Virtual Load Monitoring of Offshore Wind Farms via Uncertainty-Aware Deep Learning Models

Hlaing, Nandar; Morato, Pablo G.; de N. Santos, Francisco; Weijtjens, Wout; Devriendt, Christof

Publication date
2025

Document Version
Final published version

Citation (APA)
Hlaing, N., Morato, P. G., de N. Santos, F., Weijtjens, W., & Devriendt, C. (2025). *Virtual Load Monitoring of Offshore Wind Farms via Uncertainty-Aware Deep Learning Models*. Paper presented at 6th ECCOMAS Thematic Conference on Uncertainty Quantification in Computational Science and Engineering, Rhodes, Greece.

UNCECOMP 2025
6th ECCOMAS Thematic Conference on
on Uncertainty Quantification in Computational Science and Engineering
M. Papadrakakis, M. Fragiadakis (eds.)
Rhodes, Greece, 15–-18 June 2025

VIRTUAL LOAD MONITORING OF OFFSHORE WIND FARMS VIA UNCERTAINTY-AWARE DEEP LEARNING MODELS

Nandar Hlaing[1], Pablo G. Morato[2], Francisco de N. Santos[1], Wout Weijtjens[1], and Christof Devriendt[1]

[1]OWI-Lab, Vrije Universiteit Brussel
1050 Brussels, Belgium
e-mail: {nandar.hlaing, francisco.de.nolasco.santos, wout.weijtjens, christof.devriendt}@vub.be

[2] Faculty of Architecture and the Built Environment, Delft University of Technology
2628 BL Delft, Netherlands
e-mail: p.g.moratodominguez@tudelft.nl

Keywords: Structural health monitoring, virtual load monitoring, probabilistic deep learning, uncertainty quantification, offshore wind farms.

Abstract. *This work explores probabilistic deep learning models as offshore farm-wide virtual load sensors, including Bayesian neural networks, Monte Carlo dropout, and deep neural network ensembles. The aim is to develop models offering uncertainty-aware predictions of damage equivalent loads using SCADA and accelerometer data. This study uses the data from a Belgian offshore wind farm's five fleet-leader turbines. After training the neural networks with one year of collected data, these models are deployed to another year, facing out-of-distribution data due to changes in operational conditions. The analysis assesses generalization and uncertainty quantification abilities, providing insights into their strengths and weaknesses. Ultimately, our work supports the industrial adoption of probabilistic deep learning virtual monitoring models, enabling informed asset management decisions with predictions and uncertainty measures.*

1 INTRODUCTION

Benefiting from the continuous improvement of sensor technologies, learning algorithms, and high-performance computational hardware, data-driven solutions for the assessment and management of offshore wind assets are increasingly being adopted. In particular, the global approximation qualities of deep neural networks can be leveraged to effectively extract complex patterns from large datasets, rendering deep learning approaches highly useful in wind energy applications—ranging from wind speed and direction forecasting [Mbuvha et al., 2021] to fault detection [Cho et al., 2021] and structural health monitoring [Hlaing et al., 2022], among others.

Deterministic neural networks do not, however, explicitly capture model uncertainty, which is a crucial factor in the safety and reliability of offshore wind assets exposed to stochastic loads and stressors. Since environmental and operational conditions vary over time, already trained deep learning models may receive out-of-distribution data in practical scenarios, i.e., data unseen by the model during the training stage. In other scenarios, such as farm-wide load monitoring, prediction models trained on one or multiple fleet leaders may be applied to other wind turbines, some of which may belong to different design clusters. When load measurements are unavailable to validate model predictions, deterministic estimates do not provide sufficient information for effective decision-making.

On the other hand, probabilistic neural networks explicitly provide model uncertainty indicators [Pimenta et al., 2022, Zou et al., 2023, Singh et al., 2024]. It has been shown that probabilistic models based on Bayesian principles are capable of intrinsically reporting higher model uncertainties when tested on out-of-distribution data, for which the predictions are likely to be inaccurate [Hlaing et al., 2024]. This capability is of importance for the safe deployment of deep learning models in practical decision-making applications— such as inspection and maintenance planning, lifetime extension— where the quantification of imperfect information plays a vital role [Hlaing, 2024].

In this work, we investigate the effectiveness of probabilistic deep learning models as farm-wide virtual load sensors. Specifically, we compare probabilistic neural network variants, including Monte Carlo dropout, Bayes by backprop, and neural network ensembles. The primary objective of this comparative study is to identify models that not only effectively predict damage equivalent loads but also capture the uncertainty associated with model predictions. Such uncertainty-aware models can support informed decision-making in the dynamic and uncertain environment characteristic of offshore wind farms. Our analysis is based on data collected over 2 years from five fleet-leader turbines (i.e., turbines equipped with SCADA system and strain gauges) at a Belgian offshore wind farm. Data collected from the first year is used to train the neural networks, while data collected from the second year— during which the turbines operate under different control conditions— is used for testing. This scenario introduces a distribution shift between the training and testing datasets. Our evaluation focuses on the generalization and uncertainty quantification of the tested probabilistic neural networks. Throughout the analysis, we highlight the strengths and limitations of each model. The findings of this study are expected to accelerate the industrial adoption of probabilistic deep learning models for virtual load monitoring.

2 PROBABILISTIC NEURAL NETWORKS

2.1 Bayesian neural networks

A Bayesian neural network (BNN) is an artificial neural network that integrates deep learning with Bayesian inference principles to generate probabilistic predictions. Unlike deterministic artificial neural networks (ANNs), the distinctive feature of BNNs lies in their stochastic neural network components, including stochastic weights and/or biases. Stochastic model parameters, $\boldsymbol{\theta}$, each characterized by a probability distribution $P(\boldsymbol{\theta})$, encode multiple possible model parametrizations, thus capturing the model's epistemic uncertainty. BNNs are more interpretable than their deterministic counterparts, as they signal high model uncertainty when making predictions for samples that are far from those contained in the training dataset. This feature makes them particularly useful in scenarios with limited data availability.

BNNs conduct Bayesian inference to estimate the posterior distribution of model parameters given the training data. Conditioned to the training dataset D, the posterior $P(\boldsymbol{\theta} \mid D)$ can be computed via Bayes' theorem:

$$P(\boldsymbol{\theta} \mid D) = \frac{P(D \mid \boldsymbol{\theta})P(\boldsymbol{\theta})}{\int P(D \mid \boldsymbol{\theta})P(\boldsymbol{\theta})d\boldsymbol{\theta}}. \tag{1}$$

Computing the posterior distribution is intractable in practical high-dimensional scenarios. To overcome this limitation, approximate inference methods have been proposed in the literature, e.g., stochastic gradient Markov Chain Monte Carlo [Welling and Teh, 2011, Zhang et al., 2019] and variational methods [Graves, 2011].

2.1.1 Variational inference

Variational inference (VI) is a technique used in Bayesian statistics to approximate complex posterior distributions [Graves, 2011, Osawa et al., 2019]. In VI, a simpler distribution, known as the variational distribution, is used to approximate the posterior distribution of model weights. The main objective of VI is to derive a variational distribution $q_\lambda(\boldsymbol{\theta})$ as close as possible to the true unknown posterior $P(\boldsymbol{\theta} \mid D)$. The similarity between both distributions is measured using the Kullback-Leibler (KL) divergence, which quantifies the non-symmetric statistical difference between the two distributions. The KL divergence between $q_\lambda(\boldsymbol{\theta})$ and $P(\boldsymbol{\theta} \mid D)$ can be mathematically expressed as:

$$\mathbb{KL}\left(q_\lambda(\boldsymbol{\theta}) \mid\mid P(\boldsymbol{\theta} \mid D)\right) = \int q_\lambda(\boldsymbol{\theta}) \log \frac{q_\lambda(\boldsymbol{\theta})}{P(\boldsymbol{\theta} \mid D)} d\boldsymbol{\theta}. \tag{2}$$

VI seeks to find the variational parameters λ that minimize the KL divergence formulated above, resulting in the maximizatioin of the evidence lower bound objective (ELBO) function:

$$\text{ELBO} = -\int q_\lambda(\boldsymbol{\theta}) \log \frac{q_\lambda(\boldsymbol{\theta})}{P(\boldsymbol{\theta})} d\boldsymbol{\theta} + \int q_\lambda(\boldsymbol{\theta}) \log P(D \mid \boldsymbol{\theta}) d\boldsymbol{\theta}. \tag{3}$$

The ELBO loss includes two terms: a regularization term enforcing consistency with prior knowledge, and an expected log-likelihood term controlling that the model fits the training data. The relative importance of each term in the loss can be adjusted to optimize model performance. During training, the negative log-likelihood can be computed by drawing only one neural network sample per forward pass, rather than computing the expectation over multiple model realizations, thus reducing computational effort. Gradient descent can still be effectively performed despite the presence of noisy gradient estimates.

Bayes by backprop

Bayes by backprop follows the aforementioned VI approach to approximate the posterior distribution over neural network weights [Blundell et al., 2015]. This approach combines the principles of Bayesian inference with the optimization techniques of backpropagation. By assuming the variational posterior as a multivariate standard Gaussian, i.e, $q_\lambda \sim \mathcal{N}(\mathbf{0}, I)$, Bayes by backprop uses the reparametrization trick:

$$\boldsymbol{\theta} = \mu_{\boldsymbol{\theta}} + \sigma_{\boldsymbol{\theta}} \cdot \varepsilon, \text{ where } \varepsilon \sim \mathcal{N}(0,1). \tag{4}$$

Random weights θ are sampled from the variational distribution using the reparameterization trick. This formulation also enables the computation of the gradients and updates of the variational parameters using standard optimization techniques. Following Equation 3, the model loss at the k^{th} epoch is can be computed as:

$$\mathcal{L} = \alpha \sum_{j=1}^{N_\theta} \mathbb{KL}\left(q_\lambda(\theta_k^j) \,||\, P(\theta_0^j)\right) + \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{5}$$

where α is the KL loss factor, N_θ is the number of model parameters, N is the size of training dataset, y and $\hat{y}$ are the target labels and model predictions.

Monte Carlo dropout

Dropout is a regularization technique where, during training, random units (neurons) are "dropped out" or set to zero with a certain probability. This helps prevent overfitting and encourages robustness of the model [Wager et al., 2013]. Monte Carlo dropout (MC-Dropout) is an approach that extends the idea of dropout regularization in neural networks to approximate Bayesian inference [Gal and Ghahramani, 2016]. During the training of MC-Dropout networks, a dropout rate $p_d \in [0, 1]$ is applied, randomly deactivating a portion of neurons in each forward and backward pass, introducing stochasticity to the model. The loss function can be formulated as:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 + \phi\|\boldsymbol{\theta}\|_2^2. \tag{6}$$

in which the second term corresponds to L2 regularization and ϕ is the regularization coefficient. During the forward prediction of BNNs, dropout is still applied, being opposed to the traditional dropout in which dropout is only applied during training, but all neurons are active during prediction. Multiple forward passes are performed through the network, each time randomly deactivating neurons with a specified dropout probability. This process is similar to drawing multiple samples from a probabilistic model. MC-Dropout provides a practical and efficient way to obtain uncertainty estimates in neural network predictions and can be seen as an approximation of Bayesian neural networks, while full Bayesian inference is computationally expensive.

2.2 Deep ensembles

Deep ensemble networks, also known as model ensembles, are a technique in machine learning that involves training multiple deep neural networks independently and then combining their predictions to produce more robust and accurate predictions. Since they do not apply Bayesian principles, deep ensembles may not be categorized as Bayesian neural networks. The key idea behind deep ensemble networks is to leverage the diversity of multiple models to improve generalization and reduce overfitting, standing as a sound reference for comparison.

Instead of training a single deep neural network, multiple networks are trained independently through a different random initialization and/or different training hyperparameters. Once all the individual networks are trained, they are used to make predictions on new, unseen data points. Each network produces its own prediction which are then combined to produce a final prediction. Mathematically, if N_n individual neural networks are considered, each producing a prediction denoted as $\hat{y}_i$ for the same input data point, the ensemble prediction $\hat{y}_{\text{ensemble}}$ can be computed as:

$$\hat{y}_{\text{ensemble}} = \sum_{i=1}^{N_n} w_i \hat{y}_i \tag{7}$$

where w_i represents the importance weight assigned to each neural network model's prediction. The weights can be uniform (equal for all models) or learned during the training process to optimize the performance of the ensemble.

3 BENCHMARKING PROBABILISTIC NEURAL NETWORKS AS VIRTUAL SENSORS FOR OFFSHORE WIND FARMS

Typically, equipping an entire offshore wind farm with strain sensors is not cost-effective, primarily due to the high installation and maintenance expenses associated with this process. In this context, virtual load monitoring offers a practical solution. It involves generating load-related information using readily accessible monitoring data, which often includes SCADA and accelerometer data. In this regard, one or a selected group of turbines, often called the 'fleet-leader(s)', are fully equipped, allowing the collection of concurrent SCADA, acceleration, and load data required for the training of virtual models. The trained model can later be used to replace the damaged strain sensors of the fleet-leader(s) or predict the loads on other turbines in the farm.

In this study, we compare the afore-mentioned probabilistic neural network models with a specific focus on fleet-leader-based virtual monitoring application. This benchmark aims to assess these methods in terms of generalization ability and robustness of uncertainty indications. In addition, the computational requirements for the training and deployment of the models are also compared.

3.1 Uncertainty estimation in probabilistic neural networks

The collected predictions from multiple passes of the probabilistic neural networks form a distribution of possible outcomes. One can calculate various statistics of this distribution from the predicted samples. For instance, the expected value and the predictive uncertainty can be quantified as:

$$\mathbb{E}[\hat{y} \mid \boldsymbol{x}] = \frac{1}{N_f} \sum_{i=1}^{N_f} f\left(\hat{y} \mid \boldsymbol{x}, \boldsymbol{\theta}^{(i)}\right), \tag{8}$$

$$\mathbb{V}(\hat{y} \mid \boldsymbol{x}) = \frac{1}{N_f} \sum_{i=1}^{N_f} \left(f\left(\hat{y} \mid \boldsymbol{x}, \boldsymbol{\theta}^{(i)}\right)\right)^2 - \left(\mathbb{E}[\hat{y} \mid \boldsymbol{x}]\right)^2, \tag{9}$$

where the network parameters, $\boldsymbol{\theta}$ are random samples from the posterior weights and biases associated distributions, f represents the probabilistic neural network model, and N_f is the number of forward predictions.

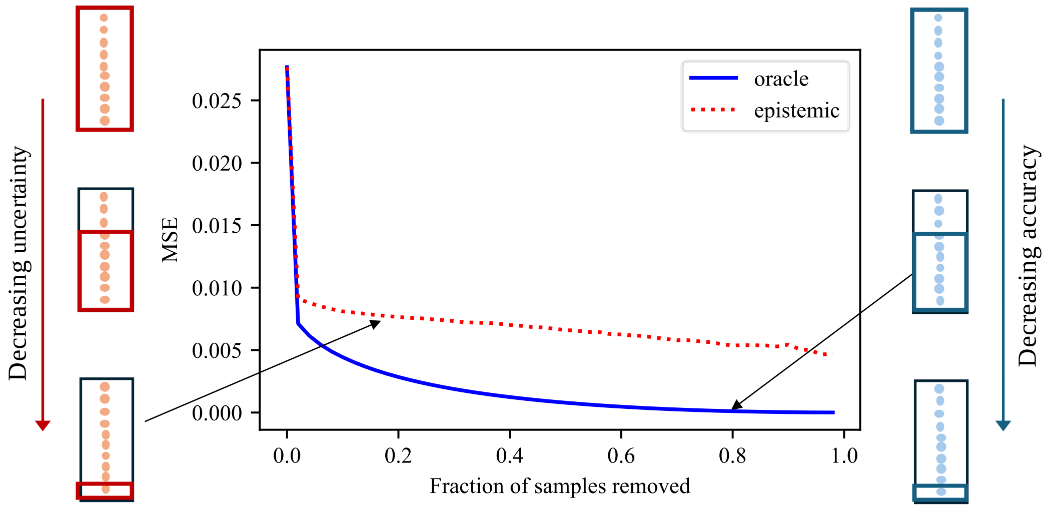


Figure 1: Illustration of an uncertainty-error sparsification plot.

If the output of neural network models is a distribution, i.e., aleatory uncertainty is also considered, the retrieved predictive uncertainty estimate $\mathbb{V}(\hat{y} \mid \boldsymbol{x})$ encompasses both aleatory and epistemic contributions. If the network is modeled to produce point estimates only, the predictive uncertainty solely represents the epistemic part. For a detailed explanation on the uncertainty decomposition of Bayesian neural networks, the reader is referred to [Hlaing et al., 2024].

3.2 Metrics for comparing probabilistic models

To compare the model uncertainty provided by the tested probabilistic neural networks, we evaluate the coverage probability, represent sparsification plots, and quantify the AUSE (area under sparsification error) metric. In order to compute the coverage probability, multiple forward simulations are generated for each test input. From the simulated predictions, we estimate the 2.5^{th} and 97.5^{th} percentiles and check if the target value lies within the interval. The coverage probability on the test dataset can thus be computed as:

$$CP = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}\left[\hat{y}_i^{(2.5)} \leq y_i \leq \hat{y}_i^{(97.5)}\right] \tag{10}$$

where $\hat{y}_i^{(2.5)}$ and $\hat{y}_i^{(97.5)}$ are the $2.5^{th}$and 97.5^{th} quantiles of the predicted samples for observation i, N is the total number of test observations, and $\mathbb{I}[.]$ is an indicator function that returns 1 if the condition inside is true and 0 otherwise.

Sparsification plots have already been used in computer vision to visualize and compare different probabilistic neural network models with respect to their performance in estimation tasks while considering uncertainty estimates [Gustafsson et al., 2020, Ilg et al., 2018]. The vertical axis in a sparsification plot is the metric of accuracy, e.g., mean absolute error, mean squared error, etc., and the percentage of removed samples is indicated on the horizontal axis. Sparsification plots are obtained through step-by-step removal of test samples and re-assessing the model accuracy, as illustrated in Figure 1. The *oracle* curve which eliminates the samples in a decreasing order of the model accuracy is always monotonic and used as the reference. The *epistemic* curve plots the model accuracy of test samples sorted from high to low uncertainty. Note that when the model uncertainty is biased- e.g., over-confidence for wrong predictions, the epistemic curve can be non-monotonic. The closer the epistemic curve aligns with the oracle curve, the better the model's uncertainty ranking.

Subsequently, AUSE - the area between the oracle and epistemic curves - is a performance metric used to quantitatively evaluate the quality of uncertainty estimation. A lower AUSE value indicates that the model-provided uncertainties align with the error with respect to ground truths. For non-monotonic epistemic curves, AUSE will increase, indicating inconsistency between accuracy and uncertainty.

3.3 Experimental setup

Monitoring setup and data description

This case study uses the data collected form five monopile-supported offshore wind turbines collected during a 2-year period (from late 2022 to 2024). The monitoring setup consists of a SCADA system which records turbine's environmental and operational conditions, nacelle accelerometers and tri-axial strain gauges on the transition piece specifically located at 14 m above the water level. The signals collected from SCADA and accelerometers are stored in the form of statistics at every 10-minute - mean, standard deviation, minimum, maximum, and root mean square (rms), etc.

In common practice, the temporal strain signals are often post-processed into bi-axial damage equivalent moments, DEM_{tl} in side-to-side (SS) direction and DEM_{tn} fore-aft (FA) directions, as follows:

$$DEM = \left(\frac{\sum_{i=1}^{N_m} n_i \cdot M_{r,i}^m}{N_{eq}} \right)^{1/m}. \tag{11}$$

The term damage equivalent load (or moment) is defined such that the damage caused by a predefined number of DEL range N_{eq} (usually specified as 10^7) is equal to that of the original load spectrum. N_m represents the number of bins in the load spectrum, $M_{r,i}$ and n_i are the reference moment value of the i^{th} bin and the corresponding number of cycles and 'm' is the slope of the linear SN curve. In this work, we train the model to predict the moment in SS direction DEM_{tl}.

Table 1 lists the input and output features of the virtual monitoring model. Although all SCADA and acceleration statistics are recorded, some statistical features are not used as model inputs depending on their influence on the output load (based on pre-conducted feature importance analysis), and data availability constraints. The collected data that spans 2 years is split into one year each for training and testing. The test data is further decomposed into two different sets: the samples that are similar to the training data (Test 1) and the others outside the distribution of the training data (Test 2). In this regard, the test samples are put in the second test dataset if one input feature is beyond 3 standard deviations in its normalized space of the training data. In particular, 35% of the 1-year test data is Test 1 and the remainder forms Test 2.

Neural network parameters

The probabilistic neural networks used in this investigation are implemented in PyTorch, a Python-supported tensor library for deep learning. Table 2 summarizes the network configurations and training parameters for all the methods. The same network architecture is used for all neural network models with three hidden layers with 32, 64, and 32 neurons respectively,

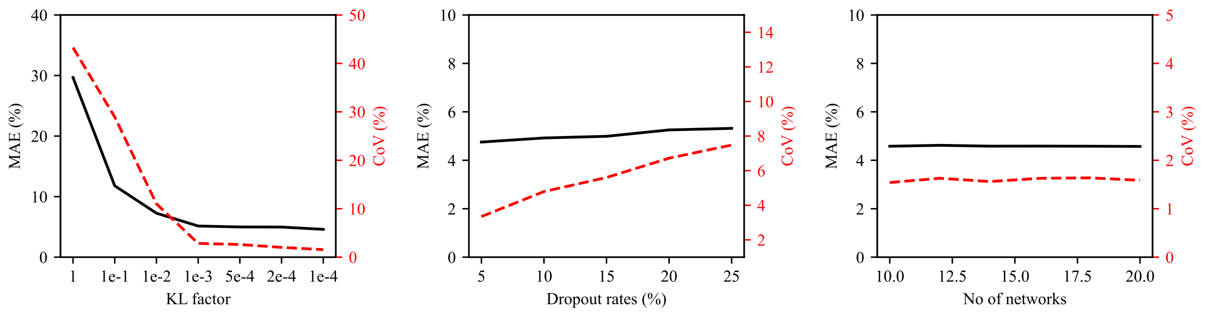
Table 1: Description of the input and output features.

	Sensor	Monitoring signal	Symbol	Units
Input	SCADA	Wind speed (mean)	$\mu[WSpd]$	m/s
		Wind speed (std)	$\sigma[WSpd]$	m/s
		Power (mean)	$\mu[Power]$	kW
		Yaw angle (mean)	$\mu[Yaw]$	deg
		Pitch angle (mean)	$\mu[Pitch]$	deg
		Pitch angle (std)	$\sigma[Pitch]$	deg
		Rotational speed (mean)	$\mu[RPM]$	rpm
		Rotational speed (std)	$\sigma[RPM]$	rpm
		Wind direction (mean)	$\mu[WDir]$	deg
		Turbulence intensity (mean)	$\mu[Turb]$	-
	Nacelle accelerometers	Z acceleration (mean)	$\mu[acc_Z]$	g
		Z acceleration (max)	$max[acc_Z]$	g
		Z acceleration (min)	$min[acc_Z]$	g
		Z acceleration (rms)	$rms[acc_Z]$	g
		FA acceleration (max)	$max[acc_{FA}]$	g
		FA acceleration (min)	$min[acc_{FA}]$	g
		FA acceleration (rms)	$rms[acc_{FA}]$	g
		SS acceleration (max)	$max[acc_{SS}]$	g
		SS acceleration (min)	$min[acc_{SS}]$	g
		SS acceleration (rms)	$rms[acc_{SS}]$	g
Output	Strain gauges	DEM (side-to-side)	DEM_{tl}	MNm

Table 2: Neural network architecture and training parameters.

	Bayes by backprop	MC-Dropout	Deep ensembles
No of neurons:			
Input layer	20	20	20
Hidden layers	32, 64, 32	32, 64, 32	32, 64, 32
Output layer	1	1	1
Optimizer	Adam	Adam	Adam
Learning rate	0.001	0.001	0.001
Batch size	2400	2400	2400
No of episodes	1000	1000	100
*KL loss factor	1-1e-4	-	-
*Dropout probability	-	0.1-0.25	-
*No of networks	-	-	10-20

* A sensitivity analysis is performed to analyze the hyper-parameters.



(a) Bayes by backprop (b) MC-Dropout (c) Deep ensembles

Figure 2: Sensitivity analysis. The influence of key hyperparameters is assessed for all tested probabilistic neural networks.

activated by rectified linear unit (*ReLU*) functions. The input layer receives 20 statistical features of SCADA and acceleration parameters and the output layer predicts DEM_{tl}. We do not implicitly model the aleatory uncertainty in this case study, and therefore the output layer has only one neuron. However, the BNN can also be alternatively laid out with additional neurons in the output layer to capture aleatory uncertainty [Hlaing et al., 2024].

During the course of the training task, the neural networks are trained on mini-batches of 2400 training samples, minimizing the respective formulated loss functions. The networks' weights are adjusted according to *Adam* optimizer at the learning rate of 0.001. Bayes by backprop and MC-Dropout are trained for 1000 episodes and the best weights with minimum loss are stored. In deep ensembles, each deterministic network is trained for only 100 epochs to avoid potential overfitting problems. Since each implemented inference method consists of different hyper-parameters to optimize, we also performed sensitivity analyses on those parameters. The hyper-parameters and tested values are also shown in Table 2.

The virtual sensing models trained by changing the values of hyperparameters are tested on the same test dataset to study their effects on the accuracy (MAE) and uncertainty estimation (CoV), computed as:

$$MAE = \frac{1}{N}\sum_{i=1}^{N} \mid \hat{y}_i - y_i \mid, \tag{12}$$

$$CoV = \frac{1}{N}\sum_{i=1}^{N} \frac{\sqrt{\mathbb{V}(\hat{y}_i \mid \boldsymbol{x})}}{\mathbb{E}[\hat{y}_i \mid \boldsymbol{x}]}. \tag{13}$$

The test dataset used in the sensitivity analysis only consists of the samples that are inside the training region. In Bayes by backprop, the weight factor of the KL loss α is varied between 1 and 1e-4. With higher weights of the KL loss, the mean absolute error (MAE) and uncertainty tend to increase, as shown in Figure 2a. With high KL factor, the posterior weights remain closer to the initial standard Gaussian weights and the model's fitting capacity to the training data is limited, thus affecting the prediction accuracy. The MAE and model uncertainty converges for KL factors less than 1e-3.

The dropout rates in the MC-dropout do not have a significant effect on the averaged mean absolute error. However, the epistemic uncertainty changes by varying the dropout rates such that it increases for higher dropout rates. However, smaller uncertainties might not necessarily mean that the model predictions are better, as it can be seen in Figure 2b that the MAE remains the same. With small values of dropout rates, the model weights just becomes almost deterministic, thus showing smaller CoVs.

Deep ensembles of multiple deterministic networks are tested - varying the number of models from 10 to 20. Neither MAE nor COV is dependent on the number of models, with consistent values of $\approx 4.5\%$ of accuracy and ≈ 1.5 for CoV in Figure 2c. Based on the sensitivity results, the models trained by Bayes by backprop with KL loss factor 1e-4, MC-Dropout with dropout probability 10%, and deep ensembles of 20 neural networks are selected for the benchmark study.

4 RESULTS AND DISCUSSION

4.1 Comparative study

The goal of this comparative study is to test popular BNN variants in terms of their ability to generalize and represent uncertainty. Table 3 lists the mean absolute error (MAE) and area under the sparsification error (AUSE) curve on the two test datasets. All models achieve an MAE of approximately 4.5-5% on the first dataset (i.e., Test 1), where the samples are closer to the training data distribution. Additionally, the models are tested on out-of-distribution data (i.e., Test 2) to assess their generalization capabilities. As expected, the MAE for all models increases significantly on Test 2 compared to Test 1. Interestingly, Bayes by backprop outperforms the other probabilistic neural network variants, resulting in an MAE of 10%, whereas MC-Dropout and deep ensembles result in MAEs of 24% and 29%, respectively.

To analyze the ability of the models to rank predictions in terms of model uncertainty, the sparsification curves are shown in Figure 3 for Test 1, Test 2, and the combined dataset (i.e., Test 1+2). The sparsification curves are mostly monotonic for Test 1, showing an overall correspondence between model error and uncertainty ranking. The sparsification curves do not, however, fully align with their corresponding oracle curves, and the MAE only slightly reduces, although the model becomes more confident. We attribute this result to the fact that prediction errors are mostly small in this case; the model will not give significantly higher model uncertainties, resulting in flat epistemic curves. This behavior can also be expected considering that test samples are close to the training dataset, and the model is naturally confident in the predictions despite possible deviations from ground-truth labels.

It is worth noting that the epistemic sparsification curve of MC-Dropout is not monotonic on Test 2, which can be caused by incorrect uncertainty estimates. On the other hand, despite high prediction errors, deep ensemble models' oracle and epistemic curves almost fully align with each other, indicating the model's strong uncertainty awareness. The epistemic curve of Bayes by backprop rapidly decreases as the sparsity increases, yet it does not completely align with the oracle.

Table 3: Performance of the tested probabilistic neural networks on two test datasets. Note that samples contained in *Test 2* dataset are farther to the training data distribution than samples included in *Test 1*.

	Test 1		Test 2	
	MAE (%)	AUSE	MAE (%)	AUSE
Bayes by backprop	4.59	0.41	10.19	0.33
MC-Dropout	4.92	0.47	24.2	0.24
Deep ensembles	4.58	0.37	29.36	0.11

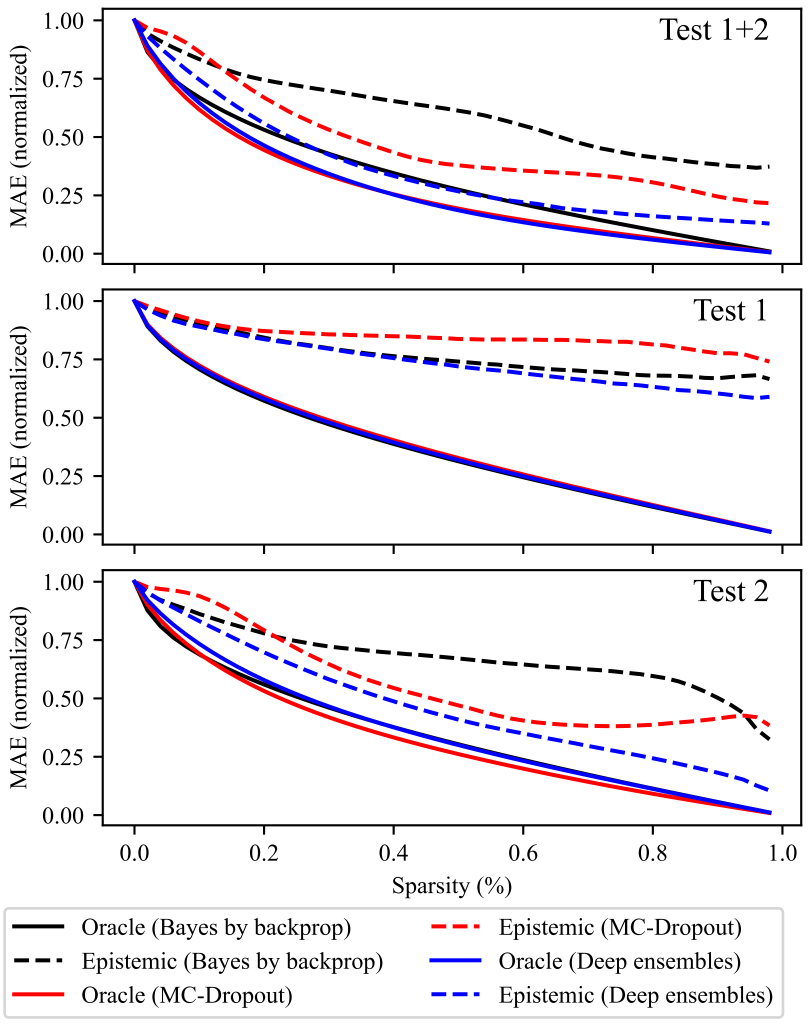


Figure 3: Sparsification curves for the investigated probabilistic models. Samples contained in *Test 2* dataset are farther to the training data distribution than samples included in *Test 1*.

By observing the sparsification curves of Test 1+2, it can be seen that the epistemic curves of Test 1+2 up to ∼70% sparsity also correspond to those of Test 2, and the remaining tails are similar to Test 1. It demonstrates that the models themselves can distinguish between similar data and shifted data. Especially, Bayes by backprop and deep ensembles are able to provide robust uncertainty estimates for out-of-distribution data. Areas under the sparsification error are also listed in Table 3. As one can expect, the AUSE values on Test 1 are similar among all the models. In Test 2, the minimum AUSE of deep ensembles represents the afore-mentioned accuracy-uncertainty correspondence.

To further assess the quality of uncertainty indicators, we computed the coverage probability — a frequentist measure of uncertainty representativeness— which checks if the ground-truth values lie within the credible intervals predicted by the models. The 95% intervals are computed and described as a ratio with respect to the predicted values. The CPs are computed for test samples of which the interval width to mean ratio is higher than the indicated threshold. The CPs of the tested approaches are plotted in Figure 4 for combined Test 1+2. In Bayes by

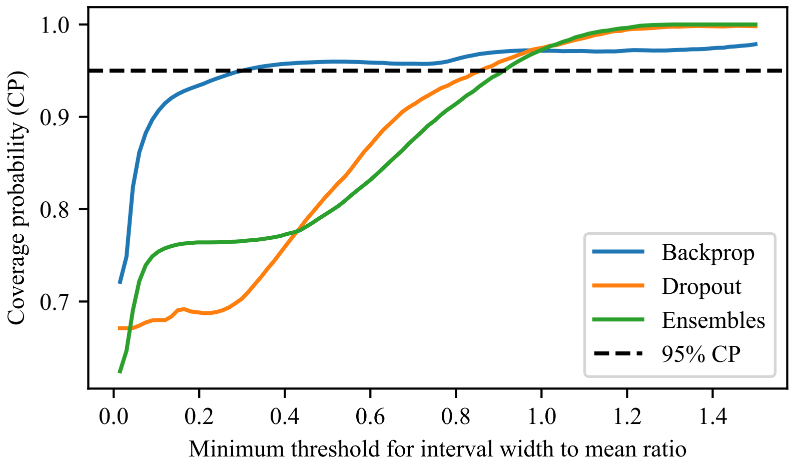


Figure 4: The 95% coverage probability (CP) of the investigated probabilistic neural networks is represented as a function of the specified threshold. CP is computed by selecting test samples for which the predicted interval width to mean ratio is higher than the indicated threshold.

backprop, 95% coverage probability is achieved when the interval width to mean ratio goes higher than 20%. For the other test samples, i.e., the interval width to mean ratio is less than 0.2, the BNN model is confident but the target value is out of the predicted interval. However, it does not necessarily mean that the prediction error, e.g., mean absolute error, will be high. The other two models, MC-Dropout and deep ensembles achieve the 95% CP only when the interval width is higher than 80% of the mean value, meaning that the model is, in general, over-confident for most of the test samples. If one would like to deploy these models in the decision-making problem, re-calibration of the uncertainty bounds should be considered for safe application.

4.2 Computational requirements

We summarize in Table 4 the computational requirements for training and deployment of the investigated approaches on an 8-core processor operating at a clock speed of 3.2 GHz and 8 GB memory. For Bayes by backprop and MC-Dropout models, the training time is mainly influenced by the model complexity, i.e., the number of neurons, and the number of training samples. The indicated computational requirements are for training the dataset consisting of 219,820 samples. The training time of deep ensembles further depends on the number of neural networks. In the above case study, we use 20 models and require around 2300 seconds of training time, making it the most computationally expensive.

The computational efforts for prediction are indicated for 237,305 test points. One needs to simulate multiple forward passes to compute the predicted value and the model uncertainty. Whereas the number of forward simulations required is case-specific, we run 100 passes of BNNs (Bayes by backprop and MC-Dropout) in this work. Bayes by backprop is more efficient, only requiring 2.7 seconds, while MC-Dropout takes 15.7 seconds. In deep ensembles, it is not necessary or impossible to perform a large number of forward passes as in Bayes by backprop and MC-Dropout, and the number of forward simulations is just the same as the number of neural networks, making it the most efficient during the prediction phase.

Table 4: Computational requirements for training and deployment of the probabilistic models.

	Training	Prediction
Bayes by backprop	$\sim$1100 s	$\sim$2.7 s (100 simulations)
MC-Dropout	$\sim$1200 s	$\sim$15.7 s (100 simulations)
Deep ensembles	$\sim$2300 s	$\sim$0.5 s (20 simulations)

5 CONCLUSIONS

This study assesses the effectiveness of probabilistic deep learning models in serving as virtual load sensors for offshore wind turbines. The results show that the tested probabilistic models can generally signal uncertain samples, which are most likely characterized by high prediction errors, resulting in monotonic uncertainty-error sparsification curves. Among the approaches investigated, deep ensembles lead to a better agreement between error and uncertainty ranking compared to the other tested Bayesian neural networks, which can yield over-confident uncertainty estimates when receiving out-of-distribution data. In terms of probability calibration, the results reveal that all tested models tend to be over-confident when generating predictions for samples close to the training data distribution, whereas more reliable uncertainty metrics can be obtained when the models are presented with out-of-distribution data. Overall, probabilistic models are able to accurately predict loads under conditions similar to the training data while signaling high uncertainty when making predictions from data collected under different conditions, thereby offering a useful tool for managing the risk of offshore wind assets.

While a set of probabilistic deep learning models is investigated in terms of generalization, uncertainty rankings, and probability calibration, this research can be extended by exploring inference methods, such as Markov Chain Monte Carlo, to quantify the discrepancy between the computed approximate and exact posterior distributions over neural network parameters.

ACKNOWLEDGMENTS

This research is funded by the Belgian Energy Transition Fund (FPS Economy) through the SmartLife project, and the European Union (GA No. 1011122184) through the WILLOW project. However, views and opinions expressed do not necessarily reflect those of the funders. None of the granting authorities can be held responsible for them.

DATA AVAILABILITY

Due to its proprietary nature and confidentiality concerns, supporting data of this research cannot be made publicly available.

REFERENCES

[Blundell et al., 2015] Blundell, C., Cornebise, J., Kavukcuoglu, K., and Wierstra, D. (2015). Weight uncertainty in neural network. In *International conference on machine learning*, pages 1613–1622. PMLR.

[Cho et al., 2021] Cho, S., Choi, M., Gao, Z., and Moan, T. (2021). Fault detection and diagnosis of a blade pitch system in a floating wind turbine based on kalman filters and artificial neural networks. *Renewable Energy*, 169:1–13.

[Gal and Ghahramani, 2016] Gal, Y. and Ghahramani, Z. (2016). Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR.

[Graves, 2011] Graves, A. (2011). Practical variational inference for neural networks. In *Advances in Neural Information Processing Systems*, volume 24, pages 2348–2356.

[Gustafsson et al., 2020] Gustafsson, F. K., Danelljan, M., and Schon, T. B. (2020). Evaluating scalable bayesian deep learning methods for robust computer vision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 318–319.

[Hlaing, 2024] Hlaing, N. (2024). *Data-driven virtual monitoring and life-cycle management of offshore wind support structures*. PhD thesis, Universite de Liege (Belgium).

[Hlaing et al., 2024] Hlaing, N., Morato, P. G., de Nolasco Santos, F., Weijtjens, W., Devriendt, C., and Rigo, P. (2024). Farm-wide virtual load monitoring for offshore wind structures via bayesian neural networks. *Structural Health Monitoring*, 23(3):1641–1663.

[Hlaing et al., 2022] Hlaing, N., Morato, P. G., and Rigo, P. (2022). Probabilistic virtual load monitoring of offshore wind substructures: A supervised learning approach. In *The Proceedings of the 32nd International Ocean and Polar Engineering Conference, ISOPE-2022*, volume 4, pages 3137–3144.

[Ilg et al., 2018] Ilg, E., Cicek, O., Galesso, S., Klein, A., Makansi, O., Hutter, F., and Brox, T. (2018). Uncertainty estimates and multi-hypotheses networks for optical flow. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 652–667.

[Mbuvha et al., 2021] Mbuvha, R., Mongwe, W. T., and Marwala, T. (2021). Separable shadow Hamiltonian hybrid Monte Carlo for Bayesian neural network inference in wind speed forecasting. *Energy and AI*, 6:100–108.

[Osawa et al., 2019] Osawa, K., Swaroop, S., Khan, M. E. E., Jain, A., Eschenhagen, R., Turner, R. E., and Yokota, R. (2019). Practical deep learning with Bayesian principles. In *Advances in Neural Information Processing Systems*, volume 32.

[Pimenta et al., 2022] Pimenta, F., Pacheco, J., Pereira, S., and Magalhães, F. (2022). Reconstructing the bending moments time history of wind turbine tower from acceleration measurements using Gaussian processes. *Journal of Physics: Conference Series*, 2265(3):032080.

[Singh et al., 2024] Singh, D., Dwight, R., and Viré, A. (2024). Probabilistic surrogate modeling of damage equivalent loads on onshore and offshore wind turbines using mixture density networks. *Wind Energy Science Discussions*, 2024:1–28.

[Wager et al., 2013] Wager, S., Wang, S., and Liang, P. S. (2013). Dropout training as adaptive regularization. In Burges, C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K., editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

[Welling and Teh, 2011] Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer.

[Zhang et al., 2019] Zhang, R., Li, C., Zhang, J., Chen, C., and Wilson, A. G. (2019). Cyclical stochastic gradient mcmc for bayesian deep learning. *arXiv preprint arXiv:1902.03932*.

[Zou et al., 2023] Zou, J., Cicirello, A., Iliopoulos, A., and Lourens, E.-M. (2023). Gaussian process latent force models for virtual sensing in a monopile-based offshore wind turbine. In *EWSHM 2022: European Workshop on Structural Health Monitoring.*, pages 290–298. Springer International Publishing.