Information fusion and machine learning for sensitivity analysis using physics knowledge and experimental data

Berkcan Kapusuzoglu, Sankaran Mahadevan*

Department of Civil and Environmental Engineering, Vanderbilt University, TN 37235

Abstract

Sensitivity analysis is often performed using computational models of the physical phenomena; such models could be physics-based or data-driven, and the sensitivity estimate is affected by the accuracy and uncertainty of the physics model. If the physics-based computational model is expensive, an inexpensive surrogate model is built and used to compute the Sobol' indices in variance-based sensitivity analysis (also referred to as global sensitivity analysis, or GSA) since many input-output samples are required for such computation; and the surrogate model introduces further approximation in the sensitivity estimate. This paper considers GSA for situations where both a physics-based model and a small number of experimental observations are available, and investigates strategies to effectively combine the two sources of information in order to maximize the accuracy and minimize the uncertainty of the sensitivity estimate. Physics-informed and hybrid machine learning strategies are proposed to achieve these objectives. Two machine learning (ML) techniques are considered, namely, deep neural networks (DNN) and Gaussian process (GP) modeling, and two strategies for incorporating physics knowledge within these ML techniques are investigated, namely: (i) incorporating loss functions in the ML models to enforce physics constraints, and (ii) pre-training and updating the ML model using simulation data and experimental data respectively. Four different models are built for each type (DNN and GP), and the uncertainties in these models are also included in the Sobol' indices computation. The DNN-based models, since they have many degrees of freedom in terms of

*Corresponding author

Phone: 1-615-322-3040. Fax: 1-615-322-3365. Postal address: Vanderbilt University, VU Station B 356077, Nashville, TN 37235-6077, United States.

E-mail address: sankaran.mahadevan@vanderbilt.edu (S. Mahadevan).

model parameters, are found to result in more consistent sensitivity computation results with much smaller prediction bounds when compared to GP-based models. The proposed methods are illustrated for an additive manufacturing example.

Keywords: Global sensitivity analysis, Sobol index, Additive manufacturing, Fused filament fabrication, Physics-informed machine learning, Deep learning

1. Introduction

Engineering systems are often described by a model to predict the response of the system since conducting experiments to directly measure the true response is often not feasible. The computational model used to represent the system is usually an incomplete representation of the complex physical process by approximating the reality. In this case, the system response is affected by the model uncertainty. The various sources of uncertainty are classified as (a) epistemic uncertainty due to lack of knowledge, and (b) aleatory uncertainty due to the inherent variability in the system. Global sensitivity analysis (GSA) [1] provides a quantitative assessment of the relative importance of each uncertainty source to the uncertainty in the model response [2, 3, 4]. GSA techniques can be either data-driven (i.e., based on analysis of variance ANOVA) or model-based (e.g., the computation of Sobol indices [5]). Generally, two types of Sobol indices are estimated, namely the first-order Sobol index and total-effect Sobol index [6]. A straightforward way to estimate Sobol indices is to use a double-loop Monte Carlo (MC) simulation.

Model outputs can have uncertainty even for a fixed input model-based prediction when there exists model uncertainty. When the model is computationally expensive, we replace it with a surrogate model, which introduces additional uncertainty, to facilitate the estimation of Sobol indices. Several surrogate models are used in the literature: polynomial chaos, Gaussian process regression, and neural networks, etc, to train a parametric relationship between the inputs and outputs. The quality and quantity of the training data affect the accuracy of these surrogate models, which directly affects the uncertainty in the model response [7, 8, 9]. Thus, it is important to quantify the surrogate model uncertainty. In high dimensional problems,

the use of surrogate model-based GSA may be an issue [7, 10]. To overcome this issue, data-driven methods are proposed for computing the Sobol indices based on available input-output data [10, 11].

In recent years, research efforts have focused on model-based methods for representing physical processes. Although, physics-based models do not require large amounts of data, they are generally limited by their computational complexity or incomplete physics. Further, the estimation of Sobol indices require a significant number of model runs. Nowadays, several studies have built experimental data-driven machine learning (ML) models to predict the quantity of interest (QoI). Generally, the construction of ML models does not require in-depth knowledge of the complex physics inherent in the physical process. ML models can learn complex systems using available observations, but the accuracy of these models depends on the quality and quantity of the experimental data. If the available data is limited, then the true nature of the process cannot be captured. Further, since ML model predictions do not consider physical laws, they can produce physically inconsistent results.

A particular problem of interest is the combined use of physics knowledge with experimental data to estimate Sobol indices. The accuracy of Sobol indices depend on the prediction capabilities of the ML models. Thus, fusing two types of information can improve the accuracy and efficiency of GSA computations. The Gaussian process (GP) surrogate model uncertainty is included in the Sobol index estimates with a Monte Carlo procedure using multiple realizations. This allows us to derive prediction intervals for the Sobol index estimates. In a similar manner, Monte Carlo (MC) dropout is used to estimate the deep neural network (DNN) model uncertainty and propagate that uncertainty into the GSA results. Gal and Ghahramani [12] showed that performing approximate variational inference is equivalent to MC dropout, which infers the posterior by performing dropout not only while training a model but also at test time. The optimization of Bayesian neural networks with the MC dropout is equivalent to using dropout as regularization on neural networks. The simplicity of the MC dropout strategy provides an efficient way of Bayesian inference to quantify the model prediction uncertainty with variety of neural networks, such as feedforward neural networks, convolutional neural networks, and

recurrent neural networks.

The combined use of physics-based and ML models achieve more accurate and physically consistent predictions by leveraging the advantages of each method [13]. In this work, we incorporated the physics knowledge into the ML models to better capture the physics of the process by leveraging physical laws while improving the generalization performance of data-driven models. In this paper, two ML models are considered in this paper, namely, GP and DNN. Four different physics-informed machine learning (PIML) models are developed for both GP and DNN models to predict the QoI of the physical process through combinations of two strategies: (1) incorporating loss functions in the ML models to enforce physics constraints, and (2) pre-training the ML model with simulation data and then updating it with experimental data. The proposed methods use multiple physics-based loss functions and address a physical QoI for enhancing the experimental data-driven ML models for the considered physical process. The physics constraints exploit the physical laws related to the physical process.

In order to identify the relative contribution of model inputs to the uncertainty in the model output, the physics process needs to be repeated by running multiple experiments with different values of process parameters. This is practically impossible; thus physics-based modeling approaches are used to estimate Sobol indices. However, the computation of Sobol indices require a significant number of MC simulations; thus it is not affordable when the physics models are computationally expensive. To handle this problem, we built eight different PIML models (four GP-based and four DNN-based) to approximate the physical process and to estimate the Sobol indices with less computational effort and experimental data. The proposed methodology for estimating the Sobol indices also takes model uncertainty into account.

In summary, the contributions of this paper are as follows:

- Effective fusion of physics knowledge with experimental observations in order to maximize the accuracy and minimize the uncertainty of sensitivity estimates.

- Two PIML strategies and their combinations are developed for model predictions using physics knowledge and experimental data.

- Four different models are built for both GP and DNN, and the uncertainties in these models are also included in the Sobol' indices computation.

- Evaluation of the relative contributions of uncertainty sources to the uncertainty in the model output, which facilitates a quantitative basis for optimal resource allocation.

The outline of the rest of this paper is as follows. Section 2 provides background information on related methods. Section 3 presents the proposed methodology. The proposed methodology is illustrated for a numerical example in Section 4. Concluding remarks are provided in Section 5.

2. Background

2.1. Probabilistic sensitivity analysis

A probabilistic sensitivity analysis, commonly referred to as global sensitivity analysis (GSA) is used to assess the relative contribution of each uncertainty source towards the uncertainty of the model output. Variance-based GSA, using Sobol' indices, is adopted in this paper, as briefly described below. The GSA results provide a quantitative basis for optimal resource allocation.

2.1.1. Variance-based global sensitivity analysis

Consider a deterministic real integrable one-to-one system response function $\mathrm{Y} = f(\mathbf{X})$, where $f(\cdot)$ is the computational model, $\mathbf{X} = \{\mathrm{X}_1, ..., \mathrm{X}_k\}$ are mutually independent model inputs, and Y is the model output. As shown by [5], the variance of Y can be decomposed as

$$V(\mathrm{Y}) = \sum_{i}^{k} V_i + \sum_{i_1}^{k} \sum_{i_2=i_1+1}^{k} V_{i_1 i_2} + \sum_{i_1}^{k} \sum_{i_2=i_1+1}^{k} \sum_{i_3=i_2+1}^{k} V_{i_1 i_2 i_3} + ... + V_{12...k} \tag{1}$$

where V_i is the variance of Y caused by only $\mathrm{X_i}$, $V_{i_1...i_p} (p \geq 2)$ indicates the variance of Y caused by the interactions of $\{\mathrm{X_{i_1}}, ..., \mathrm{X_{i_p}}\}$.

The Sobol indices are defined by dividing both sides of Eq. (1) with $V(\mathrm{Y})$

$$1 = \sum_{i}^{k} S_i + \sum_{i_1}^{k} \sum_{i_2=i_1+1}^{k} S_{i_1 i_2} + \sum_{i_1}^{k} \sum_{i_2=i_1+1}^{k} \sum_{i_3=i_2+1}^{k} S_{i_1 i_2 i_3} + ... + S_{12...k} \tag{2}$$

where S_i is the first-order or main effects index that assess the contribution of $\mathrm{X_i}$ individually to the variance of the output Y without considering the interactions with other inputs. The variables with higher first-order index are the important ones to be accounted for. Other indices $S_{i_1 \ldots i_p} (p \geq 2)$ in Eq. (2) are higher-order indices that measure the contribution of the interactions of $\{\mathrm{X_{i_1}}, ..., \mathrm{X_{i_p}}\}$.

The evaluation of S_i is defined as follows:

$$S_i = \frac{V_i}{V(\mathrm{Y})} = \frac{V_{X_i}(E_{\mathbf{X}_{-i}}(\mathrm{Y}|\mathrm{X}_i))}{V(\mathrm{Y})} \tag{3}$$

where $\mathbf{X}_{-i}$ are all the model inputs other than X_i.

The overall contribution of $\mathrm{X_i}$ considering both individual effect and interactions with other inputs is measured by the total effects index S_i^T:

$$S_i^T = 1 - \frac{V_{-i}}{V(\mathrm{Y})} = \frac{V_{\mathbf{X}_{-i}}(E_{\mathrm{x}_i}(\mathrm{Y}|\mathbf{X}_{-i}))}{V(\mathrm{Y})}. \tag{4}$$

An important challenge is the computation of Sobol indices. The computation of S_i analytically is nontrivial since $E_{\mathbf{X}_{-i}}(\cdot)$ requires multi-dimensional integrals. The methodologies that have been used to reduce the computational cost of the Sobol indices can be categorized into analytical and sample-based methods. In the former one, the original computational model $f(\cdot)$ is generally approximated by a surrogate model in order to convert the multi-dimensional integral into multiple uni-variate integrals. Whereas, the latter one, which has been explained earlier, generates samples such as the double-loop MCS to evaluate the indices.

2.2. Gaussian process surrogate modeling

A GP surrogate model is constructed to approximate multi-physics model. The GP surrogate model accounts for the statistical dependence between the inputs and outputs.

In the GP model, the response at prediction point $\mathbf{u}$, $G(\mathbf{u})$ is described by:

$$G(\mathbf{u}) = \mathbf{h}(\mathbf{u})^T \boldsymbol{\beta} + Z(\mathbf{u}) \tag{5}$$

where $\mathbf{h}(\cdot)$ is the trend of the model, $\boldsymbol{\beta}$ is the vector of trend coefficients, and $Z(\cdot)$ is a zero-mean stationary GP which describes the deviation of the model from the trend. The covariance between the outputs $Z(\cdot)$ of the GP surrogate at points $\mathbf{a}$ and $\mathbf{b}$ is defined as:

$$\mathrm{Cov}[Z(\mathbf{a}), Z(\mathbf{b})] = \sigma_Z^2 R(\mathbf{a}, \mathbf{b}) \tag{6}$$

where σ_Z^2 is the process variance and $R(\boldsymbol{\cdot}, \boldsymbol{\cdot})$ is the correlation function. A squared exponential function with separated length scale parameters l_i for each input dimension has often been used in the literature:

$$R(\mathbf{a}, \mathbf{b}) = \exp\left[-\sum_{i=1}^{\mathrm{M}} \frac{(a_i - b_i)^2}{l_i}\right] \tag{7}$$

The hyperparameters of the GP model $\boldsymbol{\Theta} = \{l, \sigma_Z, \sigma_{obs}\}$, where σ_{obs} is the observation error, are inferred from the given data. A common method is to maximize the log marginal likelihood function, which is defined as

$$\log\, p(\mathbf{Y}|\mathbf{X}; \boldsymbol{\Theta}) = -\frac{1}{2}\mathbf{Y}(\mathbf{K}_{TT} + \sigma_{obs}^2\mathbf{I})^{-1}\mathbf{Y} - \frac{1}{2}\log|\mathbf{K}_{TT} + \sigma_{obs}^2\mathbf{I}| + \frac{n}{2}\log 2\pi. \tag{8}$$

The outputs of the GP model are the mean prediction $\mu_G(\cdot)$ and the variance of the prediction $\sigma_G^2(\cdot)$, defined as:

$$\mu_G(\mathbf{u}) = \mathbf{h}(\mathbf{u})^T\boldsymbol{\beta} + \mathbf{r}(\mathbf{u})^T\mathbf{R}^{-1}(\mathbf{g} - \mathbf{F}\boldsymbol{\beta}) \tag{9}$$

$$\sigma_G^2(\mathbf{u}) = \sigma_Z^2 - \mathbf{A}\begin{bmatrix} \mathbf{0} & \mathbf{F}^T \\ \mathbf{F} & \mathbf{R} \end{bmatrix}^{-1}\mathbf{A}^T \tag{10}$$

where $\mathbf{r}(\mathbf{u})$ is a vector containing the covariance between $\mathbf{u}$ and each of the training points $\{x_1, x_2, ..., x_n\}$, $i \in \{1, ..., n\}$, $\mathbf{R}$ is an $n \times n$ matrix containing the correlation between each pair of training points, $\mathbf{R}(x_i,\ x_j) = \mathrm{Cov}[Z(x_i), Z(x_j)]$; $\mathbf{g}$ is the vector of original physics model outputs at each of the training points, $\mathbf{F}$ is a $n \times q$ matrix with rows $\mathbf{h}(\mathbf{u}_i)^T$, and $\mathbf{A} = \left[\mathbf{h}(\mathbf{u})^T\ \mathbf{r}(\mathbf{u})^T\right]$.

2.3. Deep neural networks

A deep neural network (DNN) is composed of multiple hidden layers and has four major components: neuron, activation function, cost function, and optimization. Figure 1 shows a neural network consisting of three inputs, two hidden layers, each having four neurons, and two output neurons. The values of various input variables of a particular neuron are multiplied by their associated weights, then the sum of the products of the neuron weights and the inputs are calculated at each neuron. The summed value is passed through an activation function that maps the summed value to a fixed range before passing these signals on to the next layer of neurons.

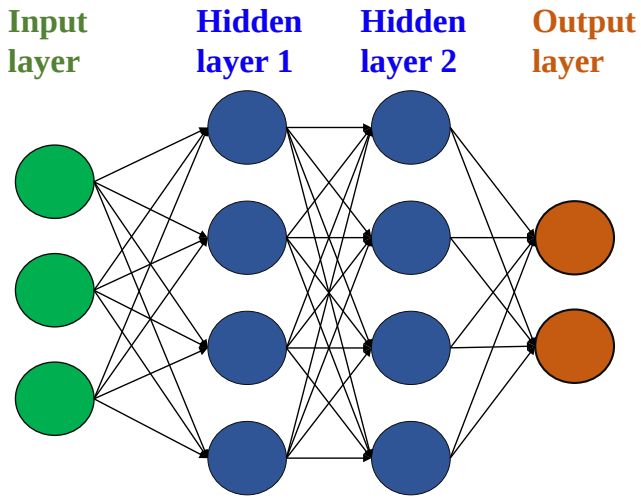


Fig. 1. A deep neural network with two hidden layers.

The predictions of the DNN after a forward propagation, $\hat{\mathbf{Y}}$, are compared against the true response of the system, $\mathbf{Y}$, by defining a loss function (e.g., root mean squared error (RMSE); $\mathcal{L}_{\text{RMSE}}(\mathbf{Y}, \hat{\mathbf{Y}}) = \sqrt{\sum_{i=1}^{n}(\mathrm{y}_i - \hat{\mathrm{y}}_i)^2/n}$), which measures how far off the predictions are from the observations for the n training samples. After the forward propagation, backpropagation algorithms are employed to keep track of small perturbations to the weights that affect the error at the output and to distribute this error back through the network layers by computing gradients for each layer using the chain rule. In order to minimize the value of the loss function, necessary adjustments are applied at each iteration to the neuron weights in each layer of the network. These procedures are performed at each iteration until the loss value converges to a stable value.

2.3.1. Bayesian neural network

In the Bayesian context, the distribution of the model parameters (neuron weights $\mathbf{w}$) that are most likely to generate the observed data are inferred. A prior distribution over the neuron weights $p(\mathbf{w})$ and a likelihood function $p(\mathbf{Y}|\mathbf{X}, \mathbf{w})$ are defined to represent the probability of generating the observed data given model parameters. A common way to define the prior distribution is to place a Gaussian distribution $p(\mathbf{w} = \mathcal{N}(0, \mathcal{I}))$ to replace the deterministic network's weight parameters with distributions, which results in a model referred to as a Bayesian neural network (BNN) [14, 15, 16] that captures epistemic uncertainty in a network. Following the Bayes' theorem, a posterior distribution over the model parameters given the training set $\{\mathbf{X}, \mathbf{Y}\} = \{\{\mathbf{x}_1, ..., \mathbf{x}_N\}, \{\{\mathbf{y}_1, ..., \mathbf{y}_N\}\}$ is defined by

$$p(\mathbf{w}|\mathbf{X}, \mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{X}, \mathbf{w})p(\mathbf{w})}{p(\mathbf{Y}|\mathbf{X})}. \tag{11}$$

The Bayesian inference of the model outputs is given by the predictive distribution on a new data $\mathbf{x}^*$ as follows:

$$p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) = \int_{\Omega} p(\mathbf{y}^*|\mathbf{x}^*, \mathbf{w})p(\mathbf{w}|\mathbf{X}, \mathbf{Y})d\mathbf{w}. \tag{12}$$

The posterior distribution of model parameters $p(\mathbf{w}|\mathbf{X}, \mathbf{Y})$ cannot be evaluated analytically over the whole parameter space Ω due to highly non-linear behavior in the neural network, which is caused by the non-linear activation functions and their combinations across multiple hidden layers. Thus, it becomes difficult to perform inference in BNNs.

There exists different approximate inference techniques to infer the posterior distribution $p(\mathbf{w}|\mathbf{X}, \mathbf{Y})$ [17, 18, 19, 20], such as variational inference, which fits a simple and tractable distribution $q_\theta(\mathbf{w})$ to the posterior, parametrized by a variational parameter θ [17]. This approximates the intractable problem by optimizing the parameters of $q_\theta(\mathbf{w})$. The closeness of the variational distribution is often measured by the Kullback-Leibler (KL) divergence between the approximate distribution $q_\theta(\mathbf{w})$ and the true model posterior $p(\mathbf{w}|\mathbf{X}, \mathbf{Y})$. The posterior distribution inferred using Monte Carlo (MC) dropout is equivalent to performing approximate variational inference. In MC dropout, dropout is not only applied while training a model but also at test time. The

randomly chosen neurons are temporarily removed from the network along with its connections. Next, the gradients of neurons weights are calculated on a sub-neural network for each training data and these gradients are then averaged over the training sets in the mini-batch to obtain the weights of overall network. The optimization of Bayesian neural networks with the MC dropout is equivalent to using dropout as regularization on neural networks. However, in contrast to standard neural networks, the MC dropout performs dropout and generates random samples following a Bernoulli distribution for each neuron in the input and hidden layers during testing. The dropout is applied to the neuron that takes the value of 0 with a given dropout probability p_d. The outputs of the network are predicted using the collection of generated random samples from the posterior predictive distribution and the uncertainty in the prediction of a new data is quantified with the trained network. Moreover, the MC dropout strategy provides an efficient way of Bayesian inference to quantify the model prediction uncertainty with variety of neural networks, such as feedforward neural networks, convolutional neural networks, and recurrent neural networks.

3. Proposed methodology

The proposed methodology for sensitivity analysis using physics knowledge and experimental data consists of the following steps:

1. Two PIML strategies
2. Implementation in GP
3. Implementation in DNN
4. Model uncertainty quantification in GP and DNN
5. Sobol indices computation with model uncertainty

The following subsections describe these steps in detail.

3.1. PIML strategies

In PIML models, complementary strengths of both models are leveraged by integrating physics knowledge and experimental data in a synergistic manner [21]. The aim is to improve

the predictions beyond that of physics-based models or ML models alone by coupling physics-based models with ML models. In the following, two different strategies to combine physics knowledge and ML models are pursued: (1) incorporate physics constraints in the ML models, and (2) pre-train and update ML models using physics model input-output and experimental data, respectively.

3.1.1. Physics-informed loss functions

A direct strategy to improve ML model predictions is by including physics-based loss functions [13]. Consider a PIML model with inputs $\boldsymbol{X}$ and outputs $\hat{\boldsymbol{Y}}$ trained using physical laws that are incorporated as constraints into the loss function:

$$\mathcal{L} = \mathcal{L}_{\text{ML}} + \lambda_{\text{phy}} \mathcal{L}_{\text{phy}}(\hat{\boldsymbol{Y}}), \tag{13}$$

where $\mathcal{L}_{\text{ML}}$ is the log marginal likelihood of the data $\log p(\mathbf{Y}|\mathbf{X};\boldsymbol{\Theta})$ for a GP and regular training loss for a DNN that evaluates a supervised error (e.g., root mean squared error (RMSE); $\mathcal{L}_{\text{DNN}}(\boldsymbol{Y}, \hat{\boldsymbol{Y}}) = \sqrt{\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2/n})$, that measures the accuracy of predictions $\hat{\boldsymbol{Y}}$ for n training samples. The physics-based loss function $\mathcal{L}_{\text{phy}}$ is controlled by a hyperparameter λ_{phy} and its inclusion of ensures physically consistent model predictions and helps decreasing the generalization error even when there is a small amount of training data [13]. We note that only simulation data is used to evaluate physics-based loss functions.

The physical inconsistencies in the model predictions are evaluated using the physics-based loss functions. The generic forms of these physical relationships can be expressed using the following equations:

$$\begin{aligned} \mathcal{F}_1(\boldsymbol{Y}, \boldsymbol{\Gamma}) &= 0, \\ \mathcal{F}_2(\boldsymbol{Y}, \boldsymbol{\Gamma}) &\leq 0. \end{aligned} \tag{14}$$

where Γ denotes model parameters and physical variables. These equations can involve algebraic manipulations or partial differentials of $\boldsymbol{Y}$ and $\boldsymbol{\Gamma}$. The physics-based loss functions for these

equations can be defined as:

$$\mathcal{L}_{\text{phy}}(\hat{\boldsymbol{Y}}) = ||\mathcal{F}_1(\hat{\boldsymbol{Y}}, \boldsymbol{\Gamma})||^2 + \text{ReLU}(\mathcal{F}_2(\hat{\boldsymbol{Y}}, \boldsymbol{\Gamma})), \tag{15}$$

where ReLU represents the rectified linear unit function.

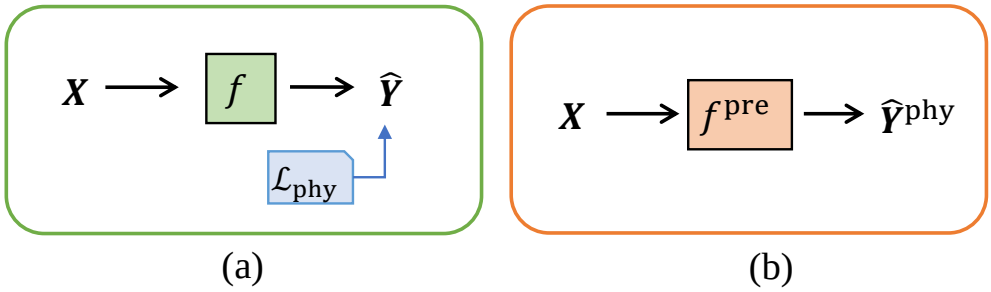
3.1.2. Pre-trained PIML model

The model predictions in AM is dependent on the quality and quantity of the data, especially when there exists an uncertainty in the model or in measurement. The high cost associated with conducting experiments makes it infeasible to have ample amount of training data. Thus, it is important to effectively combine the physics-based model and a small amount of experimental data in order to maximize the accuracy and minimize the uncertainty of the sensitivity estimate. The complex physical knowledge inherent in the physics-based models can be leveraged by generating simulation data for multiple input combinations. This simulation data can be used to pre-train an ML model, that is used as the initial model to be updated with experimental observations. The transfer of physical knowledge using a pre-trained ML model can prevent poor initialization due to lack of knowledge of initial choice of ML model parameters prior to training. In addition, it also allows the ML model to be fine-tuned even with limited experimental data.

The pre-training can use a large amount of training data (with multiple input parameter combinations) over a wide range of values, which is not possible in experiments that could be expensive; as a result, the pre-training may help the eventual ML model to have wider generalization beyond experimental data. In the numerical example in Section 4, the pre-training strategy exercises the physics model over 1310 input combinations. Note that if the physics model is computationally expensive, then the advantage of the pre-training strategy in using a larger input data set (for physics model runs) compared to the experiments becomes limited.

The two proposed strategies to predict the QoI are shown in Fig. 2. Figure 2(a) shows the first method, where the physical knowledge is included through constraints within the loss function of an ML trained with experimental data. Figure 2(b) shows the second method,

where an ML model is first trained with data generated using the physics-based model and then updated using experimental data. The proposed PIML strategies can be applied to physical process by leveraging the related physical constraints or physics-based models.



(a) (b)

Fig. 2. PIML strategies: (a) incorporating physics-based loss functions in the ML models to enforce physics constraints, (b) pre-training an ML model with physics model input-output and updating it with experimental data.

3.2. Implementation of PIML strategies in GP and DNN

Based on the proposed two strategies to incorporate physics knowledge into the ML model, four separate ML models can be constructed for both GP and DNN:

1. GP
2. $\text{GP}^{\mathcal{L}_{\text{phy}}}$
3. GP^{upd}
4. $\text{GP}^{\text{upd},\mathcal{L}_{\text{phy}}}$
5. DNN
6. $\text{DNN}^{\mathcal{L}_{\text{phy}}}$
7. DNN^{upd}
8. $\text{DNN}^{\text{upd},\mathcal{L}_{\text{phy}}}$

The following subsections describe the implementation of these models in detail.

3.2.1. Implementation of PIML in GP

In the first model, GP, experimental observations are used for training. The difference between the true response of the system $\mathbf{Y}_{\text{true}}$ and observations $\mathbf{Y}$ is attributed to observation error ϵ_{obs}, which is often treated as a zero-mean Gaussian random variable with variance σ^2_{obs}.

Model 2 $\text{GP}^{\mathcal{L}_{\text{phy}}}$ incorporates the first strategy by enforcing physics constraints within the optimization of GP hyperparameters. More specifically, the physics constraints are incorporated into the maximization of the log marginal likelihood function (Eq. (8)) while inferring the

hyperparameters of the GP model. Thus, the training of the second model is achieved by maximizing the following function:

$$\mathcal{L}_{\text{GP}} = \log\, p(\mathbf{Y}|\mathbf{X};\boldsymbol{\Theta}) + \lambda_{\text{phy}}\mathcal{L}_{\text{phy}}(\hat{\boldsymbol{Y}}), \tag{16}$$

Consider a physics model f^{phy} that maps input variables $\mathbf{X}$ and model parameters $\boldsymbol{\theta_m}$ to the numerical model output $\mathbf{Y}_m$:

$$\mathbf{Y}_m = \mathbf{G}\big(\mathbf{X};\, \boldsymbol{\theta}_m(\mathbf{X})\big). \tag{17}$$

Let n_D be the number of collected observation data $\mathbf{Y}$ from experiments with input variables $\mathbf{x}^{(1)}, ..., \mathbf{x}^{(n_D)}$, where $\mathbf{x}^{(i)}$ are the input variables for the ith experiment.

There exists uncertainty in the model predictions due to missing physics or approximations as discussed in Section 1. Thus, a model discrepancy term $\boldsymbol{\delta}(\mathbf{X})$ as a function of model inputs is introduced to capture the missing information between $\mathbf{Y}_{\text{true}}$ and $\mathbf{Y}_m$ [22]. Thus, the true system response can be described as

$$\mathbf{Y}_{\text{true}}(\mathbf{X}) = \mathbf{Y}(\mathbf{X}) + \epsilon_{\text{obs}}(\mathbf{X}) = \mathbf{Y}_m(\mathbf{X}) + \boldsymbol{\delta}(\mathbf{X}) = \mathbf{G}\big(\mathbf{X};\, \boldsymbol{\theta}_m(\mathbf{X})\big) + \boldsymbol{\delta}(\mathbf{X}). \tag{18}$$

When the physics model $f^{\text{phy}}(\mathbf{X}) = \mathbf{Y}_m$ is computationally expensive, it is often replaced by a cheaper surrogate model. Here a GP model $\text{GP}^{\text{phy}}(\mathbf{X}) = \hat{\mathbf{Y}}_m$ is used to approximate the physics-based simulation model.

The model discrepancy can be evaluated for different input values of experimental tests and realizations of observation errors as follows:

$$\boldsymbol{\delta}(\mathbf{X}) = \mathbf{Y}(\mathbf{X}) - \hat{\mathbf{Y}}_m(\mathbf{X}) - \epsilon_{\text{obs}}(\mathbf{X}). \tag{19}$$

In model 3, GP^{upd}, where the second PIML strategy is pursued, two GP models are trained; (i) first GP model ($\text{GP}^{\text{phy}}(\mathbf{X})$) replaces the physics model to predict the QoI; and (ii) the second

GP model ($\mathrm{GP}^{\mathrm{exp}}(\mathbf{X})$) is constructed for the remaining discrepancy (i.e., the difference between the model prediction and experimentally observed data) using experimental observations.

The GP model for the remaining model discrepancy ($\mathrm{GP}^{\mathrm{exp}}(\mathbf{X}) = \hat{\boldsymbol{\delta}}$) captures the combined contribution of model form and measurement error for a given prediction. Thus, the predictions of the first GP model (pre-trained) are corrected/updated with the second GP model representing the model discrepancy term and can be written as

$$\hat{\mathbf{Y}}_{corr}(\mathbf{x}) = \hat{\mathbf{Y}}_{m}(\mathbf{X}) + \hat{\boldsymbol{\delta}}. \tag{20}$$

Model 4 is a combination of the two strategies, in which both the pre-trained model is corrected with $\hat{\boldsymbol{\delta}}$ and physics constraints are enforced.

3.2.2. Implementation of PIML in DNN

In model 5, a DNN is trained using only experimental data. Model 6 implements the first strategy: physical knowledge related to the physical process is enforced through constraints within the loss function of the DNN, $\mathrm{DNN}^{\mathcal{L}_{\mathrm{phy}}}$. Model 7 pursues the second strategy: where a DNN model is pre-trained using the coupled multi-physics model input-output $\mathrm{DNN}^{\mathrm{phy}}$ and then updated with experimental data. Model 8 is a combination of the two strategies for DNN.

3.3. Sobol indices computation with model uncertainty

As discussed earlier, often in physics-based and data-driven models it is necessary to quantify the model uncertainty and its contribution to the Sobol index estimates. In this section, we present the inclusion of model uncertainty in the Sobol index estimates.

The model uncertainty pertaining to the GP model is propagated to the Sobol index estimations using the following estimator [7]:

$$S_{m,n}^{X_{d_1}} = \frac{V_{m,n}^{X_{d_1}}}{V_{m,n}} = \frac{\frac{1}{m}\sum_k^m g_n(X_k) g_n(\overline{X}_k) - \frac{1}{m}\sum_k^m g_n(X_k) \sum_k^m g_n(\overline{X}_k)}{\frac{1}{m}\sum_k^m g_n(X_k)^2 - (\frac{1}{m}\sum_k^m g_n(X_k))^2}, \tag{21}$$

where g is the Gaussian process.

A similar approach is implemented to the DNN models with the use of MC dropout.

4. Numerical illustration

A commercial material Ultimaker Black Acrylonitrile butadiene styrene (ABS) was extruded from an Ultimaker 2 Extended+ 3D printer to manufacture fused filament fabrication (FFF) parts with unidirectionally aligned filaments and then measured with appropriate monitoring techniques. FFF is a widely used additive manufacturing (AM) process due to its easy operation, low cost and suitability for complex geometries. As the molten filament is deposited layer upon layer through a nozzle, it cools down, solidifies and bonds with the surrounding filaments. Rectangular ABS amorphous polymer specimens of length 35 mm, width 12 mm, and thickness 4.2 mm are manufactured with constant filament height, width and length (0.7, 0.8, and 35 mm, respectively).

The porosity of an FFF part is dependent on the temperature history at the interfaces between filaments. Thus, it is important to predict the temperature evolution of filaments for estimating the final mesostructure of the printed part. The analytical solution proposed by Costa et al. [23] for transient heat transfer during the printing process in FFF is used to predict the temperature evolution of filaments. A physics-based sintering model is developed, which considers realistic filament geometry, and allows the filament geometry to change during the printing process, to predict the porosity of FFF parts using the temperature evolution of filaments, material properties, part geometry, and process parameters as inputs.

The statistical properties of QoI do not vary along the length of the specimens; thus the porosity measurements were taken at the midpoint of each part with the use of microscopy images processed through the ImageJ software [24]. Filaments were extruded through a nozzle with 0.8 mm diameter. The build plate temperature was constant and set to 110°C. Using Latin hypercube sampling, 39 sets of process parameters $\boldsymbol{X}$ are generated. The ranges considered for the variables are printer extrusion temperature T: ($210°C$ - $260°C$), and extrusion speed S: (15 mm/s - 46 mm/s).

In this work, we impose two different physics-based loss functions (i.e., two separate physical relationships, $\mathcal{L}_{\text{phy},k}(\hat{\boldsymbol{Y}})$, where $k = \{1, 2\}$ and $\hat{\boldsymbol{Y}}$ is the porosity predictions of AM parts). The physical inconsistencies in the model predictions are evaluated using the physics-based loss

functions defined as follows:

$$\mathcal{L}_{\text{phy},1}(\hat{\boldsymbol{Y}}) = \frac{1}{N}\sum_{i=1}^{N} \text{ReLU}(-\hat{Y}_{\text{i}}),$$
$$\mathcal{L}_{\text{phy},2}(\hat{\boldsymbol{Y}}) = \frac{1}{N}\sum_{i=1}^{N} \text{ReLU}(\hat{Y}_{\text{i}} - \phi_{0,\text{i}}), \tag{22}$$

where these loss functions consider the physical violations related to the porosity across N samples. In the first loss functions, negative values of porosity are treated as physical violations. The second loss function penalizes the model when porosity predictions $\hat{Y}_i$ are greater than the initial porosity $\phi_{0,i}$ of ith part. This is based on the physics knowledge that the total void area decreases as the bond formation takes place. Thus, the porosity predictions are ensured to be physically meaningful with the inclusion of these physics-based penalty functions.

For the second PIML strategy, the ML models are pre-trained using the multi-physics model input-output. The pre-trained ML models are then updated using limited experimental observations (i.e., porosity), that helps to learn a 3D printer-specific physical process much faster and with less samples.

The pre-trained models are first trained with coupled multi-physics model input-output data consisting of 1310 input parameter combinations over a range of experimental values, i.e., ($210°C \leq T \leq 260°C$, 15 mm/s $\leq S \leq$ 46 mm/s). (Note that there are only 39 physical experiments are available). The input data are normalized prior to the training of the ML models (the output quantity porosity is dimensionless and between 0 and 1),

The time it takes for training and computation of Sobol indices using models 2-4 are significantly greater than model 1. First-order and total-effect Sobol indices computations based on 5000 samples takes 20-35 minutes due to the pre-training phase, where a large amount of physics input-output samples are used to train the GP models and inclusion physics constraints. Whereas, the computation time for training of each DNN model is on average 15 sec using a desktop computer (Intel® Xeon® CPU E5-1660 v4@3.20GHz with 32 GB RAM and GPU NVIDIA Quadro K620 with 2 GB) and the Sobol index estimations based on 5000 samples take 5 minutes for models 5-8.

4.1. GSA using GP models

The Sobol index computations are based on 5000 MC samples and 100 realizations of the Gaussian process. The effect of number of observations in the first-order Sobol index estimates of temperature for the first four models are illustrated in Fig. 3. The mean values are denoted with solid dots at given number of observations n. The 2.5% and 95.5% prediction intervals are represented with bars above and below the solid dots for the corresponding model.

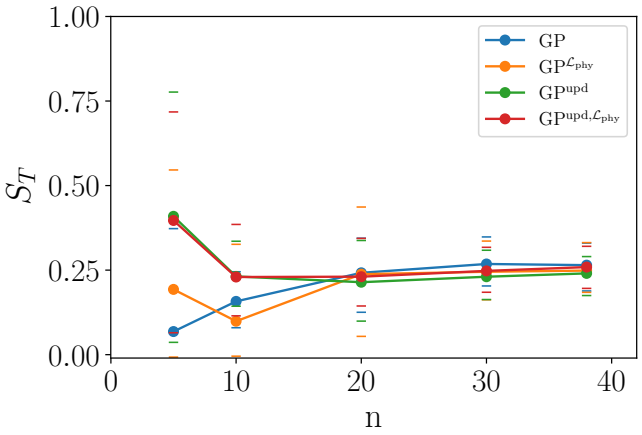


Fig. 3. First-order sensitivity index estimators for temperature using GP.

The effect of number of observations in the first-order Sobol index estimates of temperature for the first four models are illustrated in Fig. 3.

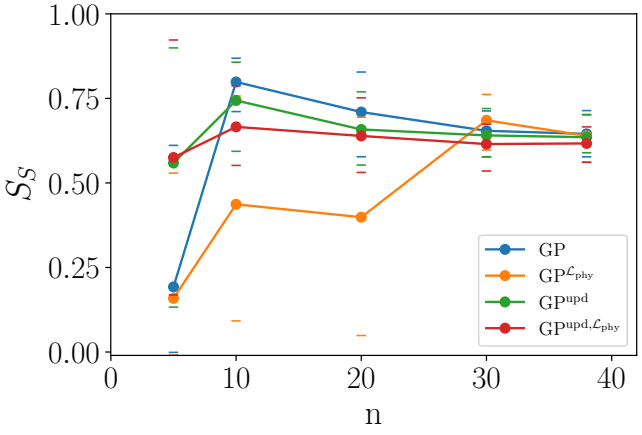


Fig. 4. First-order sensitivity index estimators for speed using GP.

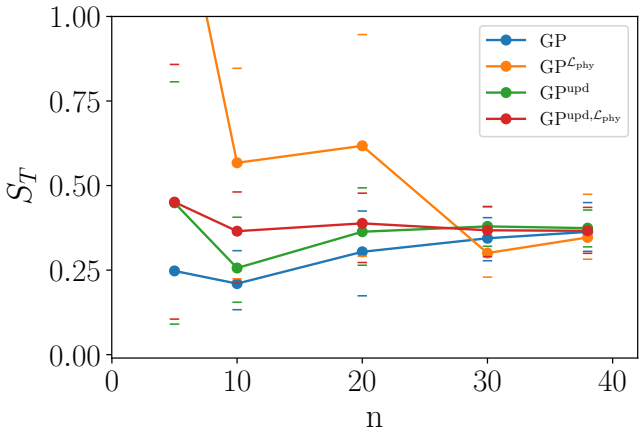


Fig. 5. Total effects index estimators for temperature using GP.

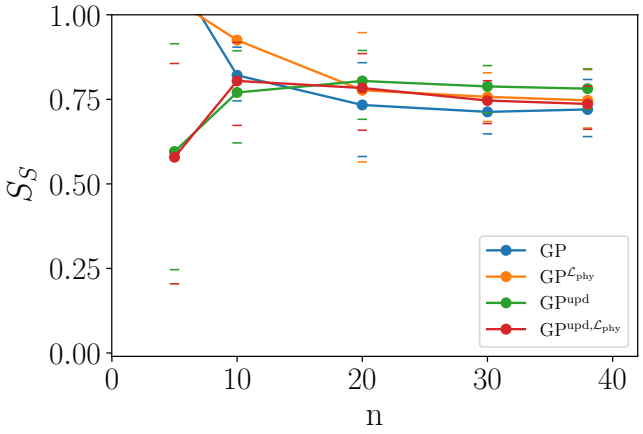


Fig. 6. Total effects index estimators for speed using GP.

4.2. GSA using DNN models

The four DNN models were implemented using the Keras package [25] with Tensorflow in the backend. The hyperparameters of these models are tuned with grid search ($\lambda_{\text{phy}} = 0.15, 0.15$). Fully-connected DNN models with 2 hidden layers and 5 neurons in each hidden layer are constructed. The Rectified Linear Unit (ReLU) activation function and Adam optimizer are used to perform stochastic gradient descent in learning the model parameters.

4.2.1. GSA using DNN models without MC dropout

4.2.2. GSA using DNN models with MC dropout

I WILL ADD the effect of dropout rate later.

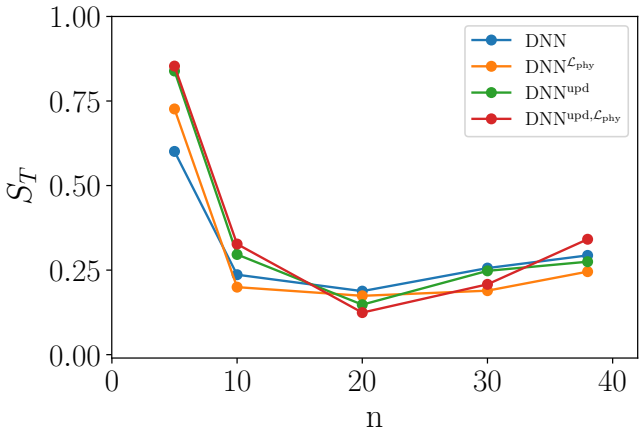


Fig. 7. First-order sensitivity index estimators for temperature using DNN.

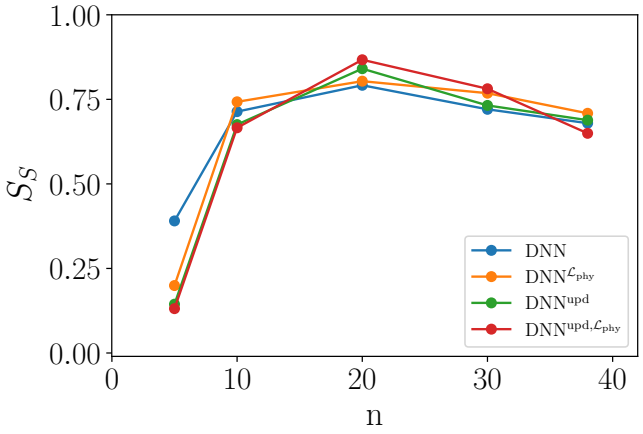


Fig. 8. First-order sensitivity index estimators for speed using DNN.

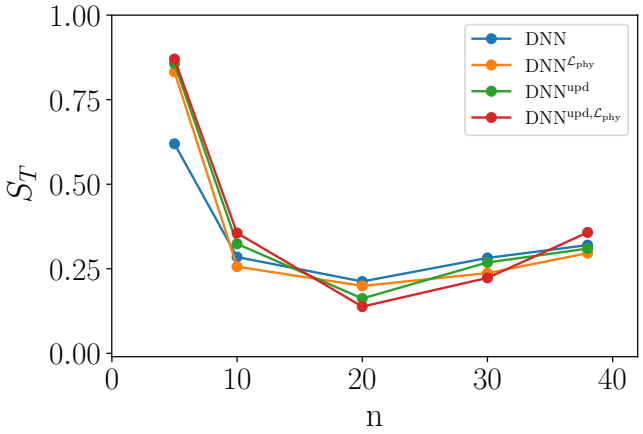


Fig. 9. Total effects index estimators for temperature using DNN.

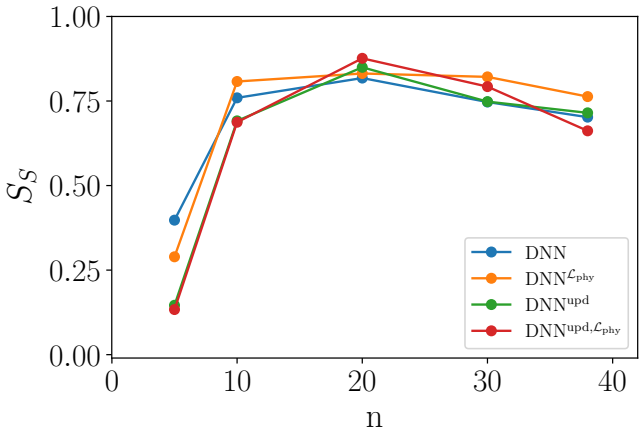


Fig. 10. Total effects index estimators for speed using DNN.

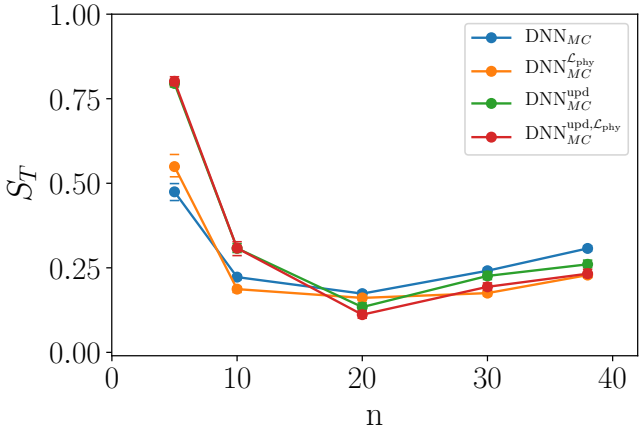


Fig. 11. First-order sensitivity index estimators for temperature using DNN with MC dropout.

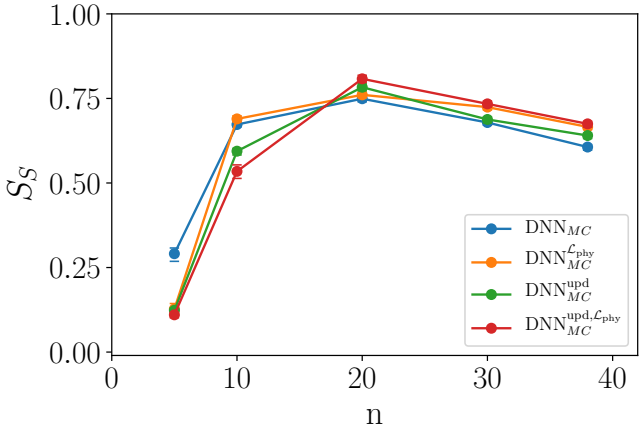


Fig. 12. First-order sensitivity index estimators for speed using DNN with MC dropout.

5. Discussion and conclusions

In this paper, a formulation for information fusion and machine learning for sensitivity analysis using physics knowledge and experimental data while accounting for model uncertainty

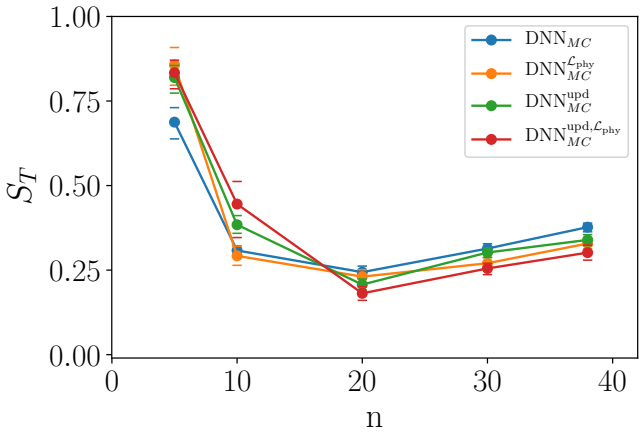


Fig. 13. Total effects index estimators for temperature using DNN with MC dropout.

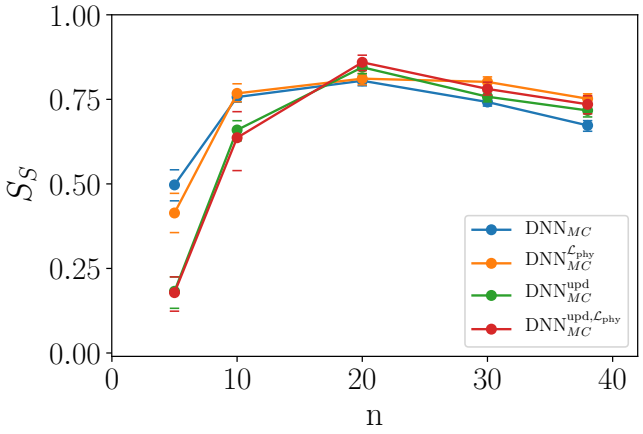


Fig. 14. Total effects index estimators for speed using DNN with MC dropout.

is developed. Variance-based sensitivity analysis is used to quantify the relative contribution of uncertainty source to the variability of the output quantity (porosity). Several PIML models are developed to predict the porosity of FFF parts by leveraging two strategies for incorporating physics knowledge into ML models: (1) incorporating loss functions in the ML models to enforce physics constraint, (2) pre-training an ML model with simulation data and then updating it with experimental data. The physics-based loss functions exploit the physical laws related to the mesostructure of FFF parts.

The results show that the application of PIML strategies to both GP and DNN allows accurate Sobol index computations even with smaller amounts of experimental data while producing physically meaningful results. Thus, the proposed approach helps to fill the physics knowl-

edge gap in the ML models while estimating the Sobol indices accurately by correcting for the approximation in the physics-based models.

In future work, the proposed framework needs to be tested using experimental data that consists of higher dimensional input, with multiple combinations to further analyze the convergence of Sobol index estimations for different PIML strategies. Future work can also explore the weighting of the two sources of data since the data produced by physics-based models and experiments have different levels of credibility.

References

[1] A. Saltelli, M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, S. Tarantola, Global sensitivity analysis: the primer, John Wiley & Sons, 2008.

[2] A. Saltelli, S. Tarantola, K.-S. Chan, A quantitative model-independent method for global sensitivity analysis of model output, Technometrics 41 (1) (1999) 39–56.

[3] T. A. Mara, S. Tarantola, Variance-based sensitivity indices for models with dependent inputs, Reliability Engineering & System Safety 107 (2012) 115–121.

[4] E. Borgonovo, E. Plischke, Sensitivity analysis: a review of recent advances, European Journal of Operational Research 248 (3) (2016) 869–887.

[5] I. M. Sobol, Global sensitivity indices for nonlinear mathematical models and their monte carlo estimates, Mathematics and computers in simulation 55 (1-3) (2001) 271–280.

[6] S. Sankararaman, K. McLemore, S. Mahadevan, S. C. Bradford, L. D. Peterson, Test resource allocation in hierarchical systems using bayesian networks, AIAA journal 51 (3) (2013) 537–550.

[7] L. Le Gratiet, C. Cannamela, B. Iooss, A bayesian approach for global sensitivity analysis of (multifidelity) computer codes, SIAM/ASA Journal on Uncertainty Quantification 2 (1) (2014) 336–363.

[8] J. E. Oakley, A. O'Hagan, Probabilistic sensitivity analysis of complex models: a bayesian approach, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 66 (3) (2004) 751–769.

[9] A. Marrel, B. Iooss, B. Laurent, O. Roustant, Calculations of sobol indices for the gaussian process metamodel, Reliability Engineering & System Safety 94 (3) (2009) 742–751.

[10] Z. Hu, S. Mahadevan, Probability models for data-driven global sensitivity analysis, Reliability Engineering & System Safety 187 (2019) 40–57.

[11] E. C. DeCarlo, S. Mahadevan, B. P. Smarslok, Efficient global sensitivity analysis with correlated variables, Structural and Multidisciplinary Optimization 58 (6) (2018) 2325–2340.

[12] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: international conference on machine learning, 2016, pp. 1050–1059.

[13] A. Karpatne, W. Watkins, J. Read, V. Kumar, Physics-guided neural networks (pgnn): An application in lake temperature modeling, arXiv preprint arXiv:1710.11431 (2017).

[14] J. S. Denker, Y. LeCun, Transforming neural-net output levels to probability distributions, in: Advances in neural information processing systems, 1991, pp. 853–859.

[15] D. J. MacKay, A practical bayesian framework for backpropagation networks, Neural computation 4 (3) (1992) 448–472.

[16] R. M. Neal, Bayesian learning for neural networks, Vol. 118, Springer Science & Business Media, 2012.

[17] C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight uncertainty in neural networks, arXiv preprint arXiv:1505.05424 (2015).

[18] A. Graves, Practical variational inference for neural networks, in: Advances in neural information processing systems, 2011, pp. 2348–2356.

[19] J. M. Hernández-Lobato, Y. Li, M. Rowland, D. Hernández-Lobato, T. Bui, R. Turner, Black-box α-divergence minimization (2016).

[20] Y. Gal, Z. Ghahramani, Bayesian convolutional neural networks with bernoulli approximate variational inference, arXiv preprint arXiv:1506.02158 (2015).

[21] J. Willard, X. Jia, S. Xu, M. Steinbach, V. Kumar, Integrating physics-based modeling with machine learning: A survey, arXiv preprint arXiv:2003.04919 (2020).

[22] M. C. Kennedy, A. O'Hagan, Bayesian calibration of computer models, Journal of the Royal Statistical Society: Series B (Statistical Methodology) 63 (3) (2001) 425–464.

[23] S. Costa, F. Duarte, J. Covas, Estimation of filament temperature and adhesion development in fused deposition techniques, Journal of Materials Processing Technology 245 (2017) 167–179.

[24] C. A. Schneider, W. S. Rasband, K. W. Eliceiri, Nih image to imagej: 25 years of image analysis, Nature methods 9 (7) (2012) 671.

[25] F. Chollet, et al., keras. github repository, https://github. com/fchollet/keras¿. Accessed on 25 (2015) 2017.