scientific reports

Check for updates

OPEN

Multi-fidelity information fusion with concatenated neural networks

Suraj Pawar[1], Omer San[1]✉, Prakash Vedula[2], Adil Rasheed[3,5] & Trond Kvamsdal[4,5]

Recently, computational modeling has shifted towards the use of statistical inference, deep learning, and other data-driven modeling frameworks. Although this shift in modeling holds promise in many applications like design optimization and real-time control by lowering the computational burden, training deep learning models needs a huge amount of data. This big data is not always available for scientific problems and leads to poorly generalizable data-driven models. This gap can be furnished by leveraging information from physics-based models. Exploiting prior knowledge about the problem at hand, this study puts forth a physics-guided machine learning (PGML) approach to build more tailored, effective, and efficient surrogate models. For our analysis, without losing its generalizability and modularity, we focus on the development of predictive models for laminar and turbulent boundary layer flows. In particular, we combine the self-similarity solution and power-law velocity profile (low-fidelity models) with the noisy data obtained either from experiments or computational fluid dynamics simulations (high-fidelity models) through a concatenated neural network. We illustrate how the knowledge from these simplified models results in reducing uncertainties associated with deep learning models applied to boundary layer flow prediction problems. The proposed multi-fidelity information fusion framework produces physically consistent models that attempt to achieve better generalization than data-driven models obtained purely based on data. While we demonstrate our framework for a problem relevant to fluid mechanics, its workflow and principles can be adopted for many scientific problems where empirical, analytical, or simplified models are prevalent. In line with grand demands in novel PGML principles, this work builds a bridge between extensive physics-based theories and data-driven modeling paradigms and paves the way for using hybrid physics and machine learning modeling approaches for next-generation digital twin technologies.

The modeling of spatiotemporal dynamics of multiscale and multiphysics systems is an open problem relevant to many scientific and engineering applications. For instance, wind energy is a highly complex system whose dynamics is governed by global atmospheric processes to turbulent boundary layer formed around the blades that span nine orders of magnitudes[1]. Over the past several decades, we have improved our understanding of such multiphysics systems by developing accurate numerical models for governing equations of the system, such as Navier-Stokes equations for fluid flows. However, these numerical models can be computationally prohibitive, especially for nonlinear multiscale systems, and their use in real-time optimization and control is scarce. The recent advancement in machine learning (ML) and deep learning (DL) holds the great potential for tackling the challenge of modeling and analysis of high-dimensional systems and has been successful in diverse applications, such as fluid mechanics[2], earth science[3], and material science[4]. These advances have been driven by a vast amount of data generated from high-resolution numerical simulations, experimental and satellite measurements, and computing power along with the emergence of effective and efficient algorithms that can extract relevant patterns from the data. ML/DL techniques has been successfully applied for turbulence closure modeling[5], super-resolution of climate data[6], predicting clustered weather patterns[7], reduced-order modeling[8], and many more.

ML/DL models are capable of providing insights from data, exploiting these insights in building predictive tools, and continuously updating themselves as the new streams of data get available. Despite these advantages, ML/DL techniques lack interpretability and suffer from the curse of dimensionality. The interpretability issue can be addressed by understanding the physical implications of ML/DL models[9], and understanding the neural network correlations discovered from the data[10,11]. By the curse of dimensionality, we mean that DL models are

[1]School of Mechanical and Aerospace Engineering, Oklahoma State University, Stillwater, OK 74078, USA. [2]School of Aerospace and Mechanical Engineering, The University of Oklahoma, Norman, OK 73019, USA. [3]Department of Engineering Cybernetics, Norwegian University of Science and Technology, 7465 Trondheim, Norway. [4]Department of Mathematical Sciences, Norwegian University of Science and Technology, 7491 Trondheim, Norway. [5]Department of Mathematics and Cybernetics, SINTEF Digital, 7034 Trondheim, Norway. ✉email: osan@okstate.edu

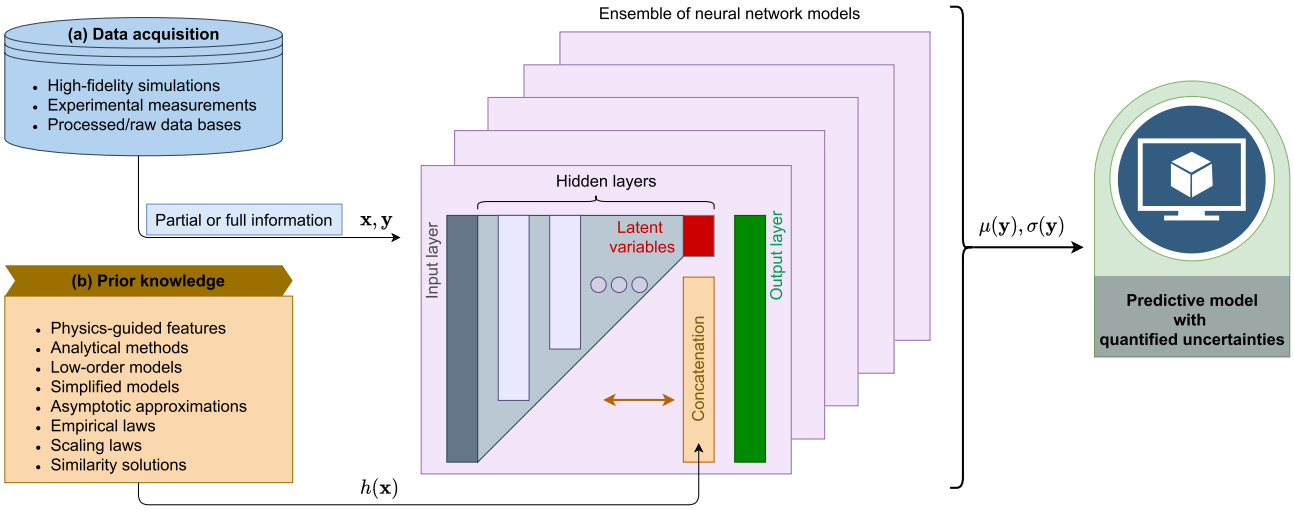

Figure 1. The proposed multi-fidelity information fusion framework. The prediction from the low-fidelity models (e.g., self-similarity solution or law of the wall) is concatenated with the latent variables at the certain hidden layer of the neural network. An ensemble of neural networks is trained using the negative log-likelihood loss function to estimate the uncertainty associated with the prediction. Here, $\mathbf{x}$ refers to independent variables or design parameters, or $\mathbf{y}$ indicates the quantities of interest, and $h(\mathbf{x})$ represents a low-fidelity or simplified model that is fast-to-compute.

data-hungry in nature. For instance, Bonavita and Laloyaux[12] showed that the amount of the training data for nonlinear dynamical systems grows exponentially with the dimensionality of the system. Furthermore, pure ML/DL models lead to poor extrapolation/generalization, i.e., they fit the observations data very well, but predictions may be poor and physically inconsistent for data beyond the distribution of the training dataset. To this end, physics-informed learning algorithms that leverage prior knowledge based on the physical and mathematical understanding of the system are proposed in several studies[13].

There are two main techniques based on inductive biases, and learning biases to embed physics into ML/DL models in combination with the observations data[13]. Inductive biases techniques relate to building tailored ML/DL model architecture that exploits the prior knowledge about the problem at hand to build physically consistent data-driven models. The representative examples includes embedding invariance property into neural network architecture[14,15], imposing conservation laws of physical quantities or analytical constraints into neural network[16–18], using prediction from simplified models as the bias[19], enhancing feature space[20], and adopting equivariant transformer networks[21]. The other approach based on learning biases imposes constraints such as governing equations in a soft manner by penalizing the loss function of ML/DL models. Some of the examples of this approach pertain to physics-informed neural networks[22], statistically constrained generative adversarial networks[23], and Bayesian framework with auto-regressive model[24]. There is also a class of hybrid analysis and modeling approaches that utilizes pure data-driven and physics-based models in tandem[25]. While many methods have been demonstrated to be successful in enforcing physics into ML/DL models, they offer many possibilities to fuse domain knowledge to improve the generalizability and data efficiency of data-driven models[26].

In this work, we introduce a physics-guided machine learning (PGML) fraamework based on tailored neural network architecture that utilizes a concatenation layer for surrogate modeling of high-fidelity data given the solution from a computationally inexpensive low-fidelity model. There is a hierarchy of numerical models ranging from simple empirical relations to highly accurate numerical discretization-based models across all scientific disciplines. For example, flow around airfoils can be modeled using tools extending from panel methods[27] to the fully resolved direct numerical simulation (DNS)[28]. Another example is the wake modeling of wind turbines, where fast but typically inaccurate analytical models are adopted for tasks like layout optimization and wind farm control[29]. However these models are insufficient to take the unsteady nature of interactions of turbine wake with other wakes as well as atmospheric turbulence into account, and such effects can be modeled with computationally demanding but accurate models like large-eddy simulation (LES)[30]. Our work draws inspiration from these multi-fidelity modeling approaches and exploits the real-time prediction from low-fidelity models to inform a DL model of high-fidelity observations. The information fusion from the multi-fidelity sources of data leads to a robust and generalizable surrogate model in comparison to a purely data-driven model trained solely on high-fidelity data.

Figure 1 graphically illustrates the proposed data fusion approach from multimodal data streams in the process of generating PGML models. As shown in Fig. 1, a data-driven model can be constructed using the data obtained from high-resolution simulations and experimental measurements. The PGML framework further augments the ML model with the information from a low-fidelity representation of the system (such as analytical models, scaling laws, empirical relations, etc.). Ensemble of PGML models is trained using both the sources of data along with the uncertainty quantification mechanism that finally gives the predictive model to be employed in online tasks. In a nutshell, our work puts forth a novel data-driven framework to take prior knowledge about the system into account when generating a black-box deep learning predictive model. How should we inject

physics and domain knowledge into machine learning models? How deep learning can be constructed as a trustworthy approach toward more accurate real-time prediction of nonlinear complex systems such as turbulent flows? These are the fundamental research questions that we tackle in this paper, and provide our insights about these questions.

We demonstrate the application of our framework for boundary layer flows. Boundary layer phenomenon is one of the most important flows and is of engineering concern in many scientific and industrial applications[31]. The behavior of flow in the boundary layer has implications on the drag force in ship hulls and aircraft, the energy required to move oil through pipes, and the distribution of heat in the atmosphere[32–34]. Given that boundary layer flows are prevalent in engineering applications, building a computationally efficient and accurate surrogate model is of paramount importance for online tasks like boundary layer control to achieve lift enhancement, noise mitigation, drag reduction, and wall cooling. Additionally, boundary layer flows can be described using hierarchies of models that have different levels of fidelity spanning analytical models to DNS and hence represent an interesting test case for illustrating the effectiveness of PGML framework. While in this work we consider only two levels of fidelity, the proposed framework can be applied for blending information from various levels of fidelity. Moreover, the neural architecture search tools can be utilized to discover more complex and optimal architectures automatically[35].

Methods

In this Section, we first provide the details of the concatenated neural network architecture as a PGML framework that blends the information from models of different levels of fidelity and then present the deep ensemble method used to quantify the uncertainty associated with the prediction. Then, we discuss the multi-fidelity data fusion for laminar and turbulent boundary layer flows over a flat plate.

Multi-fidelity concatenated neural network. A neural network is a computational graph composed of several layers consisting of the predefined number of neurons. Each neuron is associated with certain coefficients called weights and some bias. The input from the previous layer is multiplied by a weight matrix as shown below

$$\mathbf{S}^l = \mathbf{W}^l \mathbf{X}^{l-1}, \tag{1}$$

where $\mathbf{X}^{l-1}$ is the output of the $(l-1)$th layer, $\mathbf{W}^l$ is the matrix of weights for the lth layer, and $\mathbf{S}^l$ is the input-weight product. The summation of the above input-weight product and the bias is then passed through a node's activation function which is usually some nonlinear function. The introduction of nonlinearity through activation function allows the neural network to learn highly complex relations between the input and output. The output of the lth layer can be written as

$$\mathbf{X}^l = \zeta(\mathbf{S}^l + \mathbf{B}^l), \tag{2}$$

where $\mathbf{B}^l$ is the vector of biasing parameters for the lth layer and ζ is the activation function. If there are N_L layers between the input and the output in a neural network, then the neural network mapping $\mathscr{F} : \mathbf{X} \rightarrow \mathbf{Y}(\mathbf{X})$ can be represented as follows

$$\mathbf{Y} = \zeta_{N_L}(\cdot; \boldsymbol{\Theta}_{N_L}) \circ \cdots \circ \zeta_2(\cdot; \boldsymbol{\Theta}_2) \circ \zeta_1(\mathbf{X}; \boldsymbol{\Theta}_1) \tag{3}$$

where $\boldsymbol{\Theta}$ represents the weight and bias of the corresponding layer of the neural network and $\mathbf{Y}$ is the final output of the neural network. For the concatenated neural network, the information from the low-fidelity model is injected at a certain intermediate layer of the neural network as follows

$$\mathbf{Y} = \zeta_{N_L}(\cdot; \boldsymbol{\Theta}_{N_L}) \circ \cdots \circ \underbrace{\mathscr{C}\ (\zeta_i(\cdot; \boldsymbol{\Theta}_i), h(\mathbf{X}))}_{\text{Concatenation layer}} \circ \cdots \circ \zeta_1(\mathbf{X}; \boldsymbol{\Theta}_1), \tag{4}$$

where $\mathscr{C}\ (\cdot, \cdot)$ represents the concatenation operation and the information from the low-fidelity model, i.e., $h(\mathbf{X})$ is injected at ith layer, and $\boldsymbol{\Theta}_i$ are the trainable parameters of the corresponding layer. The concatenation operator takes the latent variables at a particular layer and combines them with information from the low-fidelity model to return a vector. Specifically, if the ith layer has D_i neurons and $h(\mathbf{X}) \in \mathbb{R}^{D_h}$, the input to the $(i+1)$th layer will be in $\mathbb{R}^{D_i + D_h}$. During the training of the neural network, the weights of the neural network are updated using the backpropagation and gradient descent algorithm. The backpropagation algorithm involves computation of the gradient of the loss function with respect to each of the trainable parameters and the trainable parameters are updated using a gradient descent algorithm. In a concatenated neural network, the prediction from the low-fidelity model is injected at an intermediate layer of the network. Therefore, the low-fidelity model prediction is also the input feature to the network, and the training does not involve computing the gradient of the loss function with respect to this injected feature. However, the number of trainable parameters is increased and the change in the number of trainable parameters will depend on the dimension of the low-fidelity data and the number of neurons in the subsequent hidden layer.

One important caveat in a concatenated neural network is the selection of an appropriate intermediate layer at which to inject the low-fidelity model information, and tools like automated machine learning (AutoML)[36] can be applied to automate this search. The concatenation operator given by Eq. 4 can also be applied to all hidden layers simultaneously as

$$\mathbf{Y} = \zeta_{N_L}(\cdot;\boldsymbol{\Theta}_{N_L}) \circ \underbrace{\mathscr{C}\ (\zeta_{N_L-1}(\cdot;\boldsymbol{\Theta}_{N_L-1}), h(\mathbf{X})) \circ \cdots \circ \mathscr{C}\ (\zeta_i(\cdot;\boldsymbol{\Theta}_i), h(\mathbf{X})) \circ \cdots \circ \mathscr{C}\ (\zeta_1(\cdot;\boldsymbol{\Theta}_1), h(\mathbf{X}))}_{\text{Concatenation layers}}. \tag{5}$$

We highlight here that the proposed PGML framework is modular and Equation 5 can be generalized for fusing information from multiple low-fidelity models. However, we dedicate this study to investigate the feasibility of the proposed PGML framework for only two levels of approximations. An additional optimization problem can be constructed to search the best architecture in terms of relevant hyperparameters such as the number of hidden layers, and the location and sparsity of the concatenation structure. Such auto PGML investigations will be a topic that we will pursue in our future works.

Deep ensembles: training and prediction. Deep learning algorithms like neural networks approximate the mapping from inputs to outputs using trainable parameters called weights and biases. The parameters of the neural network are determined through the minimization of the loss function. The prediction from the neural network is usually a point estimate, i.e., continuous outputs for regression tasks and discrete classes for classification problems. However, the information about the confidence in the model's prediction might be crucial for many scientific applications[37]. The uncertainty estimates can also be useful for applications like sensor placement and Bayesian optimization. In this study, we apply the deep ensembles algorithm for estimating the probabilistic distribution function (PDF) of output conditioned on the inputs[38]. While there are state-of-the-art methods like Bayesian neural networks[39] that quantifies uncertainty by learning the distribution of weights, deep ensembles is adopted due to their simplicity and scalability.

Here, we briefly discuss the uncertainty quantification mechanism of deep ensembles. We assume that our training dataset $\mathscr{D}$ consists of N samples $\mathscr{D} = \{\mathbf{X}_i, \mathbf{Y}_i\}_{i=1}^N$, where $\mathbf{X} \in \mathbb{R}^P$ represents the $P-$dimensional features and the label is $Q-$dimensional, i.e., $\mathbf{Y} \in \mathbb{R}^Q$. A neural network is trained by minimizing the loss function $\mathscr{L}(\mathbf{Y}, \tilde{\mathbf{Y}}(\mathbf{X}; \underline{\Theta}))$, where $\tilde{\mathbf{Y}}$ is the predicted label from a neural network parameterized by $\underline{\Theta}$ (i.e., trainable parameters of the whole neural network). The most common loss function for regression tasks is the mean squared error (MSE) between true and predicted labels averaged over all samples in the dataset. The MSE loss function does not give an estimate of the probability distribution of $\mathscr{P}(\tilde{\mathbf{Y}}|\mathbf{X})$ and hence the uncertainty estimate is usually absent with the prediction from neural networks.

In order to quantify the predictive uncertainty, the neural network is trained to output the mean and variance of the Gaussian distribution in the output layer. The weights of the neural network are determined by minimizing the negative log-likelihood $\mathscr{L}$ as follows

$$\underline{\Theta} = \arg\min_{\underline{\Theta}}[\mathscr{L}], \quad \text{where} \quad \mathscr{L} = \sum_{i=1}^{N} \frac{1}{2}\log\sigma^2(\mathbf{X}_i) + \frac{(\mathbf{Y}_i - \mu(\mathbf{X}_i))^2}{2\sigma^2(\mathbf{X}_i)}, \tag{6}$$

where the mean μ and the variance σ^2 are parameterized by the neural network. The positivity constraint is enforced for the variance by passing the output corresponding to variance of distribution though the *softplus* function $\log(1+\exp(\cdot))$, and adding a minimum variance (for example 10^{-6}) for numerical stability. If we assume the variance to be *constant* in Equation 6 (i.e., it does not depend on input features), then the negative log-likelihood loss function becomes analogous to the MSE loss function. Therefore, from a probabilistic point of view, minimizing the MSE is equivalent to minimizing negative log-likelihood with an assumption of Gaussian distribution with constant standard deviation[40,41].

The ensemble of neural networks has been demonstrated to be successful in improving the predictive performance of machine learning models[42]. There are broadly two methods of generating ensembles, (i) randomization-based approaches where the ensembles can be trained in parallel without any interaction, and (ii) boosting-based approaches where the ensembles are trained sequentially[43]. The randomization procedure for generating ensembles of neural networks should be such that prediction from individual models are de-correlated and each individual models are strong (i.e., high accuracy). In this work, the random initialization of weights of the neural network is used for generating ensembles. There are other schemes such as bagging where the ensembles of neural networks are trained on a different subset of the original training data. However, random initialization is better than bagging for improving predictive accuracy and uncertainty[38,44]. This simple and yet robust randomization approach is highly scalable as it allows for distributed training of neural networks and can be applied to many scientific problems. For computing the predictive probability distribution, we approximate the ensemble prediction as a Gaussian whose mean and variance are computed as follows

$$\mu_*(\mathbf{X}) = \frac{1}{M}\sum_{j=1}^{M} \mu_{\underline{\Theta}_j}(\mathbf{X}), \tag{7}$$

$$\sigma_*^2(\mathbf{X}) = \frac{1}{M}\sum_{j=1}^{M} (\sigma_{\underline{\Theta}_j}^2(\mathbf{X}) + \mu_{\underline{\Theta}_j}^2(\mathbf{X})) - \mu_*^2(\mathbf{X}), \tag{8}$$

where $\mu_{\underline{\Theta}_j}$ and $\sigma_{\underline{\Theta}_j}$ is the mean and standard deviation of predicted probability distribution by the jth neural network. We employ an ensemble of five neural networks in this study (i.e., $M = 5$).

Multi-fidelity data fusion for laminar boundary layer. The first test case considered in this study is the laminar boundary layer flow. Boundary layer flows can be characterized by dividing the flow into two regions, one inside the boundary layer where the viscosity dominates and one outside the boundary layer where the effect of viscosity can be neglected. The low-fidelity model considered for laminar flow is the steady-state two-dimensional laminar boundary layer described using Blasius equation[45]. The core idea behind Blasius equation is transforming a partial differential equation (PDE) comprised of the flat plate boundary layer equations, with zero pressure gradient, into a single ordinary differential equation (ODE) by using a similarity solution approach. The derivation of the Blasius equation can be found in many texts on fluid mechanics and we describe only the final form. The Blasius equation and its boundary conditions can be written as

$$2f''' + ff'' = 0, \tag{9}$$

$$f(0) = f'(0) = 0,\ f'(\infty) = 1, \tag{10}$$

where $f(\eta)$ is a function of similarity variable η. The similarity variable η is defined as $\eta = y\sqrt{u_\infty/(x\nu)}$, where y is the direction normal to the plate, x is the direction along its length with zero being the leading edge, u_∞ is the freestream velocity, and ν is the kinematic viscosity of the fluid. The third-order ODE is first split into a coupled system of three first-order ODEs. Then we apply the shooting method to determine the initial value for $f''(0)$, and the first-order ODEs are numerically integrated with the fourth-order Runge-Kutta scheme[46]. The velocity profile from the Blasius solution can be determined using the relation $\bar{u} = u_\infty f'$, where overbar symbol is used to indicate the low-fidelity model estimate. The high-fidelity observations are generated by solving the RANS equations with the PISO algorithm[47] available in OpenFoam. We get the velocity (components along streamwise and wall-normal directions) and the pressure distribution from CFD simulation. The Reynolds number based on the length of the flat plate used for generating data is $\mathrm{Re}_L = 5 \times 10^4$, where L is the length of the flat plate. The training data for the concatenated neural network is sampled from the whole domain and the velocity field is contaminated by adding a white Gaussian noise with zero mean and a standard deviation of 0.05. Advanced sampling methods like Latin hypercube sampling, clustered sampling can be utilized to reduce the number of samples required for training and we will consider this as part of our future work.

For the laminar boundary layer reconstruction task, the input to the neural network is the location of the sensor, i.e., $\mathbf{X} = [x, y]$, where x and y are the positions of the sensors in streamwise and wall-normal directions. The output of the neural network is the probability distribution of u, v, p represented by their mean and standard deviation, where u is the velocity in the streamwise direction, v is the velocity in the wall-normal direction, and p is the pressure at the sensor's location. Additionally, the velocity profile obtained from the Blasius solution is used as the low-fidelity model, i.e., $h(\mathbf{X}) = [\bar{u}]$. Following our previous discussion, the problem formulation can be written as

$$\{\mu(\mathbf{X}), \sigma(\mathbf{X})\} = \mathscr{F}(\mathbf{X}, h(\mathbf{X})), \quad \mathscr{P}(\mathbf{Y}|\mathbf{X}, h(\mathbf{X})) = \mathscr{N}(\mu_*(\mathbf{X}), \sigma_*(\mathbf{X})). \tag{11}$$

Multi-fidelity data fusion for turbulent boundary layer. The boundary layer around the flat plate transitions to turbulence at high Reynolds number. Before the advent of supercomputing, it was not possible to numerically solve the Navier-Stokes equations for turbulent flows and fluid dynamicists had to resort to experimental studies to derive empirical relations for high Reynolds number flows. In this study, the low-fidelity approximation for the turbulent boundary layer is obtained using the one-seventh power law. The one-seventh power law[48] for computing the mean (or ensemble-averaged) velocity profile for flat-plate turbulent boundary layer is given as follows

$$\frac{\bar{u}}{u_\infty} \approx \begin{cases} \left(\frac{y}{\delta}\right)^{1/7} & \text{for} \quad y \le \delta, \\ 1 & \text{for} \quad y > \delta, \end{cases} \tag{12}$$

where u_∞ is the freestream velocity, y is the direction normal to the plate, and the turbulent boundary layer thickness δ is computed as follows[48]

$$\delta \approx \frac{0.38x}{(\mathrm{Re}_x)^{1/5}}, \tag{13}$$

where Re_x is the Reynolds number at a given x-location. There are many such empirical relations available to approximate turbulent boundary layers such as the log law, and Spalding's law of the wall[49]. The high-fidelity data is generated for the flat-plate turbulent boundary layer with zero pressure gradient by solving the incompressible RANS equations with the SIMPLE algorithm and $k-\omega$-SST turbulent model implemented in OpenFoam. The Reynolds number based on the length of the flat plate for turbulent boundary layer simulation is $\mathrm{Re}_L = 1 \times 10^7$. One important parameter in turbulence modeling is the dimensionless distance in the normal direction called wall y^+ and is defined as $y^+ = \sqrt{yu_\tau/\nu}$, where u_τ is the friction velocity. The friction velocity is calculated based on the wall shear stress as $u_\tau = \sqrt{\tau_w/\rho}$. The mesh is refined near the flat plate in such a way that the near-wall y^+ is below 5. The locations for collecting the data are sampled in such a way that more points are clustered near the leading edge and the wall, and are contaminated by adding a white Gaussian noise with zero mean and a standard deviation of 1.0 to mimic the measurement error. The formulation of turbulent boundary layer reconstruction is similar to the laminar boundary reconstruction as given in Equation 11 except for the low-fidelity model. The low-fidelity model prediction for turbulent boundary layer flow is calculated using Equation 12.

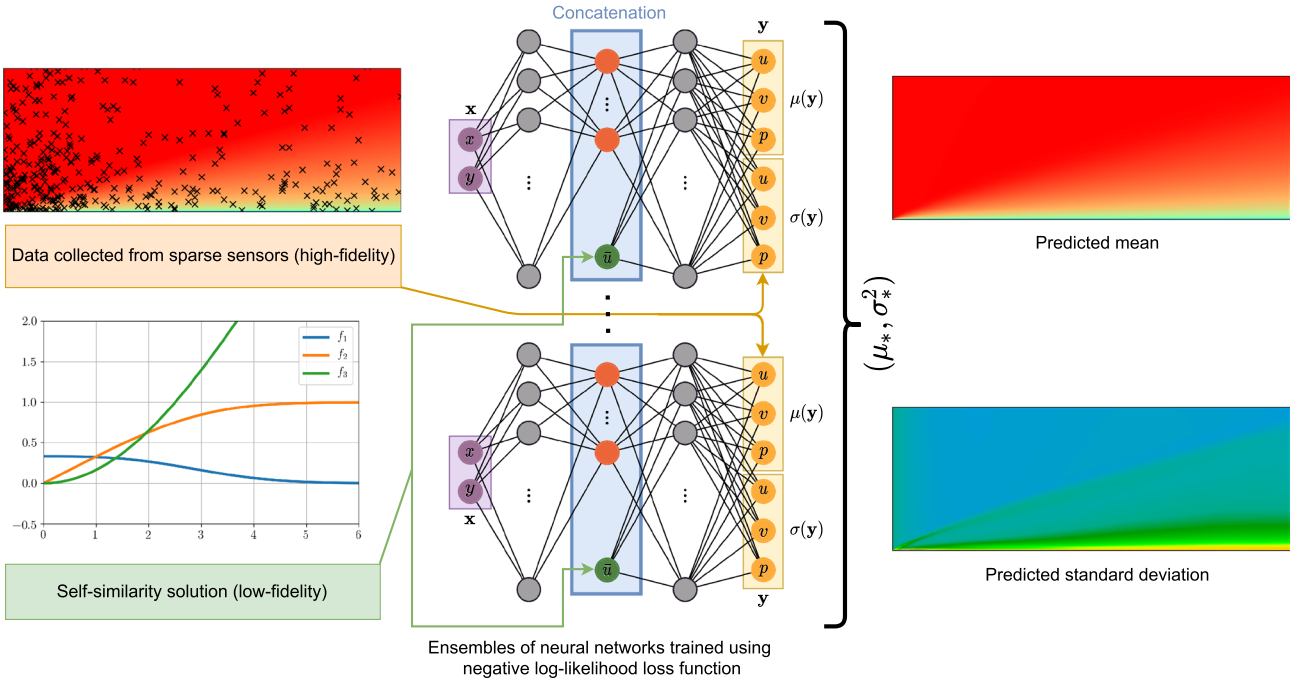

Figure 2. Illustration of the multi-fidelity data-fusion framework applied to the laminar flat plate boundary layer prediction task. The self-similarity Blasius solution is replaced by the one-seventh power law when applying to the turbulent boundary layer flows.

For both laminar and turbulent boundary layer reconstruction problem, we use a neural network with three hidden layers and twenty neurons in each hidden layer. The three hidden layers were found to be enough for providing sufficiently accurate prediction and hence we chose this architecture to reduce the computational overhead of training and inference. The prediction from the low-fidelity model is concatenated at the second hidden layer. Specifically, the equation for the concatenated neural network employed in this study can be written as follows

$$\mathbf{Y} = \zeta_4(\cdot;\mathbf{\Theta}_4)\circ\zeta_3(\mathbf{X};\mathbf{\Theta}_3)\circ\underbrace{\mathscr{C}\ (\zeta_2(\cdot;\mathbf{\Theta}_2),h(\mathbf{X}))}_{\text{Concatenation layer}}\circ\zeta_1(\mathbf{X};\mathbf{\Theta}_1), \tag{14}$$

where $\zeta_1,\ \zeta_2,\ \zeta_3$ are the ReLU activation functions, and ζ_4 is the linear activation function. The neural network architecture shown in Fig. 2 is representative of the network used within the proposed multi-fidelity data-fusion framework for the boundary layer reconstruction task. In terms of the trainable parameters, the ML model has 1,026 parameters, and the PGML model has 1,046 parameters.

Results

In this section, we demonstrate the capability of the proposed approach presented in Methods section to reconstruct laminar and turbulent boundary layer flows around the flat plate.

Laminar flow past a flat plate. We refer to a simple feed-forward neural network as the machine learning (ML) model and the concatenated neural network augmented with low-fidelity data is called the physics-guided machine learning (PGML) model. The ML model is trained solely based on the high-fidelity data, while the PGML model uses the prediction from a physics-based low-fidelity model (Blasius equations, see Eqs. 9 and 10) along with the high-fidelity data. Figure 3 shows the profile of the horizontal component of velocity versus distance from the wall, at the location $x/L = 0.5$, for different amounts of data used for training the ML and PGML models. The velocity is normalized with the freestream velocity and the vertical distance is normalized using the boundary layer thickness for laminar flow over the flat plate. The ML model fails to capture the accurate velocity profile when the velocity field information at 10% of locations within the computational domain is utilized for training. The mean velocity profile predicted by the PGML model is highly accurate even with just 10% of the observations. The predicted mean velocity profile is also accompanied by a confidence interval spanning one standard deviation (SD) on either side of the mean velocity profile and is an outcome of the uncertainty quantification mechanism built into deep ensembles. Deep ensembles achieve uncertainty quantification by training an ensemble of neural networks with the negative log-likelihood loss function. The uncertainty estimate associated with the PGML model is lower than the ML model and this is particularly notable near the wall within the boundary layer, i.e., for $y/\delta < 0.8$. However, we note that the prediction of laminar flow past a flat plate is a relatively simple task and therefore even the ML model is giving sufficiently accurate prediction. The improvement in the prediction by the PGML framework is noticeable for the turbulent flow past a flat plate, which will be presented in the following section.

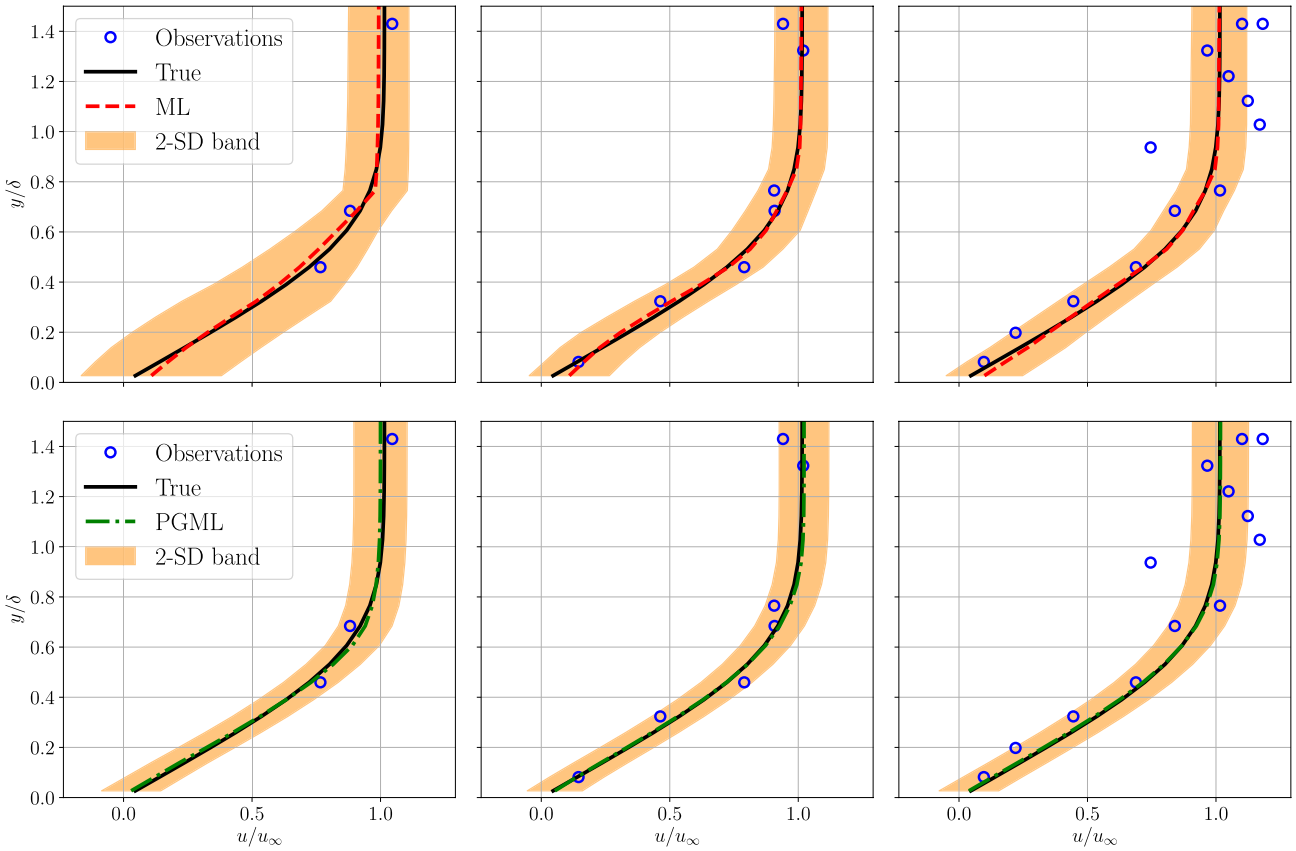

Figure 3. Boundary layer prediction for laminar flat plate flow at $x/L = 0.5$ along with the observations used for training the ML and PGML model. The amount of observations data used for training the model is 10% (left), 30% (middle), and 50% (right). The shaded area corresponds to two standard deviation (2-SD) band.

Turbulent flow past a flat plate. Next, we evaluate the performance of the proposed PGML approach for reconstruction of turbulent boundary layer flow over a (smooth) flat plate. The prior knowledge we concatenate in this case is the one-seventh power law velocity profile (i.e., see Eq. 12). Figure 4 displays the variation of the normalized velocity profile in the vertical direction at $x/L = 0.5$ for different amounts of data used for training the ML and PGML models. We can observe that the ML model performs very poorly for the data-sparse regime (i.e., 5% and 10% of the observations). Such situations are very common in scientific applications where a collection of high-fidelity data either from experiments or numerical simulations can be prohibitive. The PGML model on the other hand leads to an accurate prediction by exploiting the correlation between low- and high-fidelity data. The prediction from the ML model is also not reliable as indicated by the high width of the confidence band. Figure 5 shows the velocity profile in the *near-wall region* to illustrate how well the boundary condition is satisfied by the prediction obtained from both ML and PGML models. The velocity field predicted by the ML model is highly inaccurate in the *near-wall region* even when 30% of the observations are available for training. The PGML model provides very accurate prediction even in the low-data regime, and the prediction improves further as more data is used for training. The PGML model captures the slope of the velocity profile at the wall with very high accuracy. The slope of the boundary layer profile $\partial u/\partial y$ at the wall determines the skin friction drag along the wall. This quantity is not predicted accurately with the ML model and this can lead to poor estimation of the quantity of interests like the total drag. The PGML model is successful in predicting the correct slope of the velocity profile at the wall and therefore will lead to a more accurate estimation of total drag.

Figures 6, 7, 8 show the spatial variation of the predicted mean of the velocity field, confidence interval of two standard deviations, and the error with respect to the true velocity field near the wall region for 5%, 10%, and 30% of the training data, respectively. As the training data increases, the error decreases for both ML and PGML models. The confidence estimate associated with the PGML model is substantially higher (i.e., lower uncertainty) than the ML model for all three datasets. Moreover, the error of the PGML model is greatly reduced compared to the ML model. One other benefit of constructing a PGML approach is its modular nature that can provide an opportunity of bridging the gap between domain-specific knowledge and physics-agnostic models.

In our previous numerical experiments, we focused on the reconstruction task within the interpolation region. Both ML and PGML models were trained using the data sampled from the whole domain, i.e., up to $L = 2.0$, where L is the length of the flat plate. In our next numerical experiments, we sample observations only from the region till $L = 1.5$. Therefore, the region between $L = 1.5$ to $L = 2.0$ corresponds to the extrapolation region. We quantify the performance of the ML and PGML model using the variation of root mean squared error (RMSE) percentage along the streamwise direction as follows

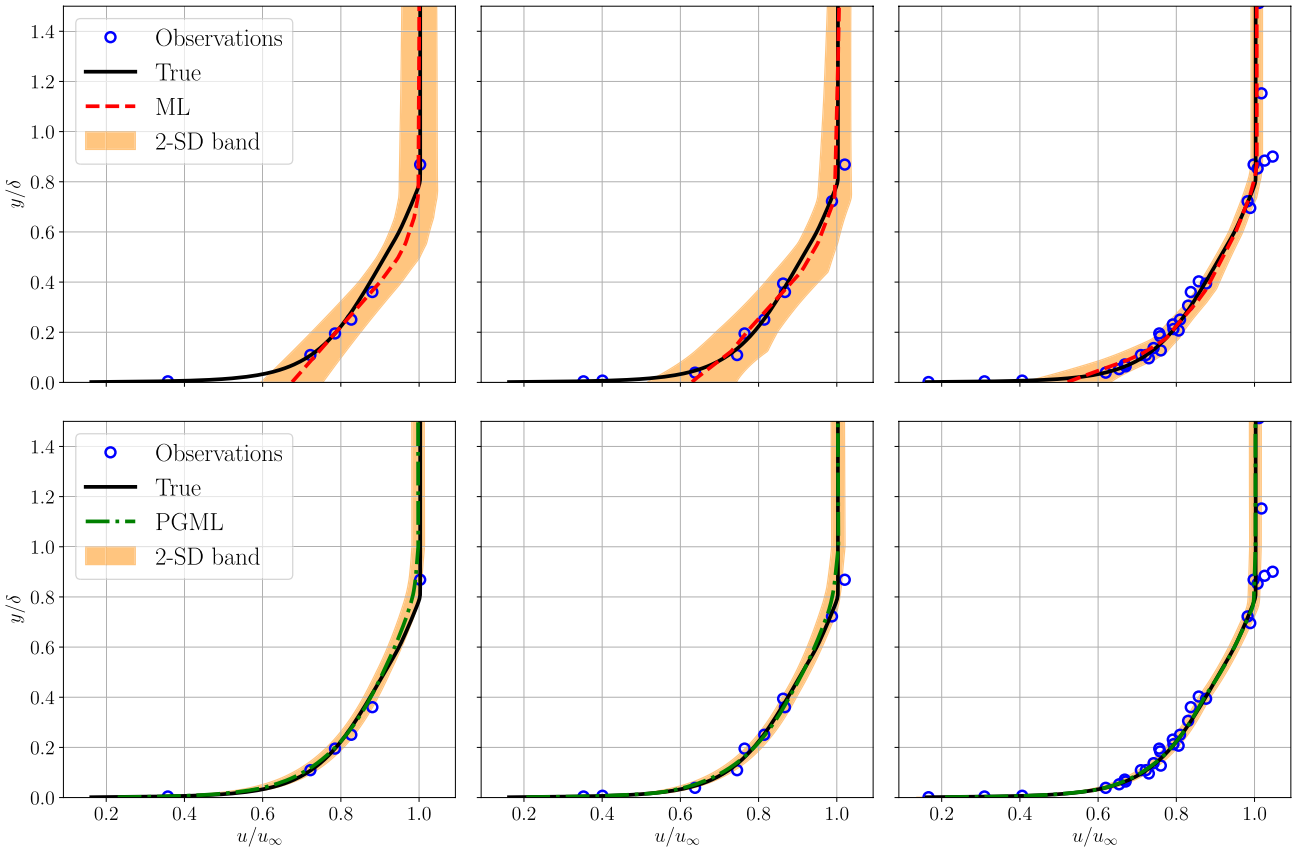

Figure 4. Boundary layer prediction for turbulent flat plate flow at $x/L = 0.5$ along with the observations used for training the ML and PGML model. The amount of observations data used for training the model is 5% (left), 10% (middle), and 30% (right). The shaded area corresponds to two standard deviation (2-SD) band.

$$\text{RMSE}(x) = 100 \times \left(\frac{1}{N_y} \sum_{j=1}^{N_y} \left(\frac{u_T(y_j) - u_P(y_j)}{u_T(y_j)} \right)^2 \right)^{1/2} \tag{15}$$

where u_T is the velocity of the high-fidelity model, u_P is the velocity predicted from the data-driven model, N_y is the spatial resolution in the wall-normal direction. From Figure 9, we can see that the RMSE increases substantially in the extrapolation region for the ML model, especially when the observations are very sparse, i.e., 5% of the data. This is a well-known limitation of DL models to extrapolate poorly in the absence of dense data. The PGML model on the other hand has RMSE almost one order of magnitude less than the ML model in the interpolation region. Additionally, the increase in RMSE is not significant in the extrapolation region. This shows that the PGML model is robust for the unseen condition, and it performs well for out-of-distribution examples.

Discussion

This study aims to develop a physics-guided machine learning (PGML) framework to improve data-driven models using prior knowledge from low-fidelity models. The PGML is a new deep neural network architecture that makes it possible to inject known physics during the training and *deployment* processes to reduce uncertainty and consequently improve the trade-off between efficiency and accuracy. Our design of a hierarchically sequential learning algorithm allows us to embed simplified theories, low order models, or empirical laws directly into deep learning models. These physics-based injections assist the neural network models in constraining the output to a manifold of the physically consistent solution and leads to improved reliability and generalizability. The PGML model trained using the deep ensembles algorithm provides us an estimate of the uncertainty associated with the prediction. This uncertainty information can be used for several applications like active learning, sensor placement, and optimization. Some of the questions addressed in this study are as follows

- How prior information on the physics of the problem can be used to improve black-box machine learning models?
- Can a concatenated neural network architecture augmented with a simplified or empirical model outperform a pure data-driven reconstruction model?
- How can these data-driven predictive models be quantified regarding their uncertainties?
- What is the generalizability of predictive performance across ML and PGML architectures, when these are applied to unseen conditions?

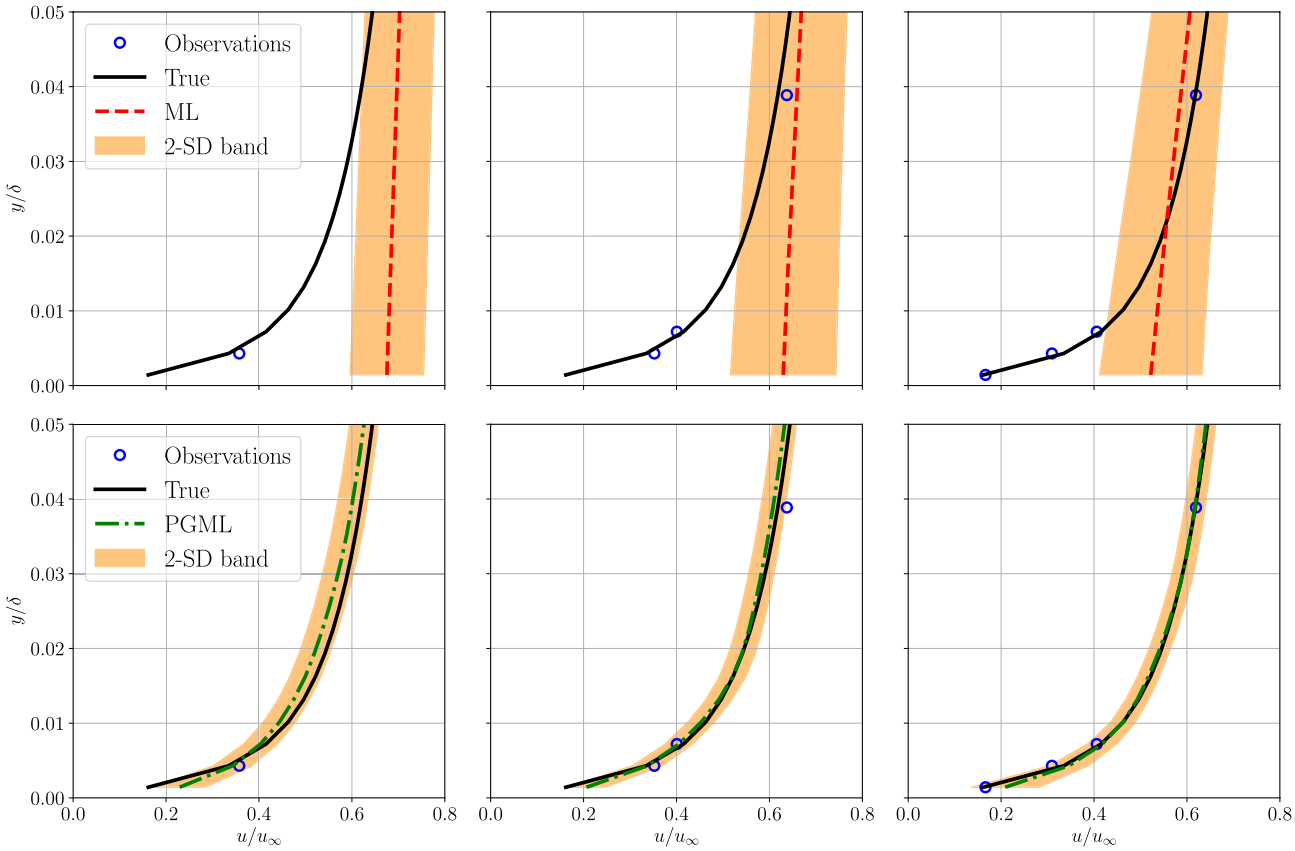

Figure 5. Boundary layer prediction in the *near-wall region* for turbulent flat plate flow at $x/L = 0.5$ along with the observations used for training the ML and PGML model. The amount of observations data used for training the model is 5% (left), 10% (middle), and 30% (right). The shaded area corresponds to two standard deviation (2-SD) band.

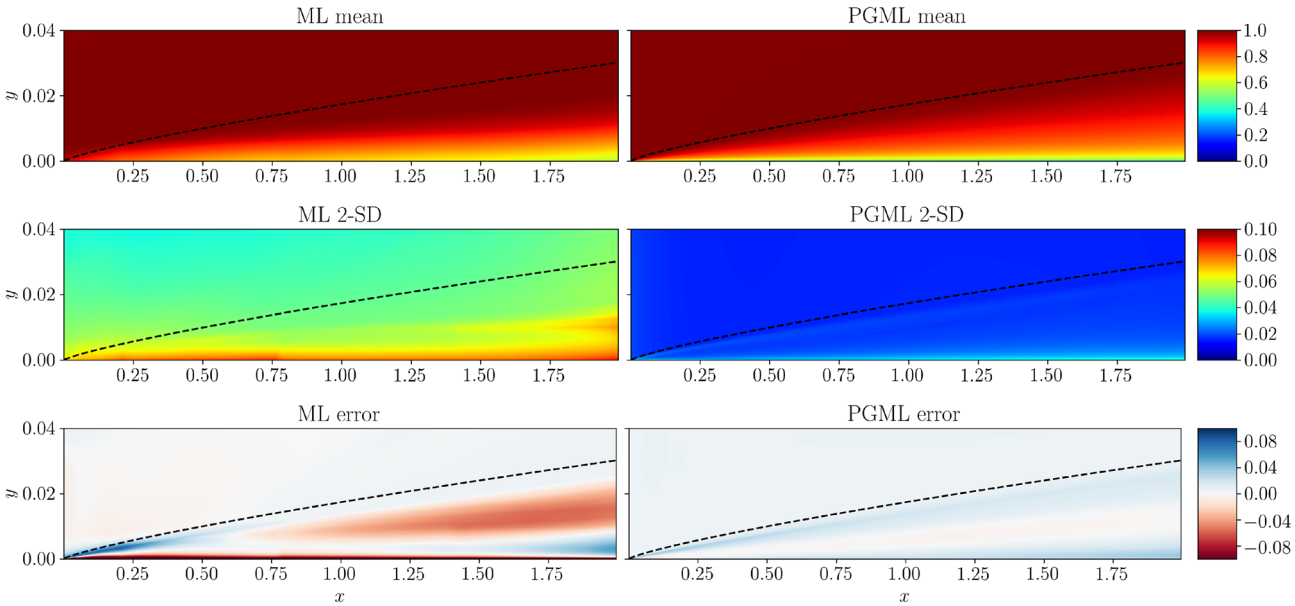

Figure 6. Prediction of the turbulent flat plate boundary layer with 5% of data used for training the ML and PGML model. The error is calculated as the difference between the true flow field and the flow field predicted by ML and PGML models. The figure is created by Matplotlib v3.5.1[50].

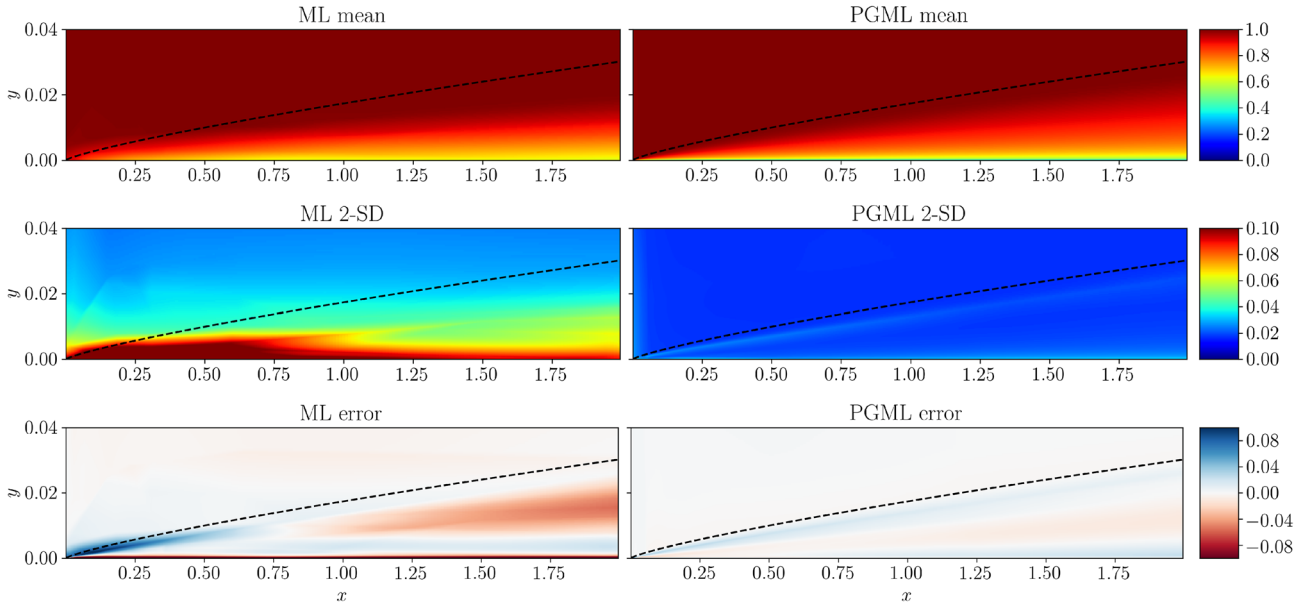

Figure 7. Prediction of the turbulent flat plate boundary layer with 10% of data used for training the ML and PGML model. The error is calculated as the difference between the true flow field and the flow field predicted by ML and PGML models. The figure is created by Matplotlib v3.5.1[50].

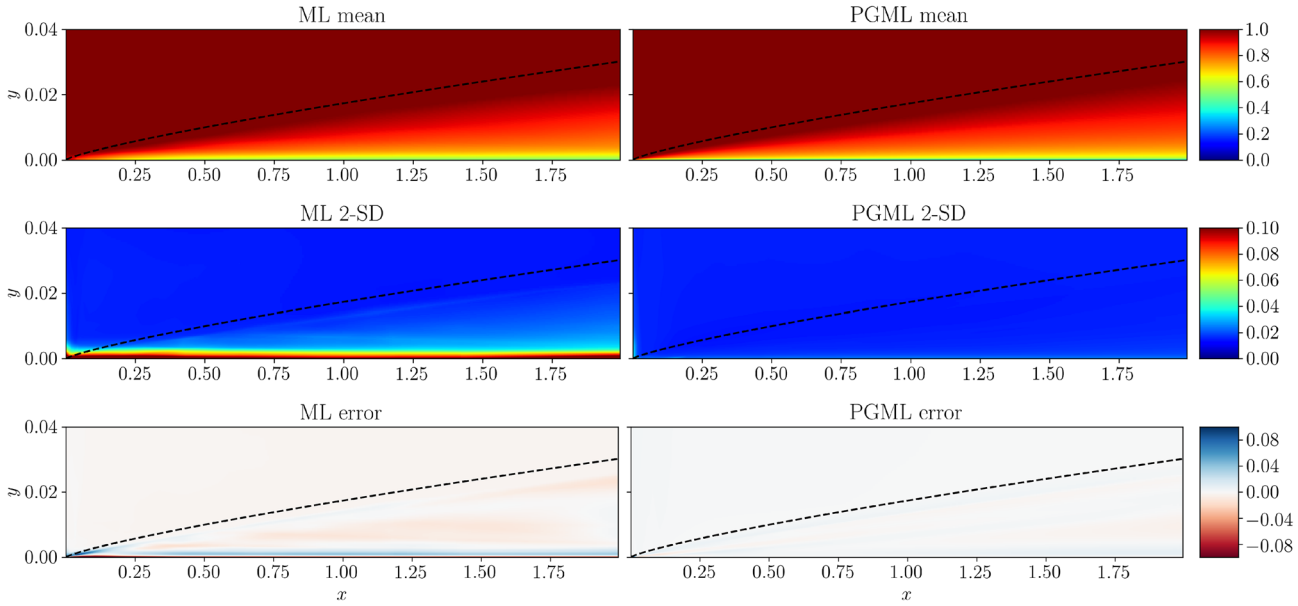

Figure 8. Prediction of the turbulent flat plate boundary layer with 30% of data used for training the ML and PGML model. The error is calculated as the difference between the true flow field and the flow field predicted by ML and PGML models. The figure is created by Matplotlib v3.5.1[50].

To provide a proof-of-concept of the PGML framework, we use the laminar and turbulent flow over a flat plate as the prototypical test cases where the self-similar solution is used as the low-fidelity model, and Reynolds-Averaged Navier-Stokes equations (RANS) is a high-fidelity model. Although our notion of high-fidelity here is relative to the selected low-fidelity model, the chief idea, which is scalable with different notions, is to use low-fidelity models to restrict the neural network to a manifold, such that less data is required to train the network and to improve its predictive capability under extrapolation. Our analysis indicates that an injection of the empirical relations like *one-seventh power law* improves the predictive model significantly for estimating canonical turbulent flat plate boundary layer flows. We found that the PGML outperforms its ML counterpart by reducing the RMSE to nearly an order of magnitude lower levels. We also demonstrated that the proposed PGML framework substantially reduces the model uncertainty even when only sparse observations are available. Furthermore, generalizability of results is also supported when we integrate our predictive models for unseen

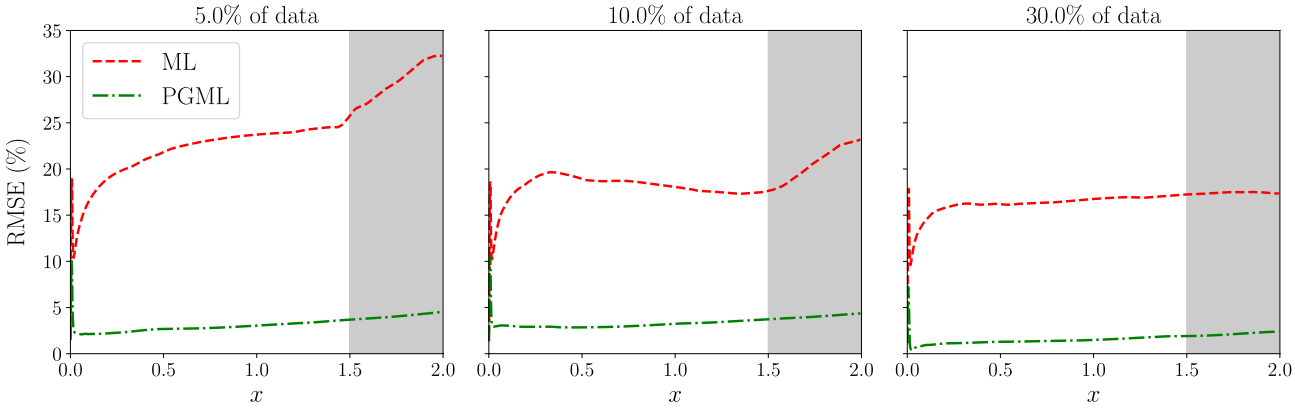

Figure 9. Variation of the normalized RMSE (in percentage) along the streamwise direction for ML and PGML models. The amount of observations data used for training the model is 5% (left), 10% (middle), and 30% (right). The gray color shaded area represents the extrapolation region.

conditions. Specifically, the RMSE distributions for the ML model (with respect to true data) are around 20% and 30% for interpolation and extrapolation regions, respectively. However, the PGML model has superior performance in both interpolation and extrapolation regions with the RMSE distribution in the range of 3%. For laminar flows, we also showed that a PGML approach (via injecting the Blasius solution) outperforms the ML approach. The Blasius approximation can also be extended for heat transfer problems (based on Falkner-Skan solutions) and for strongly nonlinear problems using non-similarity solutions[51].

The concatenated neural networks are capable of discovering a correlation between low- and high-fidelity data allowing for smaller training dataset sizes, training time, and improved extrapolation performance. Therefore, the PGML model has a great potential for a vast number of physical systems where a hierarchy of models is commonly used. The proposed framework is very modular and can be applied to a wide range of problems in fluid mechanics. Additionally, the framework is compatible with different neural network architectures making it suitable for complex high-dimensional problems. For example, one can treat the flow over a two-dimensional cylinder as the low-fidelity model for reconstructing the flow around a three-dimensional cylinder. Another interesting example is the wake prediction behind wind turbines, where analytical wake models can serve as the low-fidelity model for high-fidelity models like RANS solver. One of the challenges with the PGML framework is coupling of steady and unsteady solver, i.e., RANS as the low-fidelity model and large eddy simulation (LES) as the high-fidelity model. If one is interested only in the time-averaged quantities from LES simulation, injecting information from the RANS solution within the neural network will help in pruning the space of possible solutions. We plan to extend the PGML framework for these high-dimensional systems with more complex low- and high-fidelity data fusion in our future studies. We also highlight that the PGML approach could be useful for generating physics consistent initial conditions to accelerate large-scale high-fidelity computations with eliminating non-physical initial transient time.

Received: 1 September 2021; Accepted: 30 March 2022
Published online: 07 April 2022

References

1. Veers, P. *et al.* Grand challenges in the science of wind energy. *Science* **366**, eaau2027 (2019).
2. Brunton, S. L., Noack, B. R. & Koumoutsakos, P. Machine learning for fluid mechanics. *Ann. Rev. Fluid Mech.* **52**, 477–508 (2020).
3. Reichstein, M. *et al.* Deep learning and process understanding for data-driven earth system science. *Nature* **566**, 195–204 (2019).
4. Schmidt, J., Marques, M. R., Botti, S. & Marques, M. A. Recent advances and applications of machine learning in solid-state materials science. *NPJ Comput. Mater.* **5**, 1–36 (2019).
5. Duraisamy, K., Iaccarino, G. & Xiao, H. Turbulence modeling in the age of data. *Ann. Rev. Fluid Mech.* **51**, 357–377 (2019).
6. Stengel, K., Glaws, A., Hettinger, D. & King, R. N. Adversarial super-resolution of climatological wind and solar data. *Proc. Natl. Acad. Sci.* **117**, 16805–16815 (2020).
7. Chattopadhyay, A., Hassanzadeh, P. & Pasha, S. Predicting clustered weather patterns: A test case for applications of convolutional neural networks to spatio-temporal climate data. *Sci. Rep.* **10**, 1–13 (2020).
8. Fresca, S., Dede, L. & Manzoni, A. A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized pdes. *J. Sci. Comput.* **87**, 1–36 (2021).
9. McGovern, A. *et al.* Making the black box more transparent: Understanding the physical implications of machine learning. *Bull. Am. Meteorol. Soc.* **100**, 2175–2199 (2019).
10. Montavon, G.,Binder, A., Lapuschkin, S.,Samek, W. & Müller, K.-R. Layer-wise relevance propagation: an overview. *Explainable AI: Interpreting, Explaining and Visualizing Deep lLarning*193–209 (2019).
11. Ebert-Uphoff, I. & Hilburn, K. Evaluation, tuning, and interpretation of neural networks for working with images in meteorological applications. *Bull. Am. Meteorol. Soc.* **101**, E2149–E2170 (2020).
12. Bonavita, M.&Laloyaux, P. Machine learning for model error inference and correction. *J. Adv. Model. Earth Syst.* **12**,e2020MS002232 (2020).
13. Karniadakis, G. E. *et al.* Physics-informed machine learning. *Nat. Rev. Phys.* **3**, 422–440 (2021).
14. Ling, J., Kurzawski, A. & Templeton, J. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *J. Fluid Mech.* **807**, 155–166 (2016).
15. Zanna, L.&Bolton, T. Data-driven equation discovery of ocean mesoscale closures. *Geophys. Res. Lett.* **47**,e2020GL088376 (2020).

16. Mohan, A.T.,Lubbers, N., Livescu, D.&Chertkov, M. Embedding hard physical constraints in neural network coarse-graining of 3D turbulence. arXiv preprint arXiv:2002.00021 (2020).
17. Beucler, T. *et al.* Enforcing analytic constraints in neural networks emulating physical systems. *Phys. Rev. Lett.* **126**, 098302 (2021).
18. Greydanus, S.,Dzamba, M. & Yosinski, J. Hamiltonian neural networks. arXiv preprint arXiv:1906.01563 (2019).
19. Pawar, S., San, O., Aksoylu, B., Rasheed, A. & Kvamsdal, T. Physics guided machine learning using simplified theories. *Phys. Fluids* **33**, 011701 (2021).
20. Maulik, R., San, O., Rasheed, A. & Vedula, P. Subgrid modelling for two-dimensional turbulence using neural networks. *J. Fluid Mech.* **858**, 122–144 (2019).
21. Tai, K. S.,Bailis, P. & Valiant, G. Equivariant transformer networks. In *International Conference on Machine Learning*, 6086–6095 (PMLR, 2019).
22. Raissi, M., Perdikaris, P. & Karniadakis, G. E. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *J. Comput. Phys.* **378**, 686–707 (2019).
23. Wu, J.-L. *et al.* Enforcing statistical constraints in generative adversarial networks for modeling chaotic dynamical systems. *J. Comput. Phys.* **406**, 109209 (2020).
24. Geneva, N. & Zabaras, N. Modeling the dynamics of pde systems with physics-constrained deep auto-regressive networks. *J. Comput. Phys.* **403**, 109056 (2020).
25. San, O.,Rasheed, A. & Kvamsdal, T. Hybrid analysis and modeling, eclecticism, and multifidelity computing toward digital twin revolution. *GAMM-Mitteilungen* e202100007 (2021).
26. Kashinath, K. *et al.* Physics-informed machine learning: case studies for weather and climate modelling. *Philos. Trans. R. Soc. A* **379**, 20200093 (2021).
27. Hess, J. L. Panel methods in computational fluid dynamics. *Annu. Rev. Fluid Mech.* **22**, 255–274 (1990).
28. Moin, P. & Mahesh, K. Direct numerical simulation: a tool in turbulence research. *Annu. Rev. Fluid Mech.* **30**, 539–578 (1998).
29. Archer, C. L. *et al.* Review and evaluation of wake loss models for wind energy applications. *Appl. Energy* **226**, 1187–1207 (2018).
30. Breton, S.-P. *et al.* A survey of modelling methods for high-fidelity wind farm simulations using large eddy simulation. *Philos. Trans. R. Soc. A Math. Phys. Eng. Sci.* **375**, 20160097 (2017).
31. Smits, A. J. & Marusic, I. Wall-bounded turbulence. *Phys. Today* **66**, 25–30 (2013).
32. Jiménez, J. Cascades in wall-bounded turbulence. *Annu. Rev. Fluid Mech.* **44**, 27–45 (2012).
33. Bhaganagar, K., Kim, J. & Coleman, G. Effect of roughness on wall-bounded turbulence. *Flow Turbul. Combust.* **72**, 463–492 (2004).
34. Wu, X. & Moin, P. Direct numerical simulation of turbulence in a nominally zero-pressure-gradient flat-plate boundary layer. *J. Fluid Mech.* **630**, 5–41 (2009).
35. Elsken, T., Metzen, J. H. & Hutter, F. Neural architecture search: A survey. *J. Mach. Learn. Res.* **20**, 1997–2017 (2019).
36. Hutter, F.,Kotthoff, L. & Vanschoren, J. *Automated Machine Learning: Methods, Systems, Challenges* (Springer, 2019).
37. Amodei, D. *et al.* Concrete problems in ai safety. arXiv preprint arXiv:1606.06565 (2016).
38. Lakshminarayanan, B., Pritzel, A. & Blundell, C. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 6405-6416 (Curran Associates Inc., 2017).
39. Neal, R. M. *Bayesian Learning for Neural Networks*, Vol. 118 (Springer, 2012).
40. Davison, A. C. *Statistical Models* Vol. 11 (Cambridge University Press, 2003).
41. Nielsen, M. A. *Neural Networks and Deep Learning* Vol. 25 (Determination Press, 2015).
42. Dietterich, T. G. Ensemble methods in machine learning. In *International Workshop on Multiple Classifier Systems* 1–15 (Springer, 2000).
43. Ferreira, A. J.& Figueiredo, M. A. Boosting algorithms: A review of methods, theory, and applications. In *Ensemble Machine Learning*, 35–85 (Springer, 2012).
44. Lee, S.,Purushwalkam, S., Cogswell, M.,Crandall, D. & Batra, D. Why m heads are better than one: Training a diverse ensemble of deep networks. arXiv preprint arXiv:1511.06314 (2015).
45. White, F. M. & Majdalani, J. *Viscous Fluid Flow* Vol. 3 (McGraw-Hill, 2006).
46. Moin, P. *Fundamentals of Engineering Numerical Analysis* (Cambridge University Press,2010).
47. Issa, R. I. Solution of the implicitly discretised fluid flow equations by operator-splitting. *J. Comput. Phys.* **62**, 40–65 (1986).
48. Çengel, Y. A. & Cimbala, J. M. *Fluid Mechanics: Fundamentals and Applications* (McGraw-Hill, 2010).
49. Spalding, D. B. A Single Formula for the "Law of the Wall". *J. Appl. Mech.* **28**, 455–458 (1961).
50. Hunter, J. D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* **9**, 90–95 (2007).
51. Liao, S.-J. A general approach to get series solution of non-similarity boundary-layer flows. *Commun. Nonlinear Sci. Numer. Simul.* **14**, 2144–2159 (2009).

Acknowledgements

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research under Award Number DE-SC0019290. O.S. gratefully acknowledges their Early Career Research Program support.

Author contributions

S.P. conducted the numerical experiments, analysed data and wrote the manuscript. O.S. and A.R. conceived the proposed PGML approach. O.S. supervised the project. S.P., O.S., P.V., A.R., and T.K. reviewed the manuscript, discussed the results, and contributed to the final manuscript.

Competing interests

The authors declare no competing interests.

Additional information

Correspondence and requests for materials should be addressed to O.S.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.