Asymmetric Loss Functions for Deep Learning Early Predictions of Remaining Useful Life in Aerospace Gas Turbine Engines

Divish Rengasamy[1], Benjamin Rothwell [1], Grazziela P Figueredo[2,3]

[1] Gas Turbine and Transmissions Research Centre

[2]The Advanced Data Analysis Centre

[3]School of Computer Science, The University of Nottingham, UK

Email: {Divish.Rengasamy, Benjamin.Rothwell, Grazziela.Figueredo}@nottingham.ac.uk

***Abstract*—Asymmetric loss functions have been successfully applied to deep learning for image analysis and imbalanced classification. In this paper, we extend the use of particular types of weighted loss functions, namely asymmetric loss functions, to investigate how predictions of engine remaining useful life (RUL) in aerospace are affected. Within prognostics and health management, the main metric used to evaluate deep learning RUL predictions is the scoring function. Our hypothesis is that by using asymmetric loss functions we will improve results for this metric. In order to investigate our hypothesis, we test 4 different asymmetric loss functions, i.e, Mean Square Logarithmic Error-Mean Square Error, Linear-Mean Square Error, Linear-Linear, and Quadratic-Quadratic and evaluate whether and how much they affect different deep learning architectures performance. Results show that the use of asymmetric loss functions improve RUL predictions for the case study investigated.**

***Index Terms*—Deep Learning, Weighted Loss Functions, Asymmetric Loss Functions, Prognostic Health and Management, Condition-Based Management, Remaining Useful Life. Predictive Maintenance, Aerospace Maintenance Repair and Overhaul**

I. Introduction

In aerospace industry, modern aircrafts are fitted with a wide array of sensors with the objective to optimise safety, economy and efficiency. Sensors constantly perform surveillance of equipment parts and collect information on their current condition. Examples of the type of data collected are components' temperature, vibration, pressure and possible faults. The data collected are used for automated, intelligent diagnostics and prognostics that will determine the aircraft's maintenance needs. This methodology using sensor data coupled with intelligent methods for prognostics and health management is known as condition-based maintenance (CBM). One of the main goals of CBM for aerospace is to accurately predict the Remaining Useful Life (RUL) of an equipment. For safety reasons, effective CBM solutions for those problems are achieved when early or exact RUL predictions are performed, as opposed to late predictions. And we are particularly interested in predictions for aircraft gas turbine engines.

Over the past decade, deep learning has been used as a tool to assist CBM for aerospace with satisfactory results [1]. The role of deep learning in this context is to make sense of the sensor data captured, and to perform reliable predictions of the current health state of the aircraft parts. In this paper, we extend the work from Rengasamy *el al.* [2] and propose the use of asymmetric loss functions for exact and early predictions of gas turbine engines' RUL. Asymmetric loss functions work by modifying the learning phase and influencing the loss corresponding to the the prediction error of each data instance. These functions have been successfully coupled with deep learning for image analysis [3], [4], handling imbalanced dataset [3], [5] and economic forecasting [6]–[8]. A review of the literature has revealed that those functions are not widely used in sensor data. This is mostly due to the fact that they add bias to the learning process and are likely to affect the overall error of predictions in time series and signal data. For aerospace CBM, however, there is the need to penalise solutions that produce late RUL predictions, as they increase safety risks. In addition, the literature survey shows that current research in data-driven CBM mostly focuses on assessing the predictive power of different deep learning architectures. To the best of our knowledge, the investigation of different types of asymmetric loss functions for deep learning RUL predictions is scarce.

To address this gap, we investigate 4 asymmetric loss functions, namely, Mean Square Logarithmic Error-Mean Square Error (MSLE-MSE), Linear-Mean Square Error (LIN-MSE), Linear-Linear (LIN-LIN), and Quadratic-Quadratic (QUAD-QUAD). In addition, we assess LIN-MSE and LIN-LIN performance under different parameters. The objective is to force deep learning models to penalise those instances where late prediction errors occur. We employ the loss functions investigated to Deep Feedforward Neural Network, 1-Dimensional Convolutional Neural Network, and Bidirectional Long Short-Term Memory architectures. The methodology is tested on the Commercial Modular Aero-Propulsion System Simulation (CMAPSS) data from NASA [9]. Experimental results show that asymmetric loss functions help us achieve significant improvement for RUL prediction for this case study.

This paper is organised as follows. Section II provides a background on loss functions and asymmetrical loss functions. In addition, it reviews the deep learning architectures tested in this paper applied to RUL. Sections III outlines the methodology and experimental design adopted. Section IV presents the results and discussions. Finally, the conclusions and future work are drawn in Section V.

II. Background

This section provides the background on asymmetric loss functions, the damage propagation within a gas turbine engine CMAPSS data set, and the current approaches to predicting RUL for CMAPSS using deep learning models for aerospace applications.

A. Asymmetric Loss Functions

The process of training a deep learning neural network aims at adjusting the network's weights to enable the mapping of a set of inputs to into a set of outputs, based on the available training data. A neural network model is usually trained using the backpropagation algorithm with stochastic gradient descent optimisation to obtain the most suitable set of weights that will minimise prediction error for the training set. Within deep learning, loss function represents the error that needs to be minimised for training process. In this paper, we use the term loss function interchangeably with cost function, error function and the objective function.

For regression tasks the typical loss functions are the absolute error loss (AE) and the squared error loss. Given x as an array of attributes (or independent variables) within the training set, y as the dependent variable and $\hat{y}$ the deep neural network predictions for y, the absolute error function f_{AE} is calculated as the difference between actual and predicted values, as follows:

$$f_{AE}(\hat{y_i}, y_i) = |\hat{y} - y| \tag{1}$$

The square error loss function f_{SE} is determined by the following equation:

$$f_{SE}(\hat{y_i}, y_i) = (\hat{y} - y)^2 \tag{2}$$

Using f_{AE} as an example, we can transform the loss function from (1) to a weighted loss function, by multiplying it with a weight a, as follows:

$$f_{weightedAE}(a, \hat{y_i}, y_i) = a|\hat{y} - y| \tag{3}$$

The weight, a enable the control of loss's magnitude by changing gradient of loss function. Additionally, we can redefine the weighted f_{AE} as follows:

$$f_{weightedAE}(a, \hat{y_i}, y_i) = \begin{cases} -a(\hat{y} - y) & \text{if } d \text{ is} \leq 0 \\ a(\hat{y} - y) & \text{otherwise} \end{cases} \tag{4}$$

Where $d = \hat{y_i} - y_i$. To modify the weighted loss function from (4) to an asymmetric loss function, we assign another weight parameter, b when $\hat{y_i} - y_i > 0$ as follows:

$$f_{AsymmetricLoss}(a, \hat{y_i}, y_i) = \begin{cases} -a(\hat{y} - y) & \text{if } d \text{ is} \leq 0 \\ b(\hat{y} - y) & \text{otherwise} \end{cases} \tag{5}$$

In cases when $a \neq b$ the function represented in (5) becomes a LIN-LIN asymmetric loss function.

The deep learning models are then implemented using the asymmetric loss function instead of the traditional symmetric loss functions, such as MSE and MAE. In addition to the LIN-LIN loss functions, we also employs 3 other asymmetric loss functions, namely, LIN-MSE, MLSE-MSE, and QUAD-QUAD as follow:

$$LIN\text{-}MSE = \begin{cases} -a(\hat{y} - y) & \text{if } d \text{ is} \leq 0 \\ \dfrac{\sum_{i=1}^{N}(\hat{y_i} - y_i)^2}{N} & \text{otherwise} \end{cases} \tag{6}$$

$$MSLE\text{-}MSE = \begin{cases} \dfrac{\sum_{i=1}^{N}(\log\hat{y_i} - \log y_i)^2}{N} & \text{if } d \text{ is} \leq 0 \\ \dfrac{\sum_{i=1}^{N}(\hat{y_i} - y_i)^2}{N} & \text{otherwise} \end{cases} \tag{7}$$

$$QUAD\text{-}QUAD = \begin{cases} 2a(\hat{y_i} - y_i)^2 \\ & \text{if } d \text{ is} \leq 0 \\ 2(a + (1 - (2a)))(\hat{y_i} - y_i)^2 \\ & \text{otherwise} \end{cases} \tag{8}$$

The behaviour of the different loss functions investigated is illustrated in Fig 1. From Fig 1, we observe that when the $error < 0$ ($error = \hat{y} - y$), the loss values are lower compared to $error \geq 0$ using asymmetric loss functions. The asymmetric loss functions therefore allow more severe penalisation for late RUL prediction.

B. Damage Propagation in Gas Turbine Engine

As mentioned in Section I, the gas turbine engine degradation data used for testing asymmetric loss functions in this work is CMAPSS [9]. The data is obtained from a high fidelity simulation of a complex thermo-dynamical system that closely models a real aerospace engine. The failure of the simulated engine is initiated through random point of deterioration. The deterioration continues with increasingly worsening effect. This is modelled after the Arrhenius Model, Coffin-Mason Mechanical Crack Growth Model, and Eyring Model [9], [10]. The commonality between the three deterioration models listed are the exponential behaviour of fault evolution. The system measures the loss of efficiency and flow until a failure criterion is reached. The degradation and damage propagation trend are modelled as follows:

$$h(t) = 1 - exp(a(t)t^{b(t)}) \tag{9}$$

Equation (9) is a generalised Equation for the health index of gas turbine engine, $h(t)$ with respect to time. Subsequently, the system will include a non-zero initial degradation, d. The initial non-zero degradation are common in real system due to manufacturing inefficiencies or error.

$$h(t) = 1 - d - exp(a(t)t^{b(t)}) \tag{10}$$

The decay of efficiency, $e(t)$ and the loss of flow, $f(t)$ can be described using (10) as follow:

$$e(t) = 1 - d_e - exp(a_e(t)t^{b_e(t)}) \tag{11}$$

$$f(t) = 1 - d_f - exp(a_f(t)t^{b_f(t)}) \tag{12}$$

Efficiency and flow are modelled separately as different faults exhibit different trajectories of degradation for each of the terms. The terms from (11) and (12) are combined to as follow form the final model of damage propagation in gas turbine engine:

$$H(t) = g(e(t), f(t)) \tag{13}$$

Using the damage propagation model in (13) the data is generated as follows:

1) Define initial deterioration parameters, e_0 and f_0 for (11) and (12).
2) Impose an exponential rate of change for flow and efficiency loss for each data set, denoting an otherwise unspecified fault location with increasingly worsening effect by setting the parameter a and b in (10).
3) Stop when failure criterion (loss of flow and efficiency) is reached. The failure criterion in this case is when $H(t) = 0$
4) Add mixture noise model into final output data reflect real world scenario. Additionally, the feature data are collected from the sensors measurements listed in Table I. By combining the output and feature data, the complete dataset can be obtained for training and testing deep learning models to perform RUL prediction.

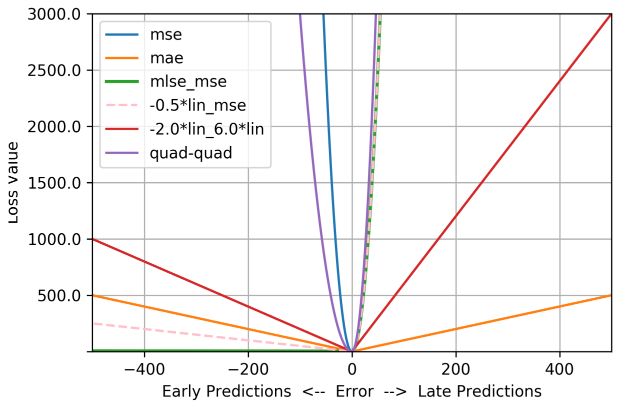

Fig. 1: Symmetric loss functions such as MSE and MAE compared to different asymmetric loss functions, namely MLSE-MSE, LIN-MSE, and LIN-LIN. The numerical value -0.5, -2.0, and 6.0 represents the weights, a and b of the linear part of LIN-MSE and LIN-LIN loss.

C. Related Work on Remaining Useful Life Prediction for CMAPSS

In this section, we review the relevant applications of the deep learning architectures tested with our approach. One of the major shortcomings of deep learning applied to aerospace research, as pointed out by Rengasamy *et al.* [1], is the lack of real-world, publicly available large engine sensor data for benchmarking. For this reason, within the majority of the related literature, including the work reviewed in this section, deep learning architectures are tested only on the Commercial Modular Aero-Propulsion System Simulation (CMAPSS) dataset. CMAPSS is a gas turbine engine degradation data introduced by Saxena and Goebel [9]. This dataset is further described in Section III-A.

Tamilselvan *et al.* [11] applies a Deep Belief Network (DBN) classifier consisting of 3 hidden-layers to CMAPSS. Conjugate gradient approach introduced by Hinton *et al.* [12] fine-tunes the DBN classifier after training. DBN fault classification is compared to Support Vector Machines (SVM), Backpropagation Neural Network (BNN), Self-Organising Maps [13] and Mahalanobis Distance. For the authors' experiments DBN reaches better accuracy for 5 out of 6 operating conditions tested. The authors uses a health index classification approach to determine the RUL based on the remaining cycles.

Zhang *et al.* [14] test Multiobjective Deep Belief Networks Ensemble (MODBNE) to CMAPSS. The authors focus on minimising RUL prediction errors, while maximising the diversity of outputs produced. The optimised DBNs are combined using single-objective differential evolution to create an ensemble of predictors. Results show that MODBNE achieves the most accurate estimation of RUL when compared to DBN, Sequential Kalman Filter, Multi Layer Perceptron, Extreme Learning Machine (ELM), Hierarchical ELM, SVM, least absolute shrinkage and selection operator (LASSO), Extra Tree Regressor, K Neighbors Regressor, Gradient Boosting and Random Forest.

Yuan *et al.* [15] and Zheng *et al.* [16] employ LSTM to predict RUL on CMAPSS. In their approach RUL to is converted to piece-wise RUL. RUL is set to a constant value to mimic the initial condition of the engine before degradation; subsequently, it linearly decreases. LSTM is compared with standard RNN, GRU AdaBoost LSTM, CNN, SVM, Relevance Vector Regression (RVR), and MLP. Results show that the LSTM has the best performance for both RUL estimation and fault occurrence predictions.

Ellefsen *et al.* [17] uses Restricted Boltzmann Machine (RBM) to pretrain the model in an unsupervised manner to automatically generate new degradation related features from the raw data. Subsequently, the newly generated features are used as input for LSTM to predict RUL. Hyperparameters are optimised by Genetic Algorithms (GA). Results show that the combination of RBM and LSTM achieves outcomes comparable those from the state-of-the-art at the time the work was published.

Wang *et al.* [18] show that Bi-LSTM's hidden layers implicitly extract degradation features without unsupervised pretraining of the model, producing good results. Babu *et al.* [19] obtain increase in prediction accuracy using Deep Convolutional Neural Networks (DCNN) when compared to MLP, SVM, and RVR. Li *et al.* [20] uses DCNN to estimate the RUL of aircraft turbofan engines. The authors employ a DCNN and training is conducted using mini-batch gradient descent [21]. Results from the DCNN are compared to LSTM, RULCLIPPER [22], Random Forest, Gradient Boosting, SVM, Echo State Network with Kalman filter [23], Multi-objective deep belief networks ensemble [14] and Time window-based NN [24]. Results reveal that CNN outperforms LSTM, RNN, Deep Neural Network (DNN) for RMSE.

Rengasamy *et al.* [2] uses a dynamically weighted loss function to improve the RUL prediction. The dynamically weighted loss function adjust the loss based on the error produced. Higher error are further penalised more using the dynamically weighted loss function, putting more focus on harder instances similar to the idea of Focal loss [3]. A comparison is carried out between a non-weighted and dynamically weighted loss function using DNN, CNN1D, Bi-LSTM, and Bi-gated recurrent unit (Bi-GRU). Results show that by using dynamically weighted loss function, the deep learning models shows improved RUL score. While a dynamically weighted loss function is employed by Rengasamy *et al.* [2], the loss function can be further improved by incorporating domain knowledge of CBM and penalising late predictions to create an early predictions bias. The early prediction bias in deep learning models should produce an overall better RUL score. Therefore, in the subsequent sections we investigate the difference between asymmetric and symmetric loss functions using deep learning models for RUL predictions.

III. Methodology

This section discusses the methodology and the experimental design to test our approach. We initially discuss the CMAPSS dataset pre-processing process. Subsequently, we present the deep learning architectures employed and the asymmetric loss functions investigated. Finally we introduce the evaluation metrics adopted.

A. Benchmark Dataset

The CMAPSS dataset contains training and test sets with 6 different operating conditions. The training set comprises of the complete engine life cycle data (run until failure) while the testing set data has a RUL range of 10 to 150 cycles. The training data consist of 100 engines with a total of 20631 cycle while the testing data consist of 100 engines with a total of 13096 engine cycles. The dataset consists of the engine unit number, the operating cycle number of each unit, the operating settings and the raw sensor measurements. The raw sensor features are shown in Table I.

1) Data Pre-processing: The CMAPSS data consists of 3 operation settings and 21 sensors features. We discard 8 of the 21 features as they remained constant throughout the gas turbine engine degradation process and provide no useful information. Additionally, we use the $[0, 1]$ normalisation on the features to ensure they are represented equally in the learning process. Furthermore, the value of the maximum cycle

TABLE I: Description of the CMAPSS dataset sensor features

Symbol	Description	Unit
T2	Total temperature at fan inlet	$^\circ$R
T24	Total temperature at Low Pressure Compressor outlet	$^\circ$R
T30	Total temperature at High Pressure Compressor outlet	$^\circ$R
T50	Total temperature at Low Pressure Turbine outlet	$^\circ$R
P2	Pressure at fan inlet	psia
P15	Total pressure in bypass-duct	psia
P30	Total pressure at HPC outlet	psia
Nf	Physical fan speed	rpm
Nc	Physical core speed	rpm
epr	Engine pressure ratio (P50/P2)	—
Ps30	Static pressure at HPC	psia
phi	Ratio of fuel flow to Ps30	pps/psi
NRf	Corrected fan speed	rpm
NRc	Corrected core speed	rpm
BPR	Bypass Ratio	—
farB	Burner fuel-air ratio	—
htBleed	Bleed Enthalpy	—
Nf$_{dmd}$	Demanded fan speed	rpm
PCNfR$_{dmd}$	Demanded corrected fan speed	rpm
W31	High Pressure Turbine coolant bleed	lbm/s
W32	Low Pressure Turbine coolant bleed	lbm/s

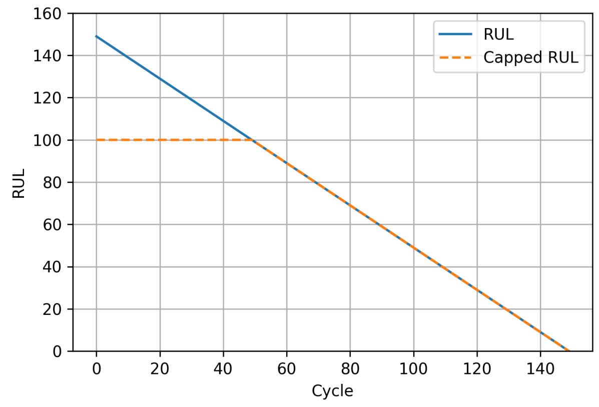

Fig. 2: Maximum RUL of gas turbine engine are capped to 100 cycle to distinguish the healthy state and degradation state during preprocessing stage.

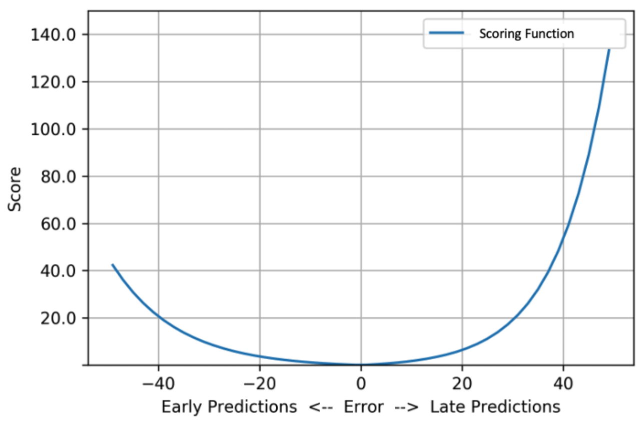

Fig. 3: Score value as the error increases. The score is calculated using the scoring function (Equation (14)), where late predictions (positive errors) receive higher penalisation.

is capped at 100 and remained constant until degradation has occur as shown in Fig 2. This allows the deep learning models to differentiate between the healthy state ($RUL = 100$) and unhealthy state ($RUL < 100$). Even though degradation can happen randomly, the early stages of engine cycle are assumed to be usable and functional. The labels are the RUL cycle for each instance of the data.

B. *Deep Learning Architectures*

We employ the following deep learning model architectures to test the different loss functions in Section II-A, (1) Bi-LSTM, (2) DNN, and (3) CNN1D. Their hyperparameters are listed in Table II. [1]

An L2 regulariser is added to the dense layers of all models shown in Table II to reduce overfitting. The L2 regulariser prevents overfitting by limiting the complexity of the network through the penalisation of larger weights, thus, keeping the weights of the network smaller. Similarly, a dropout [25] rate of 0.2 is also added to all dense layers in models tested to mitigate overfitting. Dropout is a technique for regularising the network by randomly setting the output of units to zero.

C. *Loss Functions*

For the choices of loss functions, we implement MSE and MAE for symmetric loss. As for asymmetric loss, we employ MLSE-MSE, LIN-MSE, LIN-LIN, and QUAD-QUAD as described in Section II-A. LIN-LIN, LIN-MSE, and QUAD-QUAD have weight parameters that we can control as shown in (5), (6), and (8) respectively. Therefore, we uses $a = [0.1, 0.5, 1.0]$ for LIN-MSE, $(a, b) = [(1.0, 2.0), (2.0, 4.0), (2.0, 6.0)]$ for LIN-LIN, and $a = [0.45]$ for QUAD-QUAD to experiment with different level of asymmetricities in loss functions. The a and b values are chosen to approximate properties of MSE and MAE with slight bias as large deviation will cause extreme biasing in the model that leads to bad performance.

[1]The CMAPSS data and code in this paper can be accessed on https://github.com/divishrengasamy/Asymmetric-Loss-Functions-for-Deep-Learning-Early-Predictions-of-Remaining-Useful-Life-in-Aerospace-/

D. *Evaluation*

NASA published a preferred method of performance evaluation for CMAPSS using the idea of asymmetric scoring, named scoring function. The scoring function produces a score that evaluate the RUL predictions. As mentioned previously, in the context of CBM, it is desirable to predict the time of failure early so we can intervene before further damage occurs. Therefore, the scoring is asymmetric around the actual time of failure such that late predictions are more heavily penalised than early predictions. The scoring function is calculated as follows:

$$\text{Score} = \begin{cases} \sum_{i=1}^{n} e^{\frac{-d_i}{10}} - 1 & \text{if } d_i \leq 0 \\ \sum_{i=1}^{n} e^{\frac{d_i}{13}} - 1 & \text{otherwise} \end{cases} \tag{14}$$

Where $d_i = RUL_{predicted} - RUL_{actual}$, i is the current cycle, and n represents the maximum cycle. Fig 3 shows the asymmetric property of the scoring function in (14) where lower scores are given to early predictions. Therefore, the objective is to reach predictions that minimise the score.

Finally, each combination of deep learning models and loss functions are repeated 3 times to obtain the mean and 95% confidence intervals.

IV. Results

Table III shows the comparison of RUL score between Bi-LSTM, DNN, and CNN1D using MSE, MAE, MLSE-MSE, LIN-MSE, LIN-LIN, and QUAD-QUAD loss functions. The results shows that the best overall score of 1027.1 and 1725.0 for Bi-LSTM and DNN is using asymmetrical loss function LIN-LIN with parameters $(a = 1.0, b = 2.0)$ while CNN1D

TABLE II: Hyperparameters of all models used to test the asymmetric loss functions

Deep Learning Architecture	Hyperparameters
Bi-LSTM	Number of layers: 3
	LSTM Layer 1 units: 50
	LSTM Layer 2 units: 25
	Dense Layer 1 units: 50
	Activation function: ReLU
	Dropout rate: 0.2
	Kernel Regularisation: L2 Regularisation
DNN	Number of layers: 6
	Dense Layer 1 units: 100
	Dense Layer 2 units: 250
	Dense Layer 3 units: 100
	Dense Layer 4 units: 250
	Dense Layer 5 units: 12
	Dense Layer 6 units: 6
	Activation function: ReLU
	Dropout rate: 0.2
	Kernel Regularisation: L2 Regularisation
CNN1D	Number of layers: 5
	Conv1D Layer 1: [Filter: 60, Kernel Size: 2]
	Conv1D Layer 2: [Filter: 60, Kernel Size: 2]
	MaxPooling 1D Layer
	Dense Layer 1 units: 100
	Dropout rate: 0.2
	Kernel Regularisation: L2 Regularisation

TABLE III: Final score using Bi-LSTM and DNN with symmetrical loss functions: MSE, and MAE and asymmetrical loss functions: MLSE-MSE, LIN-MSE, LIN-LIN, and QUAD-QUAD. The gray numbers in the square brackets represent the 95% confidence interval and the bolded numbers highlight the best score for each deep learning model.

Loss Functions	Score [95% Confidence Interval]		
	Bi-LSTM	DNN	CNN1D
MSE	1554.4 [1039.5, 2069.2]	4289.9 [3810.8, 4768.9]	1044.7 [690.59, 1398.9]
MAE	1162.4 [721.84, 1603.0]	2001.1 [1263.9, 2738.4]	1365.6 [975.42, 1755.9]
MLSE-MSE	3037.1 [1688.4, 4385.8]	5049.9 [4694.3, 5405.5]	3761.8 [2741.4, 4782.3]
0.1LIN-MSE	2847.4 [1963.1, 3731.7]	4509.0 [4026.9, 4991.1]	3653.1 [2853.2, 4453.0]
0.5LIN-MSE	2965.0 [0, 6776.0]	3227.3 [2820.1, 3634.5]	3821.2 [430.62, 7211.9]
1.0LIN-MSE	2301.5 [1902.3, 2700.6]	3255.6 [2941.7, 3569.6]	2132.3 [600.31, 3664.2]
1.0LIN-2.0LIN	**1027.1** [498.20, 1556.1]	**1725.0** [1401.7, 2048.4]	1357.3 [814.79, 1899.8]
2.0LIN-4.0LIN	1670.5 [1596.6, 1744.3]	2034.9 [1942.3, 2127.5]	1392.1 [81.903, 2702.4]
2.0LIN-6.0LIN	2125.0 [1908.2, 2341.7]	1769.0 [1447.5, 2090.5]	1665.5 [1550.8, 1780.1]
QUAD-QUAD	1647.3 [993.25, 2301.5]	2203.8 [0, 5597.5]	**803.90** [670.78, 937.03]

has the best score of 803.9 using the QUAD-QUAD loss functions. This shows that deep learning models are able to take advantage of the bias created by asymmetric loss functions to achieve early prediction and produce the lowest score. However, we also observe that symmetrical loss achieves a better score than asymmetrical loss in many occasions. For example, Bi-LSTM and DNN with MAE achieve a better score than all asymmetrical loss functions except for LIN-LIN with parameters $(a = 1.0, b = 2.0)$. Similarly for CNN1D, MSE achieves the best score after QUAD-QUAD loss function. This shows that using asymmetrical loss require extensive parameters search and optimisation to obtain an optimal score. We conjecture that high score from using some asymmetrical loss functions is caused by the strong bias it created. We observe that the best performing asymmetrical loss functions are only slightly skewed compare to its symmetrical loss function counterpart.

In addition, we use $a = [0.6, 0.7, 0.8, 0.9]$ for the QUAD-QUAD loss functions and found that it does not converge for Bi-LSTM unless the architecture is altered. Therefore, we omitted the results to keep the architecture unchanged throughout the experimentation process. Furthermore, we experimented with recurrent dropout [26] for Bi-LSTM but found no improvement in the model's performance. The scores within the grey square bracket in Table III represent the lower and upper bound of the 95% confidence interval. The limit of the lower bound is set to zero as a negative is not possible, as shown in Fig 3. For example, we set the confidence lower bound of Bi-LSTM with 0.5LIN-MSE and DNN with QUAD-QUAD loss to zero due to negative lower bound scores.

V. Conclusions and Future Work

In this paper, we demonstrated that RUL predictions of gas turbine engine were improved by using deep learning models with asymmetrical loss functions. Those functions act by creating a bias toward early predictions, with the objectives to optimise CBM and to reasure safety. Asymmetrical loss functions are aimed at improving RUL score, while the deep learning architectures remain unchanged. The deep learning models were tested on symmetrical loss functions, MSE and MAE and on asymmetrical loss functions, MSLE-MSE, LIN-MSE, LIN-LIN, and QUAD-QUAD. Experimental results revealed improvements in the RUL score on CMAPSS dataset using 3 different deep learning architectures, i.e., DNN, CNN1D, and Bi-LSTM when asymmetrical loss function was used. We also showed that not all cases of asymmetrical loss functions perform better than symmetrical loss functions; and it requires optimisation and experimentation to select the most suited asymmetrical loss functions.

For future work, we will improve the asymmetrical loss function by incorporating domain knowledge from principles of physics and engineering related to gas turbine engine. Furthermore, further analysis will be conducted to understand the effects of asymmetrical and symmetrical loss functions on other aerospace sub-system RUL datasets. Finally, we will investigate uncertainty-based asymmetrical loss functions to create deep learning models for implicit uncertainty quantification.

Acknowledgements

This work is funded by the INNOVATIVE doctoral programme. The INNOVATIVE programme is partially funded by the Marie Curie Initial Training Networks (ITN) action (project number 665468) and partially by the Institute for Aerospace Technology (IAT) at the University of Nottingham.

References

[1] D. Rengasamy, H. P. Morvan, and G. P. Figueredo, “Deep learning approaches to aircraft maintenance, repair and overhaul: a review,” in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 150–156.

[2] D. Rengasamy, M. Jafari, B. Rothwell, X. Chen, and G. P. Figueredo, “Deep learning with dynamically weighted loss function for sensor-based prognostics and health management,” *Sensors*, vol. 20, no. 3, p. 723, 2020.

[3] T. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, “Focal loss for dense object detection,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017, pp. 2999–3007.

[4] B. Yang, W. Luo, and R. Urtasun, “Pixor: Real-time 3d object detection from point clouds,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[5] N. Abraham and N. M. Khan, “A novel focal tversky loss function with improved attention u-net for lesion segmentation,” in *2019 IEEE 16th International Symposium on Biomedical Imaging (ISBI 2019)*, April 2019, pp. 683–687.

[6] Y. Ulu, “Optimal prediction under linlin loss: Empirical evidence,” *International Journal of Forecasting*, vol. 23, no. 4, pp. 707–715, 2007.

[7] P. H. Franses, R. Legerstee, and R. Paap, “Estimating loss functions of experts,” 2011.

[8] C. Pierdzioch, J.-C. Rülke, and G. Stadtmann, “Oil price forecasting under asymmetric loss,” *Applied Economics*, vol. 45, no. 17, pp. 2371–2379, 2013.

[9] A. Saxena *et al.*, “Damage propagation modeling for aircraft engine run-to-failure simulation,” in *I Conf on Prog and Health Man*, Oct 2008.

[10] K. Goebel, H. Qiu, N. Eklund, and W. Yan, “Modeling propagation of gas path damage,” in *2007 IEEE Aerospace Conference*, March 2007, pp. 1–8.

[11] P. Tamilselvan and P. Wang, “Failure diagnosis using deep belief learning based health state classification,” *Rel Eng & Sys Safety*, vol. 115, 2013.

[12] G. E. Hinton, S. Osindero, and Y. W. Teh, “A fast learning algorithm for deep belief nets,” *Neural Comp*, vol. 18, no. 7, pp. 1527–1554, 2006.

[13] T. Kohonen, “The self-organizing map,” *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep 1990.

[14] C. Zhang *et al.*, “Multiobjective deep belief networks ensemble for remaining useful life estimation in prognostics,” *IEEE Tran on Neural Nets and Learning Systems*, vol. PP, pp. 1–13, 07 2016.

[15] M. Yuan *et al.*, “Fault diagnosis and remaining useful life estimation of aero engine using LSTM neural network,” *IEEE Int Conf on Aircraft Utility Systems*, pp. 135–140, 10 2016.

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

[17] A. L. Ellefsen, E. Bjørlykhaug, V. Æsøy, S. Ushakov, and H. Zhang, “Remaining useful life predictions for turbofan engine degradation using semi-supervised deep architecture,” *Reliability Engineering System Safety*, vol. 183, pp. 240 – 251, 2019.

[18] J. Wang, G. Wen, S. Yang, and Y. Liu, “Remaining useful life estimation in prognostics using deep bidirectional lstm neural network,” in *2018 Prognostics and System Health Management Conference (PHM-Chongqing)*, Oct 2018, pp. 1037–1042.

[19] G. S. Babu, P. Zhao, and X. Li, “Deep convolutional neural network based regression approach for estimation of remaining useful life,” in *DASFAA*, 2016.

[20] X. Li *et al.*, “Remaining useful life estimation in prognostics using deep convolution neural networks,” *Rel Eng & Sys Safety*, vol. 172, 12 2017.

[21] S. Ruder, “An overview of gradient descent optimization algorithms,” *CoRR*, vol. abs/1609.04747, 2016.

[22] E. Ramasso, “Investigating computational geometry for failure prognostics,” *International Journal of Prognostics and Health Management (2153-2648)*, vol. 005, pp. 1–18, 07 2014.

[23] Y. *et al.* Peng, “A modified echo state network based remaining useful life estimation approach,” in *IEEE C. on Prog and Health Man*, 2012.

[24] L. Pin, C.-K. Goh, and K. Chen Tan, “A time window neural network based framework for remaining useful life estimation,” in *Int Joint Conf on Neural Networks*, 07 2016, pp. 1746–1753.

[25] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[26] Y. Gal and Z. Ghahramani, “A theoretically grounded application of dropout in recurrent neural networks,” pp. 1019–1027, 2016.