View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Repository of the Academy's Library

*Conference on Modelling Fluid Flow (CMFF'18)*
*The 17th International Conference on Fluid Flow Technologies*
*Budapest, Hungary, September 4-7, 2018*

# Predicting the flow field in a U-bend with deep neural networks

Gergely HAJGATÓ[1], Bálint GYIRES-TÓTH[2], György PAÁL[3]

[1] Corresponding Author. Department of Hydrodynamic Systems, Faculty of Mechanical Engineering, Budapest University of Technology and Economics. Műegyetem rkp. 3, H-1111 Budapest, Hungary. Tel.: +36 1 463 3097, E-mail: ghajgato@hds.bme.hu
[2] Department of Telecommunications and Media Informatics, Faculty of Electrical Engineering and Informatics, Budapest University of Technology and Economics. E-mail: toth.b@tmit.bme.hu
[3] Department of Hydrodynamic Systems, Faculty of Mechanical Engineering, Budapest University of Technology and Economics. E-mail: gypaal@hds.bme.hu

## ABSTRACT

This paper describes a study based on computational fluid dynamics (CFD) and deep neural networks that focusing on predicting the flow field in differently distorted U-shaped pipes. The main motivation of this work was to get an insight about the justification of the deep learning paradigm in hydrodynamic hull optimisation processes that heavily depend on computing turbulent flow fields and that could be accelerated with models like the one presented. The speed-up can be even several orders of magnitude by surrogating the CFD model with a deep convolutional neural network.

An automated geometry creation and evaluation process was set up to generate differently shaped two-dimensional U-bends and to carry out CFD simulation on them. This process resulted in a database with different geometries and the corresponding flow fields (2-dimensional velocity distribution), both represented on 128x128 equidistant grids. This database was used to train an encoder-decoder style deep convolutional neural network to predict the velocity distribution from the geometry.

The effect of two different representations of the geometry (binary image and signed distance function) on the predictions was examined, both models gave acceptable predictions with a speed-up of two orders of magnitude.

**Keywords: convolutional neural networks, deep learning, deep neural networks, flow field predicition, metamodelling, surrogate model**

## NOMENCLATURE

| | | |
|---|---|---|
| $v$ | $[m/s]$ | velocity |
| $x$ | $[-]$ | coordinate in the x direction |
| $y$ | $[-]$ | coordinate in the y direction |
| $x'$ | $[-]$ | running coordinate in the x direction |
| $y'$ | $[-]$ | running coordinate in the y direction |
| $\Omega$ | $[-]$ | set in the two-dimensional Euclidean space |
| $\overline{\Omega}$ | $[-]$ | complement of set $\Omega$ |
| $\partial\Omega$ | $[-]$ | boundary of $\Omega$ |

**Subscripts and Superscripts**

| | |
|---|---|
| *mag* | magnitude of a vector |
| *x* | x component of a vector |
| *y* | y component of a vector |

## 1. INTRODUCTION

The role of machine learning (and particularly deep learning) evolved in the recent years in many research fields that were mastered by highly qualified human experts traditionally. The actual trends and future possibilities are summarised by the authors of [1]. As computational resources keep growing, even larger tasks can be aided or completely surrogated by deep neural networks (DNN). Computational fluid dynamics (CFD) simulations are typical time-consuming processes in the area of fluid-dynamic design. The simulations are necessary to analyse a new idea but are resource-intensive as well, and the common numerical methods in commercial software are weakly parallelisable compared to DNNs. These drawbacks are amplified when hundreds of CFD simulations have to be computed, e.g. during an optimisation session. The need for appropriate surrogate models is growing, as optimisation-driven design is spreading.

Satisfying results were achieved by Beigzadeh et al. [2], Verstraete et al. [3] and Duvigneau and Visonneau [4] among others in surrogating CFD simulations during optimisation. Each of these solutions tries to predict the value of some integral quantity (e.g. Nusselt number, pressure loss over the domain, etc.) from some set of parameters that are modified step by step by an evolutional algorithm. Although

this approach gives satisfying results in most cases, they depend on the parametrisation and microscopic information in the flow domain vanishes.

Guo et al. [5] present the state-of-the-art result in this topic by predicting the complete velocity field from a parameter-free description of the geometry. Although the topology of the neural network is more complicated compared to [2], [3] and [4], it nearly provides the same amount of information that would be delivered by the accurate CFD simulation. The convolutional neural network (CNN) used in [5] was trained with laminar, external flows; its ability to predict the flow field under turbulent flow conditions has not been tested yet. Although the results were satisfactory, there is only a limited application area where fluid flows can be viewed as laminar.

Besides, several studies are present in the literature in the topic of machine learning and turbulence. Singh et al. [6] try to improve a simple turbulence model to get better CFD simulation results in the flow fields computed around airfoils. In this case, the knowledge-based model is augmented with a part that is acquired by machine learning techniques. In contrast, the authors of [7] keep the former knowledge-based model intact and machine learning is used to predict the discrepancy of the model from accurate simulation data. In [8] deep learning techniques are utilized instead of conventional machine learning techniques. The authors aim at predicting the anisotropic properties of turbulence with granting Galilean invariance at the same time. Lguensat et al. [9] detected large-scale eddies by DNNs. These studies show that machine learning techniques have the potential to predict certain properties of turbulence.

This paper presents a convolutional neural network inspired by Guo et al. [5], wherewith the velocity field can be predicted for internal, turbulent flows under specific boundary conditions and in a given geometry with a bearable error. The speed-up is two orders of magnitude compared to the CFD simulation if only one geometry has to be evaluated. The benefit would grow further by increasing the number of geometries to evaluate, e.g. in optimisation sessions.

## 2. NEURAL NETWORKS AS SURROGATE MODELS

Feed-forward artificial neural networks are universal function approximators as stated by Hornik et al. [10] and Cybenko [11]. Although the model is biologically inspired, it is superfluous to examine the neuron nerve cell system of an animal to get an understanding of artificial neural networks. Moreover, artificial and real neural networks are similar only from a sufficiently large distance of view.

The topology of a neural network is flexible enough to get it tailored to specific problems, thus many types of building blocks exist. The present work relies on multi-layer perceptrons and convolutional layers whose basics are described here.

### 2.1. Multi-layer perceptron

One of the basic applications of neural networks is the prediction of valuable information from the corresponding features of an entity. To accomplish this goal, the features are quantified and multiplied with a specific weight that measures the importance of that feature. The sum of these products is the signal that is fed to an activation function, which can be, e.g. sigmoid, hyperbolic tangent or rectified linear unit [12]. Considering that the weights are known, the output of the activation function is suitable for classification or regression.

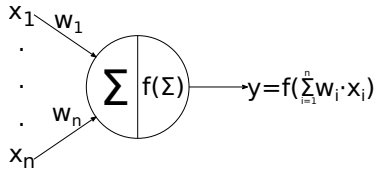


**Figure 1. A single perceptron, where $x_i$ is the $i$th feature of the entity and $w_i$ is the weight belonging to the $i$th feature.**

This simple setup is called perceptron[1] (depicted in Figure 1), while the combination of the summation and the activation function is called neuron. As Minsky [13] stated, a standalone perceptron is unable to cope with situations when the features have to model exclusive-or logic. To overcome this drawback, neurons are arranged into layers and stacking these layers results in the multi-layer perceptron.

The neurons in a topology described above can be organized as shown in Figure 2. The first layer is called the input layer, thereafter come the so-called hidden layers and finally the output layer. As all of the neurons on a layer are connected to all of the neurons of the next layer, this topology is also called densely or fully connected. When these layers are stacked different level of abstractions are presented in different depths of a proper network topology that is appropriately trained [14]. This is the reason why it is a common practice to apply many layers for complex tasks that lead to deep neural networks and hence comes the name deep learning.

To be able to give good predictions, the weights of the neurons have to be properly aligned. The appropriate weights are computed during the training of the network that is an iterative process and is not discussed here. The interested readers are encouraged to read the related chapters of the book of Yaser et al. [15] and the work of LeCun et al. in [16].

### 2.2. Convolutional layers

Densely connected layers are inefficient in pattern recognition tasks because they do not have a sense on the data spatiality. Convolutional layers learn (during training) and apply (during inference) filters on the input (or on the preceding layer). These

[1] The original perceptron uses sigmoid activation function.

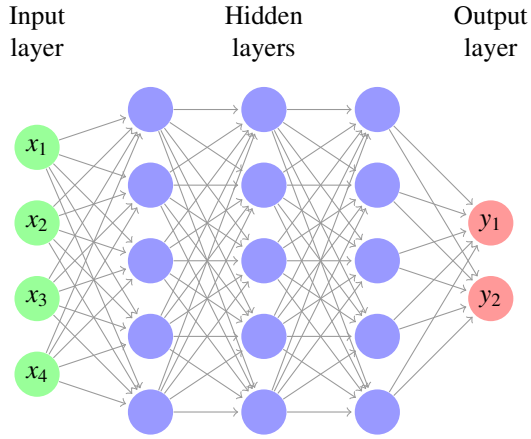


**Figure 2. Example of a multi-layer perceptron with 4 input features, 3 hidden layers and 2 outputs.**

filters are cross-correlated with the input, so the output of a filter is the sum of the Hadamard product of the filter and the appropriate parcel of the data where the filter is applied. This leads to the reduction of the dimensionality as depicted in Figure 3. As the size of the filter is intended to be much smaller as the size of the input layer the filter has to be slid over the input. The stride of a filter counts the number of elements (pixels, neurons, etc.) wherewith the filter is slid over the domain during cross-correlation. In such cases when the filter size with the prescribed stride does not match the size of the data, the original data is augmented and the size of the augmentation is called padding.

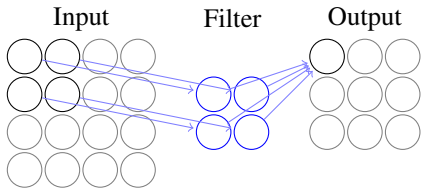


**Figure 3. Simplified example of a convolutional layer.**

Training and applying filters on the preceding layer is conventionally called convolution in deep learning and is very effective in representation learning, and thus, pattern recognition tasks even when images or sequential data are the origin [17]. Deep convolutional neural networks are the state-of-the-art solutions to object detection. See e.g. the ImageNet Challenge winner works of Krizhevsky et al. [18], Szegedy et al. [19]. and He et al. [20].

### *2.3. Motivation*

The present work was motivated by the results achieved recently by deep convolutional neural networks in pattern recognition and object detection. Guo et al. [5] used a deep convolutional neural network to predict the laminar flow field „seeing" only the geometry of the fluid domain. The present work relies on the assumption that convolutional networks have the power to recognize turbulent structures in the flow field and thus make acceptable predictions of the velocity field in turbulent internal flows as well.

## 3. METHODOLOGY

There are no preprocessed datasets publicly available wherewith convolutional neural networks could be trained to predict velocity fields in turbulent flows. Therefore, the data generation and the preprocessing were also part of the present work. This chapter describes the numerical model of the basic fluid dynamics problem. The preprocessing of the data wherewith the convolutional neural network was fed and the topology of the CNN itself was the result of a hyperparameter optimisation. The utilized software and packages are also mentioned in the last subsection.

### *3.1. Fluid dynamics problem*

The fluid dynamics problem is based on the one presented by Verstraete et al. in [3]. The original paper focuses on the shape optimisation of the geometry, thus the authors of [3] connected a CFD model with an evolutional algorithm to find the shape with the least pressure loss. The authors of the present article rebuilt the CFD model according to [3] but it was used to generate random shapes wherewith the convolutional neural network could be trained in this case. Besides, the different objective permitted minor simplifications in the simulation.

#### 3.1.1. Geometry

The geometry is a curved channel that is a common part of cooling systems. The planar view of the conventional shape is depicted in Figure 4. This shape can be altered in the plane as the geometry is built up from Bézier-curves.[2] A distorted bend is also depicted in Fig. 4.

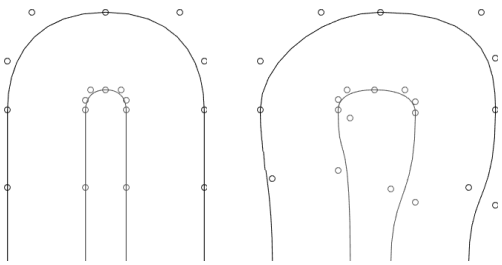

**Figure 4. Planar view of the of the U-bend. Conventional shape in the left, a distorted shape in the right with circles denoting the endpoint tangents of Beziér curves.**

The spatial geometry would be the straight extrusion of the planar view in the $z$ dimension but the 3 dimensionality was omitted and the flow was considered planar. Additional straight parts on the up- and downstream sides were applied to handle the

[2]The actual parameterization can be found in [3].

boundary conditions of the simulation in a proper way. The lengths of these legs were chosen such that the boundaries could not affect the flow pattern in the U-turn.

#### 3.1.2. Boundary conditions and simulation setup

In the physical model constant temperature, incompressible air with a kinematic viscosity of $1.5 \cdot 10^{-5}\ m^2/s$ flowed through the channel. The medium entered the domain on the left side with a constant uniform velocity of $8.8\ m/s$ and left it on the right side through a constant pressure outlet of $0\ Pa$ as shown in Fig. 5. Although the velocity profile was uniform at the inlet, it got fully developed in the straight section before the curve.

The initial channel width was 75 *mm*, but it was subject to change in the vicinity of the turn due to the parametrisation. The Reynolds number was 44000 in the initial geometry which clearly classifies the flow as turbulent. Moreover, the fluid domain was bounded with hydraulically smooth walls, the bottom and top surfaces of the geometry were treated as symmetries.

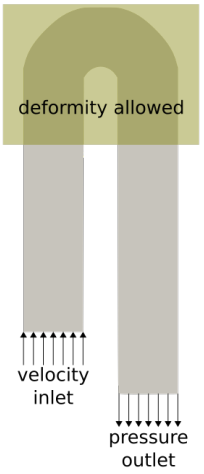


**Figure 5. Position of inlet and outlet boundaries and the area where the geometry was a subject to change.**

The geometry was discretised with a structured numerical grid with near-wall refinement. The total number of elements is 14406 and the resolution of the boundary layer was aligned to apply the use of the high-Re k-$\varepsilon$ turbulence model with the corresponding *epsilonWallFunction* wall function of `OpenFoam`. It also means that the anisotropy properties of turbulence were not considered in this study.

The numerical model was solved with the *simpleFoam* solver of `OpenFOAM` with a convergence criterion of $10^{-4}$ for the residuals.

### 3.2. Data processing

The CFD simulation discussed in Section 3.1 was used to generate data for training the convolutional neural network. The CNN aimed at predicting the velocity field in a given geometry, thus these data had to be extracted from the CFD simulations. First, the data from the numerical grid of the CFD were converted to a rectangular, equidistant 128x128 mesh to make them applicable to the convolutional neural network. The subsequent steps were taken as follows.

#### 3.2.1. Geometry

[5] suggests two different kinds of geometry representation for predicting velocity fields around immersed bodies. Both of these methods were applied in the present work for internal flows.

The simpler method is to convert the geometries to binary images, where the value of a grid node can be 1 or 0 only according to whether it is inside the fluid domain or not, respectively. A binary image is shown in the left side of Figure 6.

A more sophisticated approach is to calculate the signed distance field in the fluid domain, that is a common technique in computer graphics, see e.g. [21]. *Distance field* relates to a variable field where the value in a grid point refers to the distance between the actual point and the closest point on the boundary, while the term *signed* means, that the sign of the value refers to whether the actual point is inside or outside the boundary. This approach was used with a slight modification here, as the signed distance function was positive inside the fluid domain and 0 everywhere else.

The applied method is formulated in Eq. 1, considering the fluid domain as a set $\Omega$ in the two-dimensional Euclidean space, $x$ and $y$ as arbitrarily chosen coordinates in the space and $x'$ and $y'$ as running variables on the boundary of the fluid domain.

A geometry converted to a signed distance field is depicted on the right hand side of Fig. 6 as well.

$$f(x,y) = \begin{cases} \min\limits_{(x',y')\in\partial\Omega} |(x,y)-(x',y')| & \text{if } x,y \in \Omega \\ 0 & \text{if } x,y \in \overline{\Omega} \end{cases} \tag{1}$$

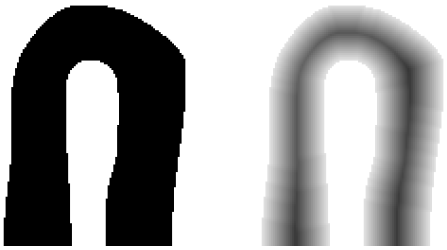

**Figure 6. Geometry represented as binary image (left) and signed distance field (right).**

Both representations were available on a 128x128 equidistant mesh as mentioned earlier and images were standardised based on the data in the training set.

#### 3.2.2. Velocity field

To make the velocity vectors reconstructable, the velocity fields in the $x$ and $y$ directions were extracted separately. Fields were resampled on the same equidistant, rectangular grid as the representations of the geometry with a resolution of 128x128 grid points. Sample velocity fields are depicted in Figure 7. The background (region outside the fluid domain) was set to zero, but the scale is different for the $v_x$ and $v_y$ fields.

**Figure 7. Velocity $x$ and $y$ components in the left and in the right, respectively. The figure serves as illustration only, thus scales were omitted.**

### 3.3. Convolutional neural network

The convolutional neural network presented by Guo et al. [5] was considered as a baseline for the present work but some parts of the network topology were subject to hyperparameter optimisation. The final architecture is depicted in Figure 8. where *conv* stands for convolution, *deconv* for transposed convolution and *dense* refers to a densely connected layer. The properties of the convolutional/deconvolutional layers are summarized as follows: {depth}@{input_x}x{input_y}, f:{filter_x}x{filter_y}, s:{stride_x}x{stride_y}, z:{padding_x}x{padding_y}. Please note that the filter properties of a convolutional/deconvolutional layer are denoted at the preceding layer since the filter is applied there. In case of the dense layer only the number of neurons are indicated.

This is an autoencoder-style neural network with convolutional-deconvolutional layers with a dense layer as a bottleneck. The encoder part takes the geometry represented as binary image or signed distance field on a 128x128 grid, while the outputs of the network are the standalone velocity fields in $x$ and $y$ directions. Thus, the decoder part is divided after the dense layer for predicting $v_x$ and $v_y$ fields. The output is masked with the binary image of the input to omit the prediction coming from the CNN outside of the fluid domain. It has been done so regardless of whether the input itself was a binary image or a signed distance field.

The error of the prediction was defined as the root mean squared error between the predicted and the CFD simulated (considered here as ground-truth) velocity fields. The mask was applied to the output of the CNN in the training phase as well, thus the network could not reach small prediction error values by predicting the constant field outside the fluid domain, but it was forced to make better predictions inside the fluid domain.

A residual connection is presented between the first convolutional and the second deconvolutional layer means, that the output of the first convolutional layer is added to the output of the second deconvolutional layer in each branch. This type of connection is discussed by He et al. in [20] as a technique to improve accuracy and convergence of CNNs.

The optimisation of the network's weights was carried out with Adam optimiser, introduced by Kingma and Ba in [22].

### 3.4. Technical details

The geometry was built and meshed in the freely accessible `gmsh` [23] mesher software, the flow pattern was solved with the open-source numerical solver `OpenFOAM`, the postprocessing was carried out in `ParaVIEW` scripted in `Python`.

Training and inference with neural network were done in `Keras` [24] deep learning framework, while hyperparameters were optimised with `Hyperopt` using the `Hyperas` frontend. Both the hyperparameter optimisation and the training were carried out on an NVIDIA Titan Xp GPU with on-line data augmentation running on an Intel i5-7500 CPU.

## 4. EXPERIMENT SETUP AND RESULTS

4500, 900 and 900 U-bend variants were generated randomly with uniform distribution for training, validation and test sets, respectively. The training and validation data were generated such, that the geometries could not evolve to the upper left corner of the available space. The flow stagnates in this area in most geometry variants if there is a bulge there. Similar flow patterns can develop in other regions of the geometry, so if the CNN generalises well, it should learn to predict the flow field in the restricted area as well. To evaluate the generalization capabilities of the proposed system geometry was allowed to grow in the upper left corner in the case of test sets. The restricted area is depicted in Figure 10.

Moreover, a data augmentation function was utilized that was allowed to rotate images by 90 degrees counter-clockwise and/or flip them upside down with a probability of 0.44. The transformation was carried out on both the geometry representation and on the velocity fields. By applying this technique the neural network was forced to gain inference connected to the shape features instead of the absolute location of them. Please note that this type of augmentation makes sense here only because of the restricted type of geometries; it is not a general approach for surrogating CFD models.

Summarising the steps, the experiment was conducted as follows.

1. The parameter lists for training, validation and test sets were generated.

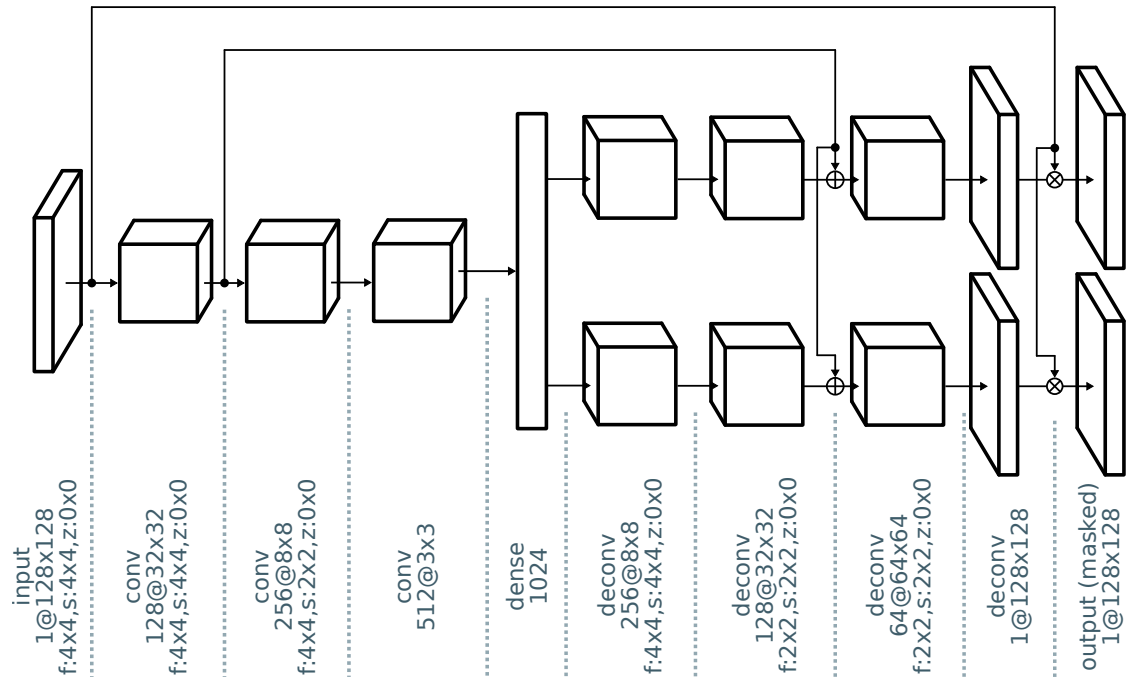


**Figure 8. Architecture of the artifical neural network.**

2. The geometries were built and the numerical simulations were performed.

3. Data were post-processed to get the binary images and signed distance fields as geometry representations.

4. The hyperparameters of the convolutional neural network were optimised with the training data.

5. The best neural network topology was trained with the training data.

6. Tests were performed on the test data.

The signed distance field was chosen as input during the hyperparameter optimisation as it is stated a better performer in [5] than binary image. The best topology was trained first with signed distance field input with and without data augmentation, moreover with and without residual connection. Thereafter, the best combination (both data augmentation and residual connection) was trained and tested with binary image input as well. Results are summarised in Table 1.

Both data augmentation method and residual connection in the network contributed to a better prediction on the test set. However, it is not clear which geometry representation is better for convolutional neural networks. Treating the geometry as binary image was better in this case but it contradicts the results of [5]. Likely, the distance function could not reveal any additional information here compared to binary image as geometries were very similar.

**Table 1. Average prediction RMS error on the test velocity fields. (RC – residual connection, DA – data augmentation, BIG – binary image, SDF – signed distance field)**

| Training method | Prediction error, m/s |
|---|---|
| SDF, no RC, no DA | 0.1071 |
| SDF, RC, no DA | 0.0820 |
| SDF, no RC, DA | 0.0673 |
| SDF, RC, DA | 0.0350 |
| BIG, RC, DA | 0.009 |

The time requirements of the prediction and the simulation are stated in Table 2. for 1 and for 900[3] evaluations carried out on the hardware described in Section 3.4.

**Table 2. Computational time for the CNN and for the CFD.**

| Utilized hardware | time, s |
|---|---|
| 1xCNN prediction on CPU | 0.826 |
| 1xCNN prediction on GPU | 0.0325 |
| 1xCFD simulation on 4 CPU cores | 12.6 |
| 900xCNN prediction on GPU | 7.45 |
| 900xCFD simulation on 4 CPU cores | 11 340.0 |

Results of test set are depicted in Figure 9., where such a geometry is shown, where the fluid domain is expanded to the area, that was restricted for the geometries in the training set. Although a

[3]Such amount of evaluation is common in optimisation-driven design.

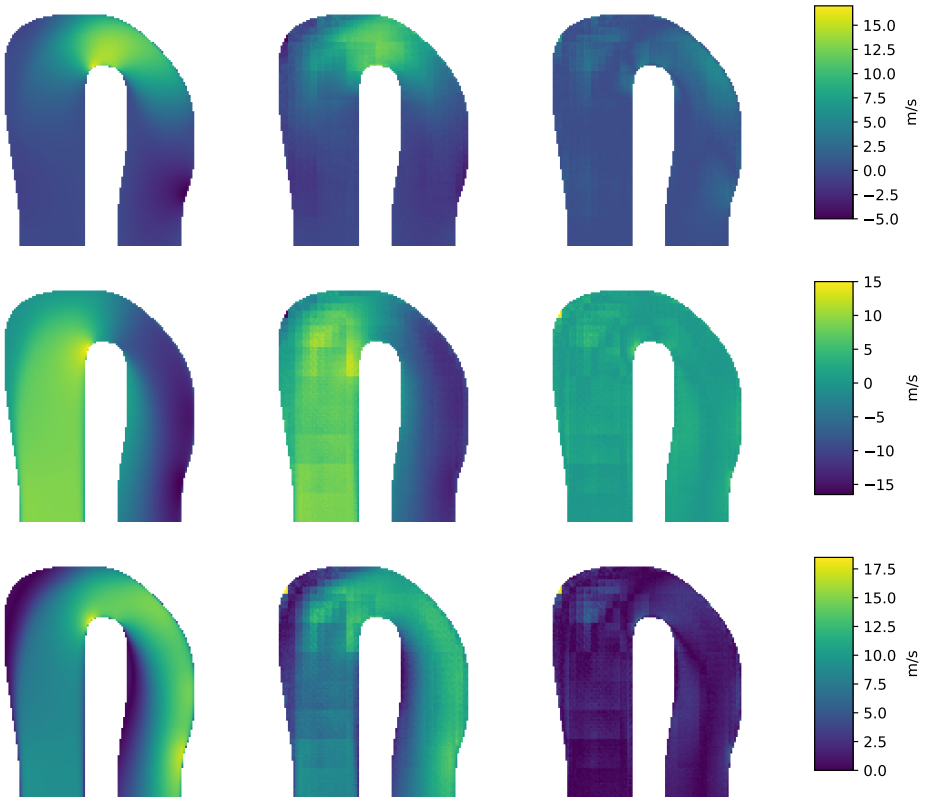


**Figure 9. Velocity fields belonging to a geometry from the test set. From top to bottom: $v_x$, $v_y$ and $v_{mag}$ fields. From left to right: CFD result, CNN prediction and absolute difference.**

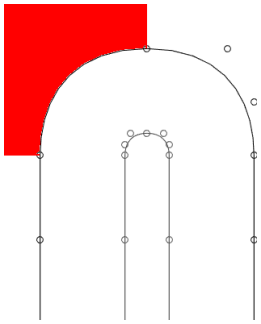

**Figure 10. U-bend with the restricted area marked red for the geometries in the training set**

small discrepancy is seen in the upper left corner the CNN was able to predict the flow stagnation in the problematic area, implying that the network was able to learn some generalised knowledge about this phenomenon.

## 5. CONCLUSION AND OUTLOOK

The presented convolutional neural network was successful in learning abstractions from velocity fields by being able to predict the stagnating flow in a zone that was not seen during the training. The CNN is two orders of magnitude faster than the CFD simulation when just one geometry is evaluated. The gain grows when a larger number of geometries are evaluated as the prediction is highly parallelisable even on a consumer-grade GPU.

This suggests that using CNNs to surrogate CFD models can be advantageous in optimisation tasks, based on some kind of evolution algorithm as a population of geometries can be evaluated at once. In cases, where the accuracy is crucial, the convolutional network could be utilised to warm-start the accurate CFD simulation.

On the other hand, although the presented convolutional neural network performed well on the specific case, its scope of usability is limited owing to the similar patterns presented in the training data. Further work has to aim at improving generalisation to make the benefits of CNN-based surrogate modelling available without excessive preliminary computations.

## ACKNOWLEDGEMENTS

The research presented in this paper has been supported by the European Union, co-financed by the European Social Fund (EFOP-3.6.2-16-2017-00013) and by the BME-Artificial Intelligence FIKP grant of EMMI (BME FIKP-MI/SC). We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

## REFERENCES

[1] Karpatne, A., Atluri, G., Faghmous, J. H., Steinbach, M., Banerjee, A., Ganguly, A.,

Shekhar, S., Samatova, N., and Kumar, V., 2017, "Theory-Guided Data Science: A New Paradigm for Scientific Discovery from Data", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 29 (10), pp. 2318–2331.

[2] Beigzadeh, R., Rahimi, M., Jafari, O., and Alsairafi, A. A., 2016, "Computational fluid dynamics assists the artificial neural network and genetic algorithm approaches for thermal and flow modeling of air-forced convection on interrupted plate fins", *Numerical Heat Transfer, Part A: Applications*, Vol. 70 (5), pp. 546–565.

[3] Verstraete, T., Coletti, F., Bule, J., Vanderwielen, T., and Arts, T., 2013, "Optimization of a U-Bend for Mnimal Pressure Loss in Internal Cooling Channels – Part I: Numerical Method", *Journal of Turbomachinery*, p. 10.

[4] Duvigneau, R., and Visonneau, M., "Hybrid genetic algorithms and artificial neural networks for complex design optimization in CFD", *International Journal for Numerical Methods in Fluids*, Vol. 44 (11).

[5] Guo, X., Li, W., and Iorio, F., 2016, "Convolutional Neural Networks for Steady Flow Approximation", *KDD 2016 Conference proceedings: ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, p. 10.

[6] Singh, A. P., Medida, S., and Duraisamy, K., "Machine-Learning-Augmented Predictive Modeling of Turbulent Separated Flows over Airfoils", *AIAA Journal*, (7), pp. 2215–2227.

[7] Wang, J.-X., Wu, J.-L., and Xiao, H., 2017, "Physics-informed machine learning approach for reconstructing Reynolds stress modeling discrepancies based on DNS data", *Phys Rev Fluids*, Vol. 2, p. 034603.

[8] Ling, J., Kurzawski, A., and Templeton, J., 2016, "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance", *Journal of Fluid Mechanics*, Vol. 807, pp. 155–166.

[9] Lguensat, R., Sun, M., Fablet, R., Mason, E., Tandeo, P., and Chen, G., 2017, "EddyNet: A Deep Neural Network For Pixel-Wise Classification of Oceanic Eddies", *CoRR*, Vol. abs/1711.03954.

[10] Hornik, K., Stinchcombe, M., and White, H., "Multilayer feedforward networks are universal approximators", *Neural Networks*, (5), pp. 359–366.

[11] Cybenko, G., "Approximation by superpositions of a sigmoidal function", *Mathematics of Control, Signals, and Systems*, (4), pp. 303–314.

[12] Nair, V., and Hinton, G. E., 2010, "Rectified Linear Units Improve Restricted Boltzmann Machines", *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ISBN 978-1-60558-907-7, ICML'10, pp. 807–814.

[13] Minsky, M., and Papert, S., 1969, *Perceptrons: An Introduction to Computational Geometry*, MIT Press, Cambridge, MA, USA.

[14] LeCun, Y., Bengio, Y., and Hinton, G., "Deep learning", *Nature*, (7553), pp. 436–444.

[15] Yaser, S. A.-M., Magdon-Ismail, M., and Lin, H., *Learning from Data: A Short Course*, ISBN 9781600490064.

[16] LeCun, Y., Bottou, L., Orr, G. B., and Müller, K. R., 1998, *Efficient BackProp*, Springer Berlin Heidelberg, Berlin, Heidelberg, ISBN 978-3-540-49430-0, pp. 9–50.

[17] LeCun, Y., and Bengio, Y., 1995, "Convolutional Networks for Images, Speech, and Time-Series", M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks*, MIT Press.

[18] Krizhevsky, A., Sutskever, I., and Hinton, G. E., 2012, "ImageNet Classification with Deep Convolutional Neural Networks", *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pp. 1097–1105.

[19] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S. E., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A., 2014, "Going Deeper with Convolutions", *CoRR*, Vol. abs/1409.4842.

[20] He, K., Zhang, X., Ren, S., and Sun, J., "Deep Residual Learning for Image Recognition", *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, ISBN 978-1-4673-8851-1, pp. 770–778.

[21] Bridson, R., *Fluid Simulation for Computer Graphics, Second Edition*, ISBN 9781482232844.

[22] Kingma, D. P., and Ba, J., 2014, "Adam: A Method for Stochastic Optimization", *CoRR*, Vol. abs/1412.6980.

[23] Geuzaine, C., and Remacle, J.-F., "Gmsh: a three-dimensional finite element mesh generator with built-in pre-and post-processing facilities", *Int J Numer METHODS Eng*, pp. 1–24.

[24] Chollet, F., et al., 2015, "Keras", `https://github.com/keras-team/keras`.