14th World Congress on Computational Mechanics (WCCM)
ECCOMAS Congress 2020)
19–24 July 2020, Paris, France
F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)

CONVOLUTIONAL NEURAL NETWORKS FOR APPROXIMATION OF INTERNAL NON-NEWTONIAN MULTIPHASE FLOW FIELDS

Clemens Birkenmaier[1,2] **and Lars Krenkel**[1,2]

[1] Technical University of Applied Sciences Regensburg, Dept. of Biofluid Mechanics
Galgenbergstr. 30, 93053 Regensburg, Germany
clemens.birkenmaier@oth-r.de, bfm.rcbe.de

[2] Regensburg Center of Biomedical Engineering, OTH and University Regensburg
Galgenbergstr. 30, 93053 Regensburg, Germany
lars.krenkel@oth-r.de, rcbe.de

Key words: Deep Learning, Convolutional neural networks, Non-Newtonian multiphase flow

Abstract. Neural networks (NNs) as an alternative method for universal approximation of differential equations have proven to be computationally efficient and still sufficiently accurate compared to established methods such as the finite volume method (FVM). Additionally, analysing weights and biases can give insights into the underlying physical laws. FVM and NNs are both based upon spacial discretisation. Since a Cartesian and equidistant grid is a raster graphics, image-to-image regression techniques can be used to predict phase velocity fields as well as particle and pressure distributions from simple mass flow boundary conditions. The impact of convolution layer depth and number of channels of a Convolution-Deconvolution Regression Network (CDRN), on prediction performance of internal non-Newtownian multiphase flows is investigated. Parametric training data with 2055 sets is computed using FVM. To capture significant non-Newtownian effects of a particle-laden fluid (e.g. blood) flowing through small and non-straight channels, an Euler-Euler multiphase approach is used. The FVM results are normalized and mapped onto an equidistant grid as supervised learning target. The investigated NNs consist of $\mathfrak{n} = \{3, 5, 7\}$ corresponding encoding/decoding blocks and different skip connections. Regardless of the convolution depth (i.e. number of blocks), the deepest spacial down-sampling via strided convolution is adjusted to result in a $1 \times 1 \times f \cdot 2^{\mathfrak{n}}$ feature map, with $f = \{8, 16, 32\}$. The prediction performance expressed is as channel-averaged normalized root mean squared error (NRMSE). With a NRMSE of $< 2 \cdot 10^{-3}$, the best preforming NN has $f = 32$ initial feature maps, a kernel size of $k = 4$, $\mathfrak{n} = 5$ blocks and dense skip connections. Average inference time from this NN takes $< 7 \cdot 10^{-3}$ s. Worst accuracy at NRMSE of approx $9 \cdot 10^{-3}$ is achieved without any skips, at $k = 2$, $f = 16$ and $\mathfrak{n} = 3$, but deployment takes only $< 2 \cdot 10^{-3}$ s Given an adequate training, the prediction accuracy improves with convolution depth, where more features have higher impact on deeper NNs. Due to skip connections and batch normalisation, training is similarly efficient, regardless of the depth. This is further improved by blocks with dense connections, but at the price of a drastically larger model. Depending on geometrical complexity, spacial resolution is critical, as it increases the number of learnables and memory requirements massively.

1 INTRODUCTION

The flow mechanics inside membrane oxygenators (MOs), as they are used in extracorporeal membrane oxygenation (ECMO), are of great interest, since observed coagulation phenomena are believed to be mediated by prevalent flow [1–3]. However, blood flowing through tubes or channels with diameters $<$ 300μm exhibits strong non-linear effects, some of which are sometimes referred to as Fåhræus-Lindqvist-Effekt (FLE) [4, 5]. This breakdown of the Newtonian assumption is due to the actual particulate nature of blood and the properties of these particles. Blood is a multi component mixture of corpuscles, such as red blood cells (RBCs) and platelets, proteins and a water-like fluid. RBCs are highly deformable, oval biconcave disk-like corpuscles, additionally prone to biological processes. As simplification, from a fluid mechanics point of view at length scales approximately ranging from 50μm to 500μm, blood can be perceived as a mixture of particulated RBCs suspended in a Newtonian fluid phase.

Computing blood flows in complex geometries at aforementioned length scales is a non-trivial task, considering the trade off between precision and computational cost in real world applications. Using methods such as dissipative particle dynamics or smoothed particle hydrodynamics, for example, yield good results, but are far from being computational feasible in large and complex geometries as represented by MOs [6–8]. Multi-phase finite volume approaches are feasible, but still computationally costly. A trade off in terms of computational cost and information value on local viscosity and shear rate distributions could potentially be achieved by mixture theory using an Euler-Euler formulation [9, 10].

However, multi phase finite volume method (FVM) in general is still laborious, especially with a potential application in patient specific studies, require a huge number of individual parameters. The advent of deep learning (DL) approaches in various fields demonstrated broadly the advantages of inference speed from DL models. Neural network (NN) as an alternative method for universal approximation of differential equations have proven to be computationally efficient and still sufficiently accurate compared to established methods such as the FVM [11, 12]. The general advantage of employing DL models, is their deployment efficiency, outperforming classical computational methods by far. FVM are based on spatial discretisation and resulting flow fields are represented only on this discrete grid. Since a Cartesian, equidistant grid basically is a raster graphics, image-to-image regression techniques can be used to predict phase velocity fields as well as particle and pressure distributions from simple mass flow boundary conditions.

Here, the feasibility of employing convolutional neural networks (CNNs) for predicting internal non-Newtonian multiphase flows is shown by way of blood flow in wavy geometries. Different NN designs and hyperparameters are compared regarding inference duration and model accuracy.

2 METHODS

In order to test different NN designs and parameters, a set of training data capturing significant non-Newtonian effects of a particle-laden, blood like fluid flowing through small and non-straight channels is generated. Three different NN designs with different hyperparameters, such as convolution depth or number of feature maps for example, are compared. As target measure a normalized deviation averaged over all computed fields is used.

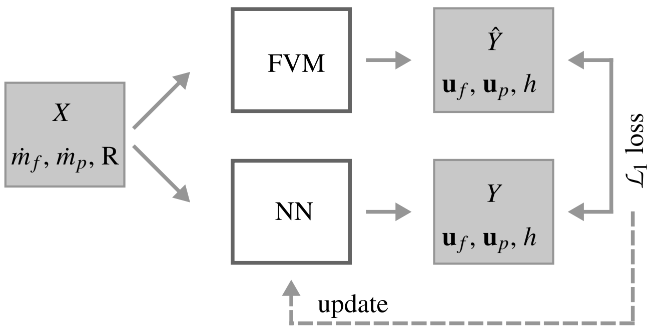

X
$\dot{m}_f$, $\dot{m}_p$, R
FVM
NN
$\hat{Y}$
$\mathbf{u}_f$, $\mathbf{u}_p$, h
Y
$\mathbf{u}_f$, $\mathbf{u}_p$, h
$\mathcal{L}_1$ loss
update

Figure 1: General supervised machine learning work flow of training a NN with pre-computed target flow fields. Inputs are the geometry R, mass flow of the fluid and particle phase $\dot{m}_f$ and $\dot{m}_p$, respectively. Outputs are velocity fields $\mathbf{u}_f$, $\mathbf{u}_p$ and particle distributions h.

2.1 Training Data Generation

Since interpolation of known data is usually much more reliable than extrapolation, especially in DL extensive training data is essential. In order to provide enough data comprising sufficient diversity and consistency, parametric training data is generated using an FVM Euler-Euler multiphase approach. Using precomputed training targets is beneficial as all data is based on the same underlying model, while still capturing relevant flow phenomena. Making machine learning (ML) models robust against noise in the training data, as it common in real world or experimental data, demands for a large amount of data or incorporated a priori knowledge incorporated in the learning process. Since the training targets computed using FVM are parametric, the entire training target computation work flow comprising geometry generation, meshing, computing the flow fields using FVM and subsequent flow field extraction is automated using Ansys Workbench Scripting.

As geometry, a two-dimensional (2D) channel with varying width is used, which is illustrated in fig. 2. The changing width is constructed from circular segments, stitched together with a tangential continuity constraint. This resulting wavy geometry segment can be used with periodic boundary conditions (BCs). Alternatively, a certain number of waves is placed behind each other, where a straight inlet with length of 50 $\times$ throat width and outlet 10 $\times$ throat width for the use without periodicity is added for improved numerical stability.

The dimensions of the wavy geometry are normalized to be fully described with only four dimensionless parameters, solely based on the particle diameter. Hence, the geometry is defined by the set

$$\mathbb{F} = \frac{D_1}{D_p} \qquad \mathbb{D} = \frac{D_2}{D_1} \qquad \mathbb{L} = \frac{L_2}{D_1} \qquad \lambda = \frac{L_1}{L_2} \tag{1}$$

where D_p is the particle diameter, $\mathbb{F}$ a factor for the throat width, D_1, D_2, L_1 and L_2 are the dimensions according fig. 2. The geometry is meshed in Ansys Meshing with triangles without inflation layers and a constant element size of $0.5\,D_p$.

The flow fields used as training target are computed with Ansys Fluent R2019v3 using an Euler-Euler two-phase model for laminar flow, similar to [10, 13]. This mixture approach models the fluid phase (blood plasma) and the particulated phase (RBCs) as interacting and interpenetrating continua, which exchange momentum in form of lift, drag or virtual mass, for example. Additionally, there are effects that can be perceived as acting only within one phase. In terms of blood, this intra-phasic effects are collision or a reversible aggregation of RBCs. The shear dependent aggregation/disaggregation of RBCs is modelled as a shear-thinning viscosity of the particulated phase with a correction for phase fraction, which aims on collisional dissipation. This particle phase viscosity as function of shear rate reads as

$$\eta'_p(\dot{\gamma}_p) = \eta_{p,\infty} + (\eta_{p,0} - \eta_{p,\infty}) \frac{1 + \log(1+\zeta)\dot{\gamma}_p}{1 + \zeta\dot{\gamma}_p} \tag{2}$$

with a correction for particle phase fraction ϕ it follows

$$\eta_p(\dot{\gamma}_p, \phi) = \eta'_p(C_0 + C_1\phi + C_2\phi^2) \tag{3}$$

where zero shear viscosity is taken to be $\eta_{p,0} = 2.0392\,\mathrm{Pa\,s}$, infinite shear viscosity $\eta_{p,\infty} = 0.0517\,\mathrm{Pa\,s}$ and the response parameter $\zeta = 48.14$ determine the shear thinning behaviour. The experimentally fitted parameters $C_0 = 1$, $C_1 = 2.5$, $C_2 = 7.6$ account for a phase fraction dependency [10]. Like other mixture properties, the resulting mixture viscosity is assumed to be a volume fraction weighted average

$$\eta = (1-\phi)\eta_f + \phi\eta_p \tag{4}$$

The inter-phase momentum transfer in terms of drag and lift is determined by a Schiller-Naumann type drag and a generalized Saffman-Mei lift. Virtual mass effects are neglected, due to the relatively small slip Reynolds numbers.

As outlet BC constant pressure, which is shared by both phases in Euler-Euler models, is used. As inlet BC, the mass flow rate for each individual phase is used, which is derived from the Reynolds number. Conversion is done by scaling with mixture viscosity given in eq. (4) and phase fraction ϕ. Hence, it follows

$$\begin{aligned} \dot{m}_f &= (1-\phi)\eta Re \\ \dot{m}_p &= \phi\eta Re \end{aligned} \tag{5}$$

For the ML approach the Reynolds number is converted to a inlet velocity scaled by the throat width

$$\mathbf{u}_{bc} = \frac{\eta Re}{\rho D_1} \tag{6}$$

The pressure based steady and laminar solver in Fluent R2019v3 is employed with solving each volumetric phase fraction independently. This improves convergence as it initially allows more deviation from continuity. Additional stability is incorporated by using an implicit under relaxation scheme that allows different relaxations factors for each equation to be set and adapt during computation automatically. As convergence criterion, the variation of the phase fraction over the computational domain is monitored and a variation below $5 \cdot 10^{-4}$ over more than 20 iterations is considered as convergence.

The FVM flow computation is performed on a triangular mesh, where the flow information is stored at grid points not being equidistantly spaced. For the use as training and validation target, the FVM fields are mapped onto an equidistant rectangular grid of same dimensions as used by the NNs under investigation. Due to the limited memory on the used graphics processing units (GPUs), the spatial resolution is restricted to 128×128. This mapping is performed in Matlab R2019b, using a natural neighbour scheme with a Delaunay triangulation for locating neighbours. As depicted in fig. 2, the wavy

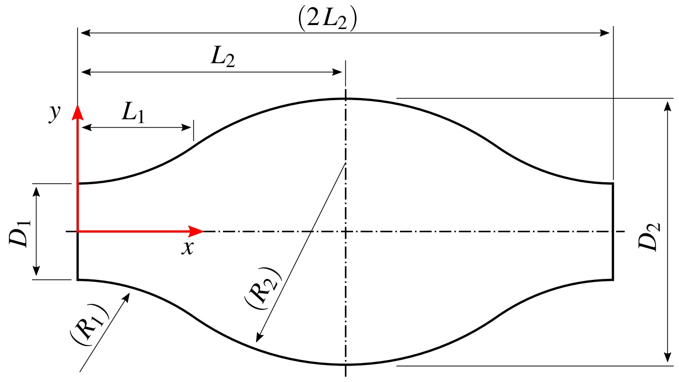


Figure 2: The non-straight confined fluid domain is parametrised according to section 2.1, depending only on the particle diameter D_p. Circular segments (radii R_1, R_2) are stitched together with tangential continuity.

geometry does not convert to a strictly convex polygon and identified neighbours might be located far away spanning acros the concave area, with values being interpolated in between. Hence, a correction by comparing the mapped field with the original geometry must be applied to ensure correct geometrical representation. In order to not being additionally limited by numerical accuracy during the training, the data is normalized to the range $[-1,1]$. As input, or BCs, for the NN, an inlet velocity $\mathbf{u}_{bc}$ as given by eq. (6) and the particle phase fraction ϕ are encoded together with the geometry in two fields with the same dimensions as the output fields.

2.2 Neural Network Designs

Considering a non-linear function in the general form of $\hat{Y} = \hat{f}(X,r)$, NNs in general can be used to approximate the true function $\hat{f}$ in the form $Y \approx f(X,r)$. Hence, the task is to get a suitable representation of $\hat{f} \to f$ capable of approximating $\hat{Y} \approx Y$ from the input X, while respecting the underlying degrees of freedom r [12]. As approximator f, NNs can be used, given a sufficient degree of freedom r and correspondingly sufficient training data. Since it is hard to balance r, which might be perceived as a design choice of the NN, and the complexity of the data $\hat{Y}$, measures against potential overfitting are necessary.

A basic design choice is to use convolutional layers as they are especially suited for working on grid-structured inputs with local context [14]. Hence, CNNs are popular for image processing, often incorporating a classification task, which usually sacrifices locality for enhancing context [15]. In order to approximate a functional $Y \approx f(X,r)$, regression rather than classification is required. For grid structured data, working on different length scales is beneficial as it combines maximum locality, i.e. where in the domain (at scale 1, $\mathfrak{n}_i = 0$), with maximum context, i.e. what are the neighbours (at scale $\frac{1}{2\mathfrak{n}}$, $\mathfrak{n}_i = \mathfrak{n}$) [15]. To represent the increasing context, an increasing number of features with progressing convolutions is required. The resulting spatial down-sapling while expanding the feature space, much like in pyramids, loosely relates to the idea of information being preserved, just like energy in any physical system [16]. Regressing a flow field, which has a certain spatial extent and a certain number of variables, from this downsampled feature rich abstract space, requires a subsequent spatial up-sampling with feature reduction in order to match the final dimensions. The downsampling and upsampling can be performed using either pooling and unpooling, or with strided convolutions and deconvolutions, respectively [17]. The term deconvolution covers an actual deconvolution as well as sometimes a transposed convolution with additional interpolation for upsampling. This upsamling interpolation, like the interpolation in unpooling, might follow a neighbouring scheme, polynomial interpolation or a learnable interpolation of any complexity. Here, deconvolution is a transposed convolution with learnable interpolation for upsampling. The repetitive up-sampling can be lossy, as information about locality is lost because neighbour relations are split. This local correspondence in formation then needs to be re-trained or simply passed through the network using paths without down-sampling [15]. Therefore, a way to preserve this locality while still incorporating context information is the use of skip connection [18].

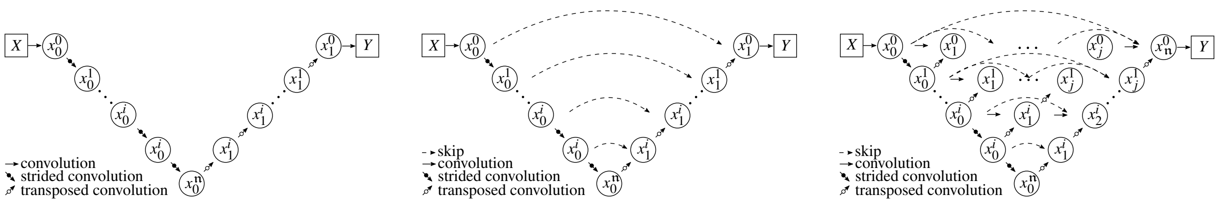
The basic NN design idea with feature expanding down-sampling, or encoding, and subsequent feature reducing up-sampling, or decoding, is referred to as Convolution-Deconvolution Regression Network (CDRN). The basic structure of skip connections in CDRNs is a well performing approach in medical image segmentation [19]. Also the idea of incorporating additional convolution paths in the skip connections, i.e. dense skips, stems from medical image segmentation [20]. Here, a straight, i.e. flat, CDRN design is compared to a CDRN with plain skip connections (skip) and a CDRN with dense skips (dense). Additionally, the impact of different hyperparameters are investigated in a simple exploration grid search. The three basic resulting designs are illustrated in fig. 3. The tested parameters are

- flat / skip / dense
- number of encoding/decoding blocks, i.e. NN depth $\mathfrak{n} = \{3, 5, 7\}$
- feature map size $f_i = f \cdot 2^{\mathfrak{n}_i}$, with $\mathfrak{f} = \{8, 16, 32\}$
- kernel size $k = \{2, 4, 8\}$

In all thee CDRN designs, each convolution or deconvolution is followed by a leaky leaky rectified linear unit (lReLU) layer with a slope of 0.1 as non-linear activation function, a channel wise batch normalisation, and a drop-out layer with 0.5 drop rate. The sequence of (de-)convolution, activation, normalization and drop out is referred to as block. Each convolution $\mathfrak{n}$ reduces the spatial footprint by a factor of 2, while the the feature map is expanded by $f \cdot 2^{\mathfrak{n}_i}$. The spatial upsampling with feature reduction in order to match space dimensions, works correspondingly, enabling skipped information to be concatenated at each individual level. Adding leakage to the rectified linear unit (ReLU) is beneficial to prevent dying neurons and ensures gradients to be propagated though the entire layer, even if a node is inactive. The batch normalisation after each convolution reduces the covariance shift and hence, it is able to reduce overfitting while preserving abstraction power of the model, and allows for higher learning rates, even if it adds two more learnables per layer [21, 22]. To further suppress over-fitting, which might happen if a powerful model is trained on only little data, a dropout layer is incorporated completing each convolution block.

2.3 Training

The basic idea of ML is to determine the parameter values of the approximating functional f, that $\hat{Y} \approx Y$, or $\min\left(|\hat{Y} - Y|\right)$ in an automated manner. As goal for this supervised learning approach, a $\mathcal{L}_1$ loss, or mean absolute error, is used, which reads as



(a) CDRN flat (b) CDRN skip (c) CDRN dense

Figure 3: The three investigated designs of CDRNs including a flat structure, additional skip connections, or dense skip connections.

$$\mathcal{L}_1(Y,\hat{Y}) = \min\left[\frac{1}{N_{mb}}\sum_{n=0}^{N_{mb}}\left(\frac{1}{N_{ch}}\sum_{i=0}^{N_{ch}}|\hat{Y}_{ni}-Y_{ni}|\right)\right] \tag{7}$$

where N_{mb} , N_{ch} is the number of observations in a mini batch and number of features, respectively. In order to evaluate the forecasting accuracy of the trained NN the deviation from its prediction Y to the true validation data $\hat{Y}$ is computed as root mean squared error (RMSE) per channel or feature, much like the loss in eq. (7). To make this measure independent from the range of data and collapse it to one measure per prediction, the RMSE is rescaled to the range of data and averaged across channels. This normalized root mean squared error (NRMSE) then reads as

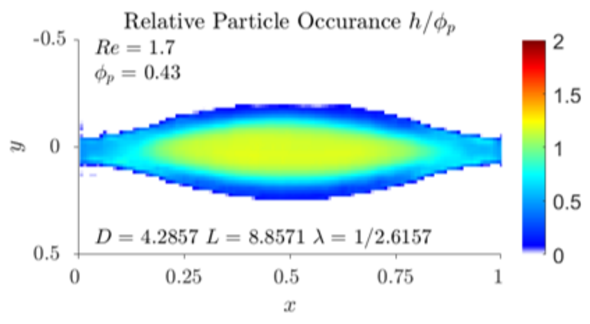
$$\text{NRMSE} = \frac{1}{N_{ch}}\sum_{n=0}^{N_{ch}}\left[\frac{\sqrt{\left(\hat{Y}_{ch}-Y_{ch}\right)^2}}{\max(\hat{Y}_{ch})-\min(\hat{Y}_{ch})}\right] \tag{8}$$

where $\max(\hat{Y}_{ch}) - \min(\hat{Y}_{ch})$ represents the range of data in each individual channel.

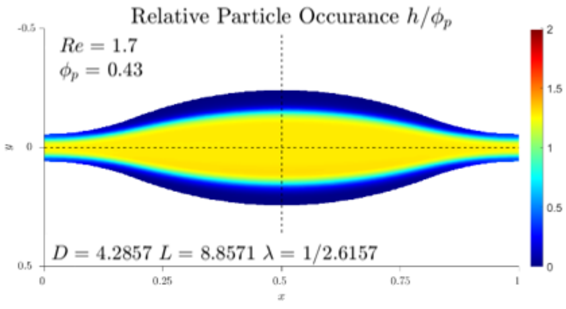
The training is performed on 2055 pre-computed parametric data sets with parameter ranges

$$\begin{aligned} 0.05 \le Re &\le 10 \\ 0.1 \le \phi &\le 0.5 \\ D_p = 7.5\cdot 10^{-6}\,&\text{m} \\ \mathbb{F} &= 10 \\ 2 \le \mathbb{D} &\le 4 \\ 2 \le \mathbb{L} &\le 4 \\ 0.2 \le \lambda &\le 0.5 \end{aligned} \tag{9}$$

using an ADAM optimizer with a mini batch size of $N_{mb} = 50$ and shuffling every epoch [23]. The data is split into training, validation and test sets at ratios of 0.7, 0.15 and 0.15, respectively. Initial learn rate is set to 10^{-3}, dropping each 100 epochs by a factor of 0.5. The training is limited to a maximum of 200 epochs or if the mean error according to eq. (7) is not improving for 20 subsequent iterations. The entire ML workflow is implemented in Matlab R2019b and performed on a Nvidia Titan X Pascal GPU, while a Nvidia RTX2080 was used for performance testing. Since the key benefit of ML models is the fast inference time, the time per prediction as average over an entire test set is given as performance measure.



(a) ML



(b) FVM

Figure 4: Resulting fields of relative volume phase fraction $\frac{h}{\phi}$. Inference from skipped CDRN with $f = 16$, $\mathfrak{n} = 5$.

3 RESULTS

The NRMSEs of the hyperparameter grid search range from approx. $9 \cdot 10^{-3}$ down to $< 2 \cdot 10^{-3}$. The accuracies in terms of NN designs are shown in fig. 5 as NRMSE vs. the number of learnables. There, the number of convolution blocks $\mathfrak{n} = \{3, 5, 7\}$ is encoded in the number of learnables.

Figure 5a illustrates the impact of the number of features $f = \{8, 16, 32\}$. The best performance for all three NN designs is achieved by $f = 32$ at approx. $2 \cdot 10^{-3}$. The dense CDRN with $f = 32$ and $\mathfrak{n} = 7$ is to large to be evaluated on a RTX2080 (8 GB memory) or Titan X (12 GB memory), even with reduced batch size.

The NN performance with different convolution kernel sizes $k = \{2, 4, 8\}$ ranges from $9 \cdot 10^{-3}$ down to approx. $3 \cdot 10^{-3}$ for several constellations, as depicted in fig. 5b.

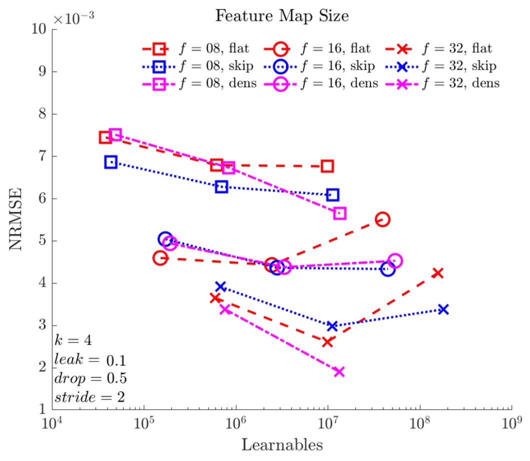
In terms of inference durations or deployment performance, which is expressed as accuracy NRMSE over time needed per prediction, is shown in fig. 6. Even the slowest prediction durations are below 10^{-2} s and thus, by far faster than a typical FVM solution run.

Exemplary, the normalized phase fraction $\frac{h}{\phi}$, resulting from FVM computation and ML inference, is shown in fig. 4. The ML model is a CDRN with skip connections, an initial feature map size $f = 16$ and $\mathfrak{n} = 5$ convolution/deconvolution blocks.

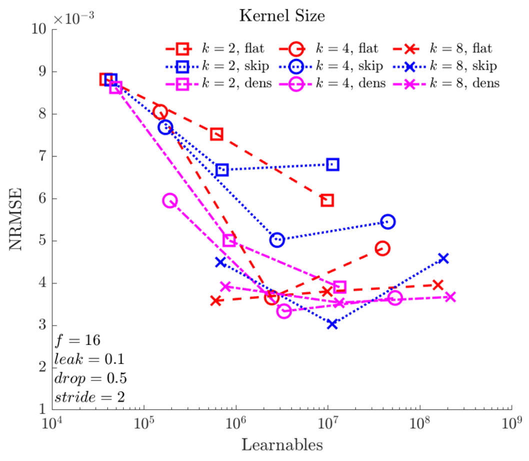
4 DISCUSSION

This feasibility study demonstrates the use of CNNs as approximating model for confined non-Newtonian multiphase flows. Even the largest models with more than 10^8 learnables compute a flow field in less than 10 milliseconds on consumer level hardware.

The resulting fields of 128×128 grid points are relatively coarse, compared to typical mesh independent FVM solutions of the Navier-Stokes equations, as depicted in fig. 4. In general, the resolution of ML computations is only limited by GPU memory. Hence, simply larger GPUs allow for better resolved fields. Another way could be to incorporate additional upsampling steps to get finer resulting fields. Since this would incorporate some kind of interpolation, it has to be taken care of discontinuities in



(a) number of features $f = \{8, 16, 32\}$



(b) convolution kernel size $k = \{2, 4, 8\}$

Figure 5: Impact of number of feature maps f and kernel size k for flat, skipped and dense CDRNs with $\mathfrak{n} = \{3, 5, 7\}$ encoder/decoder blocks on the prediction accuracy.

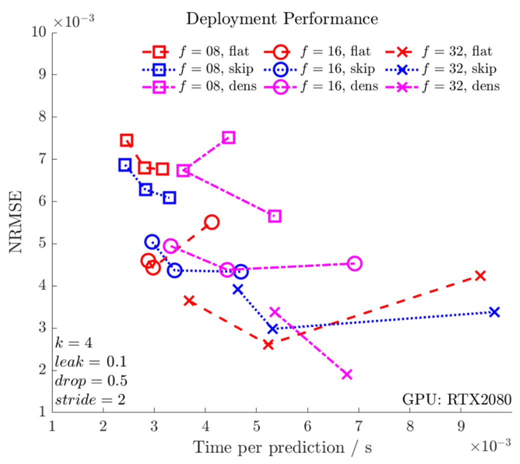
the fields as well as the geometry. If a small spatial resolution is sufficient, employing NN models is a powerful computation method, as it outperforms classical computational methods in terms of speed by far.

Maybe taking only inference durations into account is not correct, since a relevant amount of computing effort is part of the training and model prediction might only be perceived as interpolation between elaborately trained cases. If the accuracy of NN models is not sufficient for a particular application, it might also be used to create initial solutions for other, more accurate and also more trusted methods. As with most ML methods, deep field regression is prone to errors that are hard to be recognized. A way to improve this is to incorporate a priori knowledge into the training, such as continuity or impermeability of walls. An alternative is to use statistics on learnable model parameters, commonly known as explainable artificial intelligence, in order to retrieve measures for reliability of the result.

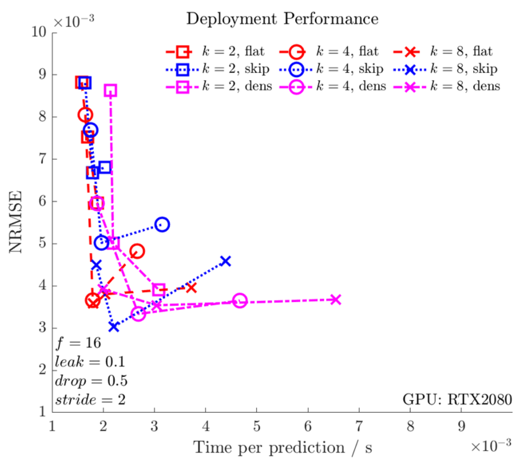
Precomputing training targets is advantageous as the data is consistent, available in almost any number, and does not comprise noise. Using experimental data as training targets in contrast would result in a ML model, representing real world data, rather than an approximation of a FVM solution. Since experimental training data is hard to provide in sufficient number, a hybrid or transfer learning approach could incorporate real world information into a DL model, pretrained on computed targets.

The challenge in NN design in general is that hyperparameters usually are not independent of each other. Therefore, a simple grid search as presented above can hardly be comprehensive and a Bayesian optimization may be more efficient.

The results of the hyperparameter search suggest that more features per layer seem beneficial in general and dense skips are beneficial if there are more features. In smaller numbers of features it depends on the number of encoder/decoder blocks, if additional skip paths improve accuracy. In terms of network depth, i.e. number of encoder/decoder blocks, it might be depending on variance in the data, as deeper networks usually have more abstraction power and therefore are more likely to be prone to overfitting.



(a) number of features $f = \{8, 16, 32\}$



(b) convolution kernel size $k = \{2, 4, 8\}$

Figure 6: Inference performance, i.e time required for one prediction, for different number of feature f and kernel size k for flat, skipped and dense CDRNs with $\mathfrak{n} = \{3, 5, 7\}$ encoder/decoder blocks.

References

[1] Tamara Steiger, Maik Foltan, Alois Philipp, Thomas Mueller, Andre Bredthauer, Lars Krenkel, Clemens Birkenmaier, and Karla Lehle. Accumulations of von willebrand factor within ecmo oxygenators: Potential indicator of coagulation abnormalities in critically ill patients? *Artif Organs*, 43:1065–1076, 2019.

[2] Han-Mou Tsai. von willebrand factor, shear stress, and adamts13 in hemostasis and thrombosis. *ASAIO Journal*, 58(2):163–169, 2012.

[3] Andrew James Doyle and Beverley Hunt. Current understanding of how extracorporeal membrane oxygenators activate haemostasis and other blood components. *Frontiers in medicine*, 5:352, 2018.

[4] R Fahraeus and T Lindquist. The viscosity of the blood in narrow capillary tubes. *Am. J. Physiol.*, 96:562–569, 1931.

[5] H Lei, D A Fedsov, B Caswell, and G E Karniadakis. Blood flow in small tubes: quantifying the transition to the non-continuum regime. *Journal of Fluid Mechanics*, 722:214–239, March 2013.

[6] N Bessonov, A Sequeira, S Simakov, Yu Vassilevskii, and V Volpert. Methods of blood flow modelling. *Mathematical Modelling of Natural Phenomena*, 11(1):1–25, 2016.

[7] Takami Yamaguchi, Takuji Ishikawa, Y Imai, N Matsuki, Mikhail Xenos, Yuefan Deng, and Danny Bluestein. Particle-based methods for multiscale modeling of blood flow in the circulation and in devices: challenges and future directions. *Annals of biomedical engineering*, 38(3):1225–1235, 2010.

[8] Prosenjit Bagchi. Mesoscale simulation of blood flow in small vessels. *Biophysical journal*, 92(6): 1858–1877, 2007.

[9] Mahrokh Bavandi and Olaf Wünsch. A study on nanoparticle transport in a micro blood vessel.

PAMM, 19(1), 2019.

[10] Wei-Tao Wu, Fang Yang, James F Antaki, Nadine Aubry, and Mehrdad Massoudi. Study of blood flow in several benchmark micro-channels using a two-fluid approach. *International journal of engineering science*, 95:49–59, 2015.

[11] Mazair Raissi. Deep hidden physics models: Deep learning of nonlinear partial differential equations. *Journal of Machine Learning Research*, 19:1–24, 2018.

[12] Nils Thuerey, Konstantin Weissenow, Harshit Mehrotra, Nischal Mainali, Lukas Prantl, and Xiangyu Hu. Well, how accurate is it? a study of deep learning methods for reynolds-averaged navier-stokes simulations. *arXiv preprint*, (1810.08217), 2018.

[13] KK Yeleswarapu, MV Kameneva, KR Rajagopal, and JF Antaki. The flow of blood in tubes: theory and experiment. *Mechanics Research Communications*, 25(3):257–262, 1998.

[14] Charu C Aggarwal. *Neural networks and deep learning*. Springer, 2018.

[15] Venkataraman Santhanam, Vlad I. Morariu, and Larry S. Davis. Generalized deep image to image regression. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5395–5405, 2017.

[16] Golnaz Ghiasi and Charless Fowlkes. Laplacian pyramid reconstruction and refinement for semantic segmentation. In *Proceedings of the 14th European Conference on Computer Vision*, volume 9907, pages 519–534, 2016.

[17] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv preprint*, (1603.07285), 2016.

[18] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[20] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. Unet++: A nested u-net architecture for medical image segmentation. *arXiv preprint*, (1807.10165), 2018.

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML15, pages 448–456, 2015.

[22] Ian Gooffellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint*, (1412.6980), 2014.