WEIGHTED LOSS GRU NETWORK FOR MACHINE FAILURE PREDICTION

Steven Spreizer & Ajeet Parmar
Rensselaer Polytechnic Institute
Troy, NY 12180, USA
{spreis,parmaa}@rpi.edu

ABSTRACT

Health monitoring of mechanical systems is an important part of ensuring their function and minimizing losses when maintenance is required. Specifically, predicting remaining useful life (RUL) of a system is used to plan and optimize when maintenance is done. Machine learning is a popular and powerful way to perform RUL analysis based on sensor signals over time. In this work, a gated recurrent unit (GRU) based artificial neural network is used to process sensor signals from a simulated jet engine and predict the RUL of the engine. Appropriate preprocessing techniques are applied for raw data of this type. Typical loss and error formulations for this problem are modified in order to account for the economics of the problem and produce predictions consistent with minimizing overall cost to the operator of such a system. The implemented weighted loss method is shown to adjust the training process to generate more conservative predictions which are favorable given cost of overestimating RUL. The proposed weighted error metric is seen to be useful in understanding model performance in the context of problem economics.

1 INTRODUCTION

Health monitoring of mechanical systems is a crucial part of developing and managing maintenance plans for these systems. Developing tools to understand when a system, such as a wind turbine, compressor, or bearing, will fail and require maintenance allows for better business operations. These tools allow for implementation of condition based maintenance (CBM) programs to manage a system (Lei et al., 2018). Given the utility of this prediction process, much research has been conducted in the field of prognostics and health management (PHM).

An important task in implementing a PHM strategy is creating predictions of remaining useful life (RUL) for the system. RUL is an indicator of how long a system has until it will fail/require maintenance and is often presented as a unit of time (e.g. days) or a measurement of system operation (e.g. cycles). In the economic context, prediction of RUL is used as a tool to drive down maintenance costs for a given system. When looking at a binary classification system of normal working condition or failure, a simple cost matrix can be generated, as seen in Table 1. When predictions are correct (true and predicted values are the same), the business either sees no cost associated with the prediction or a necessary cost for maintenance. The problem is more interesting when looking at incorrect predictions of the model. When the model predicts a failure condition but the system is working normally, a relatively low cost is endured by the business to perform, for example, an unnecessary manual inspection. In the case where the model predicts working condition but the system has failed, the business faces a high cost to perform unexpected maintenance and interruption to expected operation of the system. Thus, it is desirable for an RUL model to give conservative estimates of remaining life before maintenance is needed to avoid the high cost section of the matrix.

A number of methods exist for performing RUL estimation and can be classified in a number of groups: knowledge-based models, life expectancy models, machine learning approaches, and physical models (Sikorska et al., 2011). Knowledge-based models, some of which include expert systems and fuzzy systems, compare the systems current state against previous states/events in order to predict the life expectancy. Experts in the field/domain of interest are needed to develop these

Table 1: Prediction cost matrix

	True Working	**True Failure**
Predicted Working	No cost	Very high cost
Predicted Failure	Minimal cost	Necessary cost

approaches and they can be complicated to develop but produce easily interpretable models. Life expectancy models such as stochastic or statistical models utilize historical data to produce their predictions. These methods likewise provide human understandable insights into their predictions and are effective in handling multivariate data. Physical models utilize understanding of the underlying physics of a system as well as experimental data to produce models that perform well in predicting remaining life. Machine learning methods for this task are the focus of this work and are discussed in the following section.

1.1 Existing Work

Machine learning for RUL prediction (and generally tasks in PHM) has taken off in the recent past decade as a way to supplement existing means of monitoring and planning maintenance of mechanical systems, in addition to providing improvements that help businesses operate their equipment more efficiently.

One review paper focuses on machine learning technology for failure prediction within industrial maintenance and addressed 1024 articles with respect to their datasets, preprocessing, and the overall training and evaluation of the prediction models. This paper noted that the most commonly used algorithm is Random Forests (RF) followed by Support Vector Machines (SVMs), Artificial Neural Networks (ANN), Decision Trees, Gradient Boosting, and Logistical Regression. Within these papers, the evaluation phases generally consisted of factor manipulation, the assessment of performance with operational data, and the reporting of results using standard metrics. Eight studies compared the performance of algorithms with five of these finding that RF had superior performance while three found that Gradient Boosting had superior performance. The material systems of focus varied immensely throughout the papers and included a wide array of prediction windows from some prediction windows measured in seconds and other prediction windows going up to several months. Evaluation of these systems did vary immensely, with a standard evaluation technique including cross validation. However, many studies depended on metrics which either represented a singular area of prediction performance or was inadequate for imbalanced data. Although nine studies did assess performance fully through precision, recall, and F-score. The conclusions of these reviews indicated that while there were up-to-date methods applied to a breadth of systems along with considerable diversity on reporting results, there is a lack of overall performance evaluation with respect to overfitting (Leukel et al., 2021).

Another review paper focuses on machine learning methods and their application to predictive maintenance (PdM) on equipment failures. This paper found that the most employed machine learning model is RF followed by neural-network based models such as ANNs, CNNs, LSTM, SVMs, and k-means. Uses of RF vary from uses as a classification algorithm to use as a method to generate dynamically predictive models and general variance in application to machinery. Similarly to RF, ANNs do vary immensely in use from uses in classification on fault recognition within a system or in soft sensing and predictive control. SVMs did also generally fall into identification (in many cases, identifying features contributing to the failure of mechanical systems and then use in classification in SVMs) or regression where an SVM determines faults on mechanical systems. For instance, one paper focused on alarm faults in rail bearings while another used SVMs and ANNs to classify electrical faults in power systems. K-means uses for PdM mainly focus on overall cluster analysis to determine commonalities within clusters as a means of analyzing faults. Such application areas include sensor data from oxygen sensors in a selective laser melting machine tool and to determine fault areas in dissolved gases in insulating oil of a transformer. The review came to the conclusion that although the machine learning models address a wide array of equipment and various types of machine learning algorithms, some of the works lack parameter tuning. Overall recommendations include additional data to facilitate the creation and validation of a PdM model and the development of novel datasets for use and comparison by PdM models Carvalho et al. (2019).

The NASA C-MAPPS dataset (which is the focus for applying this work), in particular, has been relatively well studied over the past decade and a half. It contains data from the simulation of a aircraft jet engine run to failure with the goal of predicting how many cycles until failure will occur. The dataset is described in more detail in Section 3. Heimes (2008) approaches the problem of RUL prediction for this dataset using a basic Recurrent Neural Network (RNN) architecture and provides insights about properties of the data. Recently, Li et al. (2020) looks into application of modern Gated Recurrent Unit (GRU) based networks for this problem, as a variation/improvement on RNN/LSTM based methods. Recommendations from this paper are discussed later in their application to this work. Variations on this model structure have also been investigated (Qu et al., 2022)(Sun et al., 2022).

In much of the previous work on this dataset, traditional formulations of loss (e.g. mean square error) and error reporting (e.g. mean absolute error, R^2) are used to evaluate the performance of the model. These approaches typically will lead to results which minimize error but do not consider the case where some errors are worse than others as is the case with failure prediction (failing to predict failure is more costly than other errors). The goal for this work is to investigate the different ways loss and error evaluation are applied to the RUL prediction problem with the goal of producing a model that will minimize the cost of the predictions being made rather than just the error in the predictions.

2 Model Overview

A GRU is an evolution of the basic RNN (Recurrent Neural Network) model consisting of two gates: an update gate and a reset gate. The update gate determines whether to keep certain information from previous hidden states and a current input. Cho et al. (2014)

The update gate represents the extent of past information to be passed to future iterations. In contrast, the hidden gate determines which information from the previous hidden states and current input is to be discarded. A GRU starts with the following operation to determine current memory content through the reset gate:

$$r_j = \sigma\big([W_r x]_j + [U_r h_{\langle t-1\rangle}]_j\big) \tag{1}$$

Where the gate r_j is computed through the input, denoted by x, the logistic sigmoid function (σ), the previous hidden state ($h_{\langle t-1\rangle}$), and learned weight matrices, denoted by W_r and U_r. The update gate is computed through the following equation:

$$z_j = \sigma\big([W_z x]_j + [U_z h_{\langle t-1\rangle}]_j\big) \tag{2}$$

The activation of h_j is computed through:

$$h_j^{\langle t\rangle} = z_j h_j^{\langle t-1\rangle} + (1 - z_j)\tilde{h}_j^{\langle t\rangle} \tag{3}$$

where

$$\tilde{h}_j^{\langle t\rangle} = \phi\big([Wx]_j + [U(r \odot h_{\langle t-1\rangle})]_j\big) \tag{4}$$

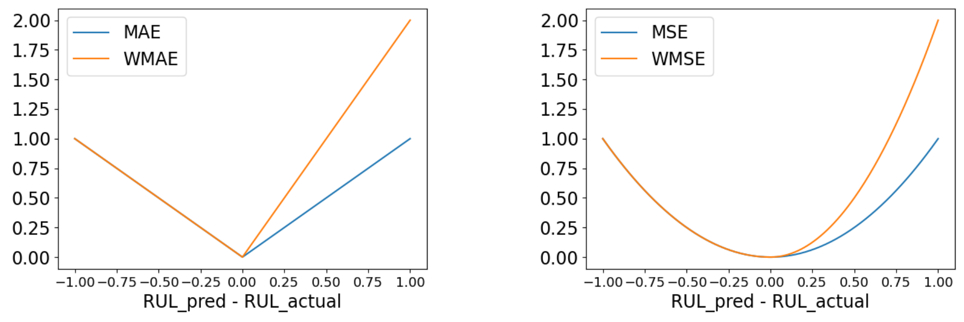
2.1 Loss/Error Formulation

When performing regression tasks two typical loss function/error formulations are mean squared error (MSE) (sometimes root mean squared error, RMSE) and mean absolute error (MAE). These functions are defined as

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\big(\hat{y}_i - y_i\big)^2 \tag{5}$$

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}\big|\hat{y}_i - y_i\big| \tag{6}$$

where subscript i indicates a given datapoint, $\hat{y}_i$ is the prediction for point i, and y_i is the true target for that point. As mentioned earlier, these formulations of error do not account for the economics



(a) Mean absolute error

(b) Mean square error

Figure 1: Comparison of unweighted and weighted loss/error formulations for $C = 2$

of the problem at hand as they purely consider deviation from the target value and not the direction in which that deviation occurs. Augmenting these loss functions through the use of a weighting function is proposed in order to modify their behavior.

Much work on loss function weighting has been conducted. Rengasamy et al. (2020) discusses dynamically weighting mean square error in order to disproportionately penalize predictions outside of a specified absolute error threshold in order to increase accuracy of predictions. Other applications and merits of weighted loss functions are discussed in Zhou et al. (2021), Zhang et al. (2019), and Rezaei-Dastjerdehei et al. (2020). Selection of a weighted MSE is made for this work as its gradient increases as error increases, which is not the case with MAE formulations. The higher gradient helps promote training that minimizes these large errors.

Equation 7 is the proposed weighted loss formulation for this problem and will be referred to as the weighted mean square error (WMSE). The weights w_i are defined in Equation 8 and the parameter $C \geq 1$.

$$\text{WMSE} = \frac{1}{n}\sum_{i=1}^{n} w_i\big(\hat{y}_i - y_i\big)^2 \tag{7}$$

$$w_i = \begin{cases} C & \hat{y}_i - y_i > 0 \\ 1 & \hat{y}_i - y_i \leq 0 \end{cases} \tag{8}$$

In this expression of error, an additional penalty is applied in situations where the prediction of RUL, $\hat{y}_i$, exceeds the target value y_i (which corresponds with the case of predicting working conditions when a failure state actually exists). When training, higher loss will be seen for unfavorable predictions like this and help promote finding a model where predictions are biased away from this more costly scenario.

A similar weighting expression can be applied for MAE which is commonly used to measure the performance of the model on the test data. Selection of appropriate weights in this case will be dependent on the economics of the problem in question which is unknown for the simulated dataset being studied here. The weighted MAE (WMAE) for this work is defined in Equation 9 and uses the same weighting scheme presented in Equation 8 (selection of C will be problem dependent). The effect the weighting has in creating both the WMSE and WMAE functions can be seen in Figure 1 where an increase in function slope is observed when the predicted RUL value is greater than the target.

$$\text{WMAE} = \frac{1}{n}\sum_{i=1}^{n} w_i\big|\hat{y}_i - y_i\big| \tag{9}$$

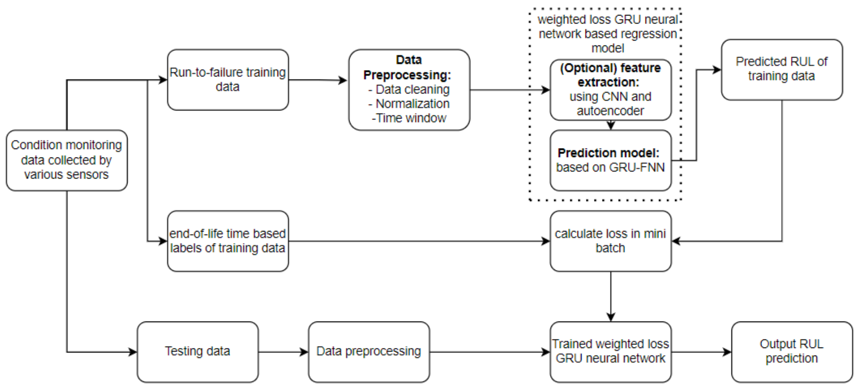

weighted loss GRU neural network based regression model
Condition monitoring data collected by various sensors
Run-to-failure training data
Data Preprocessing: - Data cleaning - Normalization -Time window
(Optional) feature extraction: using CNN and autoencoder
Prediction model: based on GRU-FNN
Predicted RUL of training data
end-of-life time based labels of training data
calculate loss in mini batch
Testing data
Data preprocessing
Trained weighted loss GRU neural network
Output RUL prediction

Figure 2: Flow chart for model

Details of the full approach used for this work can be seen in Figure 2. The upper branch of the flow chart details the training process while the lower branch shows the testing process/operation of the deployed model.

3 Data

The data used for this work comes from the NASA C-MAPPS aircraft gas turbine engine simulation software which is available to the public (Teubert, 2023). The generation of this data set is discussed in detail by Saxena et al. (2008). A full simulation model of the engine is used to study damage propagation through the engine by means of measuring loss of efficiency and deterioration of flow through the engine over time, beginning at a specified fault time. The data is generated as a multi-variate time series of sensor measurements through the engine model in addition to recording the operational settings. The sensor measurements are contaminated with noise. Examples of what the data signals include fan inlet total temperature, fan speed, and engine pressure ratio. Figure 3 shows three sample time signals from some of the sensors (which the dataset identifies by just a number).

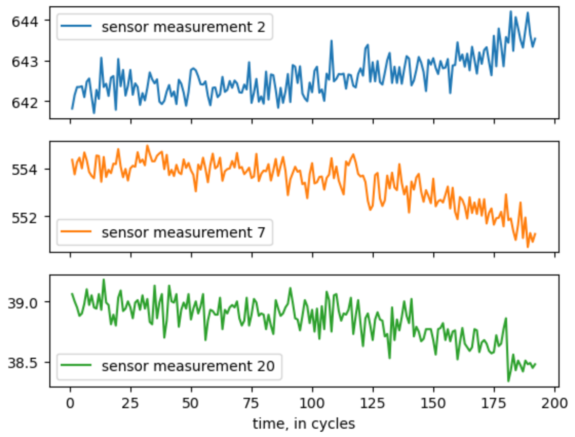


Figure 3: Sample signals from three of the sensors

The C-MAPPS data contains four distinct sets of data, each having a different combination of fault and operating conditions in the simulation. Each of these datasets has a provided training/test split. For the training data, the time series begin at normal operating conditions and run until the engine experiences failure, corresponding with zero RUL. The test set is separate and is a set of signals for each of the measurements/settings which end at a time before failure and are accompanied by the RUL for the end point of those signals. The properties of each dataset, denote FD001 through FD004 are summarized in Table 2. In general, FD001 is the least complex dataset to predict on and FD004 has the most complex behavior.

Table 2: Dataset Summaries

	FD001	**FD002**	**FD003**	**FD004**
Training Time Series	100	260	100	249
Testing Time Series	100	259	100	248
Fault Conditions	1	1	2	2
Operational Conditions	1	6	1	6

3.1 Preprocessing

In order to handle the time series data several steps of preprocessing are required. The training data for each engine unit begins with a period of time of normal operating conditions. To better predict the RUL for when the system begins to deteriorate, Heimes (2008) discusses creating a piecewise linear function for the training data RUL which can be expressed as $\min(\text{RUL}_{\text{actual}}, 130)$. Figure 4 displays the adjustment to target RUL. This recommendation is used for the work done here.

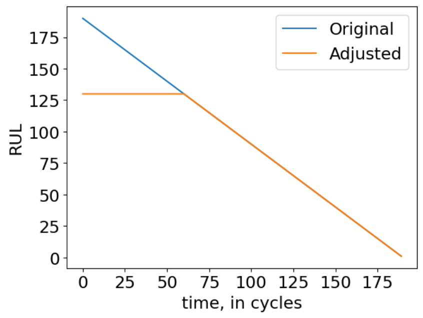
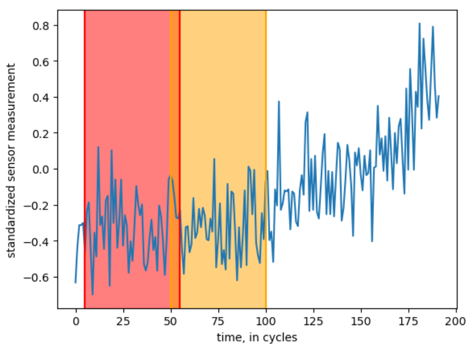


Figure 4: Target RUL adjustment

The signals from each of the sensors have a wide variety of ranges and magnitudes. Feeding these raw signals into the model leads to issues of convergence and accuracy. To alleviate this issue and standardization of the data is performed. Min-max scaling is used on the range $[-1, 1]$ for each of the signals. Min-max scaling onto this range or $[0, 1]$ is found to be effective for RUL prediction and is commonly used in work on this dataset (Li et al., 2020)(Sun et al., 2022).

The raw datasets provide between 100 and 260 time series for training. This is too little data to train a complex model on when using each full time series as a training point. Additionally, as all the training signals end with zero RUL, this would not be an appropriate way to train the model to give non-zero predictions of RUL. To alleviate these issues a data augmentation technique known as data windowing is applied.

Data windowing takes a time signal and samples from the signal for a prescribed number of time steps using a sliding window on the time signal (da Silva et al., 2021). An illustration of this can be seen in Figure 5 on a sample of the data used for this work. The two shaded boxes show two sets of data being extracted from the raw signals as windowed samples. The set of all windowed samples then becomes the data used to the train the model. Selection of the window size is a hyperparamter that affects the performance of the model on the test data. The recommendation for window sizes of 50,20,30, and 15 for the FD001-004 datasets (in order) presented in Li et al. (2020) are used for this work.


(a) Sample signal with two windows colored

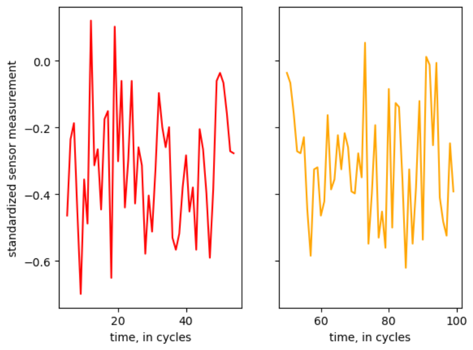

(b) Signals extracted from the red and orange windows

Figure 5: Example of data windowing process

3.2 Feature Extraction

Feature extraction on the raw data signals was planned using an autoencoder as well as a convolutional neural network (CNN). The implementation of this approach was not completed to a point where it produced useful results in the context of generating better RUL predictions when compared with not performing the extraction process. Details of what was attempted/completed are included in Appendix A.

4 Experimental Setup

The machine learning model for this work is implemented using PyTorch and uses the methods described in Section 2. Details of the implementation can be found in Table 3. Model training and testing is carried out on a Windows PC with an Intel i5 CPU, 32 GiB RAM, and an NVIDIA GeForce RTX 3080ti GPU. The model is executed on the GPU for large gains in training and testing speed. The Adam optimizer is used as a stochastic method for training the prediction model.

Table 3: Model Architecture

	Input Size	**Num Layers**	**Output Size**
Gated Recurrent Unit	24	2	256
ReLU Activation	256	-	256
Fully Connected Layer	256	1	1

Evaluation of the model is performed for each of the feature extraction methods described across a range of weighting values for the WMSE loss formulation (Equations 7 & 8). The performance of the model will be evaluated using several metrics. The MAE on the test data as well as R^2 value are recorded as is done in Li et al. (2020). MAE is typically reported for RUL prediction tasks to evaluate average model performance while the R^2 value, otherwise known as the coefficient of determination, provides insight into the variance of the model of the test data (values close to one are desired). Additionally, histograms of the error in RUL prediction are saved for each of the runs along with noting the number of predictions which are "conservative," meaning the predicted RUL is less than the target value (lower cost scenario). Weighting factor values in the range $C = [1, 1000]$ are used for the study with specific values for each case being selected to understand any patterns in the results.

For a subset of these tests, the idea of using WMAE (Equation 9) explored to attempt to translate the model predictions to a rough estimate of the total cost for an operator of the system to perform maintenance. Comparison of the weights used to constructed the loss function (WMSE) to those used to evaluate the model (WMAE) is investigated for any insights.

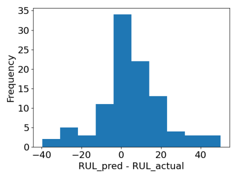
5 Results

Performance of the GRU model is first studied on the FD001 dataset. Table 4 contains the metrics used to evaluate the RUL predictions across a range of loss weighting values for this setup. Figure 6 contains histograms for the measured error on the test set.

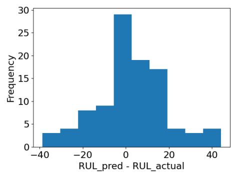
Table 4: FD001 prediction results

Weighting Factor C	MAE	R^2	# Conservative Predictions
1	11.9	0.83	35
10	11.2	0.85	39
50	11.8	0.84	46
100	12.1	0.86	55
200	12.1	0.84	60
300	13.0	0.84	68
500	13.4	0.80	89
750	17.0	0.73	83
1000	31.3	0.22	90

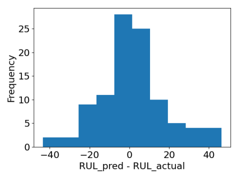
As one would expect, the addition of the penalty weighting in the loss formulation promoted training that avoids the scenario of overpredicting RUL. This is exemplified in the test results with the number of conservative predictions increasing with higher weighting factors. The histograms of error also depict this phenomena as the histogram bars shift towards negative error values (decreasing predictions of RUL) as the weighting value increases.



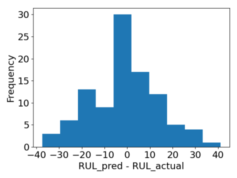
(a) $C = 1$



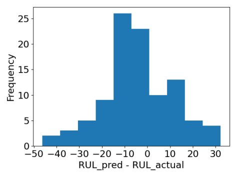
(b) $C = 10$



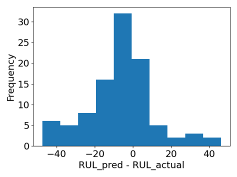
(c) $C = 50$



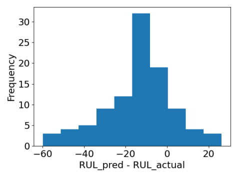
(d) $C = 100$



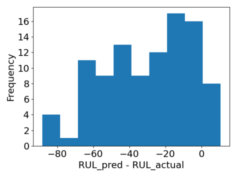
(e) $C = 300$



(f) $C = 500$



(g) $C = 750$



(h) $C = 1000$

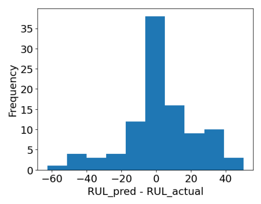
Figure 6: Error histograms for FD001

Interestingly, the results for MAE on the test data as well as R^2 values are approximately the same across the range of weigh values of 1 through 300 with a measurable degradation in performance starting with the weight of $C = 500$. Variations in the observed MAE and R^2 in this range cannot be definitively explained due to the stochastic nature of the training process leading to some noise in the output as well as differences being on the order of only a few percent. Considering this, it is observed $C = 10$ yields the lowest MAE and $C = 100$ the highest R^2 value.

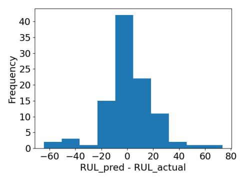
Similar behavior is observed for the FD002 through FD004. Tables 5-7 detail the model performance across ranges of training weights. Figure 7-9 provide histogram plots of the error observed for these datasets. Higher minimum MAE was achieved on FD002-004 when compared to FD001. This is expected as these datasets were created under more complex conditions for prediction of RUL. FD002 and FD003 both show a minimum MAE being achieved with a greater than one weighting factor while FD004 performed best in this metric at $C = 1$. Minimized MAE, however, has to be balanced with the rate of over predictions (opposite of number of conservative predictions) when evaluating how well the model performs in the true economic context of the final deployed model. The weighted loss GRU setup also makes predictions with good values of R^2 on these datasets. The trend of increasing number of conservative predictions is likewise observed across all four datasets (accounting for noise/stochastic nature of training).

Table 5: FD002 prediction results

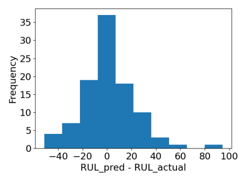
Weighting Factor C	**MAE**	**R^2**	**# Conservative Predictions**
1	15.1	0.76	51
10	13.6	0.78	51
50	15.3	0.74	57
100	15.8	0.76	67
200	19.5	0.65	79
500	23.2	0.54	83



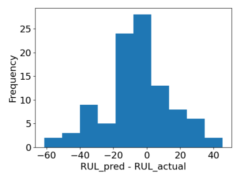
(a) $C = 1$



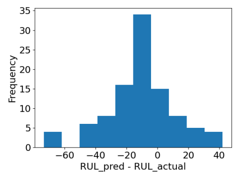
(b) $C = 10$



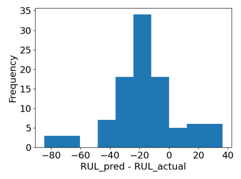
(c) $C = 50$



(d) $C = 100$



(e) $C = 200$



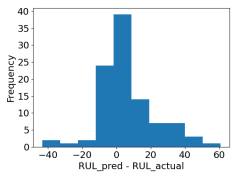
(f) $C = 500$

Figure 7: Error histograms for FD002

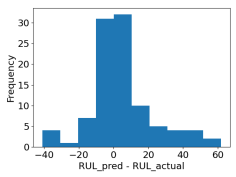
Table 6: FD003 prediction results

Weighting Factor C	**MAE**	**R^2**	**# Conservative Predictions**
1	11.7	0.80	39
10	12.0	0.78	41
50	12.7	0.79	43
100	11.7	0.84	61
200	12.0	0.84	65
300	13.1	0.82	70
400	14.1	0.80	69
500	14.4	0.78	78
1000	17.0	0.73	86

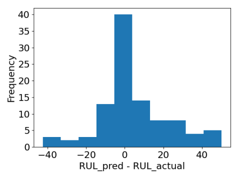
The performance of the weighted model is compared to other implemented models in Table 8. For this comparison, the training weighting for each dataset yielding the lowest MAE was selected. Data for benchmarking comes from Li et al. (2020). Across the board, this implementation yielded comparable or better results than the other works presented.



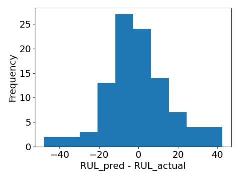
(a) $C = 1$



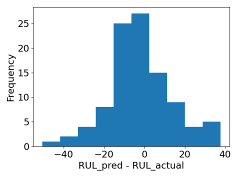
(b) $C = 10$



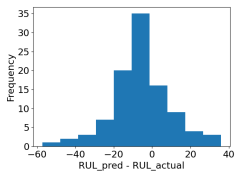
(c) $C = 50$



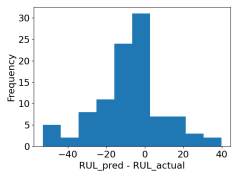
(d) $C = 100$



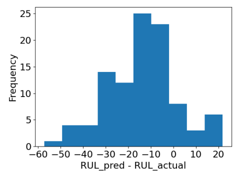
(e) $C = 200$



(f) $C = 300$



(g) $C = 500$



(h) $C = 1000$

Figure 8: Error histograms for FD003

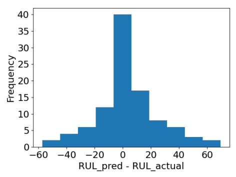
Table 7: FD004 prediction results

Weighting Factor C	**MAE**	R^2	**# Conservative Predictions**
1	15.9	0.71	45
10	16.4	0.69	59
50	17.4	0.71	69
100	20.0	0.63	67
200	22.0	0.58	77
300	21.2	0.59	82
400	22.2	0.58	79
500	23.4	0.50	81

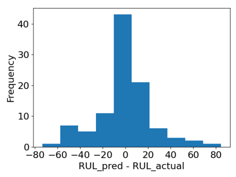
The predictions for the FD001 model were also evaluated using WMAE. Figure 10 shows the results of evaluating the model trained with varying training weighting factors (x-axis) by various error weights (different series). Some interesting trends can be observed in this data. In general, as the error weight increases, the weighted error functions achieves it's minimum value with increasing values of the training weighting factor. This makes sense as higher training weighting factors shift predictions away from the region that is penalized higher by the WMAE. As the penalty value increases, achieving a minimum error/cost requires more conservative predictions to avoid the penalty. Thus, with knowledge of what the penalty weight for WMAE would be for the economics of a given problem, an optimal training weighting factor can be obtained. This weighting factor will lead to the

Table 8: Model Benchmarking

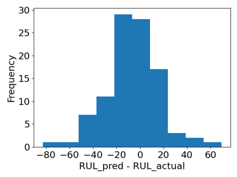
Dataset	**Metrics**	**LSTM**	**Bi-LSTM**	**SAE-GRU**	**Weighted Loss GRU**
FD001	MAE	13.6	14.4	13.5	**11.2**
	R^2	0.77	0.75	0.79	**0.85**
FD002	MAE	44.2	19.9	22.0	**13.6**
	R^2	-0.05	0.72	0.69	**0.78**
FD003	MAE	17.1	20.8	16.4	**11.7**
	R^2	0.51	0.25	0.57	**0.84**
FD004	MAE	47.9	28.1	25.4	**15.9**
	R^2	-0.20	0.53	0.56	**0.71**



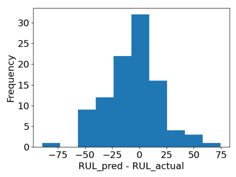
(a) $C = 1$



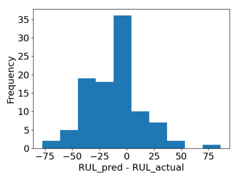
(b) $C = 10$



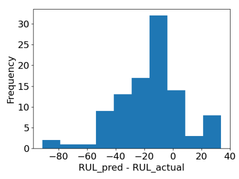
(c) $C = 50$



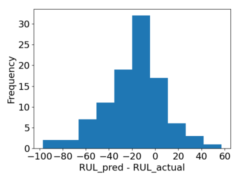
(d) $C = 100$



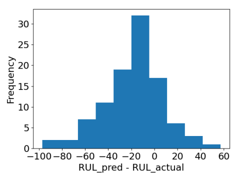
(e) $C = 200$



(f) $C = 300$



(g) $C = 400$



(h) $C = 500$

Figure 9: Error histograms for FD004

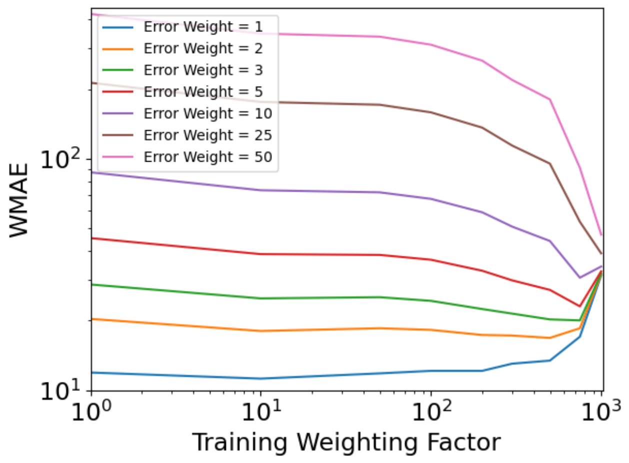


Figure 10: Evaluating FD001 model with various WMAE weights

model that has the lowest expected cost for each prediction it gives (the averaging performed in the WMAE formulation makes its output akin to an expectation). Using this error metric then allows a business to select a model that will save them the most money.

6 Concluding Remarks

A GRU based remaining useful life prediction model is explored using a simulated jet engine dataset. Use of a weighted mean square error loss function is shown to be effective in training the model. Typical evaluation metrics (MAE, R^2) are not notably changed in the presence of relatively small weighting values while creating a notable shift in the distribution of RUL predictions towards more conservative predictions as is desired. Use of WMAE to evaluate model performance is able to incorporate some insight of the economic costs involved in the failure prediction problem and give a more meaningful metric to judge the model by.

Future work should be done to investigate using more complex loss formulations that may promote even more effective training of the model. Training of the model with additional loss contributions from intermediate points in the training data signals may also provide benefits in improved training of the model. Application of this model structure should also be applied to other datasets for RUL prediction.

References

Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham Taylor, and Daniel Silver (eds.), *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pp. 37–49, Bellevue, Washington, USA, 02 Jul 2012. PMLR. URL `https://proceedings.mlr.press/v27/baldi12a.html`.

Mikel Canizo, Isaac Triguero, Angel Conde, and Enrique Onieva. Multi-head cnn–rnn for multi-time series anomaly detection: An industrial case study. *Neurocomputing*, 363:246–260, 2019. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2019.07.034. URL `https://www.sciencedirect.com/science/article/pii/S0925231219309877`.

Thyago P. Carvalho, Fabrízzio A. A. M. N. Soares, Roberto Vita, Roberto da P. Francisco, João P. Basto, and Symone G. S. Alcalá. A systematic literature review of machine learning methods applied to predictive maintenance. *Computers & Industrial Engineering*, 137:106024, November 2019. doi: 10.1016/j.cie.2019.106024. URL `https://doi.org/10.1016/j.cie.2019.106024`.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

Matheus Almeida Farias da Silva, Rafael Lima De Carvalho, and Tiago da Silva Almeida. Evaluation of a sliding window mechanism as DataAugmentation over emotion detection on speech. *Academic Journal on Computing, Engineering and Applied Mathematics*, 2(1):11–18, apr 2021. doi: 10.20873/uft.2675-3588.2021.v2n1.p11-18. URL `https://doi.org/10.20873%2Fuft.2675-3588.2021.v2n1.p11-18`.

Felix O. Heimes. Recurrent neural networks for remaining useful life estimation. In *2008 International Conference on Prognostics and Health Management*. IEEE, October 2008. doi: 10.1109/phm.2008.4711422. URL `https://doi.org/10.1109/phm.2008.4711422`.

Xuebo Jin, Xinghong Yu, Xiaoyi Wang, Yuting Bai, Tingli Su, and Jianlei Kong. Prediction for time series with cnn and lstm. In Rui Wang, Zengqiang Chen, Weicun Zhang, and Quanmin Zhu (eds.), *Proceedings of the 11th International Conference on Modelling, Identification and Control (ICMIC2019)*, pp. 631–641, Singapore, 2020. Springer Singapore. ISBN 978-981-15-0474-7.

Yaguo Lei, Naipeng Li, Liang Guo, Ningbo Li, Tao Yan, and Jing Lin. Machinery health prognostics: A systematic review from data acquisition to RUL prediction. *Mechanical Systems and Signal Processing*, 104:799–834, May 2018. doi: 10.1016/j.ymssp.2017.11.016. URL `https://doi.org/10.1016/j.ymssp.2017.11.016`.

Joerg Leukel, Julian González, and Martin Riekert. Adoption of machine learning technology for failure prediction in industrial maintenance: A systematic review. *Journal of Manufacturing Systems*, 61:87–96, October 2021. doi: 10.1016/j.jmsy.2021.08.012. URL `https://doi.org/10.1016/j.jmsy.2021.08.012`.

Li Li, Zhen Zhao, Xiaoxiao Zhao, and Kuo-Yi Lin. Gated recurrent unit networks for remaining useful life prediction. *IFAC-PapersOnLine*, 53(2):10498–10504, 2020. doi: 10.1016/j.ifacol.2020.12.2795. URL `https://doi.org/10.1016/j.ifacol.2020.12.2795`.

Guixian Qu, Tian Qiu, Yang Si, Qiyu Yuan, Qinglin Ma, and Chenghao Wang. Remaining useful life prediction for aero-engine based on hybrid cnn-gru model. In *2022 IEEE International Conference on Unmanned Systems (ICUS)*, pp. 1523–1528, 2022. doi: 10.1109/ICUS55513.2022.9987018.

Divish Rengasamy, Mina Jafari, Benjamin Rothwell, Xin Chen, and Grazziela P. Figueredo. Deep learning with dynamically weighted loss function for sensor-based prognostics and health management. *Sensors*, 20(3):723, jan 2020. doi: 10.3390/s20030723. URL `https://doi.org/10.3390%2Fs20030723`.

Mohammad Reza Rezaei-Dastjerdehei, Amirmohammad Mijani, and Emad Fatemizadeh. Addressing imbalance in multi-label classification using weighted cross entropy loss function. In *2020 27th National and 5th International Iranian Conference on Biomedical Engineering (ICBME)*, pp. 333–338, 2020. doi: 10.1109/ICBME51989.2020.9319440.

David E. Rumelhart and James L. McClelland. *Learning Internal Representations by Error Propagation*, pp. 318–362. 1987.

Abhinav Saxena, Kai Goebel, Don Simon, and Neil Eklund. Damage propagation modeling for aircraft engine run-to-failure simulation. In *2008 International Conference on Prognostics and Health Management*. IEEE, October 2008. doi: 10.1109/phm.2008.4711414. URL `https://doi.org/10.1109/phm.2008.4711414`.

J.Z. Sikorska, M. Hodkiewicz, and L. Ma. Prognostic modelling options for remaining useful life estimation by industry. *Mechanical Systems and Signal Processing*, 25(5):1803–1836, July 2011. doi: 10.1016/j.ymssp.2010.11.018. URL `https://doi.org/10.1016/j.ymssp.2010.11.018`.

Junjie Sun, Lehui Zheng, Ying Huang, and Yu Ge. Remaining useful life prediction based on CNN-BGRU-SA. *Journal of Physics: Conference Series*, 2405(1):012007, December 2022. doi: 10.1088/1742-6596/2405/1/012007. URL `https://doi.org/10.1088/1742-6596/2405/1/012007`.

Chris Teubert. Cmapss jet engine simulated data, Jan 2023. URL `https://data.nasa.gov/Aerospace/CMAPSS-Jet-Engine-Simulated-Data/ff5v-kuh6`.

Jonas Teuwen and Nikita Moriakov. Chapter 20 - convolutional neural networks. In S. Kevin Zhou, Daniel Rueckert, and Gabor Fichtinger (eds.), *Handbook of Medical Image Computing and Computer Assisted Intervention*, The Elsevier and MICCAI Society Book Series, pp. 481–501. Academic Press, 2020. ISBN 978-0-12-816176-0. doi: https://doi.org/10.1016/B978-0-12-816176-0.00025-9. URL `https://www.sciencedirect.com/science/article/pii/B9780128161760000259`.

Wen Zhang, Yuhang Du, Taketoshi Yoshida, and Ye Yang. DeepRec: A deep neural network approach to recommendation with item embedding and weighted loss function. *Information Sciences*, 470:121–140, jan 2019. doi: 10.1016/j.ins.2018.08.039. URL `https://doi.org/10.1016%2Fj.ins.2018.08.039`.

Ziyun Zhou, Hong Huang, and Binhao Fang. Application of weighted cross-entropy loss function in intrusion detection. *Journal of Computer and Communications*, 09(11):1–21, 2021. doi: 10.4236/jcc.2021.911001. URL `https://doi.org/10.4236%2Fjcc.2021.911001`.

A Feature Extraction

Two methods for performing this extraction automatically are by convolution neural networks (CNN) and auto-encoders.

Much work has been in the field of using CNNs for processing time series type data. For instance, a CNN was used in order to determine weather patterns through Beijing meteorological data Jin et al. (2020). In regards to industrial systems, a successful implementation involves CNNs used to detect anomalies in time-series data of an elevator system Canizo et al. (2019).

The model used for feature extraction is based on Convolutional Neural Networks (CNNs). In the context of this problem, the CNN will be applied to time-dependent data along with an autoencoder so that the information passed into the GRU has more meaning. The justification of utilizing CNNs for this task rather than another type of model is that CNN models present several advantages over other models such as their ability to extract deep features independent from time along with their resistance to noise. This problem requires a multi-scale CNN consisting of 3 parallel independent branches which extracts different aspects of the data. The framework consists of the transformation, local convolution, and full convolution.

Feature extraction depends on CNNs for this type of data. CNNs are multidimensional neural networks which consist of a convolutional layer, an activation layer, and a pooling layer. The convolution layer extracts features through adjusting convolution via weights and biases on a convolution filter.

$$y_n = w_m * x_n + b_m \tag{10}$$

with $w_m \in R^{h \times w} and \, b_m \in R^{h \times w}$ respectively indicating weight and bias within a specific convolution. Teuwen & Moriakov (2020)

Similarly, auto-encoders are often used to reduce the dimensionality of an input set. An autoencoder constitutes a neural network where the input is compressed into a lower dimensional latent-space representation and then the output is reconstructed from the compact initial latent-space representation. Rumelhart & McClelland (1987) Autoencoders must determine A(encoder) : $\Re^p - > \Re^n$ and B (decoder): $\Re^p - > \Re^n$ which satisfy the following expression:

$$argmin_{A,B} E[\Delta(x, B \cdot A(x)] \tag{11}$$

where E represents the expectation from the distribution of x and Δ represents the reconstruction loss function, indicating the difference between an input and the reconstructed output Baldi (2012)

In order to provide more meaningful input to the GRU layer(s) of the network, feature extraction processes are used to reduce the dimensionality of the input data. Reduced dimensionality, aside from reduced computation time, ensures less complexity in the input data and makes it easier to determine trends in a model.

The progress made regarding feature extraction involved the construction of a CNN alongside an analysis of its ability through MAE. Our objective was to achieve a MAE in the range of 15-20. However, the lowest MAE possible through the implementation of this CNN was 55.8 for 100 epochs.