ARCHIVIO ISTITUZIONALE
DELLA RICERCA

Alma Mater Studiorum Università di Bologna
Archivio istituzionale della ricerca

AI-enabled cardiovascular models trained on multifidelity simulations data

This is the final peer-reviewed author's accepted manuscript (postprint) of the following publication:

Published Version:
Chiofalo, A., Careddu, L., Ciriello, V., Tartakovsky, D. (2025). AI-enabled cardiovascular models trained on multifidelity simulations data. JOURNAL OF MACHINE LEARNING FOR MODELING AND COMPUTING, 6, 1-17 [10.1615/jmachlearnmodelcomput.2025058368].

Availability:
This version is available at: https://hdl.handle.net/11585/1012942 since: 2025-03-31

Published:
DOI: http://doi.org/10.1615/jmachlearnmodelcomput.2025058368

Terms of use:

Some rights reserved. The terms and conditions for the reuse of this version of the manuscript are specified in the publishing policy. For all terms of use and more information see the publisher's website.

This item was downloaded from IRIS Università di Bologna (https://cris.unibo.it/).
When citing, please refer to the published version.

(Article begins on next page)

11 September 2026

AI-ENABLED CARDIOVASCULAR MODELS TRAINED ON MULTIFIDELITY SIMULATIONS DATA

A. Chiofalo,[1] L. Careddu,[2] V. Ciriello,[1] & D.M. Tartakovsky[3,*]

[1]*Department of Civil, Chemical, Environmental and Materials Engineering, University of Bologna, Italy*

[2]*Pediatric Cardiac Surgery and Adult Congenital Heart Disease Program, Department of Cardio-Thoracic and Vascular Medicine, IRCCS Azienda Ospedaliero-Universitaria di Bologna, Italy*

[3]*Department of Energy Science and Engineering, Stanford University, California, USA*

*Address all correspondence to: D.M. Tartakovsky, Department of Energy Science and Engineering, Stanford University, California, USA, E-mail: tartakovsky@stanford.edu

Original Manuscript Submitted: mm/dd/yyyy; Final Draft Received: mm/dd/yyyy

Neural networks (NNs) are routinely used to accelerate ensemble-based computations (EBCs) of physical systems governed by partial differential equations. However, the high computational costs of training NN-based surrogates rival those of EBCs, limiting their practical utility. We address this challenge by integrating multifidelity data, obtained by running models of different complexity and reliability, and transfer learning to optimize the training of deep convolutional neural networks (CNNs). The data are generated via patient-specific hemodynamic simulations of an aorta, leveraging high- and low-fidelity models and systematically varying boundary conditions and blood properties within the Monte Carlo framework. Our results demonstrate that CNN surrogates benefit from a preliminary tuning on low-fidelity data and achieve high accuracy, in comparison to high-fidelity simulations, with significantly reduced computational costs, paving the way for efficient, patient-specific cardiovascular assessments.

KEY WORDS: *transfer learning, CNN, multi-fidelity, multi-model, fluid dynamics*

1. INTRODUCTION

Scientific machine learning (ML) methods are routinely used in various disciplines, including fluid mechanics. Neural networks (NNs) are a prominent ML technique that has been increasingly integrated into scientific computing (Chandra et al., 2024; Karniadakis et al., 2021; Shen et al., 2023). We focus on their use as surrogates to accelerate ensemble-based computations (EBCs).

EBCs need repeat solves of (coupled, nonlinear) partial differential equations (PDEs), e.g.,

$$\mathcal{N}(\mathbf{u};\boldsymbol{\theta}) = g(\mathbf{x},\ t;\boldsymbol{\theta}),\ (\mathbf{x},\ t) \in D \times (0,\ T]. \tag{1}$$

Equation (1) describes the spatiotemporal evolution of a state variable $\mathbf{u}(\mathbf{x}, t)$ in the computational domain D over simulation time horizon $(0,\ T]$. In a probabilistic framework, to quantify the inevitable predictive uncertainty, a large number of samples in the space of variability of the inputs $\boldsymbol{\theta}(\mathbf{x},\ t)$, that parameterize the differential operator $\mathcal{N}$, the source function g, and auxiliary functions in the initial and boundary conditions (BCs), is required. This task is notoriously challenging due to the high-dimensional parameter space and the computational cost of multiple model runs.

The computational cost of EBCs can be alleviated by employing a surrogate, which provides a cost-effective approximation of its expensive physics-based counterpart (Libero et al., 2024b; Oladyshkin et al., 2023; Xiu and Karniadakis, 2002). A key challenge faced by various surrogate models is the curse of dimensionality, which refers to the decline in efficiency as the number of random inputs increases (Barajas-Solano and Tartakovsky, 2016; Xiu, 2009).

NNs offer a robust solution for handling large sets of inputs and outputs. However, NNs thrive on big data, hence a high number of model runs for different realizations of model parameters $\boldsymbol{\theta}(\mathbf{x}, ; t)$ is required for training. Consequently, the use of a NN as a surrogate is justified only if the cost of generating training data is smaller than the cost of EBCs. However, these computational costs are often comparable, which raises questions about the practicality of NN-based

surrogates (Song and Tartakovsky, 2022).

One solution involves employing multi-fidelity simulations to accelerate the generation of training data. In various fields, multi-fidelity composite NNs leverage low-fidelity (LF) and high-fidelity (HF) data. While abundant, LF data are less accurate, whereas HF data are highly accurate but costly to generate (Zhang and Alemazkoor, 2024). For instance, Geneva and Zabaras (2020) trained surrogate models using LF simulations as a conditional input; Meng and Karniadakis (2020) developed fully connected NN surrogates by training separate networks for LF and HF data; Howard et al. (2024) introduced a continual learning approach based on multi-fidelity deep NNs; Song and Tartakovsky (2022) applied transfer learning (TL) (Donahue et al., 2014) on multi-fidelity data generated by solving PDEs on fine and coarse numerical grids; and Propp and Tartakovsky (2025) trained a CNN surrogate model on a mixture of numerical solutions to both the d-dimensional problem and its (d-1) dimensional approximation.

In this study, we use a deep convolutional neural network (CNN) as a surrogate and TL to integrate multi-fidelity data into the training process. Our approach employs a multi-model framework, where data are generated using physical models of varying complexity, rather than a single model solved on different numerical grids. Specifically, we utilize `SimVascular` (Updegrove et al., 2016), an open-source software providing a complete pipeline from medical image segmentation to patient-specific blood flow simulations (`http://simvascular.github.io/`). This software enables the generation of multi-fidelity flow simulations within the vessel pathways of a patient-specific aorta, available in the public Vascular Model Repository (`http://vascularmodel.org/`), which houses cardiovascular models developed from volumetric image datasets and relevant physiological data (Wilson et al., 2013).

Numerous studies have emphasized the role of flow-related metrics in the progression or regression of cardiac pathologies, underscoring the growing importance of fluid dynamics in clinical assessments. Patient-specific analyses are impractical in near real-time; thus, clinical practice necessitates tools that generalize insights from limited individual cases (Collia et al., 2022). Reduced-order models can reveal dominant flow features indicative of pathologies or approximate the response surface of flow-related metrics derived from physically based models.

This facilitates the optimization and design of patient-specific interventions at negligible simulation costs (Collia et al., 2022; Libero et al., 2024a; Xiu and Sherwin, 2007).

In our study, we apply variations of $\pm 30\%$ at clinical values of a set of governing parameters for the selected aorta model (Fleeter et al., 2020). Hemodynamics in the vasculature is influenced by BCs at the inlet and outlets, as well as by blood density and viscosity. We also account for downstream vasculature represented by Windkessel parameters (Vignon-Clementel et al., 2010). Using Monte Carlo simulations, we sample the variability space of these parameters and generate multi-fidelity simulations with 1D and 0D models (Pfaller et al., 2022), HF and LF model, respectively.

We demonstrate that the CNN surrogate benefits from the points of view of efficiency and accuracy from a preliminary tuning stage based on LF data. In our numerical experiments, for data generation budgets ranging from 10 to 20 hours, the optimum LF-HF data ratio is about 5, consistently with the range recommended in a theoretical multilevel Monte Carlo study Taverniers et al. (2020).

The paper is organized as follows. Section 2 and relatively subsections describe the NN architecture and the workflow of its training on multi-fidelity data. In Section 3, we briefly explain the physical models used to acquire the dataset, while our application is presented in Section 4. In Section 4.2 we demonstrate the accuracy and computational efficiency of the CNN-based surrogate. Finally, Section 5 provides concluding remarks.

2. MATERIALS AND METHODS

This section describes the deep convolutional encoder-decoder NN surrogate model, which functions as an image-to-image regression model. Additionally, it demonstrates the use of TL on multi-model data to train the surrogate.

2.1 Deep Convolutional Encoder-Decoder Network

The deep convolutional encoder-decoder network is designed to serve as an effective surrogate for various fluid flow problems. We have selected the CNN with an encoder-decoder structure,

which has been previously applied to address multi-phase flow in porous media, using multi-fidelity data generated by solving PDEs on different grid configurations (Song and Tartakovsky, 2022).

The convolutional encoder-decoder architecture begins with initial layers that process the input dimensions. The following layers encode the input into a reduced dimension known as the latent state, which serves as a low-dimensional representation of the original inputs. This encoded latent state is then decoded to match the output dimensions. The encoder-decoder framework functions as a dimensionality reduction technique, helping to address the curse of dimensionality. This is a significant factor in the failure of many surrogate methods when training on high-dimensional inputs with limited data.

Within the architecture, the CNN layers are organized into dense blocks. In each dense block, all layers are directly connected to ensure maximum information flow. To maintain the feed-forward nature of the model, each layer receives input from all preceding layers and forwards its feature maps to all subsequent layers. Each dense block is defined by two key parameters: the number of internal layers and the number of output features for each layer. Each layer comprises three consecutive operations: batch normalization, a rectified linear unit (ReLU), and a convolution. Between each set of dense blocks, there is a transition layer that modifies the feature size using a convolutional operation during the encoding process and a transpose convolutional operation during the decoding process.

For training and testing the network, we use values of $\boldsymbol{\theta}(\mathbf{x}_i)$ in N_{el} elements $\{\mathbf{x}_i\}_{i=1}^{N_{\text{el}}}$ of a numerical grid as input and the discretized solution $\mathbf{u}(\mathbf{x}_i,\ t_k)$ of PDE at N_{ts} time steps $\{t_k\}_{k=1}^{N_{\text{ts}}}$ as output. To enhance the generalizability of the trained CNN to unseen sets of the input, and to prevent overfitting to specific instances of $\boldsymbol{\theta}(\mathbf{x}_i)$, the training data comprises a large number, N_{train}, of the solutions $\mathbf{u}$ obtained from realizations $\{\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, ..., \boldsymbol{\theta}_{N_{\text{train}}}\}$ of the input $\boldsymbol{\theta}$. The CNN training consists of finding a set of weights $\mathbf{w}$ that minimizes the loss function $\mathcal{L}$

$$\mathcal{L}(\mathbf{w}) = \sum_{m=1}^{N_{\text{train}}} \sum_{i=1}^{N_{\text{el}}} \sum_{k=1}^{N_{\text{ts}}} \|\mathbf{u}(\mathbf{x}_i,\ t_k; \boldsymbol{\theta}_m) - \hat{\mathbf{u}}_{ik}(\mathbf{w}; \boldsymbol{\theta}_m)\| + \lambda \sum_{n=1}^{N_{\text{w}}} w_n^2. \tag{2}$$

The first part of Equation (2) represents the L_1-norm discrepancy between the state variable $\mathbf{u}$ predicted by solving the governing equations using classical mathematical methods, $\mathbf{u}(\mathbf{x}_i,\ t_k)$, and estimated by the CNN, $\hat{\mathbf{u}}_{ik}(\mathbf{w})$, with N_w weights $\mathbf{w} = (w_1, ..., w_{N_\mathrm{w}})^\top$. The second part of (2) involves L_2-norm regularization, which penalizes large weights to prevent overfitting and improving generalization. The amount of penalty applied is determined by the regularization parameter λ (Song and Tartakovsky, 2022).

2.2 Transfer Learning for CNN Training

Traditional ML relies on the assumption that both training and testing data share the same input feature space and follow the same distribution. However, when there is a discrepancy in data distribution between the training and testing sets, the performance of a predictive model can decline. Acquiring training data that is consistent in both feature space and distribution characteristics with the test data can often be difficult and costly (Weiss et al., 2016). This challenge can be addressed through TL, which enhances the learning of a new task by leveraging knowledge gained from a related task that has already been learned (Akhavan-Safaei and Zayernouri, 2024; Donahue et al., 2014; Krawczuk and Venturi, 2021). By utilizing the knowledge encoded in a NN trained on the initial task, it is possible to significantly reduce the number of training samples needed for the subsequent task.

Here, we apply TL to data generated by physical models with varying complexity. Let HF and LF data refer to the values of the state variable $\mathbf{u}(\mathbf{x}_i,\ t_k)$, obtained with the high-complexity physical model ($N_\mathrm{el} = N_\mathrm{el}^\mathrm{HF}$) and the low-complexity physical model ($N_\mathrm{el} = N_\mathrm{el}^\mathrm{LF}$), respectively. In the method used in this study, the CNN is initially trained using a large amount of LF data. Subsequently, a relatively small amount of HF data is used to fine-tune the network, thereby completing the surrogate model with an accuracy comparable to that of the high-complexity physical model.

The TL process consists of three phases, involving the trained CNNs denoted as CNN_1, CNN_2, and CNN_3. In Equation (2), let N_w represent the number of weights in the CNN preliminary trained on a small number of HF data. During the training of CNN_1 on the LF data,

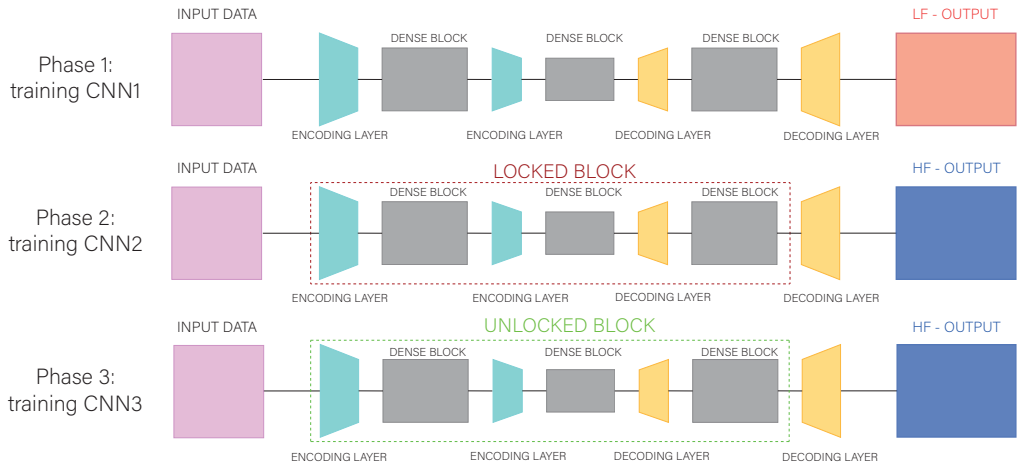

FIG. 1: Workflow of CNN training on multi-fidelity data via TL. During Phase 1, LF output data are used and the weights ($\mathbf{w}_{\mathrm{LF}}$) are updated in each layer. In Phase 2, HF data are used to update the weights in the final layer ($\mathbf{w}_{\mathrm{HF}}$) while the others are frozen. In Phase 3, all weights are unlocked and updated during training on the same HF data.

the weights $\mathbf{w}_{\mathrm{LF}} = (w_1, w_2, ..., w_{N_{\mathrm{LF}}})^\top$ with $\mathbf{w}_{\mathrm{LF}} < N_{\mathrm{w}}$ are updated. In the second step, we train CNN_2 on the HF data by minimizing the loss function (2) over the weights $\mathbf{w}_{\mathrm{HF}} = (w_{N_{\mathrm{LF}+1}}, w_{N_{\mathrm{LF}+2}}, ..., w_{N_{\mathrm{w}}})^\top$ of the last layer, while keeping the weights $\mathbf{w}_{\mathrm{LF}}$ of the remaining layers of the architecture fixed at their values from the first phase. For the last phase, we train CNN_3 on the HF data by allowing all weights $\mathbf{w}$ of the CNN_2 to vary during the minimization (see Figure 1). In Algorithm 1, a pseudo-code of the training of the CNN on multi-fidelity data is reported.

3. APPLICATION

Blood flow in the cardiovascular system is typically modeled by the incompressible Navier-Stokes equations,

$$\begin{aligned} \rho(\dot{\mathbf{u}} + \mathbf{u} \cdot \nabla \mathbf{u}) &= \nabla \cdot \boldsymbol{\tau} + \rho \mathbf{b}, \\ \nabla \cdot \boldsymbol{\tau} &= -p\mathbf{I} + \mu(\nabla \mathbf{u} + \nabla \mathbf{u}^{\intercal}), \\ \nabla \cdot \mathbf{u} &= 0, \quad \mathbf{x} \in \Omega^{3\mathrm{D}}, \quad t \in \mathbb{R}_{\geq 0}. \end{aligned} \tag{3}$$

These equations establish the relationship between the velocity field $\mathbf{u}(\mathbf{x}, t)$ and the pressure field $p(\mathbf{x}, t)$ for blood with density ρ and dynamic viscosity μ, subject to a body force, $\mathbf{b}(\mathbf{x}, t)$.

Algorithm 1: Training algorithm with phases 1, 2, and 3

Require : parameters θ_i, label data N_{el}^{LF}, N_{el}^{HF}
Ensure : final optimized model CNN_3
Phase 1: Train on LF data;
for $i \leftarrow 1$ ***to*** *epochs*$_1$ **do**
  Train CNN using training LF data;
  Compute $RMSE$ using test LF data;
  Set $RMSE_{check}$ as mean of last 10 $RMSE_{test}$;
  if $RMSE_{check} < RMSE_{best}$ **then**
    Update $RMSE_{best}$;
    Save current CNN as best CNN_1;
  end
end
Phase 2: Train last layer on HF data;
Freeze all layers of best CNN_1 except the last transition layer;
for $i \leftarrow 1$ ***to*** *epochs*$_2$ **do**
  Train last transition layer of best CNN_1 using training HF data;
  Compute $RMSE$ using test HF data;
  Set $RMSE_{check}$ as mean of last 10 $RMSE_{test}$;
  if $RMSE_{check} < RMSE_{best}$ **then**
    Update $RMSE_{best}$;
    Save current model as best CNN_2;
  end
end
Phase 3: Fine-tuning of the overall network on HF data;
Unfreeze all layers of best CNN_2;
for $i \leftarrow 1$ ***to*** *epochs*$_3$ **do**
  Train all layers of best CNN_2 using training HF data;
  Compute $RMSE$ using test HF data;
  Set $RMSE_{check}$ as mean of last 10 $RMSE_{test}$;
  if $RMSE_{check} < RMSE_{best}$ **then**
    Update $RMSE_{best}$;
    Save current model as best CNN_3;
  end
end
return best CNN_3;

Here, $\boldsymbol{\tau}(\mathbf{x}, t)$ denotes the stress tensor, $\mathbf{I}$ is the identity matrix, and Ω^{3D} represents the volume domain of the fluid (Pfaller et al., 2022).

To close this system of equations, initial and BCs must be specified. In patient-specific modeling, the anatomical portion of interest is modeled. The geometric boundaries of the model define the inlet and outlet caps, where the BCs are set to simulate the impact of blood vessels surrounding Ω^{3D}. In open-loop models, the conditions at the entry and exit are treated independently and are not mathematically related. In closed-loop models, the inlet and outlet BCs are numerically coupled (Pfaller et al., 2022). Focusing on open-loop models, a Dirichlet BC at the inlet describing the flow rate, Q_{in} with a given velocity profile, $\mathbf{u}_{\text{in}}(\mathbf{x},\ t)$, is typically prescribed as follows

$$\mathbf{u}(\mathbf{x},\ t) = \mathbf{u}_{\text{in}}(t), \qquad \mathbf{x} \in \Gamma_{\text{in}}, \quad t \in \mathbb{R}_{\geq 0}, \tag{4}$$

where Γ_{in} represents the inlet cap surface of the model. At the outlets, BCs to relate the flow rate to the pressure through algebraic-differential equations may be defined as follows

$$P_{\text{out}}(\mathbf{x},\ t) = f(\mathbf{x},\ t,\ Q_{\text{out}}(t),\ \dot{Q}_{\text{out}}(t),\ \boldsymbol{\phi}), \qquad \mathbf{x} \in \Gamma_{\text{out}}, \quad t \in \mathbb{R}_{\geq 0}, \tag{5}$$

where $\boldsymbol{\phi}$ is a set of parameters governing the lumped-parameter elements and Γ_{out} is an outlet cap surface. Finally, at the walls of the blood vessels, no-slip conditions are imposed

$$\mathbf{u}(\mathbf{x},\ t) = 0, \qquad \mathbf{x} \in \Gamma_w, \quad t \in \mathbb{R}_{\geq 0}, \tag{6}$$

where Γ_w is the wall surface domain of the fluid. These 3D PDEs are solved using numerical methods to determine spatial and temporal distributions of hemodynamics, including the velocity and pressure of patient-specific vascular anatomies.

`SimVascular` (`http://simvascular.github.io/`) is open-source software that offers a comprehensive pipeline, from medical image data segmentation to patient-specific blood flow simulations and analysis (Updegrove et al., 2016). The software enables setting diverse BCs. At the inlet, the choice is between plug, parabolic, or Wormersley velocity profiles. At the outlet,

one can model the downstream vasculature using Resistance, Windkessel resistance-capacitance-resistance, or Coronary lumped parameter BCs. Additionally, three options are available for specifying the vessel wall type: rigid, deformable constant, and deformable variable. No-penetration and no-slip BCs are applied to all options (for more details refer to Lan et al. (2018)).

For computational saving, `SimVascular` incorporates automated reduced-order 1D and 0D models from 3D patient-specific geometries. We utilize this software to conduct multi-fidelity simulations using different fidelity models, i.e., 1D and 0D models (Pfaller et al., 2022). To distinguish between the two models, we will refer to the 1D and 0D models as the HF and LF, respectively. We briefly review the two models below.

3.1 1D Model

The 1D model is obtained by integrating the 3D Navier-Stokes equations across the vessel cross-section. Therefore, the 1D equations only provide the temporal and axial distributions of bulk hemodynamic quantities

$$\begin{aligned} \frac{\partial Q}{\partial t} + \frac{4}{3}\frac{\partial}{\partial z}\left(\frac{Q^2}{S}\right) + \frac{S}{\rho}\frac{\partial P}{\partial z} &= Sf - N\frac{Q}{S} + \frac{\mu}{\rho}\frac{\partial^2 Q}{\partial z^2}, \\ \frac{\partial S}{\partial t} + \frac{\partial Q}{\partial z} &= 0, \quad z \in \Omega^{1\mathrm{D}}, \quad t \in \mathbb{R}_{\geq 0}. \end{aligned} \tag{7}$$

Equations (7) govern the interaction between the flow rate $Q(z,t)$, pressure $P(z,t)$ and cross-sectional area $S(z,t)$ subject to a body force $f(z,t)$ along the vessel's axial coordinate z. Here, $\Omega^{1\mathrm{D}}$ is the centerline domain of the blood vessel, while N depends on the choice of the velocity profile. An additional constitutive relationship between the pressure and the cross-sectional area is required to close the system. In `SimVascular` two possible constitutive material equations are available, linear and Olufsen respectively, defined as follows

$$P(z,\ t) = f(P^0(z),\ k_0,\ S(z,t),\ S^0(z)), k_0 = \frac{Eh}{r^0(z)} \tag{8}$$

$$P(z,\ t) = f(P^0(z),\ k_1,\ k_2,\ k_3,\ S(z,t),\ S^0(z)), k_1 e^{k_2 r^0(z)} + k_3 = \frac{Eh}{r^0(z)} \tag{9}$$

where E is the Young's modulus of the blood vessel, h is the wall thickness, P^0 is the reference pressure, r^0 is a reference radius, and k_0, k_1, k_2, k_3 are empirically derived material constant. This system of constitutive equations simulates deformable wall behavior. To invoke rigid-wall behavior we can set k_1 to zero and k_3 to an arbitrary large value (Pfaller et al., 2022). At the outlets, the same lumped-parameter network boundary of 3D simulation is used to represent the downstream vasculature. For detailed information, readers can refer to Pfaller et al. (2022). The open-source implementation of the 1D solver can be found at `https://github.com/SimVascular/svOneDSolver`.

3.2 0D Model

In the cardiovascular system, OD models, also known as lumped-parameter network models, can be compared to electrical circuits. In this analogy, the flow rate and pressure drop across the 0D elements correspond to current and voltage drop in an electrical circuit. These quantities are purely time-dependent. In this context, resistance models the viscous effects of blood flow, capacitance models the elastic deformability of blood vessel walls, and inductance models the inertia. The governing equations for these elements are

$$\Delta P = RQ, \qquad Q = C\Delta\dot{P}, \qquad \Delta P = L\dot{Q}, \tag{10}$$

where P is the pressure, Q is the flow rate, R is resistance, C is capacitance, and L is inductance. The values of the last three are dependent on the cardiovascular anatomy. For detailed information, readers can refer to Pfaller et al. (2022). The 0D simulation methods have been implemented as an open-source Python package available at `https://github.com/SimVascular/svZeroDSolver`.

3.3 Data Acquisition

A patient-specific aorta geometry (code: Model_0009_H_AO_H) has been chosen in the public Vascular Model Repository at `http://vascularmodel.org/` (Wilson et al., 2013). The

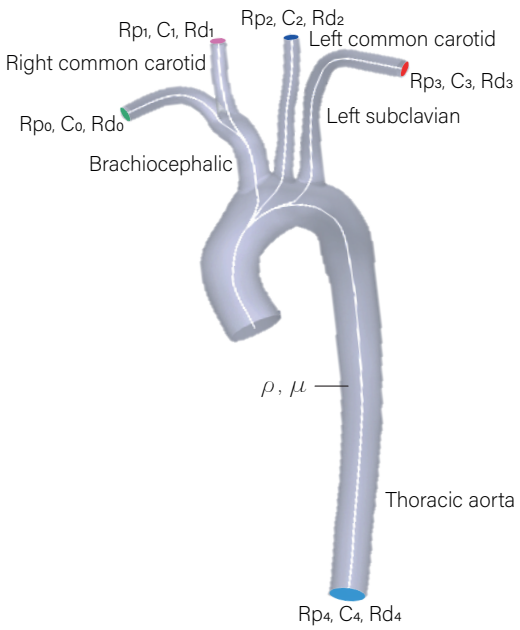

FIG. 2: Selected aorta geometry and parameters to vary described in Table 1.

selected geometry is shown in Figure 2. Under pulsatile flow conditions, we simulate blood pressure for one cardiac cycle with the 1D and 0D models. We consider linear elastic material properties for the arterial wall, at the inlet a parabolic velocity profile, and at the outlets, the Windkessel resistance-capacitance-resistance (RCR) to represent the downstream vasculature (Figure 2). The parameters and their values are included in APPENDIX A.

We identify the centerline from the 3D geometry and locate the network nodes at vessel inlets, outlets, and branching points. The length and diameter between vessel nodes are represented by a specific number of cylindrical segments. These segments are discretized into a mesh of finite elements to solve 1D equations. The 0D model, on the other hand, cannot replicate spatial patterns in hemodynamics. It only allows the estimation of pressure distribution over time at the inlet and outlet nodes of each segment within the simplified geometry. To compare the pressure data along the aorta geometry in two different models, we used linear interpolation in the 0D model between two known nodes to estimate the quantity of interest along the spatial dimension of the aorta.

Parameter	Clinical reference value
Proximal Resistance $\mathrm{R_p}$	APPENDIX A
Capacitance C	APPENDIX A
Distal Resistance $\mathrm{R_d}$	APPENDIX A
Blood density ρ	1.06
Blood viscosity μ	0.04

TABLE 1: Selected parameters to vary and correspondent clinical values for the selected aorta model: resistances ($\mathrm{dyn\,s/cm^5}$) and capacitance ($\mathrm{cm^5/dyn}$), whose values are defined in APPENDIX A for each outlet, blood density ($\mathrm{g/cm^3}$) and viscosity ($\mathrm{Pa \cdot s}$).

The model we have selected includes one inlet (ascending aorta) and five outlets (descending aorta, brachiocephalic artery, right common carotid artery, left common carotid artery, and left subclavian artery). To create the dataset needed for training the CNN, we use Monte Carlo simulations. These simulations involve treating key model parameters as independent random variables. In addition to the blood density and viscosity, we also consider the downstream vasculature represented by three-element Windkessel-type BCs (Table 1). The RCR parameters differ at each outlet (APPENDIX A). We define uniform distributions for the parameters, taking into account a variability range of $\pm\ 30\%$ around the clinical values provided by the software (Fleeter et al., 2020).

In our numerical experiments, we set the number of time steps to $N_{\mathrm{ts}} = 95$ corresponding to one cardiac cycle; in input, we consider the five independent variables mentioned above ($N_{\mathrm{p}} = 5$, including $\mathrm{R_p}$, C, $\mathrm{R_d}$, μ, ρ), multiplied by the number of outlet faces ($N_{\mathrm{outlets}} = 5$, namely descendent aorta, brachiocephalic, right common carotid, left common carotid, left subclavian), resulting in $N_{\mathrm{p}} \times N_{\mathrm{outlets}} = 25$. The number of channels in each of the six layers of the CNN is $n_1 = 64$, $n_2 = 344$, $n_3 = 172$, $n_4 = 652$, $n_5 = 326$, $n_6 = 606$. The output of all the CNNs corresponds to the pressure at each finite element ($N_{fe} = 60$) discretized along the segments that make up the branches ($N_{\mathrm{branches}} = 7$, including ascendant aorta, descendent aorta, brachiocephalic artery, right subclavian artery, right common carotid artery, left common carotid and left subclavian artery) of the aorta geometry, with dimensions as $N_{\mathrm{ts}} \times N_{\mathrm{fe}} \times N_{\mathrm{branches}}$. For the finite element discretization `SimVascular` employs a stabilized space-time finite element

method, known for its robustness, based on the Discontinuous Galerkin method in time (Updegrove et al., 2016) (`https://simvascular.github.io/documentation/rom\_simulation.html`). For the CNN architecture, the model block description and the input and output dimensions of each model block are reported in Table 2.

4. RESULTS AND DISCUSSION

The model implementation and training is done using Pytorch and other source packages. The computations are carried out utilizing 32 cores Intel Ice Lake @2.6 GHz CPU 512 GB RAM, 4 NVIDIA Ampere A100 GPU 64 GB vRAM. (Although 32 cores on a single node are available, only 1 core per node is used. All 4 GPUs are used) (Turisini et al., 2024).

4.1 Hyperparameter Optimization

The CNN's performance is primarily influenced by key hyperparameters such as Learning Rate (LR) and Weight Decay (WD), which are managed by the Adam optimizer (Kingma and Ba, 2014). The Adam optimizer is commonly used to optimize deep learning models and adjust the LR and WD during training to improve convergence and prevent overfitting. Additionally, a factor (F) and the minimum learning rate (mLR) are controlled by the `ReduceLROnPlateau` (Li and Arora, 2019) scheduler, which adjusts the learning rate during training based on the validation loss, enhancing the efficiency and convergence of the training process for CNNs. While CNN training includes numerous additional hyperparameters, we adhere to their default settings in PyTorch (Paszke et al., 2019).

We perform an investigation by varying LR, WD, F, and mLR in this specific order. Once a satisfactory value for a particular hyperparameter is identified, we move on to optimizing the next one. To carry out this optimization, we utilize 100 HF data, and the process involves 200 epochs. The hyperparameter values that result in the lowest root mean square error (RMSE) on the HF test data are selected (See Figure 3). These values serve as the starting point for optimizing hyperparameters for multi-fidelity TL. Specifically, we adjust the LR and the number of epochs in each phase to minimize the RMSE on the corresponding test data (see Table 3).

Layer	Activation functions	Input	Output size
Input: model parameters		$1 \times N_{\mathrm{p}} \times N_{\mathrm{outlets}}$	
Transition layer	conv (k (3,3), s (2,2) , p(1,1))	$n_1 \times N_{\mathrm{p}} \times N_{\mathrm{outlets}}$	$n_2 \times 3 \times 3$
Dense Block (7 layers)	BN, relu, conv (k (3, 3), s (1, 1), p (1, 1))	$n_2 \times 3 \times 3$	$n_3 \times 3 \times 3$
Transition layer (encoding)	BN, relu, conv(k(1, 1), s (1, 1));		
	BN, relu, conv(k (3, 3), s (2, 2), p (1, 1))	$n_3 \times 3 \times 3$	$n_4 \times 2 \times 2$
Dense Block (12 layers)	BN, relu, conv (k (3, 3), s (1, 1), p (1, 1))	$n_4 \times 2 \times 2$	$n_5 \times 2 \times 2$
Transition layer (decoding)	BN, relu, conv(k(1, 1), s (1, 1));		
	BN, relu, convTranspose(k (5, 3), s (5, 3), p (0, 1))	$n_5 \times 2 \times 2$	$n_6 \times 10 \times 4$
Dense Block (7 layers)	BN, relu, conv (k (3, 3), s (1, 1), p (1, 1))	$n_6 \times 10 \times 4$	$n_7 \times 10 \times 4$
Transition layer	convTranspose (k (6, 3), s (6, 2), p(0, 1))	$n_7 \times 10 \times 4$	$N_{ts} \times N_{\mathrm{fe}} \times N_{\mathrm{branches}}$
Output: Pressure		$N_{ts} \times N_{\mathrm{fe}} \times N_{\mathrm{branches}}$	

TABLE 2: Dense Encoder-Decoder CNN architecture.
Note: The values of kernel size, k, stride, s, and zero-padding, p, are denoted by pairs of numbers, with the first number corresponding to the convolutional (or transpose convolutional) operation over the height and the second over the width in pixels.
The convolution kernels are the same for all convolutional layers within the dense blocks. The first and second convolution kernels correspond to the first and second (transposed) convolutions of the encoding-decoding layers, respectively.

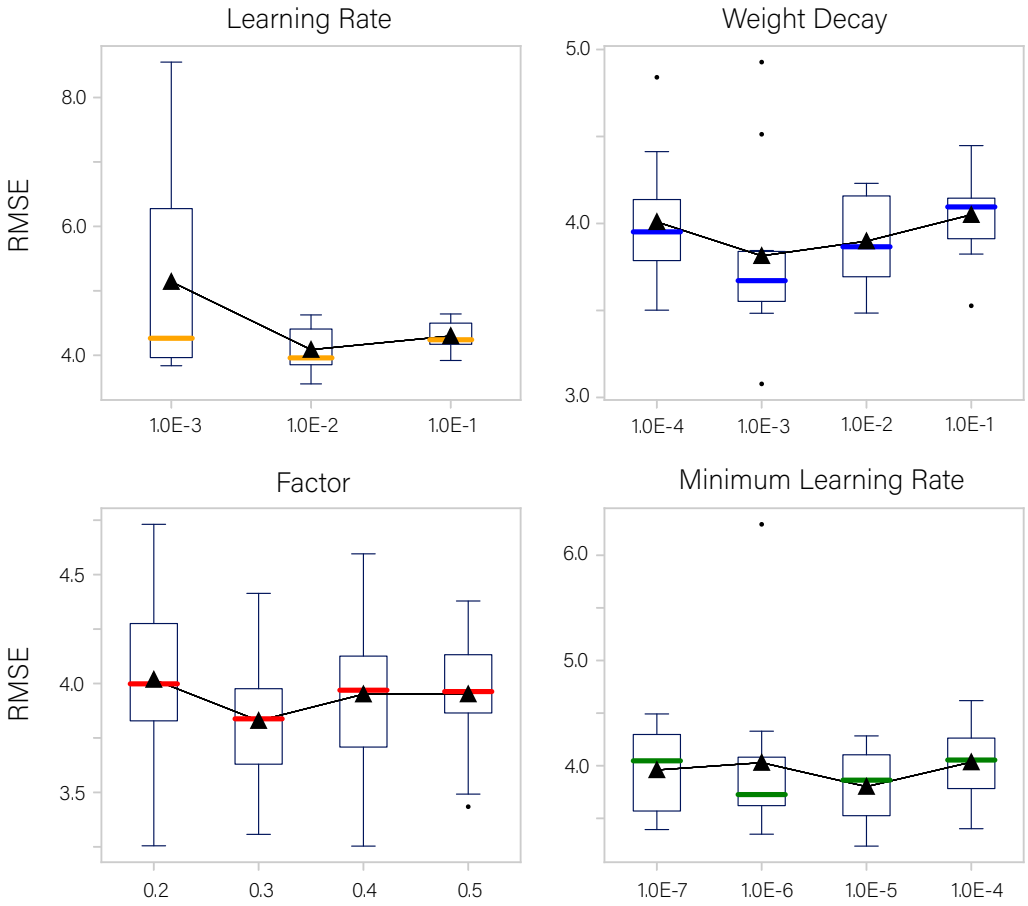

FIG. 3: Hyperparameter performance measured in terms of root mean square error (RMSE) on the test data. Each box plot displays the corresponding median (colored line) and mean (triangle point) values calculated from 10 training sessions.

TABLE 3: Learning rate and epoch values at each training Phase.

Phase	Learning rate	Epochs
1	$1e-02$	200
2	$0.5e-02$	180
3	$0.5e-03$	160

4.2 CNN Training on Multi-Model Data

We first compare the accuracy and computational cost of a CNN trained on multi-fidelity data with one trained exclusively on HF data. Additionally, we examine how different ratios of HF-LF data affect the performance of CNN when using a multi-fidelity training strategy. This analysis

is conducted for three fixed training data generation budgets: 10 hours (Figure 4 (a)), 15 hours (Figure 4 (b)), and 20 hours (Figure 4 (c)). We repeat the multi-fidelity training 10 times for each HF-LF ratio.

Using data from models with different levels of reliability can offer benefits in terms of accuracy and computational efficiency. For each scenario, the minimum RMSE value is achieved with a 1:5 ratio of HF-LF data. This value of RMSE is lower than the one achieved by training the network with varying numbers of HF data, corresponding to 240, 360, and 480, generated in 10, 15, and 20 hours, respectively, with the 1D model. Moreover, this ratio falls within the range of 1.5 to 5.5, as recommended in a theoretical multilevel Monte Carlo study (Taverniers et al., 2020), and aligns with the findings of Song and Tartakovsky (2022), which applied TL to multi-fidelity data generated by solving PDEs on both fine and coarse meshes of the same physical model. In terms of computational cost, it's important to note that generating HF data is four times more expensive than generating LF data in this specific case.

To test the prediction accuracy of the CNN on unseen HF data, we analyze the RMSE and the scatter plot of predicted versus true pressure values, i.e. computed with the 1D model. Figure 5 illustrates the scatterplot of a test case, obtained with the best CNN surrogate model among the ten associated with a data generation budget of 15 hours, and a HF-LF ratio of 1:5 (see Figure 4 b). We may observe that the CNN accurately approximates the pressure along the aorta throughout the entire time frame associated with a cardiac impulse, with the regression line sensibly close to the bisector and $R^2 = 0.95$.

Finally, using the Monte Carlo method we generate 1300 scenarios sampled in the space of variability of the input parameters. We then compare the ensemble averages of the pressure provided by the best CNN surrogate model trained with a ratio HF-LF of 1:5 and a training data generation budget of 15 hours to those provided by the HF model (1D model). The ensemble averages are computed at a given time corresponding to the peak of the cardiac cycle and at each finite element of the different branches. Figure 6 shows this comparison highlighting the results for each branch of the cardiovascular geometry. In terms of computational advantage of using the CNN surrogate model compared to the numerical 1D model, the evaluation of the CNN model

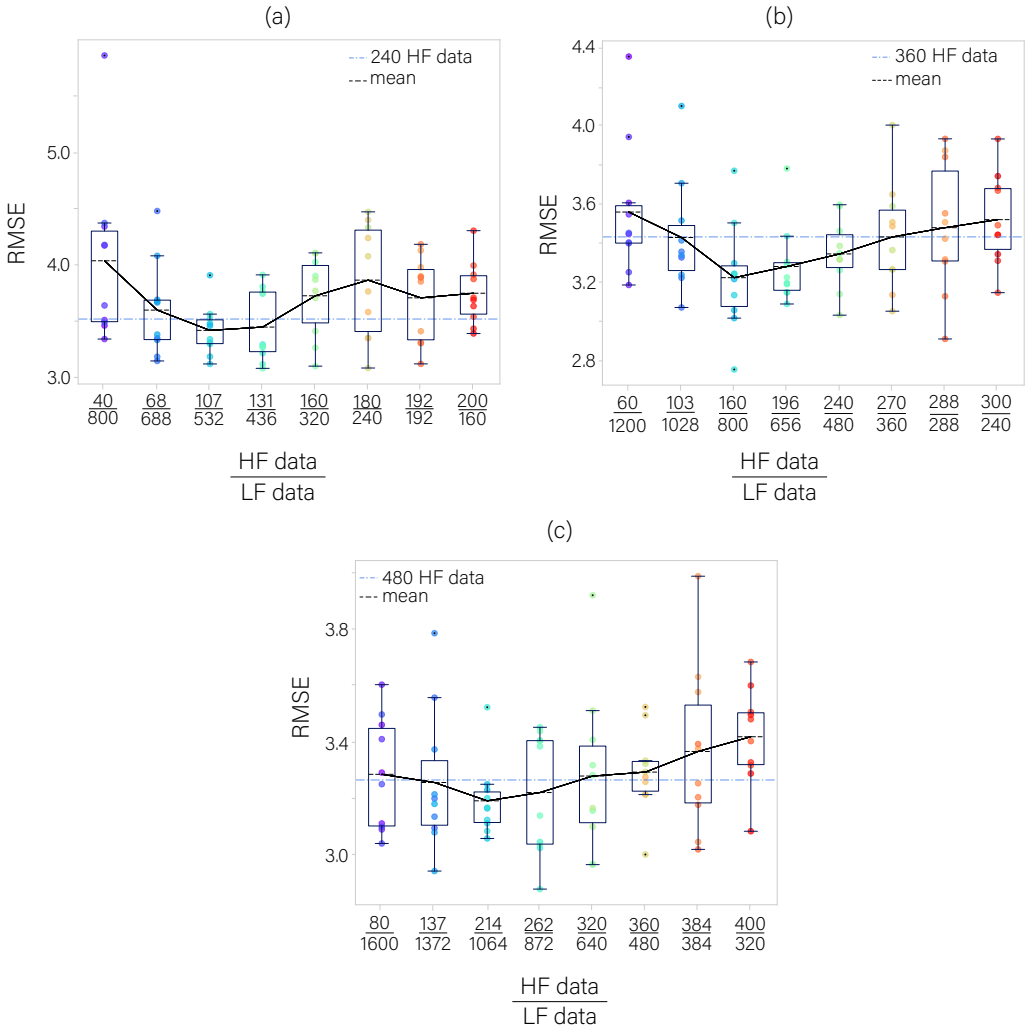

FIG. 4: Influence of different ratios of HF and LF data for fixed data generation budgets of (a) 10, (b) 15, and (c) 20 hours, on the accuracy of the CNN. Dashed lines represent the mean RMSE value obtained by training the CNN using only HF data.

over the 1300 test samples took 3.5 min, while the execution of the HF model spent 54 hours, which gives a speedup close to 1000x. The computations are carried out utilizing an Intel(R) Core(TM) i7-9700 CPU (3GHz) processor, with 32GB RAM.

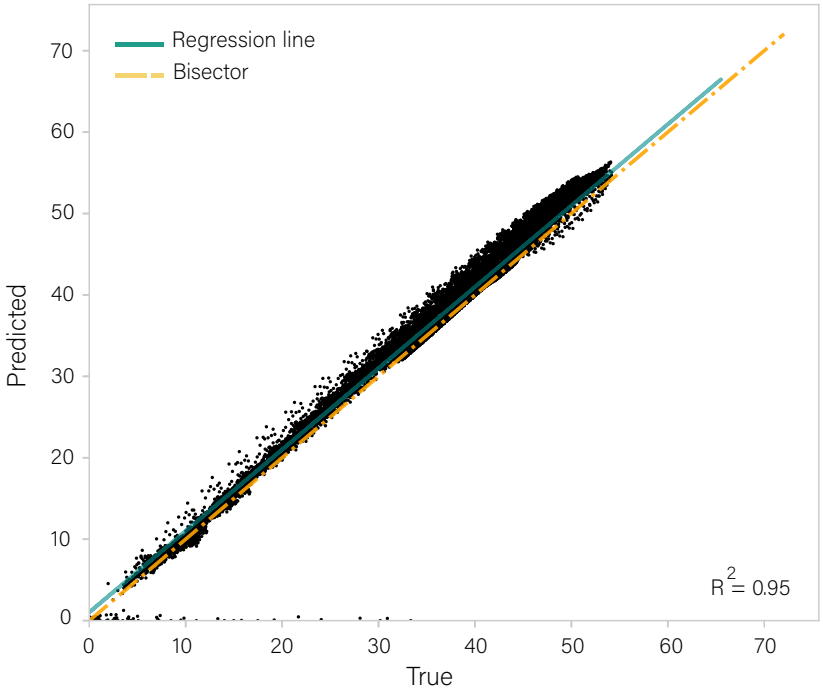

FIG. 5: Scatter plot of true versus predicted values of the pressure [mmHg], including the regression line. The results are based on a test case derived from one of the 10 iterations associated with the optimal ration 1:5 of HF and LF data during training, with a data generation budget of 15 hours.

5. CONCLUSIONS

In this work, we demonstrated how using TL on data generated by physical models with varying levels of complexity and reliability can enhance both the accuracy and computational efficiency of a CNN surrogate. This approach effectively reduces the cost associated with data generation.

We applied our approach to the prediction of pressure along a specific vascular geometry with variable parameters. For our computational example, we considered the 1D model as "high-fidelity model", where the solutions are derived by integrating the three-dimensional Navier-Stockes equations over the vessel cross-section line, while the "low-fidelity model" is given by the 0D formulation, where the cardiovascular system is analogous to an electrical circuit and the quantities of interest are purely time-dependent.

Our results demonstrate that the accuracy of the predictions made by the CNN surrogate is comparable to that of the high-fidelity model. However, it is crucial to emphasize that the

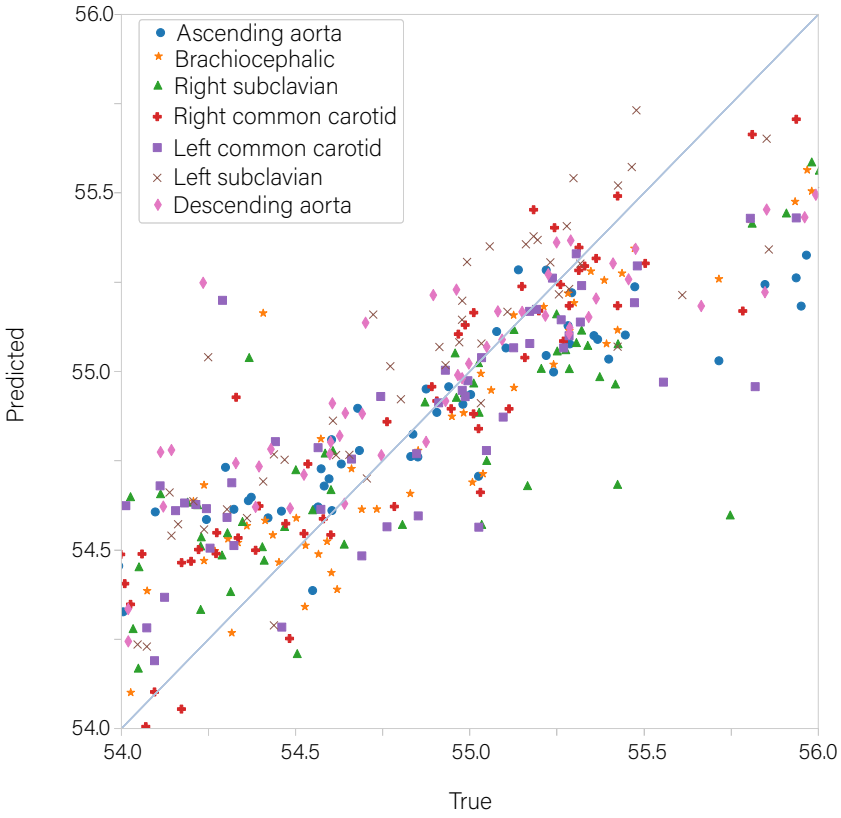

FIG. 6: Comparison of the ensemble averages of the pressure [mmHg] provided by the best CNN surrogate for a training data generation budget of 15 hours to those provided by the HF model at a given time corresponding to the peak of the cardiac cycle and at each finite element of the different branches (depicted with different colors and markers) over 1300 Monte Carlo simulations.

computational efficiency and accuracy of our multi-fidelity simulations significantly depend on the ratio of high-fidelity to low-fidelity data. In our application, the optimal performance was achieved with a 1:5 ratio, leading to a 75% global computational gain in data generation, which is in line with the range recommended in a theoretical multilevel Monte Carlo study (Taverniers et al., 2020), and aligns with the findings of Song and Tartakovsky (2022).

ACKNOWLEDGMENTS

The data that support the findings of this study are openly available in the repository `www.vascularmodel.com` (Wilson et al., 2013). This study has received funding from the Research Project of National Interest PRIN 2017 "Fluid dynamics of hearts at risk of failure: towards methods for the prediction of disease progression", funded by the Italian Ministry of Education, Universities and Research (MIUR); MIUR code: 2017A889FP_006; CUP code: J34I19001480001. AC has also received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 777822. DMT was funded, in part, by the Air Force Office of Scientific Research under grant FA9550-21-1-0381, by the Office of Advanced Scientific Computing Research (ASCR) within the Department of Energy Office of Science under award number DE-SC0023163, and by the Strategic Environmental Research and Development Program (SERDP) of the Department of Defense under award RC22-3278.

APPENDIX A.

The following table contain the values of the Windkessel parameters for each outlet of the aorta geometry.

TABLE A1: RCR Values for 0009_H_AO_H model in cgs unit.

Face Name	R_p	C	R_d
0 Brachiocephalic	1510.0	6.42e-05	20064.0
1 Right common carotid	1714.0	5.658e-05	22766.0
2 Left common carotid	1836.0	5.658e-05	22644.0
3 Left subclavian	1513.0	6.864e-05	18665.0
4 Descendent aorta	274.0	3.54e-04	3639.0

REFERENCES

Akhavan-Safaei, A. and Zayernouri, M., Deep Learning Modeling for Subgrid-Scale Fluxes in the LES of Scalar Turbulence and Transfer Learning to Other Transport Regimes, *Journal of Machine Learning for Modeling and Computing*, vol. **5**, no. 1, pp. 1–23, 2024.

Barajas-Solano, D.A. and Tartakovsky, D.M., Stochastic Collocation Methods for Nonlinear Parabolic Equations with Random Coefficients, *SIAM/ASA J. Uncert. Quant.*, vol. **4**, no. 1, pp. 475–494, 2016.

Chandra, A., Bakarji, J., and Tartakovsky, D.M., Role of Physics in Physics-Informed Machine Learning, *Journal of Machine Learning for Modeling and Computing*, vol. **5**, no. 1, pp. 85–97, 2024.

Collia, D., Libero, G., Pedrizzetti, G., and Ciriello, V., Surrogate Models Provide New Insights on Metrics based on Blood Flow for the Assessment of Left Ventricular Function, *Scientific Reports*, vol. **12**, no. 1, 2022.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T., DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition, *International Conference on Machine Learning*, vol. **32**, no. 1, pp. 647–655, 2014.

Fleeter, C.M., Geraci, G., Schiavazzi, D.E., Kahn, A.M., and Marsden, A.L., Multilevel and Multifidelity Uncertainty Quantification for Cardiovascular Hemodynamics, *Computer Methods in Applied Mechanics and Engineering*, vol. **365**, p. 113030, 2020.

Geneva, N. and Zabaras, N., Multi-Fidelity Generative Deep Learning Turbulent Flows, *Foundations of Data Science*, vol. **2**, no. 4, pp. 391–428, 2020.

Howard, A., Fu, Y., and Stinis, P., A Multifidelity Approach to Continual Learning for Physical Systems, *Machine Learning: Science and Technology*, vol. **5**, no. 2, p. 025042, 2024.

Karniadakis, G.E., Kevrekidis, I.G., Lu, L., Perdikaris, P., Wang, S., and Yang, L., Physics-Informed Machine Learning, *Nature Reviews Physics*, vol. **3**, no. 6, pp. 422–440, 2021.

Kingma, D.P. and Ba, J., Adam: A Method for Stochastic Optimization, *arXiv*, 2014.

Krawczuk, S. and Venturi, D., Improving Neural Network Predictions of Material Properties with Limited Data using Transfer Learning, *Journal of Machine Learning for Modeling and Computing*, vol. **2**, no. 1, pp. 31–47, 2021.

Lan, H., Updegrove, A., Wilson, N.M., Maher, G.D., Shadden, S.C., and Marsden, A.L., A Re-Engineered Software Interface and Workflow for the Open-Source SimVascular Cardiovascular Modeling Package, *Journal of Biomechanical Engineering*, vol. **140**, no. 2, 2018.

Li, Z. and Arora, S., An Exponential Learning Rate Schedule for Deep Learning, *arXiv*, 2019.

Libero, G., Chiofalo, A., Ciriello, V., and Tartakovsky, D.M., Extended Dynamic Mode Decomposition for Model Reduction in Fluid Dynamics Simulations, *Physics of Fluids*, vol. **36**, no. 6, p. 061911, 2024a.

Libero, G., Tartakovsky, D., and Ciriello, V., Polynomial Chaos Enhanced by Dynamic Mode Decomposition for Order-Reduction of Dynamic Models, *Advances in Water Resources*, vol. **186**, p. 104677, 2024b.

Meng, X. and Karniadakis, G.E., A Composite Neural Network That Learns from Multi-Fidelity Data: Application to Function Approximation and Inverse PDE Problems, *Journal of Computational Physics*, vol. **401**, p. 109020, 2020.

Oladyshkin, S., Praditia, T., Kroeker, I., Mohammadi, F., Nowak, W., and Otte, S., The Deep Arbitrary Polynomial Chaos Neural Network or How Deep Artificial Neural Networks Could Benefit from Data-Driven Homogeneous Chaos Theory, *Neural Networks*, vol. **166**, pp. 85–104, 2023.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S., PyTorch: An Imperative Style, High-Performance Deep Learning Library, *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, Eds., Vol. 32, Curran Associates, Inc., p. 8024–8035, 2019.

Pfaller, M.R., Pham, J., Verma, A., Pegolotti, L., Wilson, N.M., Parker, D.W., Yang, W., and Marsden, A.L., Automated Generation of 0D and 1D Reduced-order Models of Patient-specific Blood Flow, *International Journal for Numerical Methods in Biomedical Engineering*, vol. **38**, no. 10, 2022.

Propp, A. and Tartakovsky, D.M., Transfer Learning on Multi-Dimensional Data: a Novel Approach to Neural Network-Based Surrogate Modeling, *Journal of Machine Learning for Modeling and Computing*, vol. **6**, no. 2, pp. 13–27, 2025.

Shen, C., Appling, A.P., Gentine, P., Bandai, T., Gupta, H., Tartakovsky, A., Baity-Jesi, M., Fenicia, F., Kifer, D., Li, L., Liu, X., Ren, W., Zheng, Y., Harman, C.J., Clark, M., Farthing, M., Feng, D., Kumar, P., Aboelyazeed, D., Rahmani, F., Song, Y., Beck, H.E., Bindas, T., Dwivedi, D., Fang, K., Höge, M., Rackauckas, C., Mohanty, B., Roy, T., Xu, C., and Lawson, K., Differentiable Modelling to Unify Machine Learning and Physical Models for Geosciences, *Nature Reviews Earth & Environment*, vol. **4**, no. 8, pp. 552–567, 2023.

Song, D.H. and Tartakovsky, D.M., Transfer Learning on Multifidelity Data, *Journal of Machine Learning for Modeling and Computing*, vol. **3**, no. 1, pp. 31–47, 2022.

Taverniers, S., Bosma, S.B.M., and Tartakovsky, D.M., Accelerated Multilevel Monte Carlo With Kernel-Based Smoothing and Latinized Stratification, *Water Resources Research*, vol. **56**, no. 9, 2020.

Turisini, M., Cestari, M., and Amati, G., LEONARDO: A Pan-European Pre-Exascale Supercomputer for HPC and AI Applications, *Journal of Large-scale Research Facilities*, vol. **9**, no. 1, 2024.

Updegrove, A., Wilson, N.M., Merkow, J., Lan, H., Marsden, A.L., and Shadden, S.C., SimVascular: An Open Source Pipeline for Cardiovascular Simulation, *Annals of Biomedical Engineering*, vol. **45**, no. 3, pp. 525–541, 2016.

Vignon-Clementel, I., Figueroa, C., Jansen, K., and Taylor, C., Outflow Boundary Conditions for 3D Simulations of Non-Periodic Blood Flow and Pressure Fields in Deformable Arteries, *Computer Methods in Biomechanics and Biomedical Engineering*, vol. **13**, no. 5, pp. 625–640, 2010.

Weiss, K., Khoshgoftaar, T.M., and Wang, D., A Survey of Transfer Learning, *Journal of Big Data*, vol. **3**, no. 1, 2016.

Wilson, N.M., Ortiz, A.K., and Johnson, A.B., The Vascular Model Repository: A Public Resource of Medical Imaging Data and Blood Flow Simulation Results, *Journal of Medical Devices*, vol. **7**, no. 4, 2013.

Xiu, D., Fast Numerical Methods for Stochastic Computations: A Review, *Communications in Computational Physics*, vol. **5**, no. 2-4, p. 242 – 272, 2009.

Xiu, D. and Karniadakis, G.E., The Wiener--Askey Polynomial Chaos for Stochastic Differential Equations, *SIAM Journal on Scientific Computing*, vol. **24**, no. 2, pp. 619–644, 2002.

Xiu, D. and Sherwin, S.J., Parametric Uncertainty Analysis of Pulse Wave Propagation in a Model of a Human Arterial Network, *Journal of Computational Physics*, vol. **226**, no. 2, pp. 1385–1407, 2007.

Zhang, R. and Alemazkoor, N., Multi-Fidelity Machine Learning for Uncertainty Quantification and Optimization, *Journal of Machine Learning for Modeling and Computing*, vol. **5**, no. 4, pp. 77–94, 2024.

View publication stats