An image-based deep learning framework for flow field prediction in arbitrary-sized fibrous microstructures

Jean, Jimmy Gaspard; Broggi, Guillaume; Caglar, Baris

DOI
10.1016/j.compositesa.2025.109337

Publication date
2026

Document Version
Final published version

Published in
Composites Part A: Applied Science and Manufacturing

Citation (APA)
Jean, J. G., Broggi, G., & Caglar, B. (2026). An image-based deep learning framework for flow field prediction in arbitrary-sized fibrous microstructures. *Composites Part A: Applied Science and Manufacturing*, *200*, Article 109337. https://doi.org/10.1016/j.compositesa.2025.109337

Composites: Part A 200 (2026) 109337

Contents lists available at ScienceDirect

Composites Part A

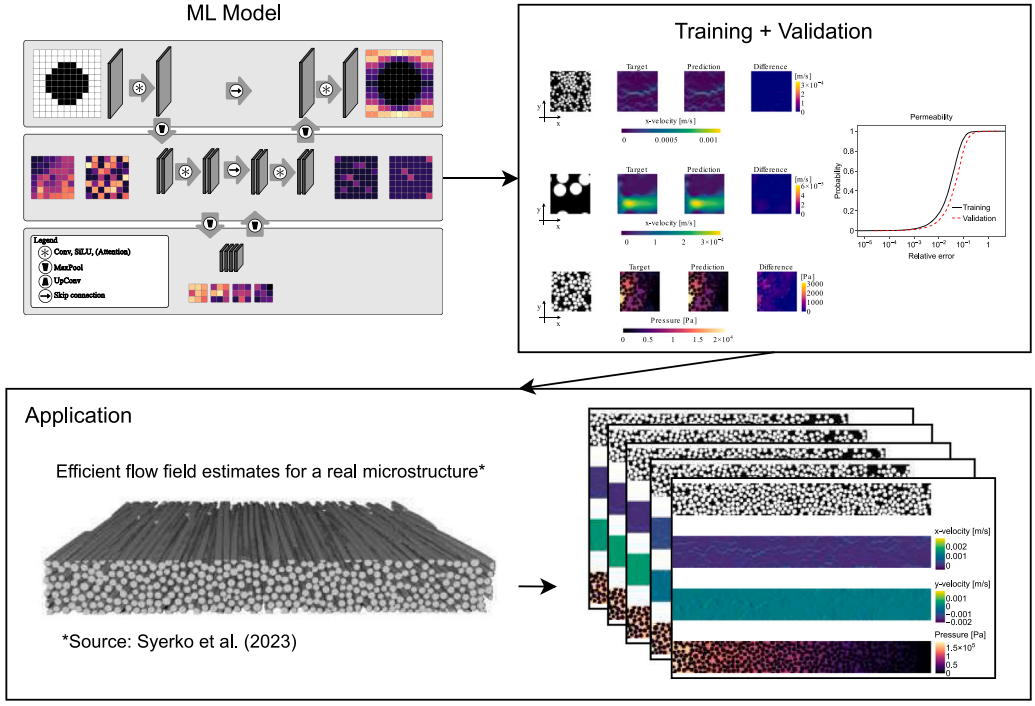
journal homepage: www.elsevier.com/locate/compositesa

An image-based deep learning framework for flow field prediction in arbitrary-sized fibrous microstructures

Jimmy Gaspard Jean, Guillaume Broggi, Baris Caglar *

Aerospace Structures and Materials Department, Faculty of Aerospace Engineering, Delft University of Technology, Kluyverweg 1, Delft, 2629 HS, South-Holland, The Netherlands

GRAPHICAL ABSTRACT



ARTICLE INFO

Dataset link: https://doi.org/10.5281/zenodo.16940478, https://github.com/camp-lab-tud/arbitrary-microstructure-flow

Keywords:
Machine learning
Resin transfer molding
Process modeling

ABSTRACT

Numerical simulations are commonly used to predict resin flow in fibrous reinforcements but exhibit a trade-off between accuracy and computational cost. As an alternative, machine learning (ML) based models pose as a potential tool to accelerate or replace such costly simulations. This work proposes an open-source image-based deep learning framework to estimate the permeability of unidirectional microstructures in arbitrarily sized domains. This presents a scalable step towards estimating the permeability of large meso-domains. First, we present two robust and accurate surrogate models capable of predicting microstructure velocity and pressure fields with varying physical dimensions, fiber diameter, and volume fraction. These predictions achieve 5%

* Corresponding author.
E-mail address: b.caglar@tudelft.nl (B. Caglar).

https://doi.org/10.1016/j.compositesa.2025.109337
Received 16 May 2025; Received in revised form 19 September 2025; Accepted 2 October 2025
Available online 16 October 2025

Permeability

error on the training set and 8% error on unseen microstructures. Secondly, based on those predicted flow fields, we infer the permeability of the microstructures with respectively 4% and 6% deviation for the training and validation sets. Third, opposed to previous works limited to microstructures with a fixed aspect ratio, we propose a so-called *sliding window* procedure, based on physics-based principles to predict the resin velocity and pressure field in microstructures with different aspect ratios. The method is validated against high-fidelity numerical simulations, and its predictive performance and computational efficiency are confirmed with μ-CT scans of real microstructures. Finally, the presented code and surrogate model are open-sourced to promote further exploration by the scientific community.

1. Introduction

Resin transfer molding (RTM) [1] accounts for a significant proportion of industrial fiber-reinforced polymer composites (FRPCs) production since it allows for high-quality parts and short cycle times [2–4]. Although RTM is well-established, the factors influencing part quality, mainly related to process and material variability, are not comprehensively understood. This includes the competition between microscopic and macroscopic flows [5], compaction behavior [6], preform permeability [7], or polymerization [8].

Notably, the dual-scale pore distribution originating from textile architectures can lead to intra-tow microvoids when viscous forces dominate the flow. As the fiber tows have a lower permeability than the space between them, the resin may flow faster in the inter-tow regions and trap air in the intra-tow regions [5,9,10]. Since voids directly influence the mechanical performance of composite parts [11], understanding the micro-, meso-, and macro-permeability is required to optimize processing conditions for part quality and low cycle times.

Numerous experimental methods have been devised to characterize the permeability of FRPCs (see Dei Sommi et al. [12] for an overview). These approaches are often labor-intensive, time-consuming, and wasteful; hence, expensive. Numerical approaches, such as the finite element method (FEM) [13–15], the finite volume method (FVM), or the lattice Boltzmann method [16] have been applied to FRPCs and address some of the drawbacks mentioned earlier, notably by reducing the human effort and the waste. The resin flow is usually modeled at the microscale by a laminar Stokes flow, given the high fluid viscosity and low velocity, resulting in a low Reynolds number. However, as the simulation scale increases, capturing the intricacies of the microscale geometry while maintaining a reasonable computation cost becomes challenging. Therefore, the numerical models typically rely on assumptions and approximations that introduce errors and variability. For instance, Syerko et al. [17] recently carried out an international benchmark exercise to estimate the numerical permeability of a $523 \times 507 \times 65\,\mu m^3$ volume extracted through X-ray computed tomography. A significant scatter, with a coefficient of variation of about 20%, was reported over 50 contributions. Worse, running times remained substantial and ranged from about a couple of hours to more than a hundred hours for individual simulations. This work illustrated the need for more efficient methods.

On the other hand, machine learning (ML) surrogates offer computational efficiency at the inference stage. In addition, neural networks are considered universal function approximators in the sense that there exist neural architectures with appropriate width, depth, or activation functions that can approximate any continuous function in a compact space to any desired degree of accuracy [18,19]. This property makes ML surrogates especially appealing for tackling high-dimensional and complex problems where traditional numerical methods might struggle. Various works have applied ML to porous media in geosciences. They include using 2D and 3D convolutional neural networks (CNNs) to predict bulk properties such as the permeability of rock samples [20,21], or more refined information such as velocity or pressure fields in rocks [22,23].

As FRPCs typically exhibit a highly anisotropic pore network, ML applications have often been restricted to the prediction of homogenized quantities, such as the permeability tensor [24–26], or the macroscale filling pattern [27]. These approaches simplify the complex flow dynamics as they assume homogeneity and cannot provide insight in cases of microstructural inhomogeneity as encountered in yarn boundaries of textiles. They also inadequately reflect boundary effects by not capturing the impact of no-slip boundary conditions on fluid flow near solid surfaces. In contrast, refined details of the evolution of the velocity and pressure field at the microscale enable more accurate flow modeling at larger scales. They can provide a better understanding of the effect of factors such as microstructural variability or high and low fiber volume fraction areas on the manufacturing process.

Therefore, we propose a 2D convolutional neural network, with an attention mechanism, to predict the steady-state resin flow velocity and pressure fields in fibrous microstructures. The U-Net architecture [28], initially proposed for segmentation tasks in biomedical images, inspires this work. U-Net has already been used with success in various deep-learning models designed for computer vision tasks, including image denoising [29], image inpainting, and notably image generation [30]. We designed two separate models to adapt U-Net to our purpose. The first is trained to predict the velocity field in square domains representing fiber-reinforced microstructures. The second is optimized to predict the pressure field. As will be detailed in subsequent sections, the obtained surrogates capture microscale flow features with satisfactory accuracy. Additionally, the results can be aggregated as flow rate and a pressure drop to evaluate the domain permeability by applying Darcy's law [31]. This approach outperforms known implementations of ML for predicting permeability in FRPCs.

Previous approaches trained neural networks to predict 2D flow fields over square domains or 3D fields in cuboid domains [22,23]. At inference time, they also restricted their model application to such types of domains. We go beyond that restriction by extending the use of the trained models to larger rectangular domains. We assume that the flow phenomenon observed on a square domain is guided by the same physics principles as the flow occurring on a larger rectangular domain. Hence, the knowledge learned by the neural network on square domains should be transferable to rectangular domains, as the typical images or micrographs used for characterization purposes have rectangular shapes instead of square ones. We took inspiration from the *sliding window* technique [32], typically used in computer science to write efficient algorithms and process sequential data, to predict the velocity and pressure fields of rectangular microstructures.

To some extent, one may regard any rectangular domain as a sequence of (potentially overlapping) square domains, given that the flow occurs in laminar conditions. The method is implemented with physics-based constraints to ensure conservation of mass and pressure continuity. Herein, we validate the proposed algorithm through comparison with numerical simulations on rectangular domains with high aspect ratios, and further assess its prediction accuracy as well as the computation time through comparison with the results reported in the permeability exercise [17] using the computed tomography images of a unidirectional microstructure.

To sum up, this work's main contributions are (i) proposition of two machine learning models whose loss functions and neural network architectures are tailored to and optimized for predicting the resin velocity and pressure field in square fiber-reinforced microstructures, and (ii) introduction and validation of a *sliding window* technique for both of the flow and pressure fields in rectangular domains. These are further explained in the rest of the manuscript, which is structured

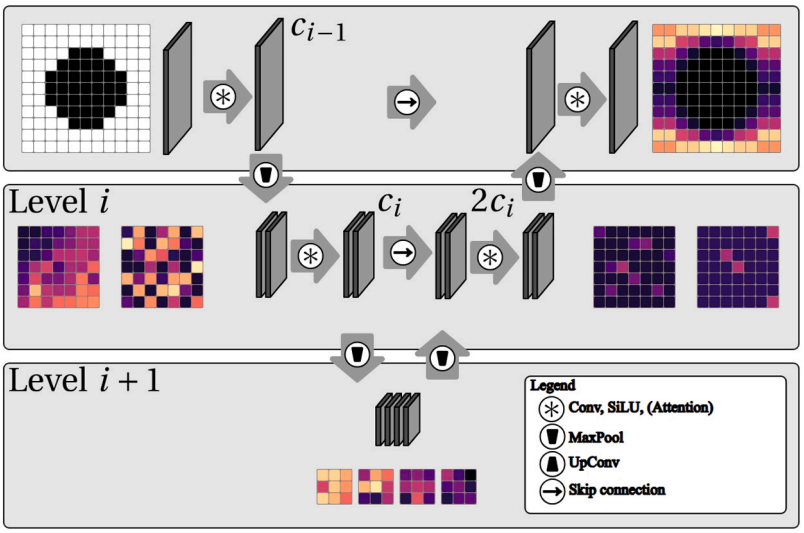


Fig. 1. A visual representation of the U-Net architecture. Individual blocks are symbolized to highlight the information flow for inferring the flow field (top right) around a black circular fiber (top left).

as follows. Section 2 provides details of the baseline U-Net model's architecture and the added attention mechanism. Section 3 explains the microstructure generation and flow simulations performed in `OpenFOAM` to generate the database, while Section 4 describes the training process of the models. Section 5 and Section 6 present the results for square and rectangular domains, respectively, while detailing the *sliding window* method, and Section 7 compares the results with the recent benchmark exercise.

2. Model architecture

The U-Net architecture is an image-to-image network that leverages an encoding-decoding approach, referred to by the U symbol. The input data, e.g., a biological tissue image in the original U-Net paper [28], is encoded into a lower-dimensional latent space. The latent space is then decoded through a symmetric process to reconstruct the desired output, e.g., segmented cells in the tissue image. The model is trained following an end-to-end supervised learning procedure to yield accurate predictions. In this work, we implement a similar architecture adapted to perform a pixel-wise regression on the velocity and pressure fields in a 2D fibrous microstructure subjected to a viscous flow, as highlighted in Fig. 1.

2.1. Building blocks

The building blocks of our U-Net are (i) *convolution blocks*, (ii) *attention blocks*, (iii) *down-sampling blocks*, (iv) *up-sampling blocks*, and (v) *skip connections*. They are combined to form the *encoder* and *decoder* part of the model. We detail hereafter these building blocks.

(i) **Convolution block**: Each convolution block is a sequence consisting of a convolution layer [33], a group normalization [34] layer, and a Sigmoid Linear Unit (SiLU) activation layer [35]. The convolution layer is the core building block of any convolutional neural network. It applies a convolution operation to its input through weight matrices, known as *kernels*. Limiting ourselves to square kernels, the size $k \times k$ of each determines the extent of the field it captures. Its *stride* s determines the number of pixels by which the kernel shifts during the convolution operations. We use a stride $s = 1$ in these convolution layers. Meanwhile, the number of individual kernels determines the number of separate channels resulting from the convolution.
We utilize group normalization instead of batch normalization, in contrast to the original U-Net paper [28]. Group normalization scales values across the feature dimension, unlike batch normalization, which does so across the batch dimension. This characteristic makes group training independent of batch size [34] as well as more stable. Moreover, contrary to classification problems as in [28], the absolute scale of features is especially critical in regression problems because the model predicts continuous values directly tied to the input data's scale. This makes group normalization all the more suitable and also explains why it is preferred in the state-of-the-art diffusion models [30]. We set the number of groups equal to 1, effectively making the group normalization equal to layer normalization [36]. Finally, a SiLU activation layer follows normalization. SiLU is defined as

$$\mathrm{SiLU}(x) = x * \sigma(x), \text{ where } \sigma(x) = \frac{1}{1+e^{-x}} \tag{1}$$

This smooth and non-monotonic activation function can capture more complex patterns and relationships in data compared to, for instance, Leaky ReLU [37,38], which is monotonic and less expressive.
(ii) **Attention block**: Attention mechanism in modern machine learning was first introduced in a recurrent neural network architecture [39] for machine translation. It later gained popularity with the transformer architecture [40] and has also been included in diffusion models [29]. We utilize an attention block composed of layer normalization [36] followed by a multi-headed attention layer [40]. To be more specific, we use 8 attention heads. We perform self-attention in the sense that the *queries*, *keys*, and *values* passed to the multi-headed attention layer are identical. The *attention values* computed through the latter layer are added to its initial input to yield the output of the attention block.
(iii) **Down-sampling block**: The purpose of the down-sampling block is to reduce the size of its input arrays and thereby gain more global insights. This is usually attained via a pooling operation that aggregates information into a reduced dimension. In our case, the down-sampling block consists first of a max-pooling layer [33]. It is implemented with a 2×2 kernel and a stride $s = 2$. This choice reduces the size of the arrays by half. We then follow the max-pooling layer with group normalization and a SiLU activation layer.
(iv) **Up-sampling block**: The up-sampling block accomplishes the opposite of the down-sampling block by expanding information into higher dimensions. In the simplest form, up-sampling can be attained via nearest neighbor or bilinear interpolation. We instead opt for a transposed convolutional layer [41] because of its greater flexibility with its trainable parameters. This layer is implemented with a 2×2 kernel and a stride $s = 2$, a choice that doubles the size of the input arrays. It is also followed by group normalization and a SiLU activation layer.

(v) **Skip connections**: Skip connections are devised to facilitate the training of deep neural network architectures. They mitigate the issue of vanishing gradients in deep model architectures by providing alternative paths for gradients to flow during back-propagation. Their use also results in deep models that are easier to optimize because of well-conditioned gradients [42].

2.2. Encoder

The contracting path of the model or *encoder* consists of a sequence of convolution blocks and down-sampling blocks. By reference to Fig. 1, we may differentiate between various *levels* of the encoder based on the height and width of the arrays. Focusing on a particular level i, there are two convolution blocks. The first convolution block changes the number of channels from c_{i-1} (that of the previous level) to c_i (that of the current level). An exception is the starting level $i = 0$, in which the convolution block maps the number of channels from the training data to c_0. The second convolution block keeps the number of channels at c_i.

We further enhanced the two models (one for velocity and one for pressure) by adding an attention block after the two convolution blocks at specific *levels* (See Supplementary Material, Table 1). Adding the attention block in the model designed for velocity prediction showed that its inclusion improved accuracy, thus all results presented herein are generated with the model that included the attention. The same exercise of adding an attention mechanism did not yield a significant improvement on the model trained for pressure prediction. This remark is, however, not conclusive; a different implementation of the attention block might show otherwise.

Finally, a down-sampling block reduces the height and width of the arrays by half for the following level $i + 1$.

2.3. Decoder

The expanding path of the model or *decoder* is constituted by upsampling blocks and convolution blocks. At each level i, we use two convolution blocks. The first convolution block maps the number of channels from $2c_i$ to c_i. The input to this block consists of the concatenation of two parts. One half originates from the lower level i+1. This data is provided by an up-sampling block that doubles the height and width of the arrays from that lower level. The other half is provided by a skip connection, which allows access to the output of the last convolution block (or attention block) from the same level i on the decoder side. The second convolution block keeps the number of channels at c_i. An attention block optionally follows afterward. Upon reaching the decoder's end at level $i = 0$, a final convolution maps the number of channels from c_0 to the number of channels in the target training data.

3. Dataset

3.1. Microstructures

To generate microstructures mimicking the transverse sections of unidirectional fiber bundles, we considered a parameter space determined by three variables: (i) the microstructure **domain size** $D \in \{x = 50 + 25k \mid k = 0, 1, \dots, 6\}$ μm, (ii) the *nominal* **fiber diameter** $d_f \in \{7, 10, 15\}$ μm, (iii) the total **fiber volume fraction** $v_f \in \{x = 0.2 + 0.1k \mid k = 0, 1, \dots, 4\}$. These variables result in a parameter grid space containing $7 \times 3 \times 5$ points, totaling 105 different combinations.

The process to generate a microstructure defined by a point in that space is as follows. We first lay out a square domain based on the domain size D. Then, fibers are randomly generated until the target volume fraction v_f is reached. The individual fiber diameters are assigned by assuming a normal distribution centered at a *nominal* fiber diameter d_f with a coefficient of variation of 5%, a value typical of what is reported in the literature [43]. The fibers are then arranged within the domain using a rigid-body simulator [44] that ensures no overlap. Additionally, we enforce periodicity on all four borders of the microstructure domain. There exists an uncountably large number of fiber configurations based on the same descriptors. We choose to generate 50 microstructures per point in the parameter space, resulting in $105 \times 50 = 5250$ microstructures. Fig. 2 shows examples of such microstructures. We additionally provide in the Supplementary Material (Fig. A1) *post-factum* statistics about the fiber volume fraction in the microstructures. With the topology of the microstructures thus defined, we then discretized each into a mesh, excluding the fiber regions because of our subsequent step of modeling the resin flow around the fibers as a fluid flow in a porous medium.

3.2. Flow simulations

Our goal is to solve the transverse flow in fibrous microstructures. This problem requires solving the Navier–Stokes equations [45], enforcing mass continuity (Eq. (2)) and momentum conservation (Eq. (3)).

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \boldsymbol{u}) = 0 \tag{2}$$

$$\rho\left(\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u}\right) = -\nabla p + \nabla \cdot \boldsymbol{\tau} + \boldsymbol{f} \tag{3}$$

Where ρ denotes the fluid density, and $\boldsymbol{u}$ the fluid velocity, p the fluid pressure, and $\boldsymbol{f}$ body forces. $\boldsymbol{\tau}$ denotes the deviatoric stress tensor. Assuming an incompressible steady-state flow, and considering body forces on the fluid to be negligible, the Navier–Stokes equations are simplified to Eqs. (4) and (5).

$$\nabla \cdot \boldsymbol{u} = 0 \tag{4}$$

$$\rho(\boldsymbol{u} \cdot \nabla \boldsymbol{u}) = -\nabla p + \mu \nabla^2 \boldsymbol{u} \tag{5}$$

Where μ is the fluid viscosity. To solve these equations, we selected `OpenFOAM` [46], a well-known open-source software for computational fluid dynamics that is based on the finite volume method. Within it, we utilized `simpleFoam`, a steady-state solver for incompressible flow that employs the SIMPLE (Semi-Implicit Method for Pressure Linked Equations) algorithm [47].

We assumed the fluid flowing between the fibers to be epoxy resin having a density ρ of $1250 \times 10^3\,\mathrm{kg \cdot m^{-3}}$ and dynamic viscosity μ of $5 \times 10^{-1}\,\mathrm{Pa \cdot s}$. The flow is considered to be velocity-driven. The inlet velocity is set to $\boldsymbol{u}_{inlet} = (1 \times 10^{-4}, 0)\,\mathrm{m\ s^{-1}}$ and is perpendicular to the inlet boundary, ensuring laminar flow conditions. The outlet pressure is set to 0 Pa. The no-slip condition is enforced at the fiber boundaries. Taking advantage of the periodicity of the microstructures, cyclic boundary conditions are enforced on the remaining sides (Supplementary Material, Fig. A3). For each microstructure, we performed one simulation with flow in the horizontal direction, and the other with flow in the vertical direction (see Supplementary Material), resulting in 10 500 results. More information about the approach used to generate the microstructures and simulations can be found in [48]. This includes a validation of the numerical framework comparing the resulting permeability in microstructures with perfect quadratic fiber distribution (estimated through Darcy's law [31]) against Gebart's equations [49].

3.3. Post-processing

We post-processed the simulation data into equivalent multi-dimensional arrays. Each sample in the dataset $D = \{(\mathbf{M}^s, \mathbf{U}^s, \mathbf{P}^s) \mid s = 1, \dots, N\}$ is a triplet consisting of the microstructure tensor $\mathbf{M}^s \in \mathbb{R}^{1\times h\times w}$, the velocity tensor $\mathbf{U}^s \in \mathbb{R}^{2\times h\times w}$, and the pressure tensor $\mathbf{P}^s \in \mathbb{R}^{1\times h\times w}$. N refers to the number of samples in the dataset, while h, w refer to the height and width of the images expected by the neural network model (in our particular case, $h = w = 256$). $\mathbf{M}^s$ is a binary array with 1 in fluid regions and 0 otherwise. $\mathbf{U}^s$ stores the x and y

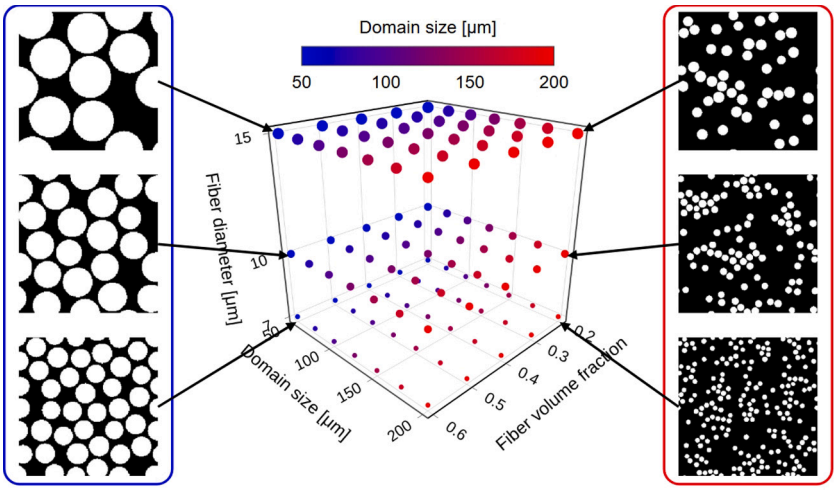


Fig. 2. Sample space used to generate the microstructures. They vary in domain size, fiber diameter, and fiber volume fraction.

Table 1
Features in the input channels and predictions at the output channels for each model.

	Input channels	Output channels
Velocity model	• microstructure	• x-velocity • y-velocity
Pressure model	• microstructure ×fiber volume fraction • 1/microstructure length	• pressure

components of the velocity field in two different channels. A single channel stores the pressure values in $\mathbf{P}^s$.

We note that we later train the model with the assumption that the inlet is located at the left border, i.e., the main flow direction is horizontal. Hence, before using simulation results with the inlet located at the bottom (Supplementary Material, Fig. A2b), we performed a 90°-clockwise *rotation* of the arrays. In the case of the velocity field, this *rotation* additionally entailed ensuring consistency of the channel order and value signs. The entire process, from geometry definition to numerical simulations and post-processing, was fully automated through a `Python` [50] script.

4. Training

4.1. Model definition

We designed two decoupled surrogate models, which we refer to as *velocity model* and *pressure model*, aimed at predicting the velocity and pressure fields, respectively. We decoupled velocity and pressure for two reasons. First, the proposed surrogate architecture does not inherently capture the relationship between these fields. While it could be modified to do so, state-of-the-art ML models in fluid dynamics [51,52] typically do not enforce a physics-informed coupling between the fields. Given the distinct characteristics of velocity and pressure, specialized decoupled surrogates are expected to perform better. Second, decoupling reduces the memory footprint during training, enabling deeper networks within the available computational budget. The *velocity model* takes as input the binary image of microstructures and predicts the corresponding velocity field (Table 1). The *pressure model* has two input channels. The first channel expects the microstructure image multiplied by its fiber volume fraction. The second channel expects a constant matrix whose value is the inverse of the microstructure length (Table 1). The *pressure model* outputs a single channel corresponding to the pressure field. The input features previously mentioned were strategically chosen after considering the velocity-driven nature of the simulations in conjunction with insights gained from Darcy's law [31].

A hyperparameter optimization revealed that a model with a depth of 7 and a kernel size of 3 is a good choice for our case. Further tests showed that using attention mechanisms was beneficial for velocity prediction but unnecessary for pressure prediction. This lack of need for attention in the *pressure model* is, however, inconclusive; it could be due to our current implementation of attention. We provide an extended explanation of model design choices in the Supplementary Material (Section B). These hyperparameter choices resulted in the *velocity model* and *pressure model* having respectively 5.53×10^8 and 4.97×10^8 trainable parameters. These numbers might seem high, but we note, for reference, that the U-Net in the popular text-to-image Stable Diffusion model has 8.60×10^8 parameters [30,53].

4.2. Cost function

Training a machine learning model is achieved by fine-tuning its trainable parameters $\boldsymbol{\theta}$, typically through gradient descent [54]. Gradient descent requires a cost function quantifying the deviations of the model predictions from specific targets. The goal of gradient descent is then to minimize this cost. In our case, we want to minimize the discrepancy between the surrogate model (velocity or pressure) field predictions $\tilde{\boldsymbol{Y}}$ and reference values $\boldsymbol{Y}$ in the dataset. Let $\mathrm{vec} : \mathbb{R}^{c \times h \times w} \to \mathbb{R}^{chw}$ be a function that collapses a three-dimensional tensor into a vector. We define the (relative) error ϵ^s of a sample s as Eq. (6).

$$\epsilon^s = \frac{\|\mathrm{vec}(\boldsymbol{Y}^s - \tilde{\boldsymbol{Y}}^s)\|_1}{\|\mathrm{vec}(\boldsymbol{Y}^s)\|_1} \tag{6}$$

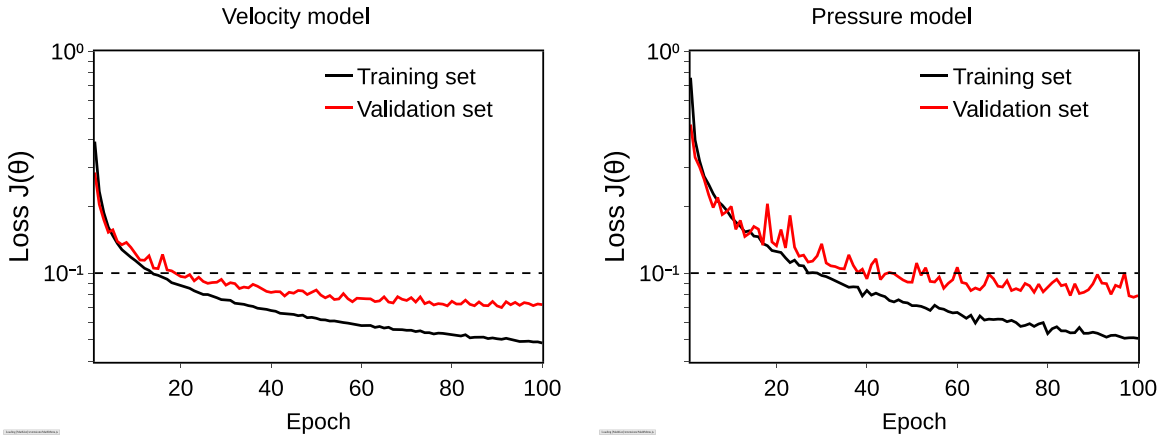
where $\|...\|_1$ denotes the L^1-norm of a vector. We then introduce Eq. (7) as the cost function $J(\boldsymbol{\theta})$ which averages the individual sample errors.

$$J(\boldsymbol{\theta}) = \frac{1}{N} \sum_{s=1}^{N} \epsilon^s \tag{7}$$

where N stands for the number of data samples.

4.3. Model training

We utilized the `PyTorch` library [55] for model implementation and leveraged its automatic differentiation capabilities for model training. Our compute platform is a workstation equipped with an NVIDIA® Quadro RTX™ 6000 graphics processing unit (GPU), with 24 GB of memory and featuring 16.3 TFLOPS of single-precision floating point performance.



(a) Average loss evolution for the velocity model (b) Average loss evolution for the pressure model

Fig. 3. Learning curve of the best surrogate models trained for (a) velocity and (b) pressure field prediction.

Table 2
Five-fold cross-validation: the accuracy of the models is similar across different splits of the dataset.

	Velocity (training/validation)	**Pressure** (training/validation)
Fold 1	$4.86 \times 10^{-2}/6.94 \times 10^{-2}$	$5.22 \times 10^{-2}/9.11 \times 10^{-2}$
Fold 2	$4.81 \times 10^{-2}/7.07 \times 10^{-2}$	$4.76 \times 10^{-2}/8.04 \times 10^{-2}$
Fold 3	$4.77 \times 10^{-2}/6.96 \times 10^{-2}$	$5.07 \times 10^{-2}/8.44 \times 10^{-2}$
Fold 4	$4.94 \times 10^{-2}/7.34 \times 10^{-2}$	$4.94 \times 10^{-2}/8.28 \times 10^{-2}$
Fold 5	$4.95 \times 10^{-2}/7.07 \times 10^{-2}$	$5.27 \times 10^{-2}/8.41 \times 10^{-2}$
Average	$4.87 \times 10^{-2}/7.08 \times 10^{-2}$	$5.05 \times 10^{-2}/8.46 \times 10^{-2}$

Before training, we partitioned the dataset into two parts: 80% used as the training set and 20% used as the validation set. Training was continued for 100 epochs for both the *velocity* and *pressure models*, using a learning rate of 1×10^{-4} and a batch size equal to 20. The training of both models lasted approximately 12 h. Fig. 3 shows the evolution of the loss during the training process. The average error (Eq. (7)) on the predicted velocity fields decreased to 4.84×10^{-2} and 7.21×10^{-2} for the training and validation sets, respectively. In the case of pressure, it decreased to 5.08×10^{-2} and 7.92×10^{-2}.

4.4. Cross-validation

The error metrics reported above show a great performance of the pressure and velocity models. However, model hyperparameters can sometimes be specific to the choice of the training set and not offer the same level of performance if the dataset were split differently. To assess the robustness of the models with respect to the data, we performed a cross-validation [56]. The core idea of cross-validation is to partition the dataset into complementary subsets, train the model on some parts, and evaluate it on the remaining parts. The process is repeated several times to get a robust estimate of the model's performance. We leveraged the `scikit-learn` [57] Python package to perform a five-fold cross-validation of the model. In each training instance, four of the five subsets are chosen as the training set. The remaining 20% of the data is then used as the validation set. This is repeated five times. Table 2 shows the resulting training and validation errors after tuning the models over 100 epochs for each fold of the dataset. Different folds result in varying error values. However, the variations are minimal. This error proximity shows that the model hyperparameters are not biased towards a specific subset of the data. The average training/validation loss is $4.87 \times 10^{-2}/7.08 \times 10^{-2}$ for the *velocity model* and $5.05 \times 10^{-2}/8.46 \times 10^{-2}$ for the *pressure model*.

5. Results

5.1. Velocity prediction

We now focus our attention on the *velocity model* (Fig. 3a) and investigate its performance on individual samples. Fig. 4a shows the histogram of individual errors calculated using Eq. (6) in the training and validation sets. The average values were reported earlier in Section 4.3. The median errors are $4.45 \times 10^{-2}/6.93 \times 10^{-2}$ for the training and validation sets. For illustration purposes, we plot the predictions for the median case in the validation set (Fig. 5a), which we recall that the model did not encounter during training. Visually, the predictions for both the x- and y-components of the velocity field are almost indistinguishable. Looking at the field of absolute errors, most error values in the domain are within the low end of the spectrum. Beyond looking accurate, the model predictions should be physically consistent. For our problem, the flow rate between the inlet and outlet should remain constant. Fig. 5b shows the volumetric flow rate Q inferred by taking line integrals of the velocity field along the length of the microstructure. The flow rate evaluated from the model prediction closely follows the reference. We note that the minor fluctuations observed in the reference line are due to pixel discretization.

In addition to the median case, knowing how the model performs at its best and worst is beneficial. This knowledge is especially valuable for uncertainty quantification and helps assess the confidence level to attach to the model when deployed to real-world scenarios. The best and worst cases in the validation set have errors of 1.47×10^{-2} and 2.287×10^{-1}. The corresponding predictions are provided in the Supplementary Material (Section C). A relative error of about 20% sounds bad, but the number itself is misleading. The values in the velocity field span multiple orders of magnitude, but the error metric (Eq. (6)) penalizes them all equally. However, the larger values matter the most when computing the flow rate. Looking at the velocity field

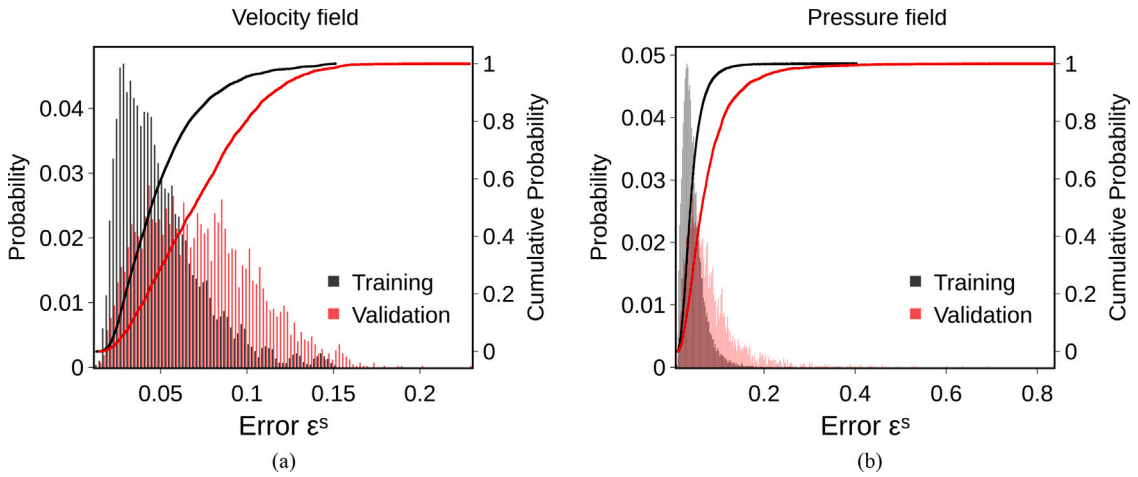


Fig. 4. Histogram of the individual errors in (a) velocity and (b) pressure field prediction for microstructures in the training and validation set.

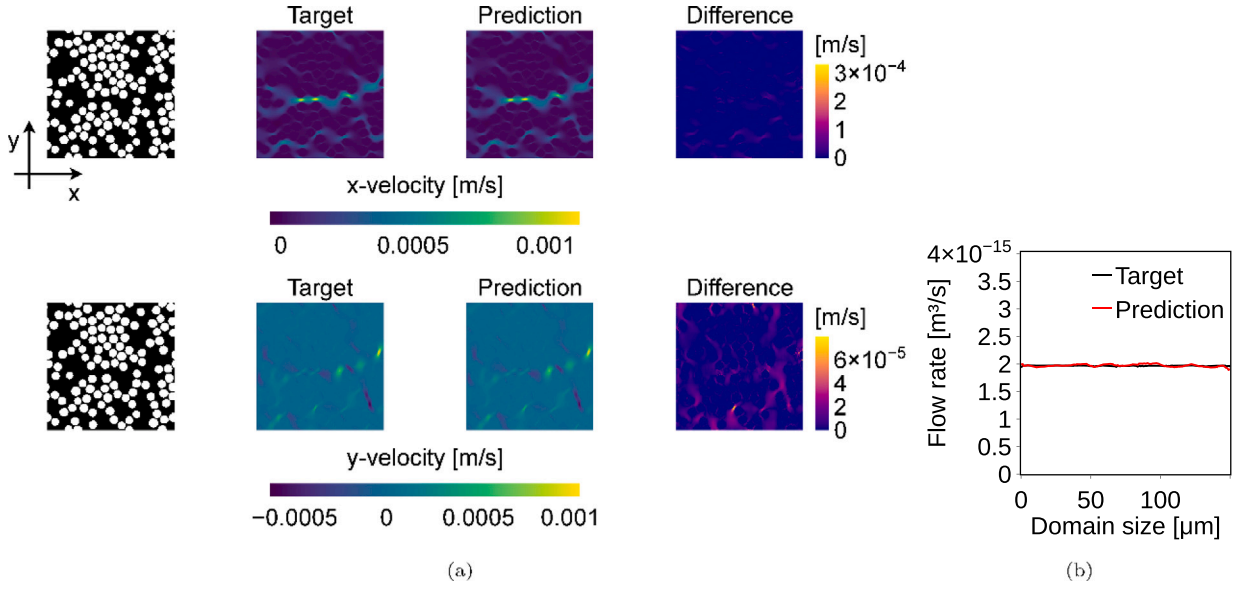


Fig. 5. (a) Resin velocity field for the microstructure with median error in the validation set. (b) Comparison between the calculated average flow rate along the microstructure length against the reference.

prediction for the worst case, the pattern is close to the reference, and the estimated flow rate is just slightly overestimated (Supplementary Material, Fig. C4).

5.2. Pressure prediction

Let us now consider the individual predictions of the *pressure model*. Fig. 4b shows a histogram of the relative errors on the predicted pressure fields. The median cases for the training and validation sets have errors of 3.81×10^{-2} and 6.34×10^{-2}. We plot in Fig. 6 the predicted pressure field for the median case in the validation set. We observe the accuracy of the model in capturing the correct pattern. The plot of the absolute difference is also revealing. The maximum deviation is an order of magnitude lower than the maximum pressure value. We additionally plot the average pressure along the length of the microstructure in Fig. 6b. There, one can notice a close agreement between the two pressure profiles.

Furthermore, we report the best and worst predictions for the model. The lowest and highest errors observed in the validation set are 9.2×10^{-3} and 8.349×10^{-1}. We provide the corresponding pressure fields in the Supplementary Material (Section C). Interestingly, the surrogate model is capable of being 99% accurate with the task of predicting $256 \times 256\,\text{px}^2$. Even in the worst case, the predicted pressure field has an overall pattern similar to the reference. The mistake of the model is in overestimating the pressure values, which are still the correct order of magnitude (Supplementary Material, Fig. C6).

5.3. Transverse permeability

Previous works [24,26] had trained surrogate models to predict microstructure permeability directly. Here, we infer the transverse permeability of the microstructures from the predicted velocity and pressure fields by using Darcy's law [31]. Darcy's law (Eq. (8)) relates the volumetric flow rate Q through a porous medium and the corresponding pressure drop Δp.

$$Q = -\frac{kA}{\mu L}\Delta p \tag{8}$$

where A represents the cross-sectional area of the medium. L is the distance traveled by the fluid with viscosity μ, k denotes the permeability of the medium.

The volumetric flow rate Q is obtained by integrating the velocity field over the mesh at the inlet. The cross-sectional area A is calculated

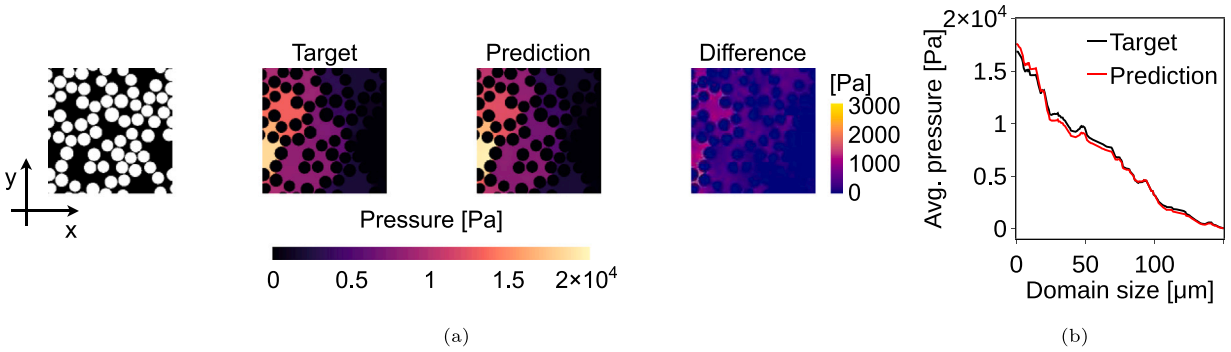


Fig. 6. (a) Pressure field for the microstructure with median error in the validation set. (b) Comparison between the computed average pressure along the microstructure length against the reference.

as the inlet length multiplied by the cell thickness (0.25 μm in the numerical simulations). The distance L is the physical length of the microstructures. Given the boundary condition of outlet pressure equal to 0 in the simulations, the pressure drop Δp is simply the weighted-average pressure at the microstructure inlet. Fig. 7a and Fig. 7b show the histogram of errors on the predicted inlet flow rate and pressure drop for microstructures in the training and validation sets. Using Eq. (8), we then calculated the permeability of all microstructures. The values cover a wide range, and the predictions match closely the targets (Fig. 7c). Fig. 7d shows the cumulative distribution function (CDF) of the relative errors on the predicted permeability values. The average error is only 4.40% on the training set and 6.85% on the validation set. Moreover, 90% of the predictions performed on the training and validation exhibit less than 10% and 14% error, respectively. This is significantly better than the results we reported in our previous work [24] on using machine learning to directly predict the permeability of fibrous microstructures. Here, we achieve not simply better permeability accuracy, but do so with the added benefit of valuable and explainable details about the microstructures' velocity and pressure fields.

6. Extension to larger 2D domains

The surrogate models have performed very well on their intended task of predicting the velocity and pressure fields from $256 \times 256\,\text{px}^2$ microstructure images. However, a restriction to square images induces a practical limitation, as most images (e.g., micrographs of typical woven or non-crimp fabrics) used for flow characterization purposes are not bound to square shapes and typically exhibit high aspect ratios. It is thus worth asking the following question. Is it possible to use the same surrogate models for prediction on microstructures with a high aspect ratio? After all, the flow phenomenon observed on a square domain is the same, in terms of the physical fundamentals, if a bigger, rectangular domain is considered. If the surrogate models appropriately learned the underlying physics of the problem at hand, extrapolation to larger microstructures should be feasible.

We propose a *sliding window* procedure, illustrated in Fig. 8, to extrapolate model predictions to rectangular domains. The *sliding window* technique [32] is an algorithmic approach used to efficiently solve problems in computer science by capturing a subset of the data through a *window*, performing an operation with it, and then repeating the process after moving the window in a stepwise fashion. For our purpose, we consider the size of the moving window to match the one expected by the surrogate models, namely $256 \times 256\,\text{px}^2$. At each step, the subdomain captured by the window is passed to the surrogate models, which then predict the corresponding velocity and pressure fields. This approach immediately raises concerns regarding the violation of the assumptions based on which the models were fine-tuned, such as inlet/outlet conditions or the (lack of) periodicity at the boundaries that merits investigation. In the following sub-sections, we first present a first-order approach to implementing the *sliding window* technique through a moving average of model predictions. This strategy resulted in a couple of issues. We then formalize our problem and provide a scheme that incorporates a weak enforcement [58] of physics principles in the *sliding window* procedure (Algorithm 1).

6.1. First-order approach

Let us consider a $256 \times 1024\,\text{px}^2$ image of a microstructure (physical dimensions: $150 \times 600\,\mu\text{m}^2$). We recall that the maximum physical dimension encountered in our dataset (Fig. 2) is 200 μm. Moreover, the microstructure has a nominal fiber diameter of 10 μm and fiber volume fraction of 0.5. Unlike the training data, the individual fiber diameters follow a half-normal distribution with a minimum equal to the nominal fiber diameter, thus yielding much larger variability in the diameters in the microstructure. The simplest way to make, for instance, velocity predictions on this rectangular domain is to partition it into four $256 \times 256\,\text{px}^2$ subdomains. Starting from one end of the microstructure, one moves the *sliding window* with a step size $\Delta t = 256\,\text{px}$, passes the four subdomains to the surrogate models, and then patches the predictions together. However, this approach led to discontinuities at the patch interfaces when we tried it, for instance, with the *velocity model* (Supplementary Material, Fig. D2). These jumps can be explained by the fact that the actual conditions at the entrance of the subdomains are not that of $\boldsymbol{u}_{inlet} = (1 \times 10^{-4}, 0)\,\text{m s}^{-1}$, as learned from the training data by the surrogate model. The flow at the patch interfaces will have nonzero y-velocity components, while the model assumes zero y-velocity at the inlet.

We then experimented with the use of step sizes $\Delta t < 256\,\text{px}$ for the sliding window, such that overlap occurs between the consecutively captured subdomains. Because of the overlaps, an averaging scheme is needed. We used a simple moving average based on the repetition of a given pixel in different windows. In other words, we calculated the velocity value at a pixel as the average of predictions originating from windows that hovered over this location. This approach smoothed the velocity profile with decreasing Δt, thus mitigating the issue of discontinuity. But different tests showed that this was accompanied by an amplification of errors both in the velocity field and inferred flow rate (see Supplementary Material). The reduction in the step size increases the number of occurrences of each pixel being close to the *inlet* or *outlet* of a window where the model's biases lead to erroneous velocity values. Thus, a physics-aware scheme is needed.

6.2. Problem formulation

A trained neural network model is a mathematical function $\boldsymbol{x} \mapsto \tilde{\boldsymbol{y}}(\boldsymbol{x})$ whose output is most reliable when its input variables fall within the sample space based on which it was trained. Moreover, the inputs must satisfy the assumptions associated with that sample space. For our case, we may formalize these requirements as follows.

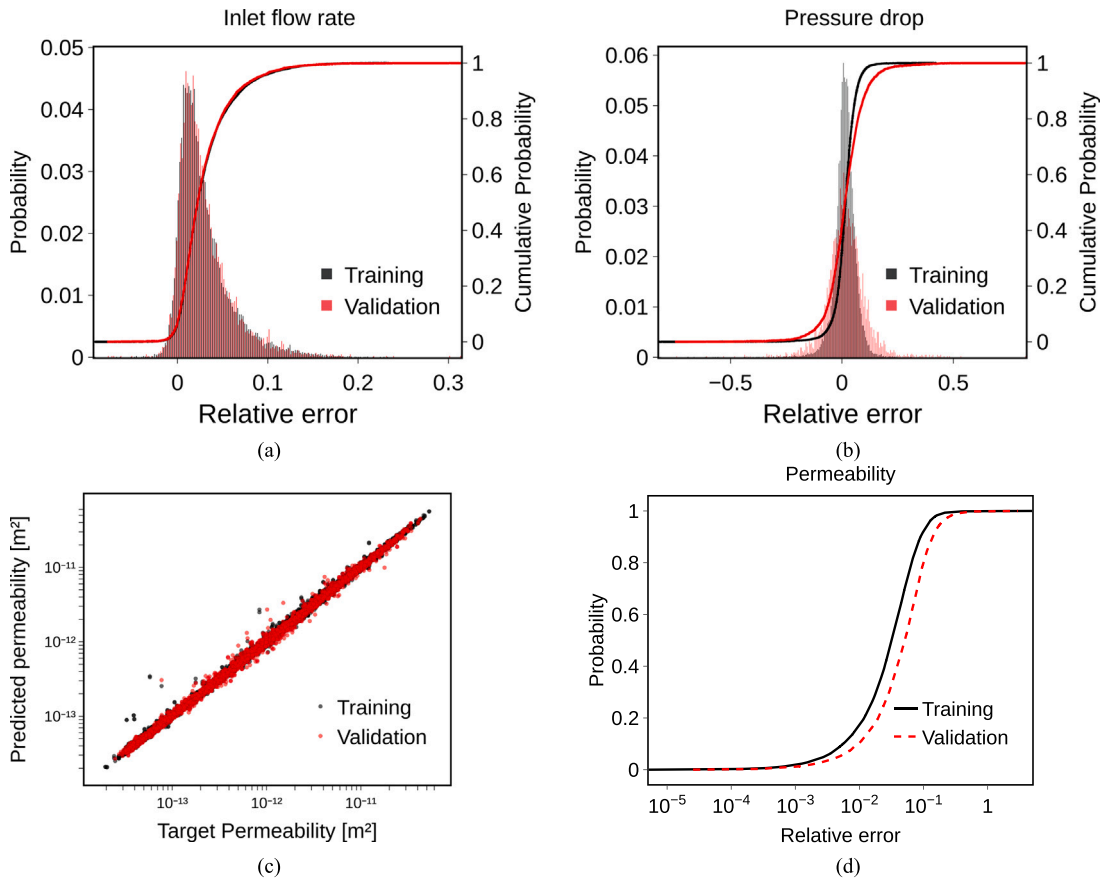


Fig. 7. Histogram of errors for the (a) inlet flow rate and (b) pressure drop. (c) Comparison between permeability predictions using Darcy's law and target values. (d) Plot of the cumulative distribution function of the relative errors on permeability.

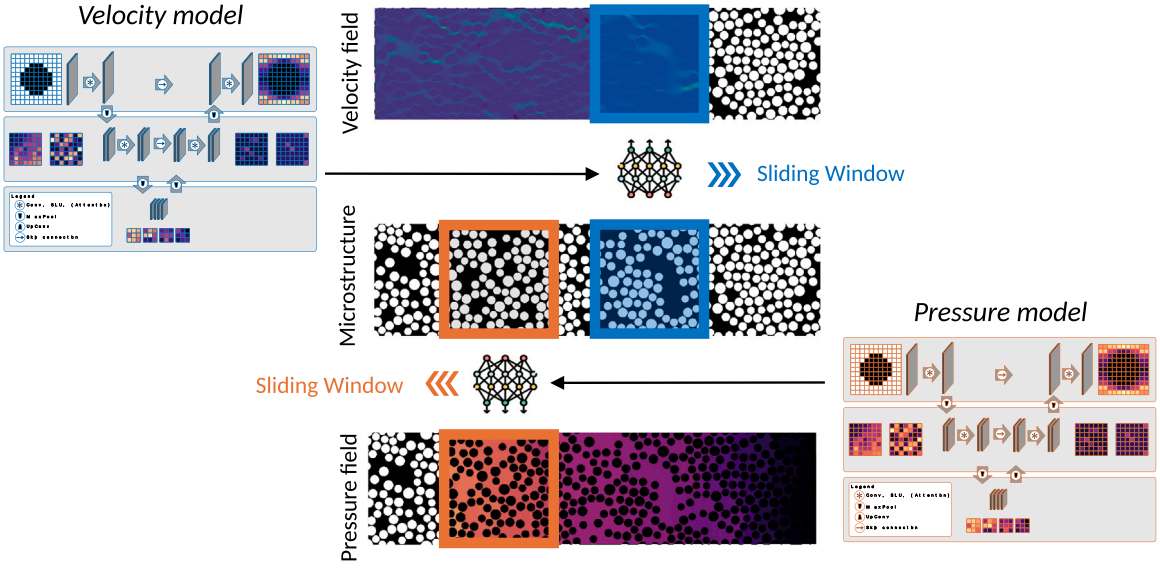


Fig. 8. *Sliding window* procedure for inference on a rectangular domain. The surrogate models make predictions on subdomains captured by a sliding window. Those predictions are then post-processed to yield the final results.

C1. The input microstructure $\boldsymbol{x}$ represented in the square image is periodic.

C2. The flow is velocity-driven, with inlet velocity $\boldsymbol{u}_{inlet} = (1 \times 10^{-4}, 0)\,\mathrm{m\ s^{-1}}$.

C3. The outlet pressure is zero.

Any violation of conditions **C1**, **C2** or **C3** leads to larger uncertainty in the precision of the model predictions. From an *uncertainty quantification* viewpoint, we may relate any model prediction $\tilde{\boldsymbol{y}}(\boldsymbol{x})$ to its true value $\boldsymbol{y}(\boldsymbol{x})$ through the equality

$$\boldsymbol{y}(\boldsymbol{x}) = \tilde{\boldsymbol{y}}(\boldsymbol{x}) + \boldsymbol{\epsilon} \tag{9}$$

Algorithm 1: Sliding Window Procedure

Input : ML model $\mathcal{M}$; microstructure image $\mathbf{I} \in \mathbb{R}^{C\times H\times W}$; sliding window size w; step size s

Output: flow field **F**

1) Fragment microstructure image $\boldsymbol{I}$ into sub-areas $\boldsymbol{a} = \{a_i\}$ using the window size w and step size s. Compute the associated weights $\boldsymbol{w} = \{w_i\}$;

2) Predict the flow field f_i on each sub-area a_i with the ML model $\mathcal{M}$;

3) Post-process the model outputs $\boldsymbol{f} = \{f_i\}$ to produce the final flow field $\boldsymbol{F}$ (with Algorithm 2 or 3).

Algorithm 2: Velocity Correction

Input : $\boldsymbol{f} = \{f_i\}$: velocity fields for sub-areas; $\boldsymbol{w} = \{w_i\}$: weights for sub-areas

Output: **F**: aggregated velocity field for large area

1) Compute weighted average of $\boldsymbol{f}$ using weights $\boldsymbol{w}$ to obtain **F**;

2) Adjust velocity values in **F** based on its inlet flow rate.

Algorithm 3: Pressure Correction

Input : $\boldsymbol{f} = \{f_i\}$: pressure fields for sub-areas; $\boldsymbol{w} = \{w_i\}$: weights for sub-areas

Output: **F**: aggregated pressure field for large area

1) Moving from outlet to inlet, sequentially shift the pressure field f_i for each sub-area;

2) Compute weighted average of $\boldsymbol{f}$ using weights $\boldsymbol{w}$ to obtain **F**.

in which ϵ denotes the inaccuracy in the prediction. This error ϵ has been minimized during training (Eq. (7)). Hence, we may approximate the relation as

$$\boldsymbol{y}(\boldsymbol{x}) \approx \tilde{\boldsymbol{y}}(\boldsymbol{x}) \tag{10}$$

However, looking at a snapshot of the moving window (Fig. 8) within the larger domain, the captured sub-domain is not guaranteed to be periodic (**C1** violated). The flow profile along the left border of the window will most likely not satisfy $\boldsymbol{u}_{inlet} = (1\times 10^{-4}, 0)\,\mathrm{m\ s^{-1}}$ (**C2** violated). Assuming that the outlet pressure of the large domain is 0, the average pressure must progressively increase from right to left, resulting in a positive pressure value just on the right side of the window. Thus, **C3** is violated for the right side of the window. Hence, we must introduce a *discrepancy function* $\boldsymbol{\delta}(\boldsymbol{x})$ to account for the model inadequacy due to the violation of conditions **C1**, **C2**, and **C3**, such that

$$\boldsymbol{y}(\boldsymbol{x}) \approx \tilde{\boldsymbol{y}}(\boldsymbol{x}) + \boldsymbol{\delta}(\boldsymbol{x}) \tag{11}$$

To correct this bias, we simply have to subtract

$$\boldsymbol{\delta}(\boldsymbol{x}) := \boldsymbol{y}(\boldsymbol{x}) - \tilde{\boldsymbol{y}}(\boldsymbol{x}) \tag{12}$$

from the model prediction. This rectification is, in principle, possible if the true value $\boldsymbol{y}(\boldsymbol{x})$ is known. However, in practice, the true (velocity or pressure) values are unknown beforehand when a neural network is being used as a replacement for numerical simulations. Nevertheless, this obstacle can be circumvented as we will later show in the following subsections. Besides, we recall that the momentum equation (Eq. (5)) states that pressure and velocity are linked, with pressure acting as a constraint on velocity. This connection implies that modifications to the velocity field should be done in relation to the pressure field. However, we had trained two independent surrogates to predict velocity and pressure. For simplicity, we decoupled velocity and pressure when implementing the correction scheme in that these fields are independently updated.

6.3. Velocity correction

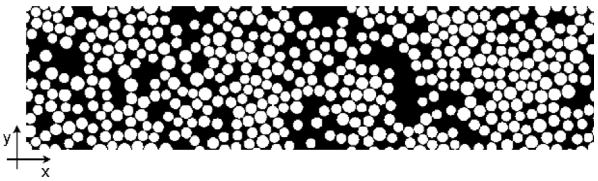
The continuity equation (Eq. (4)) states that the divergence of the velocity field should be zero everywhere. Stated differently, the flow rate should be constant from inlet to outlet in our case. Previous attempts at computing a moving average of the velocity field with the first-order approach (Section 6.1) showed different departures from the reference flow rate as the step size Δt varied. But, in all cases, the inlet flow rate consistently remained accurate. Error statistics of the inlet flow rate further support that (Fig. 7a). This particularly high accuracy can be attributed to the simulations being velocity-driven. It is thus easy for the model to predict quasi-constant values near the inlet. From this observation, we may use the predicted inlet flow rate as a proxy for the true flow rate value. Alternatively, one could simply hand-calculate (thanks to the simulations being velocity-driven) the true flow rate from the microstructure size and inlet volume fraction. Nevertheless, we stick to using the predicted inlet flow rate as a reference to correct for flow rate fluctuations in other regions of the microstructures.

Concretely, we proceed as follows. After estimating the velocity field for the whole rectangular domain by the first-order approach, we calculate the flow rate at each vertical section. At this stage, there exists a mismatch between the predicted and true flow rates (Eq. (11)). Taking the inlet flow rate as a substitute for the true value, we quantify the flow rate mismatch along the length of the microstructure through a series of coefficients obtained by division with respect to the inlet value. We then use these coefficients as scaling factors to adjust the velocity values at the corresponding sections. This procedure is summarized in Algorithm 2 and detailed on our GitHub repository. Results in the Supplementary Material (Fig. D3) show the drastic reduction in error made possible by this correction scheme. Visualization of the final velocity predictions and the corresponding flow rate are provided in Fig. 9b,c,e.

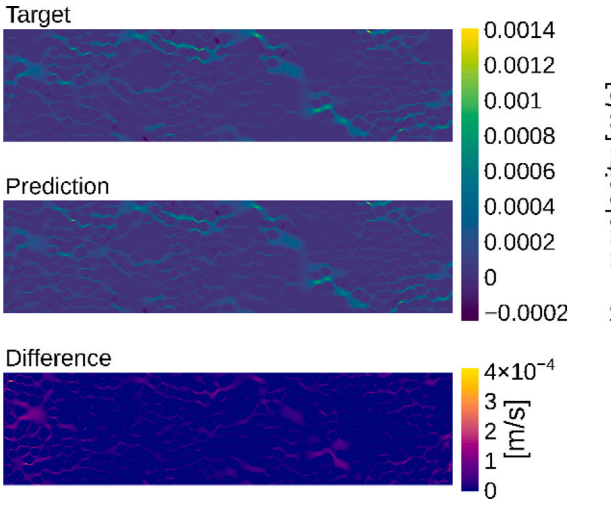
6.4. Pressure correction

The first-order approach, implemented for pressure prediction on the rectangular domain, also led to unrealistic pressure fields. We observed discontinuities in the predictions, with values differing by large margins from the reference (Supplementary Material, Fig. E1). Multiple factors play a role, with the violation of condition **C3** being the most obvious one. Condition **C3** is violated because the model output for any subdomain captured by the window assumes that pressure is 0 at the right of that subdomain (Fig. 8). This is physically incongruent if we also consider the pressure to be 0 at the outlet of the larger rectangular domain. As such, we need a device that ensures that the average pressure value at the outlet of any subdomain is congruent with the overall pressure evolution in the context of the entire domain.

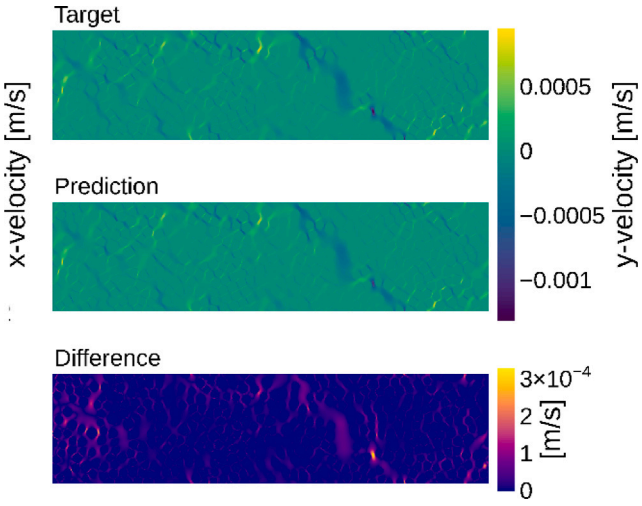
With this consideration in mind, we purposefully move the sliding window from right to left in the case of pressure prediction. This choice of direction is to enable tracking of the pressure evolution from the rectangular domain's outlet. At each step, we evaluate the pressure field and also calculate the average pressure per vertical section. Then, before moving on to the next step, we shift the entire pressure field by a scalar. This scalar is determined by the average pressure value calculated during the preceding step at the vertical section on the window's inlet. This procedure is outlined in Algorithm 3 and also detailed on our GitHub repository. The results in the Supplementary Material (Fig. E2) demonstrate its effectiveness. Fig. 9.d and .f show the resulting pressure field as well as the evolution of the average pressure. Interestingly, we observe a close agreement between predictions and targets. The maximum difference in the prediction of the pressure field is an order of magnitude lower than the maximum observable pressure.



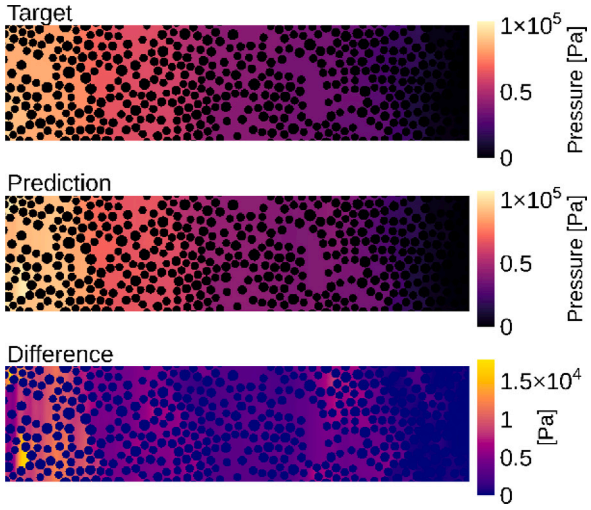
(a) Microstructure with aspect ratio equal to 4



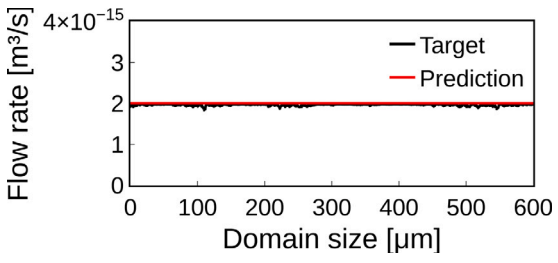
(b) x-component of the predicted velocity field



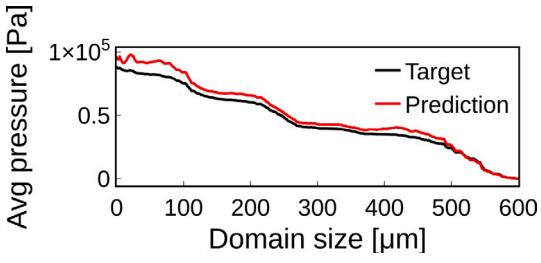
(c) y-component of the predicted velocity field



(d) Predicted pressure field



(e) Flow rate along the length of the microstructure



(f) Evolution of the average pressure

Fig. 9. (a) Microstructure with an aspect ratio equal to 4. (b) *x*- and (c) *y*-component of the predicted velocity field. (d) Pressure field prediction. (e) Flow rate and (f) average pressure predicted for the rectangular domain.

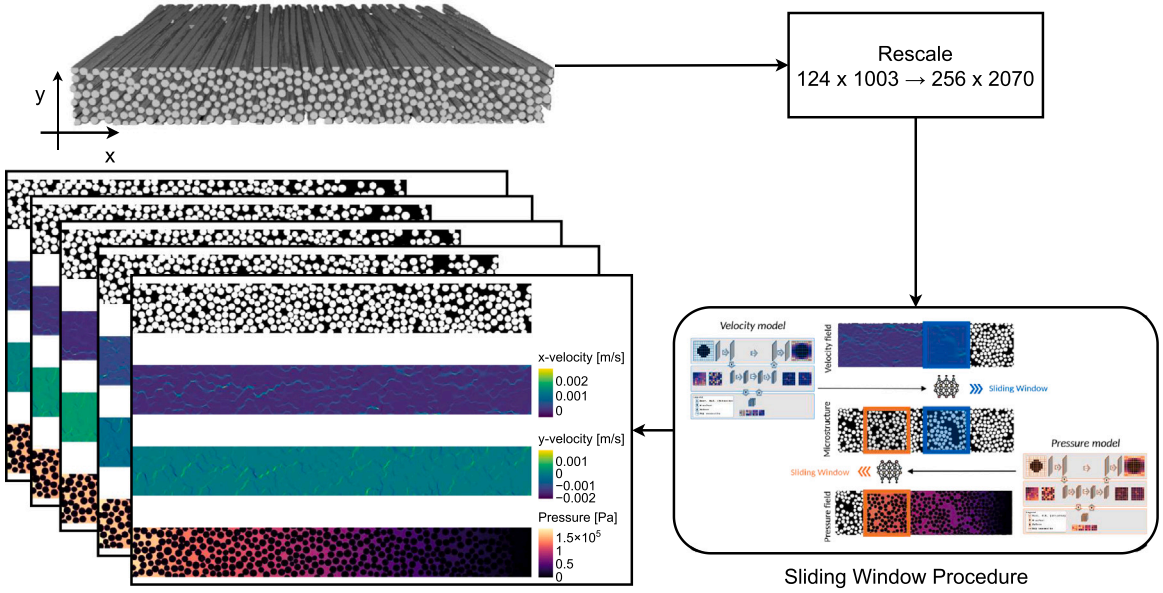


Fig. 10. Flowchart of the steps taken to predict flow fields on the micrograph scans from Syerko et al. [17].

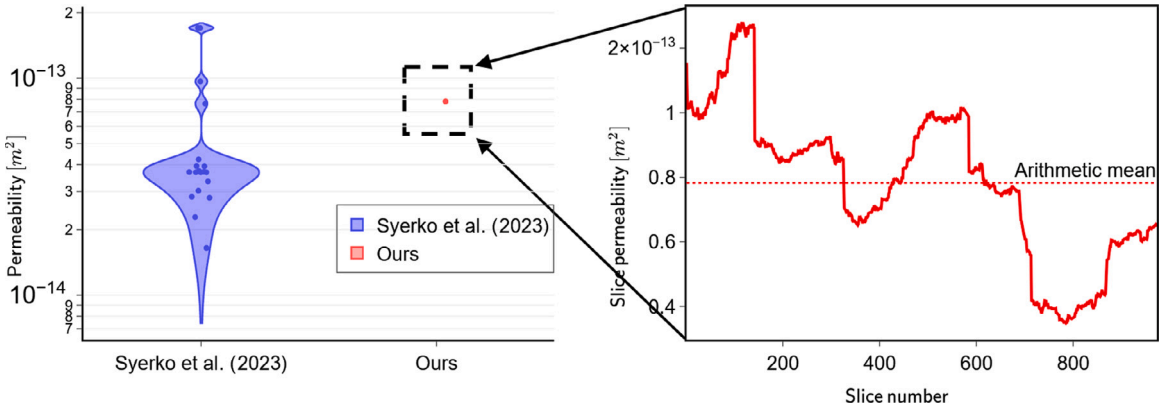


Fig. 11. Permeability prediction for a real micrograph using our *sliding window* procedure, compared against results from numerical simulations reported in Syerko et al. [17].

7. Benchmark against literature data

Beyond comparison with numerically generated data, the added value of a proposed method lies in its successful application to real data. Syerko et al. [17] performed an international virtual permeability benchmark exercise in which the data consisted of scans of a glass fiber reinforced composite sample extracted by X-ray computed microtomography. Fig. 10 shows a 3D view of the 973 consecutive 2D scan slices, with a resolution of $124 \times 1003\,\text{px}^2$. This endeavor resulted in the participants providing permeability estimations based on 2D/3D numerical simulations run under various assumptions. In Fig. 11 we report the results for transverse permeability, extracted from the supplementary data accompanying that paper [17].

We undertook the task of evaluating our *sliding window* approach on these micrograph scans and comparing how its results stand in comparison with those reported in the benchmark. We recall that our surrogate models expect square $256 \times 256\,\text{px}^2$ images. However, each scan slice has a resolution of $124 \times 1003\,\text{px}^2$, which would result in $124 \times 124\,\text{px}^2$ sub-domains if we were to fragment it into squares. As a result, this endeavor presents challenges. First, the data were experimentally acquired rather than numerically generated and were not present in the training set. Consequently, they may exhibit characteristics distinct from the synthetic data. For instance, the experimental data contain low-fiber-density regions typical of multi-scale fibrous microstructures, which our current microstructural generator does not explicitly produce, although this could be addressed in future work. Additionally, the experimental microstructure was obtained via image-based extraction methods, whose parameter choices can influence the results [59] and introduce features absent in the training data. Finally, the experimental data have lower resolution compared to the training set.

To accommodate the surrogate models' requirements, we used bilinear interpolation to upscale the scan slices from $124 \times 1003\,\text{px}^2$ to $256 \times 2070\,\text{px}^2$. Fig. 10 shows the predicted velocity and pressure fields for selected slices. The flow field patterns all look realistic. Next, we utilized Darcy's law to infer the permeability of each 2D slice from the predicted fields. The computed permeability values are plotted in the inset shown within Fig. 11. We observe the variation of permeability along the depth of the volume. Then, we use an electric circuit analogy to infer the transverse permeability of the 3D domain from the individual 2D permeability values. We note that this circuit analogy has previously been used in other works [24,60,61]. Concretely, we consider the 2D slices to be resistances in parallel, as we are interested in flow behavior in the x-direction and compute the arithmetic mean of the individual permeabilities. The equivalent permeability is $7.82 \times 10^{-14}\,\text{m}^2$ (Fig. 11)

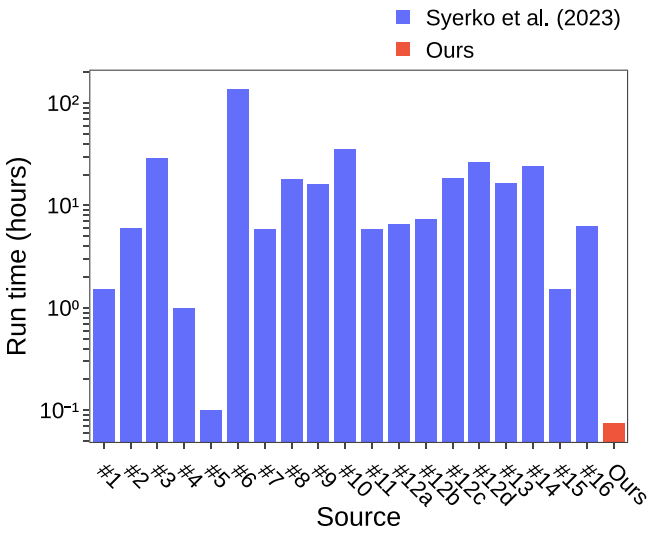


Fig. 12. Run time for predicting the permeability of the micrograph scan using our framework compared with the results of the participants in the benchmark by Syerko et al. [17].

and falls well within the range of values deemed inliers by the authors of the benchmark [17]. More interestingly, our prediction stands right next to the values reported by participants #2 and #6 who performed 2D simulations (see supplementary data from Syerko et al. [17]). But beyond that, the computational efficiency of our framework further sets it apart (Fig. 12). Evaluating the permeability of the scans required only 3 min, while the participants in the benchmark [17] reported run times up to 136 h. The reduction in computational time spans several orders of magnitude, highlighting the significant benefit of our approach.

8. Conclusions

In this work, we introduced two surrogate machine learning models to predict the velocity and pressure fields in fiber-reinforced microstructures that vary in domain size, fiber diameter, and fiber volume fraction. The reported error statistics and examples showed the robustness of the trained surrogate models, even when they were performing at their worst. In addition, we used Darcy's law to infer microstructure permeability from the field predictions by the surrogates. These permeability predictions are achieved with an accuracy that surpasses previous works.

Beyond the mere evaluation of model performance on square domains (a constraint originating from convolutional neural networks), we introduced a *sliding window* method to extend model inference to rectangular domains. We implemented physical flow constraints in this framework, without modifying the loss functions, as such a modification would potentially lead to the requirement of a more populated database and increased computational cost to train models.

The *sliding window* procedure was introduced for two-dimensional microstructures. It will be further extended to 2D mesostructures of fiber bundles in our future work, in which case, a more clever partitioning method might be needed. An example could be a quad-tree decomposition of the image [62] based on regional fiber volume fraction to process fiber-free and fiber-packed regions separately. Moreover, we foresee the development of an extended method for efficient 3D flow prediction of complex microstructures such as those investigated by Gomarasca et al. [63] by stacking 2D predictions from the surrogate models and applying local corrections based on the surrounding topology.

Comparison with numerical simulations and a test against a benchmark exercise showed its reliability and good prediction accuracy. Moreover, this approach is significantly faster than numerical simulations explored in the said benchmark, and reduces the computational time from multiple hours to a few minutes. Our source codes for the surrogate models and the proposed *sliding window* method are publicly released, thus further lowering the effort to estimate the permeability of fibrous microstructures.

CRediT authorship contribution statement

Jimmy Gaspard Jean: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Data curation. **Guillaume Broggi:** Writing – review & editing, Visualization, Supervision, Software, Methodology, Data curation. **Baris Caglar:** Writing – review & editing, Supervision, Project administration, Methodology, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work has been conducted within the framework of the CAELESTIS (Hyperconnected simulation ecosystem supporting probabilistic design and predictive manufacturing of next generation aircraft structures) project, funded by the European Climate, Infrastructure and Environment Executive Agency (CINEA) under grant agreement No. 101056886. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of CINEA. Neither the European Union nor the granting authority can be held responsible for them.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.compositesa.2025.109337.

Data availability

The dataset used in this study is freely accessible on Zenodo at https://doi.org/10.5281/zenodo.16940478 [64]. The source code of the ML model used in this paper is also available under an open-source license on GitHub at https://github.com/camp-lab-tud/arbitrary-microstructure-flow. Our purpose in doing so is to be transparent and promote the use of machine learning in the composites field. We encourage readers to explore the code and industry professionals to apply it to their manufacturing-related problems.

References

[1] Potter Kevin. Resin transfer moulding. Springer Science & Business Media; 1997, http://dx.doi.org/10.1007/978-94-009-0021-9.
[2] Verrey J, Wakeman MD, Michaud V, Månson J-AE. Manufacturing cost comparison of thermoplastic and thermoset RTM for an automotive floor pan. Compos Part A: Appl Sci Manuf 2006;37(1):9–22.
[3] Wang J, Simacek P, Advani SG. Use of centroidal voronoi diagram to find optimal gate locations to minimize mold filling time in resin transfer molding. Compos Part A: Appl Sci Manuf 2016;87:243–55.
[4] Werlen Vincent, Rytka Christian, Wegmann Stephanie, Philipp Halime, Khalaf Yara, Michaud Véronique, et al. Novel tooling for direct melt impregnation of textile with variotherm injection moulding: Methodology and proof of concept. J Compos Mater 2022;56(28):4245–57.
[5] Teixidó Helena, Staal Jeroen, Caglar Baris, Michaud Véronique. Capillary effects in fiber reinforced polymer composite processing: a review. Front Mater 2022;9:809226.

[6] Daelemans Lode, Tomme Brecht, Caglar Baris, Michaud Véronique, Van Stappen Jeroen, Cnudde Veerle, et al. Kinematic and mechanical response of dry woven fabrics in through-thickness compression: Virtual fiber modeling with mesh overlay technique and experimental validation. Compos Sci Technol 2021;207:108706.

[7] Zhang Fan, Cosson Benoît, Comas-Cardona Sébastien, Binetruy Christophe. Efficient stochastic simulation approach for RTM process with random fibrous permeability. Compos Sci Technol 2011;71(12):1478–85.

[8] Shojaei Akbar, Ghaffarian S Reza, Karimian S Mohammad-Hossien. Modeling and simulation approaches in the resin transfer molding process: A review. Polym Compos 2003;24(4):525–44.

[9] Sozer EM, Simacek P, Advani SG. Resin transfer molding (RTM) in polymer matrix composites. In: Manufacturing techniques for polymer matrix composites (pMCs). Elsevier; 2012, p. 245–309.

[10] Geoffre Aubin, Wielhorski Yanneck, Moulin Nicolas, Bruchon Julien, Drapier Sylvain, Liotier Pierre-Jacques. Influence of intra-yarn flows on whole 3D woven fabric numerical permeability: from Stokes to Stokes-Darcy simulations. Int J Multiph Flow 2020;129:103349.

[11] Vallmajó O, Arteiro A, Guerrero JM, Melro AR, Pupurs A, Turon A. Micromechanical analysis of composite materials considering material variability and microvoids. Int J Mech Sci 2024;263:108781.

[12] Dei Sommi Andrea, Lionetto Francesca, Maffezzoli Alfonso. An overview of the measurement of permeability of composite reinforcements. Polymers 2023;15(3):728.

[13] Berdichevsky Alexander L, Cai Zhong. Preform permeability predictions by self-consistent method and finite element simulation. Polym Compos 1993;14(2):132–43.

[14] Yazdchi K, Srivastava S, Luding Stefan. Microstructural effects on the permeability of periodic fibrous porous media. Int J Multiph Flow 2011;37(8):956–66.

[15] Geoffre Aubin, Moulin Nicolas, Bruchon Julien, Drapier Sylvain. Reappraisal of upscaling descriptors for transient two-phase flows in fibrous media. Transp Porous Media 2023;147(2):345–74.

[16] Li Yong, Chi Yanmeng, Zhao Chaojie, Miao Yanan, Han Shanling, Chen Long. Modelling fluid flow in carbon fibre porous media based on X-ray microtomography and lattice Boltzmann method. Compos Struct 2022;300:116085.

[17] Syerko Elena, Schmidt Tim, May David, Binetruy Christophe, Advani Suresh G, Lomov S, et al. Benchmark exercise on image-based permeability determination of engineering textiles: Microscale predictions. Compos Part A: Appl Sci Manuf 2023;167:107397.

[18] Hornik Kurt, Stinchcombe Maxwell, White Halbert. Multilayer feedforward networks are universal approximators. Neural Netw 1989;2(5):359–66.

[19] Zhou Ding-Xuan. Universality of deep convolutional neural networks. Appl Comput Harmon Anal 2020;48(2):787–94.

[20] Araya-Polo Mauricio, Alpak Faruk O, Hunter Sander, Hofmann Ronny, Saxena Nishank. Deep learning–driven permeability estimation from 2D images. Comput Geosci 2020;24(2):571–80.

[21] Gärttner Stephan, Alpak Faruk O, Meier Andreas, Ray Nadja, Frank Florian. Estimating permeability of 3D micro-CT images by physics-informed CNNs based on DNS. Comput Geosci 2023;27(2):245–62.

[22] Santos Javier E, Xu Duo, Jo Honggeun, Landry Christopher J, Prodanović Maša, Pyrcz Michael J. PoreFlow-net: A 3D convolutional neural network to predict fluid flow through porous media. Adv Water Resour 2020;138:103539.

[23] Wang Ying Da, Chung Traiwit, Armstrong Ryan T, Mostaghimi Peyman. ML-LBM: predicting and accelerating steady state flow simulation in porous media with convolutional neural networks. Transp Porous Media 2021;138(1):49–75.

[24] Caglar Baris, Broggi Guillaume, Ali Muhammad A, Orgéas Laurent, Michaud Véronique. Deep learning accelerated prediction of the permeability of fibrous microstructures. Compos Part A: Appl Sci Manuf 2022;158:106973.

[25] Hanna John M, Aguado José V, Comas-Cardona Sebastien, Le Guennec Yves, Borzacchiello Domenico. A self-supervised learning framework based on physics-informed and convolutional neural networks to identify local anisotropic permeability tensor from textiles 2D images for filling pattern prediction. Elsevier; 2024.

[26] Schmidt Tim, Natarajan Dinesh Krishna, Duhovic Miro, Cassola Stefano, Nuske Marlon, May David. Numerical data generation for building machine learning models for permeability estimation of fibrous structures. Polym Compos 2025.

[27] Wang Y, Xu S, Bwar KH, Eisenbart B, Lu G, Belaadi A, et al. Application of machine learning for composite moulding process modelling. Compos Commun 2024;48:101960.

[28] Ronneberger Olaf, Fischer Philipp, Brox Thomas. U-net: Convolutional networks for biomedical image segmentation. In: Medical image computing and computer-assisted intervention–mICCAI 2015: 18th international conference, munich, Germany, October 5-9, 2015, proceedings, part III 18. Springer; 2015, p. 234–41.

[29] Ho Jonathan, Jain Ajay, Abbeel Pieter. Denoising diffusion probabilistic models. Adv Neural Inf Process Syst 2020;33:6840–51.

[30] Rombach Robin, Blattmann Andreas, Lorenz Dominik, Esser Patrick, Ommer Björn. High-resolution image synthesis with latent diffusion models. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2022, p. 10684–95.

[31] Darcy Henry. Les fontaines publiques de la ville de Dijon: exposition et application des principes à suivre et des formules à employer dans les questions de distribution d'eau, vol. 1, Victor dalmont; 1856.

[32] Borassi Michele, Epasto Alessandro, Lattanzi Silvio, Vassilvitskii Sergei, Zadimoghaddam Morteza. Sliding window algorithms for k-clustering problems. Adv Neural Inf Process Syst 2020;33:8716–27.

[33] O'Shea K. An introduction to convolutional neural networks. 2015, arXiv preprint arXiv:1511.08458.

[34] Wu Yuxin, He Kaiming. Group normalization. In: Proceedings of the European conference on computer vision. 2018, p. 3–19.

[35] Hendrycks Dan, Gimpel Kevin. Gaussian error linear units (gelus). 2016, arXiv preprint arXiv:1606.08415.

[36] Ba Jimmy Lei, Kiros Jamie Ryan, Hinton Geoffrey E. Layer normalization. 2016, arXiv preprint arXiv:1607.06450.

[37] Maas Andrew L, Hannun Awni Y, Ng Andrew Y, et al. Rectifier nonlinearities improve neural network acoustic models. In: Proc. icml, vol. 30, (1):Atlanta, GA; 2013, p. 3.

[38] Xu Bing, Wang Naiyan, Chen Tianqi, Li Mu. Empirical evaluation of rectified activations in convolutional network. 2015, URL https://arxiv.org/abs/1505.00853.

[39] Bahdanau Dzmitry, Cho Kyunghyun, Bengio Yoshua. Neural machine translation by jointly learning to align and translate. 2014, arXiv preprint arXiv:1409.0473.

[40] Vaswani Ashish, Shazeer Noam, Parmar Niki, Uszkoreit Jakob, Jones Llion, Gomez Aidan N, et al. Attention is all you need. Adv Neural Inf Process Syst 2017;30.

[41] Dumoulin Vincent, Visin Francesco. A guide to convolution arithmetic for deep learning. 2016, arXiv preprint arXiv:1603.07285.

[42] Oyedotun Oyebade K, Al Ismaeil Kassem, Aouada Djamila. Why is everyone training very deep neural network with skip connections? IEEE Trans Neural Networks Learn Syst 2022;34(9):5961–75.

[43] Mesquita Francisco, Bucknell Steve, Leray Yann, Lomov Stepan V, Swolfs Yentl. Single carbon and glass fibre properties characterised using large data sets obtained through automated single fibre tensile testing. Compos Part A: Appl Sci Manuf 2021;145:106389.

[44] Blomqvist Victor. Pymunk 6.11.1. 2025, URL https://pymunk.org.

[45] Bistafa Sylvio R. 200 years of the Navier–Stokes equation. Rev Bras Ensino Física 2024;46:e20230398.

[46] Jasak Hrvoje. OpenFOAM: Open source CFD in research and industry. Int J Nav Archit Ocean Eng 2009;1(2):89–94.

[47] Patankar Suhas V, Spalding D Brian. A calculation procedure for heat, mass and momentum transfer in three-dimensional parabolic flows. In: Numerical prediction of flow, heat transfer, turbulence and combustion. Elsevier; 1983, p. 54–73.

[48] Broggi Guillaume, Bessa Miguel, Caglar Baris. A realistic Python 2D microstructural generator: Towards an open-source, bottom-up, and multiscale machine learning-based simulation pipeline for composite materials. Zenodo; 2024, http://dx.doi.org/10.5281/ZENODO.12771719.

[49] Gebart B Rikard. Permeability of unidirectional reinforcements for RTM. J Compos Mater 1992;26(8):1100–33.

[50] Van Rossum Guido, Drake Fred L. Introduction to python 3: python documentation manual part 1. CreateSpace; 2009.

[51] Ranade Rishikesh, Nabian Mohammad Amin, Tangsali Kaustubh, Kamenev Alexey, Hennigh Oliver, Cherukuri Ram, Choudhry Sanjay. Domino: A decomposable multi-scale iterative neural operator for modeling large scale engineering simulations. 2025, arXiv preprint arXiv:2501.13350.

[52] Choy Chris, Kamenev Alexey, Kossaifi Jean, Rietmann Max, Kautz Jan, Azizzadenesheli Kamyar. Factorized implicit global convolution for automotive computational fluid dynamics prediction. 2025, arXiv preprint arXiv:2502.04317.

[53] Calvo-Ordoñez Sergio, Cheng Chun-Wun, Huang Jiahao, Zhang Lipei, Yang Guang, Schonlieb Carola-Bibiane, et al. The missing u for efficient diffusion models. 2023, arXiv preprint arXiv:2310.20092.

[54] Ruder Sebastian. An overview of gradient descent optimization algorithms. 2016, arXiv preprint arXiv:1609.04747.

[55] Paszke Adam, Gross Sam, Chintala Soumith, Chanan Gregory, Yang Edward, DeVito Zachary, et al. Automatic differentiation in pytorch. 2017.

[56] Arlot Sylvain, Celisse Alain. A survey of cross-validation procedures for model selection. 2010.

[57] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, et al. Scikit-learn: Machine learning in python. J Mach Learn Res 2011;12:2825–30.

[58] Messenger Daniel A, Tran April, Dukic Vanja, Bortz David M. The weak form is stronger than you think. 2024, arXiv preprint arXiv:2409.06751.

[59] Lavaggi T, Gillespie JW, Mulye PD, Binetruy C, Advani SG. Influence of sample size on permeability of carbon–carbon composites with stochastic microstructure. Compos Part A: Appl Sci Manuf 2025;198:109126. http://dx.doi.org/10.1016/j.compositesa.2025.109126, URL https://www.sciencedirect.com/science/article/pii/S1359835X25004208.

[60] Salvatori Damiano, Caglar Baris, Teixidó Helena, Michaud Véronique. Permeability and capillary effects in a channel-wise non-crimp fabric. Compos Part A: Appl Sci Manuf 2018;108:41–52.

[61] Ali Muhammad A, Khan Kamran A, Umer Rehan. An electric circuit analogy-based homogenization approach for predicting the effective permeability of complex dual-scale porous media. Mater Today Commun 2021;28:102565.
[62] Marquez Gerardo Rodmar Conde, Escalante Hugo Jair, Sucar Luis Enrique. Simplified quadtree image segmentation for image annotation. In: Proceedings of the 1st automatic image annotation and retrieval workshop, vol. 2011, 2010, p. 24–34.
[63] Gomarasca S, Peeters DMJ, Atli-Veltin B, Dransfeld C. Characterising microstructural organisation in unidirectional composites. Compos Sci Technol 2021;215:109030. http://dx.doi.org/10.1016/j.compscitech.2021.109030.
[64] Jean Jimmy Gaspard, Broggi Guillaume, Caglar Baris. A benchmark dataset for steady-state flow models in fibrous microstructures. Zenodo; 2025, http://dx.doi.org/10.5281/zenodo.16940478.