Empirical Likelihood Stacking for Remaining Useful Life Estimation

Empirical Likelihood Stacking for Remaining Useful Life Estimation

Jonas Karlsson[1(✉)], Alexander Karlsson[1], Sunith Bandaru[2], and Ebru Turanoglu Bekar[3]

[1] School of Informatics, University of Skövde, Skovde, Sweden
{jonas.karlsson,alexander.karlsson}@his.se
[2] School of Engineering Science, Skövde University, Skovde, Sweden
sunith.bandaru@his.se
[3] Department of Industrial and Materials Science, Chalmers University of Technology, Gothenburg, Sweden
ebrut@chalmers.se

Abstract. Predictive maintenance (PdM) plays an important role in the transition toward Industry 4.0, offering the potential to increase operational efficiency and reduce unexpected downtime. A central application of PdM involves the estimation of a system component's remaining useful life (RUL) to enable informed maintenance decisions. Deep learning (DL) models have demonstrated strong performance in RUL prediction tasks. However, these models often lack the capability to reliably quantify predictive uncertainty, which is essential for risk-aware decision-making in industrial applications. Bayesian approximation models, like Stochastic Weight Averaging Bayesian and Variational Inference, can yield a distribution of predictions while avoiding the prohibitively costly process of true Bayesian inference. This distribution can be used to reason about the uncertainty of the model, but often shows the model being overconfident. We propose empirical likelihood stacking, a method for improving uncertainty quantification for DL models. By applying stacking and deep ensembles to estimate the RUL of turbofan engines, we show the trade-offs between these two methods when applied to DL models. We find that the stacked model has a better calibrated uncertainty at a small cost to point prediction accuracy, when compared to the ensemble.

Keywords: Artificial Intelligence · Stacking · Deep Learning · Uncertainty · Remaining Useful Life

1 Introduction

Predictive Maintenance (PdM) has been identified as a key aspect of *Industry 4.0* [27]. Due to the increasing availability of sensors and real-time monitoring, the health of a machine can be tracked over time, and components can be changed before a breakdown occurs. These large datasets allow for the application of data-driven methods, such as Deep Learning (DL) models, which have been applied

B. Vantaggi et al. (Eds.): IPMU 2026, CCIS 3019, pp. 303–317, 2026.
https://doi.org/10.1007/978-3-032-28994-0_22

to maintenance prognostics with great effect [21]. A popular method of estimating when a machine (or a component within a machine) will suffer a fault is by predicting its remaining useful life (RUL) [22]. The RUL value describes how long the component in question will operate until it requires repair or replacement. Thus, accurately predicting a machine's RUL and performing preventive maintenance before a fault occurs can prevent critical stops in production.

It is expected to have some deviation between the true RUL value and that predicted by the model. This means that the RUL prediction carries some inherent uncertainty. Quantifying this uncertainty is essential when making a decision based on the prediction of a PdM system [4]. For example, a maintenance engineer might want to take extra precautions when dealing with a high uncertainty prediction for a critical component. In this way, the reliability of a RUL prediction is affected by the quality of its associated uncertainty. This motivates the use of predictions that are represented by a distribution of RUL estimates, rather than a point RUL estimate. By providing these probabilistic RUL prognostics, we can increase the reliability of the PdM system.

Although Deep Neural Networks (DNNs) have exhibited great promise in predicting RUL, they are commonly deterministic, meaning they provide a single output per prediction. Probabilistic DL models, often utilizing Bayesian methods, represent the parameters (weights and biases) of a Neural Network (NN) as random variables. This results in probabilistic DL models that yield a distribution of RUL values per prediction. This predictive distribution can then be used to represent the uncertainty associated with the prediction. Variational Inference (VI) [3] approximates Bayesian inference and is a popular method used within probabilistic DL. Using VI with NNs, the weights and biases of the neural network are represented as probability distributions. Similarly, Stochastic Weight Averaging Gaussian (SWAG) [15] is a baseline probabilistic model for obtaining probabilistic output from a NN. The predictive distribution of a VI or SWAG model provides better uncertainty estimation than a single RUL value. This is due to the models consisting of multiple sets of NN parameters, where each unique set of parameters values yield a different prediction. However, we can improve the prediction further by using multiple models, gathered over separate training runs. This process of aggregating many trained DL models is known as *deep ensembles* [9]. Although using multiple models can improve the total predictive output, there is still a potential gain to be had by adding some logic to the ensembling process. Stacking [26] does this by weighing the output of each model before including it in the final prediction. Since stacking involves additional computations when compared to regular ensembling, we would expect the added complexity to contribute to better performance of the final aggregated model.

In this paper, we propose *empirical likelihood stacking*; a pragmatic stacking approach for probabilistic DL. To obtain the weights of the stacked distribution, our proposal is to use the fitness of each model sample on the validation dataset. To show its effect in practice, an empirical comparison between ensembling and stacking approaches on SWAG and VI was conducted on a well known RUL

prognostic dataset. Although the experiment focuses on measuring uncertainty quantification, we also include point prediction accuracy in our result.

2 Related Work

Various Machine Learning (ML) methods have been applied to the area of RUL prognostics before the current focus on DNNs. Khan and Yairi outline [11] some of these methods in their review. Previous methods include autoencoders, k-means clustering, decision trees and random forests. They also bring up NN architectures, such as the Convolutional Neural Network (CNN) and the Long Short-Term Memory (LSTM). We direct interested readers to that article for additional preliminaries of the application area.

In the article by Staber and Da Viega [23], quality of the uncertainty estimation among multiple Bayesian NNs are evaluated. They find that, among the methods tested, deep ensembles (of single NN parameters) are able to strike a balance between predictive accuracy and prediction coverage. SWAG, on the other hand, was shown to have excellent performance on prediction accuracy, while underestimating the uncertainty of its predictions. Our paper extends this work, by exploring the gains of ensembling models containing multiple sets of network parameters, like SWAG models.

Mitici et al. [16] implemented Monte Carlo Dropout (MC Dropout) with a CNN to achieve uncertainty estimation for RUL predictions on the Commercial Modular Aero-Propulsion System Simulation (C-MAPSS) dataset. They found that MC Dropout performed well with regard to uncertainty calibration as the α-Coverage was close to ideal for all subsets of the data. Xiang et al. [25] also used MC Dropout for estimating the (epistemic) uncertainty of a gated transformer NN on the same prognostic dataset. Their results included the Prediction Interval Coverage Probability (PICP), and Prediction Interval Normalized Averaged Width (PINAW) metrics. Similarly, Nguyen et al. [17] used the PICP and ormalized Mean Prediction Interval Width (NMPIW) when evaluating their proposed lognorm-LSTM model. In their paper, they explored *system RUL* by using the validation set for quantifying the system level prognostic uncertainty. Neither work [17,25] make explicit arguments regarding the uncertainty calibration of the models. In fact, complementary analysis regarding uncertainty calibration seems uncommon in many works using the PICP and NMPIW metrics.

Stacking methods have previously been applied on RUL prognostics. Han et al. [8] implemented a multi-objective evolutionary algorithm to optimize both the accuracy and diversity of the RUL prediction on a bearings dataset. The selected models were stacked to create the final RUL (point) prediction. However, the article does not deal with the uncertainty estimation of the proposed stacking method. In fact, uncertainty estimation is a topic that has recently been identified as in need of further research within predictive health management [24].

This paper aims to bridge the research gap between stacking and uncertainty estimation in the context of RUL prognostics, as well as providing a large sur-

face area for comparison by utilizing multiple metrics for evaluating uncertainty estimation.

3 Preliminaries

3.1 Probabilistic Deep Learning

Hamiltonian Monte Carlo (HMC) [1] is a well known probabilistic method which has previously been applied for RUL prognostics [2]. In the context of DL, HMC is a method for obtaining samples from the probability distribution of the NN weights given the training data, called the posterior distribution. Asymptotically, this sampling gives a complete description of fitness of the model for all NN weights. However, the process of sampling network weights with HMC is difficult to tune, and costly in computing resources and time. Thus, HMC is not suitable for practical applications [10]. Instead of exploring the posterior distribution, VI [3] instead uses a surrogate distribution. This (simpler) distribution is then made as close to the true posterior as possible by minimizing the KullbackLeibler (KL) divergence between the two. Since calculating the true posterior is intractable, this in turn means that the KL divergence cannot be computed. Instead the Evidence Lower Bound (ELBO) is introduced to be equal to the KL divergence, plus some unknown constant. The VI surrogate can then be made similar to the true posterior (up to some unknown constant) by optimizing the ELBO.

SWAG has been proposed by Maddox et al. [15] as a method to approximate full Bayesian inference, making it an alternative probabilistic approach to HMC. This method saves samples of model weights during standard optimization (usually at the end of an epoch). These samples are viewed as being drawn from a Gaussian distribution which is fitted over the model weights. The distribution then acts as the posterior distribution after training has been completed. At inference time, network weights can be drawn from this distribution to sample as many models as necessary. Although calculating the SWAG takes some additional compute time, this method can be used to cheaply enable uncertainty representation in DNNs.

Deep ensembling techniques [9] aggregates the outputs of multiple DNNs in order to increase prediction accuracy and generalizability of the total system [6]. DNNs may end up in different optima based on their randomized initial parameters. This can highlight disagreements between the models, which will be lead to a broader predictive distribution. Bayesian stacking [26], on the other hand, has been proposed as a method of creating posterior distributions from non-mixing HMC chains. This can be seen as an alternative to naive ensembling of NNs. Instead of weighing the contributions to the aggregate prediction uniformly, the proposed stacking method assigns weights to each model depending on some criteria. Applied to HMC chains, this form of stacking can aggregate parallel inferences in order to cover as many posterior modes as possible. By constructing weights for each contributing distribution, stacking can reduce the influence from models that fit poorly to the data.

3.2 Metrics

Point estimation metrics are usually based on the mean value $\bar{y}$ of the predictive set $\hat{y}$. For a dataset with n number of data points, where the prediction set consists of m number of predictions we have $\hat{y}_{i,j}$, $i \in \{1, 2, ..., n\}$, $j \in \{1, 2, ..., m_i\}$. The mean prediction per data instance can then be described as:

$$\bar{y}_i = \frac{1}{m_i} \sum_{j=1}^{m_i} \hat{y}_{i,j} \tag{1}$$

Root Mean Squared Error (RMSE) measures the squared deviations between the mean value of the predictive distribtution $\bar{y}_i$ and the target value y_i. By taking the square root over the sum of all squared deviations, the RMSE emphasizes outliers, more so than the Mean Absolute Error (MAE), but produces a result on the same scale. The PICP metric [12] captures the frequency with which the target value is covered by the prediction interval. Ideally, the true target value should be covered by the prediction interval in relation to the width of the interval. The 95% prediction interval is commonly used [17,25]. As such, that PICP interval will also be used in this paper. However, model that naively yields a very wide prediction interval will also achieve a high PICP. The NMPIW metric [12], complements the PICP by measuring the (normalized) width of the prediction interval. As the metric only measures the width of the predictive distribution, a model can reduce the distribution to a single point prediction in order to obtain a perfect score, regardless of the accuracy of the model. Therefore, in this paper the PICP and NMPIW will be viewed jointly. Together they describe both the coverage of the model's prediction interval, and the width of the distribution.

Continuous Ranked Probability Score. Continuous Ranked Probability Score (CRPS) [7] measures how centered a distribution is around a target value, while also taking into account the distribution's width. This metric can serve as a probabilistic generalization of the MAE, as for a point prediction it becomes equivalent to the MAE. We define $F_{\hat{y}_i}(x)$ as the empirical cumulative distribution function (CDF) of the predictive distribution for data sample i.

$$CRPS = \frac{1}{n} \sum_{i=1}^{n} CRPS_i \tag{2}$$

$$CRPS_i = \int_{-\infty}^{\infty} (F_{\hat{y}_i}(x) - \mathbb{1}\{y_i \leq x\})^2 dx$$

$$\mathbb{1}\{y_i \leq x\} = \begin{cases} 1, & y_i \leq x \\ 0, & \text{otherwise} \end{cases}$$

Weighted CRPS. In real-world settings, overestimating the RUL (a late prediction) usually leads to more severe consequences [14]. To address this, Ingeborg and Mitici [18] propose the Weighted Continuous Ranked Probability Score

(CRPS$^{\mathrm{W}}$), which modifies the CRPS to further penalize predictions that fall after the target RUL. The CRPS$^{\mathrm{W}}$ is defined as:

$$
\begin{aligned}
CRPS^W &= \frac{1}{n}\sum_{i=1}^{n} CRPS_i^W \qquad (3)\\
CRPS_i^W &= (2-\beta)\int_{-\infty}^{y_i}(F_{\hat{y}_i}(x) - \mathbb{1}\{y_i \le x\})^2 dx \\
&\quad + \beta\int_{y_i}^{\infty}(F_{\hat{y}_i}(x) - \mathbb{1}\{y_i \le x\})^2 dx \\
&= (2-\beta)\int_{-\infty}^{y_i}(F_{\hat{y}_i}(x))^2 dx + \beta\int_{y_i}^{\infty}(F_{\hat{y}_i}(x)-1)^2 dx
\end{aligned}
$$

with $0 \le \beta \le 2$ being a user-specified parameter that controls the weighing of early versus late predictions. Ingeborg and Mitici [18] utilize a β value of 1.5 on the same dataset as used in this paper to give greater weight to late predictions, thus we use the same β value in our experiments.

α-Coverage and Ranked Probability α-Coverage [18] is introduced as a complement to the accuracy and sharpness measurement of the CRPS. This metric constructs a credible interval around the median predicted value with a width α. For example, when ordering the predictions $\hat{y}_i$, the span $[\hat{y}_i^{0.30}, \hat{y}_i^{0.70}]$ represents all predictions between the 30$^{\text{th}}$ and the 70$^{\text{th}}$ percentiles. The α-Coverage for a dataset with n data points can then be defined as:

$$
\begin{aligned}
\alpha\text{-Coverage} &= \frac{1}{n}\sum_{i=1}^{n}\mathcal{I}(\alpha)_i \\
\mathcal{I}(\alpha)_i &= \begin{cases} 1, & y_i \in [\hat{y}_i{}^{0.5-0.5\alpha}, \hat{y}_i{}^{0.5+0.5\alpha}] \\ 0, & \text{otherwise} \end{cases}
\end{aligned}
$$

Reliability Score (RS) [18] is a calibration curve for regression (as opposed to classification) tasks. It is obtained by measuring the α-Coverage for different values of α. If the target value falls inside the predicted interval significantly more or less than α percent of the time, then the model's uncertainty estimates are not well-calibrated. The RS metric can be visualized in a diagram by plotting the measured α-Coverage for varying values of α.

4 Empirical Likelihood Stacking

In this section we describe the proposed methodology of the empirical likelihood stacking procedure for creating a meta-model from trained NNs. Our contribution is based on previous work done by Yao et al. [26], and we direct interested readers to that paper for further reading.

The stacking model is constructed by obtaining a model-wise weight $\mathbf{w} = (w_1, w_2, ..., w_K)$ where K is the number of models at our disposal. In this paper,

we propose using the validation data as a simple and computationally cheap way for calculating the weights $\mathbf{w}$. The rationale for basing the stacking weights on the validation dataset is that the validation set is a common target for regularization techniques, such as early stopping, and is often included in modern DL experiments by default. It is therefore straightforward for practitioners to implement this extra step of weight optimization. In light of these aspects, we propose basing the weights on the models' empirical performance on the validation set. Thus, to compute the weights of the k-th model we first calculate the predictive density p with a kernel density estimation (KDE)[1] using a (normalized) gaussian kernel $\mathbf{K}$:

$$p_k(y_i, (\hat{y}_{i,j})_{j=1}^m) = \sum_{j=1}^{m} \mathbf{K}(y_i - \hat{y}_{i,j}; h)$$

where $\hat{y}_i$ are the prediction samples, and y_i is the target RUL for datapoint $i \in \{1, 2, ..., n_v\}$ in a validation dataset of size n_v. The result of the KDE is highly dependent on the the kernel width h, as a misspecified value of h can lead to either an under- or over-smoothed estimation. During initial experimentation, it was observed that the predictions from both SWAG and VI were approximately normal in shape. Because of this, Scott's rule [20] should perform well, and was used to set the width parameter h. Then, as proposed by Yao et al. [26], we solve

$$\mathbf{w}^*_{1,...,K} = \arg\max_{\mathbf{w} \in \mathbb{S}(K)} \sum_{i=1}^{n_v} \log \sum_{k=1}^{K} w_k p_k(y_i, (\hat{y}_{i,j})_{j=1}^{m_k}) + \log(p_{\text{prior}}(\mathbf{w})),$$

where m_k is the number of parameter samples from k-th model, and $\mathbb{S}(K)$ is the space of a $(K-1)$-dimensional simplex

$$\mathbb{S}(K) = \{\mathbf{w} : 0 \leq w_k \leq 1, \forall\, k\,; \sum_{k=1}^{K} w_k = 1\}$$

$p_{\text{prior}}(\mathbf{w})$ is a Dirichlet prior, tuned by a scaling parameter λ that forces convexity of the optimization of $\mathbf{w}$ for all $\lambda > 1$. The Dirichlet prior and λ term is further described in the original work [26]. For this paper, the proposed rule of thumb of $\lambda = 1.001$ was used. To optimize the weights $\mathbf{w}$, the same method[2] and parameters are used as by Yao et al. [26]. Lastly, we perform importance resampling by selecting parameter draws from each model proportional to its weight. This allows us to calculate the Monte Carlo estimate as described in [26]: For the K model functions f parameterized by θ as

$$\mathbb{E}(f(\theta)|\mathbf{w}) \approx \sum_{k=1}^{K} w_k \sum_{m=1}^{m_k} \frac{f(\theta_{km})}{m_k}$$

where m_k is the number of NN parameter samples for the k-th model.

[1] https://scikit-learn.org/stable/modules/density.html#kernel-density.
[2] https://github.com/yao-yl/Multimodal-stacking-code.

5 Experiments

This section contains details about the setup of the experiment. First we describe the dataset and the data-processing steps taken in Sect. 5.1. In Sect. 5.2 we briefly outline the NN architecture used in this paper.

5.1 Description of the Dataset

The proposed method was applied to a well known turbofan degradation dataset consisting of measurements from simulated engines. The data was generated by the tool C-MAPSS [19], developed by NASA. The dataset contains four subsets: FD001, FD002, FD003, and FD004. These subsets are of varying complexity with one or more fault- and operating conditions. For each subset, the data samples are divided into predefined training and test sets. The data processing followed the same process as outlined by Costa and Sánchez [5]. The training data was split into 80-20% training and validation sets. All data is scaled with a standard scaler with respect to operating condition. The data are likely to contain some noise, which may cause problems for the DL models. Therefore, the sensor data are also passed through an exponential weighted smoothing process [5]. Of the total 21 sensors, 7 show constant measurement over time for subsets FD001 and FD003. During data preparation, these sensors were discarded for these two subsets, leaving a total of 14 sensor values for subsets FD001 and FD003, and 21 sensors for FD002 and FD004. When a machine operates without signs of degradation, it is reasonable to assume that the sensor values will be nearly constant until the engine starts degrading. This assumption is supported by looking at the sensor data. The data seem to be constant (with some measurement noise) until a point, after which they increasingly deviate until a fault occurs. Therefore, previous work [2,5] on the turbofan data apply a piece-wise linear function on the RUL. This bounds the maximum RUL for all training and validation data to 125. Another common processing step introduced in [14] was to apply a sliding window over the time axis of the data. This transforms the time series into an image-like two dimensional object, which fits a CNN well. This also allows the NN to view the immediate history of a component, which can assist in detecting ongoing trends. Similar to related work [2,18], the window sizes were chosen as 30, 20, 30, and 15 for subsets FD001, FD002, FD003, and FD004, respectively.

5.2 Model Architecture

A CNN architecture is proposed, inspired by previous work [14,18]. The model consists of 5 convolutional layers, followed by a dense (fully connected layer). The convolutional layers has a one dimensional kernel and a tanh activation function. The first four convolutional layers has a kernel size of 10 (along the time axis), while the fifth layer has a size of 3. For regularization, dropout layers were applied after convoluational layers 2, 3, and 4 during training. Each dropout layer has a probability of 0.5 for dropping any given neuron in the previous layer, which is the same probability as used in [18]. The dense layer has 420 neurons and a Leaky

ReLU activation function. The SWAG CNN is trained with the Adam optimizer [13] for 250 epochs, and with a constant learning rate of 0.005. The learning rate was kept constant to increase the diversity of the parameter samples. The weights of the NN is initialized with the default PyTorch initialization scheme for SWAG, and set to normal distributions with a prior scale of 0.1 for VI. Finally, the last 30% of the epochs are saved and added to the SWAG weights. The VI model is similarly trained with the Adam optimizer for 250 epochs, with a decaying learning rate from 0.01 to 0.001. Fifty SWAG and VI NNs were trained on each turbofan data subset. The relative high number of models were chosen to ensure statistical validity of the result. Prior to the experiments, the hyper-parameters for each model were fine tuned by manual search, in the cases where they were not already explored by previous work. For both methods, the *single* NN with the highest likelihood on the validation data was chosen as a point of comparison against the stacking and ensembling methods. The deep ensemble was constructed by uniformly aggregating the predictions from all 50 networks. The stacking was conducted on the 50 networks as detailed in Sect. 4.

6 Results and Discussion

The PICP and NMPIW offer complementary descriptions of the predictive distributions. As seen in Table 1, the NMPIW follow a clear pattern where the single model has the smallest distribution width, followed by stacking and finally the ensemble. This holds true for all subsets of the data. For the PICP, the metric tends to favor the stacking model over the ensemble, with the single model clearly performing the worst. Figure 1 shows the NMPIW and PICP metrics for the three variants of the SWAG and VI models on each subset of the data. Although not always best in either metric, the stacking model manages to often come closest to the combined optimal point (this is most clear for subset FD002 in Fig. 3). As is shown in Table 2, there is clear evidence that both empirical likelihood stacking and ensembling provides better uncertainty estimation than a

Table 1. PICP and NMPIW of the three models, per subset. For PICP higher is better, for NMPIW lower is better.

Subset	Metric	SWAG			VI		
		Single	Stacking	Ensemble	Single	Stacking	Ensemble
FD001	PICP ↑	0.43	0.71	0.79	0.43	0.68	0.64
	NMPIW ↓	0.16	0.38	0.50	0.30	0.45	0.64
FD002	PICP ↑	0.37	0.57	0.50	0.46	0.52	0.51
	NMPIW ↓	0.14	0.26	0.49	0.31	0.36	0.56
FD003	PICP ↑	0.42	0.61	0.61	0.49	0.62	0.61
	NMPIW ↓	0.14	0.28	0.36	0.38	0.55	0.70
FD004	PICP ↑	0.34	0.47	0.45	0.35	0.47	0.45
	NMPIW ↓	0.18	0.28	0.38	0.32	0.51	0.67

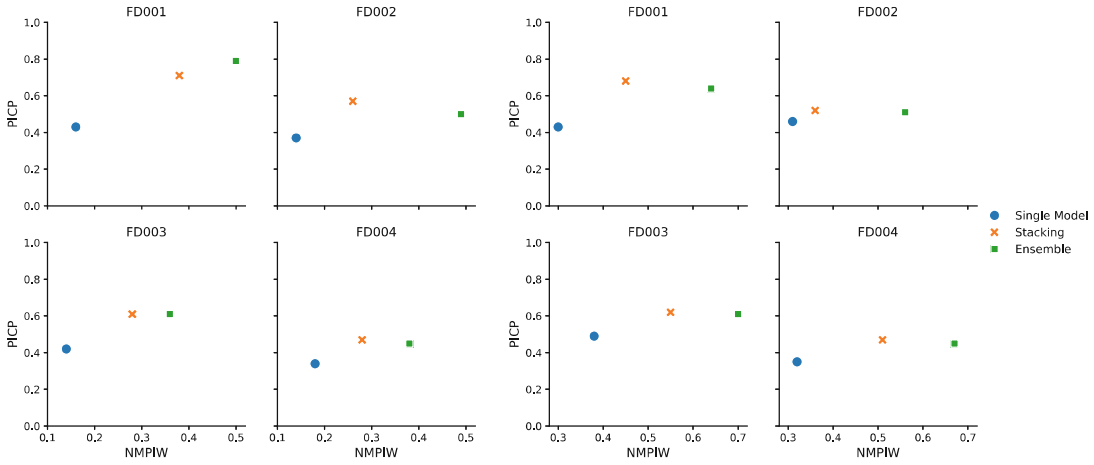

Fig. 1. PICP and NMPIW scores for SWAG (left) and VI (right). Each plot shows the result for one data subset, with the three model configurations (Single, Stacked, and Ensemble). Top left corner corresponds to perfect PICP and NMPIW scores.

single DNN. For the PICP, a wider prediction interval will have an increased RUL coverage, and thus a higher score. Therefore, it is expected that the ensemble model, which uses all predictions, achieves the highest PICP score. However, this is not always the case as the stacking model often performs as well, if not better, than the ensemble model in this regard. The NMPIW penalizes the width of the prediction interval, heavily favoring the single model. An interesting observation is that the stacking model often achieves a similar PICP as the ensemble, but with a lower NMPIW score. This indicates that the prediction intervals of the stacking model is of higher quality than either the single model, or the ensemble.

For RMSE and CRPS, the ensemble consistently outperforms the other models in the SWAG case. However, for VI the situation is the opposite, with the stacking model often performing best with respect to CRPS and CRPS^{W}. Since the mean prediction is used to calculate the RMSE, having more predictions available seems to allow the ensemble to achieve a better RMSE than both the stacking and single model variants in most cases. The CRPS measures the closeness of all predictions to the target RUL value. Similarly, the findings indicate a clear trend in which a larger number of predictions result in better CRPS performance. In terms of RS, the stacking model achieves the best score, closely followed by the ensemble. This can be seen in Fig. 2, where the stacking model falls closer to the ideal line for the majority of different credible interval widths. This shows that the additional effort of stacking and ensembling trained models yields better-calibrated uncertainty. Notably, the stacking model slightly outperforms the ensemble here, which indicates that the act of weighing the models further improves the uncertainty calibration. As stated in the introduction, DNNs are known for underestimating their uncertainty. Although stacking and ensembling improves the uncertainty calibration Fig. 2 shows that these models still underestimate their predictive uncertainty.

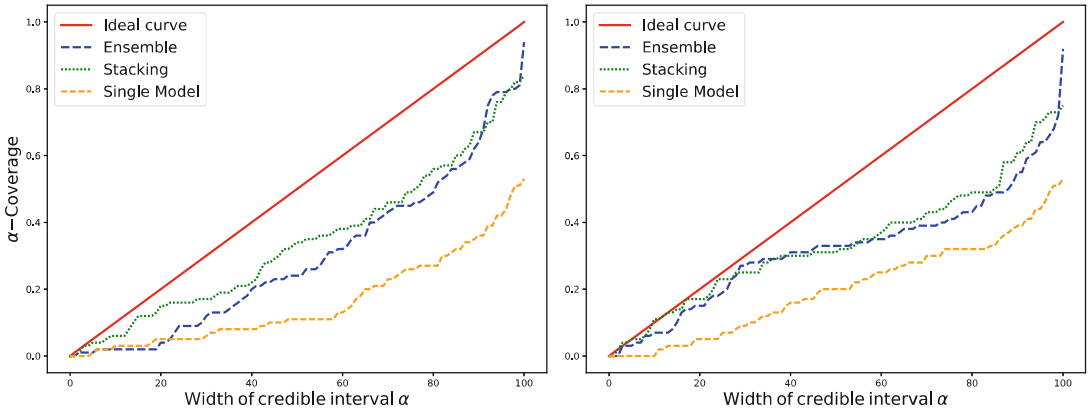

Fig. 2. Reliability diagram for SWAG (left) and VI (right), of the three models for the FD001 dataset. Lines falling below the ideal curve indicate models which are underestimating their predictive uncertainty.

Table 2. RMSE, CRPS, CRPS^{W}, and RS of the single best, stacked, and ensemble models for SWAG and VI per data subset. Top performing model for SWAG and VI (separately) highlighted in bold. Total number of times each model achieved the top performance is summarized at the bottom.

Subset	Metric	SWAG			VI		
		Single	Stacking	Ensemble	Single	Stacking	Ensemble
FD001	RMSE ↓	14.65	14.08	**13.96**	25.68	**24.22**	24.38
	CRPS ↓	9.94	9.04	**8.53**	17.39	**14.43**	15.26
	CRPS^{W} ↓	10.79	**8.91**	9.44	17.40	**15.14**	16.31
	RS ↓	0.34	**0.17**	0.21	0.30	**0.16**	0.18
FD002	RMSE ↓	26.73	27.01	**26.69**	**31.44**	31.59	31.68
	CRPS ↓	16.98	15.87	**15.79**	21.38	20.30	**20.27**
	CRPS^{W} ↓	16.98	11.75	**11.69**	21.38	**17.65**	17.71
	RS ↓	0.35	**0.27**	0.29	0.30	0.28	**0.27**
FD003	RMSE ↓	15.18	15.60	**15.03**	28.52	28.09	**28.04**
	CRPS ↓	9.76	9.31	**9.04**	18.96	**17.51**	17.85
	CRPS^{W} ↓	9.76	**9.32**	9.37	22.31	**20.15**	21.33
	RS ↓	0.34	**0.20**	0.21	0.27	0.23	**0.21**
FD004	RMSE ↓	**28.64**	29.27	29.44	35.55	35.45	**35.34**
	CRPS ↓	19.55	**19.20**	19.26	25.3	**22.53**	23.99
	CRPS^{W} ↓	15.31	**14.38**	14.47	24.81	**19.98**	22.37
	RS ↓	0.34	**0.29**	0.30	0.34	**0.28**	0.30
# of top performing		1	**8**	7	1	**10**	5

As shown in Table 2 for SWAG, the ensemble generally achieves a lower (better) CRPS score, while for VI, stacking has lowers scores in both CRPS and CRPS^{W}. For both SWAG and VI, stacking outperforms in terms of RS in the majority of cases.

Overall, there is a clear advantage of using stacking or ensembling for SWAG and VI models in all compared metrics. Judging from the combined PICP and NMPIW scores, the stacking model consistently dominates the ensemble in the trade-off for both SWAG and VI. This notion is also supported by the RS metric, where stacking often beats the ensemble.

7 Summary and Conclusions

In this paper, we propose empirical likelihood stacking, inspired by Yao et al. [26], we propose to use the validation dataset together with KDE, instead of the cross-validated training set for calculating the stacking weights. We have evaluated and compared the proposed stacking method with ensembling for DNN models trained with SWAG and VI. The models were implemented and trained on the well known C-MAPSS dataset for simulated turbofan engines. While both deep ensemble and stacking improves most aspects of the prediction when compared to the single best model, we find that stacking improved the quality of the prediction intervals of both the VI and SWAG models when compared with the ensemble. Results also showed that stacking improved the uncertainty calibration of SWAG in comparison to the deep ensemble, leading to a general reduction in over-confidence. This can be highly important for maintenance decision making, as well calibrated uncertainty is a requirement for informed decision-making in scenarios with high risk. In addition, in the case of VI, stacking outperformed the ensemble in most metrics. The predictions of the stacking model also tended to underestimate the RUL more often than the ensemble, which may be desirable in a real-world scenario where missed maintenance is more costly than early maintenance. Regardless of using stacking or ensembling, an argument can be made regarding the additional computational cost of training multiple models. However, this process is highly parallelizable as the training of the individual models can be done in complete separation. Once the models have been trained, the computational cost of calculating the stacking weights is negligible.

Questions arise about why stacking tends to outperform the ensemble to a higher degree when applied to VI, compared to SWAG. One reason for this discrepancy might be the initialization of the model weights and biases. For SWAG, these parameters are likely less diverse than the parameters sampled from the distributions used for VI. As shown by Yao et al. [26] stacking performs better on over-dispersed Markov Chain Monte Carlo (MCMC) chains. This might be a key reason for limited improvements in various metrics when comparing stacking with ensembling for the SWAG models. As the initialization of the weights follow the same distribution for all models, it might be that the final models are too similar to fully utilize the benefits of stacking. Hence, potential future work is to increase the diversity among the trained models and identifying

the factors that can improve the quality of stacking weights. Second, this paper used the validation set as the basis for the stacking weights. This was done due to the ubiquitous nature of the validation dataset, as it is commonly used in the DL context for regularization methods. Pareto smoothed importance sampling (PSIS) is used to estimate the leave-one-out (LOO) cross-validation in [26] for calculating the stacking weights. Finding the differences in empirical performance between stacking weights based on estimated LOO cross-validation versus the validation set would provide an interesting point of comparison for future work. Finally, the regularization term λ should be fine-tuned and the effect on the stacking weights evaluated.

Acknowledgments. The authors would like to thank the Advanced and Innovative Digitalization Program funded by VINNOVA for their funding of the research projects TPdM-Trustworthy Predictive Maintenance (Grant No. 2022-01710) and AIMOps-Advanced AI Architectures for Integrated and Enhanced Manufacturing Operations (Grant No. 2025-01110), within which this study has been conducted.

References

1. Andrieu, C., De Freitas, N., Doucet, A., Jordan, M.I.: An introduction to MCMC for machine learning. Mach. Learn. **50**, 5–43 (2003)
2. Benker, M., Furtner, L., Semm, T., Zaeh, M.F.: Utilizing uncertainty information in remaining useful life estimation via Bayesian neural networks and Hamiltonian Monte Carlo. J. Manuf. Syst. **61**, 799–807 (2021)
3. Blei, D.M., Kucukelbir, A., McAuliffe, J.D.: Variational inference: a review for statisticians. J. Am. Stat. Assoc. **112**(518), 859–877 (2017)
4. Chen, C., Shi, J., Lu, N., Zhu, Z.H., Jiang, B.: Data-driven predictive maintenance strategy considering the uncertainty in remaining useful life prediction. Neurocomputing **494**, 79–88 (2022)
5. Costa, N., Sánchez, L.: Variational encoding approach for interpretable assessment of remaining useful life estimation. Reliab. Eng. Syst. Saf. **222**, 108353 (2022)
6. Ganaie, M.A., Hu, M., Malik, A.K., Tanveer, M., Suganthan, P.N.: Ensemble deep learning: a review. Eng. Appl. Artif. Intell. **115**, 105151 (2022)
7. Gneiting, T., Raftery, A.E.: Strictly proper scoring rules, prediction, and estimation. J. Am. Stat. Assoc. **102**(477), 359–378 (2007)
8. Han, T., Pang, J., Tan, A.C.: Remaining useful life prediction of bearing based on stacked autoencoder and recurrent neural network. J. Manuf. Syst. **61**, 576–591 (2021)
9. Hansen, L.K., Salamon, P.: Neural network ensembles. IEEE Trans. Pattern Anal. Mach. Intell. **12**(10), 993–1001 (1990)
10. Izmailov, P., Vikram, S., Hoffman, M.D., Wilson, A.G.G.: What are Bayesian neural network posteriors really like? In: International Conference on Machine Learning, pp. 4629–4640. PMLR (2021)
11. Khan, S., Yairi, T.: A review on the application of deep learning in system health management. Mech. Syst. Signal Process. **107**, 241–265 (2018)
12. Khosravi, A., Nahavandi, S., Creighton, D., Atiya, A.F.: Lower upper bound estimation method for construction of neural network-based prediction intervals. IEEE Trans. Neural Netw. **22**(3), 337–346 (2010)

13. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization (2017). https://arxiv.org/abs/1412.6980
14. Li, X., Ding, Q., Sun, J.Q.: Remaining useful life estimation in prognostics using deep convolution neural networks. Reliab. Engi. Syst. Safe. **172**, 1–11 (2018)
15. Maddox, W.J., Izmailov, P., Garipov, T., Vetrov, D.P., Wilson, A.G.: A simple baseline for Bayesian uncertainty in deep learning. In: Advances in Neural Information Processing Systems, vol. 32 (2019)
16. Mitici, M., de Pater, I., Barros, A., Zeng, Z.: Dynamic predictive maintenance for multiple components using data-driven probabilistic RUL prognostics: The case of turbofan engines. Reliab. Eng. Syst. Saf. **234**, 109199 (2023)
17. Nguyen, K.T., Medjaher, K., Gogu, C.: Probabilistic deep learning methodology for uncertainty quantification of remaining useful lifetime of multi-component systems. Reliab. Engi. Syst. Saf. **222**, 108383 (2022)
18. de Pater, I., Mitici, M.: Novel metrics to evaluate probabilistic remaining useful life prognostics with applications to turbofan engines. In: PHM Society European Conference, vol. 7, pp. 96–109 (2022)
19. Saxena, A., Goebel, K., Simon, D., Eklund, N.: Damage propagation modeling for aircraft engine run-to-failure simulation. In: 2008 International Conference on Prognostics and Health Management, pp. 1–9. IEEE (2008)
20. Scott, D.W.: On optimal and data-based histograms. Biometrika **66**(3), 605–610 (1979)
21. Serradilla, O., Zugasti, E., Rodriguez, J., Zurutuza, U.: Deep learning models for predictive maintenance: a survey, comparison, challenges and prospects. Appl. Intell. **52**(10), 10934–10964 (2022)
22. Si, X.S., Wang, W., Hu, C.H., Zhou, D.H.: Remaining useful life estimation-a review on the statistical data driven approaches. Eur. J. Oper. Res. **213**(1), 1–14 (2011)
23. Staber, B., Da Veiga, S.: Benchmarking Bayesian neural networks and evaluation metrics for regression tasks. arXiv preprint arXiv:2206.06779 (2022)
24. Thelen, A., Huan, X., Paulson, N., Onori, S., Hu, Z., Hu, C.: Probabilistic machine learning for battery health diagnostics and prognostics—review and perspectives. NPJ Mater. Sustain. **2**(1), 14 (2024)
25. Xiang, F., Zhang, Y., Zhang, S., Wang, Z., Qiu, L., Choi, J.H.: Bayesian gated-transformer model for risk-aware prediction of aero-engine remaining useful life. Expert Syst. Appl. **238**, 121859 (2024)
26. Yao, Y., Vehtari, A., Gelman, A.: Stacking for non-mixing Bayesian computations: the curse and blessing of multimodal posteriors. J. Mach. Learn. Res. **23**(79), 1–45 (2022)
27. Zonta, T., Da Costa, C.A., da Rosa Righi, R., de Lima, M.J., Da Trindade, E.S., Li, G.P.: Predictive maintenance in the industry 4.0: a systematic literature review. Comput. Ind. Eng. **150**, 106889 (2020)