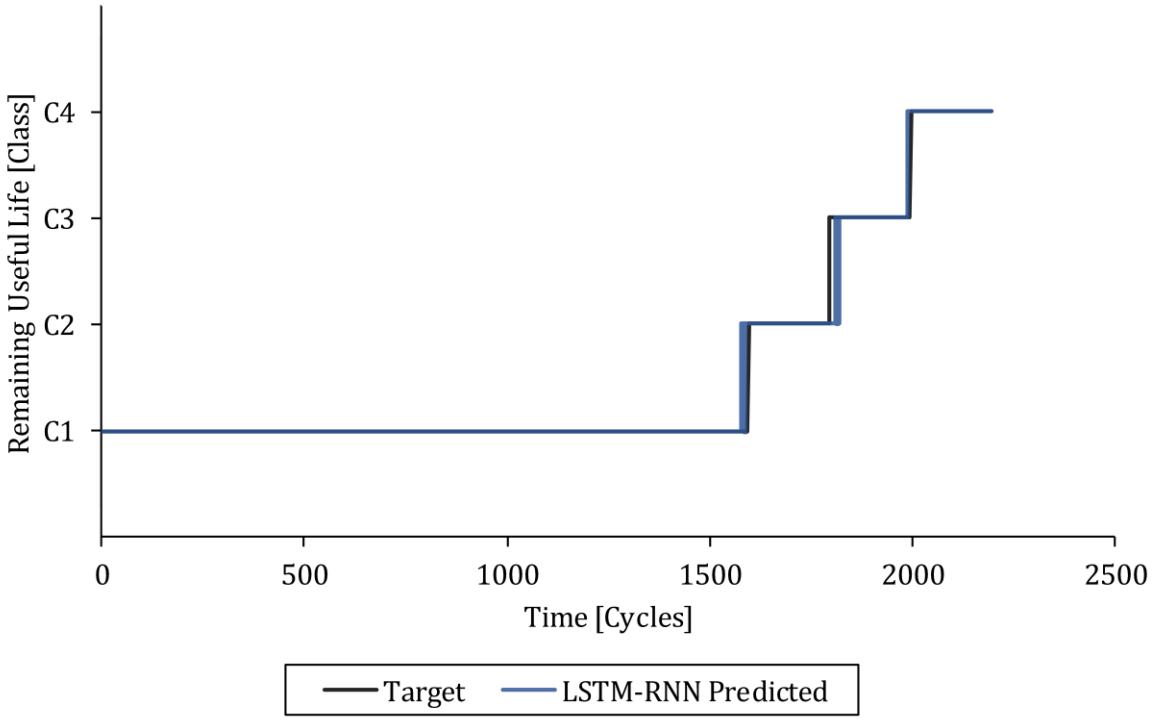

Figure 4-4: The target and LSTM-RNN classification model predicted remaining useful life classes versus time for training set example 1.

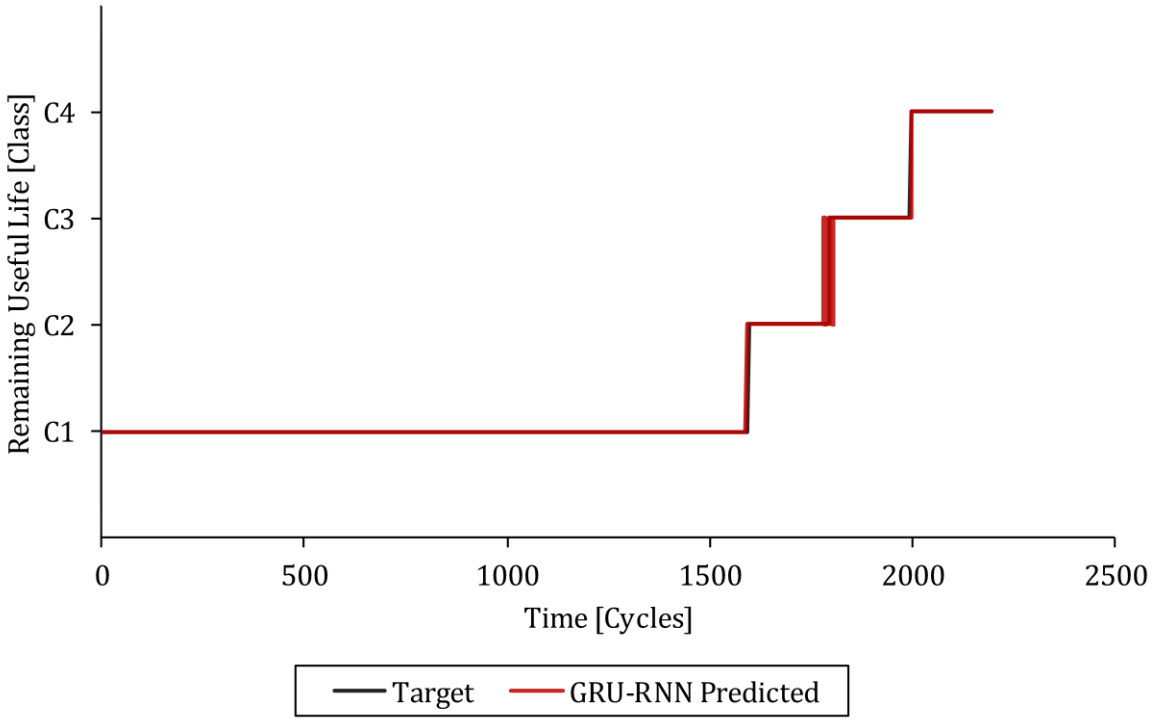

Figure 4-5: The target and GRU-RNN classification model predicted remaining useful life classes versus time for training set example 1.

Training Set Example 2

The S1 condition monitoring measurements versus time for training set example 2 are shown in Figure 4-6. The target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes versus time for training set example 2 are shown in Figure 4-7, Figure 4-8, Figure 4-9 and Figure 4-10 respectively.

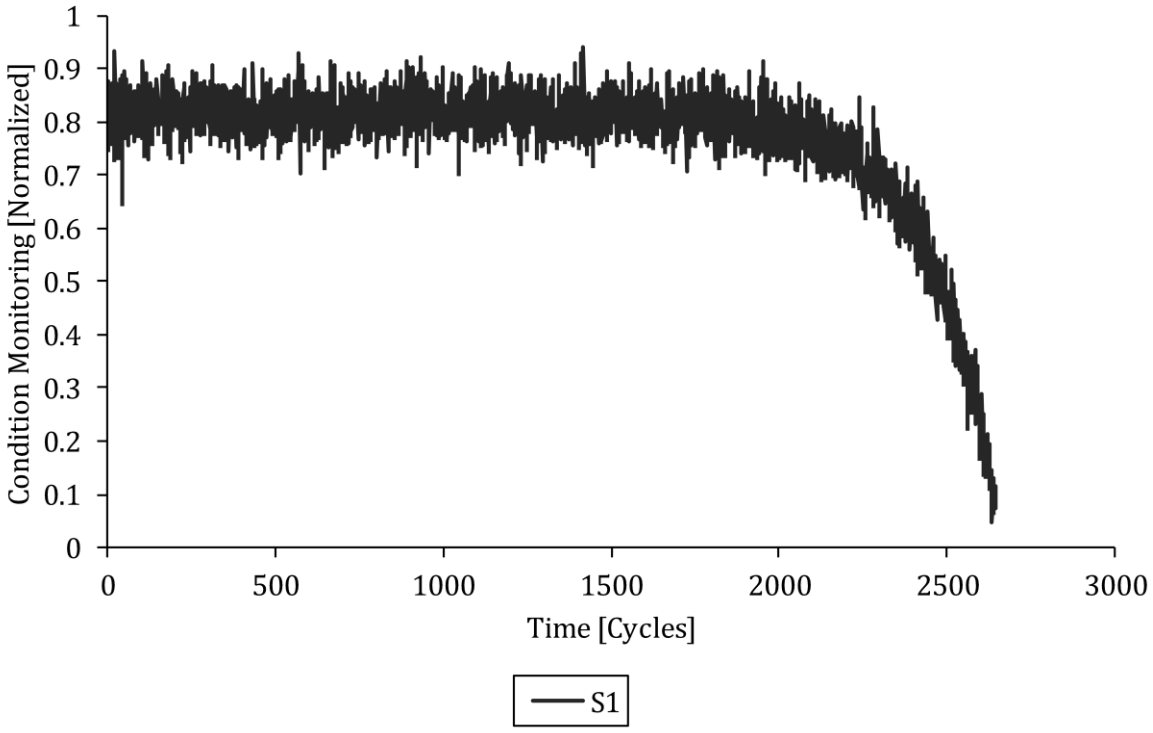

Figure 4-6: The S1 condition monitoring measurements versus time for training set example 2.

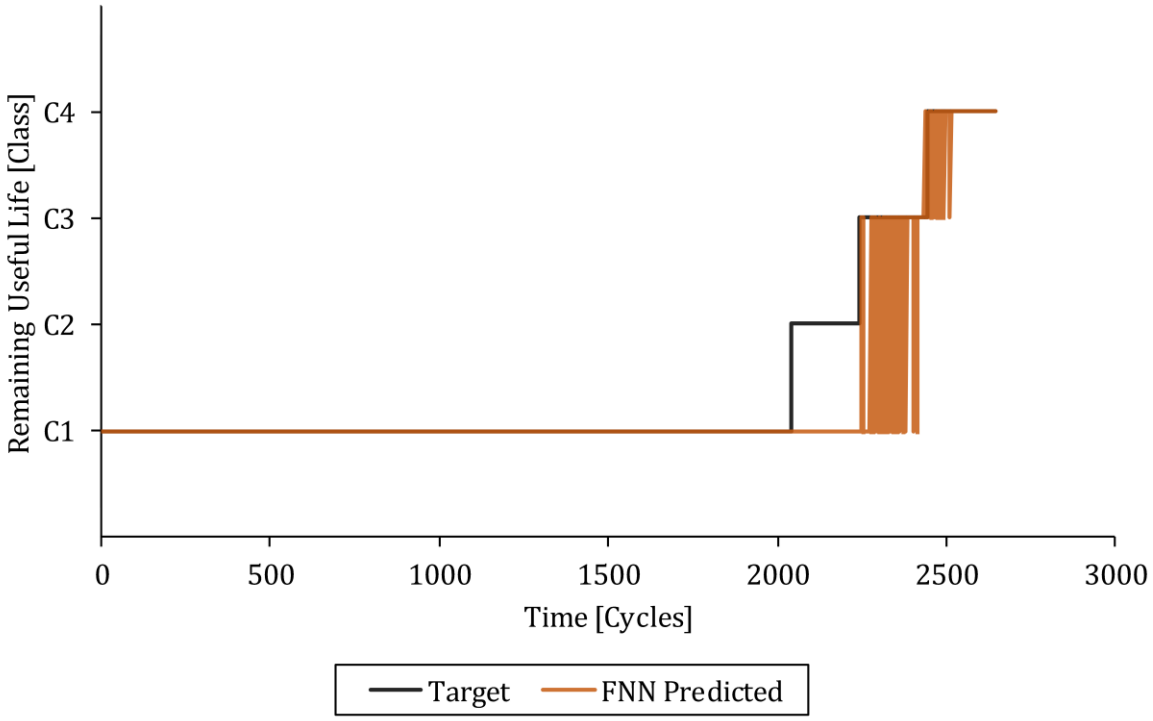

Figure 4-7: The target and FNN classification model predicted remaining useful life classes versus time for training set example 2.

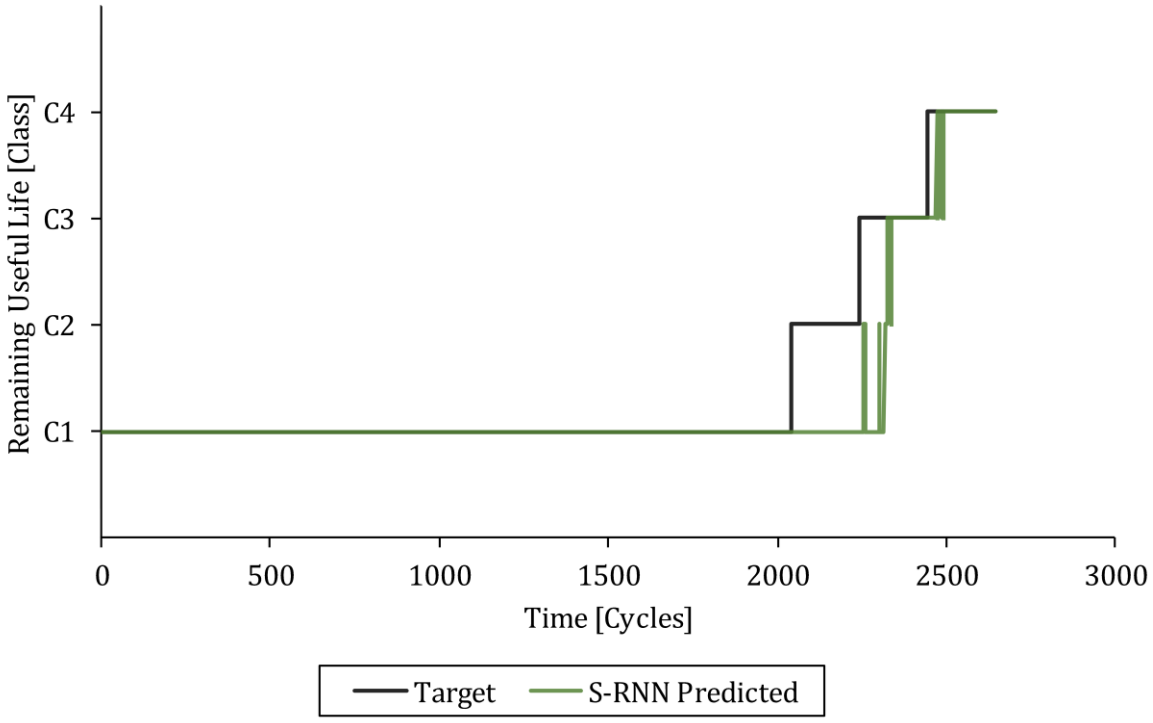

Figure 4-8: The target and S-RNN classification model predicted remaining useful life classes versus time for training set example 2.

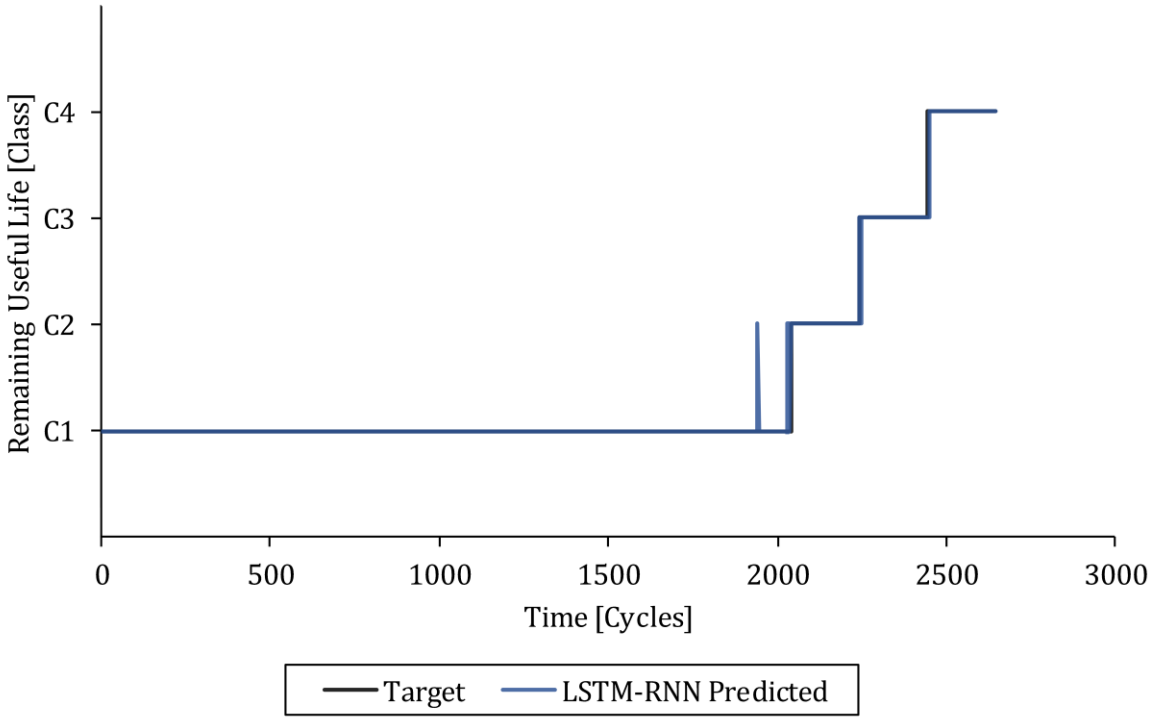

Figure 4-9: The target and LSTM-RNN classification model predicted remaining useful life classes versus time for training set example 2.

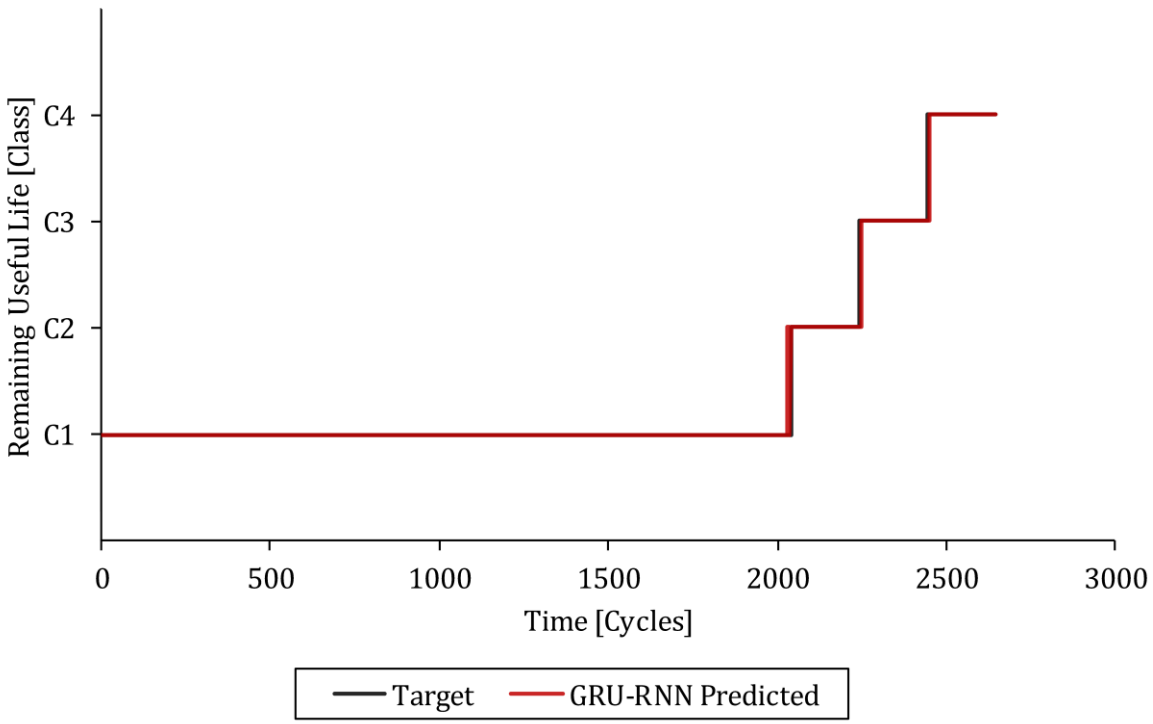

Figure 4-10: The target and GRU-RNN classification model predicted remaining useful life classes versus time for training set example 2.

4.1.3 Testing Set Examples

The objective of this section is to present and compare how accurately the trained FNN, S-RNN, LSTM-RNN and GRU-RNN classification models could predict the remaining useful life classes from the condition monitoring measurements for two randomly selected testing set examples in the general asset degradation data set fully online. The two randomly selected testing set examples represent two future (completely unseen) general assets that were run to failure.

Testing Set Example 1

The S1 condition monitoring measurements versus time for testing set example 1 are shown in Figure 4-11. The target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes versus time for testing set example 1 are shown in Figure 4-12, Figure 4-13, Figure 4-14 and Figure 4-15 respectively.

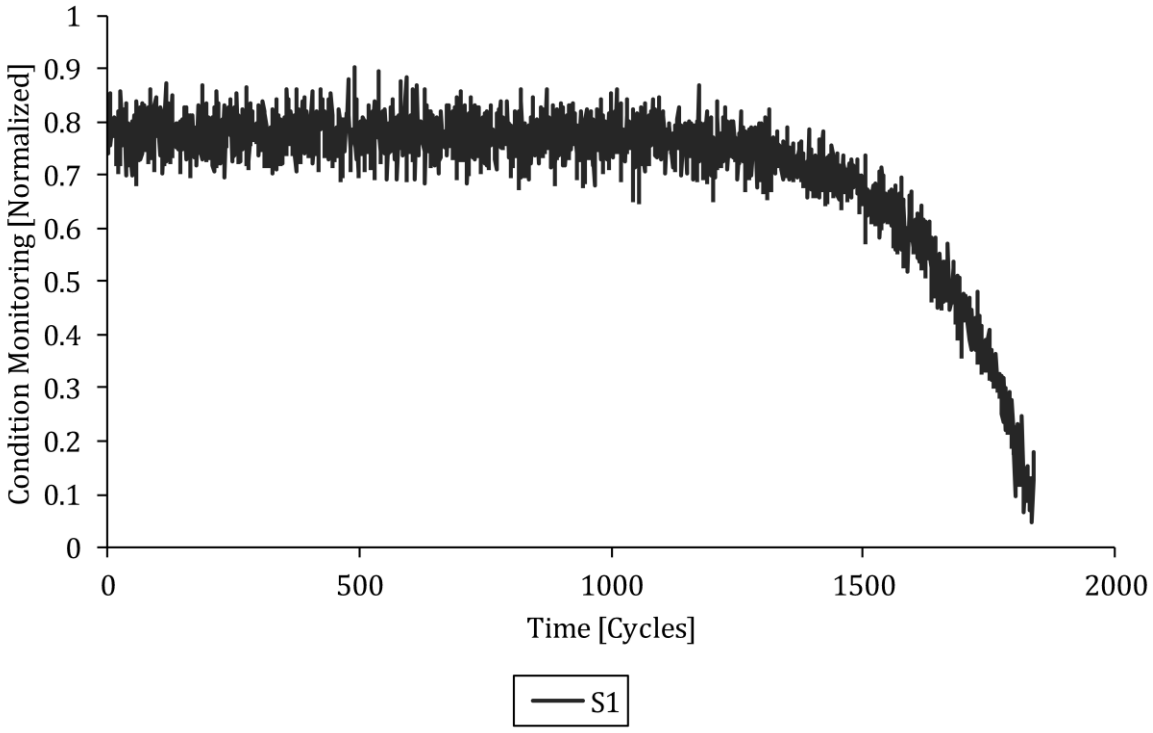

Figure 4-11: The S1 condition monitoring measurements versus time for testing set example 1.

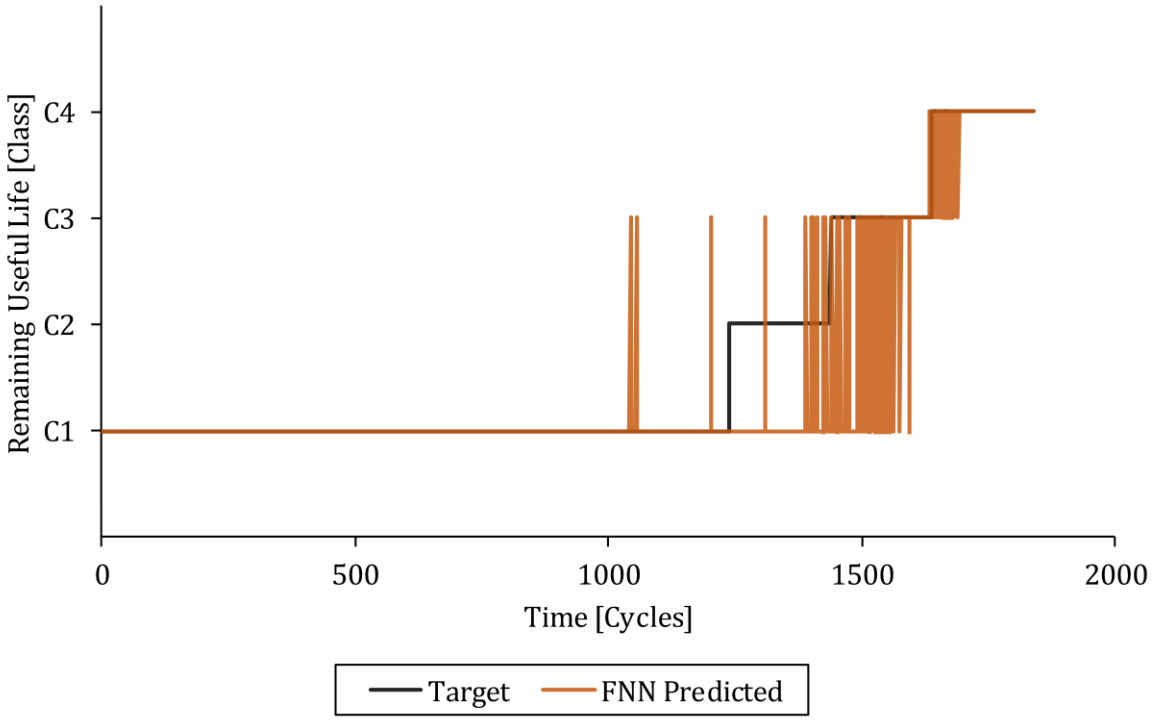

Figure 4-12: The target and FNN classification model predicted remaining useful life classes versus time for testing set example 1.

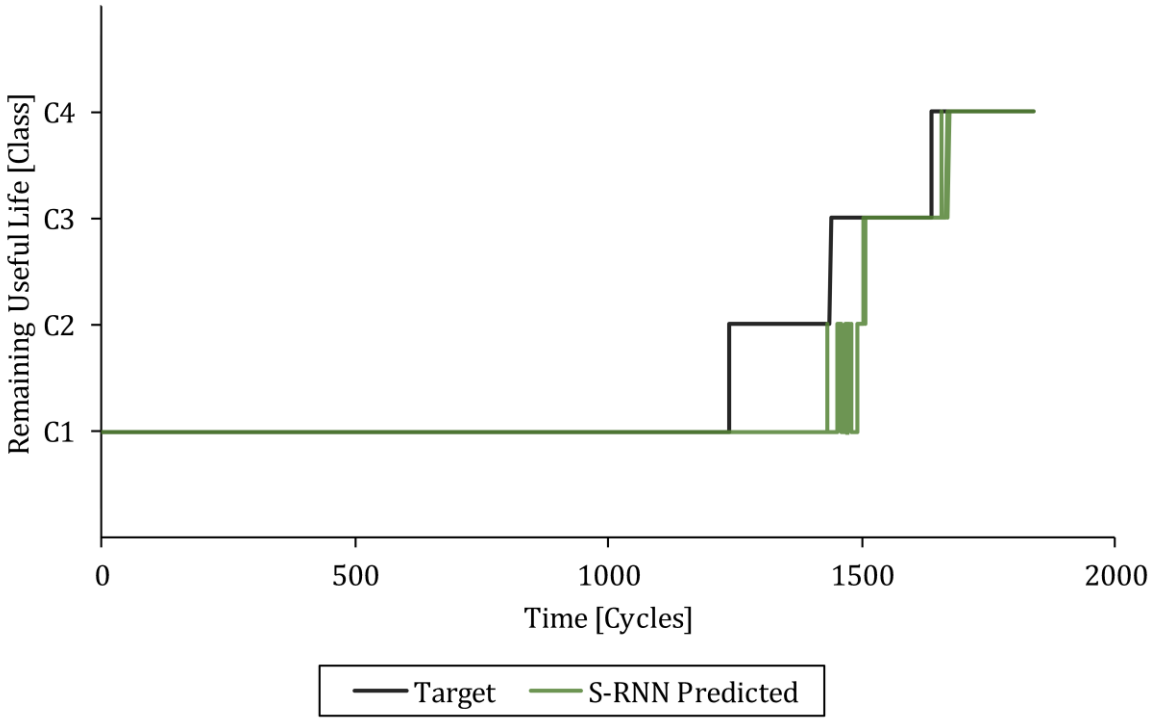

Figure 4-13: The target and S-RNN classification model predicted remaining useful life classes versus time for testing set example 1.

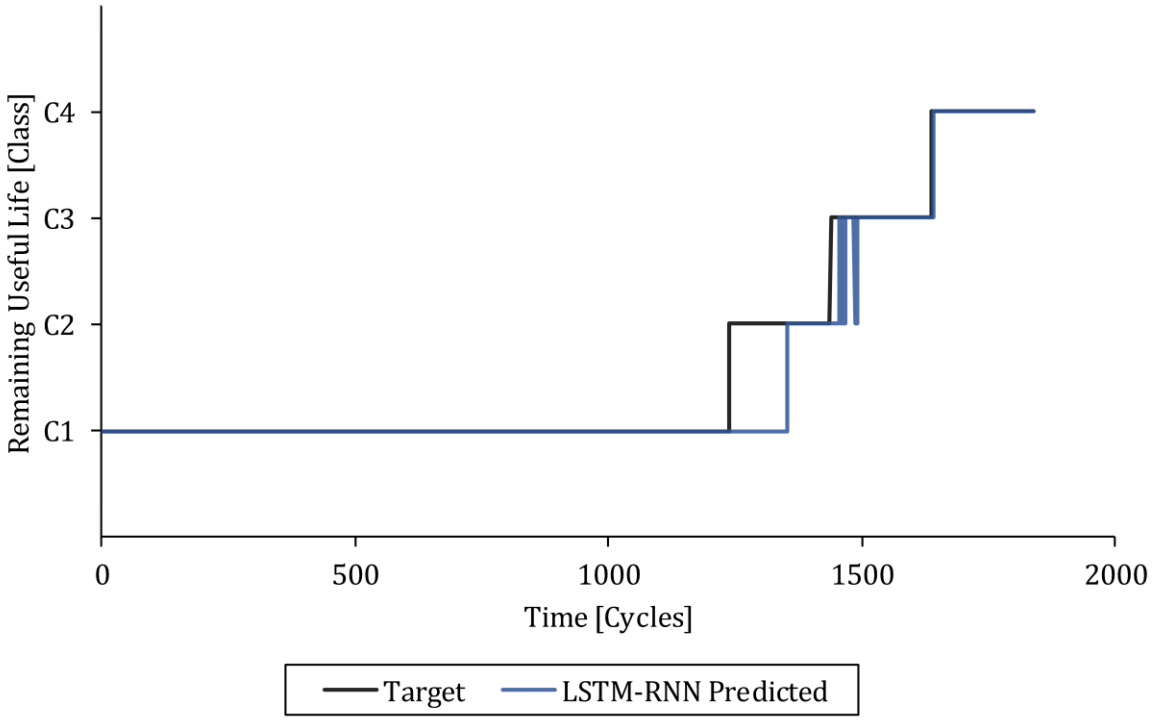

Figure 4-14: The target and LSTM-RNN classification model predicted remaining useful life classes versus time for testing set example 1.

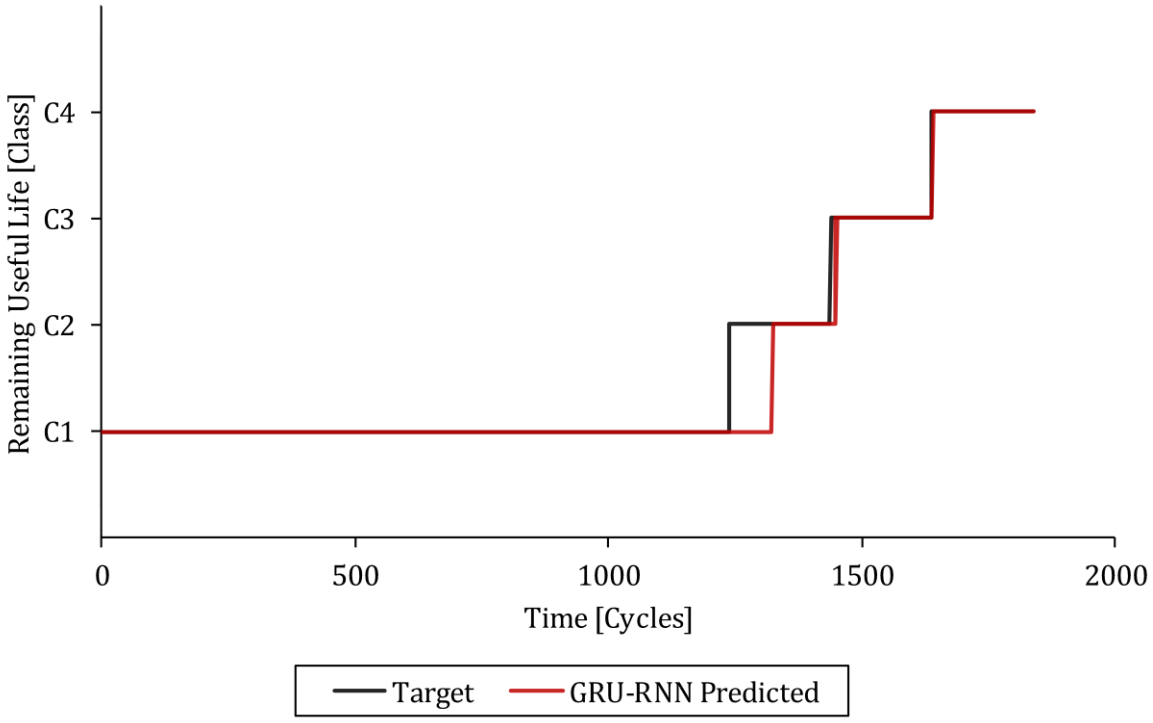

Figure 4-15: The target and GRU-RNN classification model predicted remaining useful life classes versus time for testing set example 1.

Testing Set Example 2

The S1 condition monitoring measurements versus time for testing set example 2 are shown in Figure 4-16. The target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes versus time for testing set example 2 are shown in Figure 4-17, Figure 4-18, Figure 4-19 and Figure 4-20 respectively.

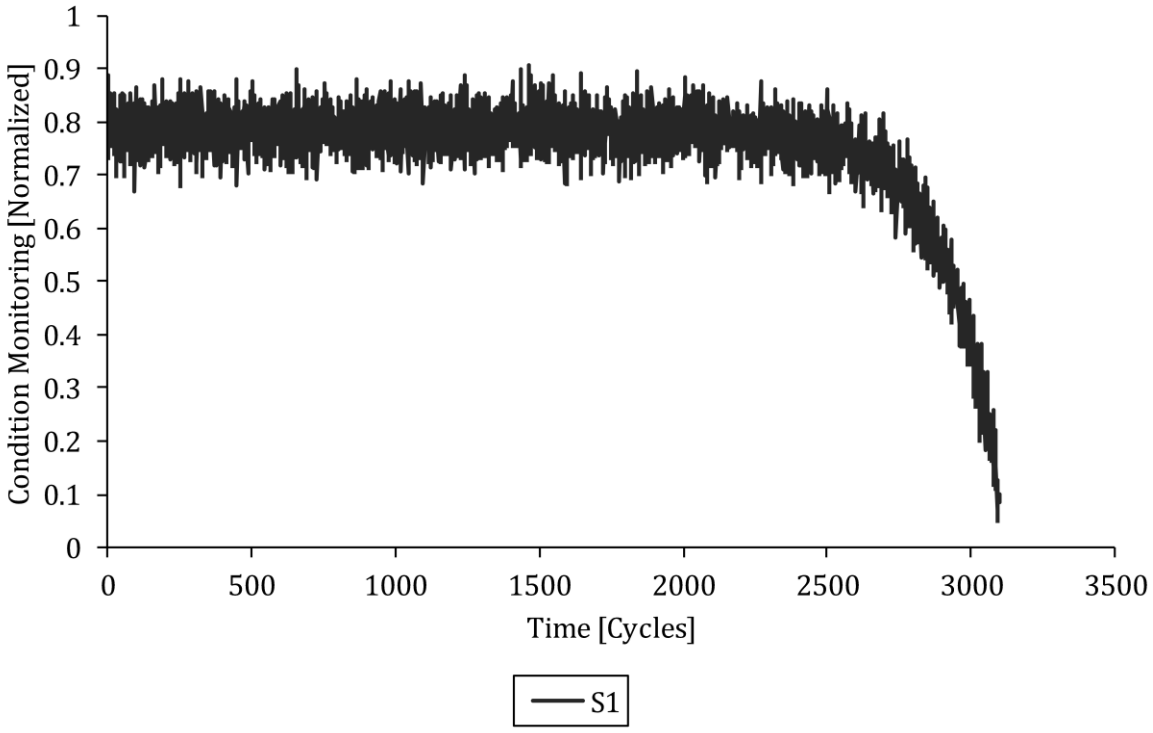

Figure 4-16: The S1 condition monitoring measurements versus time for testing set example 2.

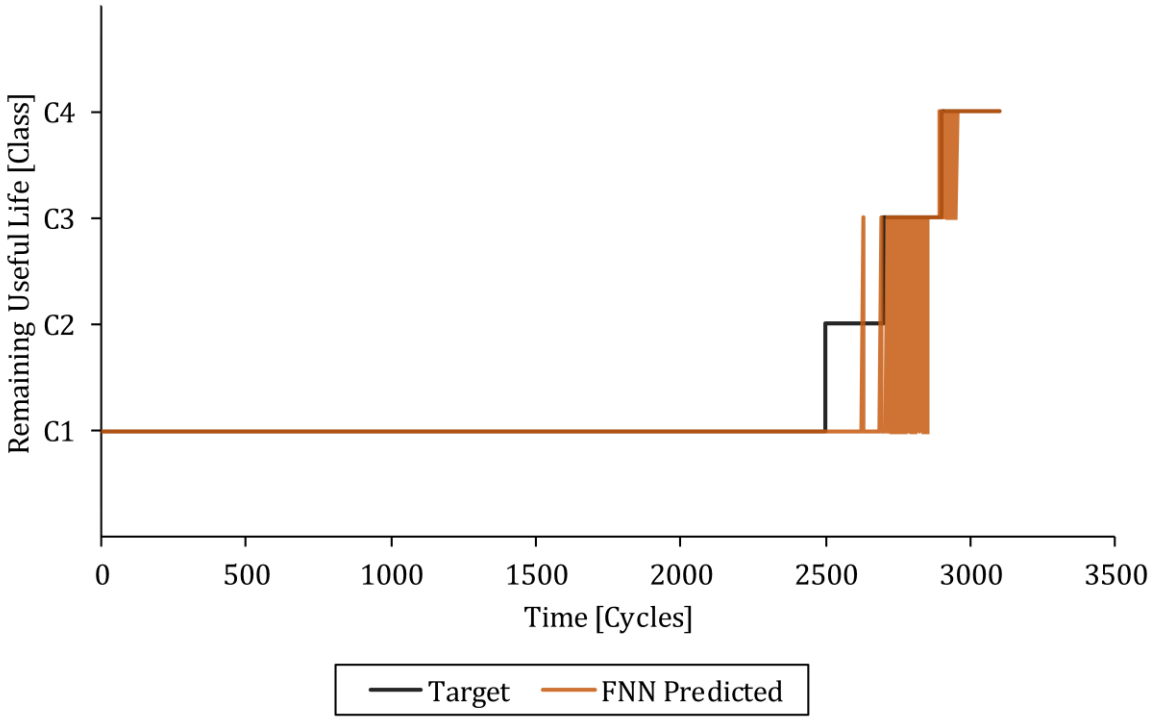

Figure 4-17: The target and FNN classification model predicted remaining useful life classes versus time for testing set example 2.

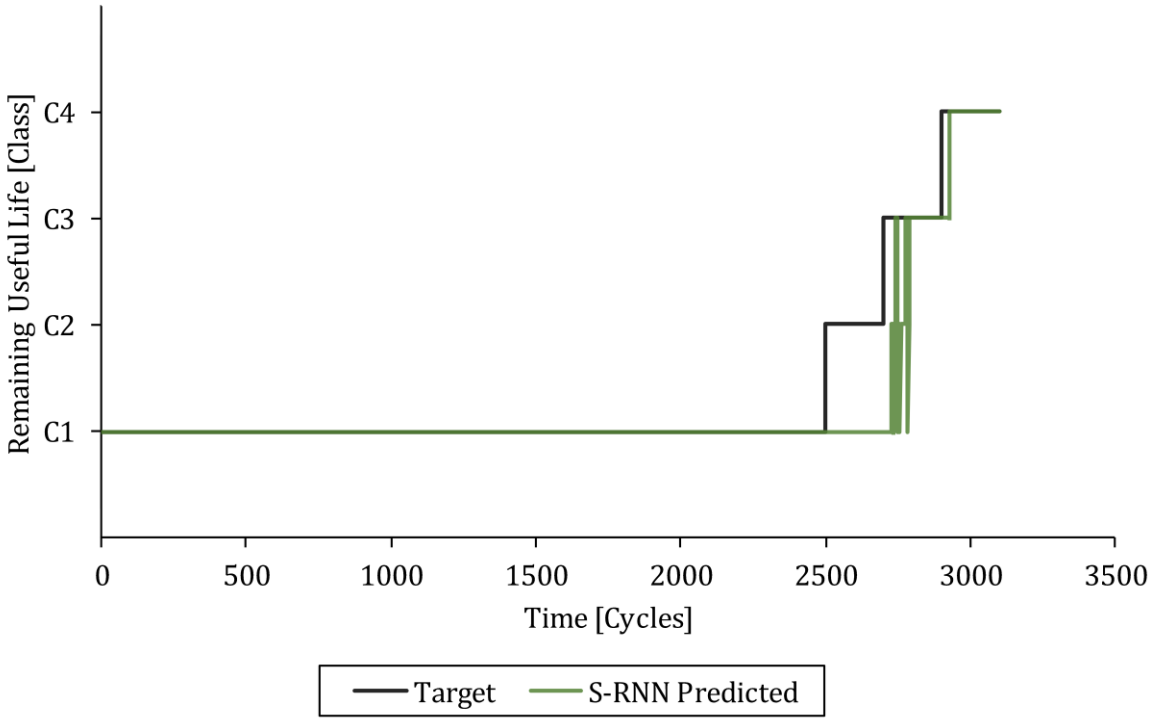

Figure 4-18: The target and S-RNN classification model predicted remaining useful life classes versus time for testing set example 2.

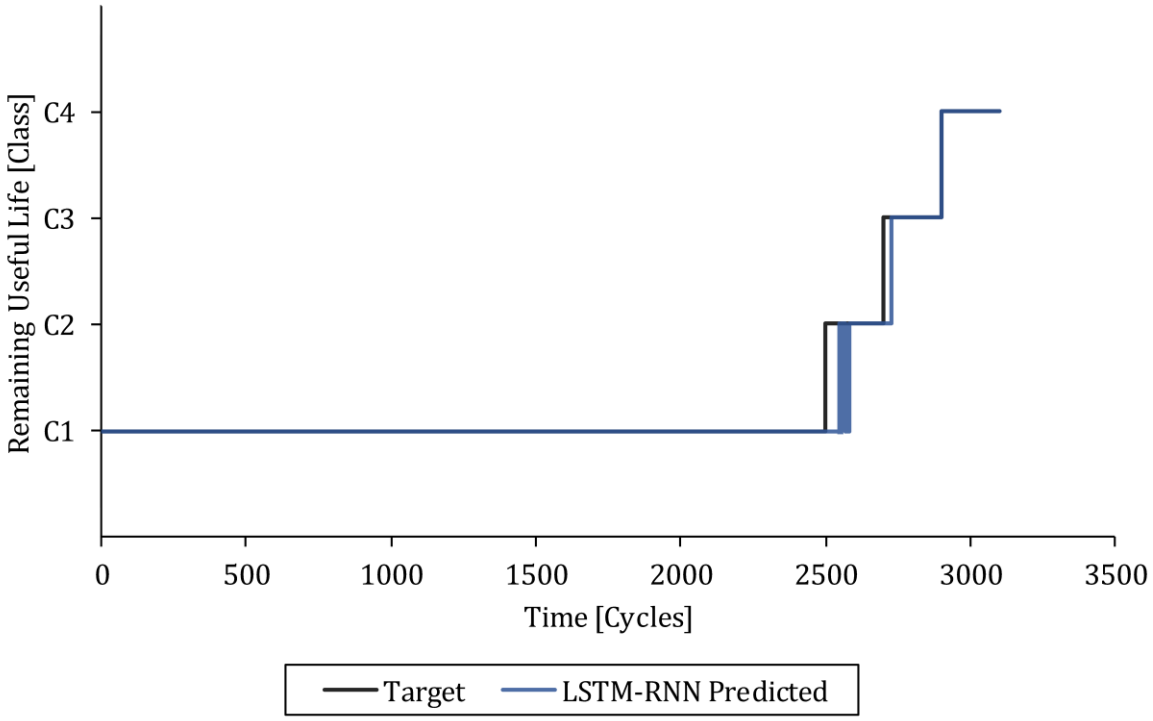

Figure 4-19: The target and LSTM-RNN classification model predicted remaining useful life classes versus time for testing set example 2.

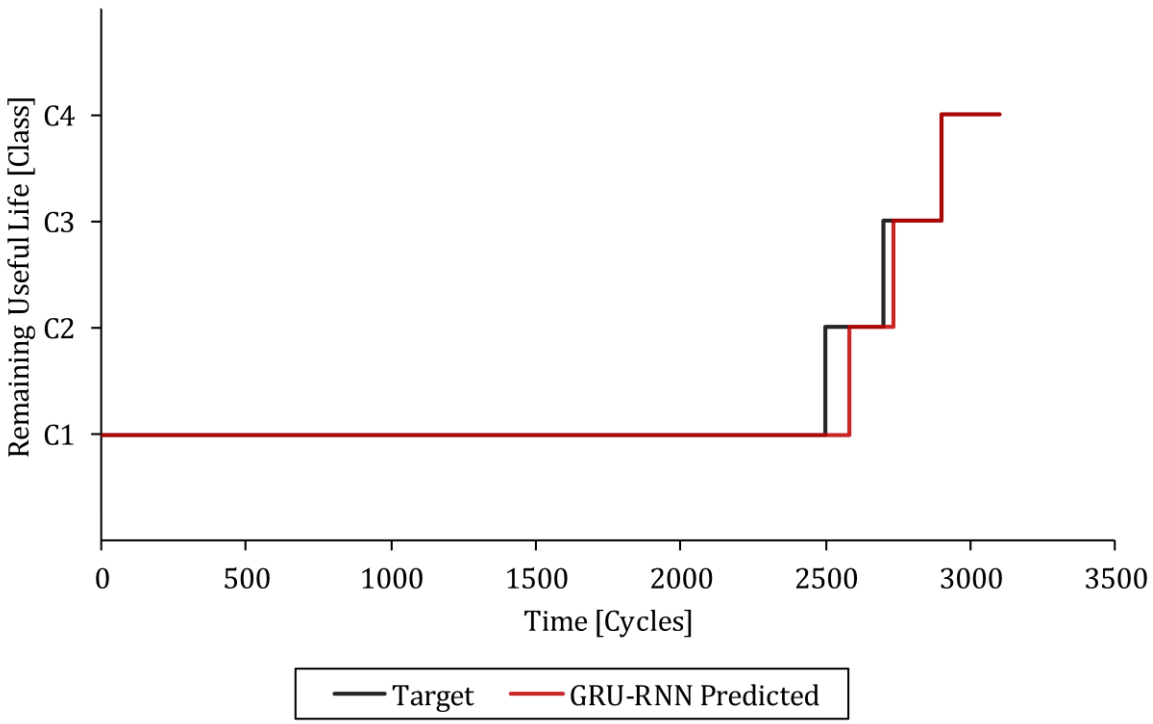

Figure 4-20: The target and GRU-RNN classification model predicted remaining useful life classes versus time for testing set example 2.

From Figure 4-1–Figure 4-20 it can be concluded that the prognostics classification strategy and trained FNN, S-RNN, LSTM-RNN and GRU-RNN classification models can successfully predict the remaining useful life classes from the condition monitoring measurements for the two randomly selected training and testing set examples in the general asset degradation data set fully online. The LSTM-RNN and GRU-RNN classification models drastically outperformed the FNN and S-RNN classification models on the two randomly selected training and testing set examples as expected. The GRU-RNN classification model slightly outperformed the LSTM-RNN classification model and the S-RNN classification model significantly outperformed the FNN classification model on average. The predicted remaining useful life classes of the FNN classification model were also drastically more noisy than that of the S-RNN, LSTM-RNN and GRU-RNN classification models. The predicted remaining useful life classes of the FNN, S-RNN, LSTM-RNN and GRU-RNN classification models were also significantly more accurate close to failure as the trendability of the condition monitoring measurements increased.

4.1.4 Model Performance Comparison

The performance of the FNN, S-RNN, LSTM-RNN and GRU-RNN classification models were compared by presenting the confusion matrix between the target and predicted remaining useful life classes for all the training set and testing set examples in the general asset degradation data set. The confusion matrix between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes for all the training set examples are shown in Table 4-2, Table 4-3, Table 4-4 and Table 4-5 respectively.

Table 4-2: The confusion matrix between the target and FNN classification model predicted remaining useful life classes for all the training set examples.

		Predicted Class			
		C1	**C2**	**C3**	**C4**
Target Class	**C1**	152330	0	1730	0
	C2	13190	0	2810	0
	C3	4685	0	10135	1180
	C4	71	0	2023	13906

Table 4-3: The confusion matrix between the target and S-RNN classification model predicted remaining useful life classes for all the training set examples.

		Predicted Class			
		C1	**C2**	**C3**	**C4**
Target Class	**C1**	152262	1018	780	0
	C2	12257	1164	2579	0
	C3	3206	999	10955	840
	C4	41	25	2090	13844

Table 4-4: The confusion matrix between the target and LSTM-RNN classification model predicted remaining useful life classes for all the training set examples.

		Predicted Class			
		C1	**C2**	**C3**	**C4**
Target Class	**C1**	152256	1797	0	7
	C2	3493	10660	1847	0
	C3	184	1599	13574	643
	C4	0	0	560	15440

Table 4-5: The confusion matrix between the target and GRU-RNN classification model predicted remaining useful life classes for all the training set examples.

		Predicted Class			
		C1	**C2**	**C3**	**C4**
Target Class	**C1**	151910	2141	9	0
	C2	3427	11713	860	0
	C3	89	2080	13425	406
	C4	0	0	389	15611

The confusion matrix between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes for all the testing set examples are shown in Table 4-6, Table 4-7, Table 4-8, Table 4-9 and respectively.

Table 4-6: The confusion matrix between the target and FNN classification model predicted remaining useful life classes for all the testing set examples.

		Predicted Class			
		C1	**C2**	**C3**	**C4**
Target Class	**C1**	34895	0	354	0
	C2	3345	0	655	0
	C3	1284	0	2445	271
	C4	11	0	558	3431

Table 4-7: The confusion matrix between the target and S-RNN classification model predicted remaining useful life classes for all the testing set examples.

		Predicted Class			
		C1	**C2**	**C3**	**C4**
Target Class	**C1**	35021	161	67	0
	C2	3169	228	603	0
	C3	899	306	2580	215
	C4	0	1	569	3430

Table 4-8: The confusion matrix between the target and LSTM-RNN classification model predicted remaining useful life classes for all the testing set examples.

		Predicted Class			
		C1	**C2**	**C3**	**C4**
Target Class	**C1**	34623	626	0	0
	C2	1054	2408	538	0
	C3	19	526	3349	106
	C4	0	0	128	3872

Table 4-9: The confusion matrix between the target and GRU-RNN classification model predicted remaining useful life classes for all the testing set examples.

		Predicted Class			
		C1	**C2**	**C3**	**C4**
Target Class	**C1**	34621	628	0	0
	C2	970	2693	337	0
	C3	0	396	3509	95
	C4	0	0	48	3952

From Table 4-2–Table 4-9 it can be concluded that the LSTM-RNN and GRU-RNN classification models drastically outperformed the FNN and S-RNN classification models on the training set and testing set as expected. The GRU-RNN classification model slightly outperformed the LSTM-RNN classification model and the S-RNN classification model significantly outperformed the FNN classification model. The predicted remaining useful life classes of the FNN, S-RNN, LSTM-RNN and GRU-RNN classification models were also significantly more accurate close to failure (with less confusion between target and predicted classes) as the trendability of the condition monitoring measurements increased.

The performance of the FNN, S-RNN, LSTM-RNN and GRU-RNN classification models were also compared by calculating the cross-entropy and accuracy between the target and predicted remaining useful life classes for all the training set and testing set examples in the general asset degradation data set. The cross-entropy and accuracy between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes for all the training set and testing set examples is shown in Table 4-10.

Table 4-10: The cross-entropy and accuracy between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes for all the training set and testing set examples.

Classification Model	Training Set Cross-Entropy	Testing Set Cross-Entropy	Training Set Accuracy	Testing Set Accuracy
FNN	0.3769	0.4014	0.8729	0.8629
S-RNN	0.3322	0.3635	0.8820	0.8732
LSTM-RNN	0.1307	0.1668	0.9499	0.9366
GRU-RNN	0.1218	0.1324	0.9535	0.9476

The performance of the FNN, S-RNN, LSTM-RNN and GRU-RNN classification models were also compared by recalculating the cross-entropy and accuracy between the target and predicted remaining useful life classes 600 cycles before failure for all the training set and testing set examples in the general asset degradation data set. This is because it excluded the remaining useful life classes above the applied target threshold and was therefore more representative of the classification model performance close to failure. The cross-entropy and accuracy between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes 600 cycles before failure for all the training set and testing set examples is shown in Table 4-11.

Table 4-11: The cross-entropy and accuracy between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes 600 cycles before failure for all the training set and testing set examples.

Classification Model	Training Set Cross-Entropy	Testing Set Cross-Entropy	Training Set Accuracy	Testing Set Accuracy
FNN	1.1713	1.200	0.4993	0.4880
S-RNN	1.1649	1.2181	0.5409	0.5198
LSTM-RNN	0.4450	0.5199	0.8265	0.8024
GRU-RNN	0.3876	0.3862	0.8489	0.8462

From Table 4-10 and Table 4-11 it can be concluded that the LSTM-RNN and GRU-RNN classification models drastically outperformed the FNN and S-RNN classification models on the training set and testing set as expected. The GRU-RNN classification model slightly outperformed the LSTM-RNN classification model and the S-RNN classification model significantly outperformed the FNN classification model. The difference between the training set and testing set cross-entropy and accuracy for the FNN, S-RNN, LSTM-RNN and GRU-RNN classification models 600 cycles before failure was also relatively small, which indicated good model regularization and an appropriately selected threshold for the prognostics classification strategy. The GRU-RNN classification model therefore had a very respectable accuracy of 0.8462 for the completely unseen testing set in the general asset degradation data set 600 cycles before failure.

4.2 Turbofan Engine Degradation Data Set

This section presents the results of the prognostics classification strategy and classification deep learning model architectures applied on the turbofan engine degradation data set.

4.2.1 Strategy and Model Description

The applied threshold for the prognostics classification strategy was tuned and selected as $AT = 120$ cycles for the turbofan engine degradation data set. The selected remaining useful life definition and corresponding degradation level and fault mode for the different classes of the prognostics classification strategy applied on the turbofan engine degradation data set is shown in Table 4-12.

Table 4-12: The selected remaining useful life definition and corresponding degradation level and fault mode for the different classes of the prognostics classification strategy applied on the turbofan engine degradation data set.

Class	Degradation Level and Fault Mode	Remaining Useful Life Definition
C1	Light High Pressure Compressor Degradation	RUL $>$ 120 Cycles
C2	Medium High Pressure Compressor Degradation	80 Cycles $<$ RUL $\leq$ 120 Cycles
C3	Heavy High Pressure Compressor Degradation	40 Cycles $<$ RUL $\leq$ 80 Cycles
C4	Extreme High Pressure Compressor Degradation	RUL $\leq$ 40 Cycles
C5	Light Fan Degradation	RUL $>$ 120 Cycles
C6	Medium Fan Degradation	80 Cycles $<$ RUL $\leq$ 120 Cycles
C7	Heavy Fan Degradation	40 Cycles $<$ RUL $\leq$ 80 Cycles
C8	Extreme Fan Degradation	RUL $\leq$ 40 Cycles

The FNN, S-RNN, LSTM-RNN and GRU-RNN classification model architectures were trained on the training set of the turbofan engine degradation data set with the Adam algorithm and regularized with a combination of the early stopping, weight decay and dropout regularization techniques.

4.2.2 Training Set Examples

The objective of this section is to present and compare how accurately the trained FNN, S-RNN, LSTM-RNN and GRU-RNN classification models could predict the remaining useful life classes from the condition monitoring measurements for two randomly selected training set examples in the turbofan engine degradation data set fully online. The two randomly selected training set examples had different fault modes and represent two historical (previously seen) turbofan engines that were run to failure.

It is important to point out that the trained FNN, S-RNN, LSTM-RNN and GRU-RNN classification models only used the condition monitoring measurements for the current time step (and previous time steps for the recurrent models) to predict the remaining useful life class for the current time step. The classification models therefore predicted the remaining useful life classes from the condition monitoring measurements fully online. The presented S6, S10, S12, S18 and S24 condition monitoring measurements were normalized with min-max scaling between 0 and 1 for the entire training and testing set. This was done to effectively present the variation in condition monitoring measurements across all the training and testing set examples. It is important to point out that the classification models were however trained and tested on all S1-S24 condition monitoring measurements for the turbofan engine degradation data set.

Training Set Example 1

The S6, S10, S12, S18 and S24 condition monitoring measurements versus time for training set example 1 are shown in Figure 4-21. The target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes versus time for training set example 1 are shown in Figure 4-22, Figure 4-23, Figure 4-24 and Figure 4-25 respectively. The fault mode for training set example 1 is high-pressure compressor degradation.

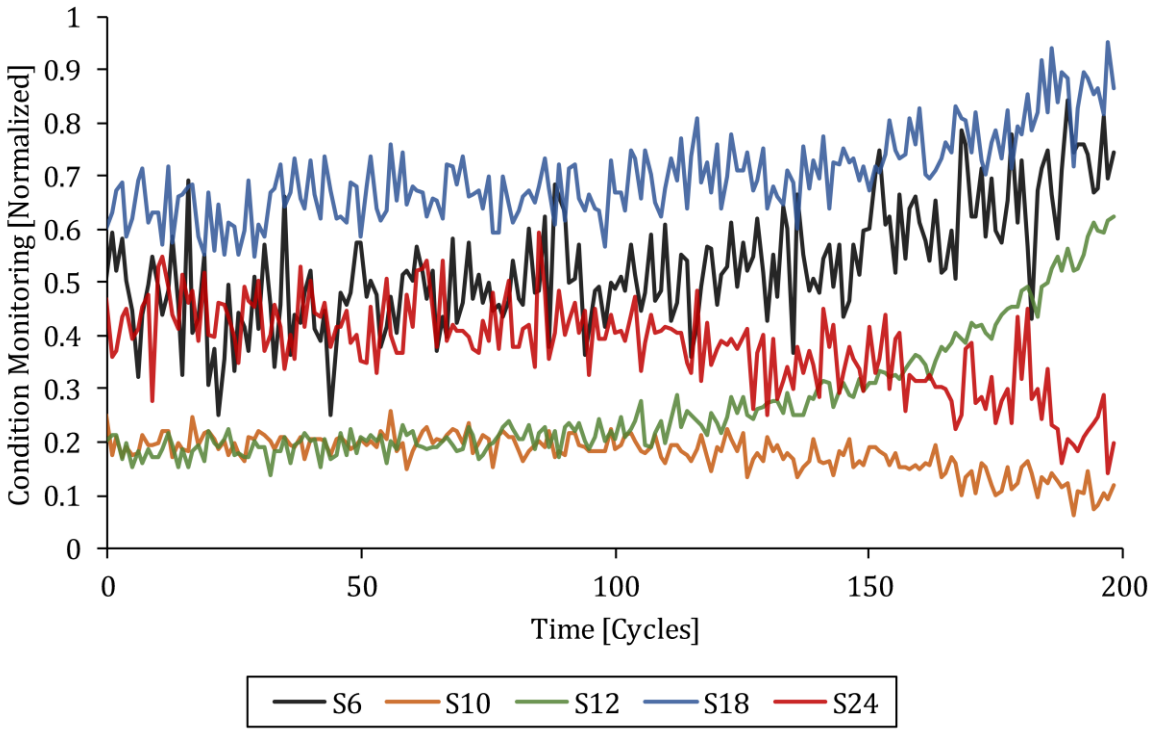

Figure 4-21: The S6, S10, S12, S18 and S24 condition monitoring measurements versus time for training set example 1.

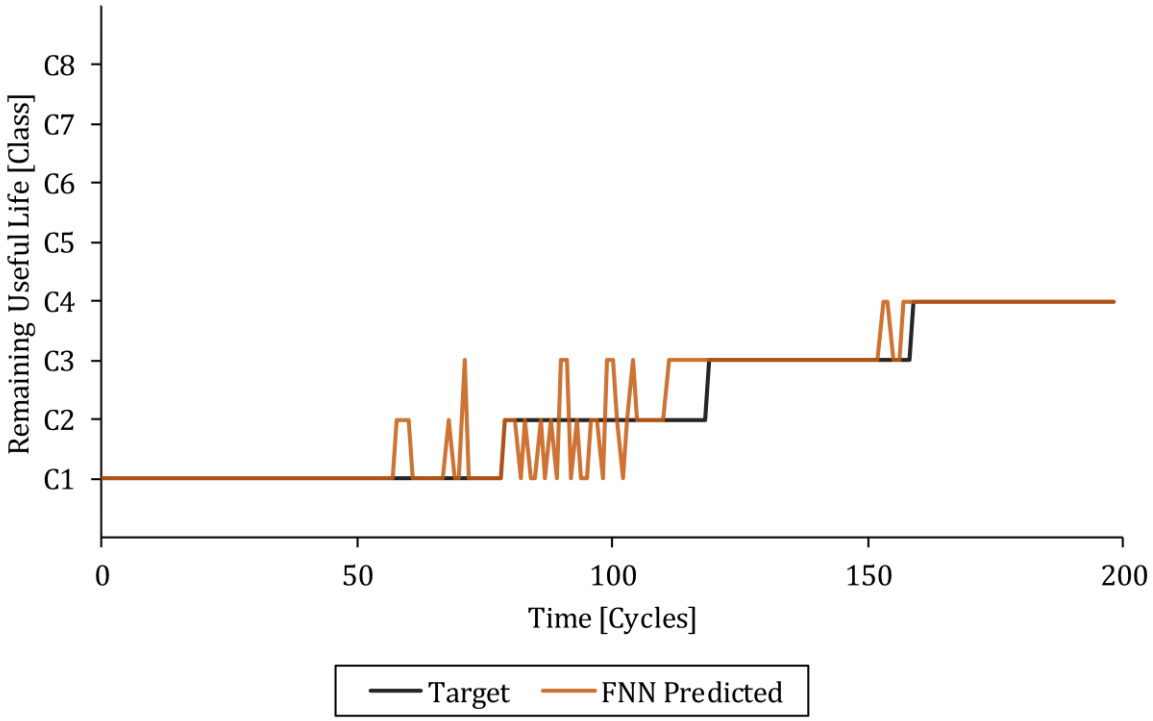

Figure 4-22: The target and FNN classification model predicted remaining useful life classes versus time for training set example 1.

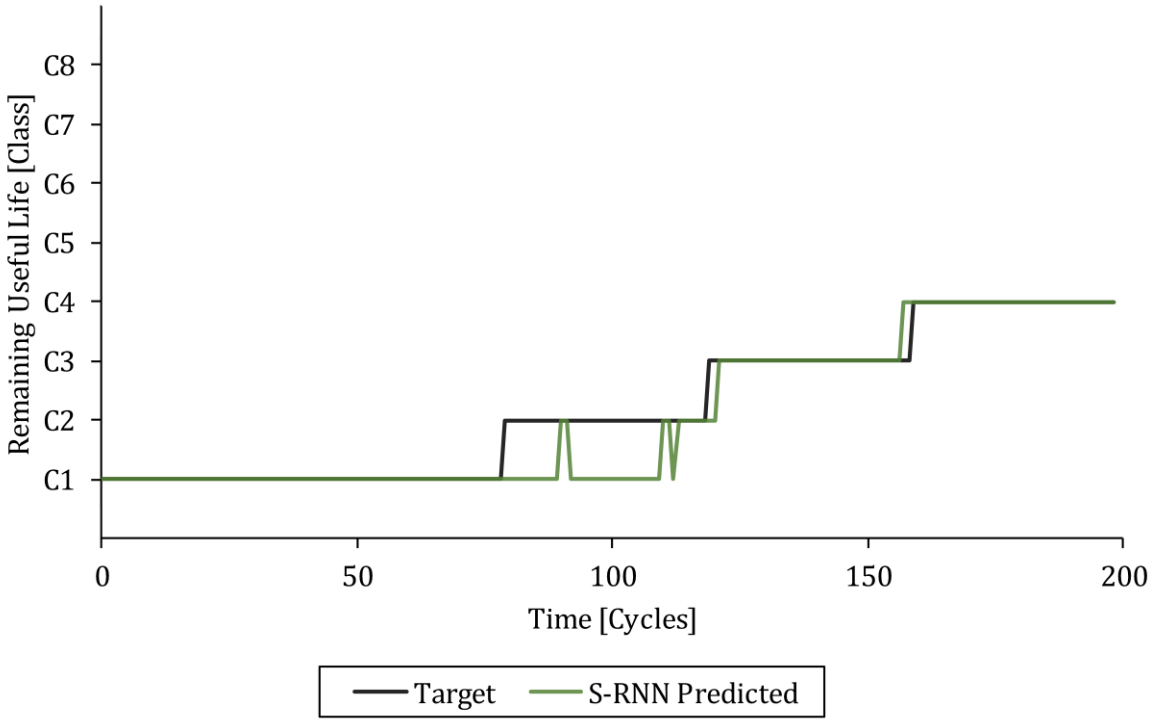

Figure 4-23: The target and S-RNN classification model predicted remaining useful life classes versus time for training set example 1.

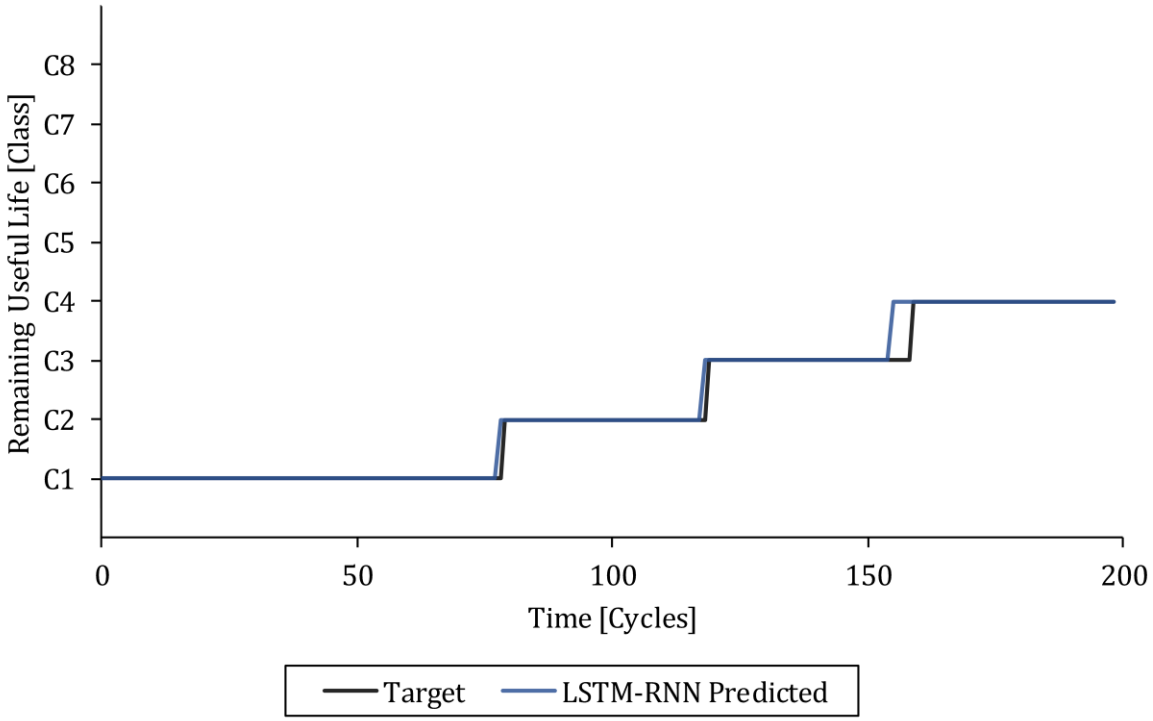

Figure 4-24: The target and LSTM-RNN classification model predicted remaining useful life classes versus time for training set example 1.

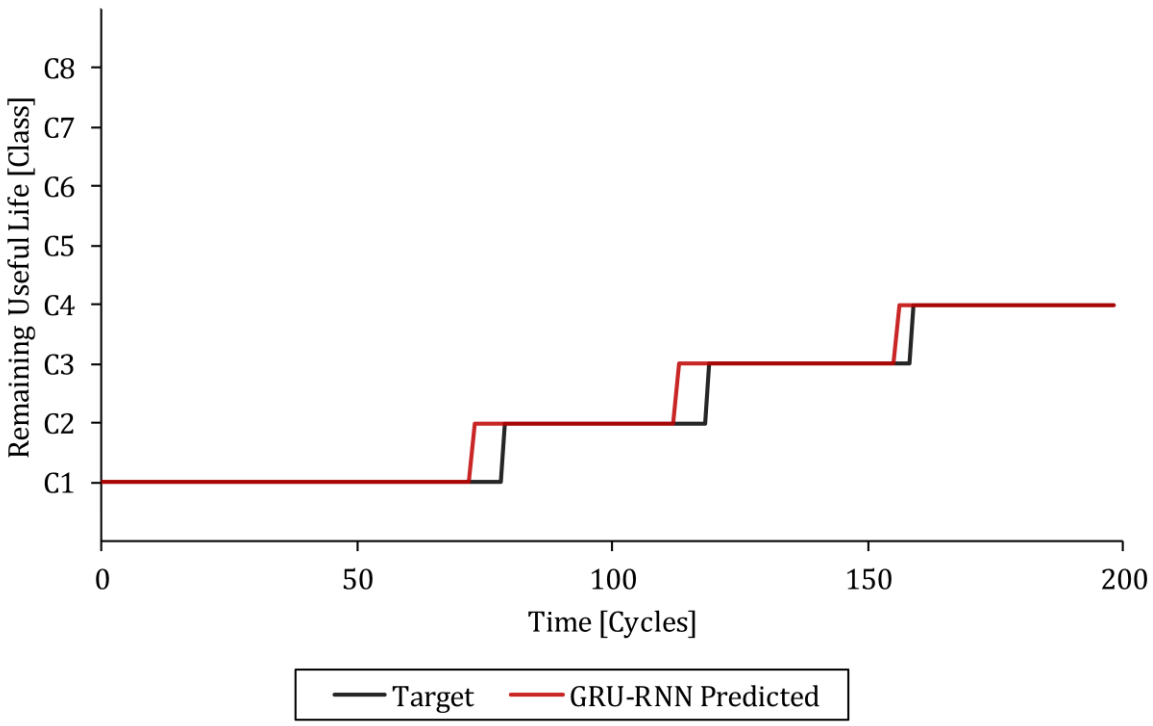

Figure 4-25: The target and GRU-RNN classification model predicted remaining useful life classes versus time for training set example 1.

Training Set Example 2

The S6, S10, S12, S18 and S24 condition monitoring measurements versus time for training set example 2 are shown in Figure 4-26. The target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes versus time for training set example 2 are shown in Figure 4-27, Figure 4-28, Figure 4-29 and Figure 4-30 respectively. The fault mode for training set example 2 is fan degradation.

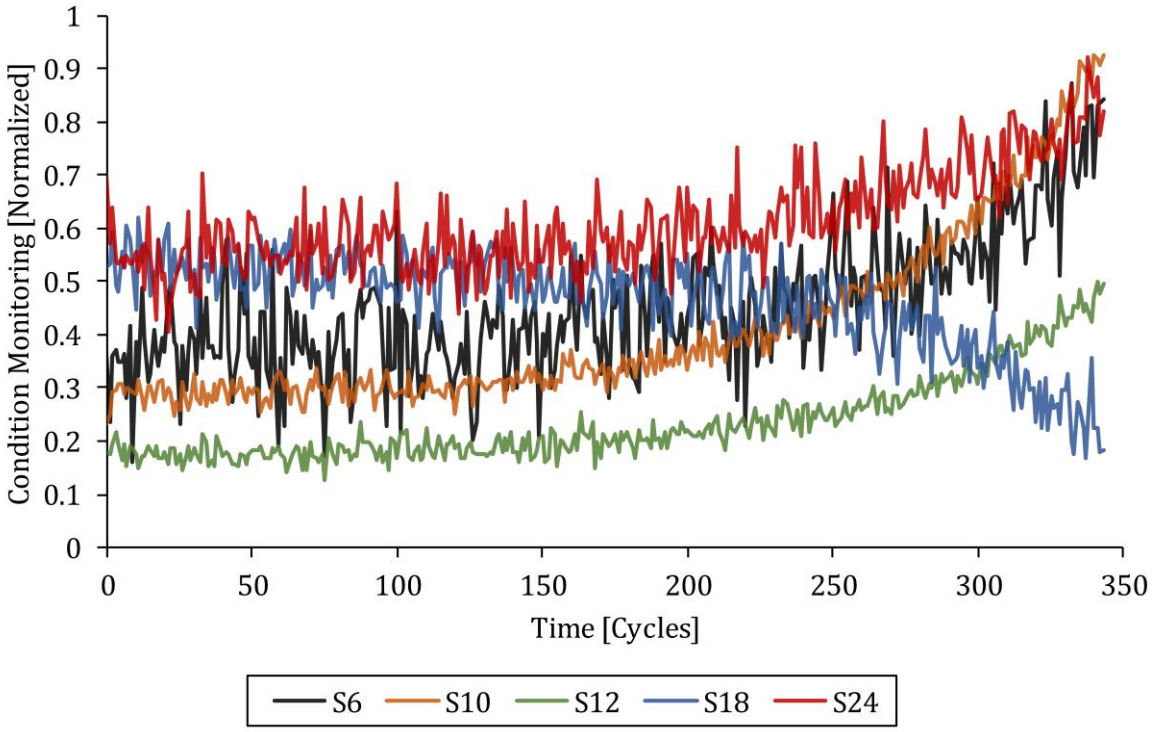

Figure 4-26: The S6, S10, S12, S18 and S24 condition monitoring measurements versus time for training set example 2.

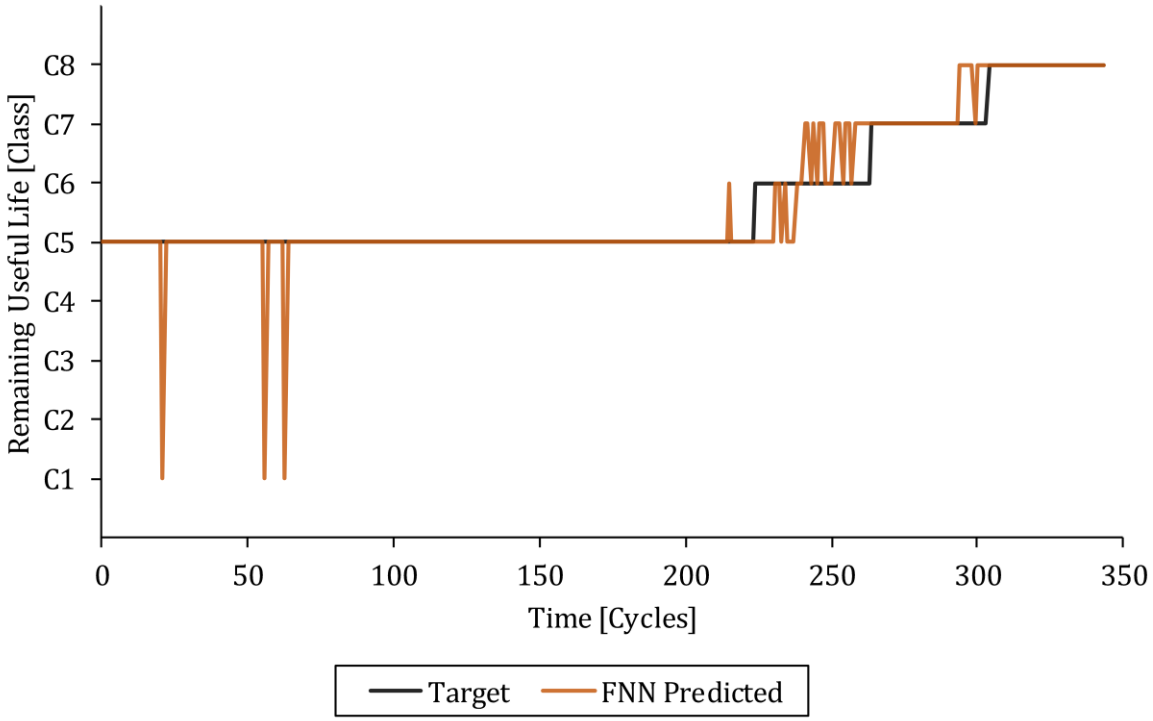

Figure 4-27: The target and FNN classification model predicted remaining useful life classes versus time for training set example 2.

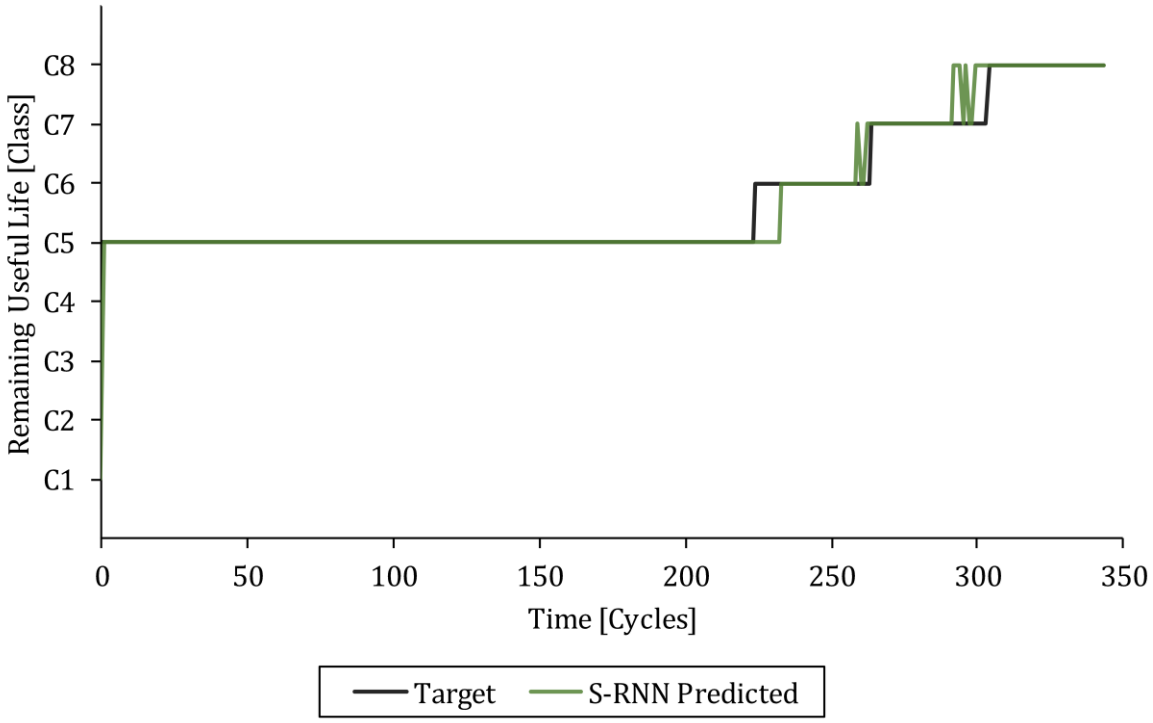

Figure 4-28: The target and S-RNN classification model predicted remaining useful life classes versus time for training set example 2.

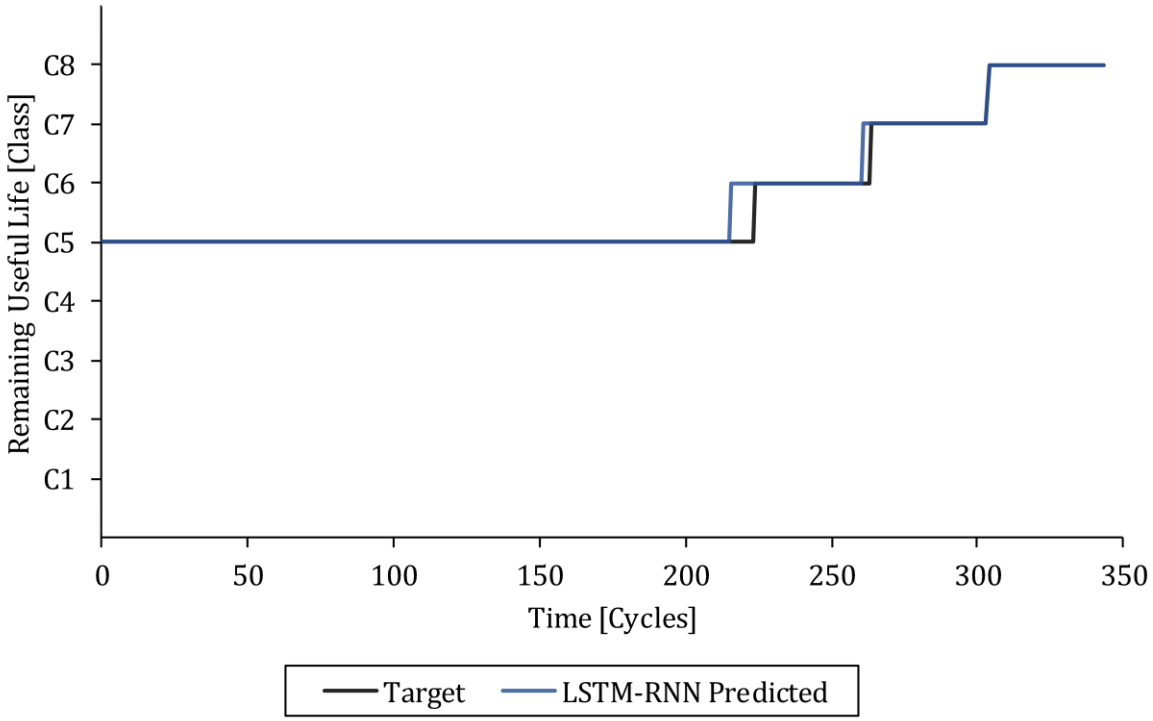

Figure 4-29: The target and LSTM-RNN classification model predicted remaining useful life classes versus time for training set example 2.

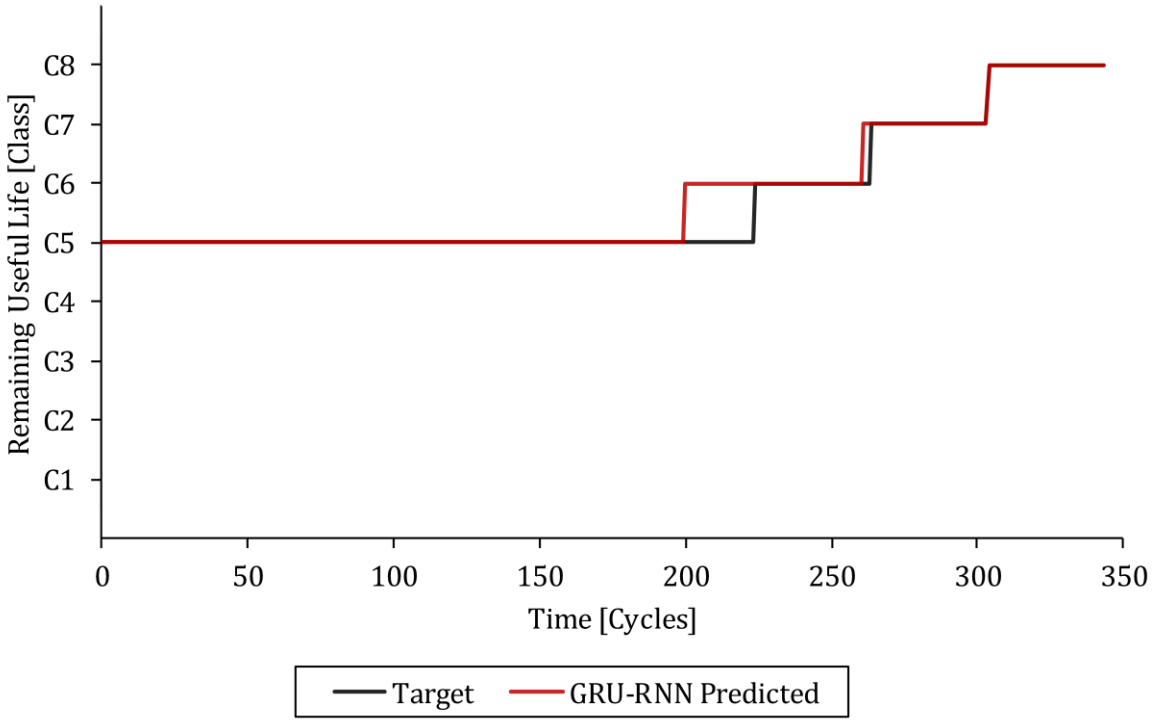

Figure 4-30: The target and GRU-RNN classification model predicted remaining useful life classes versus time for training set example 2.

4.2.3 Testing Set Examples

The objective of this section is to present and compare how accurately the trained FNN, S-RNN, LSTM-RNN and GRU-RNN classification models could predict the remaining useful life classes from the condition monitoring measurements for two randomly selected testing set examples in the turbofan engine degradation data set fully online. The two randomly selected testing set examples had different fault modes and represent two future (completely unseen) turbofan engines that were run to failure.

Testing Set Example 1

The S6, S10, S12, S18 and S24 condition monitoring measurements versus time for testing set example 1 are shown in Figure 4-31. The target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes versus time for testing set example 1 are shown in Figure 4-32, Figure 4-33, Figure 4-34 and Figure 4-35 respectively. The fault mode for testing set example 1 is high-pressure compressor degradation.

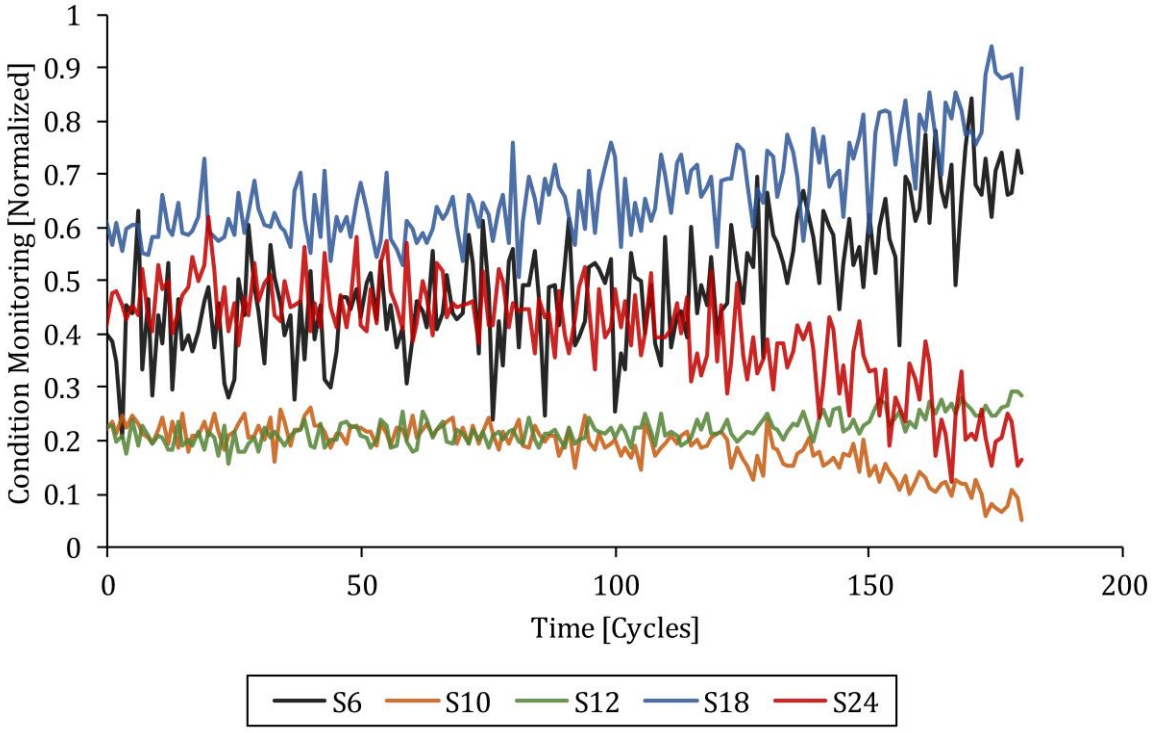

Figure 4-31: The S6, S10, S12, S18 and S24 condition monitoring measurements versus time for testing set example 1.

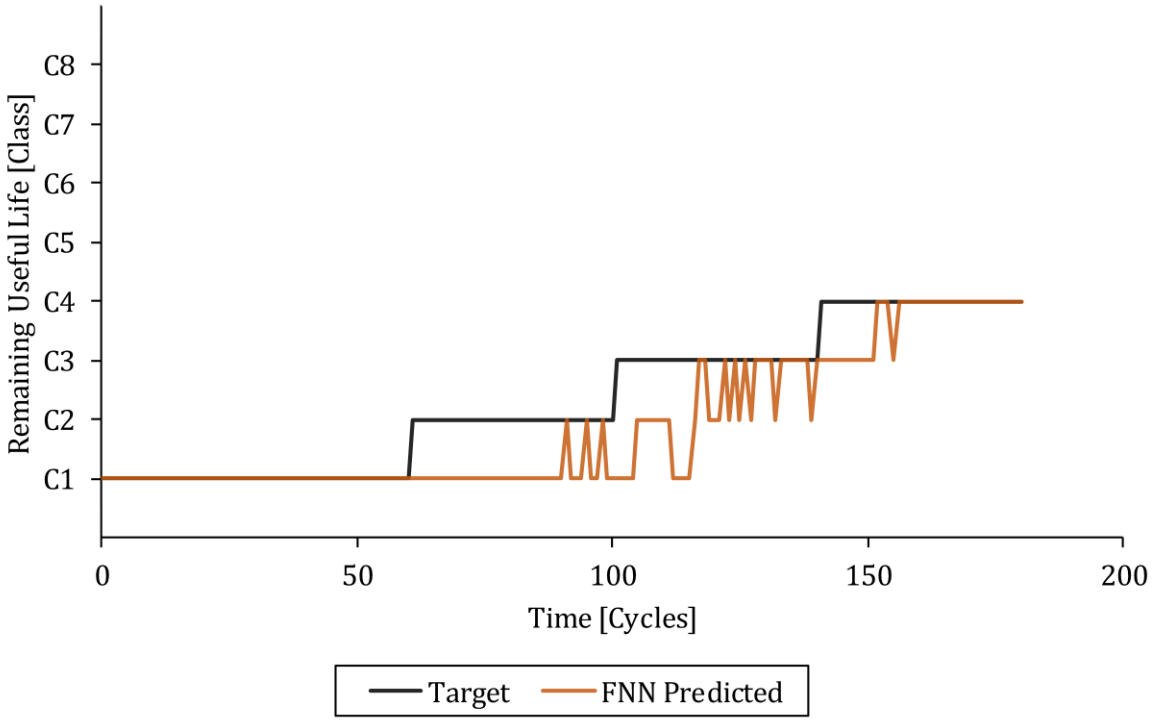

Figure 4-32: The target and FNN classification model predicted remaining useful life classes versus time for testing set example 1.

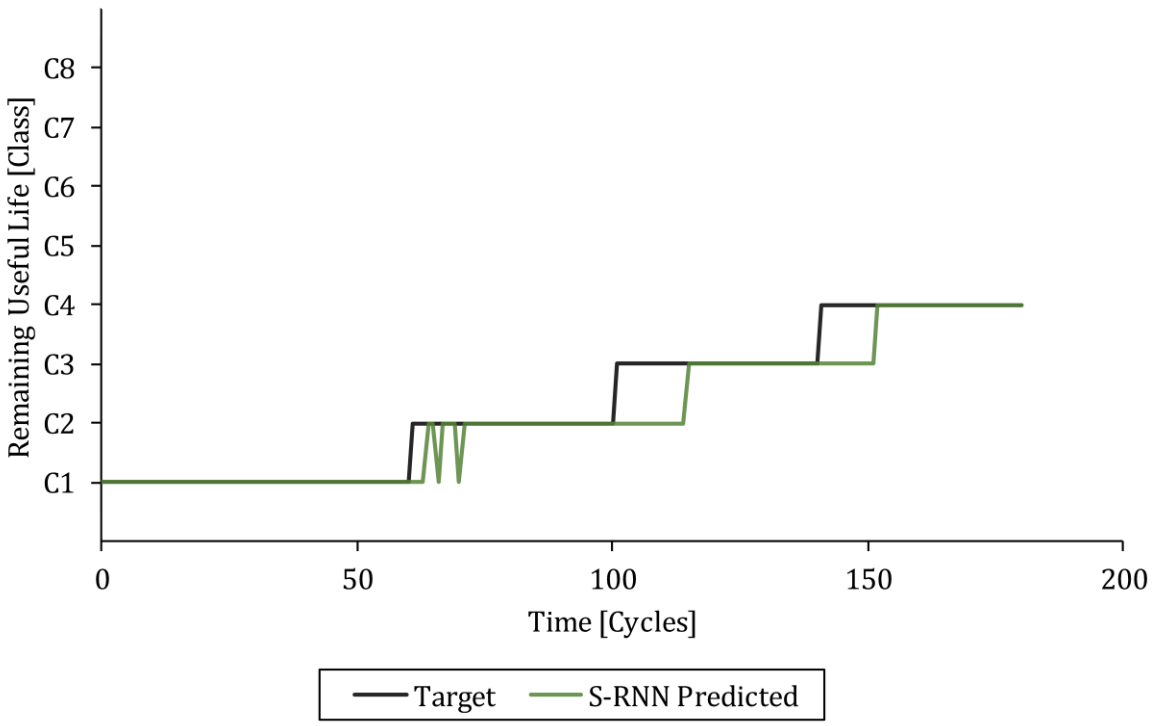

Figure 4-33: The target and S-RNN classification model predicted remaining useful life classes versus time for testing set example 1.

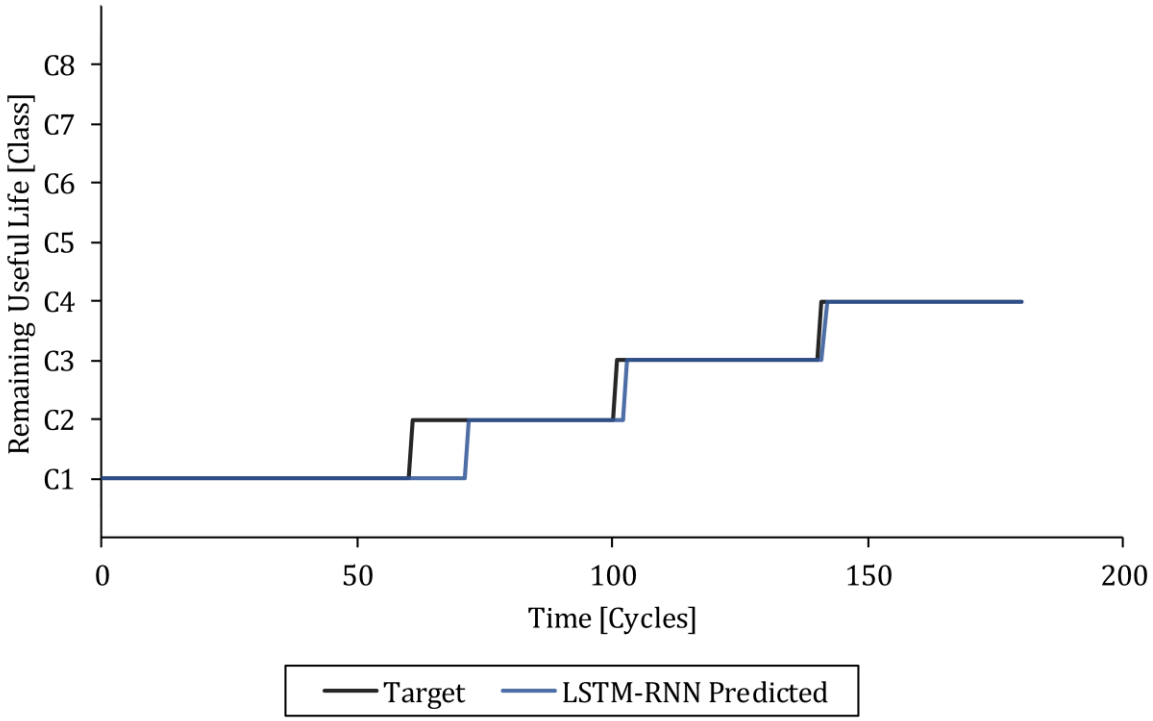

Figure 4-34: The target and LSTM-RNN classification model predicted remaining useful life classes versus time for testing set example 1.

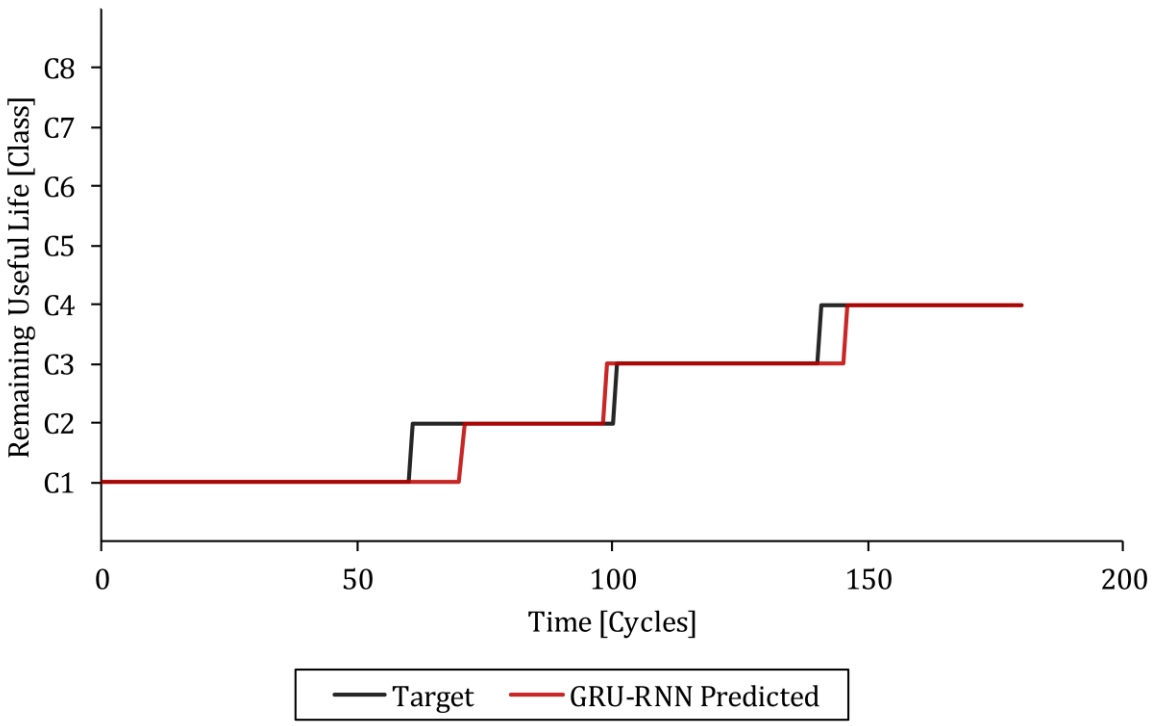

Figure 4-35: The target and GRU-RNN classification model predicted remaining useful life classes versus time for testing set example 1.

Testing Set Example 2

The S6, S10, S12, S18 and S24 condition monitoring measurements versus time for testing set example 1 are shown in Figure 4-36. The target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes versus time for testing set example 2 are shown in Figure 4-37, Figure 4-38, Figure 4-39 and Figure 4-40 respectively. The fault mode for testing set example 2 is fan degradation.

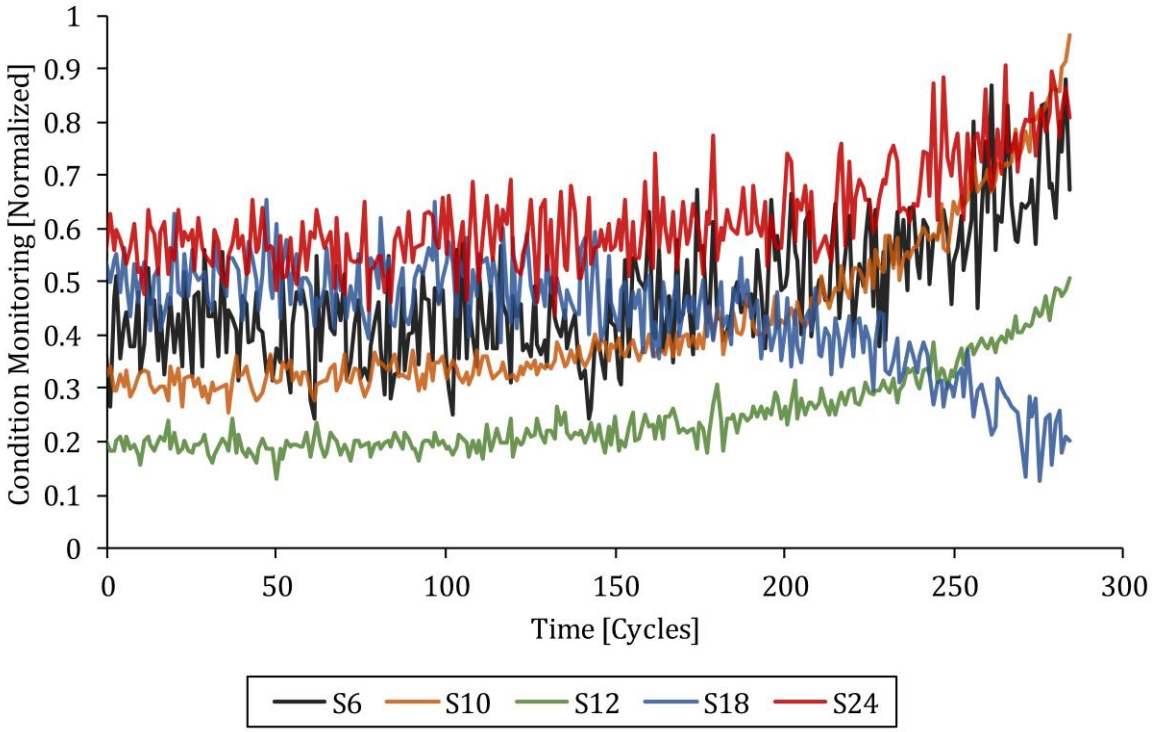

Figure 4-36: The S6, S10, S12, S18 and S24 condition monitoring measurements versus time for testing set example 2.

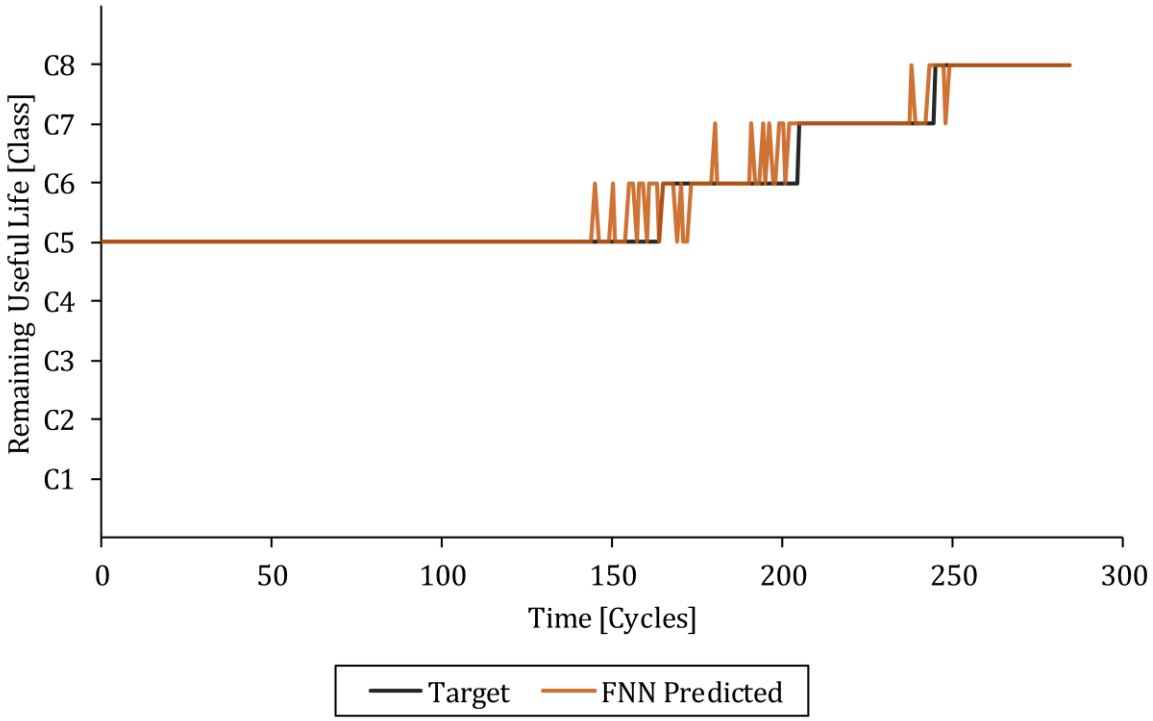

Figure 4-37: The target and FNN classification model predicted remaining useful life classes versus time for testing set example 2.

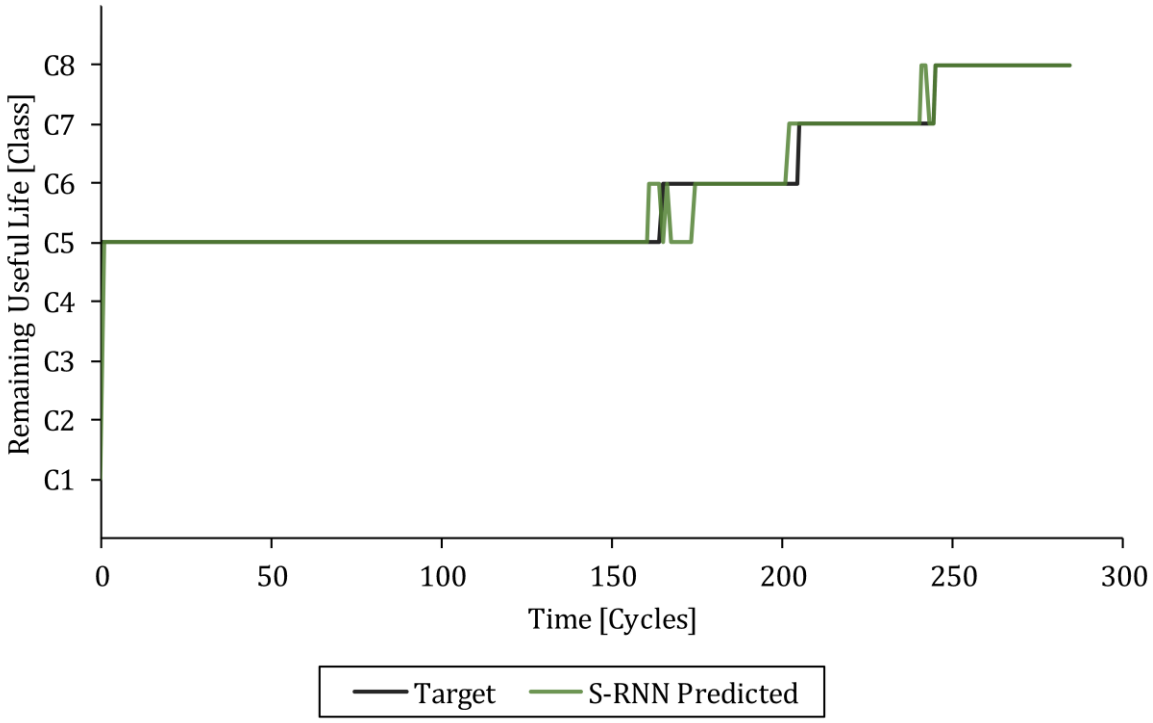

Figure 4-38: The target and S-RNN classification model predicted remaining useful life classes versus time for testing set example 2.

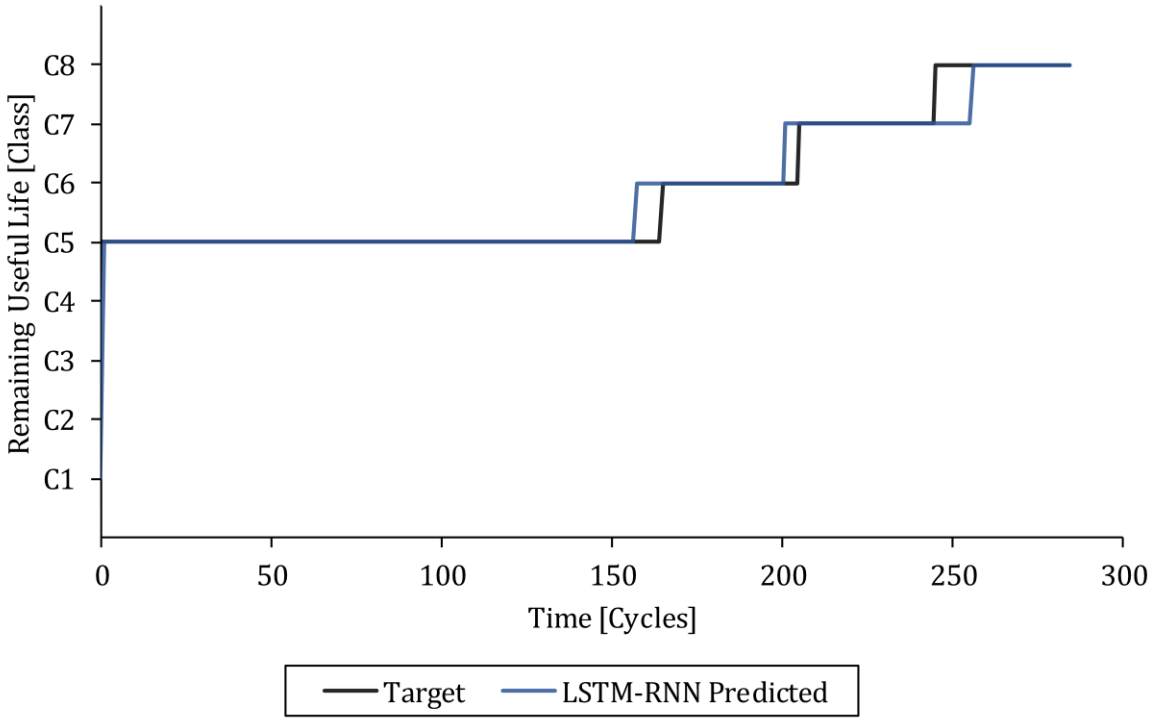

Figure 4-39: The target and LSTM-RNN classification model predicted remaining useful life classes versus time for testing set example 2.

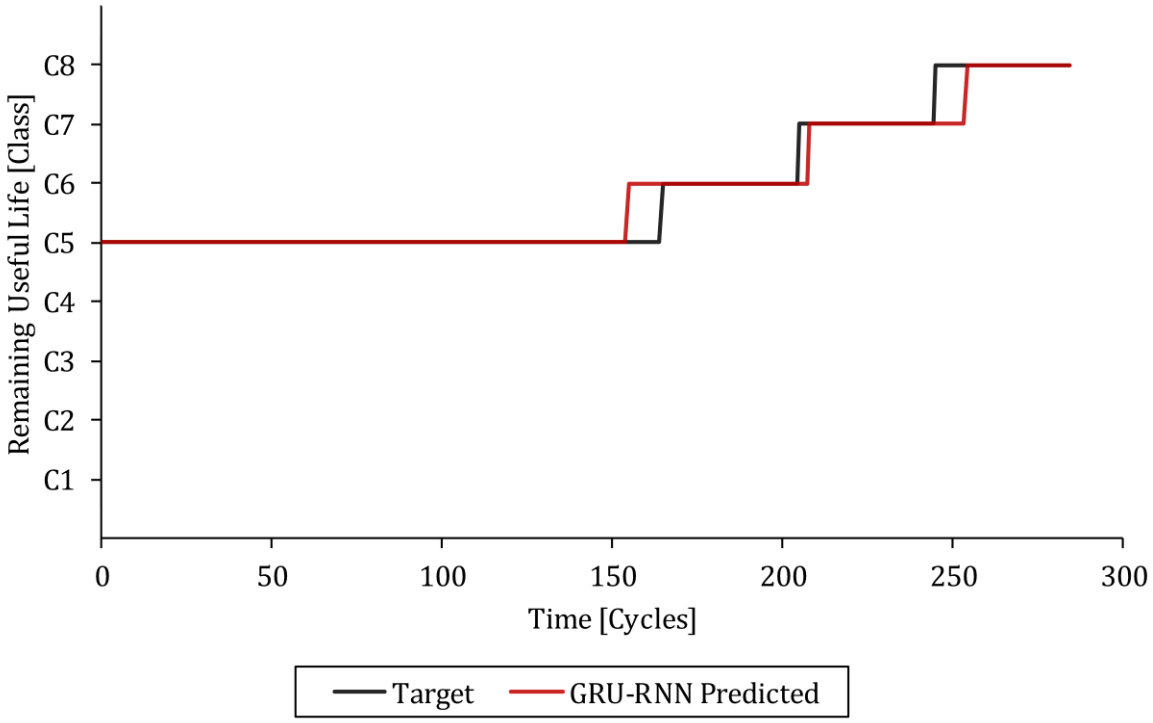

Figure 4-40: The target and GRU-RNN classification model predicted remaining useful life classes versus time for testing set example 2.

From Figure 4-21–Figure 4-40 it can be concluded that the prognostics classification strategy and trained FNN, S-RNN, LSTM-RNN and GRU-RNN classification models can successfully predict the remaining useful life classes from the condition monitoring measurements for the two randomly selected training and testing set examples in the turbofan engine degradation data set fully online. The LSTM-RNN and GRU-RNN classification models drastically outperformed the FNN and S-RNN classification models on the two randomly selected training and testing set examples as expected. The GRU-RNN classification model slightly outperformed the LSTM-RNN classification model and the S-RNN classification model significantly outperformed the FNN classification model on average. The predicted remaining useful life classes of the FNN classification model were also drastically more noisy than that of the S-RNN, LSTM-RNN and GRU-RNN classification models. The predicted remaining useful life classes of the FNN, S-RNN, LSTM-RNN and GRU-RNN classification models were also significantly more accurate close to failure as the trendability of the condition monitoring measurements increased.

4.2.4 Model Performance Comparison

The performance of the FNN, S-RNN, LSTM-RNN and GRU-RNN classification models were compared by presenting the confusion matrix between the target and predicted remaining useful life classes for all the training set and testing set examples in the turbofan engine degradation data set. The confusion matrix between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes for all the training set examples are shown in Table 4-13, Table 4-14, Table 4-15 and Table 4-16 respectively.

Table 4-13: The confusion matrix between the target and FNN classification model predicted remaining useful life classes for all the training set examples.

		Predicted Class							
		C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**
Target Class	**C1**	3632	234	76	0	30	0	0	0
	C2	1143	353	342	2	0	0	0	0
	C3	365	203	1100	172	0	0	0	0
	C4	10	6	236	1588	0	0	0	0
	C5	26	0	0	0	6057	92	32	0
	C6	0	0	0	0	820	352	188	0
	C7	0	0	0	0	232	264	770	94
	C8	0	0	0	0	0	5	236	1119

Table 4-14: The confusion matrix between the target and S-RNN classification model predicted remaining useful life classes for all the training set examples.

		Predicted Class							
		C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**
Target Class	**C1**	3508	343	116	0	5	0	0	0
	C2	814	665	357	4	0	0	0	0
	C3	63	298	1277	202	0	0	0	0
	C4	0	0	199	1641	0	0	0	0
	C5	14	0	0	0	6102	77	14	0
	C6	0	0	0	0	750	450	160	0
	C7	0	0	0	0	88	307	856	109
	C8	0	0	0	0	0	0	233	1127

Table 4-15: The confusion matrix between the target and LSTM-RNN classification model predicted remaining useful life classes for all the training set examples.

		Predicted Class							
		C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**
Target Class	**C1**	3698	271	0	0	3	0	0	0
	C2	435	1243	162	0	0	0	0	0
	C3	11	208	1541	80	0	0	0	0
	C4	0	0	125	1715	0	0	0	0
	C5	6	0	0	0	6119	82	0	0
	C6	0	0	0	0	103	1205	52	0
	C7	0	0	0	0	0	94	1266	0
	C8	0	0	0	0	0	0	145	1215

Table 4-16: The confusion matrix between the target and GRU-RNN classification model predicted remaining useful life classes for all the training set examples.

		Predicted Class							
		C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**
Target Class	**C1**	3629	321	0	0	22	0	0	0
	C2	344	1290	206	0	0	0	0	0
	C3	2	179	1586	73	0	0	0	0
	C4	0	0	118	1722	0	0	0	0
	C5	9	0	0	0	6020	178	0	0
	C6	0	0	0	0	92	1161	107	0
	C7	0	0	0	0	0	75	1267	18
	C8	0	0	0	0	0	0	140	1220

The confusion matrix between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes for all the testing set examples are shown in Table 4-17, Table 4-18, Table 4-19 and Table 4-20 respectively.

Table 4-17: The confusion matrix between the target and FNN classification model predicted remaining useful life classes for all the testing set examples.

		Predicted Class							
		C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**
Target Class	**C1**	584	21	13	0	5	0	0	0
	C2	300	40	60	0	0	0	0	0
	C3	83	72	217	28	0	0	0	0
	C4	0	2	73	325	0	0	0	0
	C5	0	0	0	0	1747	120	51	0
	C6	0	0	0	0	159	139	102	0
	C7	0	0	0	0	32	56	257	55
	C8	0	0	0	0	0	2	49	349

Table 4-18: The confusion matrix between the target and S-RNN classification model predicted remaining useful life classes for all the testing set examples.

		Predicted Class							
		C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**
Target Class	**C1**	547	62	14	0	0	0	0	0
	C2	228	113	59	0	0	0	0	0
	C3	21	76	279	24	0	0	0	0
	C4	0	0	50	350	0	0	0	0
	C5	2	0	0	0	1786	116	14	0
	C6	0	0	0	0	182	123	95	0
	C7	0	0	0	0	16	62	256	66
	C8	0	0	0	0	0	0	38	362

Table 4-19: The confusion matrix between the target and LSTM-RNN classification model predicted remaining useful life classes for all the testing set examples.

		Predicted Class							
		C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**
Target Class	**C1**	601	22	0	0	0	0	0	0
	C2	238	143	19	0	0	0	0	0
	C3	7	116	259	18	0	0	0	0
	C4	0	0	66	334	0	0	0	0
	C5	4	0	0	0	1858	56	0	0
	C6	0	0	0	0	68	292	40	0
	C7	0	0	0	0	0	47	349	4
	C8	0	0	0	0	0	0	48	352

Table 4-20: The confusion matrix between the target and GRU-RNN classification model predicted remaining useful life classes for all the testing set examples.

		Predicted Class							
		C1	**C2**	**C3**	**C4**	**C5**	**C6**	**C7**	**C8**
Target Class	**C1**	602	21	0	0	0	0	0	0
	C2	190	166	44	0	0	0	0	0
	C3	9	69	306	16	0	0	0	0
	C4	0	0	51	349	0	0	0	0
	C5	0	0	0	0	1820	98	0	0
	C6	0	0	0	0	46	335	19	0
	C7	0	0	0	0	0	25	372	3
	C8	0	0	0	0	0	0	42	358

From Table 4-13–Table 4-20 it can be concluded that the LSTM-RNN and GRU-RNN classification models drastically outperformed the FNN and S-RNN classification models on the training set and testing set as expected. The GRU-RNN classification model slightly outperformed the LSTM-RNN classification model and the S-RNN classification model significantly outperformed the FNN classification model. The predicted remaining useful life classes of the FNN, S-RNN, LSTM-RNN and GRU-RNN classification models were also significantly more accurate close to failure with less confusion between target and predicted classes, as the trendability of the condition monitoring measurements increased. It is important to point out that the there was very little confusion between the two different fault modes (C1–C4 for high pressure compressor degradation and C5–C8 for fan degradation) for all the classification models.

The performance of the FNN, S-RNN, LSTM-RNN and GRU-RNN classification models were also compared by calculating the cross-entropy and accuracy between the target and predicted remaining useful life classes for all the training set and testing set examples in the turbofan engine degradation data set. The cross-entropy and accuracy between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes for all the training set and testing set examples is shown in Table 4-21.

Table 4-21: The cross-entropy and accuracy between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes for all the training set and testing set examples.

Classification Model	**Training Set Cross-Entropy**	**Testing Set Cross-Entropy**	**Training Set Accuracy**	**Testing Set Accuracy**
FNN	0.5651	0.6567	0.7569	0.7403
S-RNN	0.4937	0.5448	0.7900	0.7723
LSTM-RNN	0.2341	0.3684	0.9101	0.8476
GRU-RNN	0.2580	0.3247	0.9047	0.8718

The performance of the FNN, S-RNN, LSTM-RNN and GRU-RNN classification models were also compared by recalculating the cross-entropy and accuracy between the target and predicted remaining useful life classes 120 cycles before failure for all the training set and testing set examples in the turbofan engine degradation data set. This is because it excluded the remaining useful life classes above the applied target threshold and was therefore more representative of the classification model performance close to failure. The cross-entropy and accuracy between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes 120 cycles before failure for all the training set and testing set examples is shown in Table 4-22.

Table 4-22: The cross-entropy and accuracy between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN classification model predicted remaining useful life classes 120 cycles before failure for all the training set and testing set examples.

Classification Model	Training Set Cross-Entropy	Testing Set Cross-Entropy	Training Set Accuracy	Testing Set Accuracy
FNN	0.8714	0.8944	0.5502	0.5529
S-RNN	0.7684	0.8190	0.6266	0.6179
LSTM-RNN	0.3619	0.6154	0.8526	0.7204
GRU-RNN	0.3682	0.5112	0.8589	0.7858

From Table 4-21 and Table 4-22 it can be concluded that the LSTM-RNN and GRU-RNN classification models drastically outperformed the FNN and S-RNN classification models on the training set and testing set as expected. The GRU-RNN classification model slightly outperformed the LSTM-RNN classification model and the S-RNN classification model significantly outperformed the FNN classification model. The difference between the training set and testing set cross-entropy and accuracy for the FNN, S-RNN, LSTM-RNN and GRU-RNN classification models 120 cycles before failure was also relatively small, which indicated good model regularization and an appropriately selected threshold for the prognostics classification strategy. The GRU-RNN classification model therefore had a very respectable accuracy of 0.7858 for the completely unseen testing set in the turbofan engine degradation data set 120 cycles before failure.

5 Prognostics Regression Strategy Results

5.1 General Asset Degradation Data Set Results

This section presents the results of the prognostics regression strategy and regression deep learning model architectures applied on the general asset degradation data set.

5.1.1 Strategy and Model Description

The applied threshold for the prognostics regression strategy was tuned and selected as $AT = 600$ cycles for the general asset degradation data set. The FNN, S-RNN, LSTM-RNN and GRU-RNN regression model architectures were trained on the training set of the general asset degradation data set with the Adam algorithm and regularized with a combination of the early stopping, weight decay and dropout regularization techniques.

5.1.2 Training Set Examples

The objective of this section is to present and compare how accurately the trained FNN, S-RNN, LSTM-RNN and GRU-RNN regression models could predict the remaining useful life values from the condition monitoring measurements for two randomly selected training set examples in the general asset degradation data set fully online. The two randomly selected training set examples represent two historical (previously seen) general assets that were run to failure.

It is important to point out that the trained FNN, S-RNN, LSTM-RNN and GRU-RNN regression models only used the condition monitoring measurements for the current time step (and previous time steps for the recurrent models) to predict the remaining useful life value for the current time step. The regression models therefore predicted the remaining useful life values from the condition monitoring measurements fully online. The presented S1 condition monitoring measurements were normalized with min-max scaling between 0 and 1 for the entire training and testing set. This was done to effectively present the variation in condition monitoring measurements across all the training and testing set examples.

Training Set Example 1

The S1 condition monitoring measurements versus time for training set example 1 are shown in Figure 5-1. The target and FNN, S-RNN, LSTM-RNN and GRU-RNN regression model predicted remaining useful life values versus time for training set example 1 are shown in Figure 5-2, Figure 5-3, Figure 5-4 and Figure 5-5 respectively.

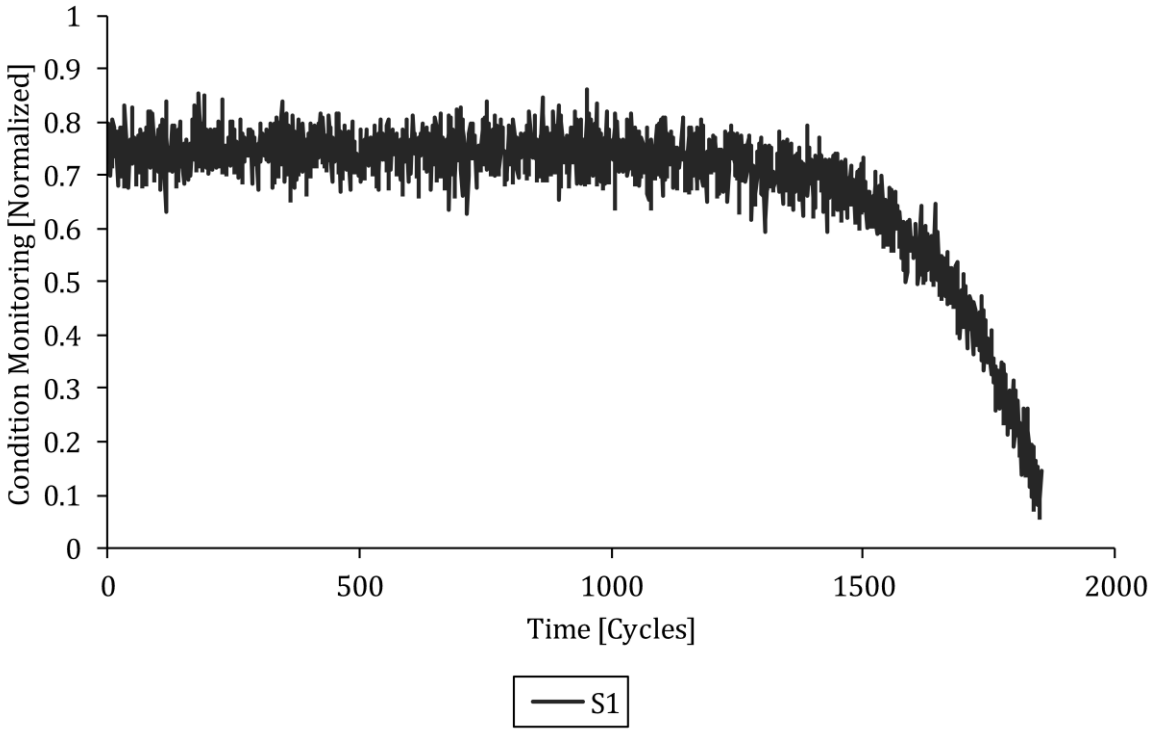

Figure 5-1: The S1 condition monitoring measurements versus time for training set example 1.

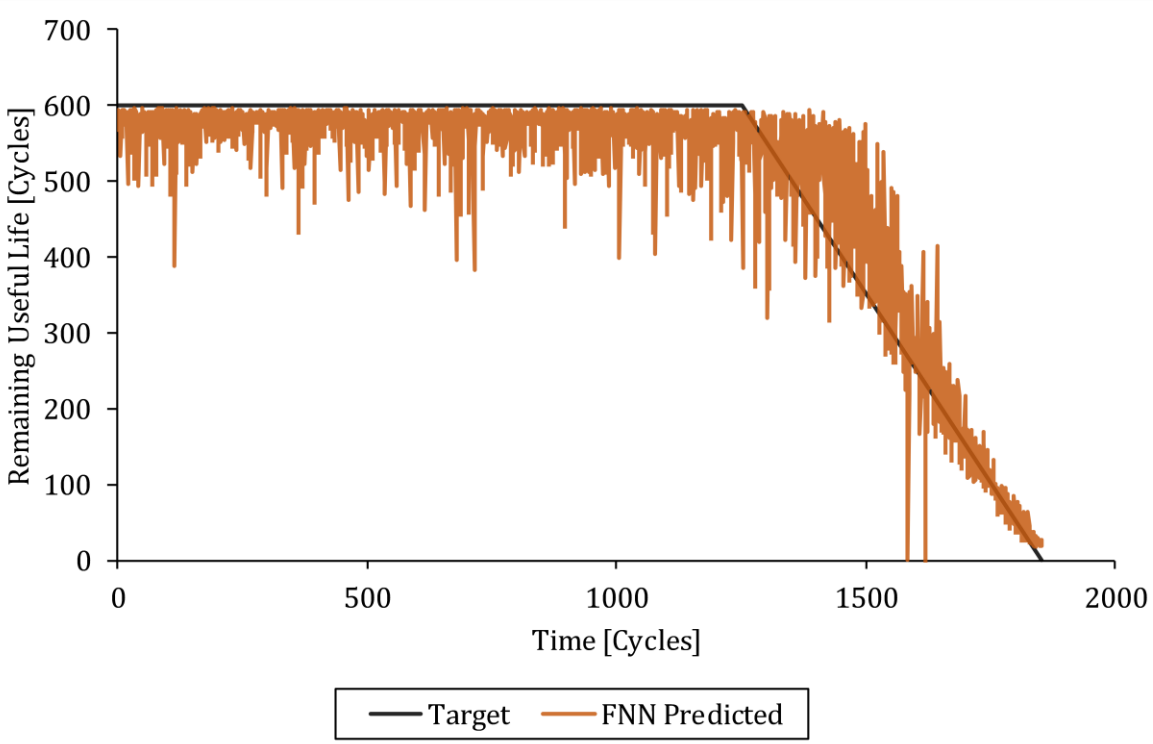

Figure 5-2: The target and FNN regression model predicted remaining useful life values versus time for training set example 1.

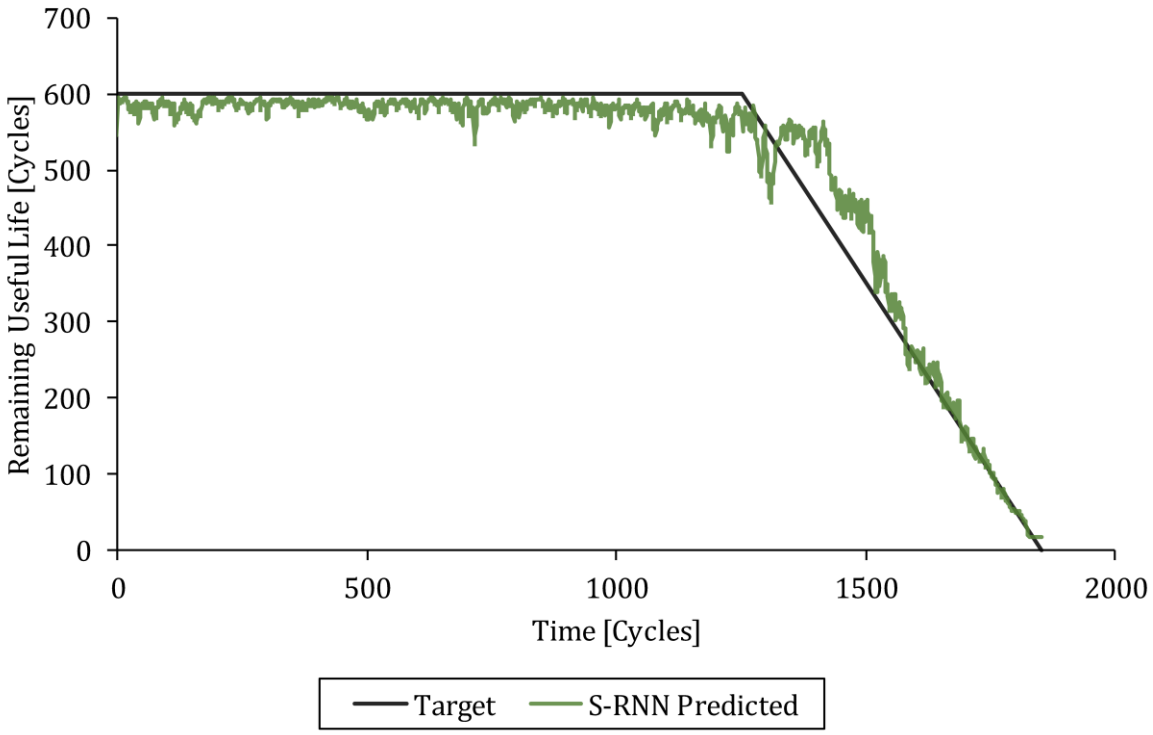

Figure 5-3: The target and S-RNN regression model predicted remaining useful life values versus time for training set example 1.

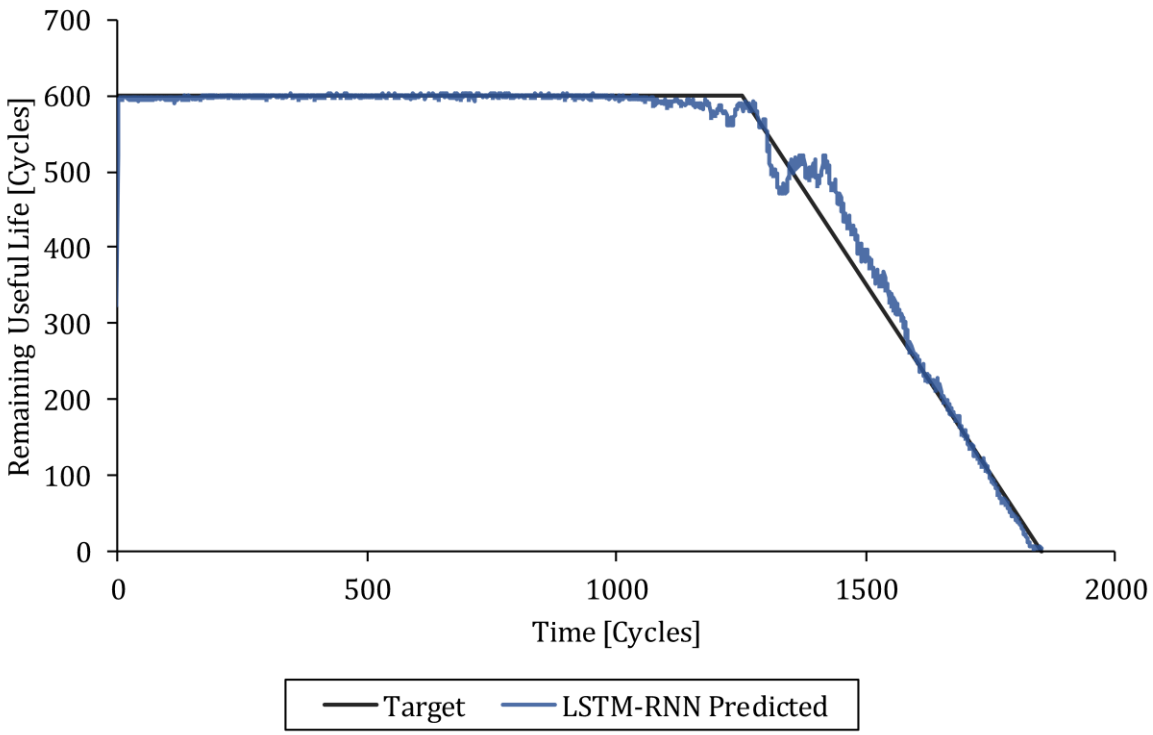

Figure 5-4: The target and LSTM-RNN regression model predicted remaining useful life values versus time for training set example 1.

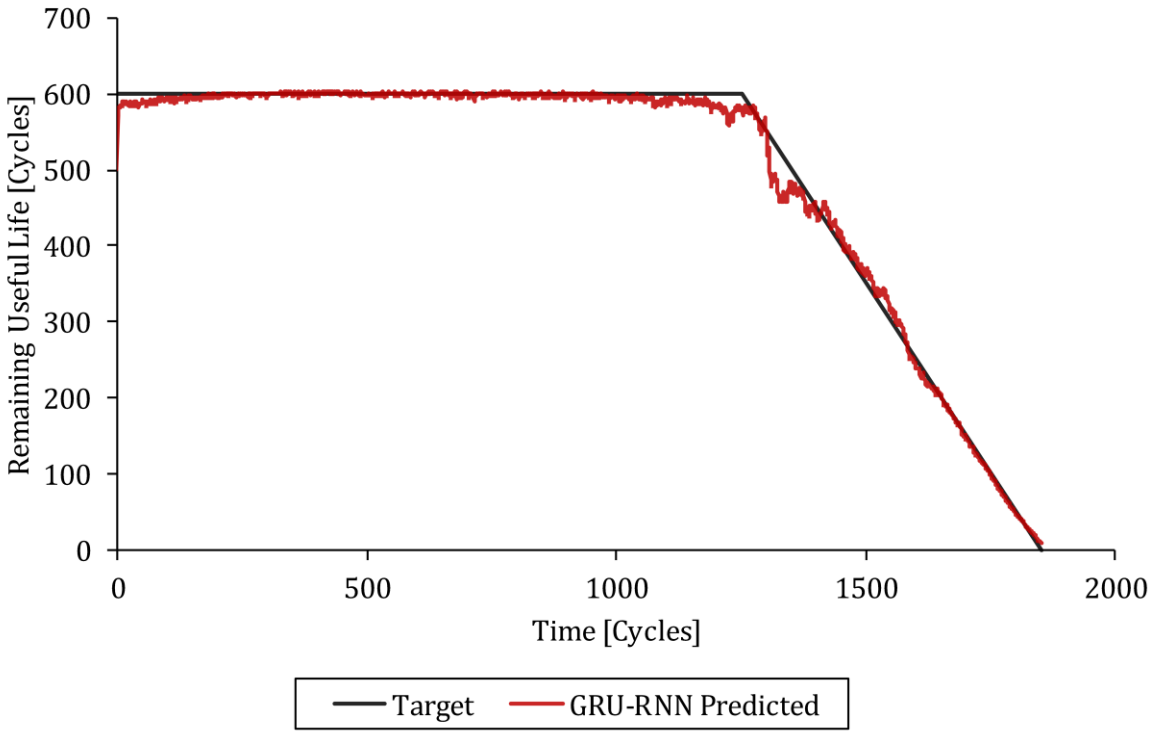

Figure 5-5: The target and GRU-RNN regression model predicted remaining useful life values versus time for training set example 1.

Training Set Example 2

The S1 condition monitoring measurements versus time for training set example 2 are shown in Figure 5-6. The target and FNN, S-RNN, LSTM-RNN and GRU-RNN regression model predicted remaining useful life values versus time for training set example 2 are shown in Figure 5-7, Figure 5-8, Figure 5-9 and Figure 5-10 respectively.

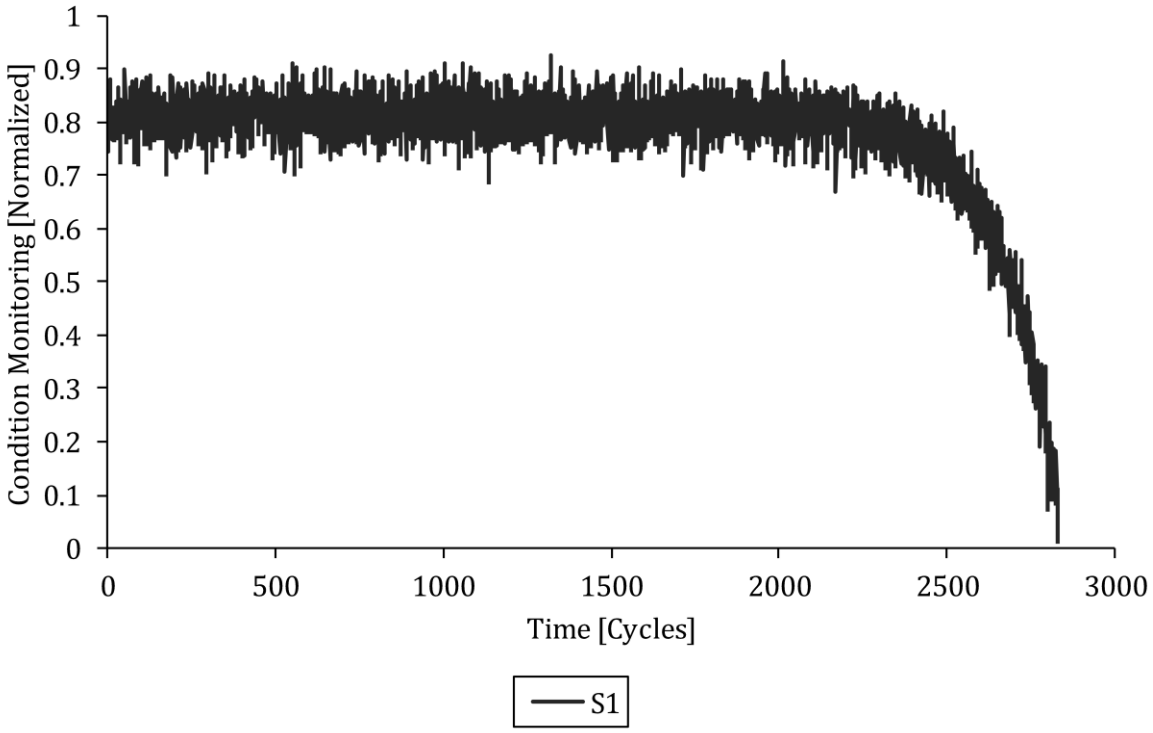

Figure 5-6: The S1 condition monitoring measurements versus time for training set example 2.

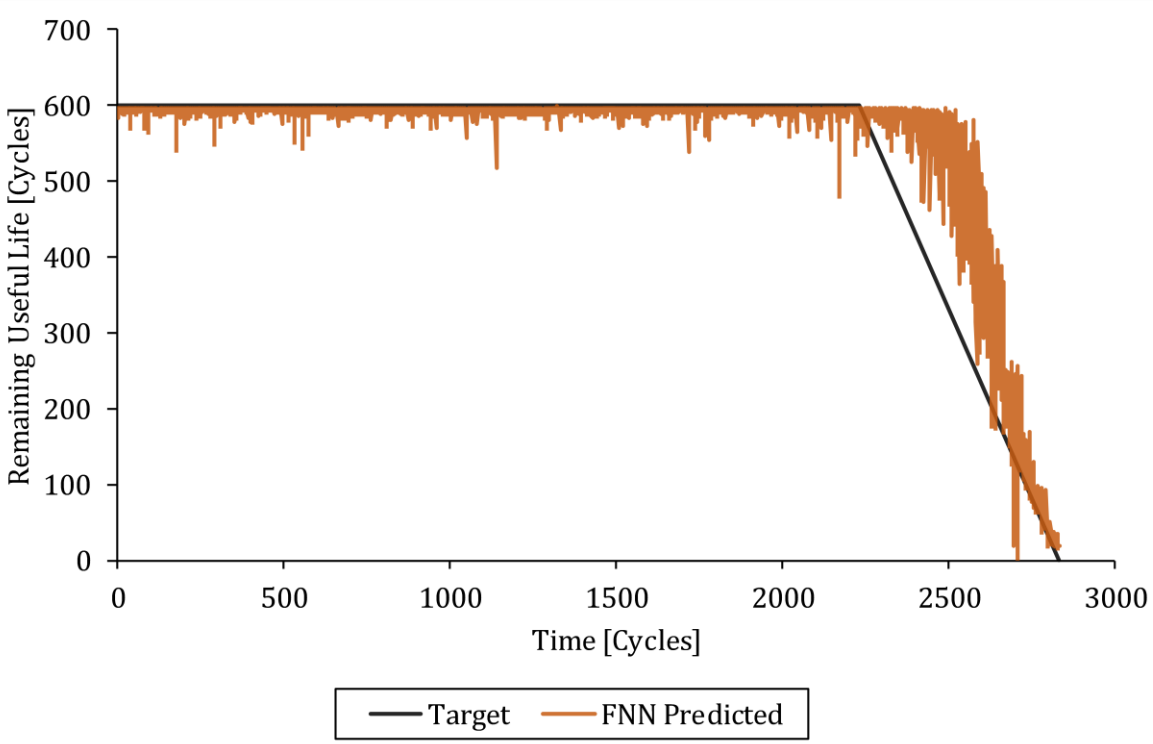

Figure 5-7: The target and FNN regression model predicted remaining useful life values versus time for training set example 2.

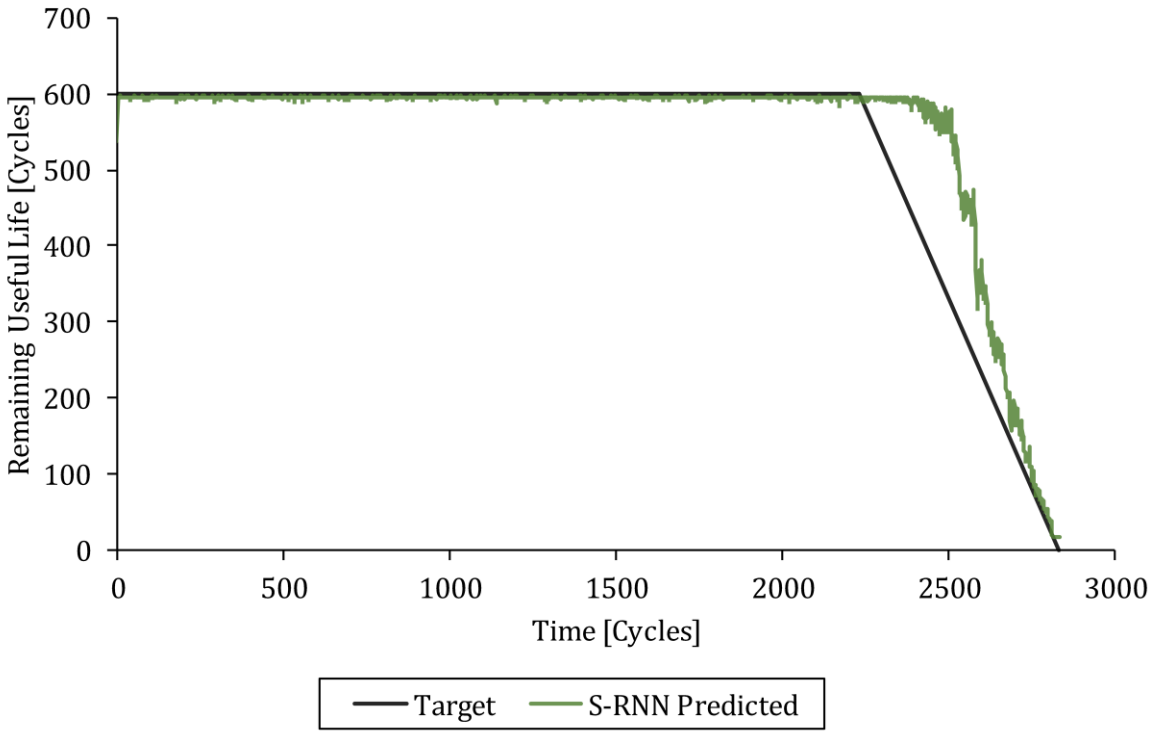

Figure 5-8: The target and S-RNN regression model predicted remaining useful life values versus time for training set example 2.

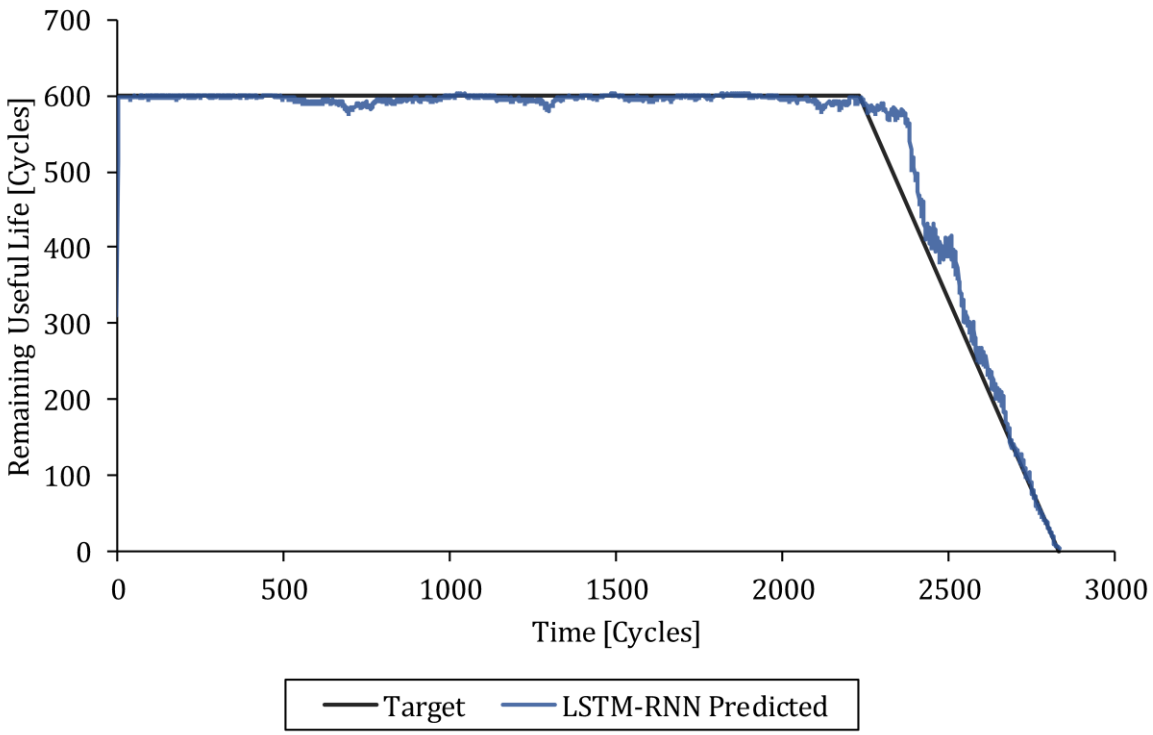

Figure 5-9: The target and LSTM-RNN regression model predicted remaining useful life values versus time for training set example 2.

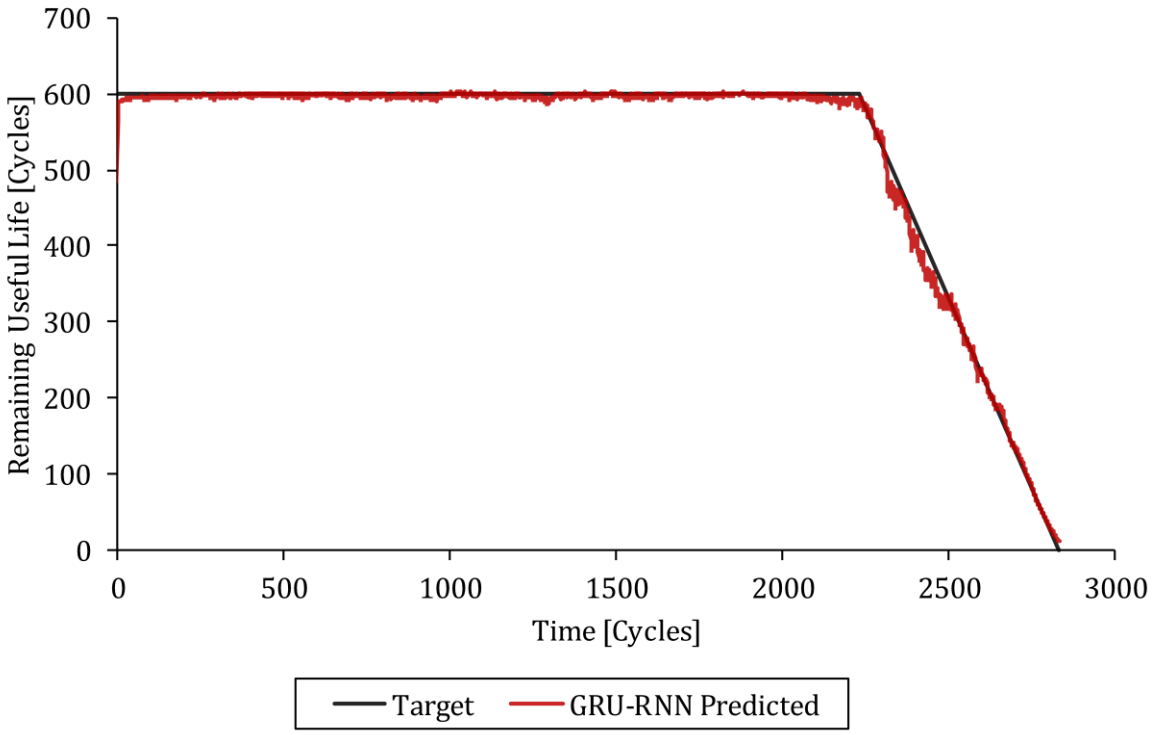

Figure 5-10: The target and GRU-RNN regression model predicted remaining useful life values versus time for training set example 2.

5.1.3 Testing Set Examples

The objective of this section is to present and compare how accurately the trained FNN, S-RNN, LSTM-RNN and GRU-RNN regression models could predict the remaining useful life values from the condition monitoring measurements for two randomly selected testing set examples in the general asset degradation data set fully online. The two randomly selected testing set examples represent two future (completely unseen) general assets that were run to failure.

Testing Set Example 1

The S1 condition monitoring measurements versus time for testing set example 1 are shown in Figure 5-11. The target and FNN, S-RNN, LSTM-RNN and GRU-RNN regression model predicted remaining useful life values versus time for testing set example 1 are shown in Figure 5-12, Figure 5-13, Figure 5-14 and Figure 5-15 respectively.

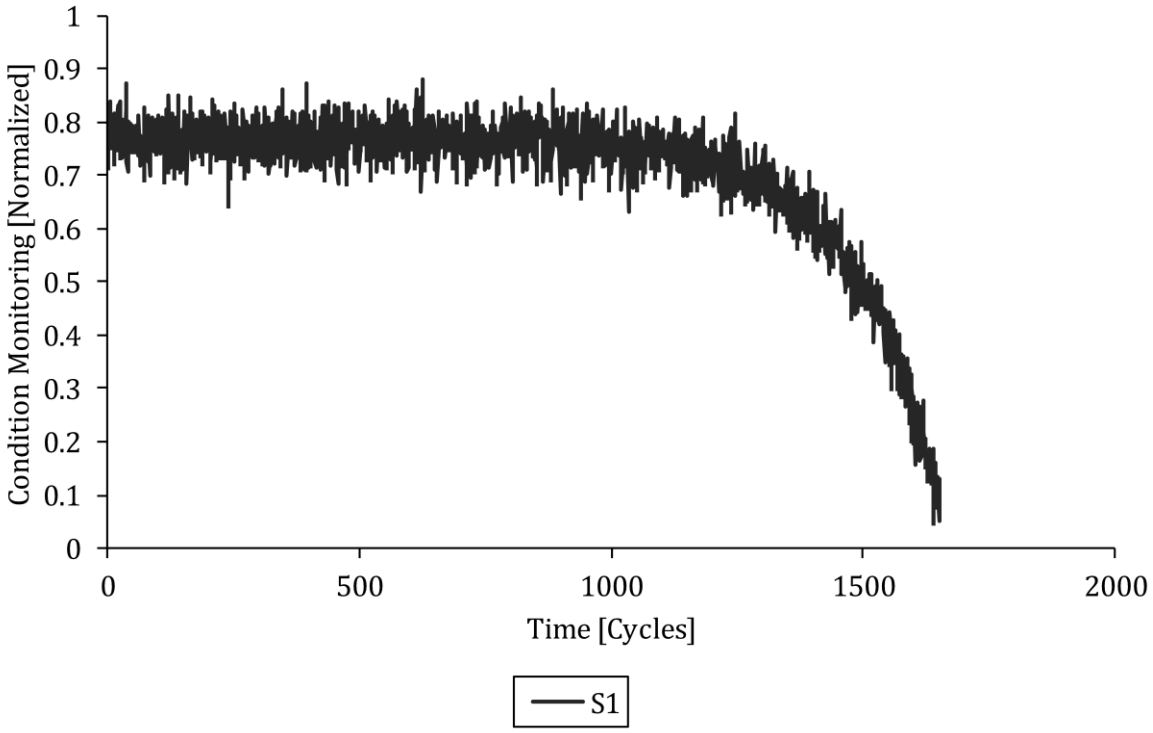

Figure 5-11: The S1 condition monitoring measurements versus time for testing set example 1.

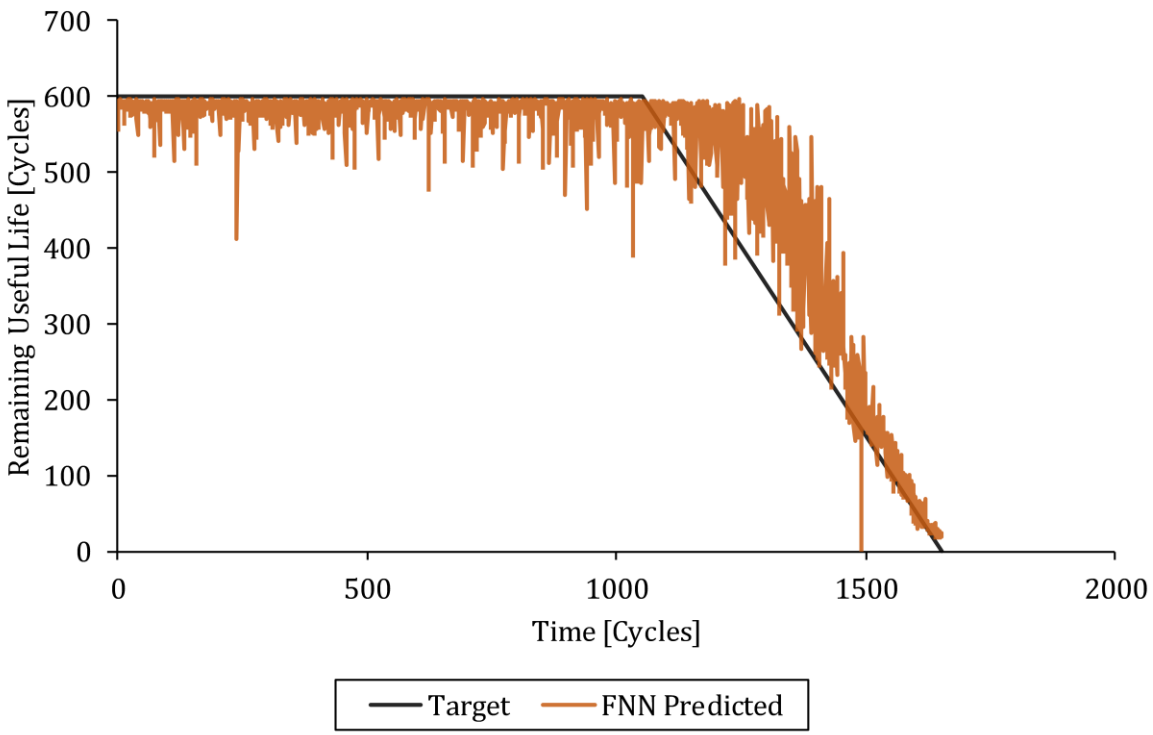

Figure 5-12: The target and FNN regression model predicted remaining useful life values versus time for testing set example 1.

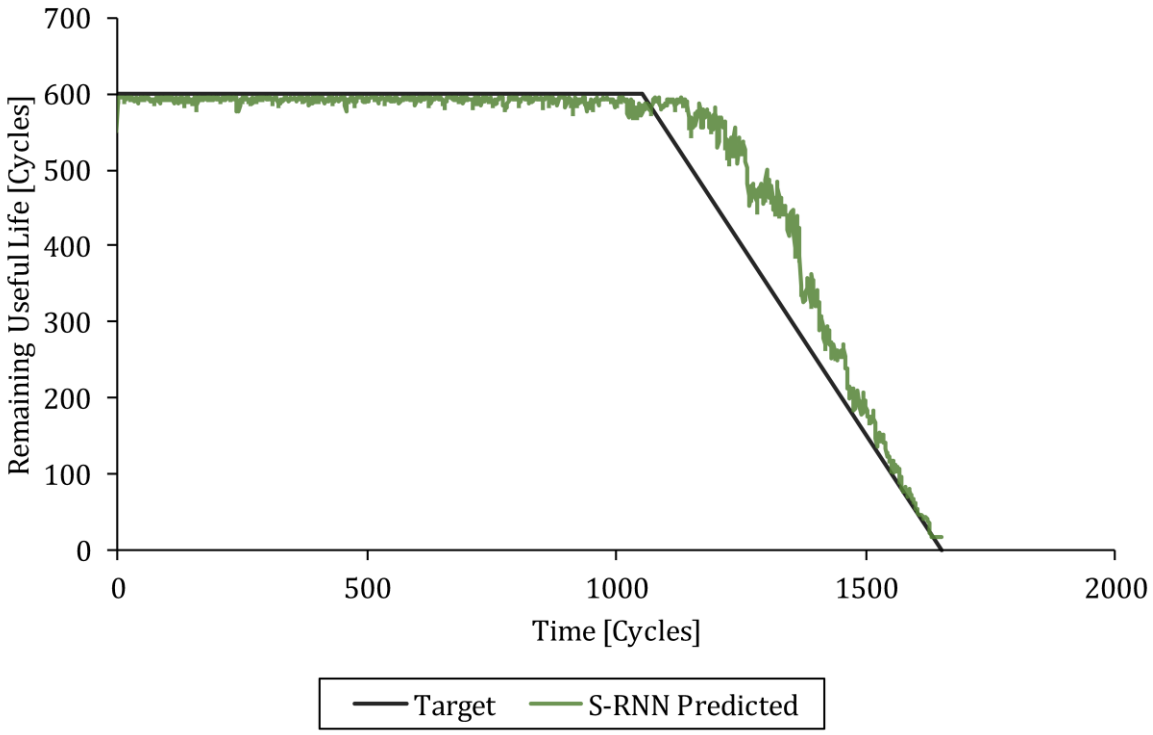

Figure 5-13: The target and S-RNN regression model predicted remaining useful life values versus time for testing set example 1.

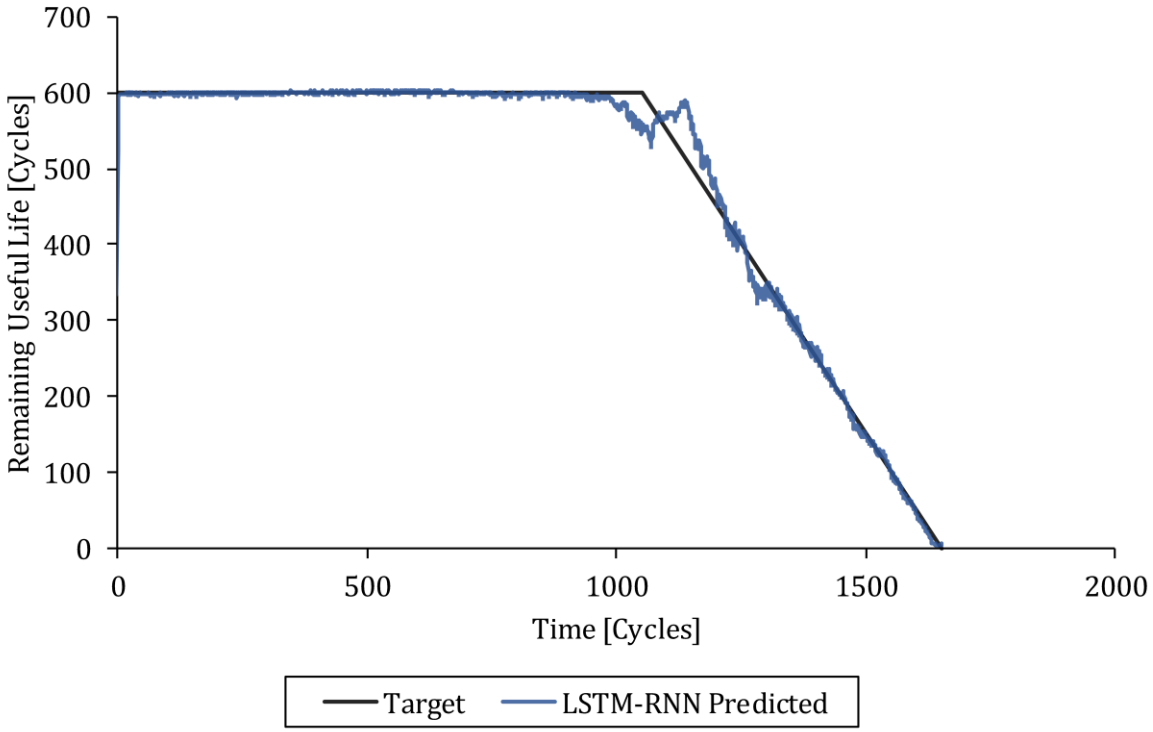

Figure 5-14: The target and LSTM-RNN regression model predicted remaining useful life values versus time for testing set example 1.

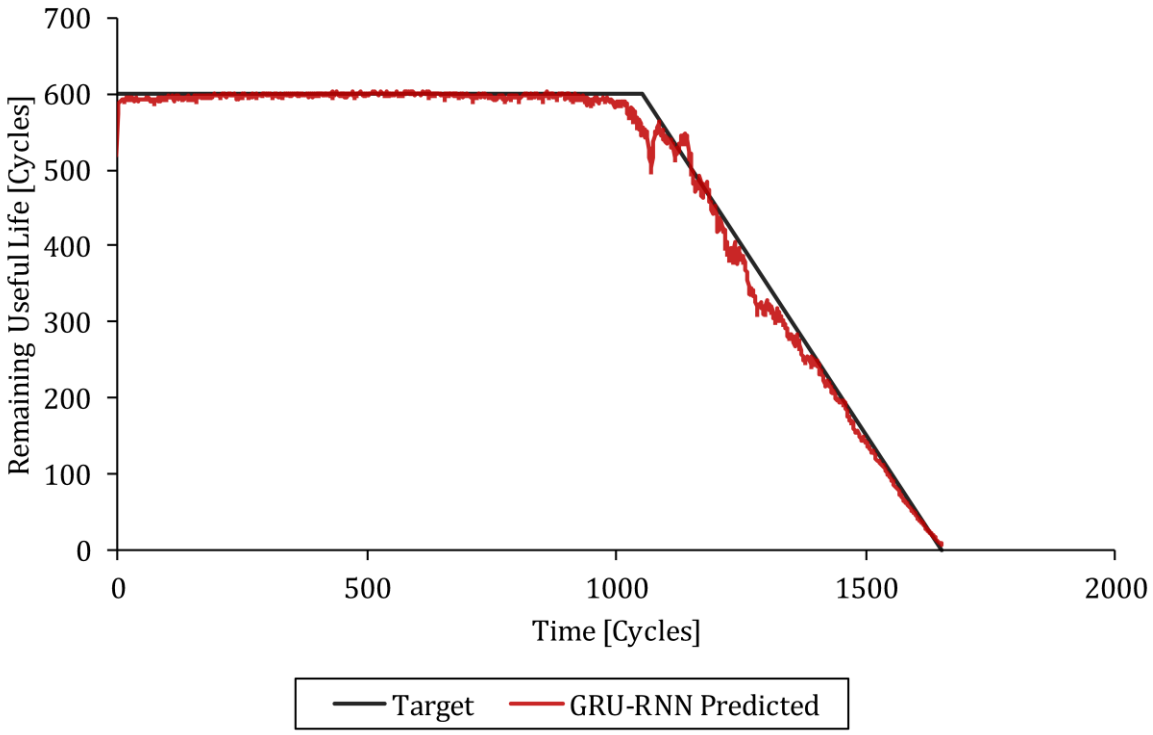

Figure 5-15: The target and GRU-RNN regression model predicted remaining useful life values versus time for testing set example 1.

Testing Set Example 2

The S1 condition monitoring measurements versus time for testing set example 2 are shown in Figure 5-16. The target and FNN, S-RNN, LSTM-RNN and GRU-RNN regression model predicted remaining useful life values versus time for testing set example 2 are shown in Figure 5-17, Figure 5-18, Figure 5-19 and Figure 5-20 respectively.

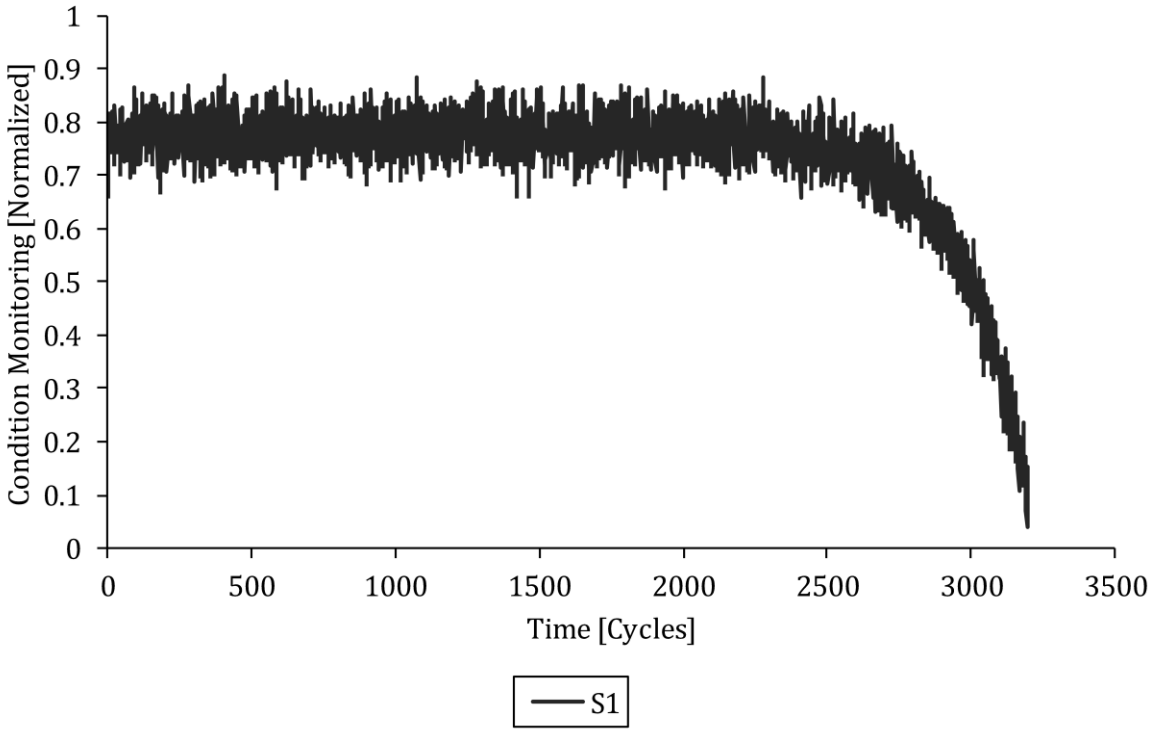

Figure 5-16: The S1 condition monitoring measurements versus time for testing set example 2.

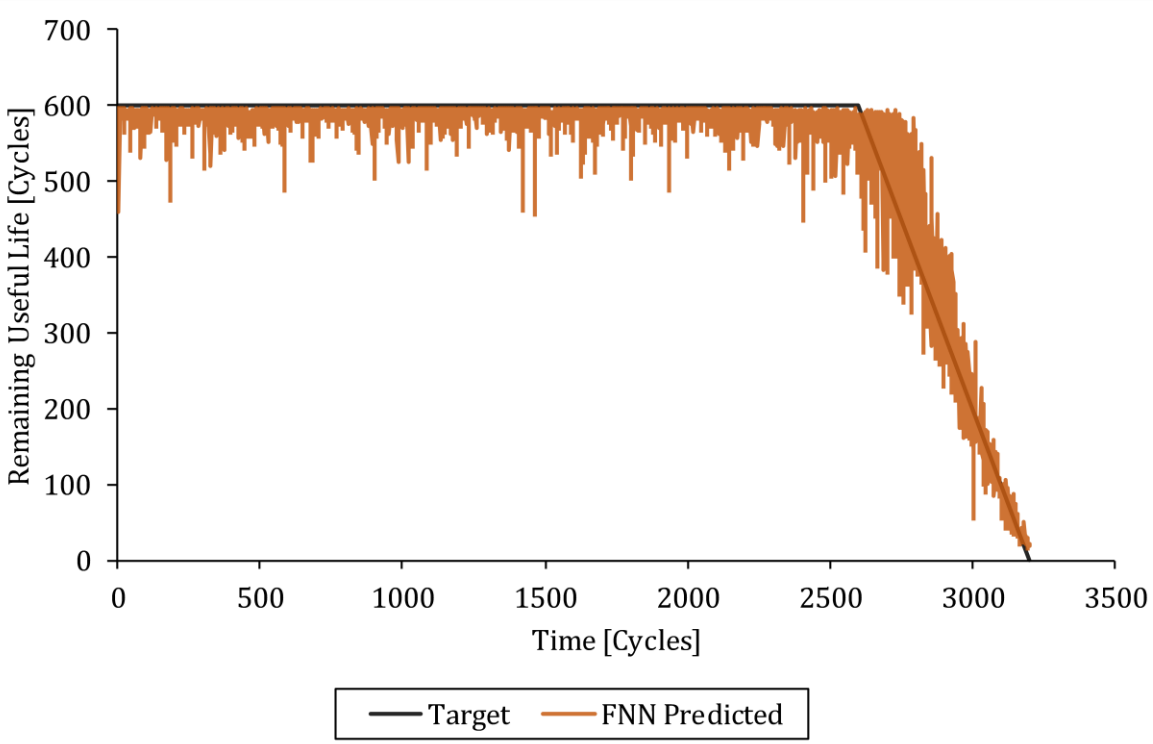

Figure 5-17: The target and FNN regression model predicted remaining useful life values versus time for testing set example 2.

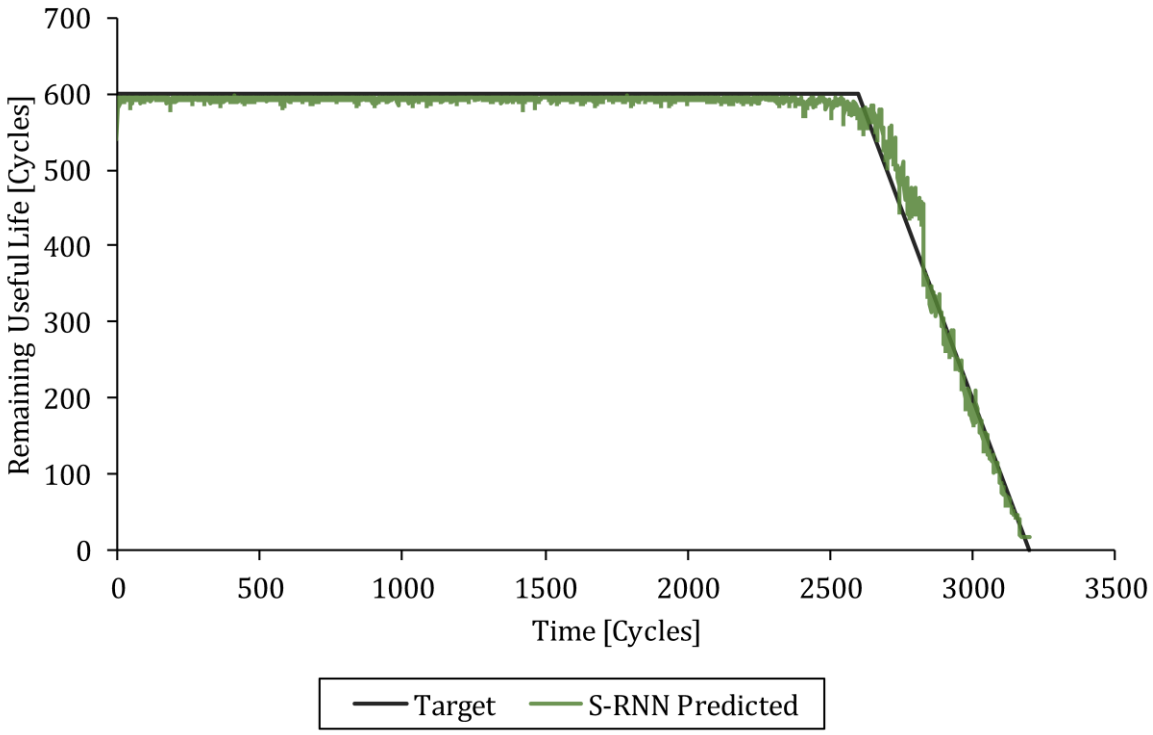

Figure 5-18: The target and S-RNN regression model predicted remaining useful life values versus time for testing set example 2.

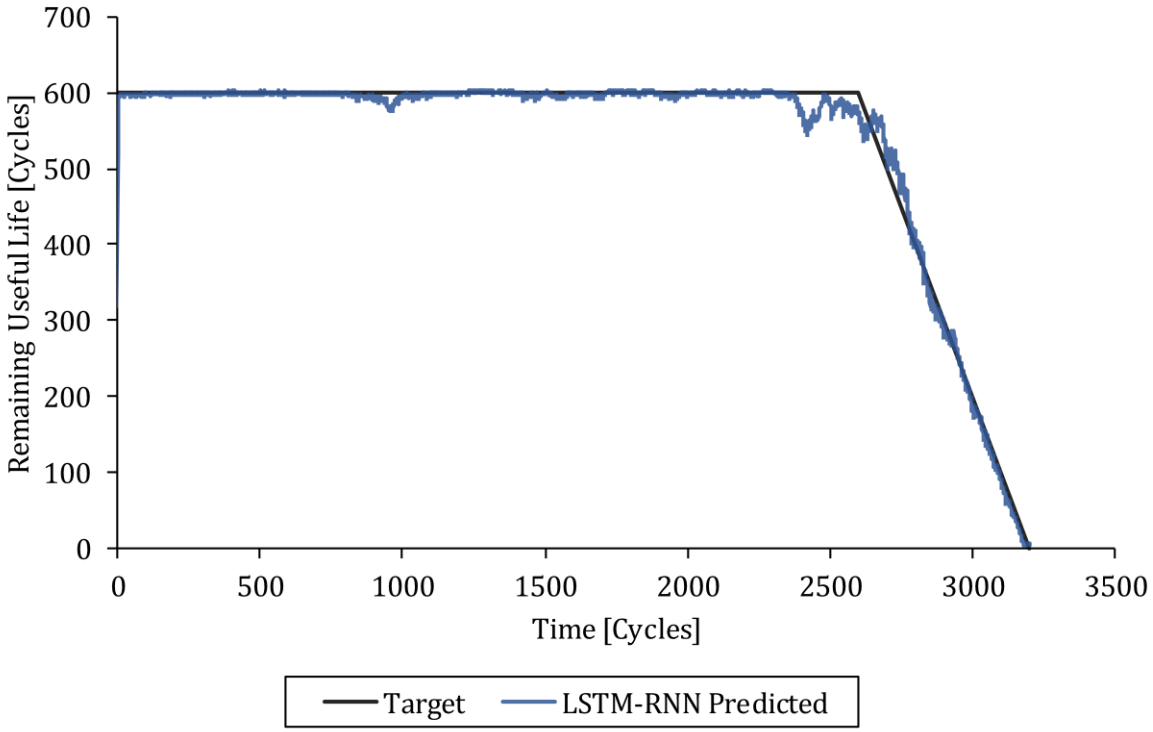

Figure 5-19: The target and LSTM-RNN regression model predicted remaining useful life values versus time for testing set example 2.

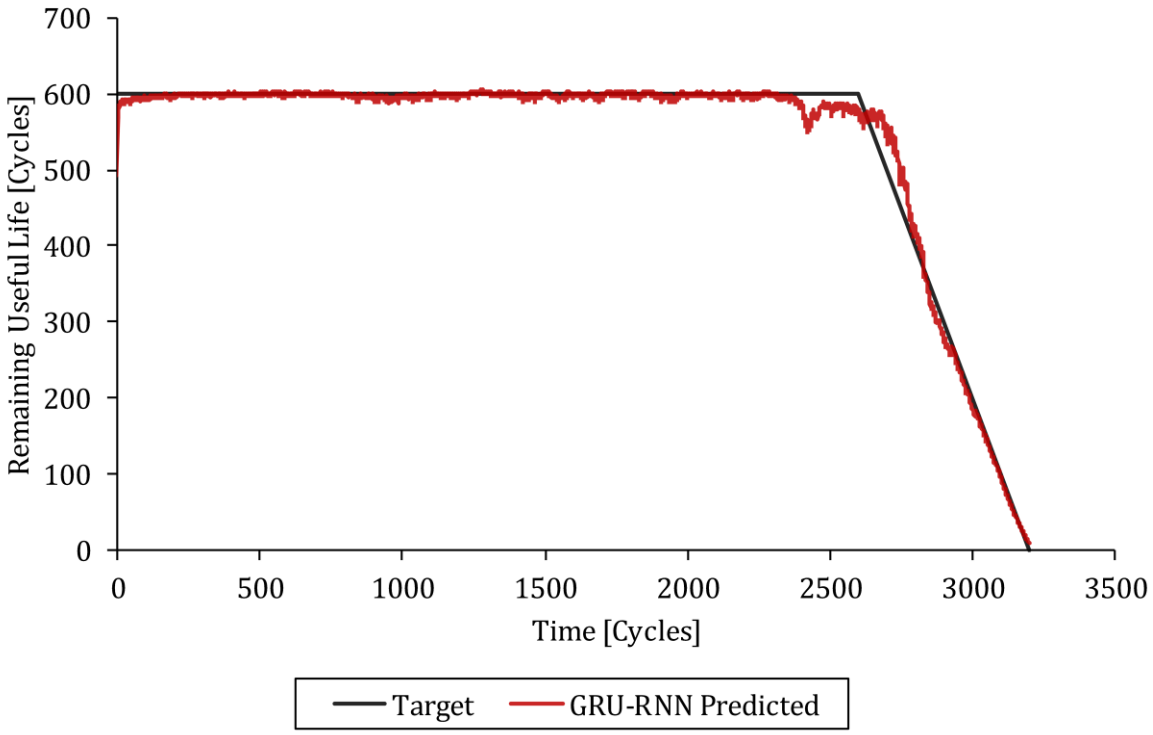

Figure 5-20: The target and GRU-RNN regression model predicted remaining useful life values versus time for testing set example 2.

From Figure 5-1–Figure 5-20 it can be concluded that the prognostics regression strategy and trained FNN, S-RNN, LSTM-RNN and GRU-RNN regression models can successfully predict the remaining useful life values from the condition monitoring measurements for the two randomly selected training and testing set examples in the general asset degradation data set fully online. The LSTM-RNN and GRU-RNN regression models drastically outperformed the FNN and S-RNN regression models on the two randomly selected training and testing set examples as expected. The GRU-RNN regression model slightly outperformed the LSTM-RNN regression model and the S-RNN regression model significantly outperformed the FNN regression model on average. The predicted remaining useful life values of the FNN regression model were also drastically more noisy than that of the S-RNN, LSTM-RNN and GRU-RNN regression models. The predicted remaining useful life values of the FNN, S-RNN, LSTM-RNN and GRU-RNN regression models were also significantly more accurate close to failure as the trendability of the condition monitoring measurements increased.

5.1.4 Model Performance Comparison

The performance of the FNN, S-RNN, LSTM-RNN and GRU-RNN regression models were compared by calculating the mean squared error and mean absolute error between the target and predicted remaining useful life values for all the training set and testing set examples in the general asset degradation data set. The mean squared error and mean absolute error between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN regression model predicted remaining useful life values for all the training set and testing set examples is shown in Table 5-1.

Table 5-1: The mean squared error and mean absolute error between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN regression model predicted remaining useful life values, for all the training set and testing set examples.

Regression Model	Training Set Mean Squared Error	Testing Set Mean Squared Error	Training Set Mean Absolute Error	Testing Set Mean Absolute Error
FNN	2938.1995	3200.3494	28.4593	30.1844
S-RNN	2138.7568	2332.6064	22.5138	24.2983
LSTM-RNN	478.8184	548.7399	9.6622	10.5086
GRU-RNN	343.2641	425.7216	8.3489	9.7803

The performance of the FNN, S-RNN, LSTM-RNN and GRU-RNN regression models were also compared by recalculating the mean squared error and mean absolute error between the target and predicted remaining useful life values 600 cycles before failure for all the training set and testing set examples in the general asset degradation data set. This is because it excluded the remaining useful life values above the applied target threshold and was therefore more representative of the regression model performance close to failure. The mean squared error and mean absolute error between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN regression model predicted remaining useful life values 600 cycles before failure for all the training set and testing set examples is shown in Table 5-2.

Table 5-2: The mean squared error and mean absolute error between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN regression model predicted remaining useful life values 600 cycles before failure for all the training set and testing set examples.

Regression Model	**Training Set Mean Squared Error**	**Testing Set Mean Squared Error**	**Training Set Mean Absolute Error**	**Testing Set Mean Absolute Error**
FNN	9530.8240	10081.4460	69.5024	72.6187
S-RNN	7524.6170	8073.2160	60.5562	64.4089
LSTM-RNN	1644.8540	1835.0643	26.7006	28.5671
GRU-RNN	1214.5525	1497.8778	21.2855	26.1174

From Table 5-1 and Table 5-2 it can be concluded that the LSTM-RNN and GRU-RNN regression models drastically outperformed the FNN and S-RNN regression models on the training set and testing set as expected. The GRU-RNN regression model slightly outperformed the LSTM-RNN regression model and the S-RNN regression model significantly outperformed the FNN regression model. The difference between the training set and testing set mean squared error and mean absolute error for the FNN, S-RNN, LSTM-RNN and GRU-RNN regression models 600 cycles before failure was also relatively small, which indicated good model regularization and an appropriately selected threshold for the prognostics regression strategy. The GRU-RNN regression model therefore made a very respectable mean absolute error of only 26.1174 cycles for the completely unseen testing set in the general asset degradation data set 600 cycles before failure.

5.2 Turbofan Engine Degradation Data Set

This section presents the results of the prognostics regression strategy and regression deep learning model architectures applied on the turbofan engine degradation data set.

5.2.1 Strategy and Model Description

The applied threshold for the prognostics regression strategy was tuned and selected as $AT = 120$ cycles for the turbofan engine degradation data set. The FNN, S-RNN, LSTM-RNN and GRU-RNN regression model architectures were trained on the training set of the turbofan engine degradation data set with the Adam algorithm and regularized with a combination of the early stopping, weight decay and dropout regularization techniques.

5.2.2 Training Set Examples

The objective of this section is to present and compare how accurately the trained FNN, S-RNN, LSTM-RNN and GRU-RNN regression models could predict the remaining useful life values from the condition monitoring measurements for two randomly selected training set examples in the turbofan engine degradation data set fully online. The two randomly selected training set examples had different fault modes and represent two historical (previously seen) turbofan engines that were run to failure.

It is important to point out that the trained FNN, S-RNN, LSTM-RNN and GRU-RNN regression models only used the condition monitoring measurements for the current time step (and previous time steps for the recurrent models) to predict the remaining useful life value for the current time step. The regression models therefore predicted the remaining useful life values from the condition monitoring measurements fully online. The presented S6, S10, S12, S18 and S24 condition monitoring measurements were normalized with min-max scaling between 0 and 1 for the entire training and testing set. This was done to effectively present the variation in condition monitoring measurements across all the training and testing set examples. It is important to point out that the regression models were however trained and tested on all S1-S24 condition monitoring measurements for the turbofan engine degradation data set.

Training Set Example 1

The S6, S10, S12, S18 and S24 condition monitoring measurements versus time for training set example 1 are shown in Figure 5-21. The target and FNN, S-RNN, LSTM-RNN and GRU-RNN regression model predicted remaining useful life values versus time for training set example 1 are shown in Figure 5-22, Figure 5-23, Figure 5-24 and Figure 5-25 respectively. The fault mode for training set example 1 is high-pressure compressor degradation.

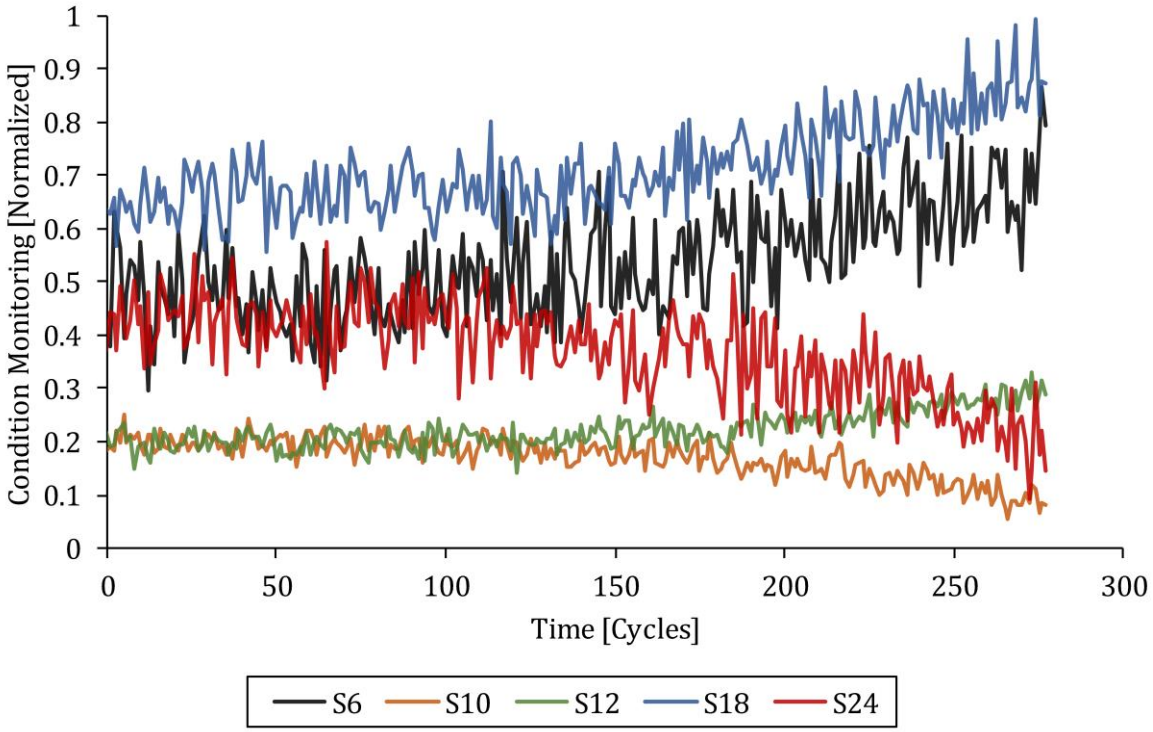

Figure 5-21: The S6, S10, S12, S18 and S24 condition monitoring measurements versus time for training set example 1.

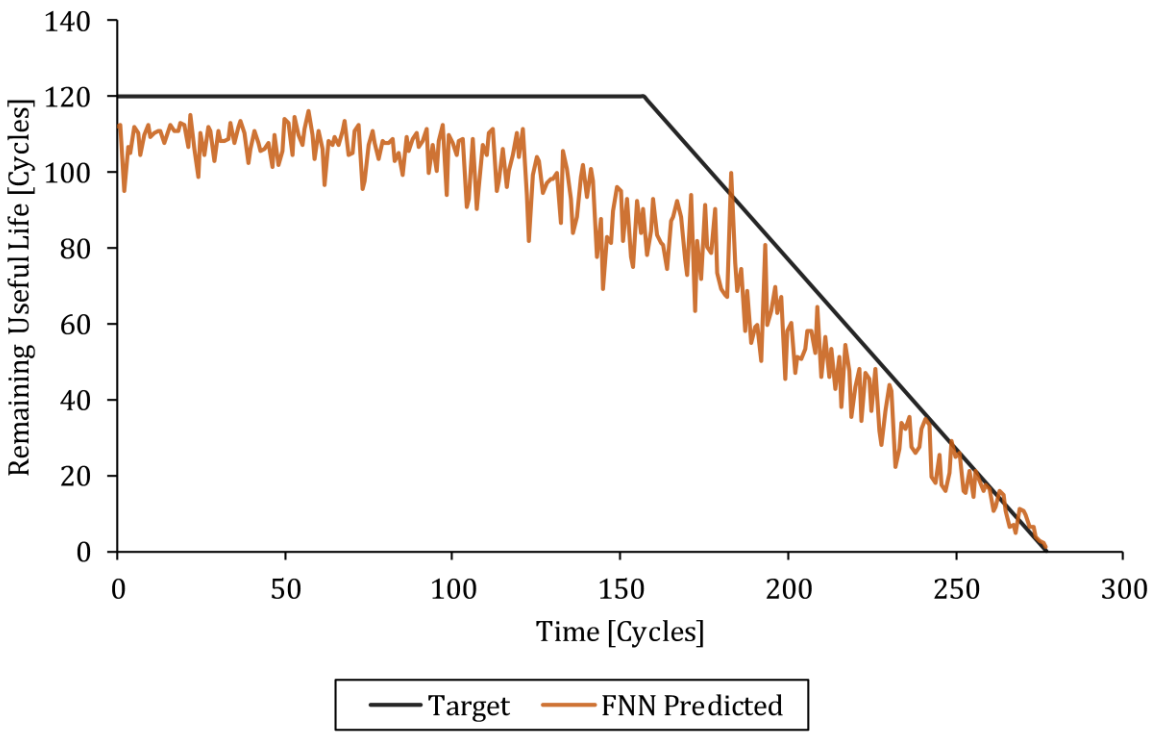

Figure 5-22: The target and FNN regression model predicted remaining useful life values versus time for training set example 1.

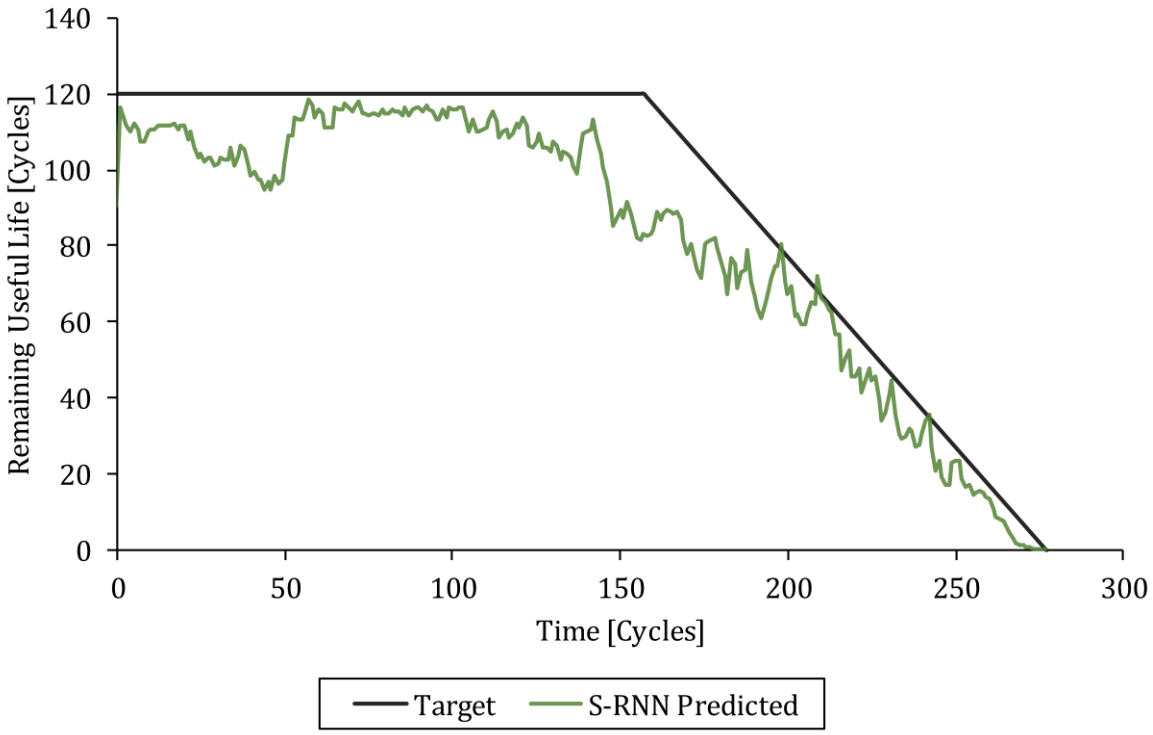

Figure 5-23: The target and S-RNN regression model predicted remaining useful life values versus time for training set example 1.

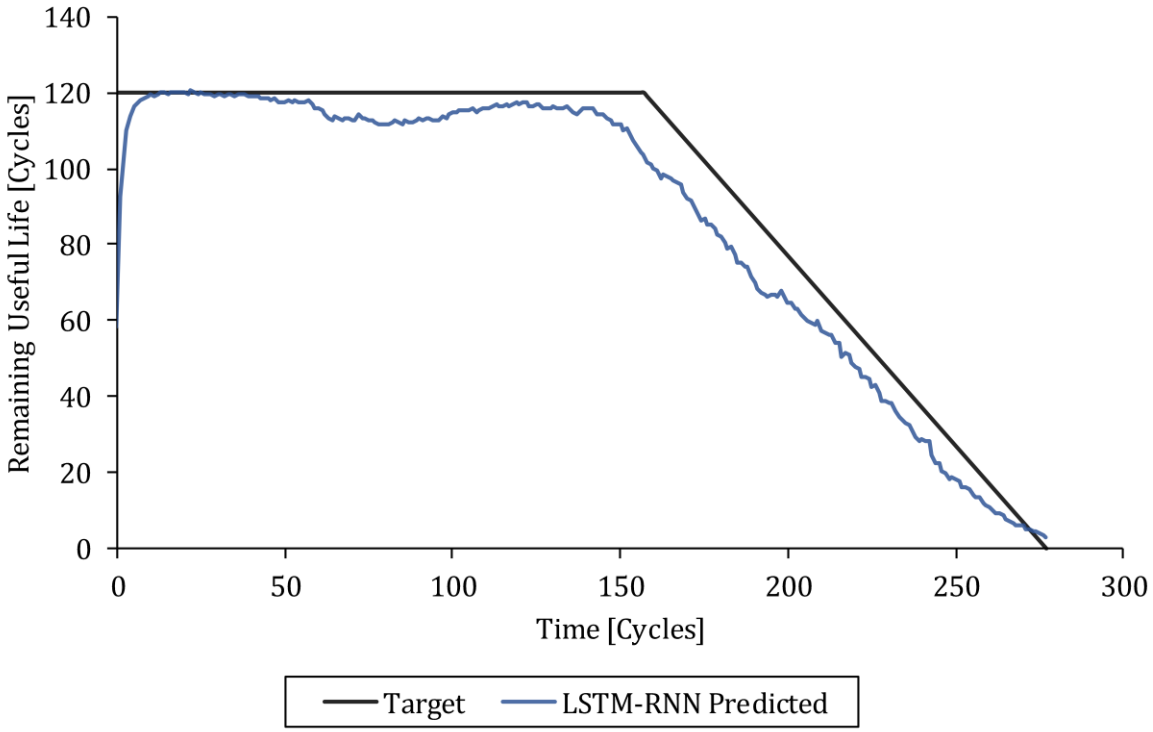

Figure 5-24: The target and LSTM-RNN regression model predicted remaining useful life values versus time for training set example 1.

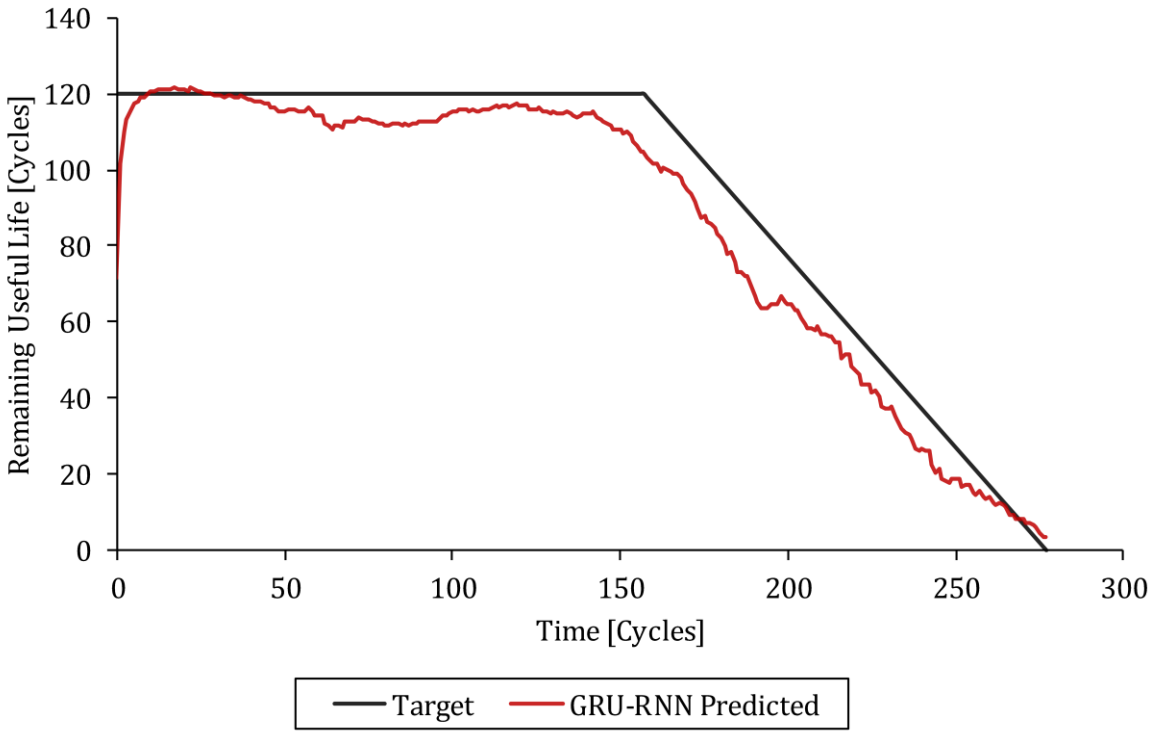

Figure 5-25: The target and GRU-RNN regression model predicted remaining useful life values versus time for training set example 1.

Training Set Example 2

The S6, S10, S12, S18 and S24 condition monitoring measurements versus time for training set example 2 are shown in Figure 5-26. The target and FNN, S-RNN, LSTM-RNN and GRU-RNN regression model predicted remaining useful life values versus time for training set example 2 are shown in Figure 5-27, Figure 5-28, Figure 5-29 and Figure 5-30 respectively. The fault mode for training set example 2 is fan degradation.

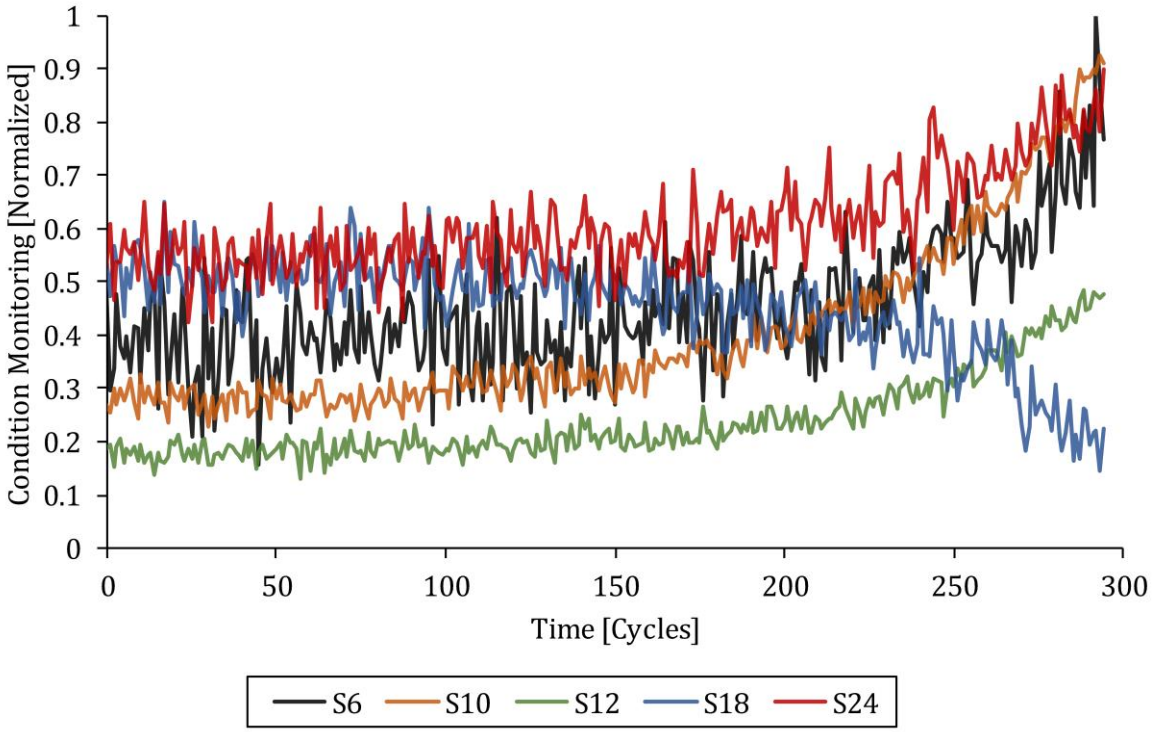

Figure 5-26: The S6, S10, S12, S18 and S24 condition monitoring measurements versus time for training set example 2.

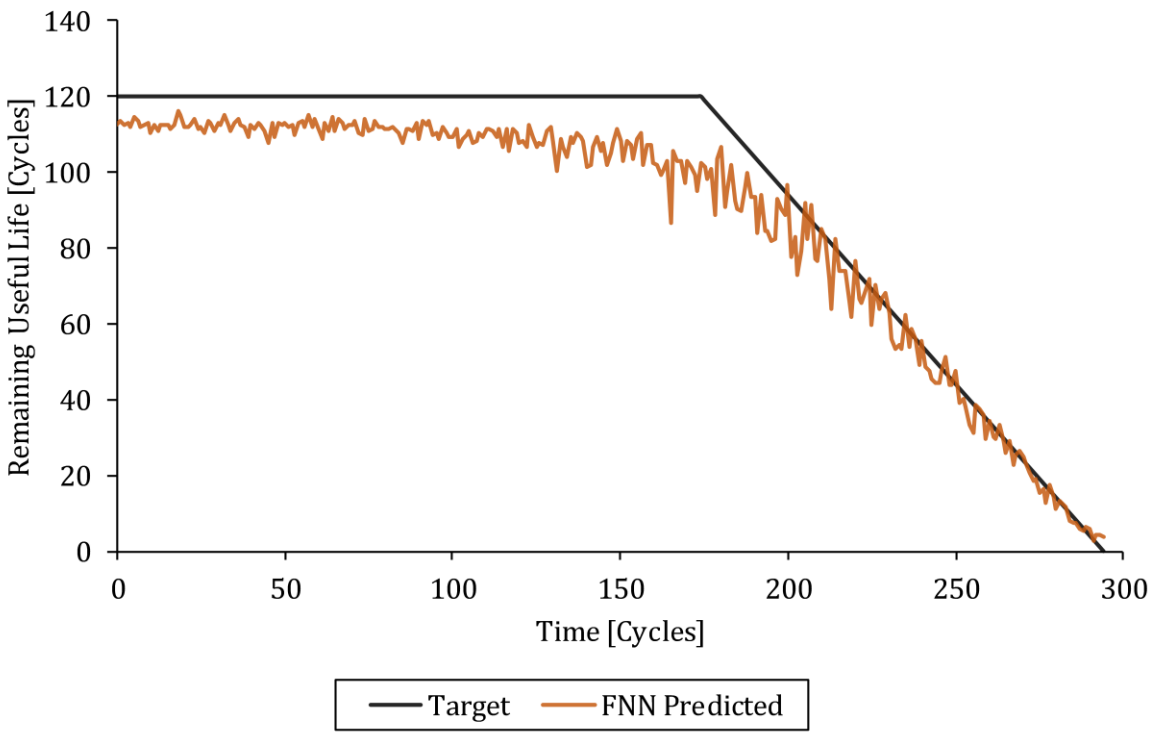

Figure 5-27: The target and FNN regression model predicted remaining useful life values versus time for training set example 2.

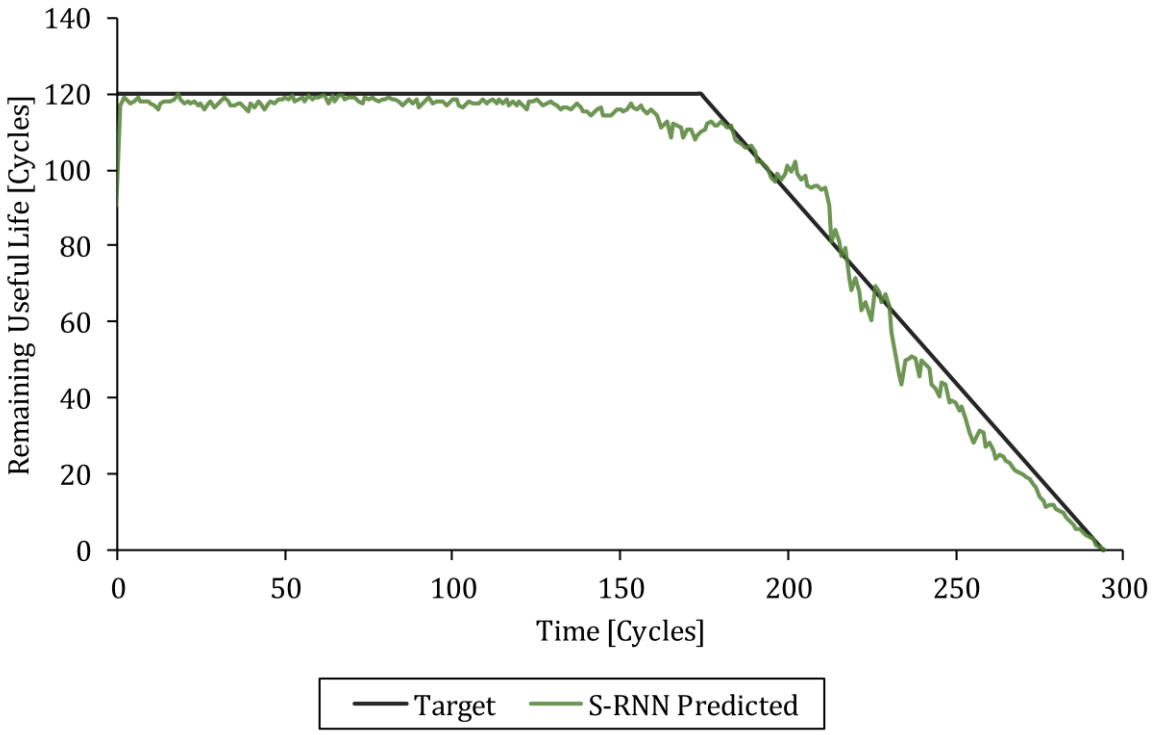

Figure 5-28: The target and S-RNN regression model predicted remaining useful life values versus time for training set example 2.

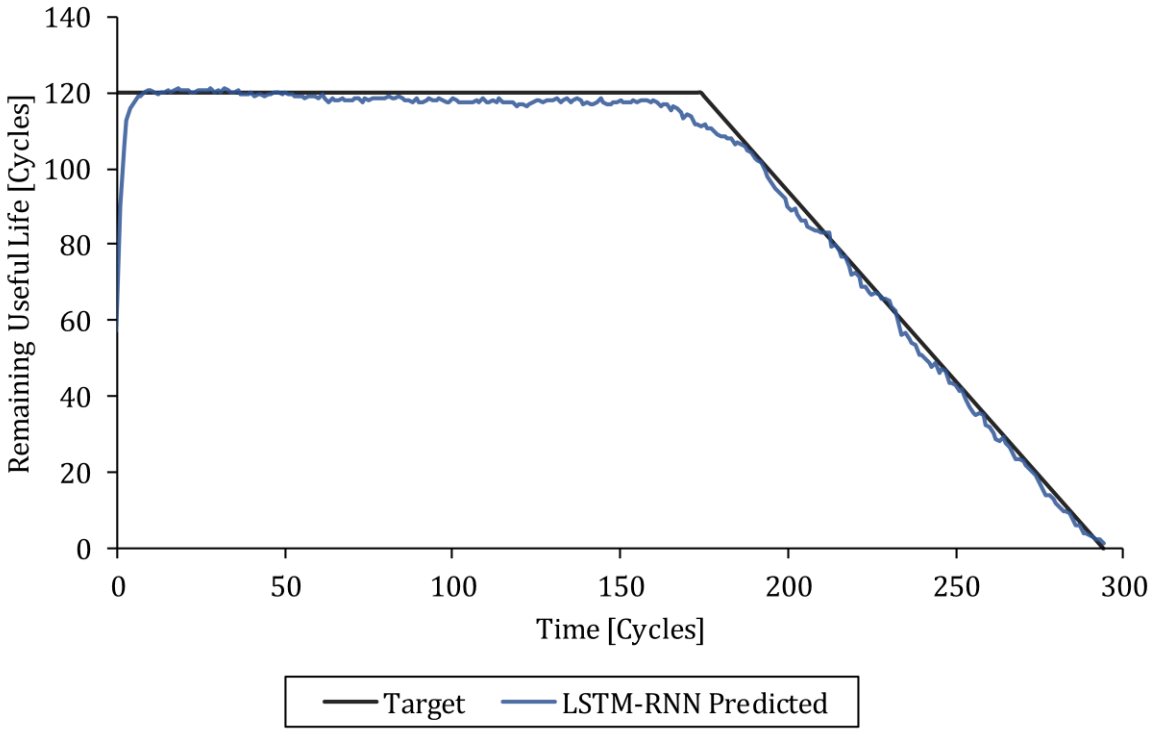

Figure 5-29: The target and LSTM-RNN regression model predicted remaining useful life values versus time for training set example 2.

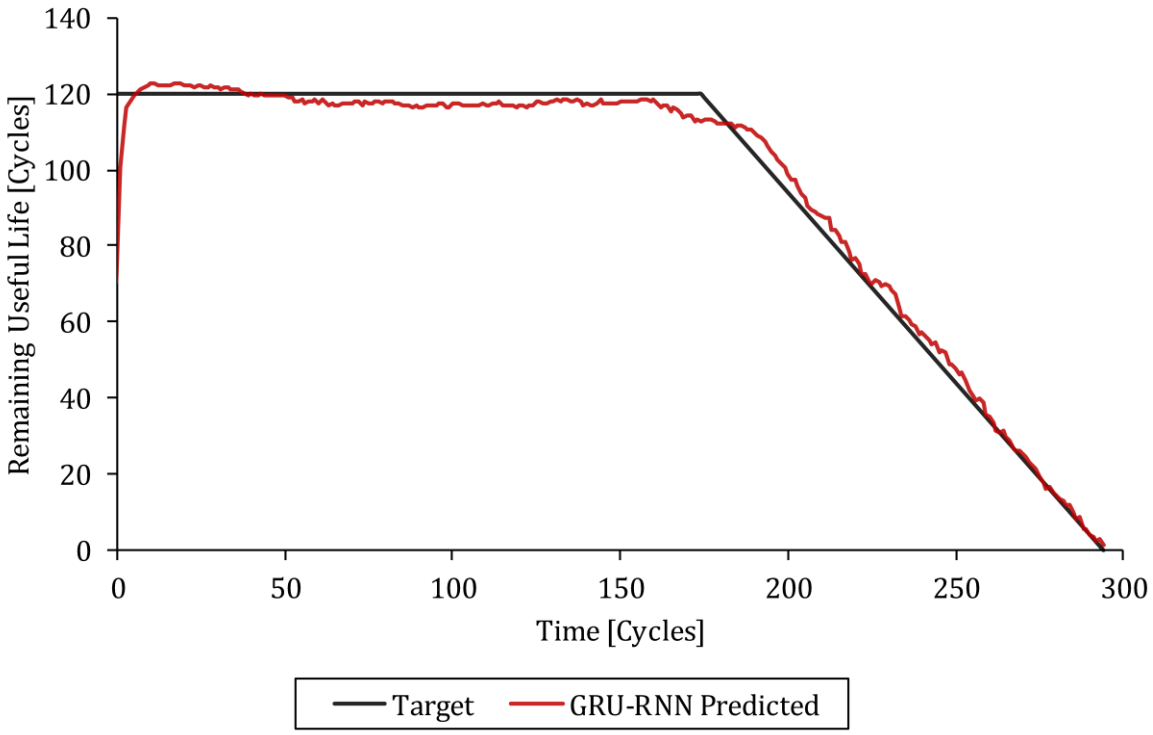

Figure 5-30: The target and GRU-RNN regression model predicted remaining useful life values versus time for training set example 2.

5.2.3 Testing Set Examples

The objective of this section is to present and compare how accurately the trained FNN, S-RNN, LSTM-RNN and GRU-RNN regression models could predict the remaining useful life values from the condition monitoring measurements for two randomly selected testing set examples in the turbofan engine degradation data set fully online. The two randomly selected testing set examples had different fault modes and represent two future (completely unseen) turbofan engines that were run to failure.

Testing Set Example 1

The S6, S10, S12, S18 and S24 condition monitoring measurements versus time for testing set example 1 are shown in Figure 5-31. The target and FNN, S-RNN, LSTM-RNN and GRU-RNN regression model predicted remaining useful life values versus time for testing set example 1 are shown in Figure 5-32, Figure 5-33, Figure 5-34 and Figure 5-35 respectively. The fault mode for testing set example 1 is high-pressure compressor degradation.

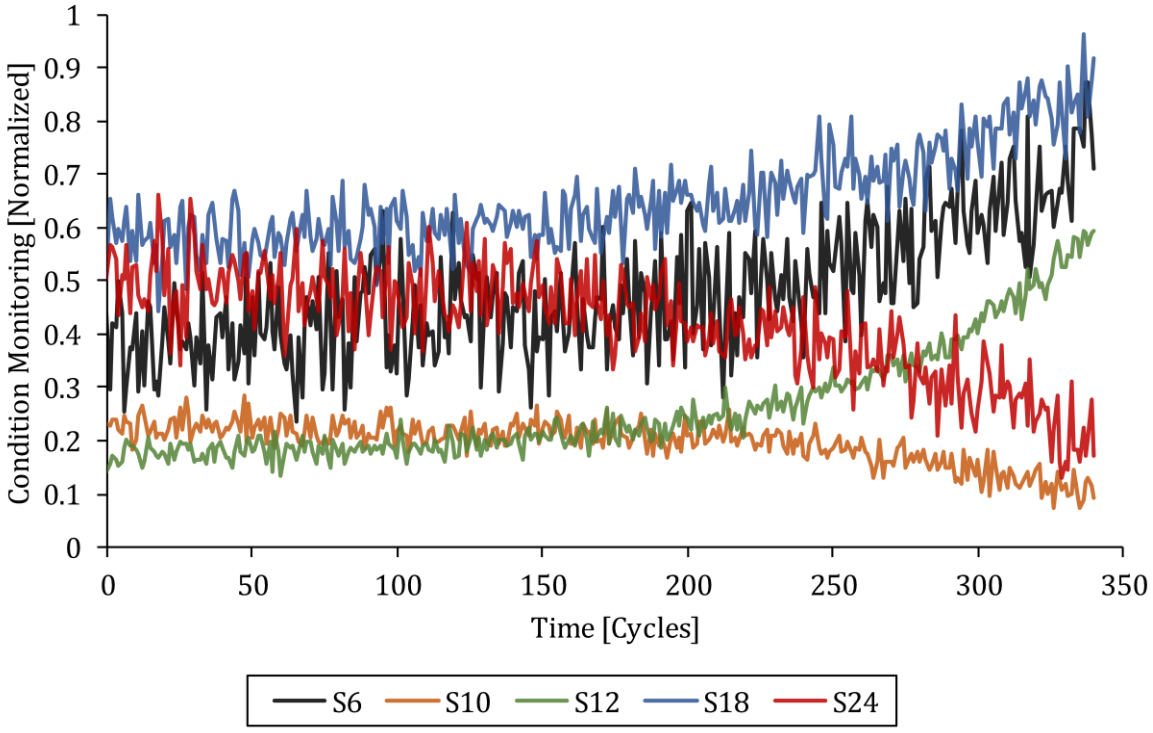

Figure 5-31: The S6, S10, S12, S18 and S24 condition monitoring measurements versus time for testing set example 1.

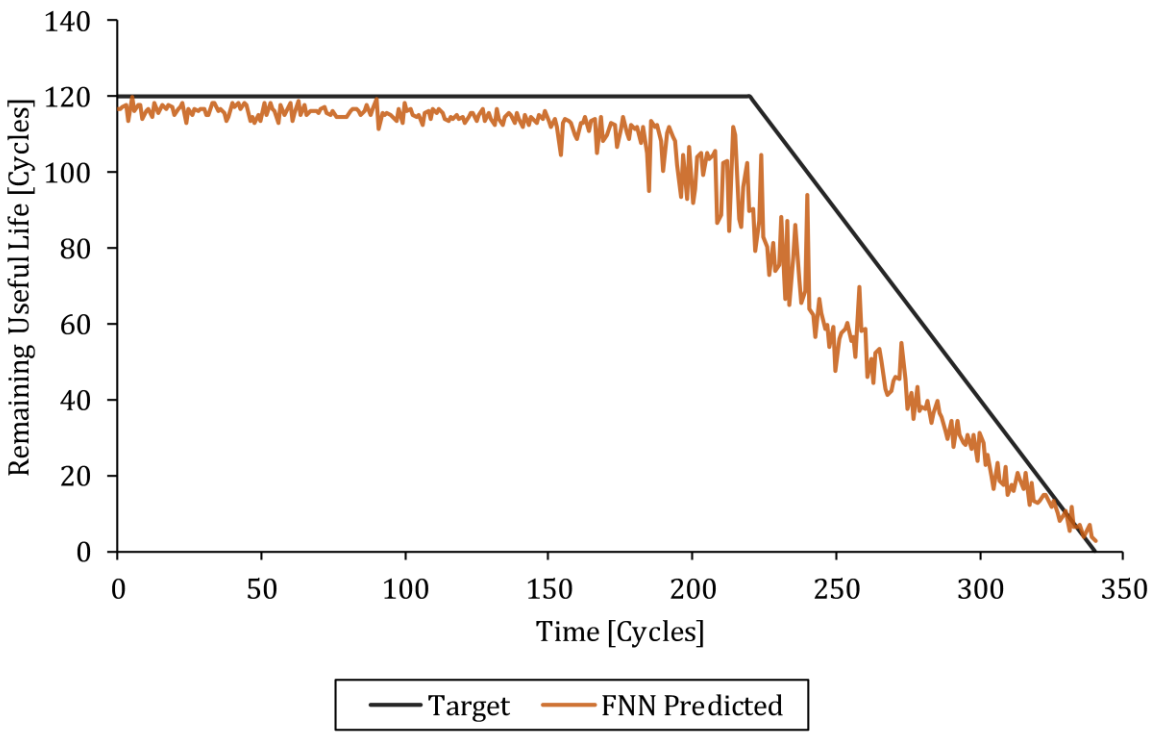

Figure 5-32: The target and FNN regression model predicted remaining useful life values versus time for testing set example 1.

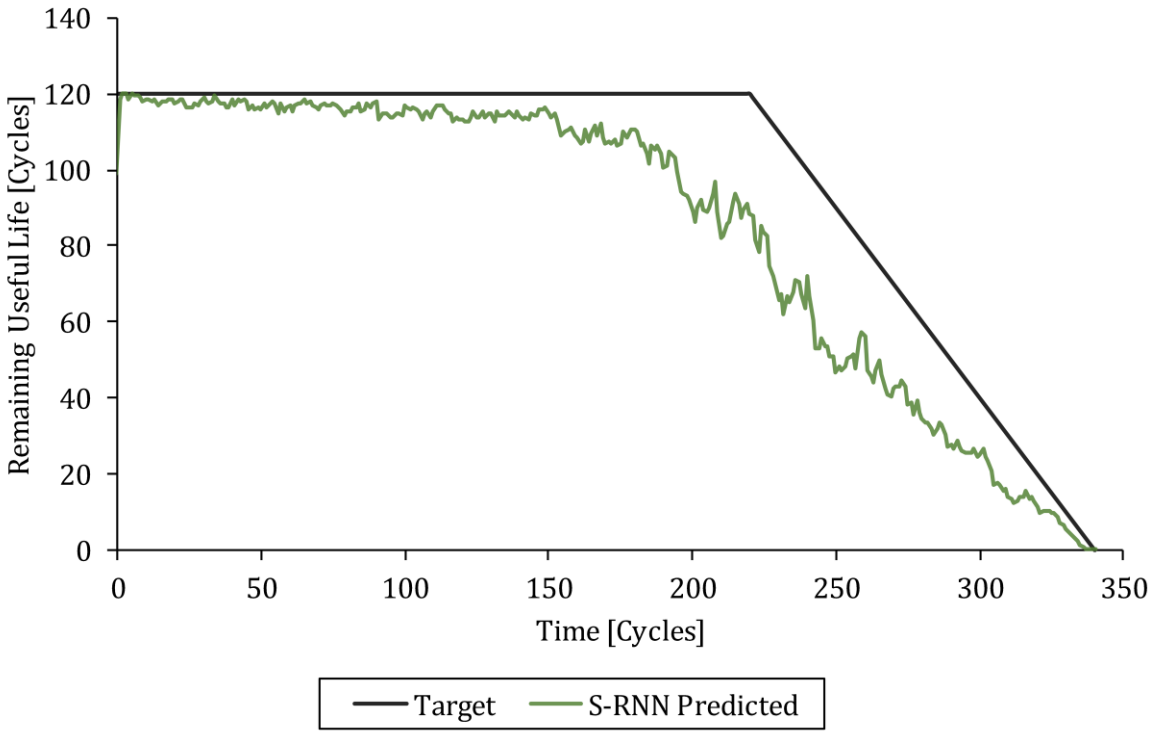

Figure 5-33: The target and S-RNN regression model predicted remaining useful life values versus time for testing set example 1.

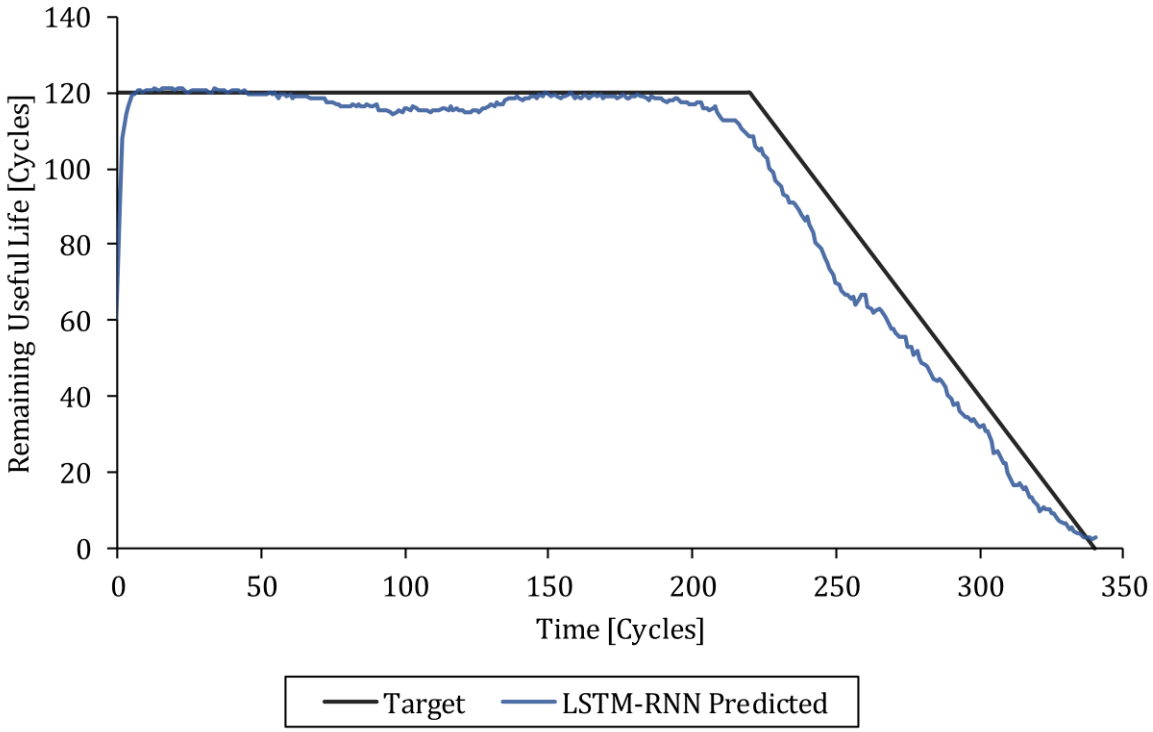

Figure 5-34: The target and LSTM-RNN regression model predicted remaining useful life values versus time for testing set example 1.

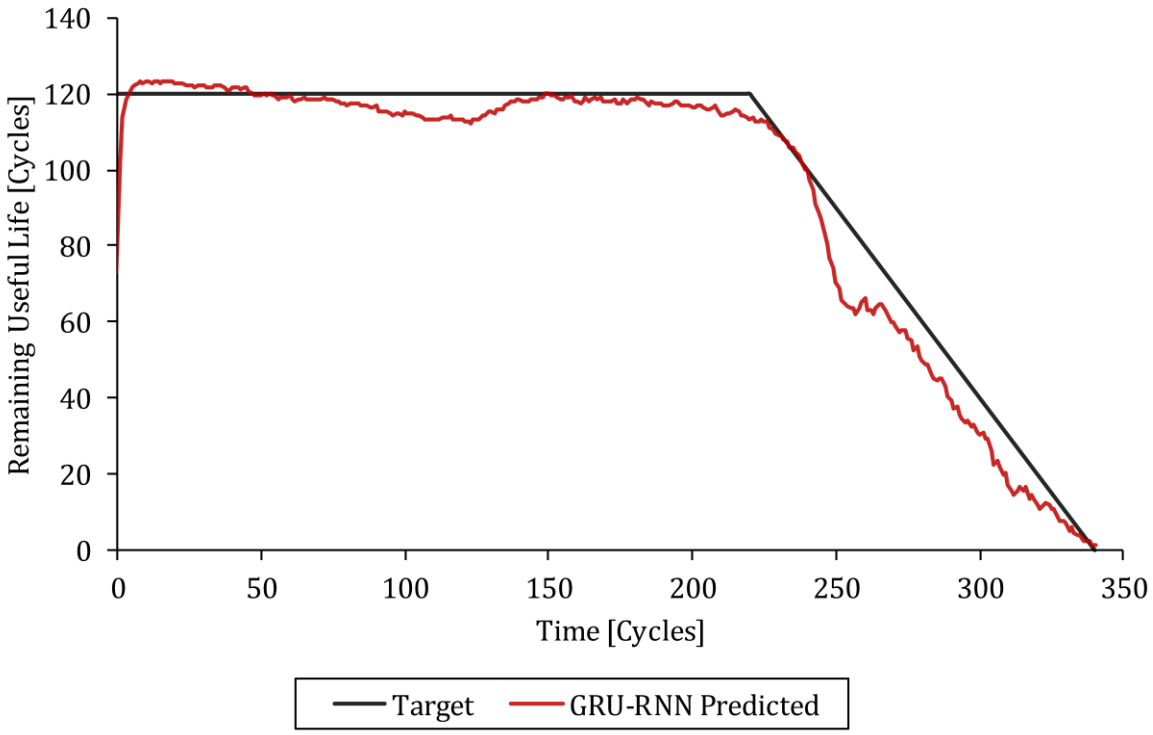

Figure 5-35: The target and GRU-RNN regression model predicted remaining useful life values versus time for testing set example 1.

Testing Set Example 2

The S6, S10, S12, S18 and S24 condition monitoring measurements versus time for testing set example 2 are shown in Figure 5-36. The target and FNN, S-RNN, LSTM-RNN and GRU-RNN regression model predicted remaining useful life values versus time for testing set example 2 are shown in Figure 5-37, Figure 5-38, Figure 5-39 and Figure 5-40 respectively. The fault mode for testing set example 2 is fan degradation.

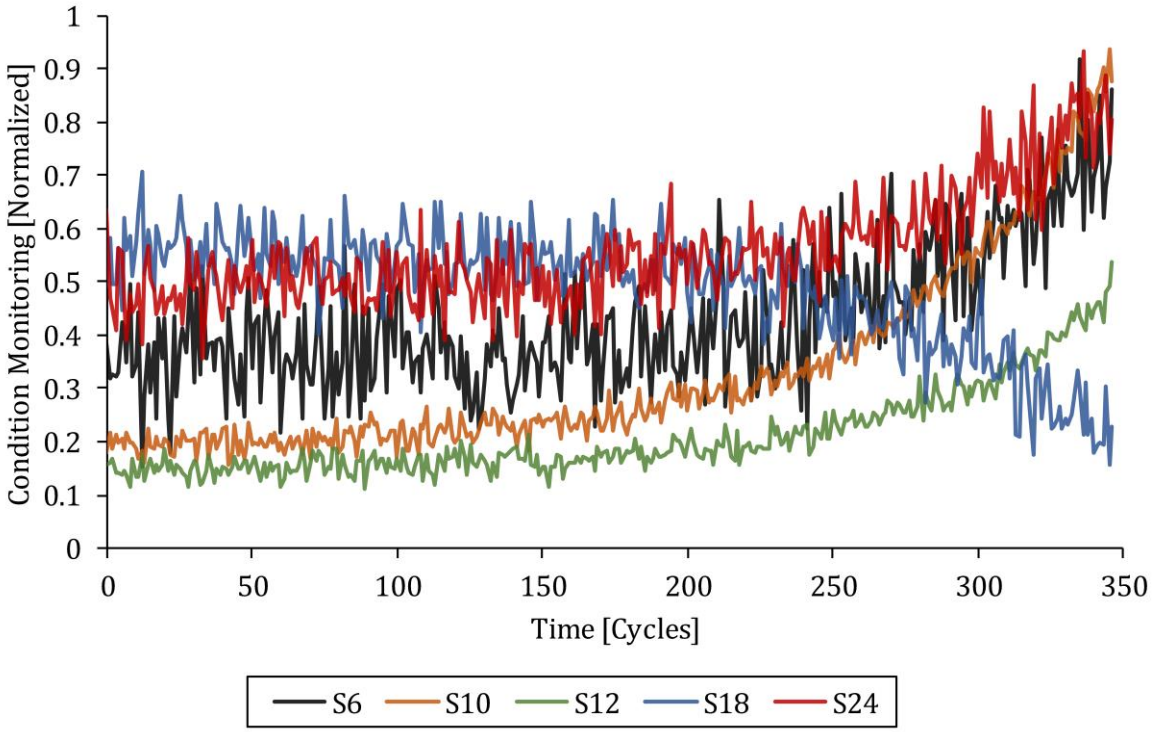

Figure 5-36: The S6, S10, S12, S18 and S24 condition monitoring measurements versus time for testing set example 2.

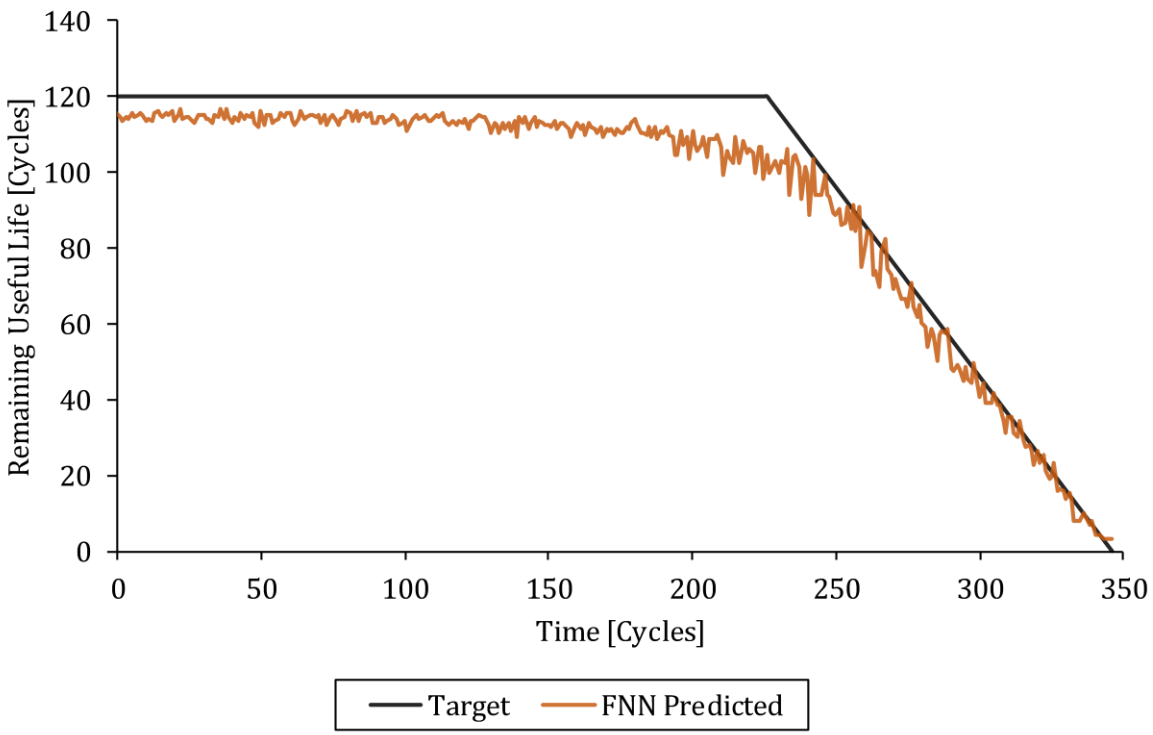

Figure 5-37: The target and FNN regression model predicted remaining useful life values versus time for testing set example 2.

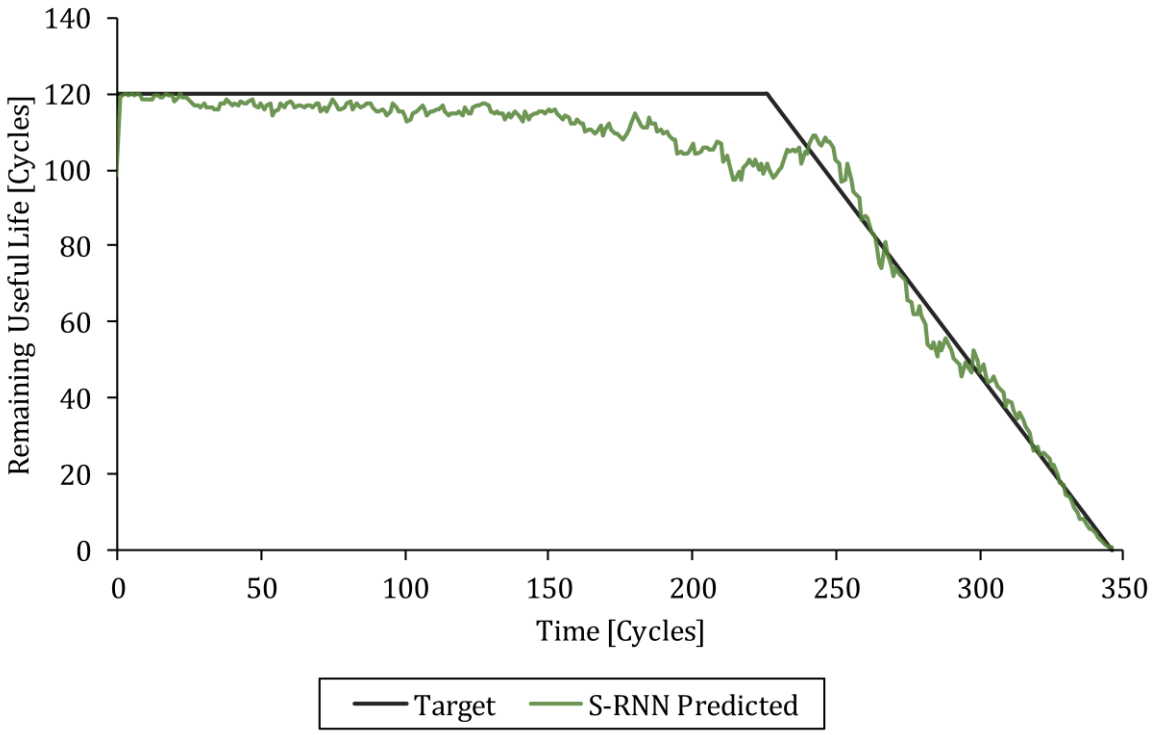

Figure 5-38: The target and S-RNN regression model predicted remaining useful life values versus time for testing set example 2.

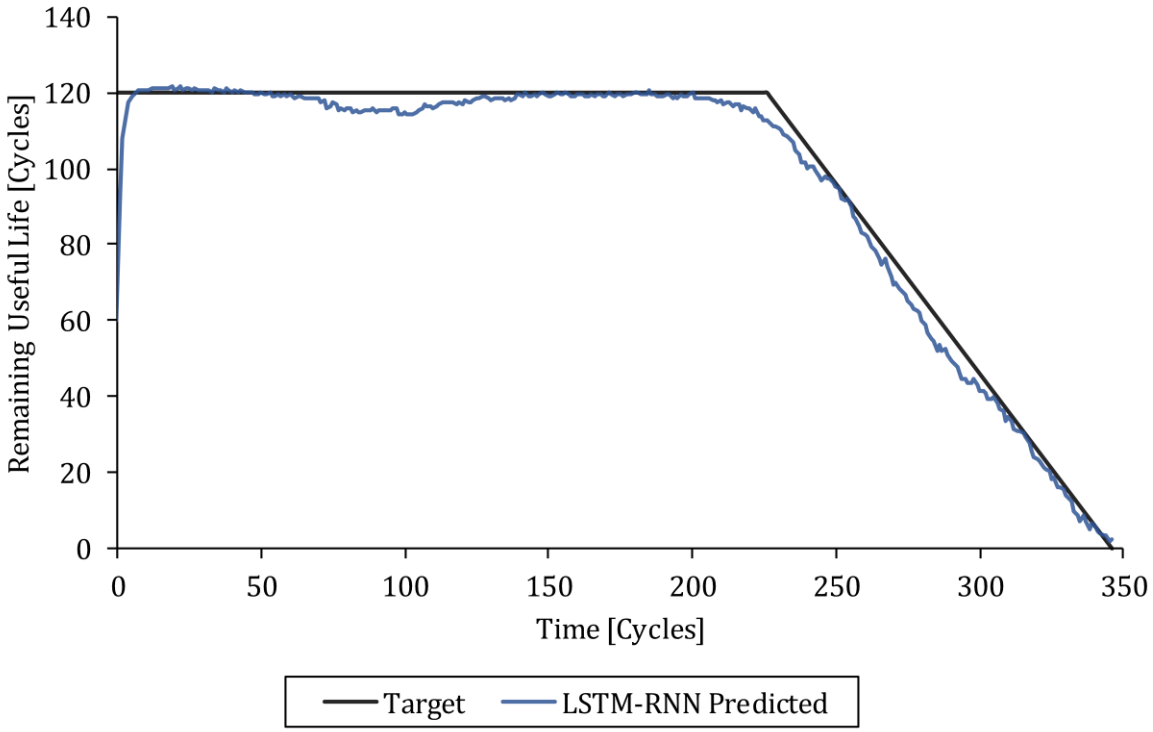

Figure 5-39: The target and LSTM-RNN regression model predicted remaining useful life values versus time for testing set example 2.

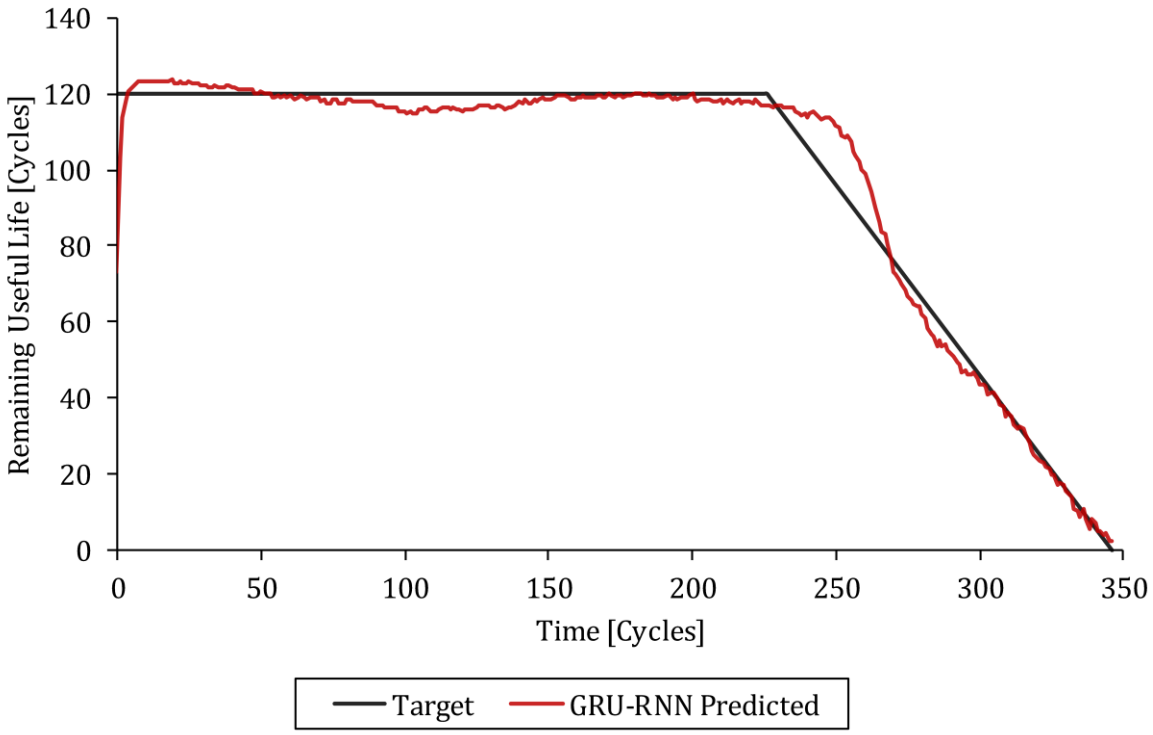

Figure 5-40: The target and GRU-RNN regression model predicted remaining useful life values versus time for testing set example 2.

From Figure 5-21–Figure 5-40 it can be concluded that the prognostics regression strategy and trained FNN, S-RNN, LSTM-RNN and GRU-RNN regression models can successfully predict the remaining useful life values from the condition monitoring measurements for the two randomly selected training and testing set examples in the turbofan engine degradation data set fully online. The LSTM-RNN and GRU-RNN regression models drastically outperformed the FNN and S-RNN regression models on the two randomly selected training and testing set examples as expected. The GRU-RNN regression model slightly outperformed the LSTM-RNN regression model and the S-RNN regression model significantly outperformed the FNN regression model on average. The predicted remaining useful life values of the FNN regression model were also drastically more noisy than that of the S-RNN, LSTM-RNN and GRU-RNN regression models. The predicted remaining useful life values of the FNN, S-RNN, LSTM-RNN and GRU-RNN regression models were also significantly more accurate close to failure as the trendability of the condition monitoring measurements increased.

5.2.4 Model Performance Comparison

The performance of the FNN, S-RNN, LSTM-RNN and GRU-RNN regression models were compared by calculating the mean squared error and mean absolute error between the target and predicted remaining useful life values for all the training set and testing set examples in the turbofan engine degradation data set. The mean squared error and mean absolute error between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN regression model predicted remaining useful life values for all the training set and testing set examples is shown in Table 5-3.

Table 5-3: The mean squared error and mean absolute error between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN regression model predicted remaining useful life values for all the training set and testing set examples.

Regression Model	Training Set Mean Squared Error	Testing Set Mean Squared Error	Training Set Mean Absolute Error	Testing Set Mean Absolute Error
FNN	208.4586	259.4999	10.2555	11.9651
S-RNN	147.9586	178.8964	8.0579	9.4524
LSTM-RNN	55.0477	65.9575	4.0908	4.7173
GRU-RNN	52.3379	55.4918	4.4563	4.6575

The performance of the FNN, S-RNN, LSTM-RNN and GRU-RNN regression models were also compared by recalculating the mean squared error and mean absolute error between the target and predicted remaining useful life values 120 cycles before failure for all the training set and testing set examples in the turbofan engine degradation data set. This is because it excluded the remaining useful life values above the applied target threshold and was therefore more representative of the regression model performance close to failure. The mean squared error and mean absolute error between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN regression model predicted remaining useful life values 120 cycles before failure for all the training set and testing set examples is shown in Table 5-4.

Table 5-4: The mean squared error and mean absolute error between the target and FNN, S-RNN, LSTM-RNN and GRU-RNN regression model predicted remaining useful life values 120 cycles before failure for all the training set and testing set examples.

Regression Model	**Training Set Mean Squared Error**	**Testing Set Mean Squared Error**	**Training Set Mean Absolute Error**	**Testing Set Mean Absolute Error**
FNN	312.6298	337.2173	12.829	13.3103
S-RNN	215.6306	233.4553	10.5511	11.0255
LSTM-RNN	53.9481	79.9956	5.07800	6.3836
GRU-RNN	62.7665	76.2654	5.5248	6.1194

From Table 5-3 and Table 5-4 it can be concluded that the LSTM-RNN and GRU-RNN regression models drastically outperformed the FNN and S-RNN regression models on the training set and testing set as expected. The GRU-RNN regression model slightly outperformed the LSTM-RNN regression model and the S-RNN regression model significantly outperformed the FNN regression model. The difference between the training set and testing set mean squared error and mean absolute error for the FNN, S-RNN, LSTM-RNN and GRU-RNN regression models 120 cycles before failure was also relatively small, which indicated good model regularization and an appropriately selected threshold for the prognostics regression strategy. The GRU-RNN regression model therefore made a very respectable mean absolute error of only 6.1194 cycles for the completely unseen testing set in the turbofan engine degradation data set 120 cycles before failure.

5.3 Turbofan Engine Degradation Benchmarking Data Sets

This section presents the results of the prognostics regression strategy and regression deep learning model architectures applied on the turbofan engine degradation benchmarking data sets.

5.3.1 Strategy and Model Description

The tuned and selected applied threshold AT for the prognostics regression strategy on the FD001, FD002, FD003, FD004 and PHM08 turbofan engine degradation benchmarking data sets are shown in Table 5-5.

Table 5-5: The tuned and selected applied threshold AT for the prognostics regression strategy on the FD001, FD002, FD003, FD004 and PHM08 turbofan engine degradation benchmarking data sets.

Data Set	AT [Cycles]
FD001	120
FD002	140
FD003	120
FD004	140
PHM08	120

The FNN, S-RNN, LSTM-RNN and GRU-RNN regression model architectures were trained individually on the training sets of the FD001, FD002, FD003, FD004 and PHM08 turbofan engine degradation benchmarking data sets with the Adam algorithm and regularized with a combination of the early stopping, weight decay and dropout regularization techniques.

5.3.2 Model Performance Benchmarking

The performance of the FNN, S-RNN, LSTM-RNN and GRU-RNN regression models for this work were compared with other state-of-the-art publications (discussed in the literature review) by calculating the benchmarking root mean square error and benchmarking score for the FD001, FD002, FD003 and FD004 turbofan engine degradation benchmarking data sets. A lower benchmarking root mean square error and benchmarking score indicated more accurately predicted remaining useful life values for the turbofan engines in the testing set on average. The benchmarking root mean square errors $\overline{RMSE}$ of the FNN, S-RNN, LSTM-RNN and GRU-RNN regression models for this work and other state-of-the-art publications on the FD001, FD002, FD003 and FD004 turbofan engine degradation benchmarking data sets are shown in Table 5-6.

Table 5-6: The benchmarking root mean square errors $\overline{RMSE}$ of the FNN, S-RNN, LSTM-RNN and GRU-RNN regression models for this work and other state-of-the-art publications on the FD001, FD002, FD003 and FD004 turbofan engine degradation benchmarking data sets.

Publication	Model	FD001	FD002	FD003	FD004
This Work	FNN	17.6940	28.5912	20.4740	31.0186
	S-RNN	16.6734	26.2828	16.6373	28.2799
	LSTM-RNN	12.6448	21.8212	13.2569	22.3102
	GRU-RNN	**12.2703**	**21.7622**	**12.4001**	**21.6811**
(Babu, et al., 2016)	MLP	37.5629	80.0301	37.3853	77.3688
	SVR	20.9640	41.9963	21.0480	45.3475
	RVR	23.7985	31.2956	22.3678	34.3403
	CNN	18.4480	30.2944	19.8174	29.1568
(Hsu & Jiang, 2018)	LSTM	16.7372	29.4325	18.0694	28.3958
(Li, et al., 2018)	NN	14.8000	25.6400	15.2200	25.8000
	DNN	13.5600	24.6100	13.9300	24.3100
	RNN	13.4400	24.0300	13.3600	24.0200
	LSTM	13.5200	24.4200	13.5400	24.2100
	DCNN	12.6100	22.3600	12.6400	23.3100
(Ramasso, 2014)	RULCLIPPER	13.2665	22.8910	16.0000	24.3311
(Zhang, et al., 2017)	GB	15.6700	29.0900	16.8400	29.0100
	RF	17.9100	29.5900	20.2700	31.1200
	MODBNE	15.0400	25.0500	12.5100	28.6600
(Zheng, et al., 2017)	LSTM	16.1400	24.4900	16.1800	28.1700

The benchmarking scores $\bar{S}$ of the FNN, S-RNN, LSTM-RNN and GRU-RNN regression models for this work and other state-of-the-art publications on the FD001, FD002, FD003 and FD004 turbofan engine degradation benchmarking data sets are shown in Table 5-7.

Table 5-7: The benchmarking scores $\bar{S}$ of the FNN, S-RNN, LSTM-RNN and GRU-RNN regression models for this work and other state-of-the-art publications on the FD001, FD002, FD003 and FD004 turbofan engine degradation benchmarking data sets.

Publication	Model	FD001	FD002	FD003	FD004
This Work	FNN	655	13075	1307	12190
	S-RNN	539	6579	791	5624
	LSTM-RNN	224	3111	262	2832
	GRU-RNN	**203**	3032	**222**	**2458**
(Babu, et al., 2016)	MLP	17972	7802800	17409	5616600
	SVR	1382	589900	1598	371140
	RVR	1503	17423	1432	26509
	CNN	1287	13570	1596	7886
(Hsu & Jiang, 2018)	LSTM	389	10654	822	6371
(Li, et al., 2018)	NN	496	18255	522	20422
	DNN	348	15622	364	16223
	RNN	339	14245	316	13931
	LSTM	432	14459	347	14322
	DCNN	274	10412	284	12466
(Ramasso, 2014)	RULCLIPPER	216	**2796**	317	3132
(Zhang, et al., 2017)	GB	474	87280	577	17818
	RF	480	70457	711	46568
	MODBNE	334	5585	422	6558
(Zheng, et al., 2017)	LSTM	338	4450	852	5550

The performance of the FNN, S-RNN, LSTM-RNN and GRU-RNN regression models for this work were also compared with other state-of-the-art publications on the PHM08 turbofan engine degradation benchmarking data set. The predicted remaining useful life values for the turbofan engine examples in the testing set of the PHM08 turbofan degradation benchmarking data set were uploaded to the NASA Prognostics Data Repository (NASA, 2018) website. The benchmarking score was then returned. The NASA Prognostics Data Repository website however only allowed for one attempt (submission) per day. The benchmarking scores $\bar{S}$ of the FNN, S-RNN, LSTM-RNN and GRU-RNN regression models for this work and other state-of-the-art publications on the PHM08 turbofan engine degradation benchmarking data set are shown in Table 5-8.

Table 5-8: The benchmarking scores $\bar{S}$ of the FNN, S-RNN, LSTM-RNN and GRU-RNN regression models for this work and other state-of-the-art publications on the PHM08 turbofan engine degradation benchmarking data set.

Publication	Model	PHM08
This Work	FNN	4231
	S-RNN	2327
	LSTM-RNN	688
	GRU-RNN	589
(Babu, et al., 2016)	MLP	3212
	SVR	15886
	RVR	8242
	CNN	2056
(Heimes, 2008)	RNN	**512**
(Lim, et al., 2014)	MLP (Linear)	118338
	MLP (Kink)	6103
	KF Ensemble	5590
	SKF Ensemble	2922
(Ramasso, 2014)	RULCLIPPER	752
(Wang, et al., 2008)	SBPA	737
(Zheng, et al., 2017)	LSTM	1862

From Table 5-6, Table 5-7 and Table 5-8 it can be concluded that the benchmarking scores and root mean square errors achieved by the prognostics regression strategy and LSTM-RNN and GRU-RNN regression model architectures were very competitive with other state-of-the-art publications on the FD001, FD002, FD003, FD004 and PHM08 turbofan engine degradation benchmarking data sets.

It is however important to point out that the author benchmarked each data-driven model only once on the testing set for each turbofan degradation benchmarking data set, in the spirit of the competition and ethics. This is because multiple submissions would allow each data-driven model to be tuned on the testing set and would therefore not be representative to the real world conditions. This was however not the case for other publications like (Heimes, 2008), where multiple submissions were made.

The files for the FNN, S-RNN, LSTM-RNN and GRU-RNN regression models that were uploaded to the NASA Prognostics Data Repository (NASA, 2018) website under the heading "7. PHM08 Challenge Data Set" are also provided digitally (Louw, 2018) if the reader wishes to validate the author's benchmarking scores.

6 Conclusions and Recommendations

6.1 Conclusions

The prognostics classification and regression strategies were successfully applied on both the univariate general asset degradation data set and the multivariate turbofan engine degradation data set with the FNN, S-RNN, LSTM-RNN and GRU-RNN classification and regression model architectures. The LSTM-RNN and GRU-RNN models drastically outperformed the FNN and S-RNN models as expected. The GRU-RNN models slightly outperformed the LSTM-RNN models and the S-RNN models significantly outperformed the FNN models on average.

The FNN, S-RNN, LSTM-RNN and GRU-RNN classification and regression models could successfully predict the remaining useful life classes and values from the condition monitoring measurements for the randomly selected training and testing set examples in the investigated data sets fully online. The predicted remaining useful life classes and values of the FNN, S-RNN, LSTM-RNN and GRU-RNN classification and regression models were significantly more accurate close to failure as the trendability of the condition monitoring measurements increased.

The FNN, S-RNN, LSTM-RNN and GRU-RNN classification and regression model architectures were successfully trained on the training sets of the investigated data sets with the Adam algorithm and successfully regularized with a combination of the early stopping, weight decay and dropout regularization techniques.

The Adam algorithm was also compared with the popular Gradient Decent, Momentum, AdaGrad and RMSProp algorithms. Surprisingly, the model training and testing performance for all the algorithms on the investigated data sets were very similar. The Adam algorithm was however found to be more stable, robust and to converge in fewer training iterations when compared to the Gradient Decent, Momentum, AdaGrad and RMSProp algorithms. The combination of the early stopping, weight decay and dropout regularization techniques were found to drastically increase the prediction performance of all the model architectures on the validation and testing sets of the investigated data sets when compared with no regularization.

It was found that adding fully-connected hidden layers before and after the simple recurrent, LSTM and GRU hidden layers in the recurrent neural network architectures resulted in a significant increase in model training and testing performance, when compared to not adding fully-connected hidden layers before and after these hidden layers. It was also found that stacking numerous LSTM or GRU hidden layers did not drastically improve the model testing performance for the data sets, and in some cases even reduced the model testing performance.

The proposed prognostics regression and classification strategies both have their merits and limitations depending on the predictive maintenance requirements. The prognostics regression strategy has the advantage that it can be more useful to know the exact predicted remaining useful

life value for maintenance planning and scheduling, than the predicted remaining useful life class. The prognostics classification strategy however has the advantage that it can be more convenient to interpret the predicted remaining useful life class as a degradation severity level for online maintenance decision-making, than the predicted remaining useful life value. The prognostics classification strategy also has the advantage that it can be used for fault mode classification for maintenance decision-making and is a significantly simpler problem to model than the prognostics regression strategy.

The applied threshold for the prognostics classification and regression strategies resulted in classification and regression deep learning models that are more conservative, fully online, easier to train, generalized better and made more accurate predictions below the applied threshold, compared to when no threshold is applied. The motivation for the applied threshold was that the condition monitoring measurement time series for the training and testing set examples in the investigated data sets include healthy and light degradation condition monitoring measurements that generally had very little to no trend. It was therefore very difficult for the classification and regression deep learning models to learn and generalize the mapping between the healthy and light degradation condition monitoring measurement time series and the linearly decreasing remaining useful life time series with no applied threshold. The applied threshold gave the healthy and light degradation condition monitoring measurements the same remaining useful life class or value target label. This implied that the healthy and light degradation condition monitoring measurements were ignored for remaining useful life class and value modeling. This made it significantly simpler and less confusing for the classification and regression deep learning models to learn and generalize the mapping between the condition monitoring measurement time series and the remaining useful life classification and regression time series.

When the value of the applied threshold was too high the classification and regression models were difficult to train and did not generalize well between the training and validation sets. When the value of the applied threshold was too low the classification and regression models were easy to train and generalized better, but were less useful and could only predict the remaining useful life classes and values close to failure. It was therefore important to tune the value of the applied threshold for each investigated data set. The strategy for selecting the value of the applied threshold for each investigated data set was successful and improved the generalization ability of the deep learning models between the training, validation and testing sets.

The objective of the classification and regression deep learning models was to learn and generalize the mapping between the condition monitoring measurement time series and the remaining useful life classification and regression time series for the training and testing set examples in each investigated data set. The condition monitoring measurement time series contains underlying asset health index information and the remaining useful life classification and regression time series were based on the time of failure. It was therefore very important that the

definition of failure was consistent with respect to the condition monitoring measurement time series and the underlying asset health index information between the training and testing set examples in each investigated data set. The definition of failure was however consistent between training and testing set examples in each investigated data set, but is important to point out when applying the prognostics regression and classification strategies on future data sets.

The prognostics regression strategy was successfully applied on the investigated FD001, FD002, FD003, FD004 and PHM08 turbofan engine degradation benchmarking data sets with the FNN, S-RNN, LSTM-RNN and GRU-RNN regression model architectures. The LSTM-RNN and GRU-RNN models again drastically outperformed the FNN and S-RNN models as expected. The GRU-RNN models again slightly outperformed the LSTM-RNN models and the S-RNN models again significantly outperformed the FNN models on average. The benchmarking scores and root mean square errors achieved by the prognostics regression strategy and LSTM-RNN and GRU-RNN regression model architectures were very competitive with other state-of-the-art publications on the FD001, FD002, FD003, FD004 and PHM08 turbofan engine degradation benchmarking data sets. The prognostics regression strategy and GRU-RNN regression model achieved a very competitive benchmarking score of 589 on the PHM08 turbofan degradation benchmarking data set.

The S-RNN, LSTM-RNN and GRU-RNN models significantly outperformed the FNN models for all the investigated data sets. This was because the S-RNN, LSTM-RNN and GRU-RNN models could model sequence information from previous time steps, where the FNN models could not. It was therefore concluded that sequence information is very important for degradation modeling.

The LSTM-RNN and GRU-RNN models significantly outperformed S-RNN models for all the investigated data sets. This was because the LSTM-RNN and GRU-RNN models were able to manage the vanishing and exploding gradient problem (Hochreiter & Schmidhuber, 1997) during model training with numerous gating operations. This drastically increased their capacity to model long-term sequence information from previous time steps, where the S-RNN models could not. It was therefore concluded that long-term sequence information is very important for degradation modeling.

The GRU-RNN models were found to outperform the LSTM-RNN models for all the investigated data sets. This was because the GRU hidden layer was slightly less complex with fewer trainable parameters when compared to the LSTM hidden layer, since it did not have an output gate. The GRU hidden layer was therefore less prone to overfitting when compared to the LSTM hidden layer. The GRU hidden layer has also been shown to outperform and generalize better on other smaller data sets when compared to LSTM hidden layer (Chung, et al., 2014).

The novelty and contribution of this research is that the state-of-the-art proposed LSTM-RNN and GRU-RNN deep learning models have to date not been comprehensively investigated and compared for data-driven prognostics for fleets of engineering assets. GRU-RNN deep learning

models have especially never been investigated or applied for data-driven prognostics for fleets of engineering assets.

The LSTM-RNN and GRU-RNN models were both very robust to the substantial noise present in the condition monitoring measurements of the investigated data sets. This was a big advantage, since the complex underlying asset health index information in the noisy condition monitoring measurements for the investigated data sets could easily be lost with filtering.

It can be argued that human intelligence can be used instead of the LSTM-RNN and GRU-RNN artificial intelligence models for the prognostics strategies. The problem with human intelligence however is that it generally struggles with high dimensional, noisy and nonlinear condition monitoring measurements for the prognostics of numerous assets at a large scale and over long periods of time. Another problem with human intelligence is that it is very expensive in time and money when compared to the LSTM-RNN and GRU-RNN artificial intelligence models. Additionally, human intelligence is also subject to human error. It was therefore concluded that the automated LSTM-RNN and GRU-RNN artificial intelligence models have numerous advantages over human intelligence for fully online prognostics and maintenances for fleets of assets at a large scale and over long periods of time. The LSTM-RNN and GRU-RNN artificial intelligence models could therefore be used as very powerful automated pattern recognition tools for automated planning, scheduling and decision-making of maintenance tasks for fleets of engineering assets.

The deep learning models were completely data-driven and did therefore not require any physical models, understanding or assumptions for the complex degradation trajectories of the assets in the investigated data sets. The deep learning models also had enough capacity to model numerous failure modes and operating conditions for the turbofan engines in the investigated turbofan engine degradation benchmarking data sets. This was a big advantage, since the same deep learning model could be used for numerous fault modes and operating conditions fully online.

The deep learning models were able to automatically discriminate between relevant and irrelevant condition monitoring sensor measurements for the investigated turbofan engine degradation data set. The prognostics classification and regression strategies did not require any sensor fusion between the multivariate condition monitoring sensor measurements. This was a big advantage, since the important and complex underlying asset health index information in the multivariate condition monitoring sensor measurements can easily by lost with sensor fusion.

The trained regression models can all be interpreted as functions. The partial derivative and gradient of the model predicted remaining useful life value with respect to the multivariate condition monitoring sensor measurements can therefore be calculated numerically at each time step for the trained regression models. The trendability and importance of each condition monitoring sensor measurement with respect to the model predicted remaining useful life value at each time step can therefore be ranked. This is therefore an effective method that can be used

to determine the usefulness of expensive, difficult or unsafe condition monitoring measurements with the trained regression models.

The prognostics classification and regression strategies were very general, clear, simple, conservative and could easily be applied to any future engineering assets with historical run-to-failure condition monitoring measurements for fully online and automated remaining useful life prediction of future assets. It was however very important that the definition of failure was consistent between historical and future assets for the proposed prognostics strategies. It was also very important that the univariate or multivariate condition monitoring measurements contains underlying asset health index information that was representative of the true asset health and was trendable with respect to the remaining useful life of the asset.

The prognostics classification and regression strategies were intended for fleets of assets with historical run-to-failure condition monitoring measurements and trendable degradation trajectories. The biggest limitation of the prognostics classification and regression strategies is therefore that they cannot be applied to assets with no historical run-to-failure condition monitoring measurements or assets with non-trendable degradation trajectories.

6.2 Recommendations and Future Work

The investigated general asset degradation and turbofan engine degradation data sets were both simulated data sets with realistic exponential degradation trajectories. It would however still be interesting and important to see how the proposed prognostics classification and regression strategies and deep learning models would perform on real-world data sets in future work.

The prognostics regression strategy and deep learning models can easily be extended for forecasting (extrapolating) the trendable condition monitoring measurements of a degrading asset to an assumed threshold, when no historical run-to-failure condition monitoring measurements are available. The difference between the forecasted threshold time and the current time would then be the remaining useful life of the degrading asset. For these forecasting deep learning models the input would be the condition monitoring measurements for the current time step and the target would be the condition monitoring measurements for the next time step. These forecasting deep learning models would therefore be trained on the available historical condition monitoring measurements and learn the degradation pattern in the available historical condition monitoring measurements. The trained forecasting deep learning models would then be used to forecast (extrapolate) the historical condition monitoring measurements for numerous time steps into the future until the assumed threshold is reached. This is however very difficult, because the degradation rate of exponentially degrading assets are constantly increasing and would therefore result in numerous late remaining useful life predictions (not conservative). The author would however not recommend these forecasting deep learning models if historical run-

to-failure condition monitoring measurements are available. This is because they are less convenient, less accurate, more computationally expensive, less conservative and not fully online when compared to the proposed prognostics classification and regression strategies and deep learning models.

The prognostics classification strategy and deep learning models can easily be extended for diagnostics (fault mode classification) on future data sets, as shown for the turbofan engine degradation data set. The classification deep learning models were all able to very accurately discriminate and classify between high-pressure compressor degradation and fan degradation for all the turbofan engines, which can be very beneficial for maintenance decision-making. It is therefore recommended to record the fault modes along with the condition monitoring measurements when implementing the prognostics classification strategy and deep learning models on future data sets.

The loss functions that were minimized in order to train the model parameters of the deep learning model architectures on each investigated data set, can also be weighted in order to penalize late remaining useful life predictions more than early remaining useful life predictions. This would result in more conservative trained deep learning models.

(Wang, et al., 2008) and (Ramasso, 2014) both proposed similarity-based approaches that achieved very good results on the turbofan engine degradation benchmarking data sets. These similarity-based approaches were outside the scope of this work, but are powerful approaches that can be investigated in future research.

The model parameters of the deep learning model architectures were successfully trained with the gradient decent based Adam algorithm, but can however also be trained with an extended Kalman filter (Heimes, 2008). Recurrent neural networks trained with the extended Kalman filter have been shown to outperform recurrent neural networks trained with gradient decent based algorithms (Perez-Ortiz, et al., 2003). The extended Kalman filter can therefore be investigated for training the model parameters of the deep learning model architectures in future work.

The hyperparameters of the implemented deep learning models were manually tuned by training numerous models with different hyperparameters and validating the performance (validation error) of the numerous trained models on the validation set of an investigated data set. This is also known as simple hold-out validation (Chollet, 2018). There are however more sophisticated methods for validating the hyperparameters of the deep learning models, for example k-fold cross-validation and iterated k-fold cross-validation with shuffling that can be used to further improve model regularization and generalization performance (Chollet, 2018). These approaches are however significantly more computationally expensive compared to simple hold-out validation approach that was used. There are also more sophisticated methods for optimizing the model hyperparameters than manual tuning. For example, the differential evolution optimization algorithm (Storn, 1996) can be investigated in future work to automatically tune the model

hyperparameters. This is because the validation error is generally a highly multi-modal function with respect to the hyperparameters of a model (Heimes, 2008).

It is important that the time intervals between the condition monitoring measurements are chosen sensibly for the prognostics strategies and deep learning models. If the time intervals between the condition monitoring measurements are unnecessarily short, the deep learning models will be significantly more difficult and computationally expensive to train. However, if the time intervals between the condition monitoring measurements are unnecessarily long, the deep learning models will be less practical and less convenient due to long waiting times between predictions.

The prognostics strategies and deep learning models will however only be practically and financially useful for assets where the degradation is trendable over relatively long periods of time, the cost of condition monitoring sensible and historical run-to-failure condition monitoring measurements of similar assets are available. It is therefore very important to investigate if these requirements are met before implementing the proposed prognostics strategies and deep learning models in industry. It is can also be very beneficial to perform a cost-benefit analysis before implementing the prognostics strategies and deep learning models in industry. For example, it would be totally viable to use a traditional time based preventative maintenance strategy over the data-driven prognostics strategies for assets that fail stochastically with no trendable degradation and where the cost of condition monitoring is very high. It will also be interesting to see how the prediction performance of the trained LSTM-RNN and GRU-RNN deep learning models (artificial intelligence techniques) compares with human intelligence in future work.

It is important to have a convenient standardized format for recording the condition monitoring measurements at a large scale and over long periods of time for the prognostics strategies and deep learning models in practice. It is also important to continuously record new condition monitoring measurements for new assets, such that the deep learning models can be continuously updated and improved.

Although the GRU-RNN model architectures were found to outperform the LSTM-RNN model architectures on the relatively small investigated data sets in a deep learning context, the author would still recommend attempting and comparing both model architectures on future data sets. The LSTM-RNN model architectures still have higher sequence modeling capacity (more trainable parameters) and might therefore perform better on larger data sets when compared to the GRU-RNN model architectures. The author would however still recommend the GRU-RNN model architectures for smaller data sets where the LSTM-RNN model architectures would be more prone to overfitting. Variations of the LSTM hidden layer (Greff, et al., 2016) and GRU hidden layer (Dey & Salem, 2017) can also be investigated in future work for possible increased model training and testing performance.

The classification and regression deep learning models can also be applied for numerous other surrogate modeling, machine learning and inference problems in mechanical engineering for future work. The data-driven LSTM-RNN and GRU-RNN models would especially achieve state-of-the-art modeling results on any complex sequence-to-sequence classification or regression modeling problem.

The author would highly recommend implementing the prognostics strategies and deep learning models with the open-source (free) Tensorflow (Google Brain, 2016) and easy to use Keras (Chollet, 2015) application programming interfaces (APIs) in Python. The prognostics strategies and some of the deep learning models can however also be implemented with the proprietary Deep Learning Toolbox (MathWorks, 2017) in MATLAB. The author would also highly recommend using a Nvidia graphics card that can parallelize and greatly speed up the required computations when training and testing the deep learning models on future data sets.

It is important to point out that the prognostics strategies and deep learning models in this work are very general and can therefore easily be applied to any similar future data sets. It is also important to point out that the deep learning models scale exceptionally well with very large data sets when compared with traditional machine learning techniques (Chollet, 2018).

7 References

Babu, G. S., Zhao, P. & Li, X.-L., 2016. Deep Convolutional Neural Network Based Regression Approach for Estimation of Remaining Useful Life. *International Conference on Database Systems for Advanced Applications,* pp. 214-228.

Baydin, A. G., Pearlmutter, B. A., Radul, A. A. & Siskind, J. M., 2018. Automatic Differentiation in Machine Learning: a Survey. *Journal of Machine Learning Research,* 18(2018), pp. 1-43.

Bishop, C. M., 2006. *Pattern Recognition and Machine Learning.* 1 ed. New York: Springer Science+Business Media, LLC.

Cauchy, A., 1847. Méthode générale pour la résolution de systèmes d'équations simultanées. *Compte rendu des séances de l'académie des sciences,* 83(225), p. 536–538.

Cho, K., Merrienboer, B. v., Bahdanau, D. & Bengio, Y., 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. *arXiv:1409.1259.*

Chollet, F., 2015. *Keras.* [Online]
Available at: https://keras.io

Chollet, F., 2018. *Deep Learning with Python.* 1 ed. New York: Manning Publications Co..

Chung, J., Gulcehre, C., Cho, K. & Bengio, Y., 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *arXiv:1412.3555.*

Dey, R. & Salem, F. M., 2017. Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks. *arXiv:1701.05923.*

Duchi, J., Hazan, E. & and Singer, Y., 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research,* 12(2011), pp. 2121-2159.

Elman, J. L., 1990. Finding Structure In Time. *Cognitive Science,* Volume 14, pp. 179-211.

Glorot, X. & Bengio, Y., 2010. Understanding the Difficulty of Training Deep Feedforward Neural Networks. *AISTATS.*

Goodfellow, I., Bengio, Y. & Courville, A., 2016. *Deep Learning.* 1 ed. USA: MIT Press.

Google Brain, 2016. TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. *arXiv:1603.04467.*

Greff, K. et al., 2016. LSTM: A Search Space Odyssey. *IEEE Transactions on Neural Networks and Learning Systems, arXiv:1503.04069.*

Heimes, F. O., 2008. Recurrent Neural Networks for Remaining Useful Life Estimation. *2008 IEEE International Conference on Prognostics and Health Management.*

Hinton, G., Srivastava, N. & Swersky, K., 2012. Neural Networks for Machine Learning. *Coursera Video Lectures.*

Hochreiter, S. & Schmidhuber, J., 1997. Long Short-Term Memory. *Neural Computation,* 9(8), pp. 1735-1780.

Hsu, C.-S. & Jiang, J.-R., 2018. Remaining Useful Life Estimation Using Long Short-Term Memory Deep Learning. *2018 IEEE International Conference on Applied System Invention,* pp. 58-61.

Jardine, A., Lin, D. & Banjevic, D., 2006. A Review On Machinery Diagnostics and Prognostics Implementing Condition-Based Maintenance. *Mechanical Systems and Signal Processing,* 20(2006), pp. 1483-1510.

Kingma, D. P. & Ba, J. L., 2015. Adam: A Method for Stochastic Optimization. *arXiv:1412.6980.*

Lei, Y. et al., 2018. Machinery Health Prognostics: A Systematic Review from Data Acquisition to RUL Prediction. *Mechanical Systems and Signal Processing,* 104(2018), p. 799–834.

Lim, P., Goh, C. K., Tan, K. C. & Dutta, P., 2014. Estimation of Remaining Useful Life Based on Switching Kalman Filter Neural Network Ensemble. *Annual Conference of the Prognostics and Health Management Society 2014,* pp. 1-8.

Li, X., Ding, Q. & Sun, J.-Q., 2018. Remaining Useful Life Estimation in Prognostics Using Deep Convolution Neural Networks. *Reliability Engineering and System Safety,* 172(2018), pp. 1-11.

Louw, C. J., 2018. *PHM08 Turbofan Engine Degradation Benchmarking Data Set Files.* [Online] Available at: https://drive.google.com/drive/folders/1DXFyYOx6ftSorLIfKUCxwe8F47Lp7zRB?usp=sharing

MathWorks, 2017. *Deep Learning Toolbox.* [Online]
Available at: https://www.mathworks.com/products/deep-learning.html

Mishra, M., Saari, J., Galar, D. & Leturiondo, U., 2014. Hybrid Models for Rotating Machinery Diagnosis and Prognosis Estimation of Remaining Useful Life. *Technical report, Lulea University of Technology.*

NASA, 2018. *NASA Ames Prognostics Data Repository, NASA Ames Research Center, Moffett Field, CA.* [Online]
Available at: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/

Perez-Ortiz, J. A., Gers, F. A., Eck, D. & Schmidhuber, J., 2003. Kalman Filters Improve LSTM Network Performance In Problems Unsolvable by Traditional Recurrent Nets. *Neural Networks,* 16(2), pp. 1-17.

Polyak, B., 1964. Some Methods of Speeding Up the Convergence of Iteration Methods. *USSR Computational Mathematics and Mathematical Physics,* 4(5), pp. 1-17.

Ramasso, E., 2014. Investigating Computational Geometry for Failure Prognostics. *International Journal of Prognostics and Health Management.*

Ramasso, E. & Gouriveau, R., 2010. Prognostics in Switching Systems: Evidential Markovian Classification of Real-Time Neuro-Fuzzy Predictions. *2010 Prognostics and System Health Management Conference.*

Ramasso, E. & Gouriveau, R., 2014. Remaining Useful Life Estimation by Classification of Predictions Based on a Neuro-Fuzzy System and Theory of Belief Functions. *IEEE Transactions on Reliability,* 63(2), pp. 1-12.

Ramasso, E. & Saxena, A., 2014. Review and Analysis of Algorithmic Approaches Developed for Prognostics on CMAPSS Dataset. *Annual Conference of the Prognostics and Health Management Society 2014,* pp. 1-11.

Rigamonti, M. et al., 2016. Echo State Network for the Remaining Useful Life Prediction of a Turbofan Engine. *European Conference of the Prognostics and Health Management Society 2016.*

Saxe, A. M., McClelland, J. L. & Ganguli, S., 2014. Exact Solutions to the Nonlinear Dynamics of Learning in Deep Linear Neural Networks. *arXiv:1312.6120.*

Saxena, A. & Goebel, K., 2008. *"PHM08 Challenge Data Set", NASA Ames Prognostics Data Repository, NASA Ames Research Center, Moffett Field, CA.* [Online]
Available at: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/

Saxena, A. & Goebel, K., 2008. *"Turbofan Engine Degradation Simulation Data Set", NASA Ames Prognostics Data Repository, NASA Ames Research Center, Moffett Field, CA.* [Online]
Available at: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/

Saxena, A., Goebel, K., Simon, D. & Eklund, N., 2008. Damage Propagation Modeling for Aircraft Engine Run-to-Failure Simulation. *2008 International Conference on Prognostics and Health Management.*

Srivastava, N. et al., 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research,* 15(2014), pp. 1929-1958.

Storn, R., 1996. On the Usage of Differential Evolution for Function Optimization. *North American Fuzzy Information Processing Society,* pp. 519-523.

Wang, T., Yu, J., Siegel, D. & Lee, J., 2008. A Similarity-Based Prognostics Approach for Remaining Useful Life Estimation of Engineered Systems. *2008 International Conference on Prognostics and Health Management.*

Zhang, C., Lim, P. & Qin, A. K., 2017. Multiobjective Deep Belief Networks Ensemble for Remaining Useful Life Estimation in Prognostics. *IEEE Transactions on Neural Networks and Learning Systems,* 28(10), pp. 2306-2318.

Zheng, S., Ristovski, K., Farahat, A. & Gupta, C., 2017. Long Short-Term Memory Network for Remaining Useful Life Estimation. *2017 IEEE International Conference on Prognostics and Health Management.*