Applied Intelligence (2024) 54:14–34
https://doi.org/10.1007/s10489-023-05178-z

Real-time prediction of gas flow dynamics in diesel engines using a deep neural operator framework

Varun Kumar[1] · Somdatta Goswami[2] · Daniel Smith[3] · George Em Karniadakis[1,2]

Accepted: 12 November 2023 / Published online: 5 December 2023

Abstract
The objective of this work is to address the need for fast and accurate models for analyzing transient gas flow dynamics in diesel engines. We employ a neural operator-based surrogate model, called DeepONet, to learn and predict the transient gas flow dynamics in real-time. The neural operator maps the relationship between engine control stimulus, namely engine speed, fuel injection per cycle, EGR, and VGT valve openings with seven output states that include intake and exhaust manifold pressures, residual gas fraction inside the cylinder, as well as the dynamics of the EGR and VGT actuators. To establish a benchmark, we compare results from the DeepONet model to a mean-value gas flow engine model simulated with Simulink. We observe a maximum relative $\mathcal{L}_2$ error of 6.5%, a reasonable accuracy for transient dynamics. The DeepONet model also exhibits good robustness to noisy input functions. Additionally, to evaluate the epistemic uncertainty in our model predictions, we adopt a mean ensemble approach, yielding a worst-case error of 12% at a standard deviation of 2σ from the mean value. In summary, our proposed framework offers real-time prediction capabilities and facilitates data-driven learning of complex input-output operator mappings. This makes the DeepONet surrogate particularly useful for preliminary analyses of system dynamics, control system optimization, and health monitoring of sub-systems.

Keywords Diesel engine · Neural networks · Gas flow dynamics · Operator network · Epistemic uncertainty

1 Introduction

Diesel engines are widely used in heavy-duty applications due to their higher peak torque and thermal efficiency, although they face stringent emissions regulations due to the release of hazardous nitrogen oxides (NO_x) and particulates. Consequently, trade-offs between optimum operating conand engine emissions are required to ensure compliance with regulatory norms. Model-based engineering is viewed as the preferred approach for engine development, given the increasing sophistication of modern engines. Engine manufacturers are constantly striving to improve such models to enable the analysis and optimization of critical performance indicators. Research on mathematical modeling of internal combustion engines dates back to the 1970s [1–5] with multifaceted applications in control system design, performance prediction, and emission optimization. Model reduction is crucial for real-time applications like control system optimization and system monitoring, as it enhances computational efficiency. The complex, non-linear relationships existing within an engine are challenging to model using traditional physics-based methods. Alternatively, artificial neural networks (ANNs) provide a good alternative and have garnered significant attention in the modeling community.

Thompson et al. [6] in 2000 presented an ANN architecture based on a partially recurrent network to predict continuous engine torque and exhaust emission for a heavy-duty diesel engine. Their model demonstrated good capability in predicting the output states in the FTP cycle. Later in 2008, Campa et al. [7] proposed a Radial Basis Function (RBF) network-based scheme to diagnose sensor failures, inspired by the work of Park and Sandberg [8]. They used pre-trained RBF networks to identify deviations in the output from the baseline prediction state to determine a faulty

✉ George Em Karniadakis
george_karniadakis@brown.edu

Varun Kumar
varun_kumar2@brown.edu

1 School of Engineering, Brown University, Providence, RI, USA

2 Division of Applied Mathematics, Brown University, Providence, RI, USA

3 Cummins Inc., Columbus, IN, USA

condition. Wang et al. [9] combined a non-linear dynamic model with a multi-layer perceptron network (MLP) to predict the NO_x emission with EGR%, combustion temperature, engine speed, and coolant temperature as inputs. This model performed well under steady-state conditions but required additional correction under transient conditions for accurate predictions. Ismail et al. [10] used a shallow network to model the relationship between bio-diesel blend and its response to emissions from a light-duty engine.

In a major shift from the previous efforts, Shamekhi et al. [11] created a holistic neural network surrogate for a mean value spark ignition (SI) engine model. They employ a Bayesian regulation MLP network to model the input-output relationship for separate sub-networks and then compare these predicted outputs with a calibrated GT-power model for the respective sub-system. Li et al. [12] used an MLP network to model emission from a light-duty diesel engine for control design. They use throttle position, EGR valve position, VGT valve position, and fueling to model the NO_x emission, but found that their model over-predicts sharp transient response. Luján et al. [13] proposed an adaptive learning algorithm based on MLP to model the volumetric efficiency of an automotive diesel engine as a function of manifold temperatures and pressures. Although this approach yielded an accurate prediction, the training dataset was limited to steady-state conditions and the results may not extend to real driving patterns. Taglialatela et al. [14] used a single layer fully connected network to predict NO_x particle size using engine speed, load, and EGR% as inputs. They found a good correlation between predicted particle size and measurements, however, the model was tested under steady-state conditions only.

Fravolini et al. [15] compare the performance of MLP and hybrid RBF (ADALINE + EMRAN) networks used in offline and online training modes respectively for prediction of differential pressure sensor signal from the exhaust after-treatment system. A hybrid RBF network was found to perform better than offline MLP models with the validation dataset. Domínguez et al. [16] used MLP and symbolic regression to map the operating conditions of a diesel engine fueled with animal fat with exhaust emissions. They found similar performance with both methods except for NO_x prediction where symbolic regression was slightly better. Zhao et al. [17] utilized the RBF network to improve the response of the PID speed controller for a diesel engine generator. They also compared the performance of single-layer perceptron networks with RBF and found that RBF adapts better to sudden load fluctuations under random weight initialization. Shin et al. [18] compared the performance of the MLP network with LSTM for predicting NO_x emission from diesel engines. They conclude that although the LSTM model has higher accuracy as compared to the MLP network, higher computational time is required for generating LSTM prediction and hence, this model is less useful for real-time predictions. González et al. [19], in an interesting work, try to integrate physics and data-driven models for predicting gas exchange processes in a diesel engine. They model each sub-system of a diesel engine using a combination of physics models and data-driven MLPs, then combine these sub-modules in concert to determine other gas flow states required.

2 Research motivation and our contributions

The review of the current research landscape related to the integration of ANNs in diesel engine modeling reveals some key research opportunities:

- Development of fast and accurate prediction of engine dynamics is required for control system modeling, fault diagnosis, and emission control. In particular, integrating ANNs for predicting gas flow dynamics is a critical area of research for engine control and system health monitoring. Integrating ANN-based surrogates with existing physics-based models [20] can help increase the accuracy and efficiency of such models for real-time applications, especially for complete engine system models.
- Analyzing uncertainty and understanding how ANNs respond to measurement noise are important research opportunities in this field. The ANN models need to generate real-time, accurate predictions using noisy, sensor measurement data.
- Current ANN techniques create discrete functional mappings between inputs and outputs. To integrate ANNs with physics-based models, often represented as ordinary differential equations (ODEs) and partial differential equations (PDEs), a continuous-time output is advantageous for leveraging fast and accurate auto-differentiation algorithms provided by machine learning packages.
- Generalizable models under various operating and transient conditions is crucial for real-world deployment Existing ANN models work well for test data generated under similar operating conditions but fail to generalize to other conditions. Hence, a generalized framework that can be adapted to different scenarios would be useful in the adoption of ANNs in engine systems.

Analytical models have been widely used for simulating the behavior of combustion engines, and several commercial packages such as AVL Boost [21], GT Power [22], Ricardo Wave [23] are used for engine development. The mean-value models for simulating diesel engine gas flow, proposed in [24, 25], are based on the manifold filling and emptying concept. These models allow simulating the engine behavior by mak-

ing approximations around engine transient behaviors and have been used extensively in control design and fault diagnosis [26–29]. However, considering the fast transients that are encountered during an engine's operation, real-time prediction capability is necessary for the deployment of engine models in the field which may not always be possible with models based on numerical solvers. Alternatively, data and physics-driven surrogate modeling have shown significant success in their ability to simulate the behavior of complex systems [30, 31]. These models rely on using empirical data for generating representations of real system behavior. Additionally, such surrogate models offer fast predictions, which is essential for field deployment. Nevertheless, applying deep learning models to handle real data that is often accompanied by noise, presents some challenges in modeling internal combustion engines with ANNs.

The objective of the current work, therefore, is to develop a robust and efficient surrogate model to simulate diesel engine operations. Specifically, we employ a deep operator-based network-based model (referred to as DeepONet herein) to predict the gas flow dynamics of a diesel engine. We choose DeepONet as our ANN surrogate over existing neural network-based models (RBF, LSTM, and MLP) that have been traditionally used in engine modeling since an operator-based framework enables mapping of one/multiple input functions to a continuous time domain. This continuous mapping is essential for linking our data-driven DeepONet approach to physics-based models containing a system of ODEs in the future. We present a concise example in the future work section to expand on our motivation for using the operator-based learning paradigm. Enlisted below are our main contributions to this work:

- Demonstrated an exemplar DeepONet architecture for learning the functional mapping between engine control stimulus signals and gas flow dynamics in diesel engines. The model learns this mapping by using engine speed, fueling, EGR valve position, and VGT valve position data as inputs to the DeepONet model. The trained model can then predict seven dynamic gas flow states (intake manifold pressure P_{im}, exhaust manifold pressure P_{em}, residual gas fraction x_r, temperature after inlet valve closes at intake completion T_1, turbo-shaft speed ω_t, EGR actuator signal $\tilde{u}_{egr}$, and VGT actuator signal $\tilde{u}_{vgt}$).
- Used the DeepONet architecture with the ability to generate real-time, multi-state gas flow predictions in *less than a second*. Once trained, the minimal time required for generating predictions makes the DeepONet model a suitable candidate for real-time applications such as engine control optimization and health monitoring.
- Evaluated the robustness of the proposed DeepONet model with noisy inputs and estimated prediction errors under noisy conditions. The DeepONet surrogate model shows a slight increase in prediction error with increasing noise levels.
- Determined epistemic uncertainty through the use of dropout in the network architecture. We quantify the maximum uncertainty that exists in the DeepONet model predictions through the use of an ensemble-based approach.

The remainder of the manuscript is arranged as follows. In Section 3, the Simulink model used for generating the ground truth for training the DeepONet-based surrogate model is presented. Minor modifications made to the Simulink model to meet the objectives of the current work are also discussed. In Section 4, we present a brief overview of the DeepONet architecture and showcase the developed surrogate model for modeling the diesel engine. We discuss the experimental setup containing data generation and pre-processing in Section 5. Section 6 presents the results from our experiments including a sequence-to-sequence prediction model. Section 7 presents the results for epistemic uncertainty estimation using dropouts in the branch network. Lastly, in Sections 8 and 9, we report the limitations of our current methodology and the summary of our findings, respectively.

3 Numerical simulation of diesel engine

The rapid exchange of air and exhaust gases alongside energy inside a diesel engine presents a challenge in creating representative models for analysis. As a practical alternative, 1-D Mean-value models based on the emptying and filling of manifold volumes have been proposed for simplicity ([24, 25]). The engine model by Wahlström and Ericsson [20] is one such simplified model that is based on the dynamics of the gas flow inside the manifolds, EGR valve, the turbocharger, we use this model for generating the simulated data used in this study. In this section, we briefly discuss the inputs and outputs associated with this model for clarity. Figure 1 presents a schematic of the various components associated with Wahlström's model. The inputs and outputs of this model are signals recorded using engine state sensors over a period of time. The input signals of this model can be defined by the input vector:

$$\text{inputs} : [n_e,\ u_\delta,\ u_{egr},\ u_{vgt}],$$

where n_e represents the engine speed, u_δ the fuel injected into the cylinders per cycle, u_{egr} and u_{vgt} represent EGR and VGT valve openings that are empirically determined during engine calibration. The output states emanating from this model are defined by the output vector:

$$\text{output states} : [P_{im},\ P_{em},\ X_{O_{im}},\ X_{O_{em}},\ \omega_t,\ \tilde{u}_{egr1},\ \tilde{u}_{egr2},\ \tilde{u}_{vgt}],$$

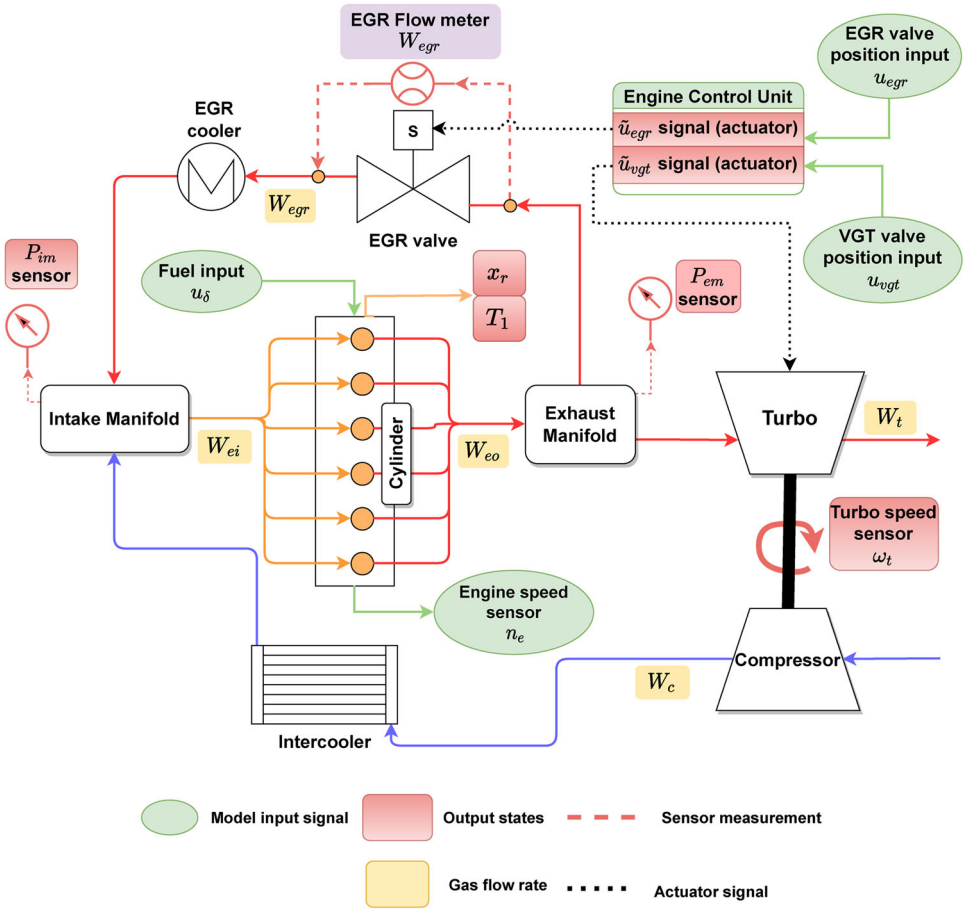

Fig. 1 Schematic of various subsystems in the mean-value diesel engine model (modified from [20]). In this schematic, n_e, u_δ, u_{egr} and u_{vgt} represent the engine speed, the fuel injected into the cylinders per cycle, valve position signals received by the EGR valve and the turbocharger valve actuators, respectively. W_{ei} represents gas flow rate into the cylinders, W_{eo} represents exhaust gas flow rate into the exhaust manifold, W_{egr} represents EGR gas flow rate, and W_c represents the fresh air flow rate into the intake manifold from the compressor

where P_{im} is the input manifold pressure, P_{em} represents the exhaust manifold pressure, $X_{O_{im}}$ represents oxygen mass fraction in intake manifold, $X_{O_{em}}$ the oxygen mass fraction in exhaust manifold, $\tilde{u}_{egr1,2}$ for EGR actuator dynamics and $\tilde{u}_{vgt}$ represents VGT valve actuator dynamics. The eight output states are obtained by solving their respective ordinary differential equations (ODEs) by using conventional numerical solvers in Simulink. Output data generated from the Simulink model is used as the ground truth in this work. Interested readers can refer to [20] for details on the relations of the inputs and the output states, and additional parameters of the Simulink model [32].

To achieve the goals of our current work discussed in Section 2, we modified the original Simulink model to extract the desired output states. These changes were required to make this work compatible with the parameter identification task in the future (see [54] and Section 10 for details). Additional blocks were added to the Simulink model for extracting the necessary output states. The new output states extracted from the modified model are represented as

$$\text{new output states} : [P_{im},\ P_{em},\ x_r,\ T_1,\ \omega_t,\ \tilde{u}_{egr},\ \tilde{u}_{vgt}],$$

where x_r represents the residual gas fraction, T_1 is the temperature once inlet valve is closed after intake stroke, and $\tilde{u}_{egr}$ represents the combined EGR valve actuator output obtained through a combination of $\tilde{u}_{egr1}$ and $\tilde{u}_{egr2}$ (see equation 39 in [20]). The input data is collected from a 6-cylinder heavy-duty truck engine on a test bed. The parameters required for generating the output data from the Simulink model are adopted from [20]. A sampling frequency of 2 Hz is used for generating the ground truth solution used for evaluating the accuracy of our surrogate model.

The Wahlström and Ericsson (WE) engine model employs a set of parameters derived from a least square fit of measured values from the engine laboratory. Hence, in order to adopt this model for different engines, these parameters must be determined. For instance, the Simulink model for the EGR valve employs coefficients c_{egr1}, c_{egr2}, c_{egr3}, and Π_{egropt}, which are determined through empirical curve fitting. Therefore, to simulate and employ the WE model for other applications, additional effort is required to determine the parameters based on the analytical relations. Computing these parameters requires one to solve inverse problems within an optimization loop that increases latency in predictions. Surrogate models serve as cost-effective approxima-

tions for high-fidelity simulations, allowing for significant computational savings while maintaining solution accuracy. Deep neural operators (DeepONet), introduced in 2019 [33], have been employed effectively as surrogate models for complex physical problems like fracture mechanics [34], bubble dynamics [35], and electro-convection [36] to name a few. One significant advantage of an operator-based model is its ability to learn nonlinear functional mappings between inputs and outputs based on data. Additionally, the prediction time for a pre-trained DeepONet is a fraction of a second, which is a critical requirement for real-time forecasting in field applications. The flexibility offered by the DeepONet allows generalization to different operating conditions such as different ambient temperatures and pressures, which is another advantage over the traditional solvers. In the next section, we outline details of the proposed operator regression model developed for the mapping of the engine control stimulus to multi-state gas flow dynamics.

4 Operator regression for diesel engine modeling

Adeep operator network allows learning a non-linear operator from data and is suited for applications where the physics-based models are difficult to ascertain or when generalized results are desired. Diesel engines, with their high-frequency dynamic processes, are a suitable candidate for using an operator-based neural network. The idea of DeepONet is motivated by the universal approximation theorem for operators [37], which states that a neural network with a single hidden layer can approximate accurately any linear/non-linear continuous function or operator. Before we begin learning the solution operators of the parametric differential equations, we must first distinguish between a function regression and an operator regression. The solution in the function regression approach is parameterized as a neural network between finite-dimensional Euclidean spaces: $\mathcal{F} : \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_1}$, where d_1 is the number of discretization points. In operator regression, however, a function is mapped to another function using an operator. With this idea in mind, we put forward the conventional architecture of DeepONet in the first part of this section and later introduce the proposed architecture of DeepONet specific to this work.

4.1 Deep operator network (DeepONet)

DeepONet consists of two deep neural networks (DNN): $\mathcal{N}_1$ (referred to as the branch net) encodes the input function, $\boldsymbol{u}$, at m fixed locations, typically known as sensors. $\mathcal{N}_2$ (referred to as the trunk net) takes as input the evaluation location, $\boldsymbol{y}$. In a general context, the branch network input can represent various aspects, such as the physical domain, initial or boundary conditions, constant or variable coefficients, and source terms, as long as the input function is discretized at these m sensor locations. For regularly spaced discretization of the input function, a convolutional neural network (CNN) can serve as the branch net, while for sparse representations, one can consider a feed-forward neural network (FNN) or even a recurrent neural network (RNN) for sequential data. In our work, we employ FNNs to represent both the trunk and branch networks.

To recognize the theoretical underpinning of a DNN, we consider a neural network with L hidden layers, with the 0-th layer denoting the input layer and the $(L+1)$-th layer denoting the output layer; the weighted input z_i^l into a i^{th} neuron on a layer l is a function of weight $\boldsymbol{W}_{ij}^l$ and bias $\boldsymbol{b}_j^{l-1}$ and is represented as

$$z_i^l = \mathcal{R}_{l-1}\left(\sum_{j=1}^{m_{l-1}}\left(\boldsymbol{W}_{ij}^l(z_j^{l-1}) + \boldsymbol{b}_j^l\right)\right), \tag{1}$$

where m_{l-1} is the number of neurons in layer $l-1$ and $\mathcal{R}_{l-1}(\cdot)$ represents the activation function of layer l. The feed-forward procedure for calculating the output $\boldsymbol{Y}_L$ is expressed as follows:

$$\begin{aligned} \boldsymbol{Y}^L &= \mathcal{R}_L(\boldsymbol{W}^{L+1}\boldsymbol{z}^L + \boldsymbol{b}^L) \\ \boldsymbol{z}^L &= \mathcal{R}_{L-1}\left(\boldsymbol{W}^L\boldsymbol{z}^{L-1} + \boldsymbol{b}^L\right) \\ \boldsymbol{z}^{L-1} &= \mathcal{R}_{L-2}\left(\boldsymbol{W}^{L-1}\boldsymbol{z}^{L-2} + \boldsymbol{b}^{L-1}\right) \\ &\vdots \\ \boldsymbol{z}^1 &= \mathcal{R}_0\left(\boldsymbol{W}^1\boldsymbol{x} + \boldsymbol{b}^1\right), \end{aligned} \tag{2}$$

where $\boldsymbol{x}$ is the input of the neural network. Equation (2) can be encoded in compressed form as $\boldsymbol{Y} = \mathcal{N}(x;\boldsymbol{\theta})$, where $\boldsymbol{\theta} = (\boldsymbol{W}, \boldsymbol{b})$ includes both the weights and biases of the neural network $\mathcal{N}$. Taking into account a DeepONet, $\mathcal{N}_1$ takes as input the function to denote the input realizations $\boldsymbol{U} = \{\boldsymbol{u}_1, \boldsymbol{u}_2, \ldots, \boldsymbol{u}_N\}$ for N samples, discretized at n_{sen} sensor locations such that $\boldsymbol{u}_i = \{u_i(\boldsymbol{x}_1), u_i(\boldsymbol{x}_2), \ldots, u_i(\boldsymbol{x}_{n_{sen}})\}$ and $i \in [1, N]$, while $\mathcal{N}_2$ inputs the location $\boldsymbol{y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_p\} = \{(\hat{x}_1, \hat{y}_1), (\hat{x}_2, \hat{y}_2), \ldots, (\hat{x}_p, \hat{y}_p)\}$ to evaluate the solution operator, where $\hat{x}_i$ and $\hat{y}_i$ denote the coordinates x and y of the point $\boldsymbol{y}_i$, respectively. Let us consider that the branch neural network consists of l_{br} hidden layers, where the $(l_{br}+1)^{\text{th}}$ layer is the output layer consisting of q neurons. Considering an input function $\boldsymbol{u}_i$ in the branch network, the network returns a feature embedded in $[b_1, b_2, \ldots, b_q]^{\text{T}}$ as output. The output $\boldsymbol{z}_{br}^{l_{br}+1}$ of the feed-

forward branch neural network is expressed as

$$\begin{aligned} z_{br}^{l_{br}+1} &= \left[b_1, b_2, \ldots, b_q\right]^{\mathrm{T}} \\ &= \mathcal{R}_{br}\left(\boldsymbol{W}^{l_{br}} \boldsymbol{z}^{l_{br}} + \boldsymbol{b}^{l_{br}+1}\right), \end{aligned} \tag{3}$$

where $\mathcal{R}_{br}(\cdot)$ denotes the nonlinear activation function for the branch net and $\boldsymbol{z}^{l_{br}} = f_{br}(u_i(\boldsymbol{x}_1), u_i(\boldsymbol{x}_2), \ldots, u_i(\boldsymbol{x}_m))$, where $f_{br}(\cdot)$ denotes a branch net function. Similarly, consider a trunk network with l_{tr} hidden layers, where the $(l_{tr}+1)$-th layer is the output layer consisting of q neurons. The trunk net outputs a feature embedding $[t_1, t_2, \ldots, t_q]^{\mathrm{T}}$. The output of the trunk network can be represented as

$$\begin{aligned} z_{tr}^{l_{tr}+1} &= \left[t_1, t_2, \ldots, t_q\right]^{\mathrm{T}} \\ &= \mathcal{R}_{tr}\left(\boldsymbol{W}^{l_{tr}} \boldsymbol{z}^{l_{tr}} + \boldsymbol{b}^{l_{tr}+1}\right), \end{aligned} \tag{4}$$

where $\mathcal{R}_{tr}(\cdot)$ denotes the non-linear activation function for the trunk net and $\boldsymbol{z}^{l_{tr}-1} = f_{tr}(\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_p)$. The key point is that we uncover a new operator $\mathcal{G}_{\boldsymbol{\theta}}$ as a neural network that can infer quantities of interest from unseen and noisy inputs. The two networks are trained to learn the solution operator such that

$$\mathcal{G}_{\boldsymbol{\theta}} : \boldsymbol{u}_i \rightarrow \mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{u}_i), \quad \forall \ i = \{1, 2, 3, \ldots, N\}. \tag{5}$$

For a single input function $\boldsymbol{u}_i$, the DeepONet prediction $\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{u})$ evaluated at any coordinate $\boldsymbol{y}$ can be expressed as

$$\begin{aligned} \mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{u}_i)(\boldsymbol{y}) &= \sum_{k=1}^{q} \left(\mathcal{R}_{br}(\boldsymbol{W}_k^{l_{br}} \boldsymbol{z}_k^{l_{br}-1} + \boldsymbol{b}_k^{l_{br}}) \cdot \mathcal{R}_{tr}(\boldsymbol{W}_k^{l_{tr}} \boldsymbol{z}_k^{l_{tr}-1} + \boldsymbol{b}_k^{l_{tr}})\right) \\ &= \sum_{k=1}^{q} b_k(u_i(\boldsymbol{x}_1), u_i(\boldsymbol{x}_2), \ldots, u_i(\boldsymbol{x}_m)) \cdot t_k(\boldsymbol{y}). \end{aligned} \tag{6}$$

DeepONet requires large annotated datasets of paired input-output observations, but it provides a simple and intuitive model architecture that is fast to train, allowing a continuous representation of the target output functions that are resolution-independent. Conventionally, the trainable parameters of the DeepONet represented by $\boldsymbol{\theta}$ in (6) are obtained by minimizing a loss function. Common loss functions used in the literature include the $\mathcal{L}_1$ and $\mathcal{L}_2$ loss functions, defined as

$$\begin{aligned} \mathcal{L}_1 &= \sum_{i=1}^{n} \sum_{j=1}^{p} \left|\mathcal{G}(\boldsymbol{u}_i)(\boldsymbol{y}_j) - \mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{u}_i)(\boldsymbol{y}_j)\right| \\ \mathcal{L}_2 &= \sum_{i=1}^{n} \sum_{j=1}^{p} \left(\mathcal{G}(\boldsymbol{u}_i(\boldsymbol{y}_j) - \mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{u}_i)(\boldsymbol{y}_j)\right)^2, \end{aligned} \tag{7}$$

where $\mathcal{G}_{\boldsymbol{\theta}}(\boldsymbol{u}_i)(\boldsymbol{y}_j)$ is the predicted value obtained from the DeepONet, and $\mathcal{G}(\boldsymbol{u}_i)(\boldsymbol{y}_j)$ is the target value.

Next, we present a DeepONet algorithm for the diesel engine where we compute the weights and biases associated with the deep neural networks based on the available labeled datasets.

4.2 DeepONet framework for diesel engine

In this work, we employ a multiple-input operator architecture of DeepONet, as discussed in previous works [38, 39], to construct a surrogate model for the diesel engine. Figure 2 illustrates the architecture of the operator network used to approximate the diesel engine's output states. To prepare the training data, we transform the temporal signals of the four inputs into feature-based representations, each containing ten-time points per signal, enhancing the network's learning process. Each of the four input signals is associated with a dedicated branch network (Branch1 $-$ 4 in Fig. 2), functioning as a function approximator. In addition, we provide another set of inputs to DeepONet, represented as initial conditions extracted from the output predictions in Branch5. This concept is akin to providing initial conditions for solving differential equations and aids in constraining the operator learning process (Fig. 3) to learn a unique solution. The initial condition values correspond to output states that can be measured in the field, specifically P_{im}, P_{em}, ω_t, $\tilde{u}_{egr}$, and $\tilde{u}_{vgt}$ (as shown in Fig. 1). The element-wise product of the output embeddings of branch networks $1-5$ and trunk networks $1-5$ maps to the five output states: P_{im}, P_{em}, x_r, T_1, and ω_t.

Furthermore, the functional representation for the inputs u_{egr} and u_{vgt} is generated separately for branch networks 6 and 7. These, coupled with branch networks 8 and 9, contain information about the initial conditions of $\tilde{u}_{egr}$ and $\tilde{u}_{vgt}$, respectively. The corresponding output of trunk networks 6 and 7 approximates the output states $\tilde{u}_{egr}$ and $\tilde{u}_{vgt}$. Separate branches are used for generating states $\tilde{u}_{egr}$ and $\tilde{u}_{vgt}$ since only u_{egr} and u_{vgt} are associated with them. Detailed equations can be found in [20], which explain the associativity between output states and their affecting inputs.

This proposed design can be adjusted to incorporate additional inputs, given the flexibility of the DeepONet architecture. If the network needs to include a new input and/or output state, the design can be changed to accommodate an extra branch and trunk. The mapping of appropriate inputs to output states is based on the associativity of relevant equations from [20], which assists the network learning process.

5 Experiment design

In this section, details of the experiment setup including data generation, data pre-processing, details of network architecture, and hyperparameter choices are discussed.

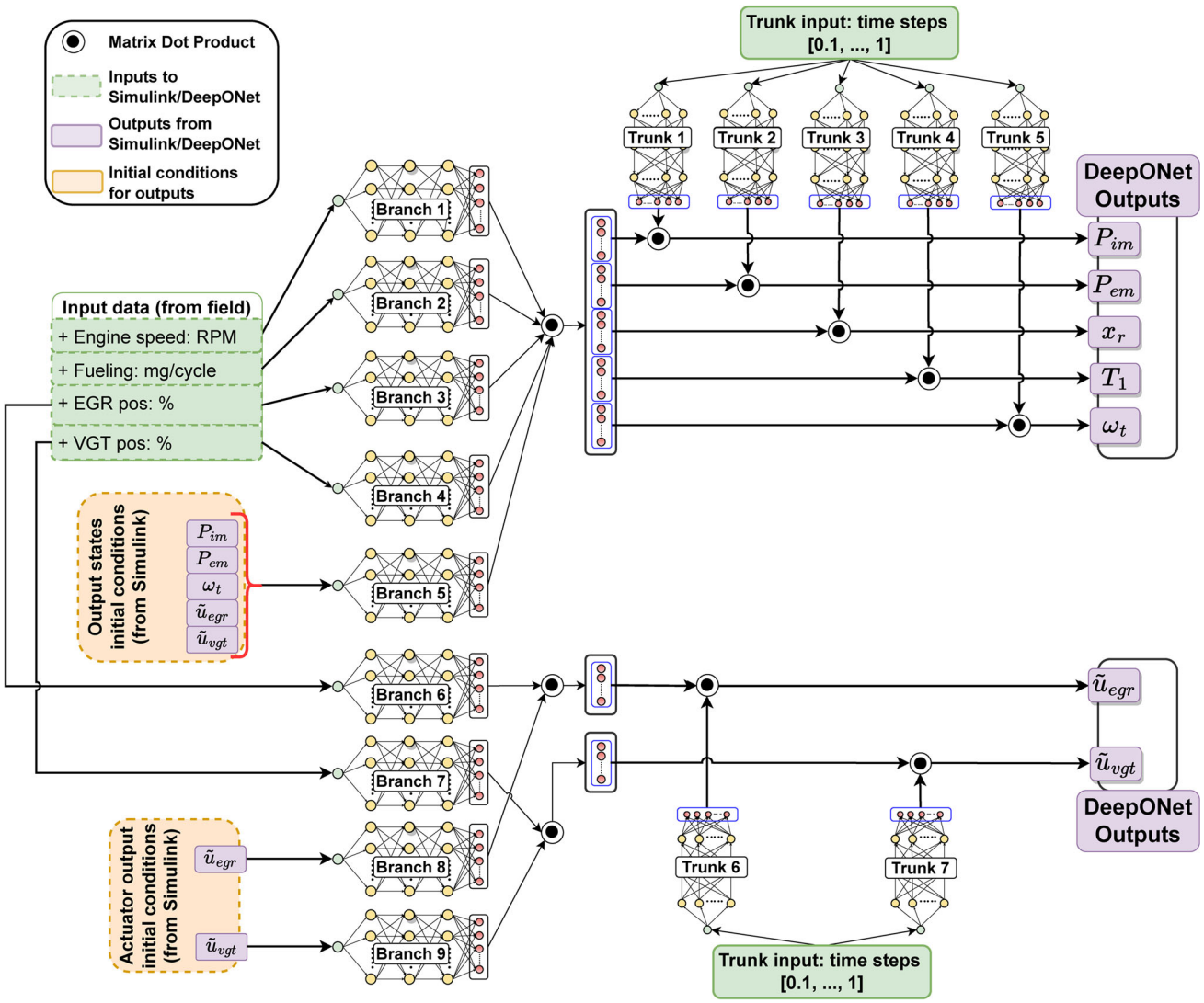

Fig. 2 Multi-input DeepONet architecture to approximate the output states of the Diesel engine. The inputs to the model are encapsulated in green boxes, while the outputs are in purple-colored boxes. The architecture employs nine branch networks for generating the functional mapping whereas the seven trunk networks are used for determining the basis functions (at each temporal point) for each output state. Separate branches are used for $\tilde{u}_{egr}$ and $\tilde{u}_{vgt}$ due to their strict dependence on u_{egr} and u_{vgt}. Initial conditions for the output states are provided as additional inputs to assist in solution convergence

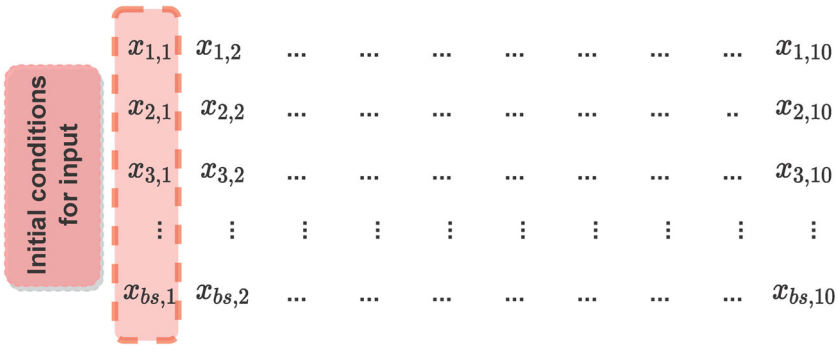

Fig. 3 Extracting initial conditions for use as input to DeepONet from output data. These initial conditions are only provided for measurable outputs P_{im}, P_{em}, ω_t, $\tilde{u}_{egr}$, and $\tilde{u}_{vgt}$

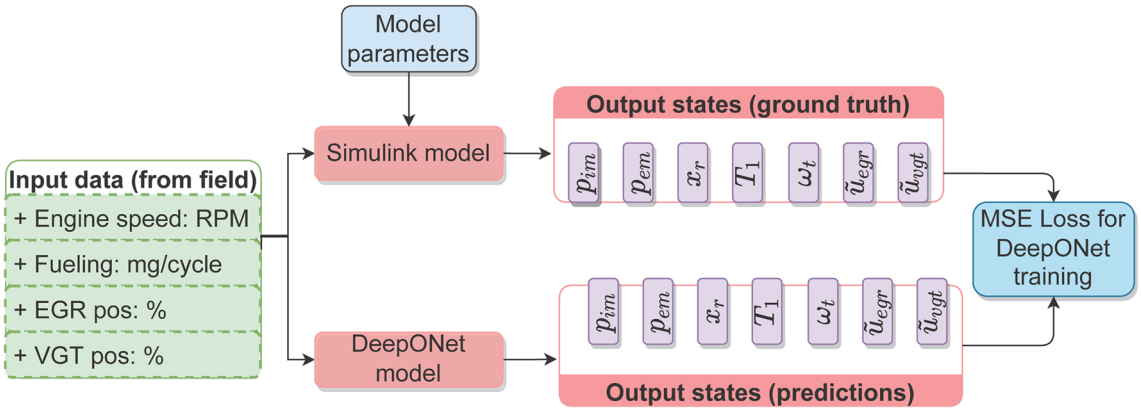

Fig. 4 Overview of DeepONet model and its use in conjunction with Simulink model as ground truth generator

5.1 Data generation

To generate training and testing data, we utilize the mean-value Simulink model discussed in Section 3 to simulate the gas flow dynamics with control stimulus collected from an engine test bed. The proposed scheme for training the surrogate model using data generated from Simulink is illustrated in Fig. 4. Approximately fifteen hours of sensory data collected over a period of time with the same engine were used. For data partitioning, we divide the input and ground truth datasets into training and testing sets, selecting a continuous section of 1000 seconds for testing ensuring its representation in the training set. Scenarios involving engine idling with insufficient representation in the training dataset were excluded from the test dataset. It is important to encompass all possible engine conditions during data generation to enable the DeepONet network to make generalized predictions. The Simulink model is simulated using parameter values obtained from [20].

5.2 Data pre-processing

To enhance learning, the input data is transformed into signal trains with a window size of 10 as shown in Fig. 3 (corresponding to a 2 Hz data collection frequency from sensors). All inputs and outputs are converted into signal trains to improve the learning efficiency of the network. Furthermore, the input and the output signals are normalized using their respective minimum and maximum values of the training dataset.

$$y_{norm} = \frac{y_{reg} - y_{min}}{y_{max} - y_{min}} \tag{8}$$

5.3 Network design

The architecture details of the surrogate DeepONet model are shown in Table 1. For weight optimization, an Adam optimizer with a learning rate scheduler is used. The starting learning rate used was $1e^{-3}$ until the first 5, 000 epochs, which was then reduced to $5e^{-4}$ until 10,000 epochs. Thereafter, a constant learning rate of $1e^{-4}$ was used until the training terminates at 20, 000 epochs. Dropouts were also incorporated as a network regularizer in the branch networks that generate the functional representation of the inputs. Dropout rates were tuned heuristically, and lower dropout rates were found to work better for actuator sub-networks. In addition to regularizing the network, dropout can also be used for model uncertainty estimation [40], and we provide uncertainty estimation results in the following sections. The model was trained on an NVIDIA A40 GPU and the training time was approximately three hours. Point-wise self-adaptive weights were also used to regularize each output temporal point during training [41, 42] to capture the fast dynamics of the system. The mean-squared loss was used as the cost function for network training while the $\mathcal{L}_2$ relative norm was used for evaluating the prediction error on the testing dataset.

Table 1 Details of DeepONet architecture used for diesel engine modeling

Network	Depth	Width	Activation function		Dropout Rate
			Hidden layers	Final layer	
Branch 1 – 2	3	128	`Swish`	`ReLU`	0.20
Branch 3 – 4	3	128	`Swish`	`ReLU`	0.10
Branch 6 – 7	2	128	`Swish`	`Linear`	0.05
Branch 8 – 9	3	128	`Swish`	`Linear`	––
Trunks	3	128	`Swish`	`Sigmoid`	––

6 Experimental results

In this section, we present and discuss the results of our experiments. In addition to experiments with clean input data, we also present results with noisy inputs and outputs to show the sensitivity of DeepONet predictions.

6.1 DeepONet prediction results

Figure 5 presents a comparative analysis between DeepONet predictions and ground truth data for the first 60-second time window in the testing data subset. The complete 1000 sec prediction is provided in Figs. 12 and 13 in Appendix. Table 2 (first row) displays the resulting $\mathcal{L}_2$ error values for the seven

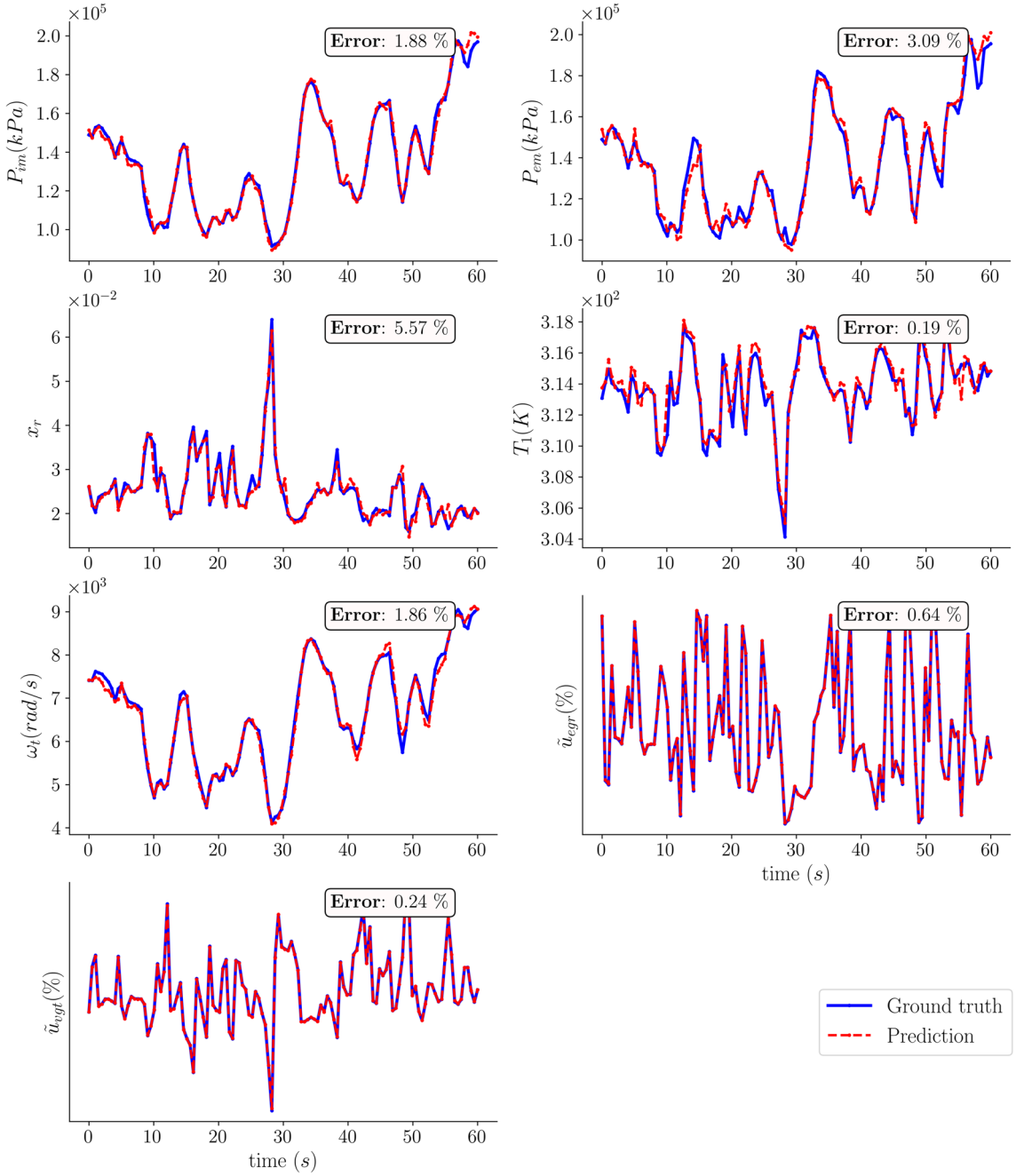

Fig. 5 DeepONet prediction results for the seven output states and comparison with ground truth obtained from Simulink. The time subset is limited to the first 60 seconds here for clarity. Error value labels on top of each plot are values evaluated for this 60 second time window. Y-axis markers for $\tilde{u}_{egr}$ and $\tilde{u}_{vgt}$ are not shown in the interest of confidentiality

Table 2 $\mathcal{L}_2$ error values for the predicted seven output states obtained using the proposed DeepONet-based surrogate model. The error results are over a testing window of 1000 seconds. A comparison of error for no noise and different noise conditions is also shown here

Noise level	$\mathcal{L}_2$ error (%)						
	P_{im}	P_{em}	x_r	T_1	ω_T	$\tilde{u}_{egr}$	$\tilde{u}_{vgt}$
No noise	4.96	6.44	5.22	0.39	3.56	0.59	1.15
1%	5.09	6.64	5.26	0.40	3.66	1.54	1.86
2%	5.15	6.75	5.30	0.41	3.71	2.64	3.03
3%	5.17	6.94	5.56	0.41	3.89	4.14	4.35

output states over the 1000-second prediction time window. The output state P_{em} exhibits the highest error rate of 6.5%. Predictions for actuator signal states $\tilde{u}_{egr}$ and $\tilde{u}_{vgt}$ demonstrate a good match with their ground truth values. This is expected due to their relatively easier-to-map functional relationship. P_{em} represents the pressure in the exhaust manifold and is determined primarily by the combustion process inside the cylinder and after the treatment system. State x_r, which indicates residual gas fraction inside the combustion chamber, is an important parameter in determining the state of exhaust gases, especially its temperature and thereby pressure. In our architecture, the initial conditions for x_r were not used as input since this state cannot be measured directly in the field. The error in x_r estimation may have a strong impact on the error for state P_{em} since the network is trained together, leading to higher prediction error for P_{em}. The higher error for P_{em} could also suggest the need to identify additional input features to augment the operator learning process for this state. Overall, the DeepONet model predictions are within reasonable accuracy considering the dynamic nature of the system. Figure 6 shows the error propagation for the seven states during model training. We note that the rate of error convergence for the seven states varies depending on the complexity of input-output mapping.

6.2 Results with noisy data

Sensor signals in an engine are associated with noise that is difficult to ascertain or determine with certainty. To evaluate the impact of noisy sensor input on model prediction, we test the DeepONet model under three additive white Gaussian noise conditions: 1%, 2%, and 3%. The selected noise levels align with the expected noise from sensors generating the four inputs to DeepONet in a real system. The DeepONet model trained on clean data previously was used for making predictions with noisy inputs. Table 2 shows a comparison of prediction errors between results obtained with clean input against noisy inputs. Among the output states, $\tilde{u}_{egr}$ and $\tilde{u}_{vgt}$ exhibit the highest sensitivity to input noise. The effect of noise addition on other output states remains minimal. Figure 7 shows a comparison between ground truth and DeepONet prediction with a 3% noise addition to input data. The surrogate model shows good robustness to input noise due to the use of dropouts during model training which helps improve the generalization capability of the network.

We also evaluated the sensitivity of our model to noise in output data. The objective of this study was to assess the model's accuracy when trained with inherently noisy output data, resembling real-world field measurements. We add

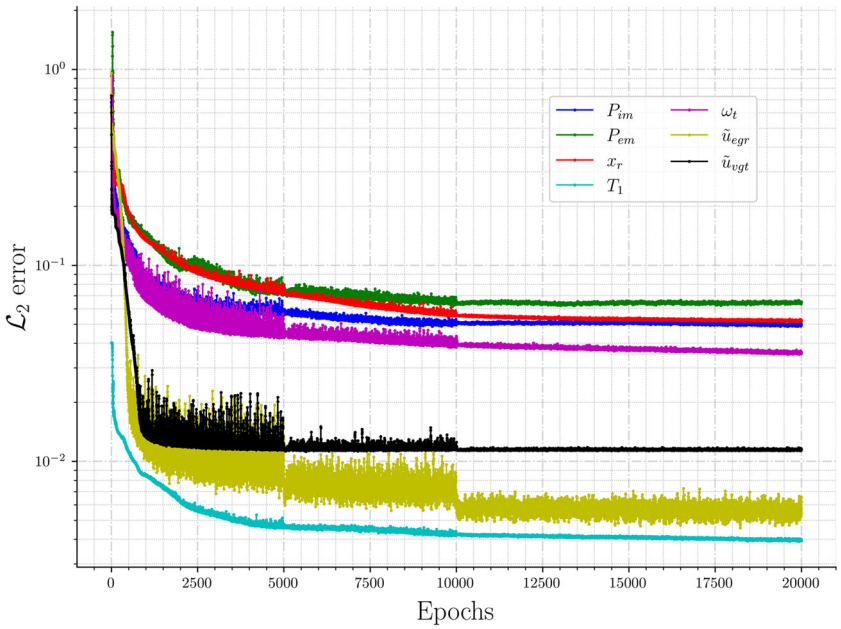

Fig. 6 Error propagation for the seven output states during training. The difference in convergence rate stems from the varying complexities in operator mapping among different output functions

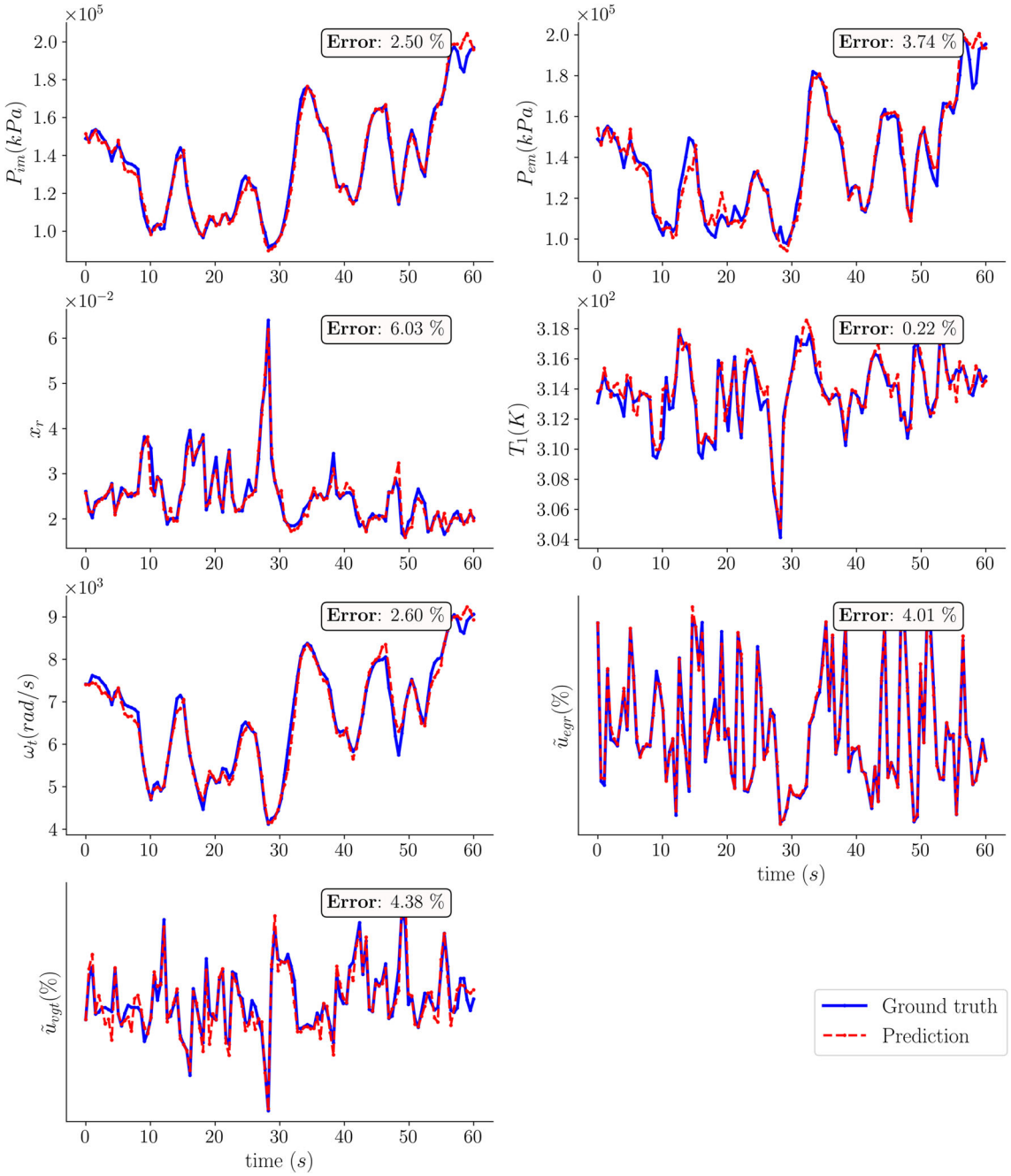

Fig. 7 DeepONet prediction results with 3% Gaussian white noise added to testing data input. Prediction error increases progressively with noise level with the actuator predictions showing the highest increase possibly due to the low dropout rate used for their respective branch networks

Gaussian white noise of magnitude up to 3% to the training labels. The 3% limit on noise level was chosen based on field experience with sensors that measure the output states on commercial diesel engines. The DeepONet model is trained with this noisy labeled data, and the trained network is used for generating predictions with noise-free data. Table 3 shows the error comparison between the DeepONet model trained with clean vs noisy outputs.

Table 3 $\mathcal{L}_2$ error comparison between model trained with clean vs noisy output training dataset

Noise level	$\mathcal{L}_2$ error (%)						
	P_{im}	P_{em}	x_r	T_1	ω_T	$\tilde{u}_{egr}$	$\tilde{u}_{vgt}$
No noise	4.96	6.44	5.22	0.39	3.56	0.59	1.15
3%	4.97	7.14	5.47	0.85	5.17	0.67	1.17

A marginal increase in relative error is observed for the network trained with added noise to labeled training data. Predictions here were made using a noise-free test dataset

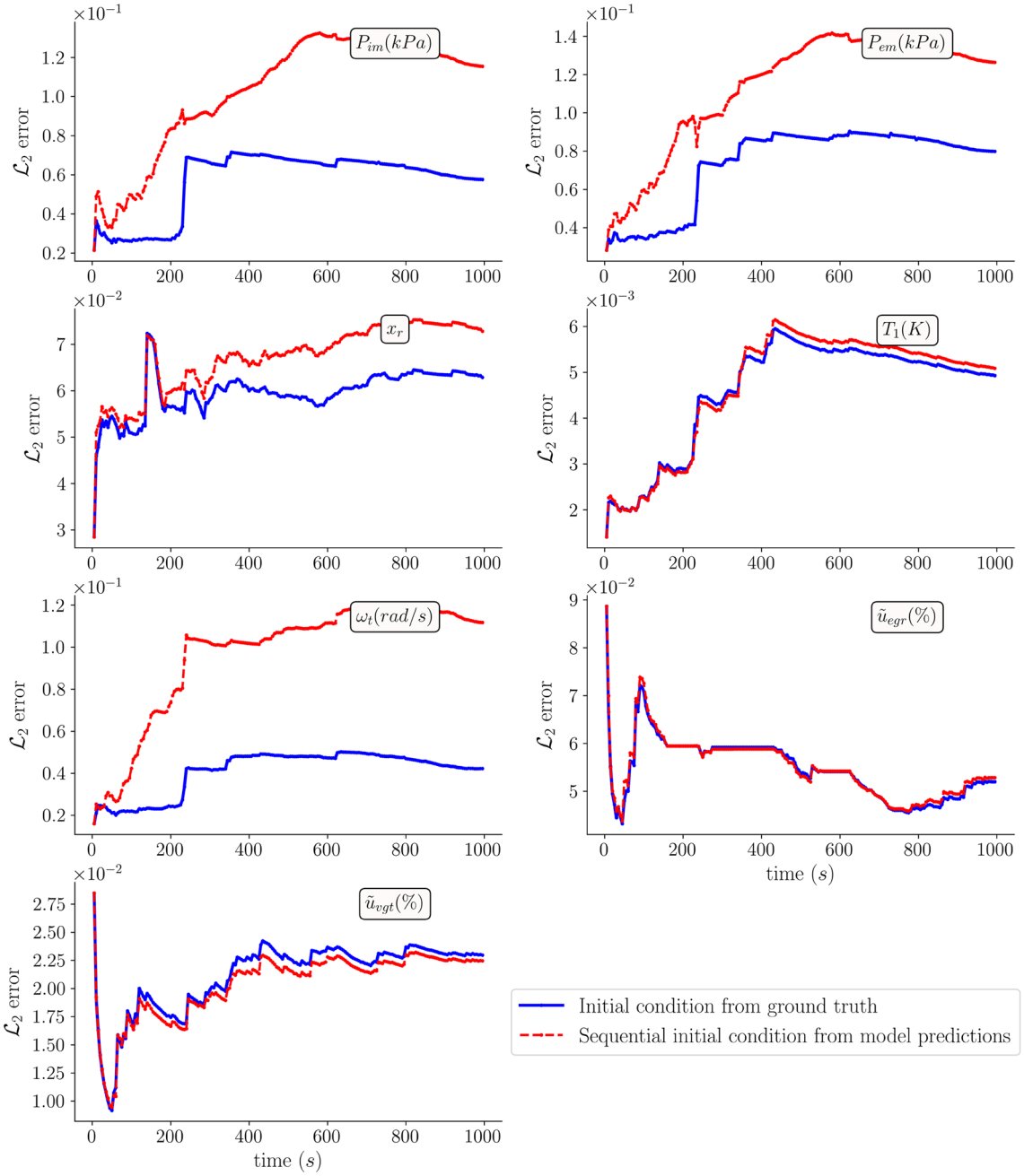

Fig. 8 Comparison of cumulative error between employing sequence-to-sequence scheme for initial condition and using initial conditions for signal trains from the ground truth. A higher error accumulation results when using a sequence-to-sequence scheme. This is attributed to an error in DeepONet outputs which when used as initial conditions for subsequent signals leads to cumulative error buildup. Notably, the cumulative error is most pronounced for states P_{im}, P_{em}, and ω_t possibly due to the higher error values associated with the DeepONet model itself

Table 4 $\mathcal{L}_2$ error comparison between prediction with ground truth as initial condition vs sequence-to-sequence initial condition generation scheme

Initial condition from	$\mathcal{L}_2$ error (%)						
	P_{im}	P_{em}	x_r	T_1	ω_T	$\tilde{u}_{egr}$	$\tilde{u}_{vgt}$
Ground truth	4.96	6.44	5.22	0.39	3.56	0.59	1.15
Seq-to-seq	11.54	12.64	7.28	0.85	11.17	5.29	2.25

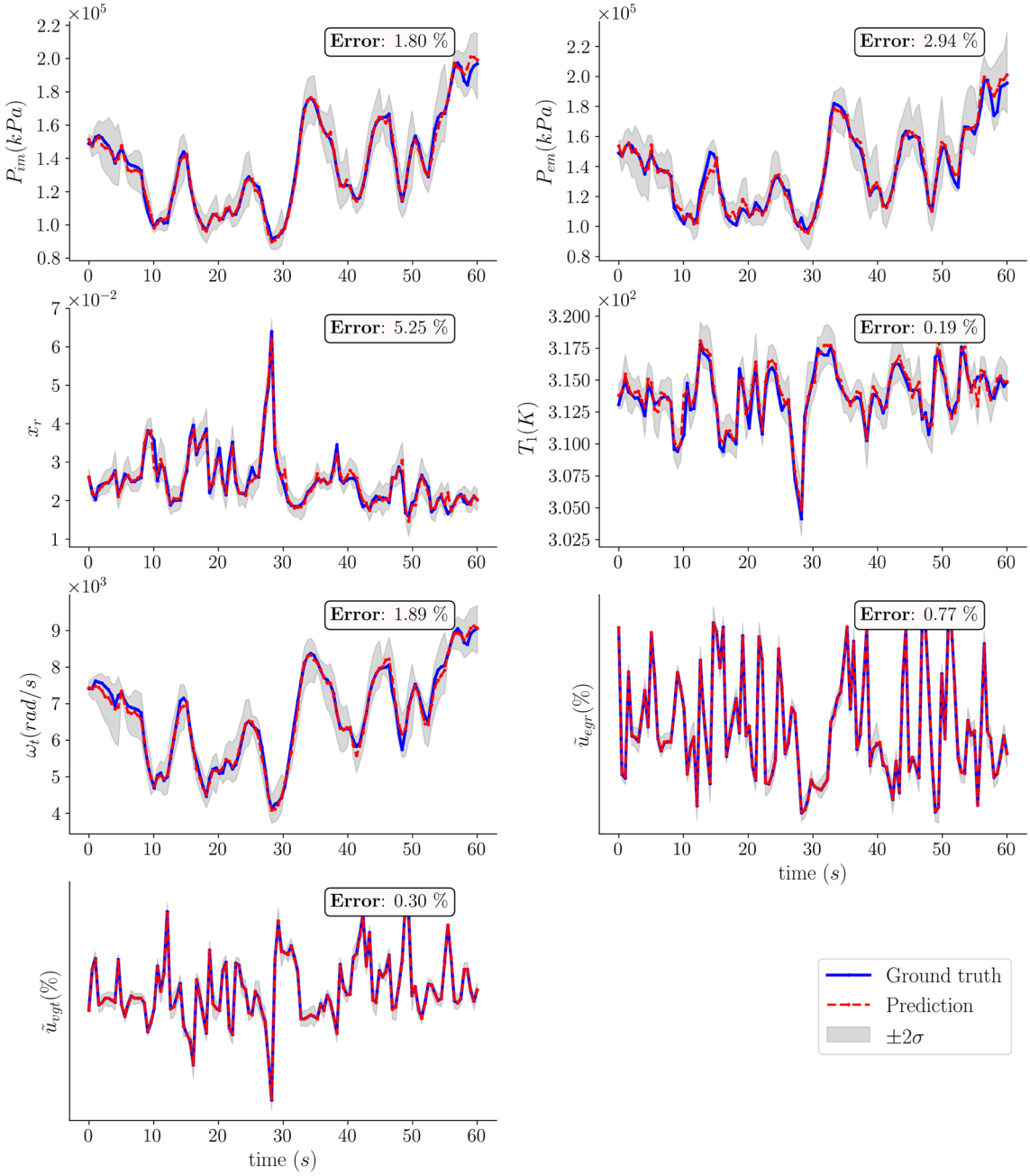

Fig. 9 DeepONet prediction and uncertainty results with an ensemble mean generated by drawing 100 samples during prediction in an MC dropout network with dropout rates as per Table 1. The predicted value is the mean of 100 samples while the uncertainty, represented by the standard deviation for these samples is shown as a $\pm 2\sigma$ zone around the ensembled mean

6.3 Sequence-to-sequence chaining of initial conditions

In addition to using initial conditions from the ground truth output data (generated from Simulink), we also investigate the impact of employing sequence-to-sequence chaining of initial conditions on model accuracy over a 1000 second time sequence. This is relevant in real applications where ground truth data is not accessible or potentially corrupted by systematic faults in the data logging system. To evaluate the sequence-to-sequence approach, we utilize the prediction from DeepONet output from the previous signal train as an initial condition for the next input signal train. This method is followed recursively over the complete test signal, generating one output per iteration. However, this approach leads to longer prediction time ($\approx$ 30 seconds in our case) and higher prediction error accumulation as the prediction sequence gets longer. Figure 8 illustrates the comparison of cumulative error between the sequence-to-sequence scheme and using initial conditions from ground truth data. Table 4 summarizes the comparison between the two initial condition schemes.

7 Model uncertainty estimation

Neural networks, when trained to learn a certain set of weights and biases to predict outputs are deterministic. Epistemic uncertainty pertains to the potential variations that may exist in the model's weight estimates and understanding and quantifying this uncertainty is essential for building confidence around the model's predictions. For uncertainty estimation, we employ probabilistic modeling rooted in the Bayesian framework [43–45], utilizing dropout as our key approach. Traditional approaches for Bayesian modeling include Monte Carlo Dropout [40, 46], HMC [47–49], Bayes-by-backprop [50–52] and others; see [45] for a comprehensive review. In this work, we use regular dropouts to understand model uncertainty due to the following reasons:

- Dropout is used as a regularizer in branch layers to prevent overfitting.
- Dropout provides a faster and easier approach for model uncertainty estimation. Other approaches such as Variational Inference can be challenging to train on complex architectures. Additionally, setting up prior and posterior distribution [53] can present additional complexities with sensory data in the field.

We demonstrate epistemic uncertainty through the following steps:

1. Train the DeepONet model with noise-free input data using MC Dropout in branch networks, with the architecture presented in Table 1.
2. Generate predictions on the test section of the data. Each time a prediction is made, a different prediction of the output states is obtained due to the stochastic nature imparted to branch networks by the dropout layers.
3. Collect 100 outputs and generate an ensemble mean to represent the mean prediction, μ. Calculate the standard deviation, σ for each point for all predictions thus generated.
4. Create an uncertainty region around the ensemble mean with a spread of $\mu \pm 2\sigma$

Figure 9 shows the comparison between the ensemble mean, computed from 100 predictions generated by the Dropout-based DeepONet model. The standard deviation for each point is calculated over these 100 predictions to generate the uncertainty band around this ensemble mean. Table 5 shows the total error for 1000 seconds of test data with an ensemble mean of 100 predictions from a dropout-based stochastic DeepONet. Relative $\mathcal{L}_2$ error % for the seven output states estimated using the ensemble mean exhibits higher errors compared to the deterministic prediction errors. This is due to the stochastic nature of the branch layers utilizing dropout which results in the network learning different mapping each time a prediction is made. To ascertain the maximum prediction error possible with this model, we calculate the worst-case prediction boundary by using the $\pm 2\sigma$ value with the ensemble mean and estimate the error with respect to the target prediction. The worst-case error values are shown in Table 5. Notably, the maximum error possible

Table 5 Error comparison between the deterministic prediction of the output states from DeepONet with an ensemble of predictions generated by using Dropouts in the branch network

Testing condition	$\mathcal{L}_2$ error (%)						
	P_{im}	P_{em}	x_r	T_1	ω_t	$\tilde{u}_{egr}$	$\tilde{u}_{vgt}$
Deterministic prediction	4.96	6.44	5.22	0.39	3.56	0.59	1.15
Ensemble μ	4.87	6.34	5.22	0.39	3.58	0.77	1.15
Ensemble $\mu + 2\sigma$	8.01	12.60	10.90	0.75	6.81	9.06	3.90

Error values for ensemble mean indicate the total error for the 1000 second time sequence that was used for testing when 100 different predictions were ensembled. Error value for ensemble $\mu + 2\sigma$ indicates the worst-case error when the prediction is at the boundary of the uncertainty band

in prediction estimates in this worst-case scenario is approximately 12.6% for state P_{em}.

8 Limitations

In this section, we address certain limitations of the proposed operator network-based surrogate model. A primary challenge is the substantial amount of training data that is required for the operator-based network to effectively learn the operator mapping in functional space. Collecting sufficient data to accurately learn the operator mapping can pose practical difficulties. The error values for the predictions, as discussed in earlier sections (Tables 2 and 5), represent the total errors calculated across the 1000 second testing window. However, these errors are not uniform across this entire time span, leading to varying results for different time segments. Figure 10 illustrates the prediction results for a different 60-second time window, between time $t = 300$ to $t = 360s$. We notice significantly higher error values for the four output states in this data section. This observation may be attributed to input data, which was collected from a test bed under generic test conditions limiting the model's ability to predict specific operating conditions. Ideally, data encompassing representative information of all possible engine operating states should be employed for training and evaluation purposes.

Extending the current DeepONet architecture to other engine models is a possibility, however, this would necessitate training the DeepONet model using data collected from individual engines, which involves significant time and effort. To overcome this challenge, machine learning methods such as transfer learning can be employed to adapt this model to other similar engine designs. It is important to note that the current DeepONet model has been trained using the parameter set provided by [20], and these values are required for data generation through Simulink. Alternatively, the DeepONet model can be used for learning the operator mapping using real engine datasets thereby eliminating the need for the Simulink model. However, such an approach will require additional time and effort for data collection and may be cost-prohibitive.

9 Summary

In this work, we introduce a novel deep-operator-based neural network model for predicting the mean-value gas flow

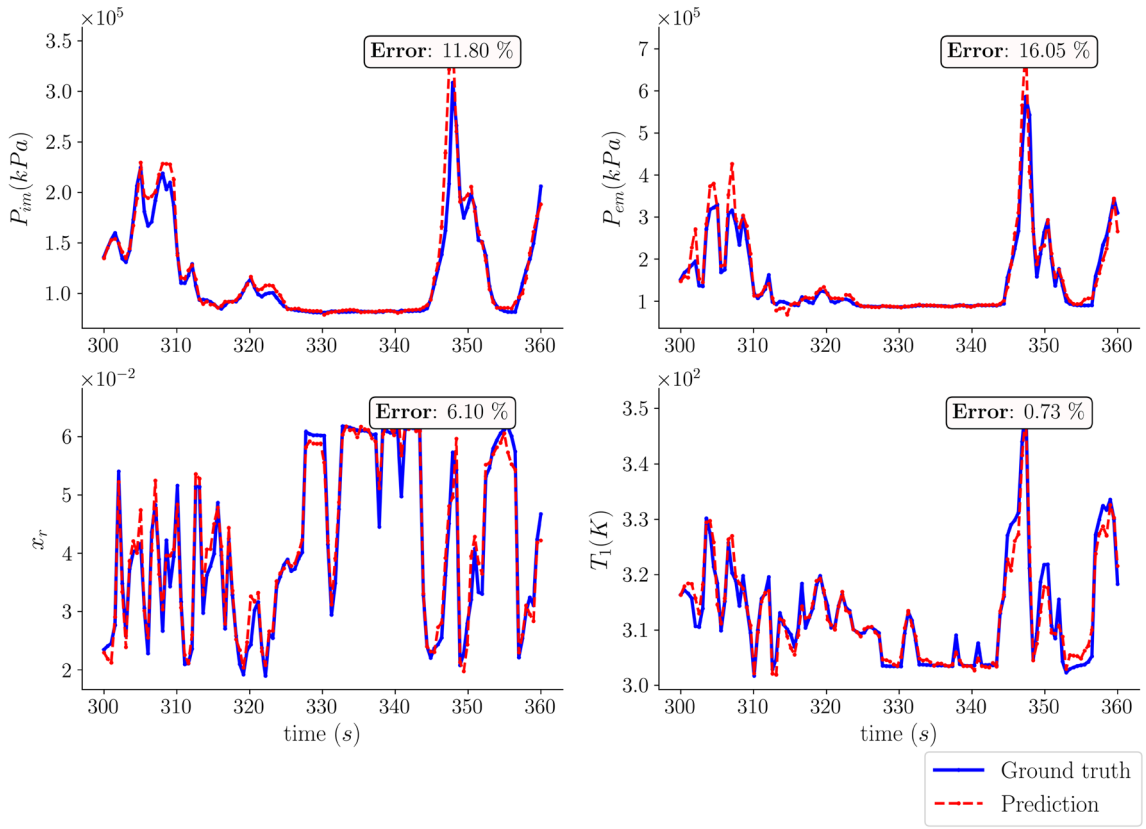

Fig. 10 DeepONet predictions for a different time segment from the testing dataset showing limitations for the current model with the training dataset used in this work. High error % are observed in this case possibly due to lack of representative data points in the training dataset

dynamics for diesel engines. Four input signals, engine speed (n_e), fueling (u_δ), EGR valve position (u_{egr}), and VGT valve position (u_{vgt}) are used for generating a continuous operator mapping to seven output states. To generate the ground truth for the output states, the Simulink model available online from [32] is used. Once trained, the DeepONet model is used for predicting the seven output gas flow states when provided with unseen testing data input. To assess the performance of the DeepONet model, we evaluate its accuracy and robustness using the $\mathcal{L}_2$ relative error metric by analyzing model performance under noise-free and noisy conditions. We summarize our findings as follows:

1. The operator-based neural network model is capable of effectively learning operator mapping that converts engine control stimulus to desired gas dynamics traditionally modeled using a mean-value gas flow model of diesel engines. The output states predicted by DeepONet show good accuracy over a 1000 sec testing window (see Table 2). The maximum relative error observed was 6.4% for output state P_{em}. Notably, P_{em} has a strong dependence on the exhaust manifold dynamics such as exhaust manifold temperature, which suggests the need to include additional relevant input features. Figure 5 also reveals interesting correlations between states ω_t, P_{im}, and P_{em} in their characteristic shapes, which can be attributed to the interconnected nature of turbocharger assembly with intake and exhaust systems.
2. DeepONet exhibits strong generalization accuracy when tested with simulated noisy data in both inputs and outputs. Adding 3% white noise to inputs during testing on a model trained with noise-free input and labeled data results in a marginal increase in prediction error (see Table 2). Similarly, when the network is trained with noisy labels and tested with noise-free data, we observe a marginal rise in prediction error in this experiment as well (see Table 3). The network's robustness to noise can be attributed to the use of Dropouts in our network architecture which allows the neural network to learn more generalized solutions for regression tasks.
3. To enable the application of our architecture in real-world scenarios where reliable ground truth may be unavailable, we propose a sequence-to-sequence linking of initial conditions to be used as a branch input. This involves using the last output prediction from the current signal train as an initial condition for predicting the next signal train. However, this process results in a higher cumulative error as compared to extracting the initial conditions from the output data (when available), as demonstrated in Table 4. From Fig. 8, it is evident that the cumulative error varies with the prediction window. This variation indicates that the accuracy of the model fluctuates based on the location of the prediction window. Thus, a sequence-to-sequence approach offers a practical solution in cases where ground truth data is limited or uncertain.
4. We employ an ensemble mean approach with Dropouts to quantify the epistemic uncertainty of our DeepONet model. The worst-case relative error at the $\mu + 2\sigma$ limit was found to be 12.6%. When compared to the relative error with respect to the ensemble mean, the epistemic error falls within the same range as the error obtained from our deterministic model (see Table 5).

In summary, the operator-based neural network model we present here shows good potential in diesel engine modeling by offering real-time, accurate, and robust predictions even under noisy conditions. We envision the use of this method in control system optimization and online health monitoring which requires real-time estimation of dynamic engine states.

10 Future work

As a continuation of this work, we are actively investigating methods to extend the DeepONet model's generalization capabilities across various operating conditions. Ambient pressure and temperature conditions can influence the engine's response. To address this, we augment the model by adding a new branch network that takes the ambient pressure and temperature as input and thereafter, training the new model with simulated data generated with varying ambient conditions. This extension holds the potential to improve the model's robustness and widen its applicability in practical scenarios where engines operate under diverse environmental conditions.

Additionally, we are also working on integrating our data-driven DeepONet model with a physics-regularized model for estimating critical, identifiable parameters such as the maximum area of the EGR valve (A_{egrmax}), the maximum area of the turbine (A_{vgtmax}), and others (see [54] for details). The primary objective of this work is to develop an accurate, real-time model by combining data and physics-based approaches to estimate parameters of engine sub-systems such as EGR valve and Turbocharger sub-system, for online diagnosis and health monitoring. To achieve this, we leverage our operator network along with the Transfer learning paradigm combined with physics-based regularization to identify these unknown parameters from the set of ODEs presented in [20]. We use automatic differentiation of output states with respect to time (which is provided as input to the trunk network in our architecture) to facilitate this approach. In Fig. 11, we present initial results for the combined operator and physics-based ANN model for the estimation of parameters A_{egrmax} and A_{vgtmax}, which represent the maximum theoretical area of the EGR and VGT valve respectively. By

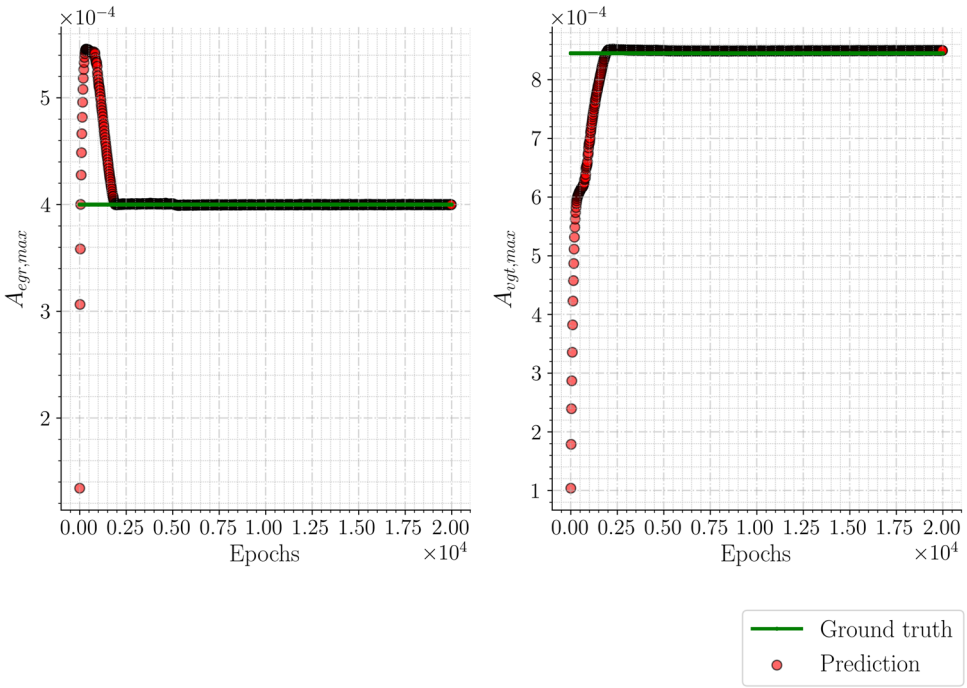

Fig. 11 Convergence of parameter A_{egrmax} (left) and A_{vgtmax} (right) for the combined DeepONet and PINN-based ANN model used for solving the inverse problem of parameter identification from a set of ODEs. Note that these results were generated from a preliminary study and are included here to highlight the advantages of operator-based networks over traditional methods for combining physics and neural network models

estimating the variation in the values of these parameters, the health of these two valve system may be ascertained. Our focus is on refining this integrated data plus physics model to further improve its accuracy and applicability for real-world applications, thereby contributing towards more effective and efficient engine performance optimization and diagnostics.

Appendix: DeepONet prediction for 1000 sec test data

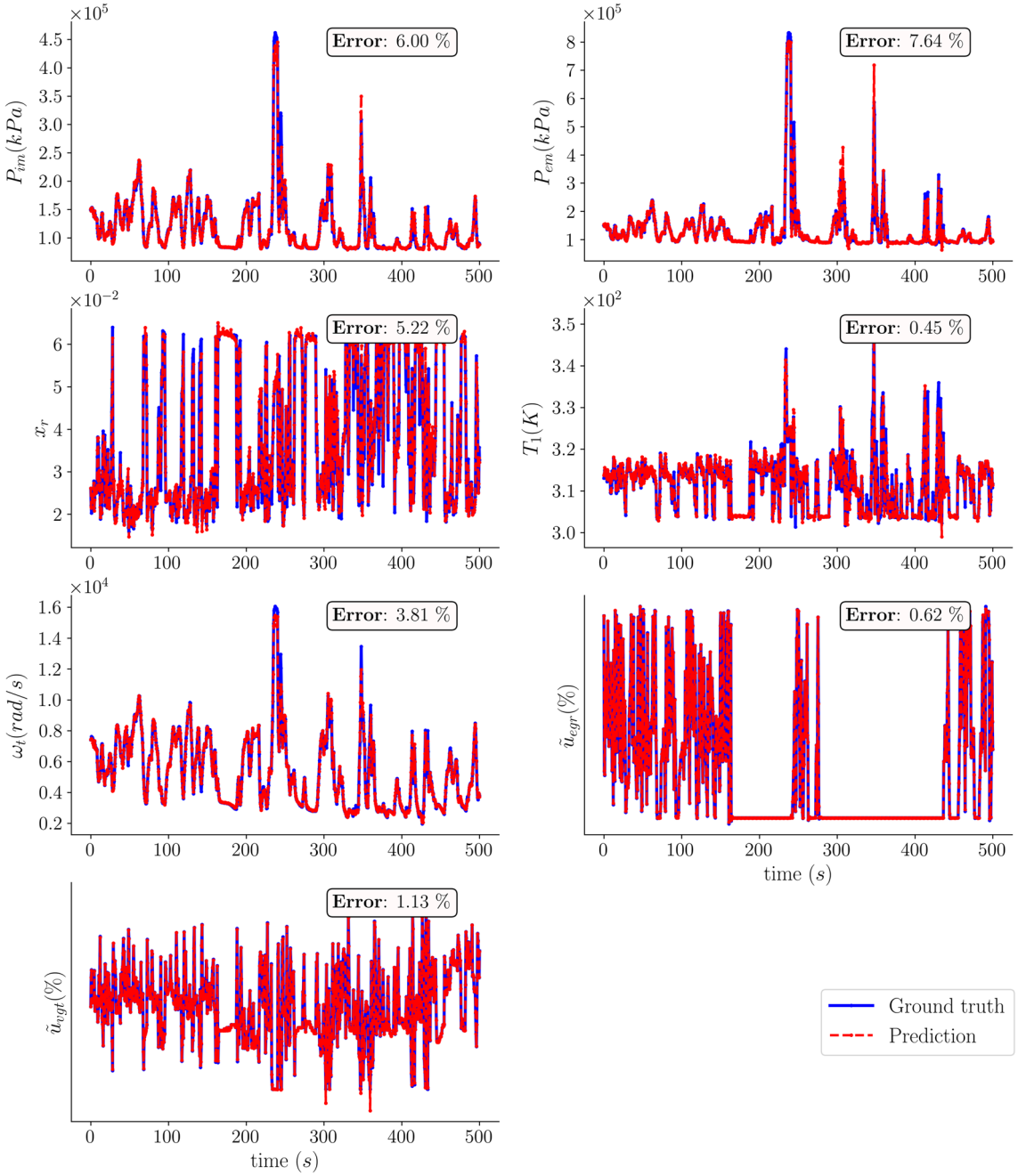

Fig. 12 DeepONet prediction results for the seven output states and comparison with ground truth for a time window $0 - 500$ sec in test data. Y-axis markers for $\tilde{u}_{egr}$ and $\tilde{u}_{vgt}$ are not shown in the interest of confidentiality

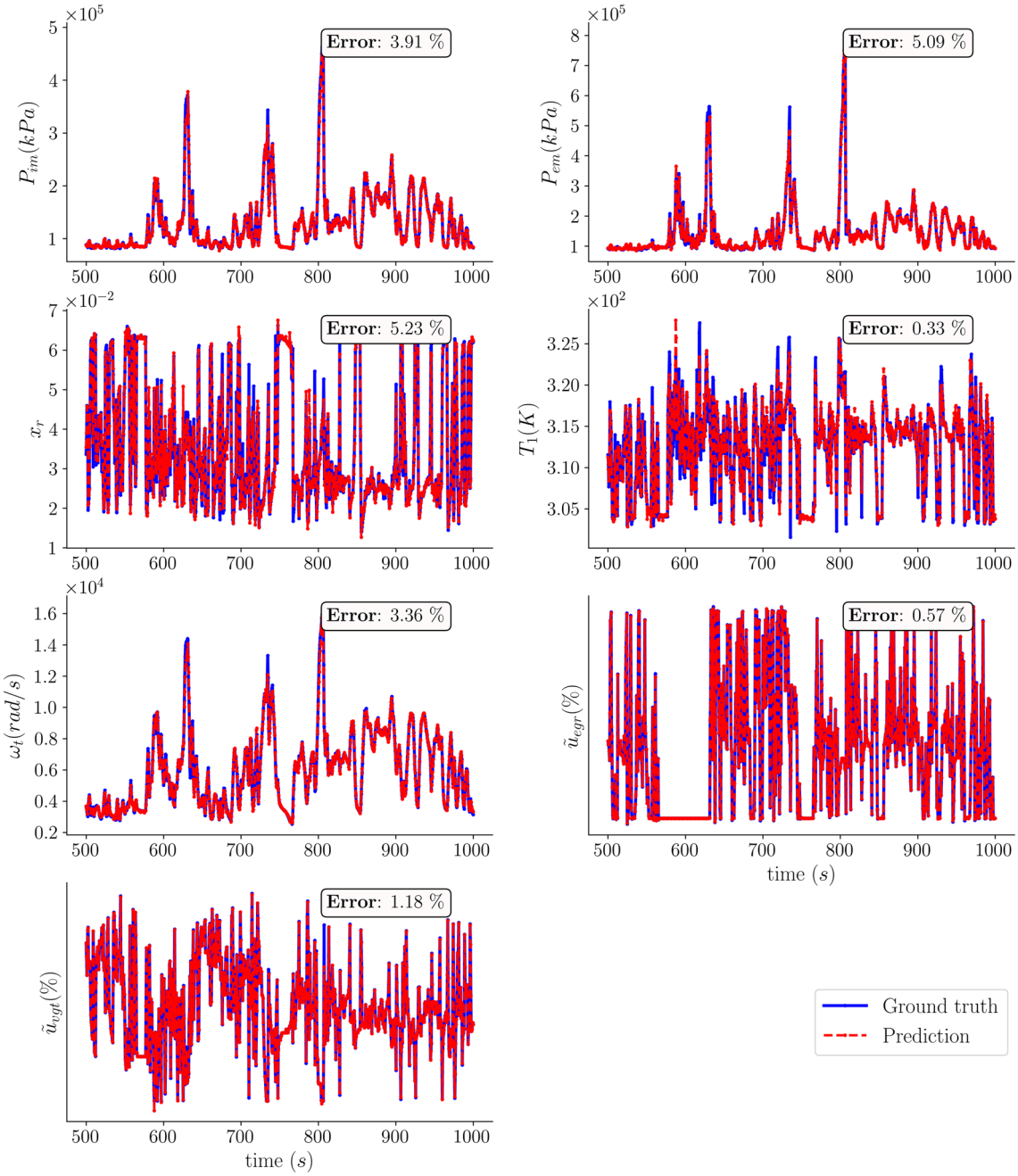

Fig. 13 DeepONet prediction results for the seven output states and comparison with ground truth for a time window $501 - 1000$ sec in test data. Y-axis markers for $\tilde{u}_{egr}$ and $\tilde{u}_{vgt}$ are not shown in the interest of confidentiality

Acknowledgements This research was conducted using computational resources and services at the Center for Computation and Visualization, Brown University.

Author Contributions Varun Kumar was responsible for data generation, data processing, machine learning model design, coding, result interpretation, and material preparation. Somdatta Goswami was responsible for providing expertise in operator network design, material preparation, and reviewing manuscripts. Daniel Smith provided the necessary guidance for data generation, problem setup, and manuscript review. George Karniadakis provided critical feedback on the manuscript and methods used in this study.

Funding This study was funded by Cummins Inc.

Data availability and access The datasets generated during and/or analyzed during the current study are available from the corresponding

author and with permission from Cummins Inc. on reasonable request post-publication.

Declarations

Competing interests Author Varun Kumar and Somdatta Goswami declare they have no financial interest. Author George Em Karniadakis has received research support from Cummins Inc. Karniadakis provides technical advice on the direction of machine learning to Anailytica, a private startup company developing AI software products for engineering. He has a very small equity in his work. The rest of the authors declare no competing interest.

Ethical and informed consent for data used Not applicable.

References

1. Monk J, Comfort J (1970) Mathematical model of an internal combustion engine and dynamometer test rig. Measure Control 3(6):T93–T100
2. Harland G, Gill K (1973) Design of a model-reference adaptive control for an internal combustion engine. Measure Control 6(4):167–173
3. Blumberg PN, Lavoie GA, Tabaczynski RJ (1979) Phenomenological models for reciprocating internal combustion engines. Prog Energy Combust Sci 5(2):123–167
4. Aizenbud BM, Band YB, Kafri O (1982) Optimization of a model internal combustion engine. J Appl Phys 53(3):1277–1282
5. Rizzoni G, Min P (1991) Detection of sensor failures in automotive engines. IEEE Trans Veh Technol 40(2):487–500
6. Thompson G, Atkinson C, Clark N, Long T, Hanzevack E (2000) Neural network modelling of the emissions and performance of a heavy-duty diesel engine. Proc Inst Mech Eng, Part D: Jf Autom Eng 214(2):111–126
7. Campa G, Thiagarajan M, Krishnamurty M, Napolitano MR, Gautam M, A neural network based sensor validation scheme for heavy-duty diesel engines
8. Park J, Sandberg IW (1991) Universal approximation using radial-basis-function networks. Neural Comput 3(2):246–257
9. Wang Y-Y, He Y, Rajagopalan S (2011) Design of engine-out virtual NOx sensor using neural networks and dynamic system identification. SAE Int J Engines 4(1):828–836
10. Ismail HM, Ng HK, Queck CW, Gan S (2012) Artificial neural networks modelling of engine-out responses for a light-duty diesel engine fuelled with biodiesel blends. Appl Energy 92:769–777
11. Shamekhi A-M, Shamekhi AH (2015) A new approach in improvement of mean value models for spark ignition engines using neural networks. Expert Syst Appl 42(12):5192–5218
12. Li H, Butts K, Zaseck K, Liao-McPherson D, Kolmanovsky I (2017) Emissions modeling of a light-duty diesel engine for model-based control design using multi-layer perceptron neural networks. Tech. rep, SAE Technical Paper
13. Luján JM, Climent H, García-Cuevas LM, Moratal A (2017) Volumetric efficiency modelling of internal combustion engines based on a novel adaptive learning algorithm of artificial neural networks. Appl Therm Eng 123:625–634
14. Taglialatela F, Lavorgna M, Di Iorio S, Mancaruso E, Vaglieco BM (2017) Real time prediction of particle sizing at the exhaust of a diesel engine by using a neural network model. SAE Int J Engines 10(4):2202–2208
15. Fravolini ML, Cone A, Napolitano M, Pradhan S, Thiruvengadam A, Selimi B (2018) Comparative analysis of performance of neural estimators for diagnostics in engine emission system. SAE Int J Engines 11(3):277–288
16. Domínguez-Sáez A, Rattá GA, Barrios CC (2018) Prediction of exhaust emission in transient conditions of a diesel engine fueled with animal fat using artificial neural network and symbolic regression. Energy 149:675–683
17. Zhao G-F, Long Y, Ding S-L, Yang L-P, Song E-Z, Ma X-Z (2020) Study of advanced control based on the rbf neural network theory in diesel engine speed control. SAE Int J Engines 13(1):63–76
18. Shin S, Lee Y, Park J, Kim M, Lee S, Min K (2021) Predicting transient diesel engine nox emissions using time-series data preprocessing with deep-learning models. Proc Inst Mech Eng Part D: J Autom Eng 235(12):3170–3184
19. Pulpeiro González J, Ankobea-Ansah K, Peng Q, Hall CM (2022) On the integration of physics-based and data-driven models for the prediction of gas exchange processes on a modern diesel engine. Proc Inst Mech Eng Part D: J Autom Eng 236(5):857–871
20. Wahlström J, Eriksson L (2011) Modelling diesel engines with a variable-geometry turbocharger and exhaust gas recirculation by optimization of model parameters for capturing non-linear system dynamics. Proc Inst Mech Eng Part D: J Autom Eng 225(7):960–986
21. AVL, AVL Boost Engine simulation, https://www.avl.com/boost. Accessed 08 Aug 2022
22. Gamma Technologies, GT Power Engine Simulation, https://www.gtisoft.com/gt-power/, Accessed 08 Aug 2022
23. Ricardo Inc, WAVE 1D simulation, https://software.ricardo.com/products/wave, Accessed 08 Aug 2022
24. Hendricks E (1986) A compact, comprehensive model of large turbocharged, two-stroke diesel engines, SAE Trans:820–834
25. Watson N (1984) Dynamic turbocharged diesel engine simulator for electronic control system development
26. Kimmich F, Schwarte A, Isermann R (2005) Fault detection for modern Diesel engines using signal-and process model-based methods. Control Eng Pract 13(2):189–203
27. Wu H, Wang X, Winsor R, Baumgard K (2011) Mean value engine modeling for a diesel engine with GT-Power 1D detail model. Tech. rep, SAE Technical Paper
28. Svard C, Nyberg M (2010) Residual generators for fault diagnosis using computation sequences with mixed causality applied to automotive systems. IEEE Trans Syst Man Cybern Part A: Syst and Humans 40(6):1310–1328
29. Han Z, Reitz RD (1995) Turbulence modeling of internal combustion engines using RNG κ-ε models. Combust Sci Technol 106(4–6):267–295
30. Goswami S, Anitescu C, Rabczuk T (2020) Adaptive fourth-order phase field analysis using deep energy minimization. Theoret Appl Fract Mech 107:102527
31. Goswami S, Yin M, Yu Y, Karniadakis GE (2022) A physics-informed variational DeepONet for predicting crack path in quasi-brittle materials. Comput Methods Appl Mech Eng 391:114587
32. Dabney JB, Harman TL (2004) Mastering Simulink, Vol. 230, Pearson/Prentice Hall Upper Saddle River
33. Lu L, Jin P, Pang G, Zhang Z, Karniadakis GE (2021) Learning nonlinear operators via DeepONet based on the universal approximation theorem of operators. Nature Mach Intell 3(3):218–229
34. Goswami S, Yin M, Yu Y, Karniadakis GE (2022) A physics-informed variational deeponet for predicting crack path in quasi-brittle materials. Comput Methods Appl Mech Eng 391:114587
35. Lin C, Li Z, Lu L, Cai S, Maxey M, Karniadakis GE (2021) Operator learning for predicting multiscale bubble growth dynamics. J Chem Phys 154(10):104118
36. Cai S, Wang Z, Lu L, Zaki TA, Karniadakis GE (2021) DeepM&Mnet: Inferring the electroconvection multiphysics fields based on operator approximation by neural networks. J Comput Phys 436:110296

37. Chen T, Chen H (1995) Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. IEEE Trans Neural Netw 6(4):911–917
38. Jin P, Meng S, Lu L (2022) Mionet: Learning multiple-input operators via tensor product. SIAM J Sci Comput 44(6):A3490–A3514
39. Goswami S, Li DS, Rego BV, Latorre M, Humphrey JD, Karniadakis GE (2022) Neural operator learning of heterogeneous mechanobiological insults contributing to aortic aneurysms. J R Soc Interface 19(193):20220410
40. Gal Y, Ghahramani Z (2016) Dropout as a Bayesian approximation: representing model uncertainty in deep learning, In: International conference on machine learning, PMLR, pp 1050–1059
41. McClenny L, Braga-Neto U, Self-adaptive physics-informed neural networks using a soft attention mechanism, arXiv:2009.04544
42. Kontolati K, Goswami S, Shields MD, Karniadakis GE (2023) On the influence of over-parameterization in manifold based surrogates and deep neural operators. J Comput Phys 479:112008
43. MacKay DJ (1992) A practical Bayesian framework for backpropagation networks. Neural Comput 4(3):448–472
44. Jospin LV, Laga H, Boussaid F, Buntine W, Bennamoun M (2022) Hands-on Bayesian neural networks—A tutorial for deep learning users. IEEE Comput Intell Mag 17(2):29–48
45. Psaros AF, Meng X, Zou Z, Guo L, Karniadakis GE (2023) Uncertainty quantification in scientific machine learning: Methods, metrics, and comparisons. J Comput Phys 477:111902
46. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15(1):1929–1958
47. Hastings WK (1970) Monte Carlo sampling methods using Markov chains and their applications, Oxford University Press
48. Bardenet R, Doucet A, Holmes CC, On Markov chain Monte Carlo methods for tall data, J Mach Learn Res 18 (47)
49. Neal RM et al (2011) MCMC using Hamiltonian dynamics. Handbook Markov Chain Monte Carlo 2(11):2
50. Blundell C, Cornebise J, Kavukcuoglu K, Wierstra D (2015) Weight uncertainty in neural network, In: International conference on machine learning, PMLR, pp 1613–1622
51. Hernández-Lobato JM, Adams R (2015) Probabilistic backpropagation for scalable learning of Bayesian neural networks, In: International conference on machine learning, PMLR, pp 1861–1869
52. Blei DM, Kucukelbir A, McAuliffe JD (2017) Variational inference: A review for statisticians. J Am Stat Assoc 112(518):859–877
53. Meng X, Yang L, Mao Z, del Águila Ferrandis J, Karniadakis GE (2022) Learning functional priors and posteriors from data and physics. J Comput Phys 457:111073
54. Nath K, Meng X, Smith DJ, Karniadakis GE (2023) Physics-informed neural networks for predicting gas flow dynamics and unknown parameters in diesel engines. Sci Rep 13:13683. https://doi.org/10.1038/s41598-023-39989-4