Neural Computing and Applications (2022) 34:6171–6186
https://doi.org/10.1007/s00521-021-06799-6

ORIGINAL ARTICLE

Data-driven wind turbine wake modeling via probabilistic machine learning

S. Ashwin Renganathan[1] · Romit Maulik[2] · Stefano Letizia[3] · Giacomo Valerio Iungo[3]

Received: 25 August 2021 / Accepted: 25 November 2021 / Published online: 11 January 2022

Abstract
Wind farm design primarily depends on the variability of the wind turbine wake flows to the atmospheric wind conditions and the interaction between wakes. Physics-based models that capture the wake flow field with high-fidelity are computationally very expensive to perform layout optimization of wind farms, and, thus, data-driven reduced-order models can represent an efficient alternative for simulating wind farms. In this work, we use real-world light detection and ranging (LiDAR) measurements of wind-turbine wakes to construct predictive surrogate models using machine learning. Specifically, we first demonstrate the use of deep autoencoders to find a low-dimensional *latent* space that gives a computationally tractable approximation of the wake LiDAR measurements. Then, we learn the mapping between the parameter space and the (latent space) wake flow fields using a deep neural network. Additionally, we also demonstrate the use of a probabilistic machine learning technique, namely, Gaussian process modeling, to learn the parameter-space-latent-space mapping in addition to the epistemic and aleatoric uncertainty in the data. Finally, to cope with training large datasets, we demonstrate the use of variational Gaussian process models that provide a tractable alternative to the conventional Gaussian process models for large datasets. Furthermore, we introduce the use of active learning to adaptively build and improve a conventional Gaussian process model predictive capability. Overall, we find that our approach provides accurate approximations of the wind-turbine wake flow field that can be queried at an orders-of-magnitude cheaper cost than those generated with high-fidelity physics-based simulations.

Keywords Machine Learning · Gaussian process · Deep neural networks · Wind energy

1 Introduction

Understanding and modeling wind farm flows still represent major challenges for wind farm designers and operators. Important aspects, such as the interaction of the atmospheric boundary layer with wind turbines [41, 46] and prediction of the wake morphology and their superposition [30], are far from being fully understood, in spite of having been the object of numerous insightful scientific studies.

The term turbine wake refers to the low momentum and highly turbulent region located downstream of an operating wind turbine, which is a direct consequence of the extraction of kinetic energy from the incoming wind field. Significant power losses [2, 13, 42] and enhanced fatigue loads [7, 10] were documented for turbines impinged by upstream wakes. The study of turbine wakes through numerical simulations is encumbered with difficulties due to the high Reynolds number flow (which entails a great span of length and time scales involved) [30], the unsteadiness of the inflow conditions [19, 23], and the relevant role of atmospheric stability [51]. Recently, large-eddy simulations (LES) have become a well-established tool for the simulation of wind farm flows [5, 28, 40]. However, their computational costs are still prohibitive for

✉ S. Ashwin Renganathan
ashwin.renganathan@utah.edu

[1] Department of Mechanical Engineering, The University of Utah, Salt Lake City, UT, USA

[2] Mathematics and Computer Science, Argonne National Laboratory, Lemont, IL, USA

[3] Wind Fluids and Experiments (WindFluX) Laboratory, Department of Mechanical Engineering, The University of Texas at Dallas, Dallas, TX, USA

large-scale tasks, such as for layout optimization and real-time performance diagnostics.

Reynolds-averaged Navier-Stokes [20, 38] and engineering wake models (e.g. [3, 14, 21]) provide faster and simpler alternatives to LES, since they do not explicitly solve the unsteady small-scale turbulent eddies. However, while the former are oftentimes affected by inaccuracy due to the turbulence closure [38], the latter require a thorough calibration to achieve a satisfactory agreement with experimental data [50].

The above-mentioned challenges have spurred the interest of wind energy scientists in the experimental characterization of the wind farm flow. In particular, the improvements of remote sensing instruments, such as wind light detection and ranging (LiDAR), have promoted the proliferation of field experimental campaign investigating the wakes of utility-scale wind turbines (e.g., see [19, 24, 50]). These studies have highlighted the great complexity and sensitivity to environmental conditions on the characteristics and downstream evolution of wind turbine wakes.

Following up on past work [27], in this work, LiDAR measurements of individual wakes generated by utility-scale wind turbines under broad ranges of atmospheric and wind conditions are leveraged to develop data-driven machine learning models to enable accurate predictions of the wake velocity field for prescribed wind/atmospheric conditions, while requiring computational costs as low as for empirical wake engineering models. Specifically, we explore the application of deep neural networks (DNN) for efficient data reduction via autoencoders, as well as to build predictive models via multilayer perceptrons. Furthermore, we also explore the combination of the data reduction via the DNN and Gaussian process (GP) models to develop predictive probabilistic models of the wake flow. This way, we show how probabilistic machine learning can be leveraged to learn predictive wake models from noisy and incomplete measurement data. To summarize, the contributions of this article are:

1. A novel convolutional autoencoder framework to obtain low-dimensional embeddings of wind LiDAR measurements for wind-turbine wakes. In this manner, we develop a compressed—and hence tractable—representation of the high-dimensional LiDAR data.
2. A DNN to learn the mapping between the input parameter space and the latent space derived from the convolutional autoencoder. Therefore, we learn the mapping between the input parameters and the spatially distributed wake measurements, via two levels of DNNs–one for data reduction and one for prediction. This serves as a cheap-to-evaluate surrogate model to predict wind turbine wake velocity fields.
3. Additionally, similar to item 2, we propose the use of probabilistic machine learning model via GP regression to learn latent-space to wake flow field mapping, which can simultaneously learn the noise in the data. To address the tractability issues associated with GP regression with *big* data, we also investigate the use of *variational* and active learned GP regression.

Overall, this work brings together state-of-the-art data-driven machine learning and the classic field of wind energy to build cheap and reliable surrogate models that can be leveraged to perform exploratory studies. Figure 1 provides a high-level summary of the work.

The remainder of this article is organized as follows. In Sect. 2, we present the details of our experimental setup and data collection via LiDAR. In sections Sects. 3 and 4 we present the theoretical details of our machine learning methods. We present the results and associated discussion in Sect. 5, followed by concluding remarks and an outlook on future work in Sect. 6.

2 Experimental data and collection methods

The wind LiDAR measurements used in this work were collected in the period from August 2015 to March 2017 at a wind farm located in North Texas[1]. The wind farm includes 25 identical wind turbines with a nameplate capacity of 2.3 MW. The rotor diameter is $d = 127$ m and the height of the hub is 89 m above the ground. The local topographic map provided by [45] with a resolution of 100 m shows that 95% of the terrain within the farm has a slope lower than 3°, which allows to rule out effects due to the terrain on the flow.

A WindCube 200S scanning pulsed Döppler LiDAR was installed at turbine 11 (see Fig. 2) and probed the wakes stemming from turbines 01 to 06 during the occurrence of southerly winds. Plan position indicator (PPI) scans were scheduled targeting the wake of the turbine with the best line-of-sight alignment with the LiDAR, which enables achieving an optimal spatio-temporal sampling resolution.

Furthermore, meteorological and SCADA data were continuously collected for the whole duration of the campaign in the form of 10-minute mean and standard deviation of wind speed, wind direction, temperature, atmospheric pressure, active power, RPM, and blade pitch angle. The list of input parameters is presented in Table 1; for a more detailed description of the experimental site and LiDAR scanning strategy please refer to [13, 51].

[1] The original LiDAR data used in this work are available upon reasonable request from the fourth author, who may be contacted at valerio.iungo@utdallas.edu.

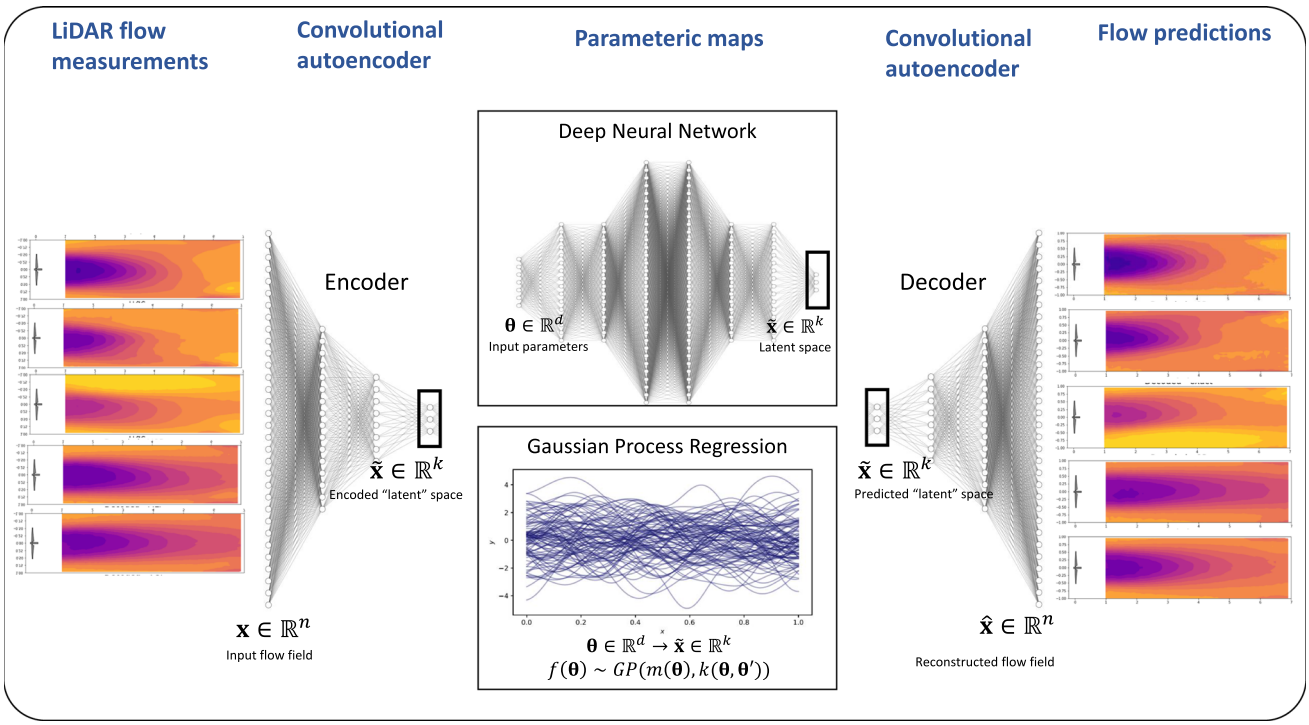


Fig. 1 Summary of the overall methodology

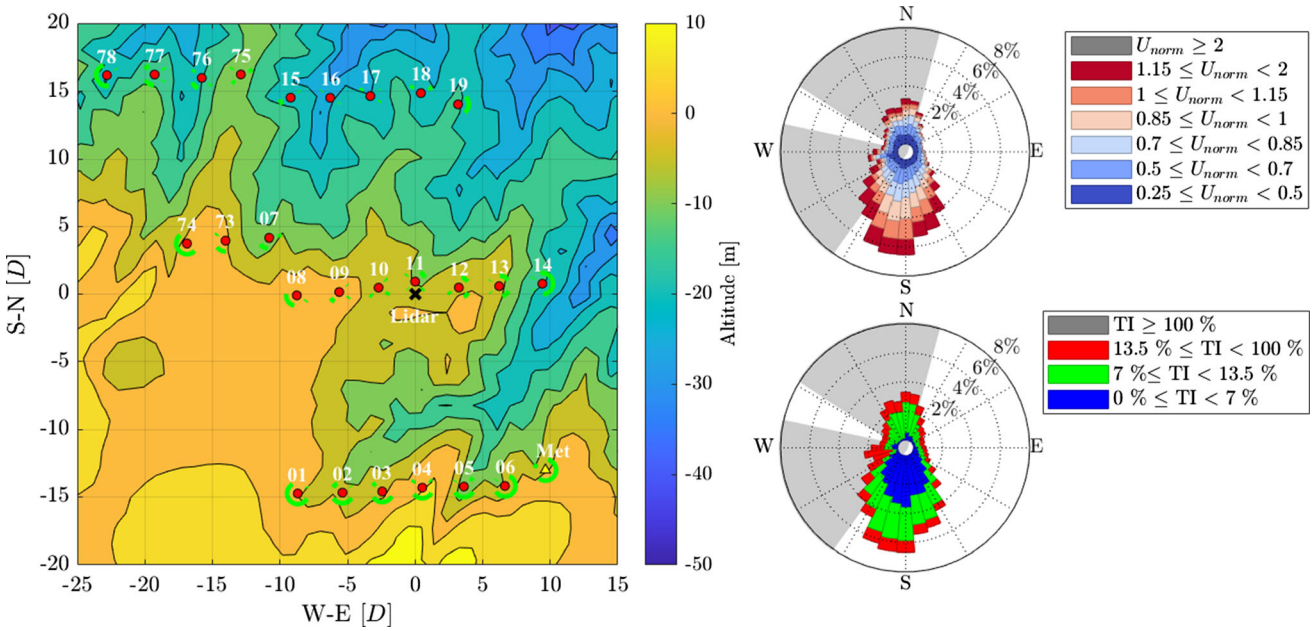


Fig. 2 Map of the experimental site. On the left, is the topographic map and farm layout, where the green arcs indicate wind sectors where no relevant wake interactions are expected. On the right, it is the directional histogram of wind speed and turbulence intensity based on the meteorological data of the 80-meters tower at the "Met" location. U_{norm} refers to the hub-height wind speed normalized by the rated wind speed of the turbines (11 m s^{-1}). Grey sectors are likely affected by wakes (Color figure online)

The wind rose of the site (Fig. 2) indicates high likelihood of southerly winds and a negligible directional dependence of the turbulence intensity at hub-height, which confirms the findings of previous studies [13, 51] on the prevalent thermally-driven cycle of the atmospheric boundary layer caused by the diurnal variation of the solar irradiance.

Table 1 List of input parameters and their ranges

Parameter	Range	Description
SCADA_WS [m/s]	[2.92, 15.22]	Mean Hub-height wind speed recorded by the SCADA
MET_WS_80m [m/s]	[3.9, 15.22]	Mean Hub-height wind speed recorded by the met-tower
SCADA_TI [–]	[0.04, 0.36]	Hub-height wind turbulence intensity recorded by the SCADA
MET_BulkRichardson [–]	$[-0.01, 0.01]$	Bulk Richardson number recorded by met-tower
SCADA_Power [kW]	[58.8, 2423]	Mean power capture recorded by the SCADA
SCADA_RPM [RPM]	[7.07, 16.95]	Mean rotor rotational speed recorded by the SCADA
SCADA_Pitch [°]	$[-2, 80]$	Mean blade pitch angle recorded by the SCADA

The LiDAR data undergo a quality control which excludes all the points characterized by a carrier-to-noise ratio lower than -25 dB. Subsequently, the wake data are realigned with the wind direction, which is estimated as the 10-minute moving averaged wake direction. This also allows estimating the horizontal equivalent velocity as:

$$u_{eq} \sim \frac{u_{\mathrm{LOS}}}{\cos(\theta - \theta_w)\cos\beta}, \tag{1}$$

where u_{LOS} is the line-of-sight velocity measured by the LiDAR, θ_w is the wind direction, θ and β are the azimuth and elevation angles of the LiDAR, respectively. The vertical variability of the incoming wind due to wind shear is corrected by normalizing the equivalent velocity by the vertical undisturbed velocity profile. After the outlined post-processing, 6654 quality-control re-aligned and non-dimensional LiDAR scans are made available for the following analysis. For more technical details on this procedure, the interested reader shall refer to [51].

3 Deep learning for parametric flow prediction

In the following section, we introduce our deep neural network architectures for establishing a viable emulation strategy for data obtained from LiDAR measurements.

3.1 Convolutional autoencoder

Autoencoders are neural networks that learn a new representation of the input data, usually with lower dimensionality. The initial layers, called the *encoder*, map the input $\mathbf{x} \in \mathbb{R}^n$ to a new representation $\tilde{\mathbf{x}} \in \mathbb{R}^k$ with $k \ll n$. The remaining layers, called the *decoder*, map $\tilde{\mathbf{x}}$ back to $\mathbb{R}^n$ with the goal of reconstructing $\mathbf{x}$. The objective is to minimize the reconstruction error. Autoencoders are unsupervised; the data $\mathbf{x}$ is given, but the representation $\tilde{\mathbf{x}}$ must be learned.

More specifically, we use autoencoders that have convolutional layers. In a convolutional layer, instead of learning a matrix that connects all m neurons of layer's input to all n neurons of the layer's output, we learn a set of filters that are convolved with regions of the layer's input. Suppose a one-dimensional (1-d) convolutional layer has filters of length L_f, then, each of the layer's output neurons corresponding to a specific filter $\mathbf{f^i}$ is connected to a patch of L_f of the layer's input neurons. In particular, a 1-d convolution of filter $\mathbf{f^i}$ and patch $\mathbf{p}$ is defined as $\mathbf{f^i} * \mathbf{p} = \sum_j f_j^i p_j$ (where f_j^i corresponds to the stencil coefficient in the filter for index j). In other words, convolutional neural networks identify stencil values f_j^i that obtain coherent translationally invariant features relevant to a particular function approximator. Then, for a typical 1-d convolutional layer, the layer's output neuron $y_{ij} = \varphi(\mathbf{f^i} * \mathbf{p_j} + B_i)$ where φ is an activation function, and B_i are the entries of a bias term. As $\mathbf{j}$ increases, patches are shifted by stride s. For example, a 1-d convolutional layer with a filter $\mathbf{f^0}$ of length $m_f = 3$ and stride $s = 1$ could be defined so that y_{0j} involves the convolution of $\mathbf{f^0}$ and inputs $\mathbf{j} - 1, \mathbf{j}$, and $\mathbf{j} + 1$. To calculate the convolution, it is common to add zeros around the inputs to a layer, which is called *zero padding*. In the decoder, we use deconvolutional layers to return to the original dimension. These layers upsample with nearest-neighbor interpolation.

Two-dimensional convolutions are defined similarly, but each filter and each patch are two-dimensional. A 2-d convolution sums over both dimensions, and patches are shifted both ways. For a typical 2-d convolutional layer, the output neuron $y_{hij} = \varphi(\mathbf{f^h} * \mathbf{p_{ij}} + B_h)$. Input data can also have a "channel" dimension, such as red/green/blue values for images. The convolutional operator sums over channel dimensions, but each patch contains all of the channels. The filters remain the same size as patches, so they can have different weights for different channels. It is common to follow a convolutional layer with a *pooling* layer, which outputs a sub-sampled version of the input. In this paper,

we specifically use max-pooling layers. Each output of a max-pooling layer is connected to a patch of the input, and it returns the maximum value in the patch.

Autoencoders have recently become popular for the nonlinear dimensionality reduction of datasets extracted from several high-dimensional systems. These have been motivated by the extraction of coherent structures that parameterize low-dimensional embeddings in manifolds [15, 29, 47], and the utilization of these embeddings for efficient surrogate models of nonlinear dynamical systems [6, 17, 22, 26, 33, 35, 37, 49]. In this work, we utilize convolutional autoencoders to identify low-dimensional representations of experimentally collected data for building parameter-observation maps where the former are obtained through meteorological and wind turbine data and the latter are LiDAR measurements collected in the wake generated by wind turbines.

3.2 Multilayered perceptron (MLP)

One technique to obtain a mapping from the meteorological and turbine datasets and the latent space embeddings of the convolutional autoencoder is through the use of a multilayered perceptron (MLP) architecture, which is a subclass of feedforward artificial neural network. A general MLP consists of several neurons arranged in multiple layers. These layers consist of one input and one output layer along with several hidden layers. Each layer (with the exception of an input layer) represents a linear operation followed by a nonlinear activation that allows for great flexibility in representing complicated nonlinear mappings. This may be expressed as

$$\mathcal{L}^l\left(\boldsymbol{t}^{l-1}\right) := \boldsymbol{w}^l \boldsymbol{t}^{l-1} + \boldsymbol{b}^l, \tag{2}$$

where $\boldsymbol{t}^{l-1}$ is the output of the previous layer, and $\boldsymbol{w}^l, \boldsymbol{b}^l$ are the weights and biases associated with that layer. The output $\boldsymbol{t}^l$, for each layer may then be transformed by a nonlinear activation, such as rectified linear activation:

$$\eta(a) = \mathrm{ReLU}(a) = \max(a, 0). \tag{3}$$

For our experiments, the inputs $\boldsymbol{t}^0$ are in $\mathbb{R}^d$ (i.e., d is the number of inputs) and the outputs $\boldsymbol{t}^K$ are in $\mathbb{R}^k$ (i.e., k is the number of outputs). The final map is given by

$$F : \mathbb{R}^d \mapsto \mathbb{R}^k, \quad \boldsymbol{t}^0 \mapsto \boldsymbol{t}^K = F(\boldsymbol{t}^0; (\boldsymbol{w}, \boldsymbol{b})),$$

where

$$F(\boldsymbol{t}^0; \boldsymbol{w}, \boldsymbol{b}) = \eta^K\left(\mathcal{L}^K \circ \eta^{K-1}_{\mathrm{act}} \circ \mathcal{L}^{K-1} \circ \ldots \circ \eta^1_{\mathrm{act}} \circ \mathcal{L}^1\right)(\boldsymbol{t}^0) \tag{4}$$

is a complete representation of the neural network and where $\mathbf{w}$ and $\mathbf{b}$ are a collection of all the weights and the biases of the neural network. These weights and biases, lumped together as $\phi = \{\mathbf{w}, \mathbf{b}\}$, are trainable parameters of our map, which can be optimized by examples obtained from a training set. The supervised learning framework requires for this set to have examples of inputs in $\mathbb{R}^{N_{ip}}$ and their corresponding outputs $\mathbb{R}^{N_{op}}$. This is coupled with a cost function $\mathcal{C}$, which is a measure of the error of the prediction of the network and the ground truth. Our cost function is given by

$$\mathbb{C} = \frac{1}{|T|} \sum_{(\boldsymbol{\theta}, \tilde{\mathbf{x}}) \in T} \|\tilde{\mathbf{x}} - F(\boldsymbol{\theta}; \phi)\|^2 \tag{5}$$

with $|T|$ indicates the cardinality of the training dataset given by

$$T = \{(\boldsymbol{\theta}_i, \tilde{\mathbf{x}}_i) : \tilde{\mathbf{x}}_i = f(\boldsymbol{\theta}_i)\}.$$

and where $f(\boldsymbol{\theta}_i)$ are examples of the true targets obtained from the compressed training data using the autoencoder introduced in the previous sections. Gradients of this cost function can then be used in an optimization framework to obtain the best weights and biases, given the training data. Finally, the trained MLP may be used for uniformly approximating any continuous function on compact domains [1, 12], provided $\eta(x)$ is not polynomial in nature.

4 Gaussian process regression

We now review the preliminaries of GP models. Our primary interest in the use of GP models stems from its promise of offering enhanced data efficiency in emulation compared to DNNs [36] as well as in sequential decision-making [34]. Although not pursued in this work, GPs offer greater potential in emulating complex functions when combined with DNNs; e.g., see [31, 32]. GP models provide a probabilistic approximation to an unknown function $f(\boldsymbol{\theta})$. Specifically, $f(\boldsymbol{\theta})$ is assumed to take the form of a GP, where each realization (or sample path) is a function. This *prior* assumption on the function can then be combined with the probability of actual observations conditional on the prior (a.k.a, the *likelihood*), using Bayes' rule [16]. In this section, we provide a brief overview of the theory behind GP models and highlight the difference between exact and approximate inference, the latter finding applications in the presence of large datasets.

4.1 Exact Gaussian process regression

We begin by placing a GP prior assumption on the unknown function, that is, $f(\boldsymbol{\theta}) \sim \mathcal{GP}(\mu(\boldsymbol{\theta}), k(\boldsymbol{\theta}, \boldsymbol{\theta}'))$, where $\mu(\boldsymbol{\theta})$ is a mean function and $k(\boldsymbol{\theta}, \boldsymbol{\theta}')$ is a covariance function (or *kernel*), and $\boldsymbol{\Theta} \in \mathcal{T}$. We assume that we have noisy observations of $f(\boldsymbol{\theta})$ of the form

$$y_i = f(\boldsymbol{\theta}) + \epsilon_i, \quad i = 1, \ldots, n$$

where ϵ_i represents the observation noise. We assume ϵ_i to take an independent and identically distributed normal distribution with zero mean and a variance of σ_ϵ^2, that is, $\epsilon_i \sim \mathcal{N}(0, \sigma_\epsilon^2)$. Furthermore, we assume a Gaussian likelihood that defines the probability of observing the data given the GP assumption.

Let $\mathbf{y}_n = [y_1, \ldots, y_n]^\top$ denote the vector of noisy observations of f at $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_n]^\top$, and $\mathcal{D}_n = \{(\boldsymbol{\theta}_i, y_i), i = 1, \ldots, n\}$, then, applying Bayes' rule, the posterior distribution [48] is given by

$$\begin{aligned} f(\boldsymbol{\theta})|\mathcal{D}_n &\sim \mathcal{GP}(\mu_n(\boldsymbol{\theta}), \sigma_n^2(\boldsymbol{\theta})), \\ \text{where } \mu_n(\boldsymbol{\theta}) &= \mathbf{k}^\top \mathbf{K}^{-1}(\mathbf{y}_n - \mu(\boldsymbol{\Theta})) \\ \sigma_n^2(\boldsymbol{\theta}) &= k(\boldsymbol{\theta}, \boldsymbol{\theta}) - \mathbf{k}^\top \mathbf{K}^{-1}\mathbf{k}. \end{aligned} \tag{6}$$

In (6), μ_n and σ_n^2 are the mean and variance of the posterior distribution, $\mathbf{K} \subset \mathcal{K}$ is the $n \times n$ covariance matrix with $\mathbf{K}_{ij} = k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j), \forall \boldsymbol{\theta}_i, \boldsymbol{\theta}_j \in \boldsymbol{\Theta}$ and $\mathcal{K}$ being the cone of symmetric positive definite matrices, $\mu(\boldsymbol{\Theta}) = [\mu(\boldsymbol{\theta}_1), \ldots, \mu(\boldsymbol{\theta}_n)]^\top$, and $\mathbf{k} = [k(\boldsymbol{\theta}, \boldsymbol{\theta}_1), \ldots, k(\boldsymbol{\theta}, \boldsymbol{\theta}_n)]^\top$.

The emulation properties of the GP are driven by the choice of the mean and covariance functions. In this work, we standarize the observations $\mathbf{y}_n$ such that they have a mean of zero; that is we set $\mu(\boldsymbol{\theta}) = 0$ in the prior assumption. On the other hand, we use a covariance function from the Matern class [11, 25, 44], given by

$$k(\boldsymbol{\theta}, \boldsymbol{\theta}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|}{\ell} \right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|}{\ell} \right), \tag{7}$$

with positive parameters ν and ℓ, where K_ν is a modified Bessel function, $\Gamma()$ is the Gamma function and $\|\cdot\|$ denotes the Euclidean distance. The parameter ν (which we set to 3/2) controls the differentiability of the sample paths of the GP and is fixed, whereas ℓ is a *lengthscale* parameter that controls the rate of change of the sample paths in $\mathcal{T}$ and is a hyperparameter. The GP hyperparameter set $\Omega = \{\ell, \sigma_\epsilon^2\}$ is estimated via a maximum likelihood estimation (MLE) procedure frequently followed in fitting GP models [39, 48].

The fact that the posterior distribution of the function given in (6) is available in closed-form, the inference of such a model is called *exact* inference. One of the main bottlenecks of the exact inference is that the computation of the posterior mean and variance in (6) involves the inversion of the matrix $\mathbf{K}$, whose computational cost scales as $\mathcal{O}(n^3)$, and hence gets expensive as n increases. This motivates the *approximate* inference technique for GPs, namely, variational GP regression, which trades some accuracy for large gains in computational efficiency in fitting GP models when n is large.

4.2 Approximate Gaussian process regression

In addition to the cost of estimating hyperparameters of the exact GP involving $\mathcal{O}(n^3)$ floating point operations, the prediction with the GP costs $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$ operations, respectively, for the posterior mean and variance computation. Sparse GP models [43] circumvent this overhead by identifying a subset $m < n$ of the training points, resulting in *reduced* computational costs that scale as $\mathcal{O}(m^2 n)$, $\mathcal{O}(m)$, and $\mathcal{O}(m^2)$ for fitting, predicting mean, and predicting variance, respectively.

The choice of the subset of m inducing points can be treated as another hyperparameter and estimated by maximizing the marginal likelihood, just like in the exact GP model; this results in an extended set of hyperparameters $\{\Omega, \bar{\boldsymbol{\Theta}}\}$, where $\bar{\boldsymbol{\Theta}} \in \mathbb{R}^{m \times d}$ are the *inducing points*.

While sparse GPs bring down the cost of inverting the covariance matrix $\mathbf{K}$ and predictions with the GP, the number of hyperparameters increases (due to the addition of the inducing points), thereby making inference computationally more expensive. To overcome this, we use variational inference [4], which provides a tractable alternative to approximate unknown probability densities in Bayesian models. Below, we briefly provide an overview of sparse GP models and variational inference.

We now present a sparse model that is computationally tractable in terms of inference and prediction with GPs. The sparsity arises because we consider a sparse dataset $\bar{D}$ of size $m < n$ with inducing inputs $\bar{\boldsymbol{\Theta}} = \{\bar{\boldsymbol{\theta}}_i, i = 1, \ldots, m\} \subset \mathcal{T}$. These inducing inputs can either be a subset of the training inputs or can be randomly sampled from $\mathcal{T}$ [43]. Let $u = f(\bar{\boldsymbol{\theta}})$ be the inducing outputs, which are sampled from the same prior on the true function f. In this work, we treat $\bar{\boldsymbol{\Theta}}$ as hyperparameters and estimate them from maximizing the marginal likelihood.

Let the prior on the inducing outputs be given as

$$p(\mathbf{u}|\bar{\boldsymbol{\theta}}) \sim \mathcal{N}(0, \mathbf{K}_{mm}), \tag{8}$$

where $\mathbf{K}_{mm}$ is the matrix of covariances between the inducing inputs. This prior follows from the assumption that the inducing outputs also behave like the latent variable f which has a Gaussian prior. Therefore, $\mathbf{u}$ and $\mathbf{f}$ have a joint Gaussian distribution [18] given by

$$p(\mathbf{f}, \mathbf{u}) = p(\mathbf{f}|\mathbf{u})\mathbf{p}(\mathbf{u}) = \mathcal{N}(\mathbf{K_{nm}}\mathbf{K_{mm}^{-1}}\mathbf{u}, \mathbf{K_{nn}} - \mathbf{Q_{nn}}) \times \mathcal{N}(\mathbf{0}, \mathbf{K_{mm}}), \tag{9}$$

where $\mathbf{Q}_{nn} = \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{nm}^\top$. To estimate the extended

hyperparameter set $\{\Omega, \bar{\boldsymbol{\Theta}}\}$, we first need to define a marginal likelihood. In this case, the marginal likelihood is marginalized over $\mathbf{f}$ and $\mathbf{u}$, that is,

$$p(\mathbf{y}|\boldsymbol{\Theta}, \Omega, \bar{\boldsymbol{\Theta}}) = \int p(\mathbf{y}|\mathbf{u})p(\mathbf{u})d\mathbf{u} \tag{10}$$

Note that the density $p(\mathbf{y}|\mathbf{u})$ still involves inverting the matrix of size $n \times n$ and hence is expensive to compute. Therefore, this density is approximated using the evidence lower bound (ELBO) [4] as

$$p(\mathbf{y}|\mathbf{u}) \geq \mathbb{E}_{p(\mathbf{f}|\mathbf{u})}[p(\mathbf{y}|\mathbf{f})]. \tag{11}$$

Substituting (11) in (10), the lower bound on the marginal log likelihood can be approximated as [18]

$$p(\mathbf{y}|\boldsymbol{\Theta}, \Omega, \bar{\boldsymbol{\Theta}}) \geq \log \mathcal{N}(\mathbf{0}, \mathbf{K}_{nm}\mathbf{K}_{mm}^{-1}\mathbf{K}_{nm}^{\top} + \sigma_{\epsilon}^2\mathbf{I}) - \frac{1}{2}\mathrm{tr}(\mathbf{K}_{nn} - \mathbf{Q}_{nn}). \tag{12}$$

Equation 12 can be maximized with respect to $\bar{\boldsymbol{\Theta}}$ and Ω to estimate the hyperparameters, where the bound in (12) costs $\mathcal{O}(nm^2)$ for computation.

4.3 Active learning for Gaussian process regression

Whereas variational GPs provide an approximation to exact GP regression, another approach to improving tractability of exact GP models is active data selection [8, 9]. Specifically, given a training dataset $\mathcal{D}_n$, the dataset is adaptively augmented as

$$\mathcal{D}_{n+i} := \mathcal{D}_{n+i-1} \bigcup \{(\boldsymbol{\theta}_{n+i}, y_{n+i})\},\ i = 1, \ldots, m \tag{13}$$

where m is the number of adaptive model building steps. Furthermore, each adaptive step can select a batch of q training points jointly; when $q = 1$ we call the approach *sequential* active learning and when $q > 1$, *batch-sequential* active learning. At the end of the active learning process, the GP is trained with a total of $n + mq$ training points. The main objective of active learning is to choose points judiciously such that they are *optimal* in the sense of improving model fit.

In this work, we choose points that are optimal in reducing the overall uncertainty about the GP model. That is, we select points such that

$$\boldsymbol{\theta}_{n+1} = \arg\max_{\boldsymbol{\theta}' \in \mathcal{T}} \int_{\mathcal{T}} -\sigma_{n+1}^2(\boldsymbol{\theta}')\, d\boldsymbol{\theta}'$$

where $\sigma_{n+1}^2(\boldsymbol{\theta})$ is the posterior variance of the GP, having observed the $(n+1)$th point, and we introduce the negative sign to solve a maximization problem. Essentially, we treat the integrated posterior variance as a measure of uncertainty over our domain $\mathcal{T}$ and seek to choose training data that are optimal in minimizing this uncertainty.

A fundamental issue with the above equation is that the posterior variance $\sigma_{n+1}^2(\boldsymbol{\theta})$ is unknown until we actually commit to $\boldsymbol{\theta}_{n+1}$ and choose a training point there. To circumvent, we simulate the choice of $\boldsymbol{\theta}_{n+1}$ via the GP trained with data $\mathcal{D}_{n+1}$, and choose $\boldsymbol{\theta}_{n+1}$ as the point that reduces the *expected* uncertainty:

$$\begin{aligned}\boldsymbol{\theta}_{n+1} &= \arg\max_{\boldsymbol{\theta}' \in \mathcal{T}} -\int_{\mathcal{Y}} \int_{\mathcal{T}} \sigma_n^2(\boldsymbol{\theta}) | \mathcal{D}_n \cup \{\boldsymbol{\theta}', y(\boldsymbol{\theta}')\}\, d\boldsymbol{\theta} dy \\ &= \arg\max_{\boldsymbol{\theta}' \in \mathcal{T}} -\mathbb{E}_{y \sim Y_n}\left[\int_{\mathcal{T}} \sigma_n^2(\boldsymbol{\theta}) | \mathcal{D}_n \cup \{\boldsymbol{\theta}', y(\boldsymbol{\theta}')\}\, d\boldsymbol{\theta}\right].\end{aligned} \tag{14}$$

Essentially, (14) seeks to find the point in $\mathcal{T}$ that likely leads to the least overall uncertainty, if the corresponding training point was chosen and the model updated.

Similarly, the batch-sequential active design, to select $\Theta = \{\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_q\}$, is performed by choosing points as

$$\boldsymbol{\theta}_{n+1:q} = \arg\max_{\Theta'} -\mathbb{E}_{y \sim Y_n}\left[\int_{\mathcal{T}} \sigma_n^2(\boldsymbol{\theta}) | \mathcal{D}_n \cup \{\Theta', y(\Theta')\}\, d\Theta\right].$$

The choice of $q > 1$ particularly has advantages when the training data are generated by running expensive computer simulations, which can be evaluated synchronously in parallel. In the cases such as the present work, where we seek to actively select training data from an existing set, batch-sequential selection results in fewer hyperparameter training steps, which in the case of exact GPs scales as $\mathcal{O}(n^3)$. However, in terms of the improvement in model fit, it is not obvious what choice of q is the best. Therefore, we perform a simple sensitivity study to investigate the effect of the choice of q on the model fit.

Finally, to assess model fit, we evaluate the GP posterior mean on a hold-out test set $\{\boldsymbol{\theta}_i, f(\boldsymbol{\theta}_i)\}$, $i = 1, \ldots, n_{\text{test}}$, and compute the log root mean squared error (RMSE), defined as

$$\log(RMSE) = \log\left\{\frac{1}{n_{\text{test}}} \sum_{i=1}^{ntest} [\mu(\boldsymbol{\theta}_i) - f(\boldsymbol{\theta})_i]^2\right\}^{1/2}. \tag{15}$$

We note that the log(RMSE) is computed at each step of the active learning process with the GP model trained with the training data selected up to the previous step. Furthermore, the log(RMSE) is independently computed for each of the latent space outputs using the corresponding GP model.

5 Results

We now demonstrate our proposed data-driven machine learning approaches toward predicting the wake velocity field, using data generated from a scanning wind LiDAR.

We begin by first discussing the latent space reconstruction accuracy of the data, purely from the convolutional autoencoder. The reconstruction results provide a visual estimate of the trade-off between compression due to the autoencoder and loss/retention of information. With the latent space representation available from the autoencoder compression, we then proceed to learn the mapping between the inputs (operating conditions) and the latent space, via the machine learning models. Finally, the predicted wake fields are "decoded" via the decoder, as shown in Fig. 1.

5.1 Compression accuracy

We first examine the ability of the convolutional autoencoder to effectively compress the LiDAR observations to a suitable latent space. Figure 3 shows the ability of the autoencoder to reconstruct observed data, with only four latent dimensions, through its bottleneck neural architecture. The figures demonstrate that despite the drastic dimensionality reduction (i.e., 2501 to 4), the reconstruction accuracy has not been compromised significantly. However, the larger goal here is that we want to be able to generalize to a similar reconstruction error everywhere in the parameter space. For this, we introduce the results of the methods to obtain parameter-output maps, which we present next.

5.2 Parametric reconstruction

The latent space $\tilde{\mathbf{x}} \in \mathbb{R}^k$ provides a very concise encoding of the high-dimensional velocity field since $k \ll d$. Therefore, we learn the function $f : \theta \in \mathbb{R}^p \rightarrow \tilde{\mathbf{x}} \in \mathbb{R}^k$. We learn f via MLP and GP regression (with exact inference). Furthermore, we also use VGP regression (approximate inference) to improve the tractability of GP models for large datasets. Finally, we also show the performance of choosing training data via active learning for the exact GP.

The original LiDAR dataset has a dimensionality $d = 2501$ which is reduced to $k = 4$, via the convolutional autoencoder. Figure 4 shows the actual-vs-predicted plot of each of the four latent space dimensions predicted via all of the three machine learning models: MLP, GP, and VGP; note that we also show results of fitting GP with active learning used for training data selection. Firstly, the plots show that the predictive accuracy for all the machine learning models is somewhat similar. Secondly, there are outliers in the dataset—as in, for example, the lower-left and upper-right corners of subfigure (a), where the prediction accuracy is poor. We attribute this primarily to the fact that the LiDAR measurements are corrupted by noise, whose structure is unknown and not captured by the models. Even though the GP models do resolve the noise in the dataset, our model makes simplifying assumptions such as independent and identically distributed noise, which might not necessarily provide a realistic model of the noise (although they are relatively computationally more tractable). Furthermore, the raw measurements have missing

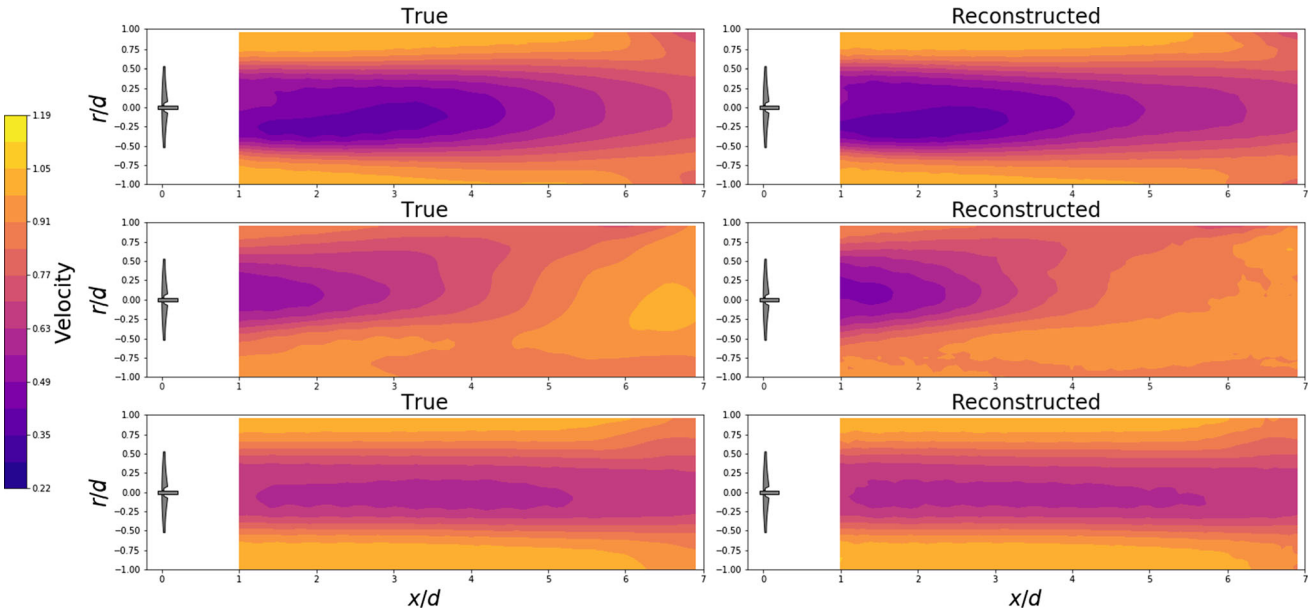


Fig. 3 The compressive effectiveness of the convolutional autoencoder on the LiDAR dataset for three test examples. The left column shows the true behavior of the wake for a set of testing data (unseen during model fit) and the right column shows reconstruction from a 4-dimensional latent space by the autoencoder

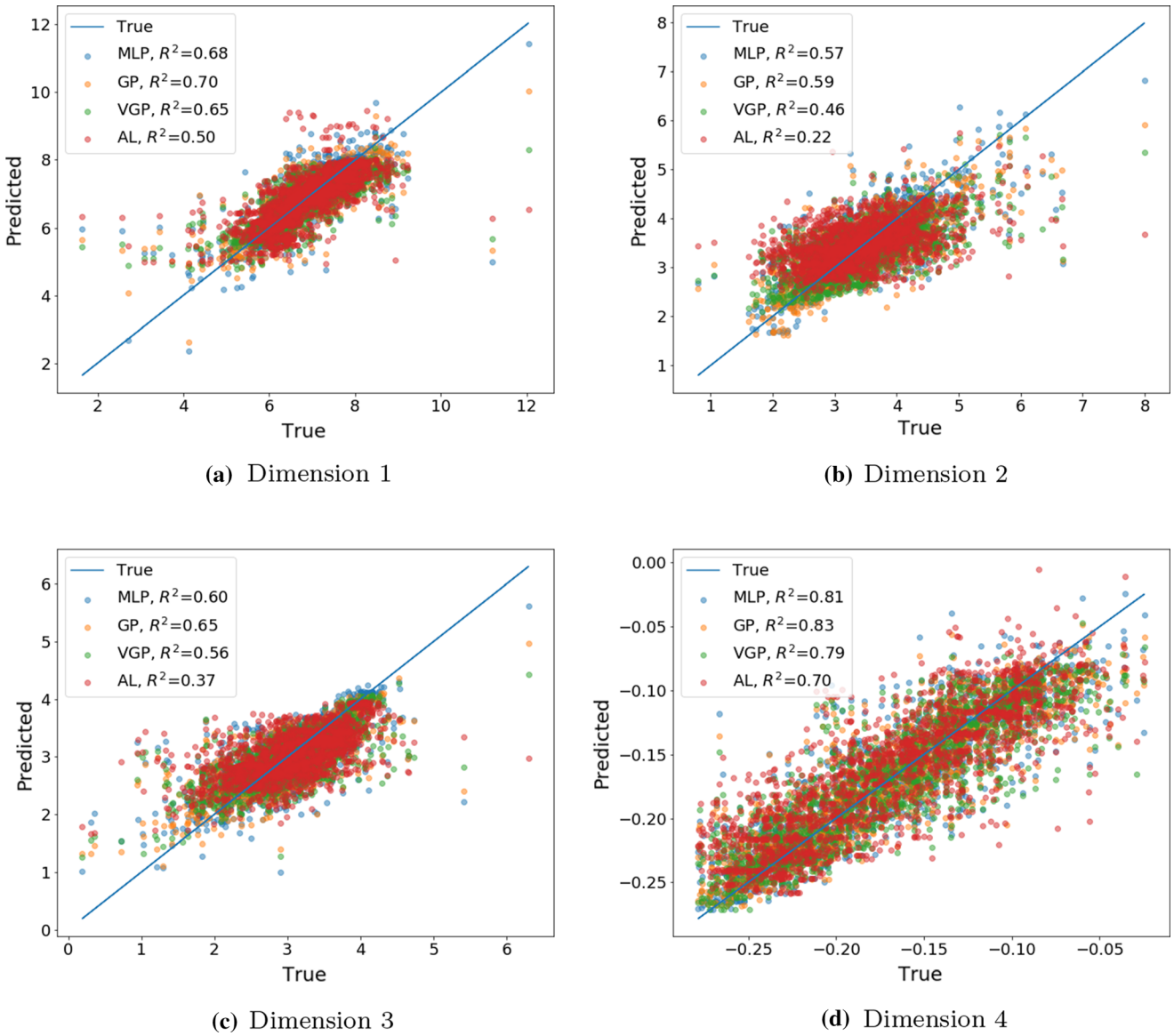
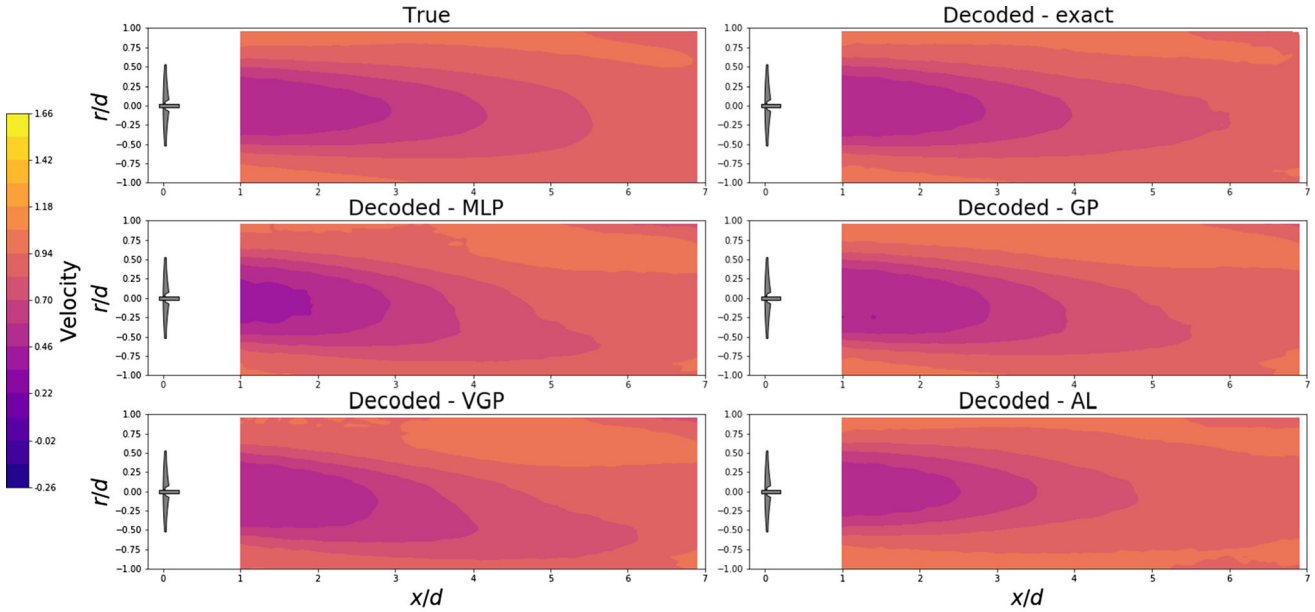


Fig. 4 Fitting a parametric map between MET data and the latent space representation extracted by the convolutional autoencoder on the LiDAR snapshots. The different colors indicate different latent space parametric mapping techniques. Similar results are obtained across different methods (Color figure online)

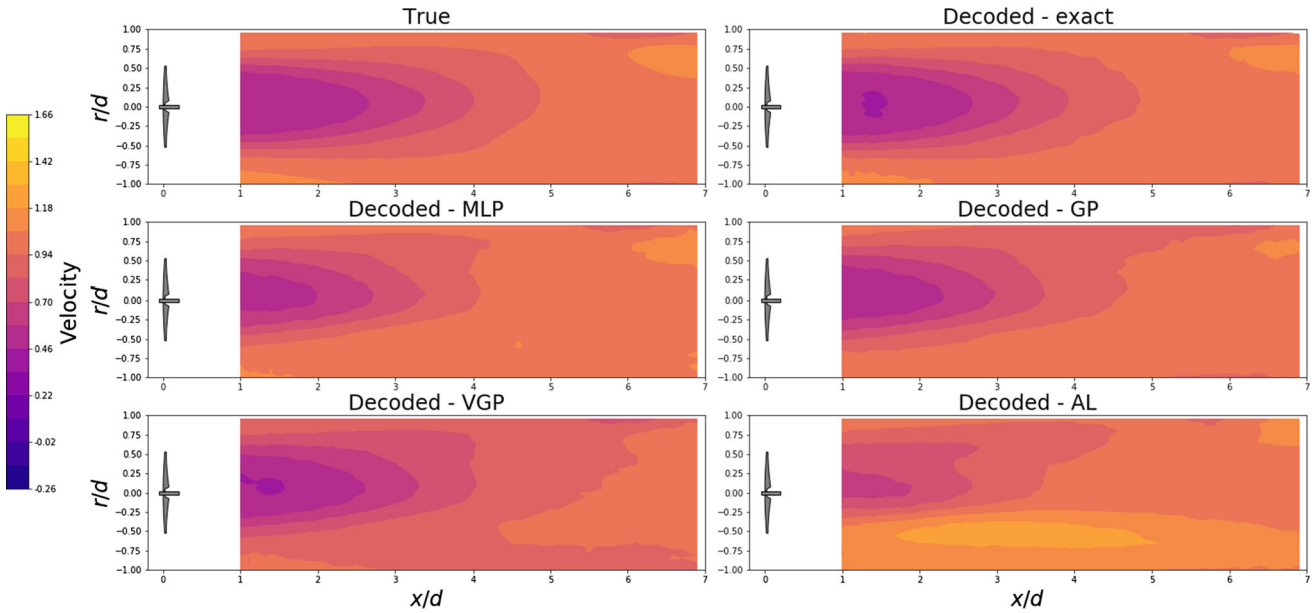
elements, which are imputed via a local interpolation, which—although inevitable—is expected to bias the dataset. Given these characteristics of our real-world dataset, including more sophistication, such as heteroscedastic noise variance, could potentially overfit the data; we reserve those approaches for future work.

We show the predicted flow field with our machine learning models, for two unseen parameters in Fig. 5. In addition to the true flow-field obtained from experimental observations, and the *best possible* reconstruction—via *autoencoding* the true flow-field, the prediction accuracy is more-or-less uniform across the various models, and visually presents a very close match to the true flow field. Note that, the machine learning models at best can only emulate the reconstruction decoded via the autoencoder and hence comparison against the *Decoded-exact* plot is most appropriate.

In Fig. 6, we also show a *worst-case* prediction via our machine learning models. These plots correspond to the outliers identified in Fig. 4. Overall, we noticed that total number of such outliers fall roughly within 10% of the overall dataset. For the specific wake measurement shown in Fig. 6, we notice that an anomalous speed-up is observed on the side of the wake for negative values of the radial position. This flow feature can be either due to flow interaction with side wind turbines or to large coherent



(a) Example 1



(b) Example 2

Fig. 5 Two examples of parametric reconstruction ability where information from the MET data is used to obtain a latent-space representation for a test data point. Following this, the decoder of the autoencoder is used to reconstruct in physical space. We show results for the true test snapshots, their exact reconstructions using only the autoencoder, and various parametric predictions in latent space followed by use of the decoder

flow structures typically present in the atmospheric boundary layer. Even though this kind of wake realizations are realistic, their occurrence can be relatively low and, thus, not captured from the training dataset.

We have used the sequential GP model fitting via AL is another approach to improve the tractability of (exact) GP regression for large datasets. Essentially, we choose the most *relevant* subset of the training dataset, instead of using the entire dataset or randomly sampling from it.

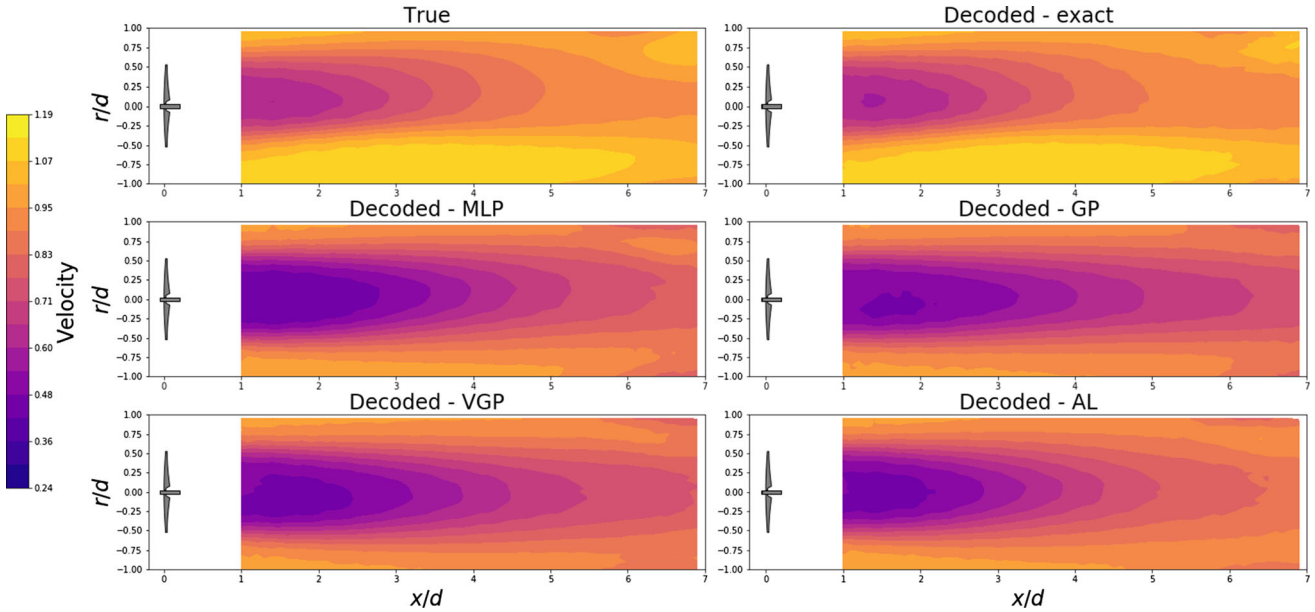


Fig. 6 Potential issue with an outlier—showing limited parametric map expressiveness across various latent space interpolation methods. Note that outliers were limited to around 10% of the dataset

Figure 7 shows the log RMSE of the GP—based on a held out test dataset of 1781 points—with sequential increments in the training data. Recall that the points are chosen sequentially to reduce the average uncertainty about the GP in the entire input domain $\mathcal{X}$, and Fig. 7 shows the resulting prediction error (log RMSE) variation with training dataset size. We start the GP fit with a randomly selected 50 points (from the full dataset of 5000 points) and sequentially add 100 more points. We also show the impact of the choice of the number of points selected (q) at each step by varying it between $q = 1$ to $q = 8$. We repeat this exercise 20 times for independently chosen random starting points and plot the mean and ± 1 standard deviation. Finally, we also show the prediction error due to selecting 250 points randomly in *one shot* (i.e., no sequential point selection). The results show that sequential point selection results in a smaller prediction error, regardless of the choice of q, for the first three latent space dimensions. For the fourth latent space dimension ($\tilde{\mathbf{x}}_4$), the $q = 1$ still outperforms the one-shot selection. It is worth noting that the one-shot selection of points still has 100 points more than what was supplied to the AL GP. The reason for the AL GP outperforming the one-shot GP is because training points are more judiciously chosen–in this case, they are chosen specifically to minimize the uncertainty in the GP about its own prediction. Furthermore, it can be shown [39] that the average uncertainty in the GP is equivalent to the mean-squared prediction error (MSPE) of the posterior mean of the GP, and therefore in effect the AL chooses points to minimize the overall prediction error.

The spatial distribution of points is visualized in Fig. 8, which is a scatterplot matrix of all possible combinations of the inputs listed in Table 1. In this figure, the blue symbols indicate the full training dataset (5000 points) and the red symbols are the points selected via AL. Notice that the red points show a better spread and hence coverage of the design space compared to the full dataset. This is further emphasized by the kernel density plots shown along the diagonal of the scatterplot matrix, where the AL shows a larger variance which is indicative of the fact that points are more spread out. Overall, we see that the choosing points via AL is another way toward building a more tractable GP model with large datasets, without compromising on the predictive accuracy.

Figure 9 shows a probability density plot of the prediction accuracy for all the machine learning models we employed in this work. Note that the densities have similar shapes indicating that the overall predictive accuracy for all the used models is somewhat similar, as mentioned previously. It is also worth noting that the AL GP still has the lowest accuracy amongst all the three GP approaches presented; this can be appreciated by considering the R^2 value in Fig. 4 and/or the sorted error and kernel density plot of the error shown in Fig. 10. Our hypothesis for this behavior is that these plots show just one realization of the GP (as opposed to Fig. 7 where we show the average of 20 independent realizations). In general, the predictive accuracy of the GP is sensitive to the choice of starting points, but Fig. 7 proves that on average, the AL GP outperforms a GP with one-shot point selection. Despite this fact, we still show just one realization of the AL GP in Figs. 4 and 10

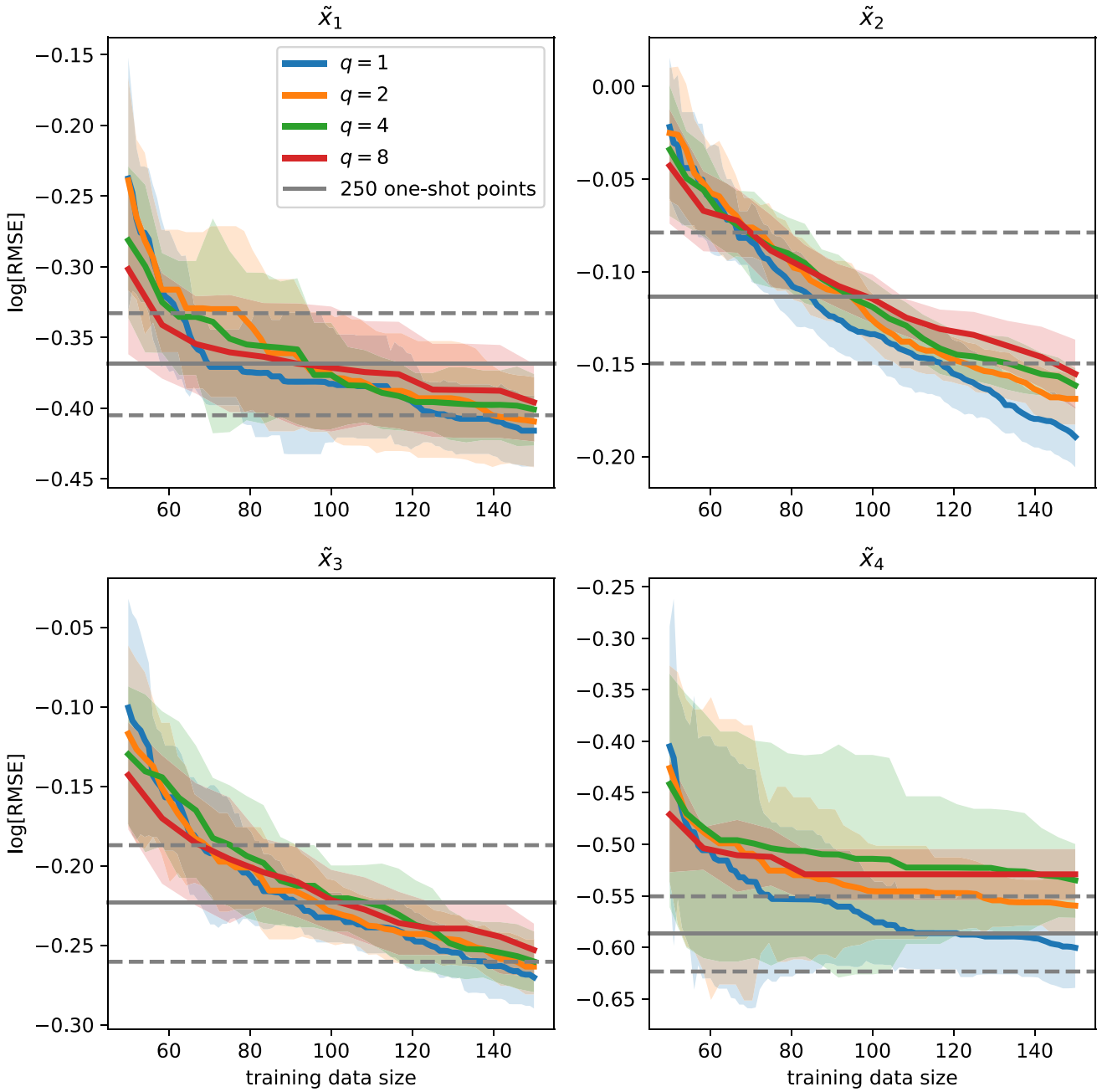


Fig. 7 One-shot vs sequential GP fit. Horizontal lines are the RMSE with a one-shot GP fit with 500 uniformly random points; solid and dashed lines are the mean and $\pm 1\sigma$, respectively. Sequential fit outperforms one-shot fit despite fewer training data

(despite showing a relatively poor predictive accuracy) because this is more representative of a realistic scenario.

6 Conclusions

Accurate predictions of wind-turbine wake flows are crucial to estimate power efficiency of wind farms, power losses due to wake interactions, and optimal design of wind farm layout. High-fidelity large-eddy simulations, which can resolve the turbulent flow fields with high accuracy, are prohibitively expensive from the computational-cost standpoint to tackle the foregoing wind energy problems. Scanning Doppler wind LiDARs provide detailed measurements of the wake flow field generated from utility-scale wind turbines; however, experimental data can be incomplete and noisy.

In this work, we develop data-driven machine learning models for prediction of wind turbine wakes by leveraging wind LiDAR measurements. Specifically, we develop

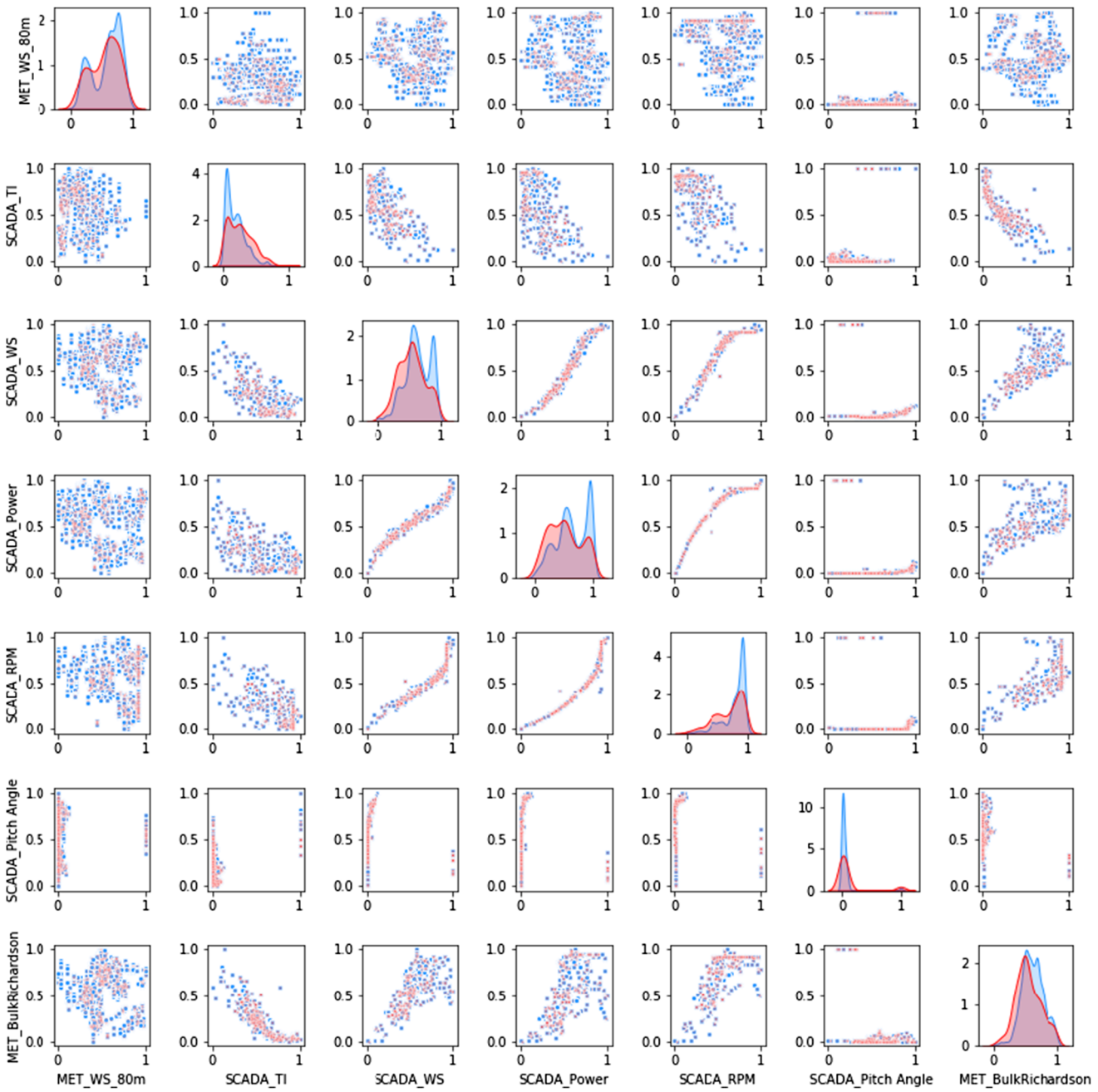
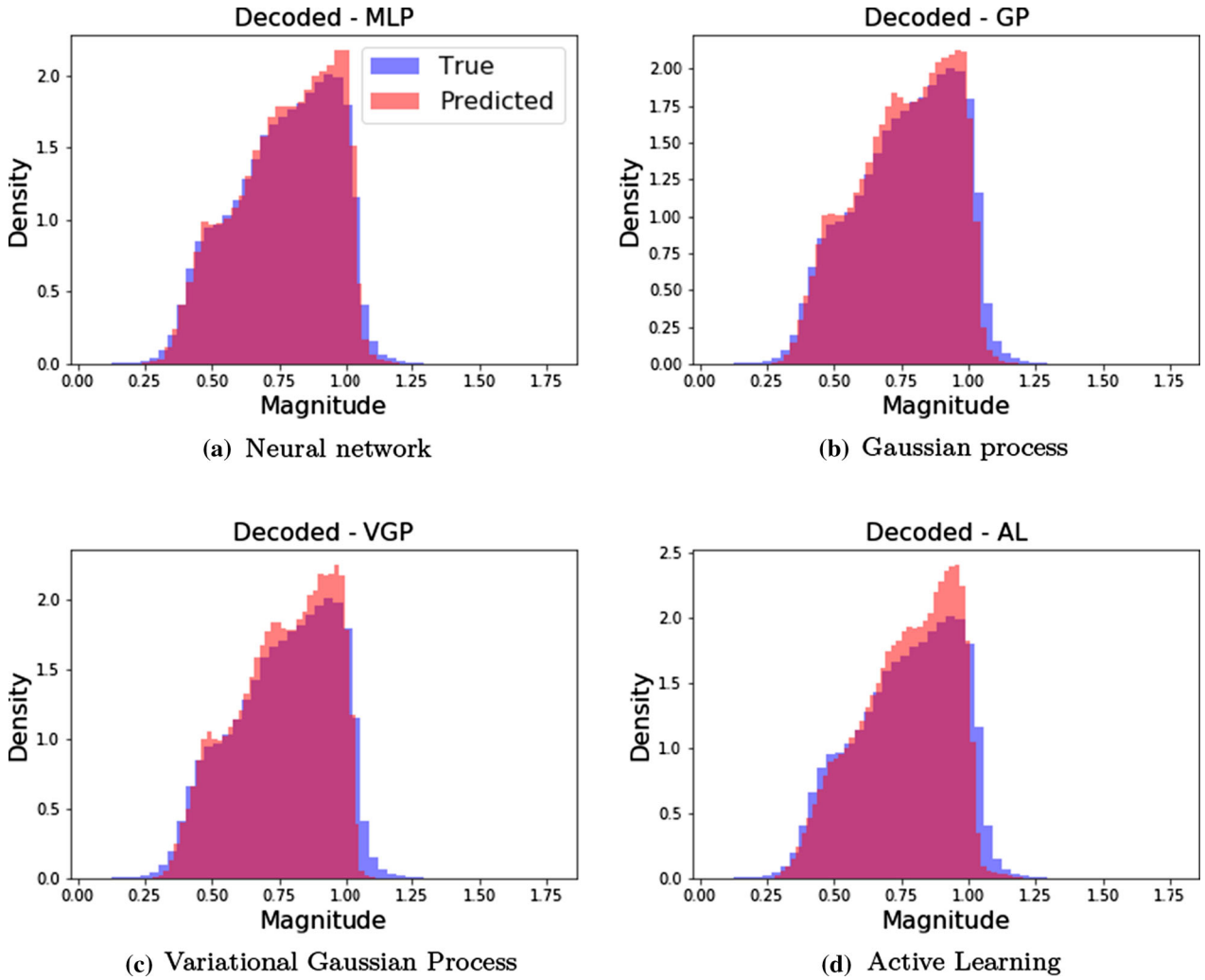


Fig. 8 Selected points via active learning. Blue represents the full training dataset (5000 points) and red represents the points selected via active learning (150 points)

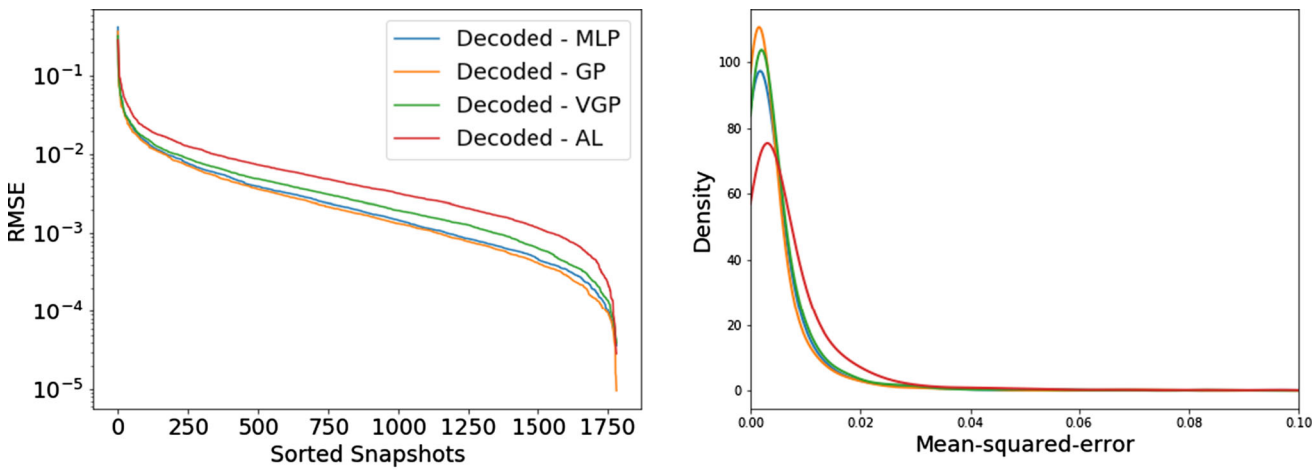
several data-driven models from high-dimensional LiDAR data and analyze their comparative performance in terms of parametric prediction. In this regard, we first use deep neural networks to achieve a drastically compressed ($2501 \rightarrow 4$) latent-space representation of the high-dimensional data. Then, we use multilayered perceptrons and Gaussian processes to learn the input parameter-latent-space map. Results show that the predictive capability of all the machine learning models is somewhat similar. However, GP regression with exact inference resulted in the least RMSE (best prediction). Furthermore, to address the well-known tractability issues with exact-inference Gaussian processes, we also propose variational sparse Gaussian processes as well as active learned Gaussian processes. These two approaches, while resulting in a significant saving in the computation necessary for model inference, are observed to make only a marginal trade in the accuracy. Overall, once trained, all of the machine



(a) Neural network (b) Gaussian process

(c) Variational Gaussian Process (d) Active Learning

Fig. 9 Density plots of data-driven predictions in latent space followed by reconstruction through the decoder for **a** a fully connected neural network **b** a Gaussian process **c** a sparse variational Gaussian process and **d** a reconstruction using a Gaussian process trained by active learning



(a) Sorted Error (b) Kernel density estimate of MSE

Fig. 10 The sorted root-mean-squared error on the testing dataset after parametric latent space prediction and reconstruction in physical space (left). Kernel density estimate of the mean-squared errors for different parametric maps (right). The RMSEs are competitive across a large portion of the test dataset with certain outliers (approximately 200 data points)

learning model takes only $O(1)$ second of wall-clock time to evaluate.

This work brings together state-of-the-art machine learning, to develop data-driven models of real-world data pertinent to wind energy. We show that robust and accurate models can be developed despite the data being noisy and incomplete. With the developed predictive models, sensitivity of the wake flow field to the input parameters can be analyzed in real-time, compared to the unrealistic cpu-hours necessary for high-fidelity simulations. In the future, we hope to streamline the whole framework for data-driven wake modeling via automated neural architecture search and improved uncertainty quantification (e.g., using heteroscedastic noise models for Gaussian processes). Furthermore, developing hybrid probabilistic machine learning models such as deep GP models, which combine multilayered perceptrons and Gaussian processes, is another focus of future work.

Acknowledgements This material is partially based upon work supported by the U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research, under Contract DE-AC02-06CH11357. This research was funded in part and used resources of the Argonne Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. SAR acknowledges the support by Laboratory Directed Research and Development (LDRD) funding from Argonne National Laboratory, provided by the Director, Office of Science, of the U.S. Department of Energy under contract DE-AC02-06CH11357. This research has been partially funded by a grant from the National Science Foundation CBET Fluid Dynamics, award number 1705837. Pattern Energy Group is acknowledged to provide access to the wind farm for the LiDAR experiment and wind farm data.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Barron AR (1993) Universal approximation bounds for superpositions of a sigmoidal function. IEEE Trans Inf Theory 39(3):930–945
2. Barthelmie R, Pryor S, Frandsen S, Hansen K, Schepers J, Rados K, Schelz W, Neubert A, Jensen L, Neckelmann S (2010) Quantifying the impact of wind turbine wakes on power output at offshore wind farms. J Atmos Ocean Technol. https://doi.org/10.1175/2010JTECHA1398.1
3. Bastankhah M, Porté-Agel F (2014) A new analytical model for wind-turbine wakes. Renew Energy 70:116–123. https://doi.org/10.1016/j.renene.2014.01.002
4. Blei DM, Kucukelbir A, McAuliffe JD (2017) Variational inference: a review for statisticians. J Am Stat Assoc 112(518):859–877
5. Breton S, Sumner J, Sørensen JN, Hansen KS, Sarmast S, Ivanell S (2017) A survey of modelling methods for high-fidelity wind farm simulations using large eddy simulation. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences 375 (20160097)
6. Cheng M, Fang F, Pain C, Navon I (2020) An advanced hybrid deep adversarial autoencoder for parameterized nonlinear fluid flow modelling. Comput Methods Appl Mech Eng 372:113375
7. Churchfield MJ, Lee S, Michalakes J, Moriarty PJ (2012) A numerical study of the effects of atmospheric and wake turbulence on wind turbine dynamics. J Turbulence 13:1–32. https://doi.org/10.1080/14685248.2012.668191
8. Cohn DA, Ghahramani Z, Jordan MI (1995) Active learning with statistical models. Tech. rep., Massachusetts Inst of Tech Cambridge Artificial Intelligence Lab
9. Cohn DA, Ghahramani Z, Jordan MI (1996) Active learning with statistical models. J Artif Intell Res 4:129–145
10. Conti D, Dimitrov N, Peña A (2020) Aeroelastic load validation in wake conditions using nacelle-mounted lidar measurements. Wind Energy Sci 5(3):1129–1154. https://doi.org/10.5194/wes-5-1129-2020
11. Cressie N, Huang HC (1999) Classes of nonseparable, spatio-temporal stationary covariance functions. J Am Stat Assoc 94(448):1330–1339
12. Cybenko G (1989) Approximation by superpositions of a sigmoidal function. Math Control Signals Syst 2(4):303–314
13. El-Asha S, Zhan L, Iungo G (2017) Quantification of power losses due to wind turbine wake interactions through SCADA, meteorological and wind LiDAR data. Wind Energy 20(June):1823–1839. https://doi.org/10.1002/we.2123
14. Frandsen S, Barthelmie R, Pryor S, Rathmann O, Larsen S, Højstrup J, Thøgersen M (2006) Analytical modelling of wind speed deficit in large offshore wind farms. Wind Energy 9(1–2):39–53. https://doi.org/10.1002/we.189
15. Fukami K, Nakamura T, Fukagata K (2020) Convolutional neural network based hierarchical autoencoder for nonlinear mode decomposition of fluid field data. Phys Fluids 32(9):095110
16. Gelman A, Carlin JB, Stern HS, Rubin DB (1995) Bayesian data analysis. Chapman and Hall/CRC, Boca Rato
17. Gonzalez FJ, Balajewicz M (2018) Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems. arXiv preprint arXiv:1808.01346
18. Hensman J, Matthews A, Ghahramani Z (2015) Scalable variational gaussian process classification. In: Artificial Intelligence and Statistics, pp. 351–360. PMLR
19. Iungo GV, Porté-Agel F (2014) Volumetric lidar scanning of wind turbine wakes under convective and neutral atmospheric stability regimes. J Atmos Oceanic Technol 31(10):2035–2048
20. Iungo GV, Santhanagopalan V, Ciri U, Viola F, Zhan L, Rotea MA, Leonardi S (2018) Parabolic rans solver for low-computational-cost simulations of wind turbine wakes. Wind Energy 21(3):184–197
21. Jensen NO (1983) A note on wind generator interaction. Tech. rep., Risø, Roskilde, Denmark. DOI Riso-M-2411. URL http://www.risoe.dk/rispubl/VEA/veapdf/ris-m-2411.pdf
22. Kim Y, Choi Y, Widemann D, Zohdi T (2020) A fast and accurate physics-informed neural network reduced order model with shallow masked autoencoder. arXiv preprint arXiv:2009.11990
23. Letizia S, Zhan L, Valerio Iungo G (2021) LiSBOA: LiDAR Statistical Barnes Objective Analysis for optimal design of LiDAR scans and retrieval of wind statistics. Part I: Theoretical framework. Atmos Measurement Tech 14:2065–2093. https://doi.org/10.5194/amt-14-2065-2021
24. Machefaux E, Larsen GC, Koblitz T, Troldborg N, Kelly MC, Chougule A, Hansen KS, Rodrigo JS (2016) An experimental and numerical study of the atmospheric stability impact on wind turbine wakes. Wind Energy 19(10):1785–1805

25. Matérn B (2013) Spatial variation, vol 36. Springer Science & Business Media, Berlin
26. Maulik R, Lusch B, Balaprakash P (2021) Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders. Phys Fluids 33(3):037106
27. Maulik R, Rao V, Renganathan SA, Letizia S, Iungo GV (2021) Cluster analysis of wind turbine wakes measured through a scanning doppler wind lidar. In: AIAA Scitech 2021 Forum, p. 1181
28. Mehta D, Zuijlen AHV, Koren B, Holierhoek JG, Bijl H (2014) Large Eddy Simulation of wind farm aerodynamics: a review. J Wind Eng Ind Aerodyn 133:1–17. https://doi.org/10.1016/j.jweia.2014.07.002
29. Murata T, Fukami K, Fukagata K (2020) Nonlinear mode decomposition with convolutional neural networks for fluid dynamics. J Fluid Mech, 882
30. Porté-agel F, Bastankhah M, Shamsoddin S (2019) Wind-Turbine and Wind-Farm Flows: a review. Boundary-Layer Meteorol. https://doi.org/10.1007/s10546-019-00473-0
31. Rajaram D, Puranik TG, Ashwin Renganathan S, Sung W, Fischer OP, Mavris DN, Ramamurthy A (2021) Empirical assessment of deep gaussian process surrogate models for engineering problems. J Aircraft 58(1):182–196
32. Rajaram D, Puranik TG, Renganathan A, Sung WJ, Pinon-Fischer OJ, Mavris DN, Ramamurthy A (2020) Deep gaussian process enabled surrogate models for aerodynamic flows. In: AIAA Scitech 2020 Forum, p. 1640
33. Renganathan SA (2020) Koopman-based approach to nonintrusive reduced order modeling: Application to aerodynamic shape optimization and uncertainty propagation. AIAA J 58(5):2221–2235
34. Renganathan SA, Larson J, Wild SM (2021) Lookahead acquisition functions for finite-horizon time-dependent bayesian optimization and application to quantum optimal control. arXiv preprint arXiv:2105.09824
35. Renganathan SA, Liu Y, Mavris DN (2018) Koopman-based approach to nonintrusive projection-based reduced-order modeling with black-box high-fidelity models. AIAA J 56(10):4087–4111
36. Renganathan SA, Maulik R, Ahuja J (2021) Enhanced data efficiency using deep neural networks and gaussian processes for aerodynamic design optimization. Aerospace Sci Technol 111:106522
37. Renganathan SA, Maulik R, Rao V (2020) Machine learning for nonintrusive model order reduction of the parametric inviscid transonic flow past an airfoil. Phys Fluids 32(4):047110
38. Sanderse B, van der Pijl S, Koren B (2011) Review of computational fluid dynamics for wind turbine wake aerodynamics. Wind Energy 14: 799–819. https://doi.org/10.1002/we.1608/full. URL http://onlinelibrary.wiley.com/doi/10.1002/we.1608/full
39. Santner TJ, Williams BJ, Notz WI, Williams BJ (2003) The design and analysis of computer experiments, vol 1. Springer, Berlin
40. Santoni C, Garcia-Cartagena EJ, Ciri U, Zhan L, Iungo GV, Leonardi S (2020) One-way mesoscale-microscale coupling for simulating a wind farm in North Texas: Assessment against SCADA and LiDAR data. Wind Energy 23(3):691–710
41. Sanz Rodrigo J, Chávez Arroyo RA, Moriarty P, Churchfield M, Kosović B, Réthoré PE, Hansen KS, Hahmann A, Mirocha JD, Rife D (2017) Mesoscale to microscale wind farm flow modeling and evaluation. Wiley Interdisciplinary Reviews: Energy and Environment 6(2). https://doi.org/10.1002/wene.214
42. Sebastiani A, Segalini A, Castellani F, Crasto G (2020) Data analysis and simulation of the Lillgrund wind farm. Wind Energy (November). https://doi.org/10.1002/we.2594
43. Snelson E, Ghahramani Z (2005) Sparse gaussian processes using pseudo-inputs. Adv Neural Inf Process Syst 18:1257–1264
44. Stein ML (2012) Interpolation of spatial data: some theory for kriging. Springer Science & Business Media, Berlin
45. U.S.G.S. (2017) U.S. Geological Survey Website. URL https://www.usgs.gov/43
46. Veers P, Dykes K, Lantz E, Barth S, Bottasso C, Carlson O, Clifton A, Green J, Green P, Holttinen H, Laird D, Lehtomäki V, Lundquist J, Manwell J, Marquis M, Meneveau C, Moriarty P, Munduate X, Muskulus M, Naughton J, Pao L, Paquette J, Peinke J, Robertson A, Rodrigo JS, Sempreviva A, Smith J, Tuohy A, Wiser R (2019) Grand challenges in the science of wind energy. Science. https://doi.org/10.1126/science.aau2027
47. Vennemann B, Rösgen T (2020) A dynamic masking technique for particle image velocimetry using convolutional autoencoders. Exp Fluids 61(7):1–11
48. Williams CK, Rasmussen CE (2006) Gaussian processes for machine learning, vol 2. MIT press, Cambridge, MA
49. Wu P, Gong S, Pan K, Qiu F, Feng W, Pain C (2021) Reduced order model using convolutional auto-encoder with self-attention. Phys Fluids 33(7):077107
50. Zhan L, Letizia S, Iungo GV (2020) Optimal tuning of engineering wake models through lidar measurements. Wind Energy Sci 5(4):1601–1622. https://doi.org/10.5194/wes-5-1601-2020
51. Zhan L, Letizia S, Valerio Iungo G (2020) Lidar measurements for an onshore wind farm: wake variability for different incoming wind speeds and atmospheric stability regimes. Wind Energy 23(3):501–527