DOI: https://doi.org/10.14311/TPFM.2025.018

ARTIFICIAL INTELLIGENCE FOR SIMULATION OF SOOT DEPOSITS: FROM 2D TO 3D ESTIMATES

M. Khýr [1,2], M. Isoz [1,2], M. Plachá [3]

[1] **Czech Academy of Sciences, Institute of Thermomechanics, Dolejškova 5, Prague 182 00, Czech Republic**

[2] **Department of Mathematics, Informatics and Cybernetics, Faculty of Chemical Engineering, University of Chemistry and Technology, Technická 5, Prague 166 28, Czech Republic**

[3] **Department of Chemical Engineering, Faculty of Chemical Engineering, University of Chemistry and Technology, Technická 5, Prague 166 28, Czech Republic**

Abstract

The accumulation of soot within the porous microstructure of catalytic filters (CFs) impacts the overall performance of the device, making the prediction of soot distribution a major challenge in the design of CFs. We previously developed a neural network that leverages convolutional autoencoders and fully-connected layers and demonstrated that this network can effectively estimate the distribution of deposited soot within two-dimensional model microstructures. In this contribution, we expand on the previously devised model, by exploring its possible extension to processing three-dimensional data from real-world catalytic filter microstructures. Specifically, we examine and compare three principally different approaches to generating three-dimensional estimates, spanning from a posteriori reconstructions to the application of fully three-dimensional convolutional networks.

Keywords: CFD, ANN, CNN.

1 Introduction

Catalytic filters are devices used in automotive exhaust gas aftertreatment to execute particulate matter filtration and chemical gas conversion simultaneously, thus reducing costs and spatial requirements. However, due to the higher complexity of catalytic filters, compared to multi-device aftertreatment systems, the overall CF performance is generally more sensitive to the morphology of their porous microstructure, including the catalytic coating [1]. Consequently, models for prediction of in-operation soot deposition are crucial for CF development.

Meanwhile, artificial neural networks might present a potentially promising tool for modelling particle deposition, as they have been shown to act as computationally efficient surrogates for numerical computational models [2]. In our previous work, we have designed an artificial neural network, which can efficiently predict the spatial distribution of particles deposited in two-dimensional model microstructures [3], serving as an alternative to a numerical model based on computational fluid dynamics (CFD). However, in order to find use in real-world CF development, the model must be improved in two key directions. First, the model must be extended to facilitate the estimation of three-dimensional particle distributions in real CF microstructures. Second, since the temporal evolution of the spatial deposit distribution is ultimately sought, the model must be adapted to allow for time-resolved estimates.

In this paper, we focus on the first of the two aforementioned tasks, namely, we explore possible extensions of the existing model to process three-dimensional data. In the following, we first briefly introduce the numerical model that was used to generate the training data for our neural network. Then, we discuss the architecture of the data-driven model and the foundational elements of the neural network used. Specifically, we focus on deep neural networks and convolutional autoencoders. Subsequently, we examine and compare three principally different approaches to generating 3D estimates. The first possible approach is to leave the network architecture unmodified and only add a post-processing step of reconstructing the 3D estimate from 2D slices. The second approach is to provide the network with additional spatial data from neighbouring 2D slices to enhance the estimates. The last option is to create a fully three-dimensional network that would provide 3D estimates of the soot distribution based on 3D input data.

2 Mathematical model

Full-order numerical model. Our previously developed numerical solver [4] was used to prepare training and validation data for the neural network. This CFD solver leverages a partially two-way coupled approach to modelling particle accumulation inside the filter microstructure. During simulation, the flow field inside the microstructure is first calculated under steady-state conditions, assuming that suspended soot particles have no effect on the flowing gas. The flow of the gas is described by the momentum and mass balance equations for an incompressible fluid,

$$\begin{aligned} \nabla \cdot (\boldsymbol{u} \otimes \boldsymbol{u}) - \nabla \cdot (\nu \nabla \boldsymbol{u}) &= -\nabla \tilde{p} + \boldsymbol{s}\,, \\ \nabla \cdot \boldsymbol{u} &= 0\,, \end{aligned} \tag{1}$$

where $\boldsymbol{u}$ is the fluid velocity, ν is the fluid kinematic viscosity and $\tilde{p}$ is the kinematic pressure, which are solved using the SIMPLE algorithm as implemented in the OpenFOAM framework [5]. Note the source term $\boldsymbol{s}$ on the right-hand side of the momentum balance in (1), which is added within the catalytic coating and the accumulated soot to represent the additional resistance of the flow in these porous zones using the Darcy permeability law.

In the precalculated velocity field $\boldsymbol{u}$, the particle trajectories are subsequently evaluated according to Newton's second law of motion in the form

$$m \frac{\mathrm{d}^2 \boldsymbol{r}}{\mathrm{d}t^2} = \boldsymbol{F}_\mathrm{D} + \boldsymbol{F}_\mathrm{B}\,, \tag{2}$$

where $\boldsymbol{r}$ is the particle location vector, t is the time and m is the particle mass. The drag force $\boldsymbol{F}_\mathrm{D}$ depends directly on the fluid velocity field via modified Stokes' law [6], and $\boldsymbol{F}_\mathrm{B}$ is the random Brownian force [7]. Once a particle comes into contact with the substrate, catalytic coating, or accumulated soot, it is considered trapped. The trapped particles are then periodically added to the computational mesh, so that they contribute to the aforementioned source term $\boldsymbol{s}$ in (1) during subsequent reevaluations of the flow field.

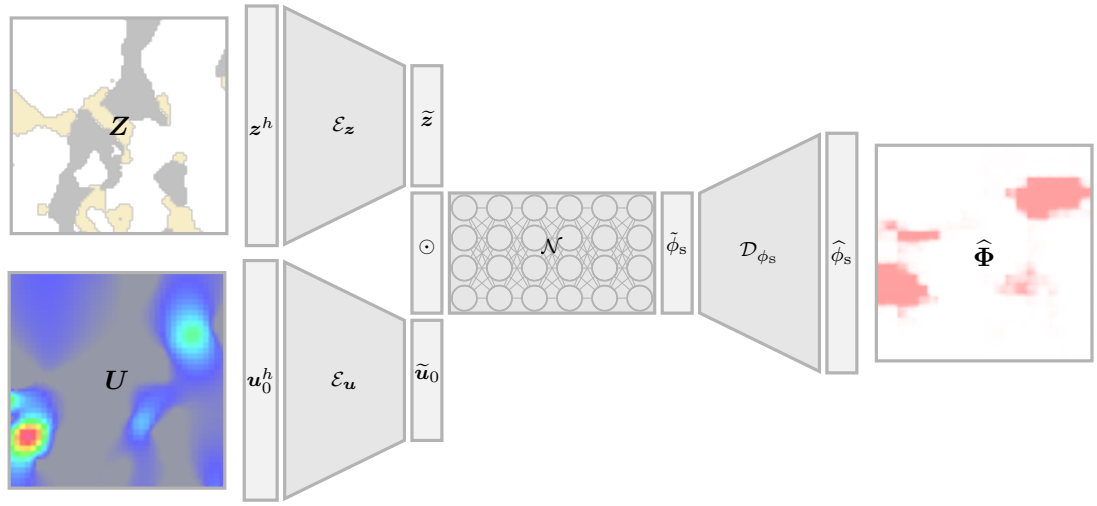

Figure 1: General structure of the neural network model used in this work.

Artificial neural network model. The neural network aims to estimate the distribution of the deposits of soot particles measured by volume fraction of soot in a cell ϕ_s at an a priori selected time based on computationally easily available data: the porous microstructure $\boldsymbol{z}$ and the precalculated velocity field $\boldsymbol{u}_0$ in the clean microstructure. Fields are represented as arrays of discrete values that correspond to cell data from an orthogonal structured computational mesh. The network then always operates on a small segment of the entire domain.

Previously, we have devised a suitable neural network architecture to process such structured high-dimensional data, which was shown to provide good qualitative estimates of particle deposits [3]. The network was implemented in the Keras TensorFlow [8, 9] framework. Altogether,

our neural network model, as depicted in Figure 1, leverages two encoders labelled $\mathcal{E}_{\boldsymbol{z}}$ and $\mathcal{E}_{\boldsymbol{u}}$ to process input data – the discrete representations of microstructure $\boldsymbol{z}^h$ and the clean velocity field $\boldsymbol{u}_0^h$ – and one decoder $\mathcal{D}_\phi$ to reconstruct the estimate of the soot volume fraction field $\widehat{\boldsymbol{\phi}}_{\mathrm{s}}^h$ at the output. Between these components, the fully-connected network $\mathcal{N}$ estimates the latent representation of the soot distribution as $\widetilde{\phi} = \mathcal{N}(\widetilde{\boldsymbol{z}}, \widetilde{\boldsymbol{u}}_0)$.

The framework building blocks, that is, (i) the autoencoders for microstructure representation, clean velocity field, and soot distribution; and (ii) the fully-connected network $\mathcal{N}$ are trained separately. The essential theoretical background on these structural elements is briefly covered in the following. For further details, the reader is referred to the literature [10].

Fully-connected networks. In principle, a neural network is a composed parametric function $\boldsymbol{g}(\boldsymbol{x};\boldsymbol{\theta})$ that can act as an approximation of any given function $\boldsymbol{f}(\boldsymbol{x})$. The quality of the approximation, which is measured by the so-called *loss function*, then depends (i) on the structure (and capacity) of the neural network and (ii) on the values of parameters $\boldsymbol{\theta}$. To reach optimal parameter values $\boldsymbol{\theta}^*$, such that minimises the loss function, some kind of learning algorithm must be employed, which can adjust parameter values based on samples containing inputs $\boldsymbol{x}$ and outputs $\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x})$. This is called *supervised learning*, and the most common learning algorithms are variants of *gradient descent* with gradient evaluation via *backpropagation*.

In the case of multilayer perceptron, the network function $\boldsymbol{g}$ is composed of individual layers, where each layer corresponds to a composed function in the form $\boldsymbol{a}\,(\boldsymbol{W}\,\boldsymbol{o})$, where a is the non-linear activation function (applied elementwise) and $\boldsymbol{W}\,\boldsymbol{o}$ is the linear combination of the outputs of the previous layer $\boldsymbol{o}$, formally extended by a unit element for bias, with coefficients given by the weight matrix $\boldsymbol{W}$. Note that each element of any given layer is then dependent on all the outputs in a previous layer, and the number of trainable parameters per layer corresponds to the elements of the weight matrix $\boldsymbol{W}$ (that is, $n^2 + n$, where n is the number of neurons per layer, which is assumed constant for simplicity). Consequently, the total number of trainable parameters for the m layers of n neurons in each is $m\,n^2 + m\,n$, and the computational complexity of forward propagation is $\mathcal{O}(m\,n^2)$ assuming naive implementation of matrix multiplication. Note that in practice, the number of neurons per layer may vary.

Convolutional neural networks. Convolutional networks are structurally similar to the fully-connected networks discussed in the previous paragraph. Their layers are connected via transformation, which is based on a discrete convolution operation. We will denote $\boldsymbol{M} * \boldsymbol{K}$ the convolution of matrices $\boldsymbol{M} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{K} \in \mathbb{R}^{k\times k}$, where the matrix $\boldsymbol{K}$ is referred to as *convolution kernel*. The convolution operation is then defined as

$$(\boldsymbol{M} * \boldsymbol{K})_p^q = \sum_{u=-\ell}^{\ell} \sum_{v=-\ell}^{\ell} (\boldsymbol{M})_{p+u}^{q+v}\,(\boldsymbol{K})_{\ell+u+1}^{\ell+v+1}\,, \quad p,q = 1,2\ldots,m\,,$$

assuming $k = 2\ell+1$, and zero padding of the matrix $\boldsymbol{M}$, so that $(\boldsymbol{M})_i^j = 0$ if $(i,j) \notin \{1,2,\ldots,m\}\times\{1,2,\ldots,m\}$. The extension of the definition to three-dimensional arrays is then straightforward. Note that since the kernel $\boldsymbol{K}$ is usually much smaller than the matrix $\boldsymbol{M}$, $k \ll m$, the number of trainable parameters k^2 is significantly smaller than in the case of fully-connected networks.

In practice, each convolutional layer in a network usually contains several so-called filters. The values in each filter are obtained as a non-linear transformation of parameter-weighted linear combination of convolutions of the filter values from the previous layer. If we consider a constant number of filters f between m identical layers, this results in $m\,f^2\,k^2$ trainable parameters for 2D convolutional layers and $m\,f^2\,k^3$ for 3D convolution (independently on the dimensions of the layer). The computational complexity of 2D convolution is $\mathcal{O}(n\,f^2\,m^2)$, where n is the number of layers, f is the number of filters, k is the kernel size, and m is the layer size. Once again, we assume naive implementation of the convolution algorithm.

Furthermore, convolutional layers usually contain strides or they are paired with pooling or dropout layers, which help to progressively decrease the overall layer size. For example, the maximum pooling layers, used in this work, are based on the transformation

$$(\boldsymbol{N})_{p,q} = \max_{\substack{u=1,2,\ldots,k\\ v=1,2,\ldots,k}} (\boldsymbol{M})_{p\,k+u}^{q\,k+v}\,, \quad p,q = 1,2\ldots,\frac{m}{k}\,,$$

where k is the pool size, which is in fact the layer size reduction factor. For their ability to decrease amounts of data and extract important information, convolutional networks are often used in image processing for feature detection. For the same purpose, our artificial neural network for particle deposit estimation employs these layers in the autoencoders.

Convolutional autoencoders. The overall neural network contains three components taken from convolutional autoencoders, which are trained separately in advance. Generally, an autoencoder $\boldsymbol{f}$ is a type of neural network designed to reproduce its input $\boldsymbol{x}$ so that $\boldsymbol{f}(\boldsymbol{x}) \approx \boldsymbol{x}$. Furthermore, the autoencoder function $\boldsymbol{f}$ can be viewed as a composed function combining an encoder function $\mathcal{E}(\boldsymbol{x})$ and a decoder function $\mathcal{D}(\widetilde{\boldsymbol{x}})$, where $\widetilde{\boldsymbol{x}} = \mathcal{E}(\boldsymbol{x})$ is the *latent representation* of the input $\boldsymbol{x}$. Since the autoencoder has a bottleneck-like structure, it is expected that the low-dimensional latent representation $\widetilde{\boldsymbol{x}}$ captures important information from the full-dimensional input $\boldsymbol{x}$.

3 Results

Simple reconstruction. The most straightforward approach to generate three-dimensional estimates is a simple reconstruction from two-dimensional estimates. Since the two-dimensional segments can be seen as slices of a three-dimensional structure, they can be stacked to form a composed three-dimensional estimate, as demonstrated in Figure 2a. Using this approach, multiple reconstruction options are possible, because the cube-shaped array can be sliced into a series of squares in each of all three axis-wise directions. In the example shown, an average of the axis-wise estimates is used. A comparison of composed 2D estimates from axis-wise directions, which constitute the averaged data, is shown in Figure 2b.

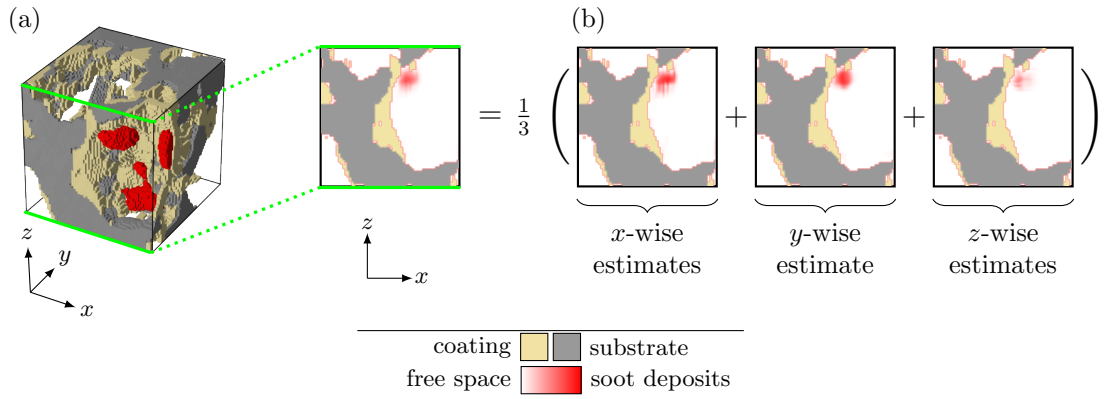

Figure 2: Exemplary (a) 3D estimate reconstructed from averaged (b) 2D estimates.

While the simple 2D network is computationally cheap, because it relies only on 2D convolutional layers, the resulting reconstructed 3D estimates contain artefacts, which appear when the constituent neighbouring 2D slices contain varying amounts of soot. Besides that, the reconstructed 3D estimates preserve (or slightly improve upon) the quality of 2D estimates measured by root mean squared error (RMSE), binary cross-entropy (BCE), and total amount of soot per sample (INT) defined as

$$\mathrm{BCE}(\boldsymbol{\phi}_\mathrm{s}, \widehat{\boldsymbol{\phi}}_\mathrm{s}) = -\frac{1}{n}\sum_{i=1}^{n}\left(\phi_{[hrmsi}\,\ln\widehat{\phi}_{\mathrm{s}i} + (1-\phi_{\mathrm{s}i})\ln(1-\widehat{\phi}_{\mathrm{s}i})\right),$$

$$\mathrm{RMSE}(\boldsymbol{\phi}_\mathrm{s}, \widehat{\boldsymbol{\phi}}_\mathrm{s}) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\widehat{\phi}_{\mathrm{s}i} - \phi_{\mathrm{s}i})^2},$$

$$\mathrm{INT}(\boldsymbol{\phi}_\mathrm{s}, \widehat{\boldsymbol{\phi}}_\mathrm{s}) = \frac{|\widehat{\Sigma}_\mathrm{s} - \Sigma_\mathrm{s}|}{\max\{\widehat{\Sigma}_\mathrm{s}, \Sigma_\mathrm{s}\}}, \quad \Sigma_\mathrm{s} = \sum_{i=1}^{n}\phi_{\mathrm{s}i}, \quad \widehat{\Sigma}_\mathrm{s} = \sum_{i=1}^{n}\widehat{\phi}_{\mathrm{s}i},$$

where $\phi_{\mathrm{s}i}$ is the value of volume fraction of soot in the i-th cell of total n cells in each sample.

Extended 2.5D network. In order to actually include three-dimensional data in the generation of two-dimensional estimates, the network was modified to use 3D convolutional layers, so that two additional slices are included with each input. The three slices at the input make up a 3D slab, the middle of which corresponds to the subject of estimation. The 3D reconstruction is then performed in a manner similar to that described in the previous paragraph. Note that although three-dimensional convolutional layers are used in this model, the pooling layers are, in fact, only two-dimensional.

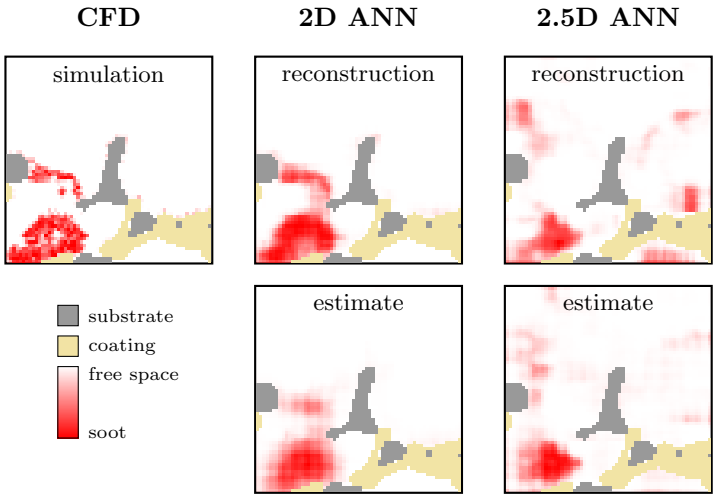

Figure 3: Two-dimensional slice from a selected validation sample – autoencoder reconstructions and model estimates based on 2D and 2.5D networks compared against CFD simulation.

This approach was expected to reduce the reconstruction artefacts described in the previous paragraph, because the additional information from neighbouring slices should, in theory, lead to smoother transitions between the layers. However, due to the generally low quality of the individual 2.5D autoencoder reconstructions, the estimates are generally worse compared to the results from the 2D network. The poor quality of autoencoder reconstructions is illustrated by a comparison between the 2D and the 2.5D networks shown in Figure 3 as well as the significantly increased autoencoder INT loss in Table 1a). With improved autoencoder performance, this model might prove usable and will be tested further in future work.

Fully 3D network. The network with 3D convolutional layers was further extended to include not just the neighbouring slices, but a full cube-shaped array of equivalent side length. The output of this network has the same shape as the input, which makes the autoencoders used fully three-dimensional.

Although this was done with the initial expectation of improving the quality of the estimates at the expense of longer network training times, we encountered two problems. First, a significant reduction in the size of the training dataset occurs, as each 3D sample corresponds to ℓ^3 data-points of the original domain, where ℓ is the sample side length, whereas in the 2D case, each sample corresponds to ℓ^2 cells. Hence, the number of distinct training samples shrinks by a factor of 3ℓ.

Another problem may be caused by the cubic reduction in the amount of information between layers of the 3D encoder. Each 3D pooling layer (with pool side length equal to 2) changes the amount of data-points by the factor of $\frac{1}{8}$, whereas a 2D pooling layer preserves $\frac{1}{4}$-th of the data-points. Consequently, for an encoder with three layers, the ratio of latent dimension to the original sample size is more than 10 times smaller for the 3D encoder compared to the 2D one.

To mitigate the described issues, we reduced the sample side length from $\ell = 64$ to $\ell = 16$ and increased the number of autoencoder inner filters from 4 to 16 (compared to the configuration used for the 2D and 2.5D networks). The resulting estimates of the 3D network are comparable to those obtained by averaging the results of the 2D network, see Figure. 4. Although there is an increase in the mean RMSE and INT losses, indicated in Table 1, the INT loss of the autoencoder reconstruction remains low, suggesting that the autoencoder is able to preserve the amount of deposited soot well.

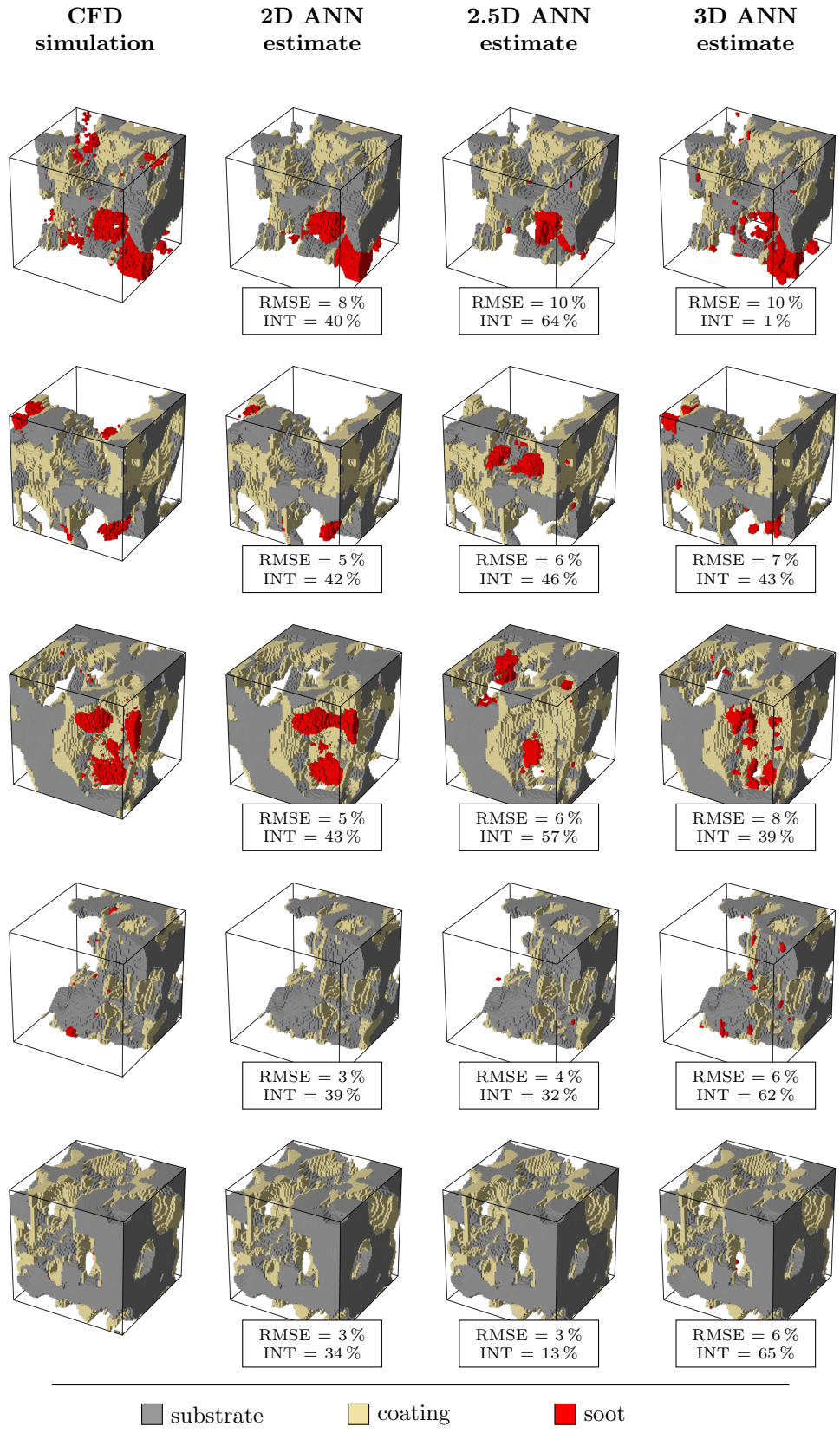

Figure 4: Examples of reconstructed 3D soot distribution estimates ($\ell = 64$) from the validation dataset.

(a) Autoencoder

	2D	2.5D	3D
# params	2,489	5,945	95,329
BCE [−]	0.02	0.06	0.02
RMSE [%]	2.6	5.1	2.8
INT [%]	3.7	43.7	4.3

(b) Estimator

	2D	2.5D	3D
# params	267,701	537,845	368,977
BCE [−]	0.02	0.05	0.04
RMSE [%]	4.1	4.8	6.7
INT [%]	41.5	51.5	55.6

Table 1: Comparison of selected metrics evaluated on the validation dataset for (a) the soot volume fraction field autoencoder reconstructions and (b) the overall estimates.

4 Conclusions

We have implemented three different proof-of-concept methods for extending our previously developed artificial neural network model, all of which allow us to obtain three-dimensional estimates of the spatial distributions of particles deposited inside a porous filter. Although the simple reconstruction method is straightforward to implement and very computationally efficient, its ability to fully exploit three-dimensional aspects of the data is rather limited, which leads to reconstruction artefacts. In comparison, the 2.5D and fully 3D networks are based on three-dimensional convolutional layers, allowing them to access the spatial aspects of the three-dimensional problem. However, these networks are more challenging to train, which is reflected in training time as well as achieved estimation quality. Nevertheless, it has been shown that the simple two-dimensional reconstruction method preserves qualities of the 2D estimates, and thus it has been established as a baseline, against which future modifications of the networks based on 3D convolutional layers may be tested.

Acknowledgment

The authors acknowledge the financial support provided by the Ministry of Education, Youth, and Sports of the Czech Republic via the project No. CZ.02.01.01/00/23_020/0008501 (METEX), co-funded by the European Union. The work was financially supported by the institutional support RVO:61388998. Special thanks belongs to Strategy AV21 of the Czech Academy of Sciences (VP20 – Water for life).

References

[1] Lao, C. T., Akroyd, J. & Kraft, M.: Modelling treatment of deposits in particulate filters for internal combustion emissions. *Progress in Energy and Combustion Science*. vol. 96: (2023). page 101043. ISSN 0360-1285. doi: 10.1016/j.pecs.2022.101043.

[2] Panchigar, D., Kar, K., Shukla, S., Mathew, R. M., Chadha, U. & Selvaraj, S. K.: Machine learning-based CFD simulations: a review, models, open threats, and future tactics. *Neural Computing and Applications*. vol. 34 no. 24: (Dec 2022). pp. 21677–21700. ISSN 1433-3058. doi: 10.1007/s00521-022-07838-6.

[3] Khýr, M., Plachá, M., Hlavatý, T. & Isoz, M.: Artificial intelligence for simulation of soot distribution inside porous filter walls. In *Topical Problems of Fluid Mechanics 2024*: TPFM: page 85–92. Institute of Thermomechanics of the Czech Academy of Sciences; CTU in Prague Faculty of Mech. Engineering Dept. Tech. Mathematics: (2024). doi: 10.14311/tpfm.2024.012.

[4] Plachá, M., Isoz, M., Kočí, P., Jones, M., Svoboda, M., Eastwood, D. & York, A.: Particle accumulation model in 3D reconstructed wall of a catalytic filter validated with time-resolved X-ray tomography. *Fuel*. vol. 356: (2024). page 129603.

[5] OpenCFD. *OpenFOAM: The Open Source CFD Toolbox. User Guide Version 1.4, OpenCFD Limited*. Reading UK: (2007).

[6] Davies, C.: Definitive equations for the fluid resistance of spheres. *Proceedings of the Physical Society*. vol. 57: (1945). page 259.

[7] Li, A. & Ahmadi, G.: Computer simulation of deposition of aerosols in a turbulent channel flow with rough walls. *Aerosol Science and Technology*. vol. 18: (1993). pp. 11–24.

[8] Chollet, F. et al. Keras. `https://keras.io`: (2015).

[9] Abadi, M. et al. TensorFlow: Large-scale machine learning on heterogeneous systems: (2015). Software available from `tensorflow.org`.

[10] Goodfellow, I., Bengio, Y. & Courville, A. *Deep Learning*. MIT Press: London, Cambridge: (2016). ISBN 9780262035613. Available from `https://www.deeplearningbook.org`.