

Multi-task neural diffusion processes for uncertainty-quantified wind power prediction

Joseph Rawson^a, Domniki Ladopoulou^a, Petros Dellaportas^{a,b}

^a*Department of Statistical Science, University College London, Gower Street, London, WC1E 6BT, United Kingdom*

^b*Department of Statistics, Athens University of Economics and Business, 28is Oktovriou 76, Athens, 104 34, Greece*

Abstract

Uncertainty-aware wind power prediction is essential for grid integration and reliable wind farm operation. We apply neural diffusion processes (NDPs)—a recent class of models that learn distributions over functions—and extend them to a multi-task NDP (MT-NDP) framework for wind power prediction. We provide the first empirical evaluation of NDPs in real supervisory control and data acquisition (SCADA) data. We introduce a task encoder within MT-NDPs to capture cross-turbine correlations and enable few-shot adaptation to unseen turbines. The proposed MT-NDP framework outperforms single-task NDPs and GPs in terms of point accuracy and calibration, particularly for wind turbines whose behaviour deviates from the fleet average. In general, NDP-based models deliver calibrated and scalable predictions suitable for operational deployment, offering sharper, yet trustworthy, predictive intervals that can support dispatch and maintenance decisions in modern wind farms.

Keywords:

Few-shot inference, Probabilistic predictions, Renewable energy, SCADA data

1. Introduction

Wind energy has become a cornerstone of the global transition to clean power. As wind power capacity expands worldwide, ensuring reliability and minimising downtime are critical to both energy security and the financial viability of wind farms. Uncertainty-aware and accurate prediction of wind

power output is vital, as it supports grid stability, market participation, and proactive maintenance planning. Beyond energy balancing, uncertainty-aware forecasting also reduces operational uncertainty for wind farm operators, enabling more efficient maintenance scheduling and reducing costly unplanned downtime. This is especially important given that operation and maintenance costs represent a significant share of total expenditure, with unexpected failures making up the largest component [1, 2].

Supervisory control and data acquisition (SCADA) systems provide a low-cost and widely available source of wind turbine data. They capture environmental and operational variables with high frequency, making them invaluable for prediction applications. However, their use is complicated by measurement noise, turbine downtime, and limited public availability [3, 4]. A practical prediction model for SCADA data must therefore satisfy several requirements: (i) scalability to large SCADA datasets covering multiple years of wind farm operation, (ii) the ability to exploit the high-dimensional feature space recorded in SCADA systems, (iii) probabilistic power forecasts with calibrated predictive densities rather than only point estimates, and (iv) the ability to generalise across turbines with limited historical records.

Several modelling approaches have been proposed to meet these requirements. Quantile regression methods generate predictive intervals directly and have been applied in spatio-temporal and deep learning settings for wind power forecasting [5, 6, 7]. Mixture density networks have been proposed to represent multimodal output distributions [8], while generative moment matching networks provide distribution-free probabilistic scenarios without assuming specific functional forms [9]. These methods enhance flexibility in modelling uncertainty, but generally lack mechanisms for transfer learning across wind turbines and can be computationally demanding when scaled to large SCADA datasets.

Gaussian processes (GPs) have been widely applied in wind power forecasting due to their ability to provide calibrated predictive distributions [10]. For example, heteroscedastic GP models have been used to capture input-dependent variance [11]. Other studies have incorporated turbine operational variables to improve forecast accuracy [12]. GPs have also been combined with numerical weather prediction outputs to enhance predictive performance [13]. Despite these strengths, the cubic scaling of kernel computations ($O(n^3)$ in the number of data points, n) makes GPs computationally expensive on large SCADA datasets, and their strict Gaussian assumptions often limit performance in highly non-linear regimes [14].

Neural network (NN)-based approaches, such as probabilistic multi-layer perceptrons (MLPs), provide scalability and enable transfer learning across turbines, but they still model predictive uncertainty by explicitly predicting the mean and variance under a Gaussianity assumption [15]. Neural processes (NPs) combine the predictive power of NNs with the uncertainty estimation of GPs using an encoder–decoder architecture [16]. However, they often suffer from underfitting and limited consistency between context and target distributions [17]. Extensions such as multi-task neural processes demonstrate the potential to capture correlations across tasks [18], a particularly relevant issue for wind power prediction, since wind turbines on the same wind farm are typically exposed to similar environmental conditions.

Recent advances in generative modelling have introduced diffusion models, which iteratively corrupt and denoise data to learn complex probability distributions [19, 20]. These are state-of-the-art models in the domain of image generation [21]. Neural diffusion processes (NDPs) extend this framework to distributions over functions, avoiding Gaussianity assumptions while retaining scalability through the neural architecture. NDPs have been shown to emulate GPs well on synthetic data and outperform NPs [22]; however, their applications to real-world datasets remain unexplored in the regression setting.

We are the first to apply NDPs to SCADA data and to develop the methodology that allows adaptation to the multi-task setting. Specifically, we develop a novel multi-task NDP (MT-NDP) framework that incorporates a task encoder to capture correlations across turbines, enabling knowledge transfer and improved predictive performance when data for individual wind turbines are limited. The key advances of this paper are: (i) the first empirical evaluation of NDPs on operational SCADA data, showing accuracy exceeding that of GPs while delivering better-calibrated uncertainty estimates and scalability to higher-dimensional inputs; (ii) the extension of NDPs to a multi-task setting via a task encoder, which leverages cross-turbine correlations and consistently outperforms single-task variants; (iii) demonstration of the ability of MT-NDPs to perform few-shot generalisation, adapting to unseen turbines with only a small number of context points; and (iv) empirical evidence of scalability and robustness, with predictions that remain uncertainty-aware and suitable for decision-making in real-world wind farm operations.

The remainder of the paper is organised as follows. Section 2 reviews the benchmark models and introduces the background for NDPs. Section 3

describes the SCADA wind farm data. Section 4 presents the NDP framework and our multi-task extension (MT-NDP). Section 5 outlines the experimental design, and Section 6 reports the main findings. Finally, Sections 7 and 8 provide discussion and conclusions.

2. Preliminaries

In this section, we present the foundational models that support our framework or serve as benchmarks for evaluating the proposed multi-task extension. Section 2.1 covers GPs, Section 2.2 reviews diffusion probabilistic models, and Section 2.3 describes NDPs and provides the theoretical basis for the multi-task extension introduced in Section 4.

2.1. Gaussian Processes (GPs)

A GP $f : \mathbb{R}^D \rightarrow \mathbb{R}$ is a stochastic process such that, for any finite collection of inputs $X = \{x_i\}_{i=1}^N \subset \mathbb{R}^D$, the vector of function evaluations $f(X) = [f(x_1), \dots, f(x_N)]^T$ follows a multivariate normal distribution [10]. GPs satisfy the Kolmogorov extension theorem, which guarantees that all finite-dimensional marginals are mutually consistent under permutation and marginalisation [23]. This provides a principled Bayesian framework for modelling distributions over functions, enabling exact inference with predictive uncertainty. In our experiments presented in Section 6, we use GP regression as a benchmark model, so we briefly outline its formulation here.

In the regression setting, we are given a data vector $y = \{y_i\}_{i=1}^N \in \mathbb{R}^N$ whose entries are noisy evaluations of some function $f(\cdot)$ on a collection of D -dimensional vectors $X = \{x_i\}_{i=1}^N \in \mathbb{R}^{N \times D}$. Each y_i is assumed to be a noisy observation of $f(x_i)$, with independent Gaussian noise of mean 0 and variance σ^2 . Placing a GP prior over $f(\cdot)$, with mean function $\mu(\cdot)$ and covariance kernel $k_\theta(\cdot, \cdot)$, gives the joint distribution

$$f(X) \sim \mathcal{N}(\mu(X), K(X, X)), \quad (1)$$

where $\mu(X) = [\mu(x_1), \dots, \mu(x_N)]^T$ and $K(X, X)_{ij} = k_\theta(x_i, x_j)$ for all i, j .

The predictive distribution for future observations y^* with covariates X^* is Gaussian with mean and variance

$$E(y^*|y) = \mu(X^*) + K(X^*, X)A^{-1}(y - \mu(X)), \quad (2)$$

$$V(y^*|y) = K(X^*, X^*) - K(X^*, X)A^{-1}K(X, X^*). \quad (3)$$

For our experiments, we use the radial basis function (RBF) kernel:

$$k_{\text{RBF}}(x_i, x_j | \theta) = \sigma_f^2 \exp \left(-\frac{1}{2} \sum_{d=1}^D \frac{(x_{i,d} - x_{j,d})^2}{\ell_d^2} \right), \quad (4)$$

where $\theta = \{\sigma_f^2, \ell_1, \ell_2, \dots, \ell_D\}$ are the kernel hyperparameters, with ℓ_d denoting the length-scale parameters and σ_f^2 representing the output variance.

While the RBF kernel is widely used and often effective in practice, encoding realistic prior assumptions through kernel design remains challenging, particularly for larger datasets and in higher-dimensional spaces. In addition, GPs assume Gaussian finite-dimensional marginals, restricting the class of functions they can represent. These issues motivate the exploration of alternative stochastic process models that relax Gaussianity while retaining uncertainty quantification and other desirable characteristics such as exchangeability.

2.2. Diffusion probabilistic models (DPMs)

DPMs were introduced as a class of generative models inspired by non-equilibrium thermodynamics [19]. The central idea is to define a forward process, in which the structure of the data distribution is gradually destroyed by the iterative addition of Gaussian noise, and a corresponding reverse process that learns to invert this corruption.

The de-noising diffusion probabilistic model (DDPM) [20] simplified and stabilised training and has become the standard formulation. Following [20, 22], the forward process begins from the input data distribution $q(s_0)$, where $s_0 \in \mathbb{R}^D$ denotes a D -dimensional data vector. It defines a fixed Markov chain over the sequence $s_{0:T} = \{s_0, \dots, s_T\}$ with density

$$q(s_{0:T}) = q(s_0) \prod_{t=1}^T q(s_t | s_{t-1}) \quad (5)$$

and conditional distributions

$$q(s_t | s_{t-1}) = \mathcal{N}\left(\sqrt{1 - \beta_t} s_{t-1}, \beta_t I_D\right), \quad (6)$$

where $\{\beta_t \in (0, 1)\}_{t=1}^T$ is a variance schedule controlling the magnitude of injected noise and I_D is the $D \times D$ identity matrix. After T steps, the distribution approaches Gaussian noise, $q(s_T) \approx \mathcal{N}(0, I)$. Early implementations

used a linear variance schedule, while later work demonstrated that a cosine schedule improves performance by distributing noise more evenly across diffusion steps [24]. A variance schedule defines the cumulative noise parameter

$$\bar{\alpha}_t = \prod_{j=1}^t (1 - \beta_j), \quad (7)$$

which quantifies the retained signal at step t . In the cosine schedule,

$$\bar{\alpha}_t = \frac{f(t/T)}{f(0)}, \quad f(\lambda) = \cos^2\left(\frac{\pi}{2} \frac{\lambda+s}{1+s}\right), \quad (8)$$

where T is the total number of steps and s a small offset. This design avoids excessive noise at the beginning and end of the process, yielding more stable training and improved sample quality.

The reverse process is intractable in closed form, but can be approximated by an NN that predicts the injected noise. The mean of the reverse transition is parameterised as

$$\mu_\theta(s_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(s_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(s_t, t) \right), \quad (9)$$

where $\alpha_t = 1 - \beta_t$, $\bar{\alpha}_t = \prod_{j=1}^t \alpha_j$, and $\epsilon_\theta : \mathbb{R}^D \times \{1, \dots, T\} \rightarrow \mathbb{R}^D$ is a neural network that predicts the injected noise vector. Training is carried out by minimising the score-matching loss

$$\mathbb{E}_{t, s_0, \epsilon} [\|\epsilon - \epsilon_\theta(s_t, t)\|^2], \quad (10)$$

where $s_t = \sqrt{\bar{\alpha}_t} s_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon$ and $\epsilon \sim \mathcal{N}(0, I)$. Since $q(s_T) \approx \mathcal{N}(0, I)$, generating new samples requires only running the learned reverse process starting from Gaussian noise.

This framework enabled diffusion models to achieve state-of-the-art generative performance, rivalling generative adversarial networks (GANs) and autoregressive models. Building on DDPMs, Repaint [21] adapted the approach to image inpainting by conditioning on context points during the reverse process, without altering the pre-trained DDPM. This method achieved high-quality and diverse reconstructions under challenging conditions, outperforming autoregressive and GAN-based baselines.

In parallel, NPs were proposed as a stochastic process model combining neural networks with an encoder–decoder architecture [16]. NPs learn to map

a set of context points into a latent task representation, which is then used by a decoder to generate predictions at target inputs. This provides scalability and the ability to adapt to new tasks with only a few observations, but standard NPs often underfit and produce poorly calibrated uncertainty [17]. Latent and attentive variants improve correlations between targets, but still fall short of fully consistent Bayesian behaviour [22].

Neural diffusion processes (NDPs) unify these two developments, drawing on diffusion-based generative modelling to overcome the Gaussian assumptions of NPs while retaining their scalability and adaptability. NDPs extend diffusion models from distributions over data to distributions over functions [22], thereby enabling uncertainty-aware regression.

2.3. Neural diffusion processes (NDPs)

The core idea of NDPs is to replace the fixed data vector s_0 in DDPMs with function evaluations (X, y) . Here $X \in \mathbb{R}^{N \times D}$ is the matrix of input locations and $y \in \mathbb{R}^N$ the corresponding outputs. In the forward process, Gaussian noise is gradually added to the function values y , while the inputs X remain fixed. After T steps, the corrupted outputs resemble white noise. The reverse process is parameterised by an NN that predicts the noise injected at each step, thereby enabling recovery of the original function values. Training minimises a denoising score-matching objective, analogous to that of DDPMs, as described in Section 4.1.3.

Compared to GPs, NDPs avoid restrictive Gaussianity assumptions by learning arbitrary distributions over functions. Compared to NPs, they offer better-calibrated predictive distributions and a more consistent treatment of context and target points. On synthetic benchmarks, NDPs have been shown to emulate GP behaviour while scaling more effectively through neural architectures. In Section 4, we provide a detailed description of NDPs, together with illustrations of the forward and reverse processes, and introduce our proposed multi-task extension (MT-NDP).

3. Dataset description

We use SCADA data from the six Senvion MM92 wind turbines at Kelmarsh wind farm in the UK [25]. The dataset spans January 2016 to July 2021 at a ten-minute resolution, comprising more than 1.7 million records across 110 variables, including wind speed, wind direction, power output,

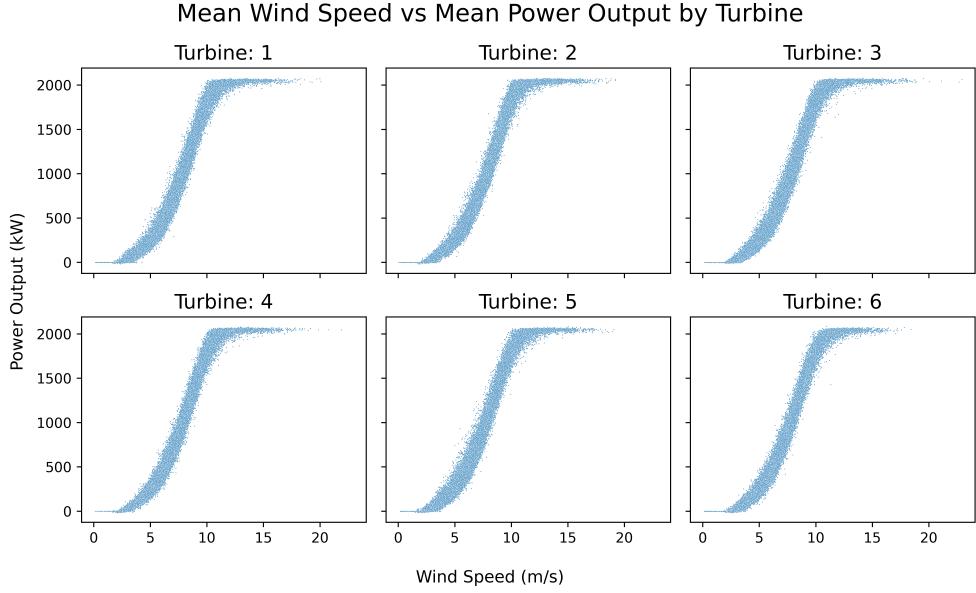
and component temperatures. Each variable is reported as mean, minimum, maximum, and standard deviation within the ten-minute bins.

An operational status and events file was used to filter the SCADA data and ensure consistent modelling of normal turbine behaviour. These logs provide detailed information on turbine operating states, including technical failures as well as operational or environmental standbys and warnings. To enable accurate modelling, all data associated with out-of-control conditions were removed so that model training was based solely on stable operating periods. Specifically, records corresponding to standbys, warnings, and operational stops were excluded. In addition, data from the week preceding each forced outage was discarded to reduce the likelihood of capturing deteriorating states. The resulting data elimination process is also described and illustrated in [15].

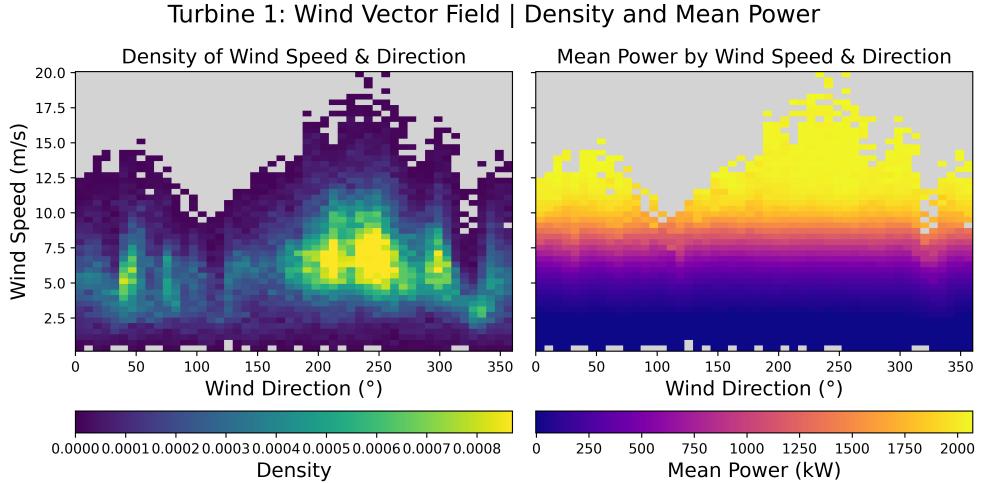
Fig. 1a shows the wind speed–power curves for the six turbines after filtering. The curves are highly consistent and non-linear across wind turbines, with relatively few points observed at very high and very low wind speeds. These sparse regions are particularly challenging for probabilistic models such as GPs (see Section 6, Fig. 6). To examine directional effects, Fig. 1b presents the joint distribution of wind speed, wind direction, and power output for wind turbine 1. As expected, wind speed is the dominant component of variability. However, even at fixed wind speeds, power output varies with wind direction, likely reflecting terrain influences. Figure 2a presents mean nacelle temperature against mean power output. The pattern is more complex: rather than a monotonic trend, the relationship is approximately S-shaped. At intermediate power output, around 1000 kW, nacelle temperatures are on average lower than at low and high outputs. This non-linear behaviour highlights nacelle temperature as a particularly interesting covariate, since it reflects both operational load and thermal dynamics within the wind turbine. In Fig. 2b, we show the relationship between mean transformer temperature and mean power output across the six wind turbines. The curves are highly consistent, suggesting that transformer temperature scales predictably with power output and may provide additional explanatory power beyond wind speed.

4. Methodology

We present the methodology underlying our proposed multi-task NDP framework. We first formalise the NDP in the regression setting, where the



(a)



(b)

Figure 1: SCADA data for the six turbines from the Kelmarsh wind farm [25] after removing standbys and warnings using the operational status and events file. (a) Wind power curves showing mean power output (kW) by mean wind speed (m/s). (b) Joint distribution of mean wind speed (m/s) and mean wind direction ($^{\circ}$) for turbine 1 (left), and mean power output (kW) by mean wind speed (m/s) and wind direction ($^{\circ}$) for turbine 1 (right).

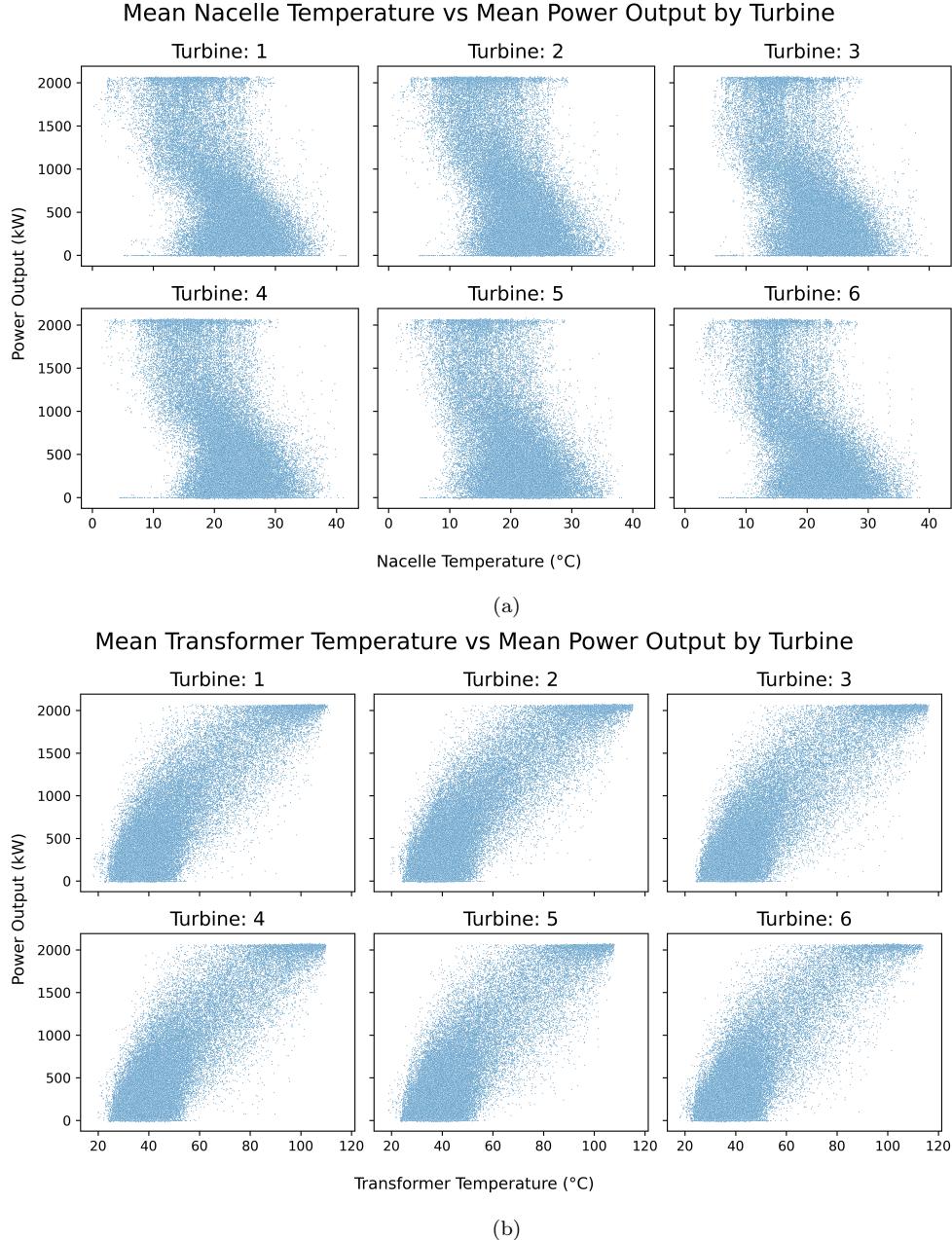


Figure 2: SCADA data for the six turbines from the Kelmarsh wind farm [25] after removing standbys and warnings using the operational status and events file. (a) Mean power output (kW) by mean nacelle temperature ($^{\circ}\text{C}$). (b) Mean power output (kW) by mean transformer temperature ($^{\circ}\text{C}$).

goal is to model a distribution over functions $f : \mathbb{R}^D \rightarrow \mathbb{R}$. Each training instance corresponds to a sampled function evaluated at a finite set of input locations $X \in \mathbb{R}^{N \times D}$ with corresponding outputs $y = f(X) \in \mathbb{R}^N$. By training across many such functions, the model learns an empirical covariance structure that captures correlations between inputs and outputs, enabling calibrated predictive distributions even in regions with sparse data.

We outline the forward and reverse processes, and the sampling procedure, highlighting parallels with the DDPM framework. We then introduce our multi-task extension, which incorporates a task encoder to capture cross-turbine dependencies in wind farms.

4.1. Neural diffusion processes (NDPs)

We begin by formalising the training data used by NDPs. Each training instance corresponds to a sampled function $f_i : \mathbb{R}^D \rightarrow \mathbb{R}$ evaluated at a finite set of inputs $x_i \in \mathbb{R}^{N \times D}$ with outputs $y_i = f_i(x_i) \in \mathbb{R}^N$. The overall training dataset is therefore

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^M, \quad x_i \in \mathbb{R}^{N \times D}, \quad y_i \in \mathbb{R}^N, \quad (11)$$

where M is the number of sampled functions, N is the number of input locations per function, and D is the input dimension.

4.1.1. Forward process

Let $x_0 = x$ and $y_0 = y$. Then the NDPs gradually add noise following

$$q\left(\begin{bmatrix} x_t \\ y_t \end{bmatrix} \middle| \begin{bmatrix} x_{t-1} \\ y_{t-1} \end{bmatrix}\right) = \mathcal{N}\left(y_t; \sqrt{1 - \beta_t} y_{t-1}, \beta_t I\right) \quad (12)$$

for each timestep t where β_t is the noise from the variance schedule [22]. Please note the similarities to Eq. 6, where now, rather than just modelling the input data s_t , we explicitly differentiate between the input locations x_t and outputs y_t . This corresponds to adding Gaussian noise to the function values y_t while keeping the input locations x_t fixed for all t . By the final timestep ($t = T$), the function values y_t will look like Gaussian noise from $\mathcal{N}(0, 1)$. Fig. 3 illustrates the forward diffusion process, where Gaussian noise is gradually added to the outputs using a cosine variance schedule [24]. As t increases, the data loses structure and converges towards white noise.

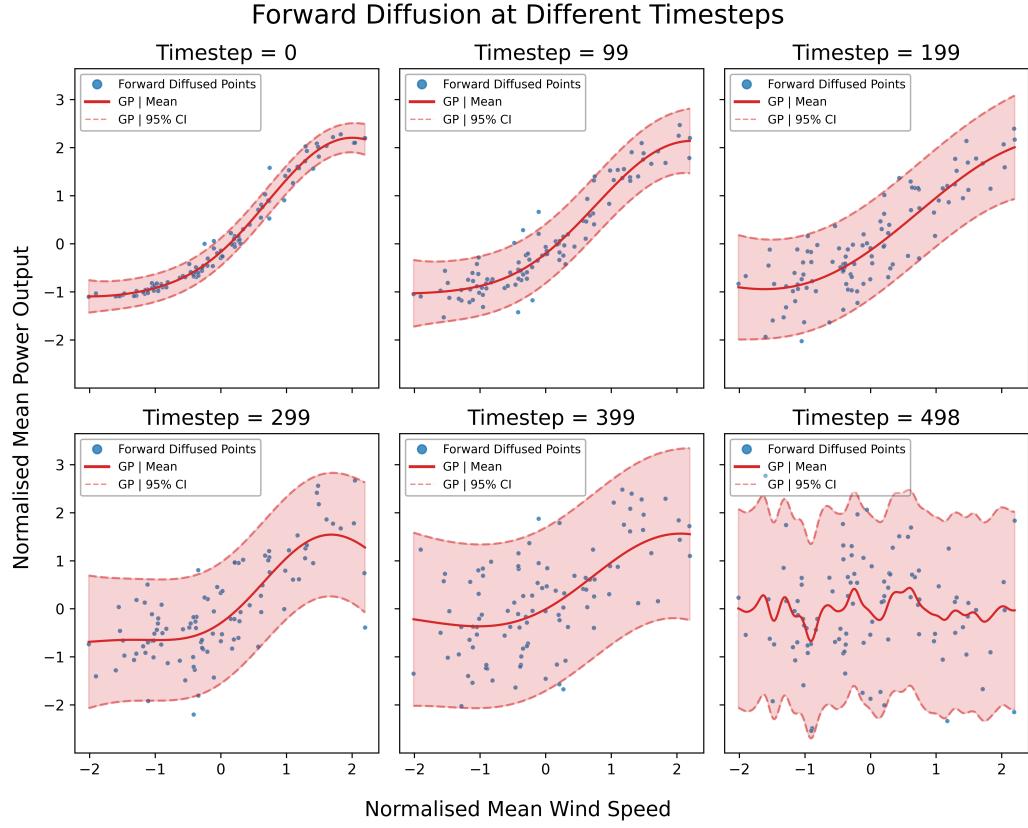


Figure 3: Illustration of the forward diffusion process. Gaussian noise is added incrementally at each timestep under a cosine variance schedule, progressively degrading the structure of the data until it converges to white noise. GP= Gaussian process, CI= Confidence interval

4.1.2. Reverse Process

NDPs use a neural network that learns to de-noise the corrupted function values, in a very similar manner to the DDPMs of [20]. The input locations x_t are not corrupted, while the y_t are. In fact, the model will use the uncorrupted x_t to help improve the de-noising model. The parametrised kernel is of the form [22]:

$$p_\theta \left(\begin{bmatrix} x_{t-1} \\ y_{t-1} \end{bmatrix} \middle| \begin{bmatrix} x_t \\ y_t \end{bmatrix} \right) = \mathcal{N} \left(y_{t-1}; \mu_\theta(x_t, y_t, t), \tilde{\beta}_t I \right) \quad (13)$$

where the mean is parametrised as:

$$\mu_\theta(x_t, y_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(y_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta(x_t, y_t, t) \right), \quad (14)$$

where the noise model $\epsilon_\theta : \mathbb{R}^{N \times D} \times \mathbb{R}^N \times \mathbb{R} \rightarrow \mathbb{R}^N$ takes as input the non-corrupted x_t as well as the corrupted y_t , and the timestep t . As previously, $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{j=1}^t \alpha_j$. Eq. 14 differs from the previous DDPM formulation Eq. 9 since it uses just the s_t as input to the noise model rather than explicitly using uncorrupted x_t and corrupted y_t . The goal of this noise model is to predict the noise added to y_0 to get y_t in line with the DDPM model in [20].

4.1.3. Objective

The objective is very similar to Eq. 10 but with s_t replaced by x_t and y_t :

$$L_\theta = \mathbb{E}_{t, x_0, y_0, \epsilon} [\|\epsilon - \epsilon_\theta(x_0, y_t, t)\|^2], \quad (15)$$

where $x_0 = x_t$ for all timesteps t , since the diffusion process is applied only to the outputs while the inputs remain fixed. The corrupted outputs are generated through the following reparameterisation:

$$y_t = \sqrt{\bar{\alpha}_t} y_0 + \sqrt{1 - \bar{\alpha}_t} \epsilon, \quad \epsilon \sim \mathcal{N}(0, I). \quad (16)$$

4.1.4. Sampling

Unconditional sampling: Given some input locations x , we can sample unconditionally from the trained model prior by starting with random noise and following the reverse process for T timesteps. Specifically, we initialise $y_T \sim \mathcal{N}(0, I)$ and then apply the parametrised kernel for $t = T, \dots, 1$. At each step, the neural network predicts the noise added in the forward process, and new noise is reintroduced. This procedure prevents the trajectories from collapsing to a local maximum and ensures coverage of the full distribution.

Unconditional samples already yield realistic outputs that resemble the true data distribution. In practice, however, if the training dataset is highly varied, unconditional sampling may not capture sufficient structure. For wind turbine data, where the relationships between inputs and power output are similar across turbines, the prior alone often produces plausible results.

Conditional sampling: One of the main strengths of NDPs, as with NPs, is their ability to quickly adapt with only a small amount of data. Conditional sampling is achieved by reinjecting context points at each timestep, using

a slight adaptation of the Repaint algorithm [21]. This guides the reverse process to remain consistent with the context set.

Formally, we are interested in $p(y_0^* | x_0^*, D)$, where $D = (x_0^c \in \mathbb{R}^{M \times D}, y_0^c \in \mathbb{R}^M)$ is the set of context points. The target is initialised as $y_T^* \sim \mathcal{N}(0, I)$. At each timestep, we add noise to the context points, obtaining y_t^c from the forward process. We then take the union of the noisy context points and the input locations $x_0 = \{x_0^*, x_0^c\}$, together with the union of the target outputs and context outputs $y_t = \{y_t^*, y_t^c\}$. Finally, the reverse process kernel samples

$$y_{t-1} \sim \mathcal{N}\left(\frac{1}{\sqrt{\alpha_t}} \left(y_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta(x_0, y_t, t) \right), \tilde{\beta}_t I\right). \quad (17)$$

Repeating this for $t = T, T-1, \dots, 1$ means that, at each backward step, the context points steer the process in the desired direction. Initially, when y_t^* is close to pure noise, the context points already contain useful structure. This pulls the reverse trajectories towards functions consistent with the context set, leading to samples that better fit the true data.

In Fig. 4, we compare unconditional and conditional sampling. Even at early timesteps, the conditional model is visibly guided by the context points, producing a closer match to the true data throughout the process. To illustrate the dynamics more clearly, Fig. 5 depicts the forward and reverse trajectories for a single input point. In the forward process (blue), Gaussian noise gradually corrupts the true value. In the reverse process (red), starting from pure noise, the trajectories converge back to the true value, while uncertainty narrows as the timestep decreases.

4.2. Noise model architecture

The noise model architecture follows [22], designed to generate sensible prior distributions over functions. While any NN could in principle serve as the denoiser, certain structural properties are desirable: There are two key requirements:

- (i) *Input-size agnosticity*: the model should work for arbitrary numbers of input points N and input dimensions D , producing consistent priors whether inference is carried out at one or many locations. This is achieved by replicating the outputs $y_t \in \mathbb{R}^N$ across D copies so that the network weights do not depend directly on N or D . As a result, one trained NDP can be applied across datasets of varying size and dimensionality.

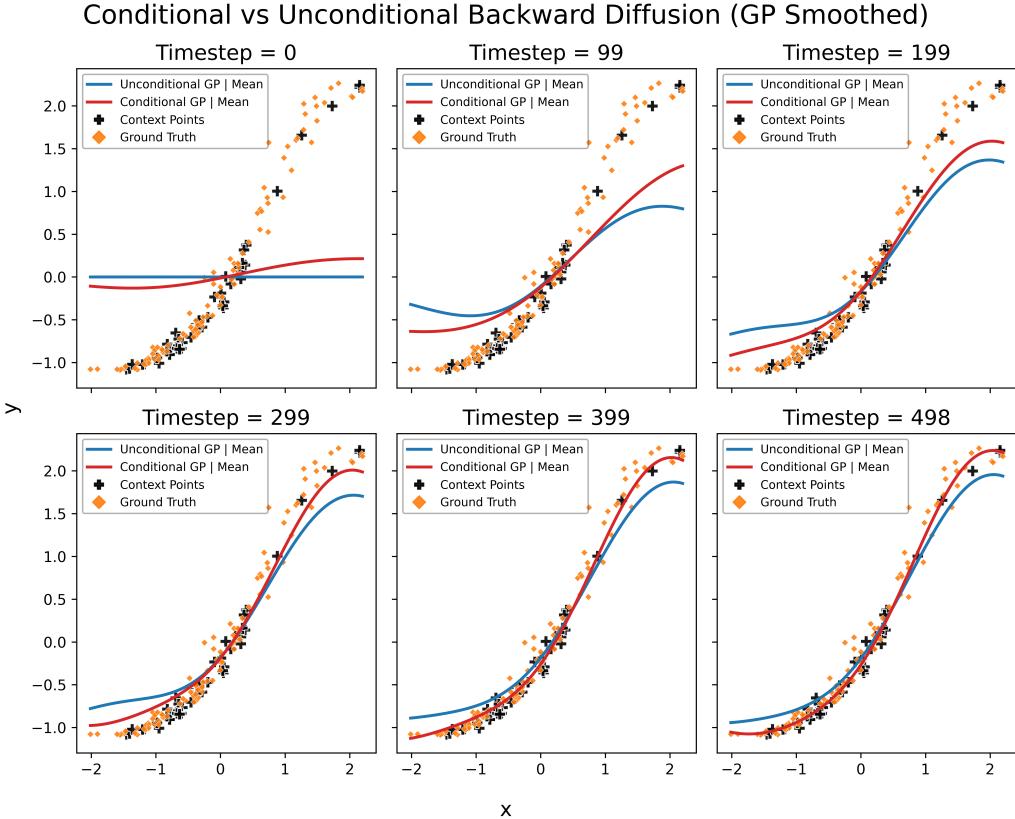
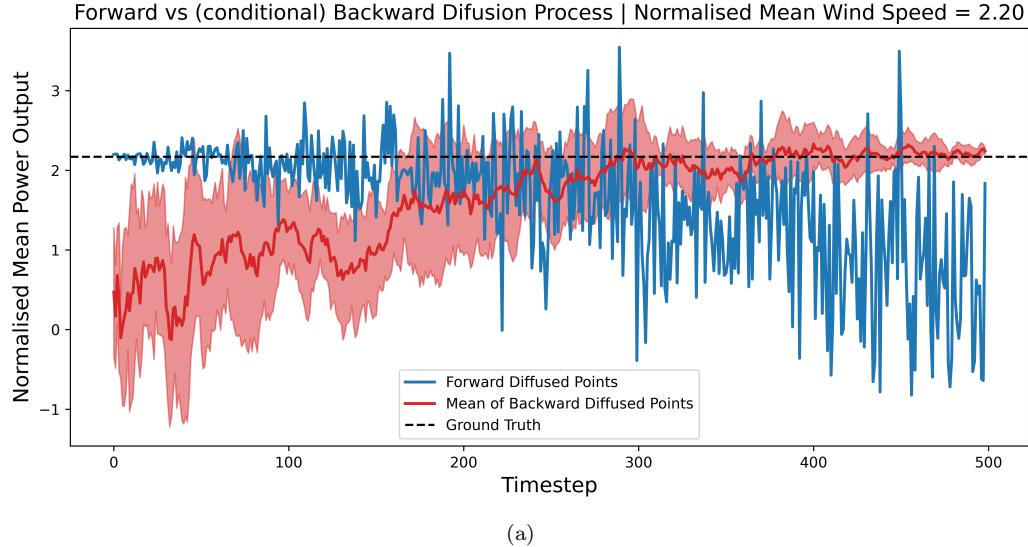
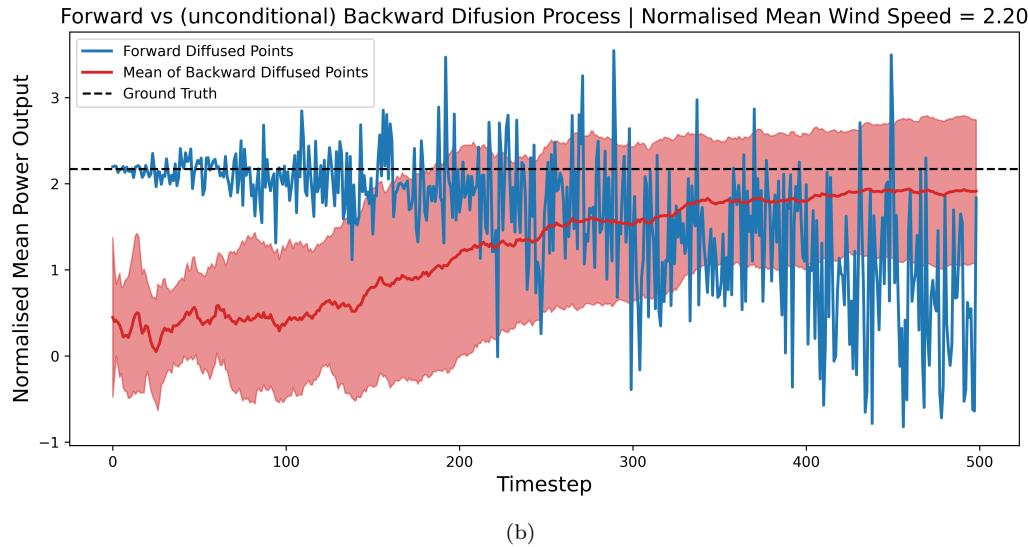


Figure 4: Comparison of unconditional and conditional reverse diffusion sampling. We sampled 10 reverse trajectories per input location, plotting the mean of these. Conditional sampling provides a closer fit to the true data across all timesteps. GP= Gaussian process

(ii) *Equivariance and invariance*: predictions should not depend on the ordering of inputs, context points, or feature dimensions. Reordering inputs should not change the probability of the data, just as kernels such as the RBF in GPs are invariant to permutations. To enforce the equivariance and invariance, [22] proposed a bi-dimensional attention block, $A_t : \mathbb{R}^{N \times D \times H} \rightarrow \mathbb{R}^{N \times D \times H}$. This multi-head self-attention mechanism operates jointly across input locations and feature dimensions. Attention weights are computed via dot-product similarity, and multiple heads are run in parallel to capture complementary relationships.



(a)



(b)

Figure 5: Forward and backward trajectories for a single input point (a) conditional sampling and (b) unconditional sampling. For both (a) and (b) in the forward process (blue), Gaussian noise is gradually added. In the reverse process (red), noise is removed, converging towards the true value with decreasing uncertainty. The shaded area shows one standard deviation.

This design ensures permutation equivariance and invariance while retaining the flexibility of neural architectures, addressing limitations

that are often overlooked in standard NN models.

4.3. Multi-task neural diffusion processes

We propose a novel architectural addition to extend NDPs to MT-NDPs, borrowing key items from [18]. The broad idea is to add a task encoder that can take context points and use that information to identify the task the model is performing a prediction on. This information is represented in the encoding space by a single vector $v \in \mathbb{R}^\kappa$. That is, the encoding space has dimension κ , allowing enough flexibility for the model to distinguish between different tasks, i.e., wind turbines. The dimension κ is assumed to be small; for example, in our application to SCADA data described in Section 5, we set $\kappa = 8$. This vector can then be propagated downstream to the diffusion architecture, which remains unchanged. The way we pass the information downstream is simply to concatenate these extra κ dimensions onto the other input features. These κ additional features will be the same for all points in the same training (or testing) sample.

4.3.1. Forward Process

Let $x_0 = x$, $y_0 = y$, and suppose we have context points $x^c = x_0^c$, $y^c = y_0^c$, then the MT-NDPs gradually add noise following:

$$q\left(\begin{bmatrix} x_t \\ y_t \\ x_t^c \\ y_t^c \end{bmatrix} \mid \begin{bmatrix} x_{t-1} \\ y_{t-1} \\ x_{t-1}^c \\ y_{t-1}^c \end{bmatrix}\right) = \mathcal{N}\left(y_t; \sqrt{1 - \beta_t} y_{t-1}, \beta_t I\right) \quad (18)$$

for each timestep t where β_t is the noise from the variance schedule. Notice the similarities to Eq. 12, where now, as well as explicitly differentiating between the input locations x_t and outputs y_t , we also incorporate the context points x_t^c and y_t^c . This corresponds to adding Gaussian noise to the function values y_t while keeping the input locations x_t , and context points (x_t^c, y_t^c) fixed for all t . By the final timestep $t = T$, the function values y_t will look like Gaussian noise from $\mathcal{N}(0, 1)$.

4.3.2. Reverse Process

MT-NDPs, like NDPs, use an NN that learns to de-noise the corrupted function values, where we now also make use of some context points (x_t^c, y_t^c) . The input locations x_t and context points (x_t^c, y_t^c) are not corrupted, while

the y_t are. In fact, the model will use the uncorrupted x_t and (x_t^c, y_t^c) to help improve the de-noising model. The parametrised kernel is of the form:

$$p_\theta \left(\begin{bmatrix} x_{t-1} \\ y_{t-1} \\ x_{t-1}^c \\ y_{t-1}^c \end{bmatrix} \mid \begin{bmatrix} x_t \\ y_t \\ x_t^c \\ y_t^c \end{bmatrix} \right) = \mathcal{N}(y_{t-1}; \mu_\theta(x_t, y_t, x_t^c, y_t^c, t), \tilde{\beta}_t I) \quad (19)$$

with

$$\mu_\theta(x_t, y_t, x_t^c, y_t^c, t) = \frac{1}{\sqrt{\alpha_t}} \left(y_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \epsilon_\theta^{MT}(x_t, y_t, x_t^c, y_t^c, t) \right), \quad (20)$$

where the noise model $\epsilon_\theta^{MT} : \mathbb{R}^{(N+M) \times D} \times \mathbb{R}^{(N+M)} \times \mathbb{R} \rightarrow \mathbb{R}^N$, now leverages the input locations x_t and context points (x_t^c, y_t^c) . This model is in fact the composition of the encoder model $\epsilon_\theta^{\text{encoder}}$ and the original NDP de-noising model ϵ_θ^{NDP} ,

$$\epsilon_\theta^{MT}(x_t, y_t, x_t^c, y_t^c, t) = \epsilon_\theta^{NDP} \circ \epsilon_\theta^{\text{encoder}}(x_t, y_t, x_t^c, y_t^c, t). \quad (21)$$

This encoder $\epsilon_\theta^{\text{encoder}}(x_t, y_t, x_t^c, y_t^c, t)$ does the following:

- (i) Acts as the identity on (x_t, y_t) , and mapping the (x_t, y_t) to some κ -dimensional space: $\mathbb{R}^{(N+M) \times D} \times \mathbb{R}^{(N+M)} \times \mathbb{R} \rightarrow \mathbb{R}^{N \times D} \times \mathbb{R}^N \times \mathbb{R}^{\kappa \times M}$,
- (ii) Taking the mean over the image of all context points in this κ -dimensional space, $\bar{v} \in \mathbb{R}^\kappa : \mathbb{R}^{N \times D} \times \mathbb{R}^N \times \mathbb{R}^{\kappa \times M} \rightarrow \mathbb{R}^{N \times D} \times \mathbb{R}^N \times \mathbb{R}^\kappa$,
- (iii) Copying this mean vector, \bar{v} , N times so that it can be concatenated with the other input features: $\mathbb{R}^{N \times D} \times \mathbb{R}^N \times \mathbb{R}^\kappa \rightarrow \mathbb{R}^{N \times D} \times \mathbb{R}^N \times \mathbb{R}^{\kappa \times N} = \mathbb{R}^{N \times (D+\kappa)} \times \mathbb{R}^N$

Therefore, given the initial input locations and targets $T = (x_0 \in \mathbb{R}^{N \times D}, y_0 \in \mathbb{R}^N)$ and context points $C = (x_0^c \in \mathbb{R}^{M \times D}, y_0^c \in \mathbb{R}^M)$, after the encoder we preserve T and are left with N copies of a vector $\bar{v} \in \mathbb{R}^\kappa$ that we concatenate to the other input features of x_o to leave us with $D + \kappa$ dimensional input features, and a modified set of input locations and targets: $T^* = (x_0^* \in \mathbb{R}^{N \times (D+\kappa)}, y_0^* \in \mathbb{R}^N)$ that is then passed through the NDP de-noising model $\epsilon_\theta^{NDP}(x_t^*, y_t^*, t) : \mathbb{R}^{N \times (D+\kappa)} \times \mathbb{R}^N \rightarrow \mathbb{R}^N$.

By constructing the task encoder in this way, we ensure agnosticity to input size. During training, the number of context points in each set is

sampled uniformly from the integers 0 to 50. This setting exposes the model to situations with no context, with few context points, and with many context points. As a result, the model learns both a global prior when no context is provided and task-specific priors when context points are available.

Equivariance is preserved by taking the mean $\bar{v} \in \mathbb{R}^\kappa$, which ensures that the mapping is invariant to the ordering of context points. The target points remain unchanged by the encoder. Invariance, however, depends on the encoder architecture. Here we use a simple MLP, which does not guarantee invariance. Achieving this property would require multi-head self-attention blocks, as in the NDP architecture, but this would substantially increase model complexity and computational cost. In the specific application to wind farms, where the ordering of input features can be fixed across training and testing, the lack of invariance is less problematic than it would be in broader meta-learning settings. Details of hidden layer sizes are provided in Section 5.2.

4.3.3. Objective

The objective is very similar to the NDP objective in Eq. 15:

$$L_\theta = \mathbb{E}_{t,x_0,y_0,x_0^c,y_0^c,\epsilon} \left[\|\epsilon - \epsilon_\theta^{MT}(x_0, y_t, x_0^c, y_0^c, t)\|^2 \right], \quad (22)$$

where $x_0 = x_t$, $x_0^c = x_t^c$, and $y_0^c = y_t^c$ for all timesteps t and y_t follows the same corruption process as in Eq. 16.

4.3.4. Sampling

Unconditional sampling remains unchanged from the NDP setup described in Section 4.1.4, as we use no context points at all. Simply take $y_T \sim \mathcal{N}(0, I)$, and then for $t = T, \dots, 1$ apply the parametrised kernel. For conditional sampling, we still follow the process for conditional sampling of NDPs described in Section 4.1.4, but now using our upgraded de-noising model ϵ_θ^{MT} :

$$y_{t-1} \sim \mathcal{N} \left(\frac{1}{\sqrt{\alpha_t}} \left(y_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}} \epsilon_\theta^{MT}(x_0, y_t, x_0^c, y_0^c, t) \right), \tilde{\beta}_t I \right). \quad (23)$$

Note that we still benefit from the Repaint algorithm as we iterate through the reverse process for $t = T, \dots, 1$.

5. Experimental design

We evaluate the proposed models on wind power prediction tasks using SCADA data from the Kelmarsh wind farm. As the number of input variables and model complexity increase, training times grow substantially. Inference with NDPs can also be slow, since both conditional and unconditional sampling require $T = 500$ diffusion steps, each involving a neural network evaluation. By contrast, GPs offer fast inference for small datasets, but quickly become intractable for larger datasets. Sparse GPs have been studied on the same dataset [15], but suffer from underconfidence in predictive uncertainty.

A key advantage of NDPs and related meta-learning models is their ability to adapt to unseen tasks with only a small number of context points. To test this property, we adopt a challenging setup where models are trained on a subset of turbines and evaluated on a previously unseen turbine. Context points, drawn from the training set of the test turbine, guide the model during inference. GPs do not use context sets explicitly; all training data acts as context.

5.1. Training and testing framework

Some aspects of the experimental design are common to all comparisons and are therefore outlined here. NDPs learn distributions over functions, which requires training on many function samples. In our setting, each function sample consists of 100 covariate–target pairs. To reduce the impact of autocorrelation in wind generation, both training and testing samples are shuffled so that they are not sequential. For these samples to represent meaningful draws from the underlying stochastic process, all points in a given sample are taken from the same turbine.

To evaluate the ability of NDPs to generalise, we train models on turbines 2–6 and test them on turbine 1. Context points, drawn from the training split of turbine 1, are provided during inference to guide predictions. This setup allows us to assess (i) how informative the learned prior is in the absence of context points, and (ii) the benefit of incorporating context points for accuracy and uncertainty calibration.

All features are standardised per wind turbine, using the mean and standard deviation from the wind turbine’s training set. This step is necessary for both NDPs and GPs. To ensure fair comparisons, we use the same test functions across all models and context sizes. Context sets of size 0, 25, and

50 points are considered. Each evaluation uses 30 test functions of length 100.

5.2. Model parameters

For all NDP variants, we use a cosine noise schedule [24] with $T = 500$ diffusion steps. The denoising neural network has 4 hidden layers of dimension 64, with multi-head self-attention using $\kappa = 8$ heads. The optimiser is ADAM [26], following the configuration of [22], with a 20-epoch warmup, 200-epoch decay, learning rate schedule ($2 \times 10^{-5} \rightarrow 10^{-3} \rightarrow 10^{-5}$), and an EMA weight of 0.995. For sampling, we generate 200 reverse diffusion trajectories in the one-dimensional case and 100 in higher-dimensional settings, balancing predictive sharpness with computational cost.

In the multi-task framework, the task encoder is an MLP with four 4 layers of size 64, GeLU activations, and an 8-dimensional output embedding. The mean embedding across all context points forms a task representation vector. Training is performed for 250 epochs using batches of 32 function samples of length 100, with 500 samples drawn per wind turbine (5 tasks/turbines in total), and 100 randomly shuffled function samples presented per epoch.

5.3. Model evaluation

We evaluated both point accuracy and predictive uncertainty. Results are averaged over 30 test function samples, each containing 100 input–output pairs, and over the specified numbers of context points $C \in \{0, 25, 50\}$.

For a given test sample s , let $\{(x_i, y_i)\}_{i=1}^N$ with $N = 100$ denote inputs and ground-truth outputs; \hat{y}_i is the model’s point prediction. For NDP/MT-NDP models, point predictions are the empirical mean over reverse-diffusion trajectories at each x_i , and central prediction intervals are the corresponding empirical quantiles across trajectories. We report summary statistics in Tables 1 and 2, obtained by averaging per-sample metrics over 30 test samples, each with $N = 100$ input–output pairs, for each (model \times context size) configuration.

To assess point prediction accuracy, we compute the following per-sample losses for each test set: (i) mean absolute error (MAE), defined as $\text{MAEs} = \frac{1}{N} \sum i = 1^N |y_i - \hat{y}_i|$ and (ii) root mean squared error (RMSE), defined as $\text{RMSEs} = \sqrt{\frac{1}{N} \sum i = 1^N (y_i - \hat{y}_i)^2}$.

To evaluate predictive uncertainty, we compute the coverage error (CE), which quantifies the discrepancy between the nominal coverage q and the actual coverage achieved by the predictive intervals:

$$CE_s = \frac{1}{|Q|} \sum_{q \in Q} \left| \frac{1}{n_s} \sum_{i=1}^{n_s} \mathbf{1}\{y_i^{(s)} \in I_q^{(s)}(x_i)\} - q \right|, \quad (24)$$

where Q is the set of nominal quantile levels and $q \in Q$ denotes a particular level, $y_i^{(s)}$ and n_s denote the observed outputs and number of points respectively in sample s , and the term $I_q^{(s)}(x_i)$ is the predicted interval at coverage level q for that input. The indicator function $\mathbf{1}\{\cdot\}$ equals 1 if the condition inside holds and 0 otherwise. The inner average computes the empirical coverage at level q , subtracting q gives the deviation from the nominal level, and the absolute value penalises both under- and over-coverage equally. Averaging across all $q \in Q$ yields the mean coverage error CE_s , with smaller values indicating better calibrated predictive intervals. Hence, a good model yields $CE_s \approx 0$, indicating predictive intervals that closely align with their nominal coverage.

6. Results

We first compare a one-dimensional GP with a one-dimensional NDP, using only wind speed as a feature. In this setting, wind turbines are nearly indistinguishable, consistent with exploratory analysis (Fig. 1a). For the GP, we mimic the NDP setup, training on turbines 2–6 and testing on turbine 1. Due to scalability limits, the GP is trained on a subsample of 2000 points evenly drawn from the five turbines. We compare this GP baseline only to the NDP with no context points, ensuring a fair assessment of prior quality.

We next compare a one-dimensional NDP, using wind speed as a feature, with a five-dimensional NDP. The additional inputs are wind direction, encoded as its sine and cosine components, nacelle temperature, and transformer temperature. These variables were selected based on exploratory analysis shown in Fig. 2, which revealed strong or nonlinear relationships with wind power output.

Finally, we assess the benefits of extending NDPs to the multi-task setting. We evaluate two variants, one single-task NDP with five input features and no task encoder, and our multi-task NDP using both encoder and conditional diffusion with Repaint.

6.1. One-dimensional GP against one-dimensional NDP

We first compare the one-dimensional NDP without context points against a GP baseline to ensure a fair evaluation. The effect of context on NDP performance is examined separately in Sections 6.2 and 6.4. Fig. 6 shows the

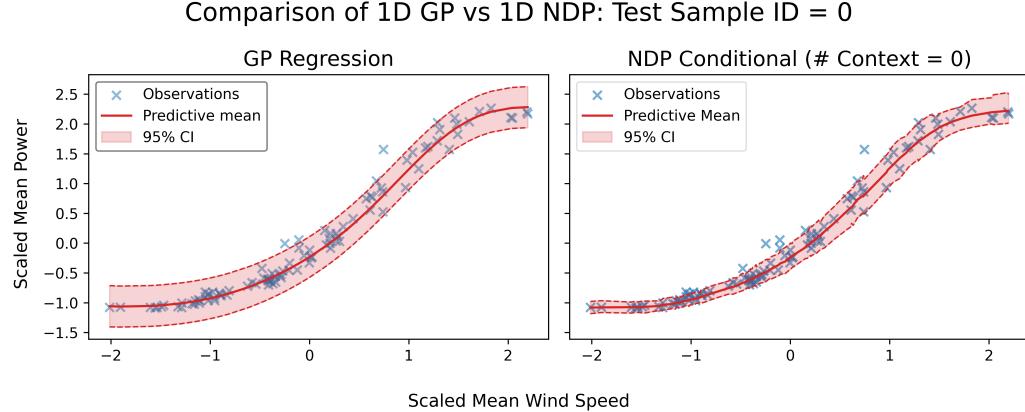
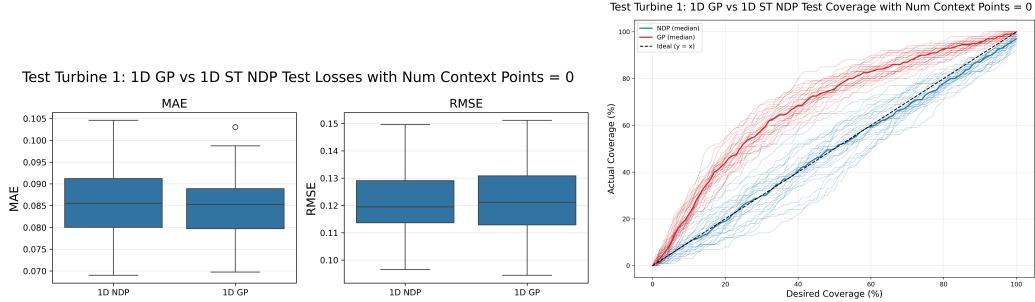


Figure 6: Fit of Gaussian process (GP) (left) and neural diffusion process (NDP) (right) on a test sample of 100 points from turbine 1, with 95% confidence intervals (CI) on normalised wind speed and power output. 1D = One-dimensional.

fit of both models on a representative test function of 100 points from turbine 1. While both capture the mean trend well, their uncertainty estimates differ substantially. The NDP derives uncertainty from quantiles of reverse-diffusion trajectories, without assuming Gaussianity, whereas the GP relies on Gaussian predictive distributions. As a result, the GP is clearly underconfident, even in dense regions ($x \in [-1, 1]$). This discrepancy becomes more pronounced in extrapolation regions ($x < -1$), where the NDP produces appropriately narrower confidence intervals. At the upper end of the domain ($x > 1$), improvements are less marked, though the NDP still provides slightly tighter uncertainty bounds than the GP.

Considering all 30 test samples of 100 points each, both models achieve comparable point prediction accuracy. As shown in Fig. 7a, the distributions of MAE and RMSE for the GP and NDP are nearly indistinguishable, indicating that both capture the mean power–wind speed relationship effectively.

The distinction between the models becomes clear when evaluating predictive uncertainty. Fig. 7b shows the empirical coverage against the nominal coverage level. A perfectly calibrated model would follow the diagonal $y = x$



(a) Mean absolute error (MAE, left) and root mean square (RMSE, right) distributions for Gaussian process (GP) and neural and neural diffusion process (NDP) over 30 test samples, each with 100 points (turbine 1).

(b) Observed vs. theoretical coverage probabilities for Gaussian process (GP) and neural diffusion process (NDP) over 30 test samples, each with 100 points (turbine 1).

Figure 7: Comparison of Gaussian process (GP) and neural diffusion process (NDP) on turbine 1 test data. (a) Point prediction accuracy (MAE and RMSE distributions). (b) Predictive uncertainty calibration (coverage probabilities). 1D = One-dimensional, ST= single-task

line. While the NDP is not flawless, it consistently lies closer to this ideal than the GP, which suffers from pronounced underconfidence across coverage levels. This highlights a key strength of NDPs: the ability to generate well-calibrated uncertainty estimates without relying on Gaussian assumptions.

6.2. One-dimensional NDP against five-dimensional NDP

The five-dimensional NDP extends the one-dimensional baseline by incorporating additional input features: the cosine and sine of wind direction, transformer temperature, and nacelle temperature, alongside wind speed.

Fig. 8 compares the point prediction errors of the one-dimensional and five-dimensional NDP models across different context sizes. Incorporating more input features leads to a clear reduction in both MAE and RMSE, demonstrating the added explanatory power of the extended feature set. Notably, the benefit of context is much more apparent in the five-dimensional NDP model, where performance improves consistently as context size increases. In contrast, the one-dimensional NDP model shows little to no improvement when additional context points are provided.

Fig. 9 presents observed versus nominal coverage for both models at different context sizes. The five-dimensional NDP shows steady improvement in calibration with increasing context, whereas the one-dimensional NDP remains largely unaffected by context size. Interestingly, the one-dimensional

Test Turbine 1: 1D ST NDP vs 5D ST NDP Test Losses by Number of Context Points

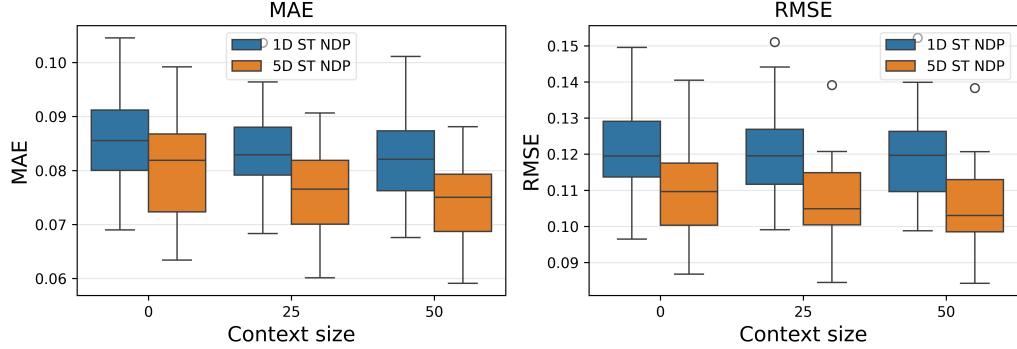


Figure 8: Distributions of mean absolute error (MAE, left) and root mean square error (RMSE, right) for the one-dimensional (1D) and five-dimensional (5D) neural diffusion process (NDP) models, evaluated with 0, 25, and 50 context points. Results are aggregated over 30 test functions of 100 points each from turbine 1. ST = single-task.

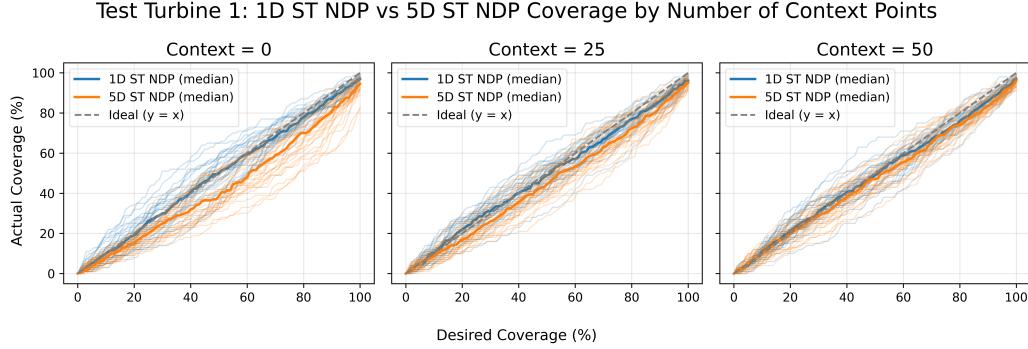


Figure 9: Observed vs. theoretical coverage probabilities for the one-dimensional (1D) and five-dimensional (5D) neural diffusion process (NDP) models with 0, 25, and 50 context points. Results are averaged over 30 test functions of 100 points each from turbine 1. ST = single-task.

NDP achieves slightly better coverage alignment than the five-dimensional NDP when both are provided with 50 context points. This raises the question of whether the five-dimensional NDP model would outperform the one-dimensional model given a larger context set, a question left open here due to computational constraints.

Finally, Fig. 10 summarises coverage performance using the CE metric.

Test Turbine 1: 1D ST NDP vs 5D ST NDP Coverage Error by Number of Context Points

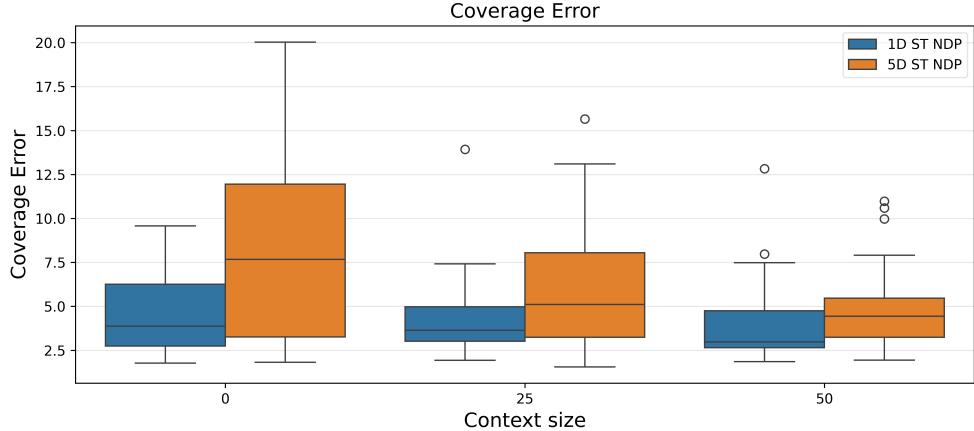


Figure 10: Coverage error for the one-dimensional single-task neural diffusion process (1D ST NDP) and five-dimensional single-task neural diffusion process (5D ST NDP) models with 0, 25, and 50 context points. Results are computed over 30 test functions of 100 points each from turbine 1.

The results confirm that context improves calibration for the five-dimensional NDP, narrowing the gap to the one-dimensional NDP model. When considered alongside the substantial accuracy gains in Fig. 8, the evidence suggests that the five-dimensional NDP provides a meaningful improvement over the one-dimensional NDP baseline, particularly in balancing accuracy with uncertainty calibration.

6.3. Aggregated results

Table 1 provides a quantitative summary of the comparisons, aggregating point prediction accuracy (MAE, RMSE) and predictive uncertainty calibration (CE) across all models and context sizes for turbine 1. The results confirm the visual trends: while the GP achieves similar MAE and RMSE to the NDP, it performs very poorly in terms of uncertainty calibration, with a coverage error more than four times higher. Among the NDP variants, incorporating additional features (from one-dimensional to five-dimensional) consistently reduces prediction error, while increasing the number of context points improves both accuracy and calibration.

Table 1: Out-of-sample performance of Gaussian process (GP) and single-task neural diffusion process (ST-NDP) models on turbine 1, averaged over 30 test functions (100 points each). Metrics reported: mean absolute error (MAE, kW), root mean squared error (RMSE, kW), and coverage error (CE, %).

<i>Model</i>	<i># of features</i>	<i>Context</i>	<i>MAE (kW)</i>	<i>RMSE (kW)</i>	<i>CE (%)</i>
GP	1	0	51.636	75.596	18.075
ST-NDP	1	0	52.603	75.139	4.584
		25	51.273	74.568	4.181
		50	50.545	74.137	3.938
ST-NDP	3	0	60.967	88.096	9.731
		25	52.338	74.312	6.750
		50	50.855	73.537	5.369
ST-NDP	5	0	49.429	67.685	8.351
		25	46.962	66.007	6.029
		50	45.791	65.301	4.816

6.4. NDP against MT-NDP

6.4.1. Encoder analysis

To understand how the encoder represents turbines, we project its κ -dimensional embeddings into two principal components using principal component analysis (PCA). Fig. 11 shows the resulting clusters when the model is trained on turbines 2–6 and tested on turbine 1, with context set sizes ranging from very small (0–4 points) to large (46–50 points). With few context points (left panel), wind turbines are essentially indistinguishable, reflecting the fact that the model has little task-specific information and must rely on its global prior. As context increases (middle and right panels), the encoder begins to separate turbines, with turbine 6 in particular forming a clearly distinct cluster. By contrast, turbine 1 remains centred near the origin across all context sizes.

These findings have two key implications for evaluation. First, turbine 1 is not an informative test case for the MT-NDP: because its embeddings remain close to the global mean, the model receives little benefit from task-specific adaptation, and its predictions are nearly identical to those of the ST-NDP. Second, turbine 6 stands out as the most challenging and distinctive case in the wind farm, forcing the model to deviate from the global prior and rely

PC Projection of Encoder Embeddings by Turbine Across Context Sizes (Trained on Turbines 2-6)

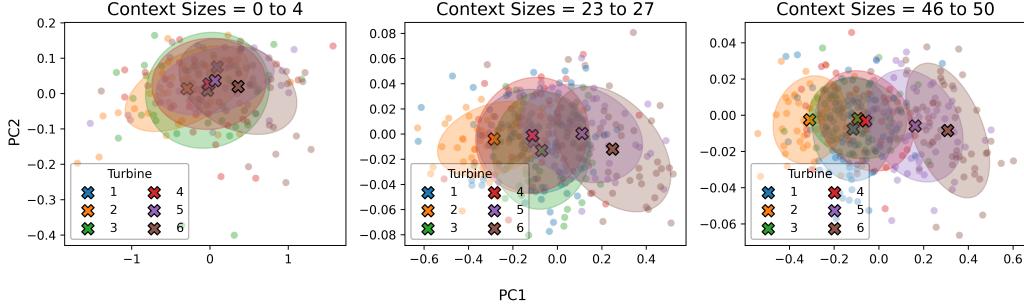


Figure 11: Principal component (PC) projection of encoder embeddings for turbines 1–6, trained on turbines 2–6. Each panel corresponds to a different range of context set sizes. Colours denote turbines, shaded ellipses show 68% coverage regions, and crosses mark cluster means. As context increases, the encoder begins to separate turbines, with turbine 6 emerging as the most distinct.

heavily on context information for effective adaptation.

For this reason, in the following sections, we focus our quantitative evaluation on turbine 6. This choice provides a stronger and more discriminative test of the MT-NDP’s ability to capture turbine-specific behaviour, highlighting the advantages of multi-task learning in settings where tasks differ substantially.

6.4.2. Model comparison

Fig. 12 reports the point prediction errors (MAE and RMSE) for three models: the one-dimensional ST-NDP, the five-dimensional ST-NDP, and the five-dimensional MT-NDP. All models improve with increasing context size, but the gains are most pronounced for the five-dimensional MT-NDP, which consistently achieves the lowest errors when 25 or 50 context points are available. This highlights the value of the task encoder in leveraging limited context to improve predictions on unseen wind turbines.

Fig. 13a compares empirical and nominal coverage for the same models. The five-dimensional MT-NDP demonstrates the best calibration overall, with coverage curves lying closest to the diagonal as context size increases. By contrast, the one-dimensional ST-NDP shows little benefit from context, underscoring its limited ability to capture multi-feature relationships.

Finally, Fig. 13b summarises these findings using the CE. The five-

Test Turbine 6: NDP Test Losses by Model and Number of Context Points

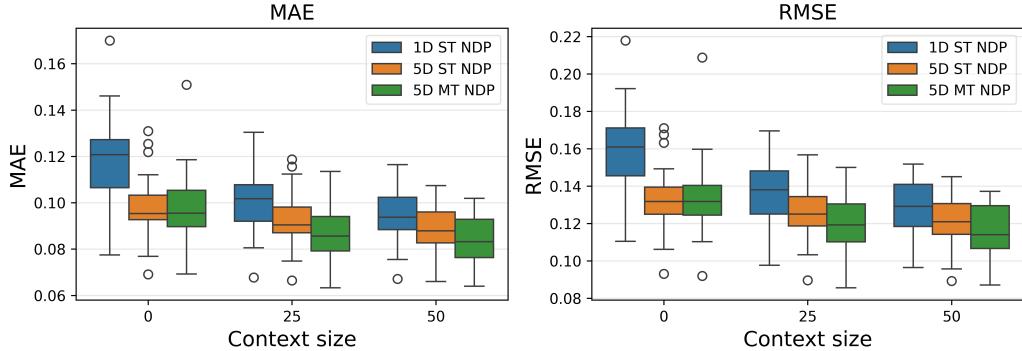
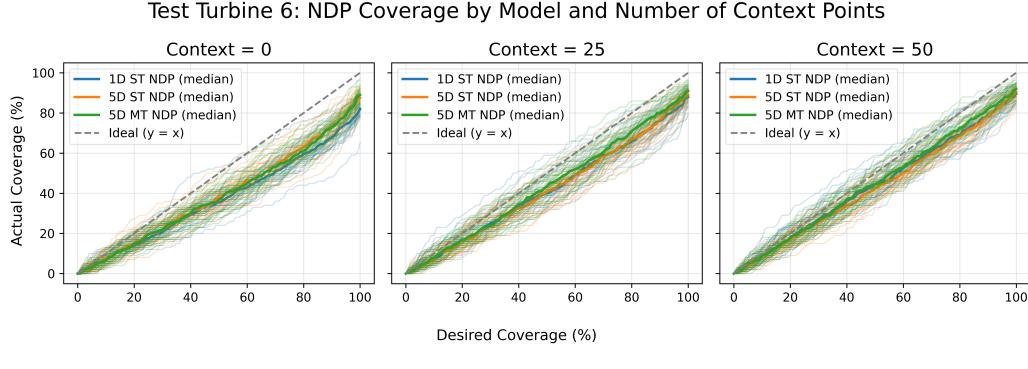


Figure 12: Mean absolute error (MAE, left) and root mean squared error (RMSE, right) distributions for the one-dimensional single-task neural diffusion process (1D ST-NDP), five-dimensional single-task neural diffusion process (5D ST-NDP), and five-dimensional multi-task neural diffusion process (5D MT-NDP) on turbine 6 across context sizes 0, 25, and 50, aggregated over 30 test samples of 100 points each.

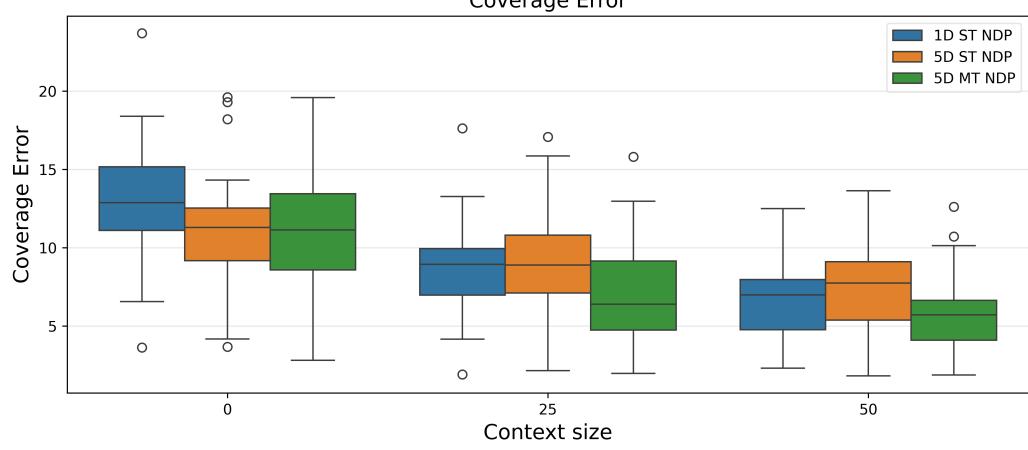
dimensional MT-NDP again exhibits the strongest performance, achieving lower CE values than both the one-dimensional and five-dimensional ST-NDPs, particularly at higher context sizes. These results confirm that multi-task extension improves both point prediction accuracy and uncertainty calibration in a challenging case of turbine 6.

Table 2 reports quantitative results for turbine 6 across all models and context sizes. Several trends are evident. First, adding context points consistently improves both accuracy (MAE, RMSE) and calibration (CE) for all models, confirming the value of conditioning on task-specific data. Second, increasing the feature set from one to five dimensions reduces prediction error, with the five-dimensional ST-NDP outperforming its one-dimensional counterpart in most cases. Finally, the MT-NDP achieves the lowest errors overall: with 50 context points, it attains both the best point accuracy and the most well-calibrated uncertainty, reducing coverage error relative to the ST-NDP. These results demonstrate the advantage of the multi-task architecture in adapting to the distinct behaviour of turbine 6 and highlight its ability to perform few-shot inference.



(a) Observed vs. nominal coverage.

Test Turbine 6: NDP Coverage Error by Model and Number of Context Points



(b) Coverage error distributions.

Figure 13: Predictive uncertainty performance of the one-dimensional single-task neural diffusion process (1D ST-NDP), five-dimensional single-task neural diffusion process (5D ST-NDP), and five-dimensional multi-task neural diffusion process (5D MT-NDP) on turbine 6 across context sizes 0, 25, and 50, aggregated over 30 test samples of 100 points each. (a) Observed vs. nominal coverage. (b) Coverage error distributions.

7. Discussion

7.1. Key findings

Our results show three consistent patterns. First, in the one-dimensional setting where wind speed is the sole input feature, NDPs match GPs in point accuracy while providing markedly better uncertainty calibration. As seen in Fig. 7, MAE and RMSE distributions are similar for the two models, but the

Table 2: Out-of-sample performance of single-task neural diffusion process (ST-NDP) and multi-task neural diffusion process (MT-NDP) models on turbine 6, averaged over 30 test functions (100 points each). Metrics reported: mean absolute error (MAE, kW), root mean squared error (RMSE, kW), and coverage error (CE, %).

<i>Model</i>	<i># of features</i>	<i>Context</i>	<i>MAE (kW)</i>	<i>RMSE (kW)</i>	<i>CE (%)</i>
ST-NDP	1	0	66.539	90.491	13.052
		25	56.641	77.766	8.779
		50	53.283	73.429	6.865
ST-NDP	5	0	55.169	75.441	11.225
		25	51.864	71.306	9.058
		50	49.676	68.465	7.376
MT-NDP	5	0	55.291	75.974	11.375
		25	49.183	67.805	6.986
		50	47.268	65.636	5.691

NDP’s empirical coverage lies substantially closer to the $y = x$ line than the GP’s, which is underconfident across coverage levels. This aligns with prior evidence that GP Gaussianity can yield miscalibrated intervals on SCADA data.

Second, increasing the input dimensionality from one dimension to five dimensions, i.e., by adding sine/cosine wind direction, nacelle temperature, and transformer temperature, improves point accuracy for NDPs, and the benefit grows with more context. Fig. 8 shows lower MAE/RMSE for the five-dimensional NDP model, with clear gains as context increases from 0 to 50 points. Calibration also improves with context (Fig. 10), narrowing the gap to the one-dimensional NDP model at 50 points. These trends suggest that richer covariates and larger context sets are complementary: the former increase explanatory power; the latter help the model specialise to the test turbine.

Third, extending NDPs to the multi-task setting (MT-NDP) yields the strongest performance. The five-dimensional MT-NDP outperforms the one-dimensional and five-dimensional ST-NDPs in both accuracy and calibration as the context grows (Fig. 12, 13; Table 2), demonstrating an effective adaptation of a few shots to the behaviour of the out-of-distribution turbine.

7.2. Limitations

Our experimental design has three main limitations. (i) *Context budget*: we evaluated context sizes up to 50 points. Given the consistent gains for the five-dimensional NDP models, larger context sets could further improve calibration and may even overturn the slight coverage advantage occasionally observed for the one-dimensional model at 50 points. (ii) *Encoder simplicity*: the task encoder is an MLP chosen for input-size agnosticity and permutation equivariance via averaging; it does not guarantee invariance across feature orderings as attention-based designs do. (iii) *Data homogeneity*: the Kelmarsh turbines are relatively homogeneous in the one-dimensional regime, which limits how much benefit context can bring and how strongly multi-task methods are stress-tested. Additionally, rare operating regions can produce occasional loss outliers, suggesting that stratified sampling could help.

7.3. Future work

Two immediate extensions are computational rather than conceptual: (i) increasing the context budget for higher-dimensional models and (ii) expanding the feature set beyond five SCADA variables by leveraging GPUs for training and sampling. Beyond these, we identify three methodological directions: (a) attention-based or bi-dimensional attention encoders to enforce invariance and capture richer task structure; (b) time-aware NDP variants that account for temporal dependence within functions; and (c) broader multi-farm evaluations—including onshore and offshore sites with diverse turbine types—to assess robustness and transferability.

7.4. Implications

Taken together, the results indicate that NDPs provide calibrated, uncertainty-aware predictions, scale effectively to richer inputs, and—when equipped with a task encoder—deliver few-shot adaptation even to turbines that deviate from the fleet average. In operational settings, this combination of accuracy and calibration matters: sharper yet trustworthy intervals support dispatch, bidding, and maintenance decisions under uncertainty. Focusing on genuinely distinct turbines, such as turbine 6 in our study, offers the clearest evidence of these advantages in practice.

8. Conclusion

This paper has presented the first empirical evaluation of NDPs and their multi-task extension for wind power prediction from SCADA data. We have

shown that NDPs match the predictive accuracy of GPs while substantially outperforming them in terms of calibrated uncertainty estimates. Unlike GPs, NDPs scale efficiently to higher-dimensional input spaces and remain tractable on large datasets, making them suitable for real-world deployment.

A key strength of NDPs is their ability to generalise to unseen turbines with only a handful of context points, enabling few-shot inference in settings where historical data are scarce. Extending the framework to the multi-task case, we introduced an MT-NDP architecture with a task encoder and demonstrated that it can exploit cross-turbine correlations to further improve accuracy and uncertainty calibration. These benefits are particularly evident on wind turbines that deviate most strongly from the fleet average, where single-task models often struggle or cannot be applied effectively.

From an operational perspective, these findings are highly relevant for wind farm monitoring and forecasting. Reliable uncertainty quantification supports decision-making for grid integration, market participation, and maintenance scheduling. By improving predictive robustness and reducing reliance on large task-specific datasets, NDP-based models could help reduce downtime and operating costs in modern wind farms.

Future work will focus on scaling MT-NDPs to the full set of SCADA variables, incorporating temporal dependencies to better capture dynamic turbine behaviour, and testing on more heterogeneous wind farms, including offshore installations. Enhancing the task encoder with attention mechanisms also offers a promising direction for capturing richer cross-turbine structure.

References

- [1] J. Maldonado-Correa, S. Martín-Martínez, E. Artigao, E. Gómez-Lázaro, Using scada data for wind turbine condition monitoring: A systematic literature review, *Energies* 13 (12) (2020) 3132.
- [2] Y. Hadjoudj, R. Pandit, A review on data-centric decision tools for offshore wind operation and maintenance activities: Challenges and opportunities, *Energy Science & Engineering* 11 (4) (2023) 1501–1515.
- [3] A. Stetco, F. Dinmohammadi, X. Zhao, V. Robu, D. Flynn, M. Barnes, J. Keane, G. Nenadic, Machine learning methods for wind turbine condition monitoring: A review, *Renewable energy* 133 (2019) 620–635.

- [4] X. Chesterman, T. Verstraeten, P.-J. Daems, A. Nowé, J. Helsen, Overview of normal behavior modeling approaches for scada-based wind turbine condition monitoring demonstrated on data from operational wind farms, *Wind Energy Science* 8 (6) (2023) 893–924.
- [5] Y. Yu, X. Han, M. Yang, J. Yang, Probabilistic prediction of regional wind power based on spatiotemporal quantile regression, in: 2019 IEEE industry applications society annual meeting, IEEE, 2019, pp. 1–16.
- [6] Y. Yu, M. Yang, X. Han, Y. Zhang, P. Ye, A regional wind power probabilistic forecast method based on deep quantile regression, *IEEE Transactions on Industry Applications* 57 (5) (2021) 4420–4427.
- [7] Y. Zhou, Y. Sun, S. Wang, R. J. Mahfoud, H. H. Alhelou, N. Hatziargyriou, P. Siano, Performance improvement of very short-term prediction intervals for regional wind power based on composite conditional nonlinear quantile regression, *Journal of Modern Power Systems and Clean Energy* 10 (1) (2021) 60–70.
- [8] H. Zhang, Y. Liu, J. Yan, S. Han, L. Li, Q. Long, Improved deep mixture density network for regional wind power probabilistic forecasting, *IEEE Transactions on Power Systems* 35 (4) (2020) 2549–2560.
- [9] W. Liao, Z. Yang, X. Chen, Y. Li, Windgmmn: Scenario forecasting for wind power using generative moment matching networks, *IEEE Transactions on Artificial Intelligence* 3 (5) (2021) 843–850.
- [10] C. K. Williams, C. E. Rasmussen, Gaussian processes for machine learning, Vol. 2, MIT press Cambridge, MA, 2006.
- [11] T. Rogers, P. Gardner, N. Dervilis, K. Worden, A. Maguire, E. Papatheou, E. Cross, Probabilistic modelling of wind turbine power curves with application of heteroscedastic gaussian process regression, *Renewable Energy* 148 (2020) 1124–1136.
- [12] R. K. Pandit, D. Infield, A. Kolios, Gaussian process power curve models incorporating wind turbine operational variables, *Energy Reports* 6 (2020) 1658–1669.

- [13] N. Chen, Z. Qian, I. T. Nabney, X. Meng, Wind power forecasts using gaussian processes and numerical weather prediction, *IEEE Transactions on Power Systems* 29 (2) (2013) 656–665.
- [14] R. K. Pandit, D. Infield, Scada-based wind turbine anomaly detection using gaussian process models for wind turbine condition monitoring purposes, *IET Renewable Power Generation* 12 (11) (2018) 1249–1255.
- [15] F. Fiocchi, D. Ladopoulos, P. Dellaportas, Probabilistic multilayer perceptrons for wind farm condition monitoring, *Wind Energy* 28 (4) (2025) e70012.
- [16] M. Garnelo, D. Rosenbaum, C. Maddison, T. Ramalho, D. Saxton, M. Shanahan, Y. W. Teh, D. Rezende, S. A. Eslami, Conditional neural processes, in: International conference on machine learning, PMLR, 2018, pp. 1704–1713.
- [17] H. Kim, A. Mnih, J. Schwarz, M. Garnelo, A. Eslami, D. Rosenbaum, O. Vinyals, Y. W. Teh, Attentive neural processes, arXiv preprint arXiv:1901.05761 (2019).
- [18] D. Kim, S. Cho, W. Lee, S. Hong, Multi-task neural processes, arXiv preprint arXiv:2110.14953 (2021).
- [19] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, S. Ganguli, Deep unsupervised learning using nonequilibrium thermodynamics, in: International conference on machine learning, PMLR, 2015, pp. 2256–2265.
- [20] J. Ho, A. Jain, P. Abbeel, Denoising diffusion probabilistic models, *Advances in neural information processing systems* 33 (2020) 6840–6851.
- [21] A. Lugmayr, M. Danelljan, A. Romero, F. Yu, R. Timofte, L. Van Gool, Repaint: Inpainting using denoising diffusion probabilistic models, in: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 2022, pp. 11461–11471.
- [22] V. Dutordoir, A. Saul, Z. Ghahramani, F. Simpson, Neural diffusion processes, in: International Conference on Machine Learning, PMLR, 2023, pp. 8990–9012.

- [23] A. N. Kolmogorov, Foundations of the theory of probability: Second English Edition, Courier Dover Publications, 2018.
- [24] A. Q. Nichol, P. Dhariwal, Improved denoising diffusion probabilistic models, in: International conference on machine learning, PMLR, 2021, pp. 8162–8171.
- [25] C. Plumley, Kelmarsh wind farm data (0.0. 3), Zenodo, February (2022).
- [26] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv preprint arXiv:1412.6980 (2014).