Uncertainty Quantification using Deep Ensembles for Safety-Critical Predictive Models

Oishi Deb
Department of Engineering Science
University of Oxford, UK
oishideb@robots.ox.ac.uk

Emmanouil Benetos
School of Electronic Engineering
and Computer Science
Queen Mary University of London, UK
emmanouil.benetos@qmul.ac.uk

Philip Torr
Department of Engineering Science
University of Oxford, UK
philip.torr@eng.ox.ac.uk

Abstract

This paper introduces a novel approach for uncertainty quantification in safety-critical predictive models by using a deep ensemble model, hence addressing a critical problem in predictive maintenance tasks. It builds a regression model to predict the Remaining Useful Life (RUL) of aircraft engines, utilizing the well-known run-to-failure turbo engine degradation dataset. Addressing the overlooked yet crucial aspect of uncertainty estimation in previous research, this paper revamps the LSTM architecture to facilitate uncertainty estimates, employing Negative Log Likelihood (NLL) as the training criterion. Through a series of experiments, the model demonstrated self-awareness of its uncertainty levels, correlating high confidence with low prediction errors and vice versa. This initiative not only enhances predictive maintenance strategies but also significantly improves the safety and reliability of aviation assets by offering a more nuanced understanding of predictive uncertainties. To the best of our knowledge, this is pioneering work in this application domain.

1 Introduction

Predictive maintenance, harnessing machine learning for timely upkeep, has become crucial in engineering and manufacturing, notably reducing costs and enhancing revenue through early aircraft engine degradation detection and accurate Remaining Useful Life (RUL) predictions [1].

While past RUL prediction research for turbo engines, such as [2] and [3], has primarily utilized logistic regression and standard Artificial Neural Networks (ANN), a significant gap remains in exploring predictive model uncertainty, especially in safety-critical systems [4, 5, 6, 7]. Deep learning models, despite their varied successes, often display overconfidence in predictions, emphasizing the need for accurate uncertainty estimation [8, 9].

This paper addresses this gap, developing a regression model using the "Deep Ensemble" [8] technique to predict turbo engine RUL, leveraging the widely recognized turbo engine degradation dataset [10]. Metrics like RMSE, R^2, and Negative Log Likelihood are employed, with the paper aiming to enhance the Deep Ensemble technique by estimating mixture, Epistemic, and Aleatoric uncertainties, offering a comprehensive understanding of predictive uncertainty.

37th Conference on Neural Information Processing Systems (NeurIPS 2023) Workshop on SSL: Theory and Practices 2023.

2 Literature Review

2.1 Uncertainty Quantification methods

Uncertainty pertains to a state of ambiguity, and this is a phenomenon that machine learning models occasionally grapple with in relation to their predictions. Statistical methodologies, such as Confidence Intervals and Monte Carlo Simulations can be used for gauging this uncertainty.

A Confidence Interval provides a specified range within which the predicted value is likely to fall, as elucidated by Cosma Shalizi [11]. In practice, if a Neural Network model undergoes multiple iterations, and the standard deviation of the output values is calculated, it becomes feasible to deduce the confidence interval in an offline manner. Nonetheless, given that Neural Network models may comprise millions of parameters, executing them repeatedly incurs substantial computational costs. To mitigate this, the strategy of ensembling multiple Neural Networks, also known as Deep Ensemble, was introduced during the training phase. A comprehensive discussion on Deep Ensemble is provided in a subsequent section.

It is crucial to highlight that calculating the confidence interval offline to ascertain uncertainty is fundamentally different from deriving uncertainty directly from the model. When a deep learning model proactively provides an estimate of uncertainty alongside its prediction, it demonstrates an intrinsic awareness of its confidence level. This self-awareness is of paramount importance, especially in safety-critical applications such as autonomous systems, where understanding and acknowledging the model's limitations and uncertainties is essential.

As previously highlighted, conventional Neural Network architectures do not inherently provide an estimate of uncertainty in their predictions. This necessitates a modification and enhancement of existing Neural Network structures to integrate statistical methodologies such as Confidence Intervals and Monte Carlo Simulations, facilitating the extraction of uncertainty from the model.

In the realm of deep learning, the most prominent methods for uncertainty estimation encompass Bayesian and non-Bayesian techniques. Monte Carlo Dropout, a Bayesian technique, stands out as a significant method and is thoroughly discussed in the work of Gal et al. [12]. On the other hand, Deep Ensembles, a non-Bayesian approach detailed by Balaji [8], is well-known for its computational efficiency. While there are various other methods available for uncertainty estimation mainly from Bayesian point of view, as outlined in Table 1, the paper at hand has opted to employ Deep Ensembles for being the most state-of-the-art model due to its computational efficiency.

Monte Carlo Dropout: Probabilistic machine learning encompasses Frequentist and Bayesian strands, with the Monte Carlo dropout technique aligning with the latter [13]. While dropout was initially explored by [14] and [15] from a Bayesian perspective, [12] enhanced it, demonstrating its use as a Bayesian Approximation to integrate over Neural Network model weights. Unlike traditional Neural Networks, Bayesian Neural Networks initialize weights with prior probability distributions, commonly Gaussian, and compute posterior distributions instead of point estimates.

Introduced by [16] to mitigate overfitting, dropout was traditionally used during training. However, [12] applied it during both training and testing, yielding non-identical outputs during test time with different dropout values, analogous to Monte Carlo (MC) sampling [12, 17]. This variance in output samples provides uncertainty estimates. Although MC Dropout is a recognized method for uncertainty quantification from a Bayesian standpoint, the subsequent introduction of a non-Bayesian method, 'Deep Ensemble,' by DeepMind [8] offered an alternative computationally efficient approach, detailed in the following section.

The limited industrial deployment of the MC dropout method is attributed to the computational expense and scalability challenges of Bayesian Inference [8]. In a more recent industrial research study conducted in 2021, MC Dropout was experimented with for Smart Grid Design application [18], however, it has not been deployed fully yet.

Deep Ensembles: Deep Ensembles, a probabilistic model, which is also known as ensembles of neural networks are not as computationally heavy as Bayesian models as they do not use Bayesian Inference. The concept of the ensemble has been widely used in the regularisation of Neural Network [19], however, this method also proves useful in uncertainty estimates as mentioned in [8].

In probabilistic regression models, the output is considered to be a Gaussian or Normal distribution with parameter mean and variance [20], hence this Distribution is used in calculating the Negative

Log Likelihood (NLL). This NLL is then used as a cost function for the Deep Ensembles model. The Deep Ensembles method implements the concept from [21] in which it modifies the neural network to output the predictive mean and the standard deviation in the output layers, as opposed to classical neural networks for regression tasks where it only outputs a point estimate of the prediction. Confidence intervals can be calculated from the standard deviation to interpret the uncertainty in the prediction model.

Below is the equation of the cost function - Negative Log Likelihood (NLL). The full derivation of the equation is shown in the Appendix.

$$NLL = 0.5 \log(2\pi\sigma^2) + 0.5\sigma^{-2} \sum_{n=1}^{N} (x_n - \mu)^2 \tag{1}$$

where - σ^2: is the variance of the Gaussian distribution; σ: is the standard deviation of the Gaussian distribution; μ: is the mean of the Gaussian distribution; x_n: is the individual data-points from 1 to N. N: is the total number of data points.

The NLL equation in the Deep Ensemble paper [8] removes the 2π in the first term and adds an additional constant term. Since the aim is to minimise the cost function and as 2π is a constant hence removing this term will not make any difference in the minimisation. An additional constant term was added in the equation as mentioned in [8], which is usually done when working with logarithms to avoid underflow and overflow errors. Negative Log Likelihood is used because it is a proper scoring rule that is widely used for evaluating predictive uncertainty [8]. The proper scoring rule is a function of the probability prediction and the output variable, and this function is minimum when the prediction is well-calibrated [11].

In other words, NLL is a way of measuring the error of the predictions. This works for models that predict sufficient statistics over some distribution (for example mean and variance for a Gaussian in this case) and then NLL can be calculated for the ground truth under the predicted distribution. The reason this metric is used in models that use uncertainty is because the model predicts the variance/standard deviation and uses it for the NLL, so the prediction of the uncertainty will affect this metric. However, variance/standard deviation is merged with the prediction of the mean, so just from the NLL it is not possible to make any claims about the uncertainty itself.

3 Procedure

3.1 Dataset

An open-source dataset [10] has been used which is available from the NASA repository. The dataset is run-to-failure turbo engine data created using the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) [22]. C-MAPSS was developed by NASA which simulates the real-life controls system for 90,000 pound thrust dual spool and high-bypass ratio turbo engines. This simulation environment was developed to facilitate research in control systems, Engine Health Monitoring (EHM), predictive maintenance etc. The dataset contains data from 4 different fleets of engines and each of the fleet data and information on the number of engine units are on Table 2. Table 3 shows the list of sensor names whose values are recorded for each engine unit in the dataset for each cycle until the engine fails. An engine cycle includes three events: engine start, take-off and landing, and engine shutdown. Take-off and landing are classed as a single event because take-off is always followed by a landing.

An explanation of the dataset extract is in Table 4, the engine unit 1 fails at 149 engine cycles whereas engine unit 2 fails at 269 engine cycles respectively.

3.2 Deep Ensemble Model Building

An LSTM (Long Short-Term Memory) network architecture model is constructed with an input size of sequence length and number of features. According to the dataset [10] there are 25 input features - which are cycle number, 3 operating settings, and 21 sensor values. The reason LSTM networks are used is due to their suitability for time series data as they can capture long-term dependencies and

patterns over time. This makes them ideal for predicting RUL, where historical sensor data and past performance trends are crucial for accurate forecasting.

The dataset, divided into training and testing files, allocated ten percent of the training data for validation. Table 6 displays the finalised parameter and hyperparameter values, determined after numerous experiments detailed in Table 7. The hyperparameters are tuned/optimized using random search. Dropout, a regularization technique that probabilistically omits inputs during network training to mitigate overfitting, is applied after each LSTM layer. It is crucial to note that the dropout used for regularization here differs from Monte Carlo Dropout [9], utilised for uncertainty quantification. The model's architecture is given in Table 5. The model is developed using Python's Keras [23] (version 2.3.0) with TensorFlow [24] (version 1.14.0) as a backend, the model employed "Keras Uncertainty" open-source resources [25] for the Deep Ensemble model creation. The raw dataset lacked RUL labels, prompting the use of labels from [26].

The Deep Ensemble model was implemented using a custom Class, which takes in two arguments: first a function that creates a Neural Network model and second the number of neural network models the user wants in the ensemble. The Neural Network model provides two outputs: prediction value and the standard deviation. Since the standard deviation output of each member in the deep ensemble does not have direct supervision, unlike the prediction value (i.e. the prediction value is supervised by the target y), the standard deviation is indirectly supervised by the loss function NLL. This has been implemented by passing the standard deviation output to the loss, in which case, the training model does not output the standard deviation directly, but it is included in the loss so it influences the loss correctly.

4 Evaluation and Discussion

4.1 Evaluating Model with Uncertainty Quantification

The Deep Ensemble model, tested on four-engine fleets, exhibited the highest error rate with the FD004 dataset, prompting numerous experiments detailed in Table 7 to enhance performance. The row highlighted in blue indicates optimal performance in both prediction and uncertainty estimates. While augmenting the number of Neural Networks (NN) in the ensemble improves performance, an ensemble exceeding three NN elevates the error rate. Upon identifying the top-performing model using the FD004 dataset, it was applied to the other three fleet datasets. Table 8 reveals the best RMSE value as 30.65 for the "FD001" dataset, with a Mixture standard deviation (uncertainty) of 42.46, placing the prediction value within a ± 42.46 range of the mean prediction for this dataset. An R^2 of 0.53 for the FD001 indicates a commendable fit.

Figure 1 displays the prediction graph, with green denoting ground truth and blue representing prediction, figure 2 illustrates model prediction uncertainty with black and red trends. Ideally, the model should exhibit minimal uncertainty to maximize prediction certainty. However, establishing an uncertainty value threshold or conducting further comparisons necessitates the availability of additional baseline results for analysis.

NLL for evaluating uncertainty: The "FD001" dataset yielded a minimum NLL value of 3.95, which, while not a direct measure of uncertainty, facilitates the evaluation of the model's standard deviation during loss function minimization, thereby considering both prediction and its associated uncertainty. In the context of the four datasets, a lower NLL not only signifies more accurate predictions but also reliable uncertainty values. Mixture uncertainty suffices for assessing a predictive model's suitability, as demonstrated by [8]. However, incorporating Epistemic and Aleatoric uncertainties provides nuanced insights from individual Neural Network (NN) models, enhancing decision-making.

For instance, the NN model reflects its confidence in its predictions. In the "FD002" dataset, a high RMSE of 39.69 corresponds with a high mixture standard deviation (uncertainty) of 63.62, indicating the model's awareness of its imperfect prediction. Low uncertainty could be perilous, signaling overconfidence in the model. In the "FD002" dataset, despite having the most units (260 engines) and the lowest Epistemic standard deviation (33.41) among the four datasets, implying a desirable decrease in Epistemic standard deviation with increasing dataset size, the mixture uncertainty remains elevated due to a high error rate. Regarding the "FD004" dataset, it presents an RMSE of 41.91 and a substantial mixture standard deviation of 57.97, revealing the model's cognizance of its prediction's limited reliability. While inaccurate, the model's lack of confidence is non-fatal. The absence of an

uncertainty value could be hazardous in safety-critical applications, as it leaves users blind to the model's confidence level.

Note that determining the threshold for uncertainty in maintenance decisions is context-dependent and involves risk assessment by the organisations. This paper does not detail specific thresholds but highlights that lower NLL values indicate more reliable uncertainty estimates. Maintenance teams might set thresholds based on acceptable risk levels and the criticality of engine components.

5 Conclusion and Further Work

Previous studies on predicting the Remaining Useful Life (RUL) of turbo engines did not address the uncertainty of the predictive model in safety-critical applications. Given the state-of-the-art deep ensemble model, the paper proposes a novel approach to address the gap in the literature by implementing the Deep Ensemble method to quantify model uncertainty. Significantly, it marks the first integration of LSTM architecture with Deep Ensemble for RUL predictions and uncertainty assessments in aircraft engines, utilizing NASA's engine degradation dataset.

While the Deep Ensemble method showcased computational efficiency and delivered acceptable results on the FD001 and FD003 datasets, with Root Mean Square Error (RMSE) values of approximately 30 and 33, it encountered challenges with the FD002 and FD004 datasets. These datasets presented larger sizes and higher levels of data noise, resulting in elevated error rates.

This paper lays the groundwork for future research, underscoring the need to augment existing predictive models with methods for providing uncertainty estimates. Prospective directions include efforts to reduce predictive uncertainty and leverage uncertainty estimates to mitigate generalization error. Additionally, uncertainty estimates could be instrumental for Out-Of-Distribution (OOD) detection [27, 28], signaling increased uncertainty for predictions related to values outside the training set's scope.

Investigations into advanced sequence models, such as Encoder-Decoder architectures [29, 30] with attention mechanisms [31], could also prove beneficial. These models have demonstrated their effectiveness in language translation and are applicable to time-series data in safety-critical autonomous systems. The evolution of dropout and model ensemble from regularization techniques to tools for uncertainty estimation opens the door for experimentation with other regularization strategies, such as parameter sharing [19], in conjunction with existing uncertainty estimation methodologies.

Further inquiries could also consider the amalgamation of deep ensemble with test-time dropout (i.e., MC dropout) for enhanced uncertainty acquisition. Despite the computational demands, this combination could potentially surpass the performance of Deep Ensemble, MC dropout, and classical methods. Viewing uncertainty estimates research through a regularization lens is promising, aiming to diminish generalization error, thereby bolstering model prediction precision and reliability.

References

[1] Y. Ran, X. Zhou, P. Lin, Y. Wen, and R. Deng, "A survey of predictive maintenance: Systems, purposes and approaches," 2019.

[2] J. Yu, "Aircraft engine health prognostics based on logistic regression with penalization regularization and state-space-based degradation framework," *Elsevier Aerospace Science and technology*, pp. 345–361, 2017.

[3] A. K. Jain, P. Kundu, and B. K. Lad, "Prediction of remaining useful life of an aircraft engine under unknown initial wear," in *5th International 26th All India Manufacturing Technology, Design and Research Conference, Guwahati, Assam, India*, 2014.

[4] U. Amin and K. D. Kumar, "Remaining useful life prediction of aircraft engines using hybrid model based on artificial intelligence techniques," in *2021 IEEE International Conference on Prognostics and Health Management (ICPHM)*, 2021, pp. 1–10.

[5] Y. Wu, M. Yuan, S. Dong, L. Lin, and Y. Liu, "Remaining useful life estimation of engineered systems using vanilla lstm neural networks," *Neurocomputing*, vol. 275, pp. 167–179, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231217309505

[6] 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)*, 2017, pp. 88–95.

[7] R. McAllister, Y. Gal, A. Kendall, M. Van der Wilk, R. Shah, A.and Cipolla, and A. Weller, "Concrete problems for autonomous vehicle safety: Advantages of bayesian deep learning.." in *In Proceedings of the International Joint Conferences on Artificial Intelligence, New York, NY, USA,*, 2016.

[8] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *NIPS 2017*, 2017.

[9] Y. Gal, *Uncertainty in Deep Learning*. University of Cambridge PhD Thesis, 2016.

[10] A. Saxena and K. Goebel, "Turbofan engine degradation simulation data set, nasa ames prognostics data repository," NASA Ames Research Center, Moffett Field, CA, 2008. [Online]. Available: https://ti.arc.nasa.gov/tech/dash/groups/pcoe/prognostic-data-repository/, 2008.

[11] C. R. Shalizi, *Advanced Data Analysis from an Elementary Point of View*. Cambridge University Press. [Online]. Available: https://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/ADAfaEPoV.pdf

[12] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning." in *International Conference on Machine Learning (ICML)*, 2016.

[13] Z. Ghahramani, "Probabilistic machine learning and artificial intelligence," 2015.

[14] S. Wang and C. Manning, "Fast dropout training." in *International Conference on Machine Learning (ICML)*, 2013.

[15] S.-i. Maeda, "A bayesian encourages dropout." in *International Conference on Learning Representation (ICLR)*, 2015.

[16] 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, no. 56, pp. 1929–1958, 2014. [Online]. Available: http://jmlr.org/papers/v15/srivastava14a.html

[17] Y. Gal, "A theoretically grounded application of dropout in recurrent neural networks," *arXiv:1512.05287*, 2015.

[18] M. Selim, R. Zhou, W. Feng, and P. Quinsey, "Estimating energy forecasting uncertainty for reliable ai autonomous smart grid design," *Energies*, vol. 14, no. 1, 2021. [Online]. Available: https://www.mdpi.com/1996-1073/14/1/247

[19] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[20] K. P. Murphy, *Probabilistic Machine Learning: An introduction*. MIT Press, 2021. [Online]. Available: probml.ai

[21] D. Nix and A. Weigend, "Estimating the mean and variance of the target probability distribution," in *Proceedings of 1994 IEEE International Conference on Neural Networks (ICNN'94)*, vol. 1, 1994, pp. 55–60 vol.1.

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

[23] F. Chollet, *Deep learning with python*. Manning Publications, New York, 2017.

[24] GoogleBrain, "Tensorflow documentation." [Online]. Available: https://www.tensorflow.org/versions

[25] M. Valdenegro, "Keras uncertainty models," 2021. [Online]. Available: github.com/mvaldenegro/keras\protect\discretionary{\char\hyphenchar\font}{}{}uncertainty/tree/master/keras_uncertainty/models

[26] Microsoft-Azure, "Deep learning for predictive maintenance," 2017. [Online]. Available: https://github.com/Azure/lstms_for_predictive_maintenance/

[27] A. Sedlmeier, T. Gabor, T. Phan, L. Belzner, and C. Linnhoff-Popien, "Uncertainty-based out-of-distribution detection in deep reinforcement learning," *CoRR*, vol. abs/1901.02219, 2019. [Online]. Available: http://arxiv.org/abs/1901.02219

[28] A. Schwaiger, P. Sinhamahapatra, J. Gansloser, and K. Roscher, "Is uncertainty quantification in deep learning sufficient for out-of-distribution detection?" in *AISafety@IJCAI*, 2020.

[29] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014. [Online]. Available: http://arxiv.org/abs/1406.1078

[30] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014. [Online]. Available: http://arxiv.org/abs/1409.3215

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017. [Online]. Available: http://arxiv.org/abs/1706.03762

[32] S. Sankararaman and K. Goebel, "Uncertainty quantification in remaining useful life of aerospace components using state space models and inverse form," in *54th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, no. AIAA 2013-1537, April 2013.

[33] C. Blundell, J. Cornebise, K. Kavukcuoglu, and D. Wierstra, "Weight uncertainty in neural network." in *International Conference on Machine Learning, PMLR*, 2015.

[34] J. Hron, A. Matthews, and Z. Ghahramani, "Variational gaussian dropout is not bayesian." in *Second workshop on Bayesian Deep Learning (NIPS 2017)*, 2017.

[35] X. Zhang, C. Guo, R. Yang, and K. Li, "The transient concept of bearings: A novel strategy for rul prediction," *Measurement Science and Technology*, vol. 35, no. 2, p. 025104, 2024, © 2023 IOP Publishing Ltd.

6 APPENDIX

6.1 Derivation

The Derivation of the Negative Log Likelihood is as follows:

$$\begin{aligned} NLL &= -\log \prod_{n=1}^{N} P(x_n|\theta) \\ P(x_n|\theta) &\text{ is the likelihood} \\ \theta &\text{ is the parameter of the likelihood function} \end{aligned} \tag{2}$$

We substitute the likelihood with the Probability Density Function (PDF) of the Gaussian Distribution because we are dealing with a regression problem.

$$NLL = -\log \prod_{n=1}^{N} \left(\sigma^{-1} (2\pi)^{-\frac{1}{2}} \exp\left(-\frac{(x_n - \mu)^2}{2\sigma^2} \right) \right) \tag{3}$$

Applying the logarithmic addition rule and simplifying it further:

$$NLL = -\sum_{n=1}^{N} \log\left(\sigma^{-1}(2\pi)^{-\frac{1}{2}} \exp\left(-\frac{(x_n-\mu)^2}{2\sigma^2}\right)\right) \tag{4}$$

Since $log(e^x) = x$ and $log(1/a) = -log(a)$, therefore we get as follows:

$$= 0.5\log(2\pi\sigma^2) + 0.5\sigma^{-2}\sum_{n=1}^{N}(x_n-\mu)^2 \tag{5}$$

[Note, the Deep Ensemble paper [8] doesn't provide the derivation, we have done the derivation for the reader's ease of understanding.]

6.2 Tables and Figures

Table 1: List of works for uncertainty quantification specifically in Deep Learning models

No.	Author,Year	Method
1	Sankararaman et. al [32],2013	Inverse FORM
1	Charles et. al [33],2015	Weight uncertainty in neural network
2	Yarin et al [12], 2016	Monte Carlo Dropout
3	Jiri et al [34], 2017	Variational Gaussian Dropout
4	Balaji et al [8], 2017	Deep Ensembles
5	Zhang et al [35], 2022	K-means-transformer network

Table 2: Number of Engine Units in each fleet

Fleet Number	File Name	Number of Engine Units
1	FD001	100
2	FD002	260
3	FD003	100
4	FD004	249

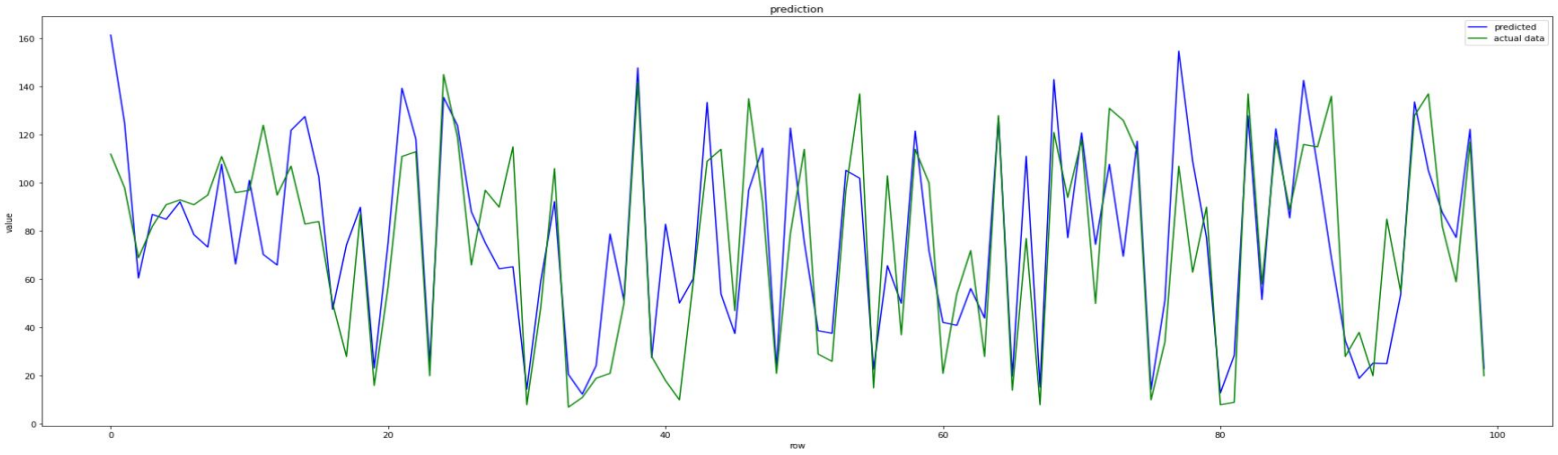

Figure 1: Deep Ensemble LSTM Model for "FD001" dataset showing the actual (green line) and predicted curve (blue line).

Table 3: List of Sensor Measurements [22]

Sensor No.	Description
1	Total temperature at fan inlet (°R)
2	Total temperature at LPC outlet (°R)
3	Total temperature at HPC outlet (°R)
4	Total temperature at LPT outlet (°R)
5	Pressure at fan inlet (psia)
6	Total pressure in bypass-duct (psia)
7	Total pressure at HPC outlet(psia)
8	Physical fan speed (rpm)
9	Physical core speed (rpm)
10	Engine pressure ratio (P50/P2)
11	Static pressure at HPC outlet (psia)
12	Ratio of fuel flow to Ps30 (pps/psi)
13	Corrected fan speed (rpm)
14	Corrected core speed (rpm)
15	Bypass Ratio
16	Burner fuel-air ratio
17	Bleed Enthalpy
18	Demanded fan speed (rpm)
19	Demanded corrected fan speed (rpm)
20	HPT coolant bleed (lbm/s)
21	LPT coolant bleed (lbm/s)

Table 4: A brief extract from the dataset file "FD002"

Unit	Cycle	Operational Setting 1	Sensor 1
1	1	34.9983	449.44
1	2	41.9982	445.00
1	149	42.0017	445.00
2	1	0.0025	518.67
2	2	35.0058	449.44
2	269	42.0047	445.00
260	1	34.9989	449.44
260	2	19.9985	491.19
260	316	35.0036	449.44

Table 5: Model Architecture

Layer Type	Size	Activation	Dropout
Input Size	(Sequence length, Number of Features) (30, 25)	-	-
LSTM	Hidden layer: (100 Neurons)	ReLU	0.2
Dense	Hidden layer: (30 Neurons)	ReLU	0.2
Dense	output layer - 1	Linear	-
Dense	output layer - 1	Softplus	-

Table 6: Parameters/Hyperparameters Settings for Deep Ensemble Model

No.	Parameter/Hyperparameter	Values
1.	Number of epochs	10
2.	Batch size	150
3.	Sequence Length	30
4.	Number of Hidden layers in NN	2
5.	Number of nodes in Layer 1 (LSTM)	100
6.	Number of nodes in Layer 2 (Dense)	30
7.	Optimizer	Adam
8.	Learning rate	0.01

Table 7: Result from Running Deep Ensembles Model run on "FD004" (Sequence size as 30 has been used). A represents 'Aleatoric', E represents 'Epistemic' and M represents 'Mixture'.

No.	Ensemble	Epochs	Nodes L2	RMSE	R^2	NLL	A std	E std	M std
1	2	10	30	59.27	-0.24	5.11	154.32	53.99	162.05
2	**3**	**10**	**30**	**41.91**	**0.38**	**4.30**	**55.4**	**50.17**	**57.97**
3	3	20	30	42.90	0.35	4.43	69.25	37.02	73.61
4	4	10	30	44.16	0.31	5.62	271.4	47.8	273.7
5	5	10	30	47.21	0.21	4.52	76.56	40.85	84.11

Table 8: Result from Running Deep Ensembles Model (Sequence size 30 has been used). In the below table A represents 'Aleatoric', E represents 'Epistemic' and M represents 'Mixture' uncertainty.

File	No. of Units	Ensemble	Epochs	Nodes L2	RMSE	R^2	NLL	A	E	M
FD001	100	3	10	30	30.65	0.53	3.95	41.45	35.16	42.46
FD002	260	3	10	30	39.69	0.40	4.31	59.83	33.41	63.62
FD003	100	3	10	30	33.21	0.50	4.28	66.0	51.77	74.54
FD004	249	3	10	30	41.91	0.31	4.30	55.4	50.17	57.97

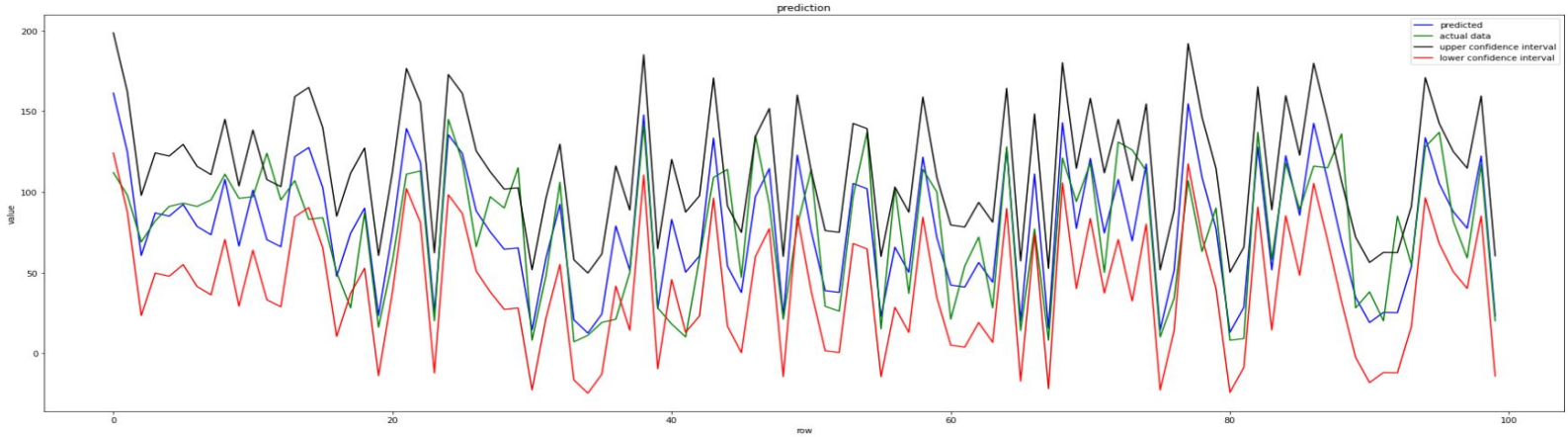

Figure 2: Mixture Uncertainty for "FD001" data-set using Deep Ensemble LSTM Model showing the actual (green line) and predicted curve (blue line) along with upper (black line) and lower (red line) confidence bound. The confidence interval represents 2 sigma.