DeepCFD: Efficient Steady-State Laminar Flow Approximation with Deep Convolutional Neural Networks

Mateus Dias Ribeiro [a,*], Abdul Rehman [b], Sheraz Ahmed [a], Andreas Dengel [a]

[a] German Research Center for Artificial Intelligence (DFKI), Kaiserslautern, Germany
[b] NUST School of Electrical Engineering and Computer Science, Pakistan

Abstract

Computational Fluid Dynamics (CFD) simulation by the numerical solution of the Navier-Stokes equations is an essential tool in a wide range of applications from engineering design to climate modeling. However, the computational cost and memory demand required by CFD codes may become very high for flows of practical interest, such as in aerodynamic shape optimization. This expense is associated with the complexity of the fluid flow governing equations, which include non-linear partial derivative terms that are of difficult solution, leading to long computational times and limiting the number of hypotheses that can be tested during the process of iterative design. Therefore, we propose DeepCFD: a convolutional neural network (CNN) based model that efficiently approximates solutions for the problem of non-uniform steady laminar flows. The proposed model is able to learn complete solutions of the Navier-Stokes equations, for both velocity and pressure fields, directly from ground-truth data generated using a state-of-the-art CFD code. Using DeepCFD, we found a speedup of up to 3 orders of magnitude compared to the standard CFD approach at a cost of low error rates.

Keywords: CFD, Deep Learning, U-Net

[*] Corresponding author
Email address: mateus.dias_ribeiro@dfki.de (Mateus Dias Ribeiro)

1. Introduction

Computational Fluid Dynamics (CFD) simulations provide detailed description of flow properties of interest for engineering by numerically solving a set of governing Navier-Stokes equations. Their solution, however, can be considerably expensive due to the complexity of the fundamental physics associated with this problem. This expense is a major limitation for the development of products in a wide range of applications, such as aerodynamic design optimization and fluid structure interaction [1, 2].

For some engineering applications, the high cost of CFD solutions can be mitigated if certain conditions hold true. For example, if the Reynolds number (relationship between inertial and viscous forces) is low enough, the flow will be laminar, which means that the fluid particles flow along parallel layers with no cross-currents perpendicular to the direction of the flow [3], as shown in Figure 1. Moreover, if the flow achieves a state in which any given property, such as velocity and pressure fields, changes along space but not with time, the problem can be treated as a non-uniform steady laminar flow. In this case, solutions will depend solely on boundary conditions and geometry of the problem. Practical examples in which these conditions can be met in engineering applications are vast, such as flows within nozzles, turbines, compressors, and heat exchangers, as well as in aerodynamic flows under certain conditions, among others [4, 5, 6].

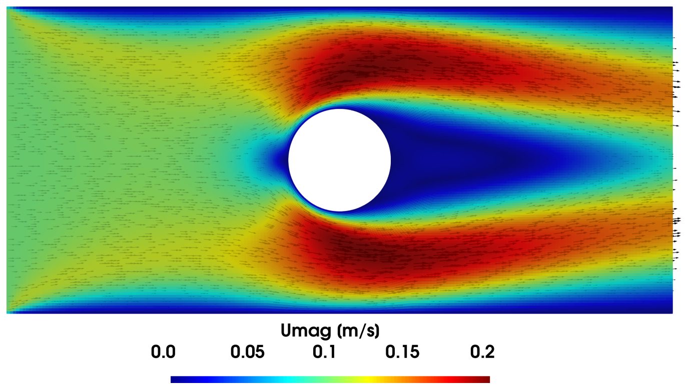

Figure 1: Example of 2D non-uniform steady laminar channel flow around a cylinder.

During early stages of a product design, such as any of the applications mentioned above, it is important for engineers to test as many hypotheses as possible in order to find a solution that maximizes the performance and/or efficiency of the product being developed. Although state-of-the-art CFD solvers can deliver highly accurate results, the time required to obtain a solution inhibits extensive iterative design due to the expensive resources necessary to run these codes. Therefore, the use of data-driven machine learning approaches to generate an accurate approximation of these simulations using a fraction of their resources is very appealing [7]. The potential of these approximated solutions for accelerating the results at a cost of low error rates [8] can lead to more efficient development of products dependent on CFD research. Finally, these approaches may also be implemented together with physics-driven techniques in order to bound the model prediction within the physical constraints of the problem [9, 10, 7].

The literature on data-driven methods for CFD provides several contributions to the field of machine learning models for fluid flow predictions. Pioneering works from [11] and [8] show how neural networks can be employed for deriving closure terms for turbulence modeling and for predicting the velocity magnitude field in steady-state flows. Moreover, other recent works from distinct sources provide further contributions to relevant fields in a wide range of applications, such as physics-informed neural networks, airfoil design optimization, acceleration of sparse linear system solutions, etc [12, 13, 14, 15].

In this paper, we present DeepCFD: a deep learning model based on CNN architectures to provide an approximated solution for both velocity and pressure fields by feeding it ground-truth data of a channel flow around randomly shaped obstacles, generated by a state-of-the-art CFD solver. The following summarizes the main contributions of this work:

1. We propose a CNN-based surrogate CFD model for 2D non-uniform steady-state laminar flows that provides a solution with up to 3 orders of magnitude speedup at a cost of low error rates.
2. We extend previous efforts from other researchers, such as in Guo et al. [8], by providing complete solutions of all components of the velocity field and the pressure field. A complete solution for velocity and pressure is essential for engineers to develop products that interact with a given flow field, or to account for the transport of a scalar of interest in industrial flows.
3. Provide the code and dataset used in this work for the general public to contribute with further expansion of the field of data-driven models for CFD: `https://github.com/mdribeiro/DeepCFD`

2. Methodology

In this section, the traditional CFD approach is presented, followed by the proposed surrogate DeepCFD approach based on CNN architectures. The traditional CFD approach provides the ground-truth data for training DeepCFD.

2.1. CFD Approach: OpenFOAM

The incompressible transient two dimensional Navier-Stokes equations for mass (1) and momentum (2) conservation read:

$$\nabla \cdot \mathbf{u} = 0 \tag{1}$$

$$\rho \left(\frac{\partial}{\partial t} + \mathbf{u} \cdot \mathrm{div} \right) \mathbf{u} = -\nabla p + \nabla \cdot \tau + \mathbf{f} \tag{2}$$

in which u is the velocity field (with x and y components for 2 dimensional flows), ρ is the density, p is the pressure field, τ is the stress tensor, and f represents body forces, such as gravity.

If a non-uniform steady-state flow condition is assumed, the accumulation term (time t dependence term) is dropped, and the momentum equation can be rewritten for velocity components u_x (3) and u_y (4) as:

$$u_x \frac{\partial u_x}{\partial x} + u_y \frac{\partial u_x}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu \nabla^2 u_x + g_x \tag{3}$$

$$u_x \frac{\partial u_y}{\partial x} + u_y \frac{\partial u_y}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu \nabla^2 u_y + g_y \tag{4}$$

in which g represents the gravitational acceleration and ν the dynamic viscosity of the fluid. The terms on the left-hand side of these equations account for the convective transport, whereas the terms on the right-hand side account for the pressure coupling and diffusive transport.

The above equations are solved numerically using the *simpleFoam* solver from *OpenFOAM* [16], an extensively validated C++ written framework for the numerical solution of partial differential equation systems. The solver is based on the "Semi-Implicit Method" or *SIMPLE* algorithm [17], which obtains the solution of velocity and pressure fields by iteratively updating initial guesses by correction terms based on the mass and momentum conservation equations. The discretized momentum equation is solved implicitly in two separate steps, with the first providing an explicit solution for the velocity field based on the current pressure field, and the second providing an implicit correction for the pressure field using information from the previous step. This procedure is repeated until convergence is achieved, which can take a considerable amount of time if the required level of accuracy is high. Moreover, if high spatial resolution is desired, more grid elements will be necessary to perform the simulation. Since the discretized equations need to be solved at each grid element, the number of evaluations at each iteration can become very large.

2.2. Machine Learning Approach: DeepCFD

In order to overcome the issues of the CFD approach, mainly regarding computational cost and solution time, DeepCFD is proposed to leverage from deep convolutional neural networks (DCNN) in order to create a surrogate model to provide efficient velocity and pressure solutions for steady-state laminar flows.

2.2.1. Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have proven great capability of learning important features from images at the pixel level in order to make useful predictions for both classification and regression problems [18, 19]. Another advantage of this approach, compared to conventional fully-connected layer networks, lies in the fact that convolutions provide weight-sharing and sparse connectivity [20]. These properties enable more efficient memory usage to learn the necessary information needed to create a surrogate model to reconstruct an approximation of whole velocity and pressure fields from a given set of boundary conditions.

In the case of steady-state flows, the solution will be dependent solely on the boundary conditions, such as the geometry of the problem. Therefore, the input layer of the CNN model needs to provide information regarding the geometric information of the flow. For this task, we use the signed distance function (SDF) proposed by [8], which provides the distance of individual points in the CFD grid to a given surface (such as the cylinder in Figure 1), given by:

$$SDF(x) = \begin{cases} d(x, \partial\Omega) \text{ if } x \in \Omega \\ -d(x, \partial\Omega) \text{ if } x \in \Omega^c \end{cases} \tag{5}$$

where Ω is a subset of a metric space, X, with metric, d, and ∂Ω is the boundary of Ω. For any x ∈ X:

$$d(x, \partial\Omega) := \inf_{y \in \partial\Omega} d(x, y) \tag{6}$$

where inf denotes the infimum. Grid positions inside the obstacle's interior (Ω^c) are assigned negative distances.

Moreover, a multi-class channel with information about the flow region in 5 different categories (0 for the obstacle, 1 for the free flow region, 2 for the upper/bottom no-slip wall condition, 3 for the constant velocity inlet condition, and 4 for the zero-gradient velocity outlet condition) is provided. The employment of consecutive down-sampling convolutional operations encode the given input into a latent geometry representation (LGR), as illustrated in Figure 2-b. Input channels are illustrated in Figure 2-a.

After encoding the geometric information, one can use transposed convolutions (also known as "deconvolutions") to find a mapping between the LGR and any variable of interest, such as the velocity field (Ux and Uy) and the pressure field (p). This can be done by performing upsampling deconvolution operations from the LGR encoding until the original CFD dimension size is achieved, with the number of output channels equal to the number of variables of interest (Figure 2-c). The goal of the training is to minimize the total error function based on a given criterion (e.g. absolute mean error or mean squared error) between the CNN model and the ground-truth CFD results.

2.2.2. Neural Network Architectures

In this work, a modified version of the architecture used by [8] is proposed. In their approach, a down-sampling encoder network compresses the geometric information (SDF) into a reduced dimension latent space or latent geometry representation (LGR), followed by an up-sampling decoder network that maps the LGR encodings back to data space (velocity components). For its similarity to autoencoder networks, this variant is named here as "Autoencoder" or simply "AE", and is considered our baseline. Moreover, each output variable is mapped from the LGR encoding using separate decoder networks. In our proposed approach, since the model must also be able to provide a solution for the pressure field, an U-Net architecture is employed. This kind of network was proposed by [21] for the task of segmentation of medical images, and we show that such networks can also be successfully applied here to find an efficient mapping between geometry and steady-state flow solution of coupled

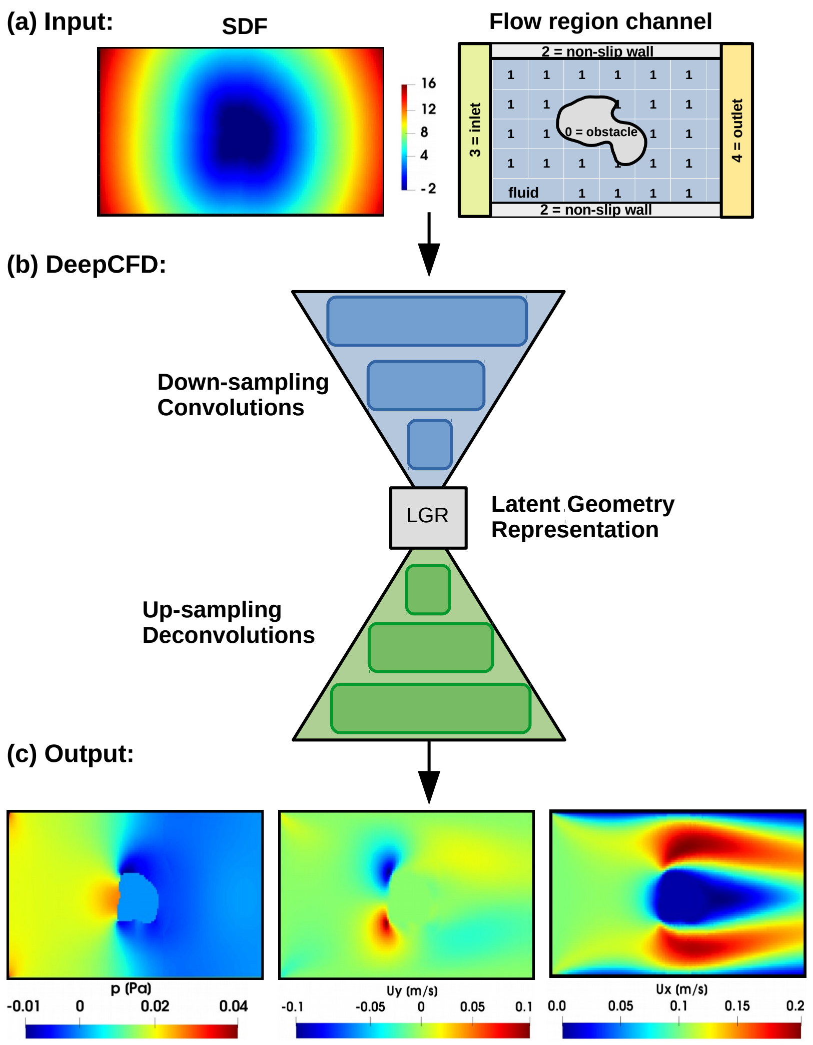

Figure 2: CNN learning approach. (a) Input channels with SDF and multi-class labeling of flow regions. (b) Down-sampling convolutional operations create a latent representation of the flow geometry from the input. (c) Up-sampling deconvolutions map the LGR to variables of interest.

pressure-velocity fields. As shown in the schematic representation provided in Figure 3, both baseline and DeepCFD models use the architecture variants with multiple decoders: AE-3 (c) and UNet-3 (d). For completeness, the effect of disentangling the model outputs in separate decoders was put to test by including architectures with 1 decoder for both Autoencoder and U-Net variants: AE-1 (a) and UNet-1 (b).

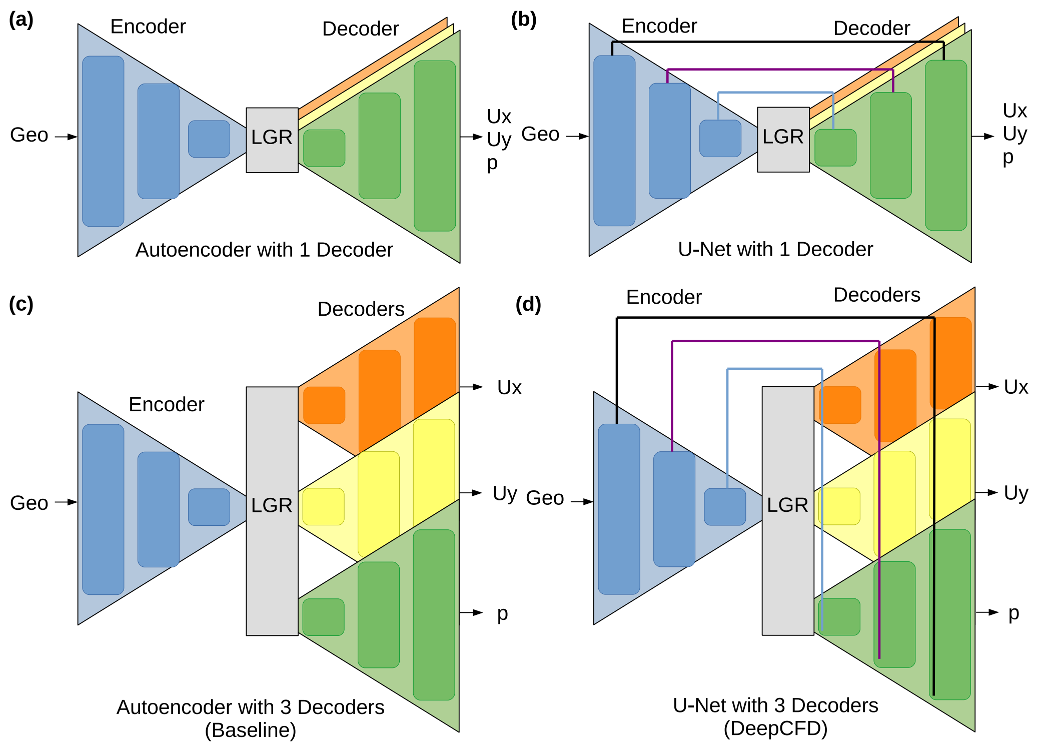

Figure 3: Convolutional neural network architectures and respective variants investigated in this work.

In order to find the ideal setup for the architectures mentioned above, a set of hyper-parameters were considered for investigation. These parameters are summarized in Table 1, and include three different learning rates, three different CNN filter sizes (kernel), and a number of encoder/decoder blocks (3, 4, or 5) with varying number of filters (ranging from 8 to 64). Each encoder/decoder block contains two convolutional layers and the number of filters used in the encoder network is reversed for the decoder network. For example, if a $[16, 32, 64]$ filter configuration is used in the encoder, a $[64, 32, 16]$ configuration is used in the decoder. Max pooling operations are performed for each layer and the ReLU activation function is applied to the layer outputs. Furthermore, batch and weight normalization may or may not be performed. Therefore, each architecture can assume 3x3x3x2x2 = 108 different configurations, which were extensively evaluated during hyper-parameter search.

Table 1: Set of parameters considered for hyper-parameter search.

Parameters			
Learning rate	1e-3	1e-4	1e-5
Kernel	3	5	7
Filters	$16, 32, 64$	$8, 16, 32, 32$	$8, 16, 16, 32, 32$
Norm			
Batch	On	Off	
Weight	On	Off	

3. Problem Setup

In this section, the setup for generating the ground-truth CFD data and the training procedure of DeepCFD are described. A diverse dataset of 2D steady-state channel flow around random shaped obstacles is simulated with the *simpleFoam* solver in more than 1000 instances. For each instance, a new obstacle mesh is used by sampling it from a random shape generator. The generator creates obstacle shapes based on five different primitive shapes (circle, square, forward-facing triangle, backward-backing triangle, and rhombus) by randomly shifting the points used to construct these shapes in a given range of directions. Figure 4 illustrates some examples of the random samples generated that were used as obstacles for the 2D channel flow dataset.

The domain dimensions are 260 mm in the stream-wise direction and 120 mm in the direction perpendicular to the flow. The number of grid elements varies according to the shape used, but with a base cell size of around 1 mm, the average cell count is about 30,000. Boundary conditions are kept fixed, with a constant radial velocity of 0.1 m/s on the inlet (left wall), a zero-gradient condition on the outlet (right wall), and no-slip boundary condition on the top/bottom and obstacle walls, as shown in Figure 5. Furthermore, the laminar dynamic viscosity is set to 1×10^{-4} m^2/s, central differencing schemes (CDS) were used for the discretization of both convective and diffusive

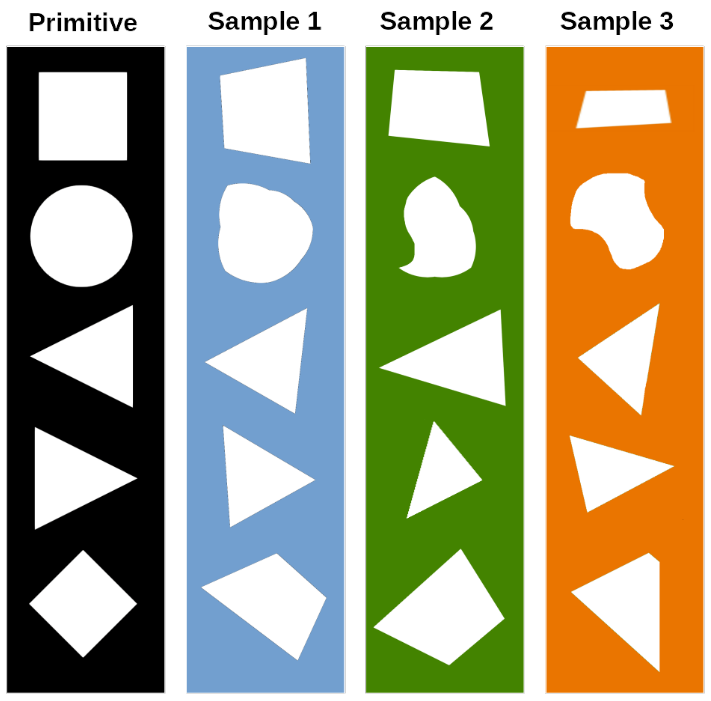

Figure 4: Primitive and derived obstacle forms.

terms of the momentum equation, and simulations were run on a single core of an Intel Xeon E-2146G processor.

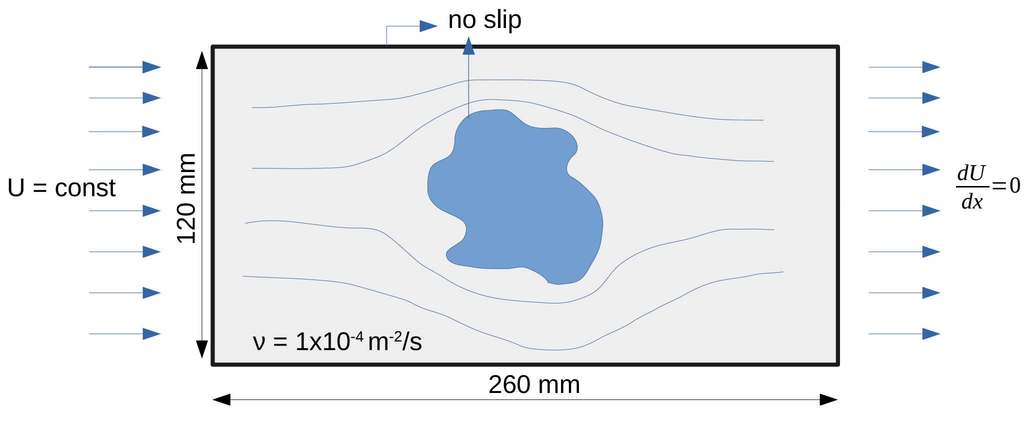

Figure 5: Simulation domain and boundary condition details.

A shell-script was written to automatize the process of dataset generation by calling the random geometry generator, the *OpenFOAM* internal mesher *blockMesh*, and the solver *simpleFoam*. The results for all components of the velocity field and the pressure field were saved for each sample together with the cell location information of the entire computational grid. In the next element of the pipeline, the CFD results of all samples are converted into numpy arrays, so that SDF and multi-class region channels can be calculated and easily inputted to the PyTorch workflow used to train the DeepCFD model.

Regarding the learning process, the AdamW optimizer was employed with a batch size of 64 and weight decay was set to 0.005. The network was trained on an Nvidia Tesla V100 SXM2 GPU using a 70 %-30 % split for training and testing. The loss function combines the errors of all three outputs, velocity components Ux and Uy and the pressure. For the velocity components, a mean squared error function was used, whereas for the pressure a mean absolute error was employed. This choice of loss functions was based on extensive experiments, which showed considerably better convergence when L2-norm was applied to the velocity components and L1-norm used for the pressure. A possible justification for that may be related to characteristics of the pressure-velocity coupling in the momentum equation. In certain situations, the coupling can become unstable and under-relaxation of the pressure becomes necessary to improve the robustness of the pressure solution. Therefore, the use of a loss function more resistant to outliers, such as the L1-norm, may provide a better coupling between velocity and pressure for the reconstruction by the machine learning approach. Care was taken to normalize each individual loss in order not to bias the model towards optimizing a particular output in detriment of the others.

4. Results

In this section, the capability of the proposed model in providing efficient approximations of steady-state laminar flow solutions is demonstrated. First, the model optimization procedure via hyper-parameter search is described, and the test error curves of DeepCFD are plotted against the ones of the baseline model [8]. Furthermore, qualitative and quantitative analyses of the results are provided together with relevant discussion about the model accuracy and performance in comparison with the baseline [8] and with the standard CFD approach.

4.1. Model Optimization

The 108 different parametric configurations introduced in Table 1 of section 2.2.2 were extensively tested for each of the four architectures (Figure 3) considered in this work, yielding a total of 432 experiments. Each experiment was conducted 5 times for 1000 Epochs in order to ensure that results are reproducible, and each model was trained with 700 samples and tested with 300 samples, generated using the setup described in section 3. The average and standard deviation of the mean squared error results on the test set for the best model from each architecture are shown in Table 2:

Table 2: Model performance comparison between best baseline and DeepCFD models. Additional 1 decoder configuration for each case was added to test effect of multiple decoders.

	N = 5 SAMPLES	
MSE	AE-1	BASELINE
UX	2.1513 ± 0.1688	1.7854 ± 0.1175
UY	0.6270 ± 0.0611	0.2956 ± 0.0045
P	1.7198 ± 0.0052	1.2125 ± 0.0150
TOTAL	4.4981 ± 0.1753	3.2935 ± 0.1171
MSE	UNET-1	DEEPCFD
UX	1.1169 ± 0.1393	**0.7730 ± 0.0897**
UY	0.3326 ± 0.0121	**0.2153 ± 0.0186**
P	1.4708 ± 0.0045	**1.0420 ± 0.0431**
TOTAL	2.9203 ± 0.1520	**2.0303 ± 0.1360**

The DeepCFD model (UNet-3 architecture) outperformed the baseline and all other architectures in regard to the mean squared error (MSE) on the test set for all variables of interest (Ux, Uy, and p). The other exact parameters of this selected configuration include a learning rate of 1×10^{-3}, a kernel size of 5, a total of eight convolutional layers for each encoder/decoder network (using the $[8, 16, 32, 32]$ number of filters configuration), and both batch and weight normalization turned off. Our hypothesis that the skip-connections of the U-Net network contributes to more accurate reconstructions than the baseline model by providing direct connections between the encoded geometry features and the decoder layers for each variable was confirmed. Moreover, the proposed DeepCFD architecture with separate decoders also shows significant advantage in comparison to the single decoder model. The total MSE using the DeepCFD network was about 70 % of the one given by UNet-1. One possible explanation for the observed differences is the modeling complexity of the pressure-velocity coupling in the momentum equation, with a non-linear convection term containing a velocity-velocity coupling and a linear pressure-velocity coupling. Therefore, better approximation of the steady-state flow can be obtained with separate decoder networks because each contains its own set of learnable parameters.

4.2. Error curves on the Test-set

Test MSE versus Epoch curves for all variables and the total combined error are shown for 1000 Epochs in Figure 6. Each plot includes the results from the baseline model (dashed orange line) and from the DeepCFD model (continuous blue line). In addition to the central tendency (mean curves), the observed standard deviations are also plotted as shaded regions around the mean curves.

Although the baseline model shows slightly better initial performance in predicting both components of the velocity field (Ux and Uy), it takes about 100 Epochs for DeepCFD to achieve the same performance while it steadily continues to improve. After 100 Epochs, overall test error ratio between baseline and DeepCFD is already 1.2, with this figure increasing to 1.6 times at 1000 Epochs. Regarding the accuracy of the pressure prediction, the proposed model outperforms the baseline from the beginning, achieving an MSE about 17 % smaller than the baseline total MSE and remaining around 1 during almost the entire training time.

4.3. Flow Visual Inspection

Qualitative plots of flow reconstructions provided by DeepCFD on test-set samples are presented and compared to the ground-truth solution. For instance, results of both velocity and pressure fields from the flow around an obstacle based on the square primitive form are shown in Figure 7. The left column refers to the ground-truth CFD data (simpleFOAM), whereas the right column presents the DeepCFD results. It is good practice to show all components of vector fields, such as the velocity, in order to provide information not only about the magnitude but also about the direction of the flow. Therefore, the first two rows show the horizontal (x) and vertical

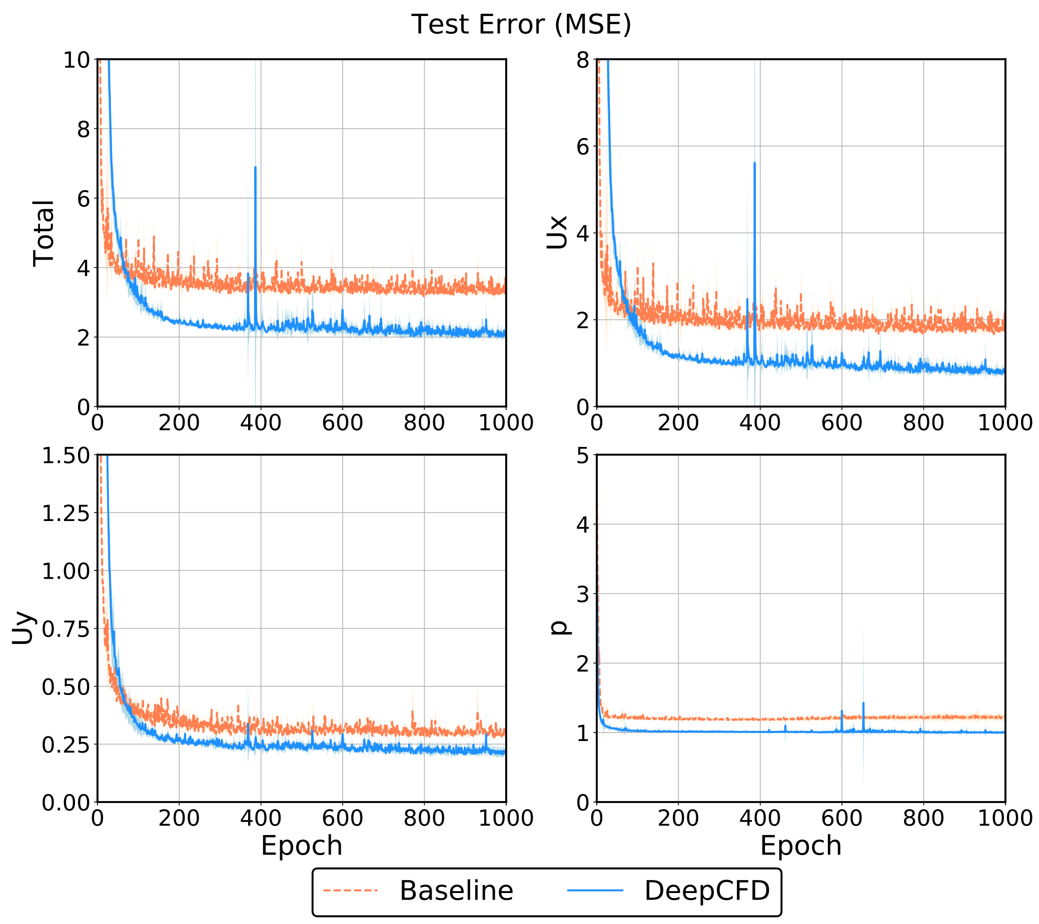

Figure 6: Test MSE vs Epoch curves for the total error, horizontal and vertical velocity components, and pressure.

(y) components of the velocity field (U), while the third row provides the pressure field (p) results.

The square shaped obstacle induces a high pressure region on its frontal edge, as well as forces the flow to separate right after the fluid reaches the frontal vertices of the obstacle. This behavior is also well captured by the DeepCFD model in both velocity and pressure plots.

Next plots, in Figures 8-9, include the absolute error information between the ground-truth and the machine learned results. Due to space limitation, the velocity magnitude is shown instead of the separate components, and plots

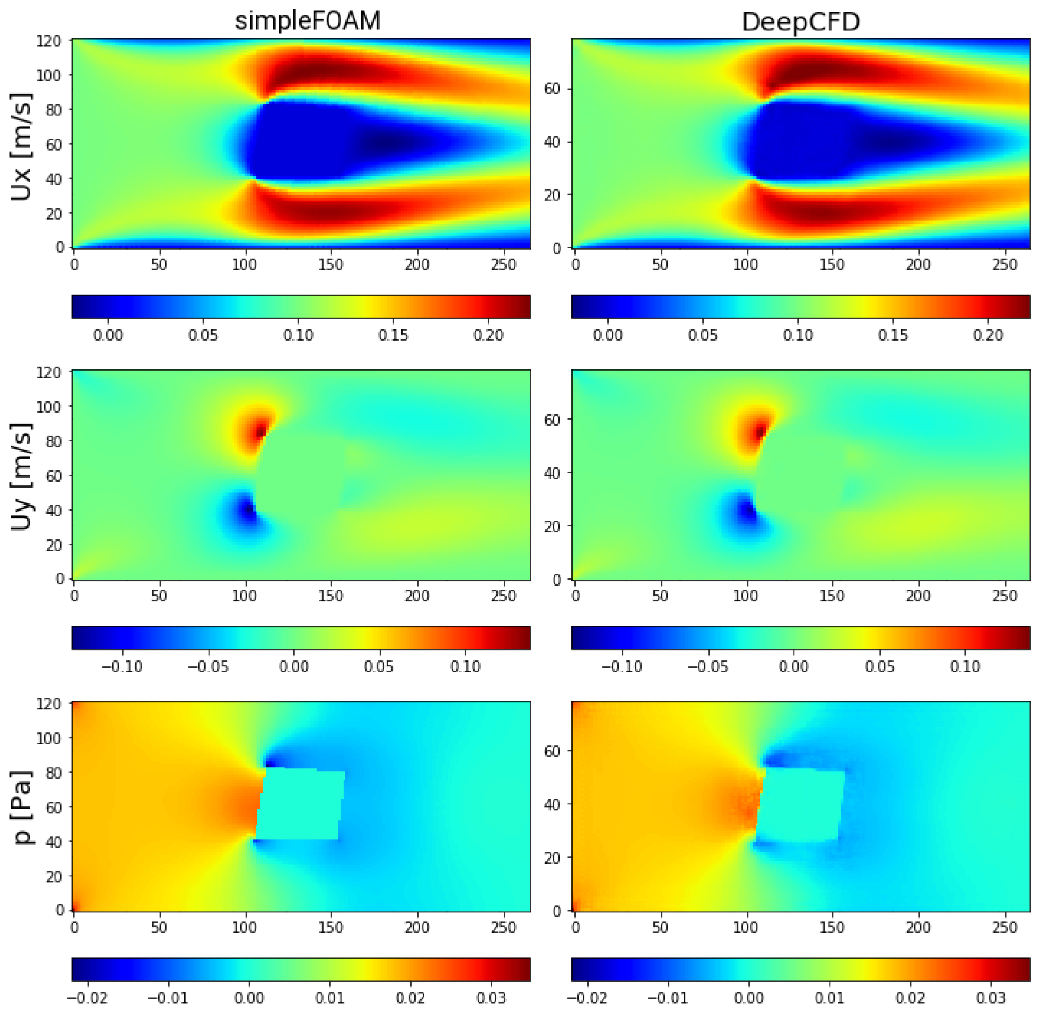

Figure 7: Comparison between ground-truth CFD (simpleFOAM) and DeepCFD prediction, showing both velocity components and pressure fields in flow around square based shape.

positions have been rearranged. The first two rows show, respectively, ground-truth CFD and DeepCFD data, whereas the last row shows the absolute error for both velocity magnitude (first column) and pressure (second column). For complete plots of this and other flows with separate velocity components, the reader can refer to the supplementary material provided with the paper.

Figure 8 shows the flow around a circle based shape, which creates a round

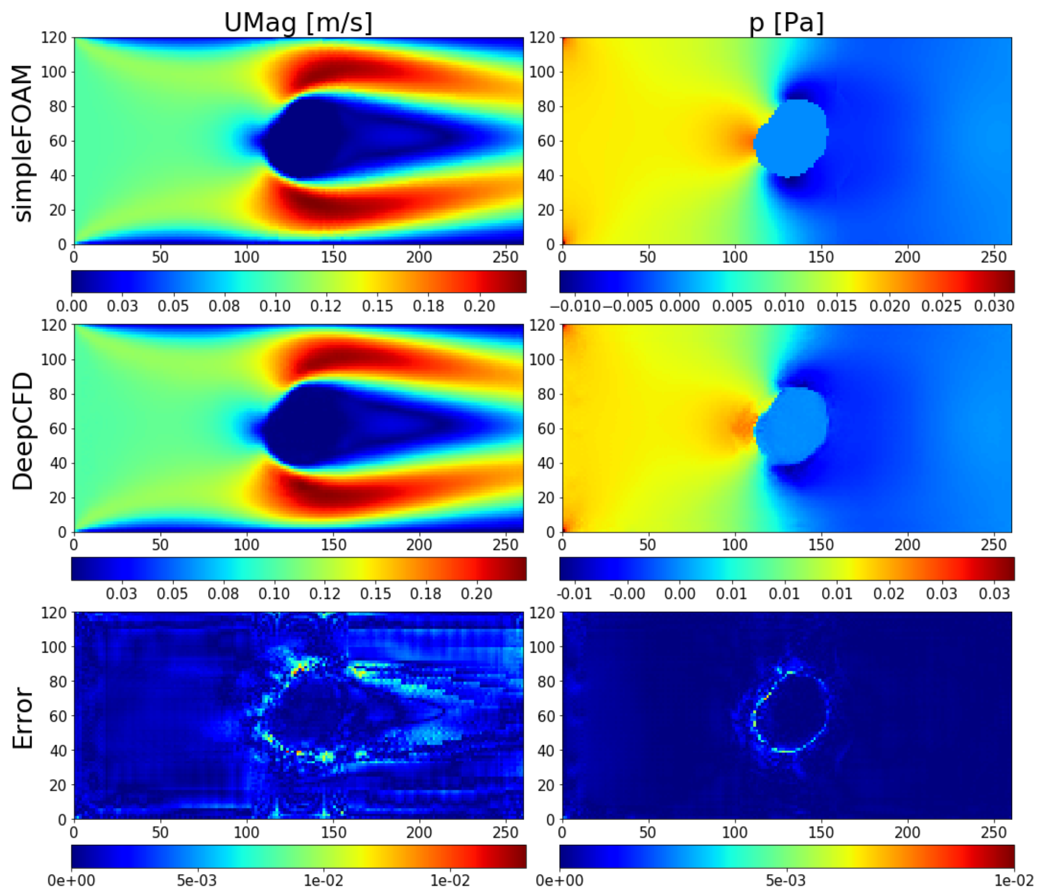

Figure 8: Comparison between ground-truth CFD (simpleFOAM) and DeepCFD prediction, showing velocity magnitude, pressure, and absolute error in flow around circle based shape.

region of high pressure on its leading edge, and flow separation happens as the fluid approaches the middle of the obstacle from both top and bottom surfaces. In Figure 9, the forward-facing triangle shape forms a considerably smaller region of high pressure at its frontal vertex, with flow separation occurring only further downhill. The DeepCFD model is able to correctly capture all these phenomena at a cost of very low error rates.

4.4. Quantitative Analysis

In Figure 10, the ground-truth CFD data distribution is plotted against the DeepCFD modeled data distribution from 295 test samples. For consistency

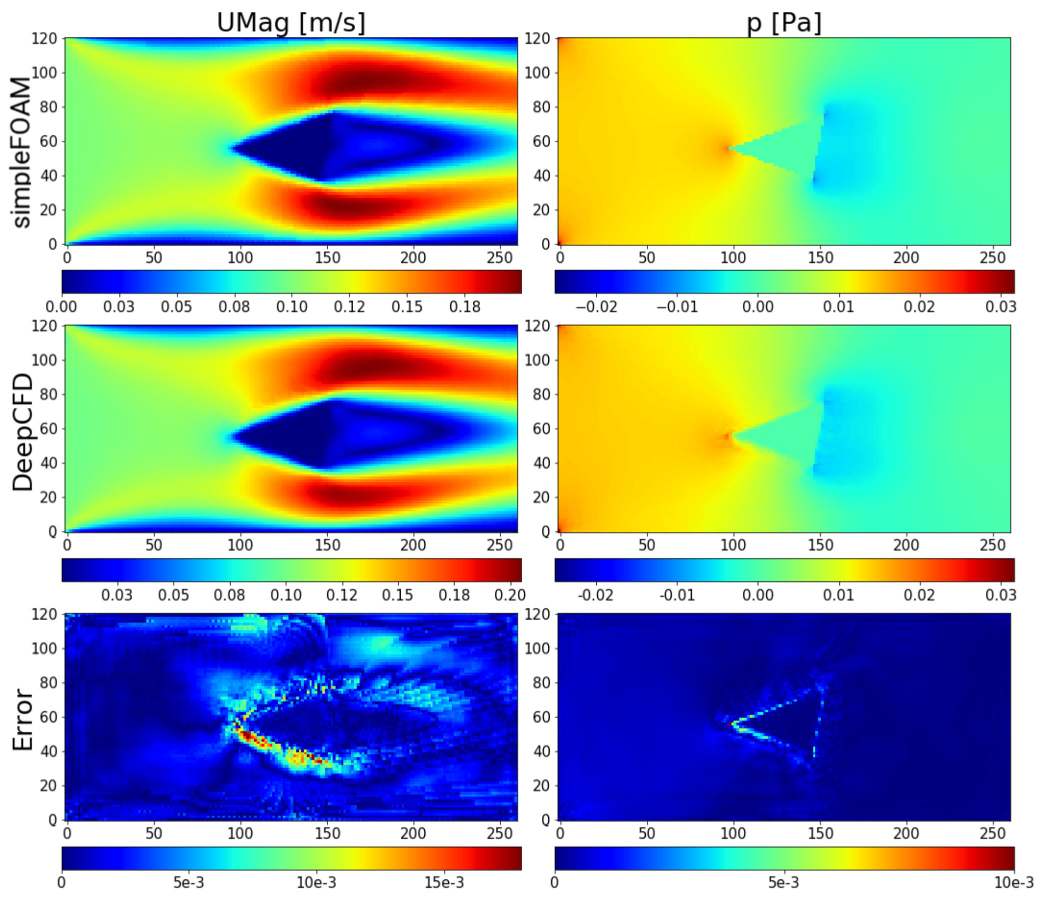

Figure 9: Comparison between ground-truth CFD (simpleFOAM) and DeepCFD prediction, showing velocity magnitude, pressure, and absolute error in flow around forward-facing triangle shape.

with previously shown figures, the first plot on the top left presents the combined data with all variables (Ux, Uy, and p), whereas the other plots show the data distribution for each specific variable.

In agreement with the qualitative plots formerly presented, the approximated DeepCFD solution on the test-set produces data distributions with shapes very similar to the ones from the ground-truth CFD simulation for all quantities analysed. Furthermore, mean and standard deviation differences between the two approaches are very small. In order to evaluate how these deviations from DeepCFD compare to those obtained with the baseline,

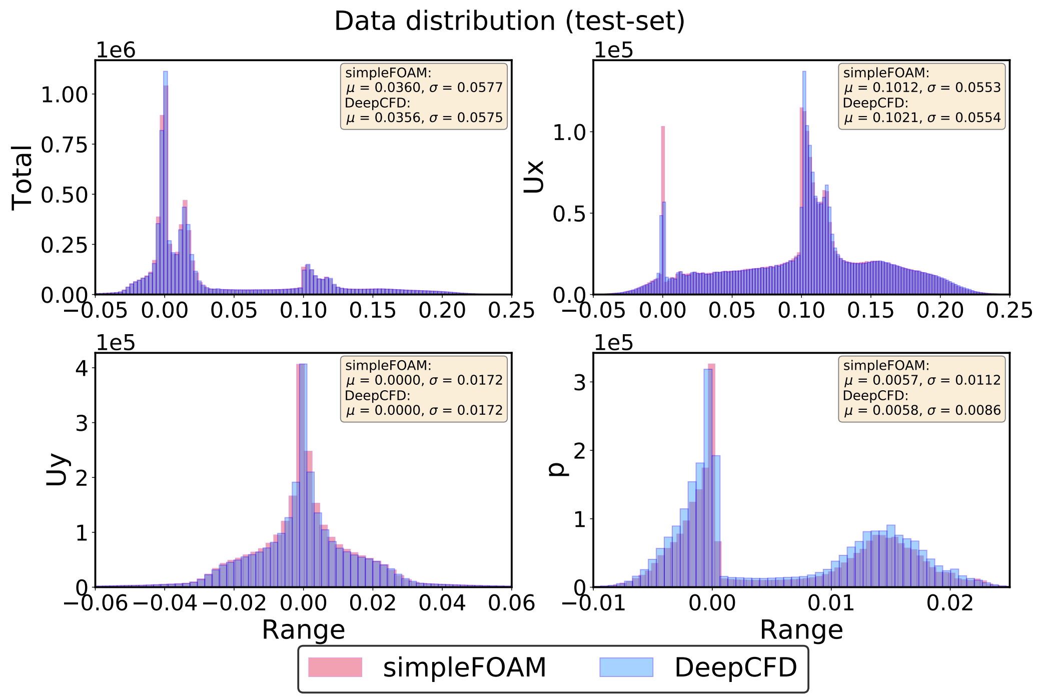

Figure 10: Ground-truth CFD data distribution against DeepCFD predicted distribution from 295 test samples.

Figure 11 shows the relative error distributions for each of these models.

The relative error is given by $\left|\frac{p-gt}{gt+k}\right| \times 100\%$, where p is the prediction, and gt is the ground-truth value. Because this metric fails when gt approaches zero (division by zero), an adjusting scalar $k = 1 \times 10^{-4}$ is plugged in the denominator to address the issue. As shown, the predictions made by the proposed model tend to be concentrated on the lower end of relative errors (most values with less than 10% error), whereas the baseline shows a wider error distribution with considerably more elements with higher error rates.

Finally, the performance of DeepCFD, in terms of prediction time, is tested and compared against the standard CFD solution in Table 3. Since the steady-state flow solver used here is not implemented for GPU runs, the reference time of the standard CFD approach was taken from the average of 50 random runs on one single core of the Intel Xeon E-2146G processor. In

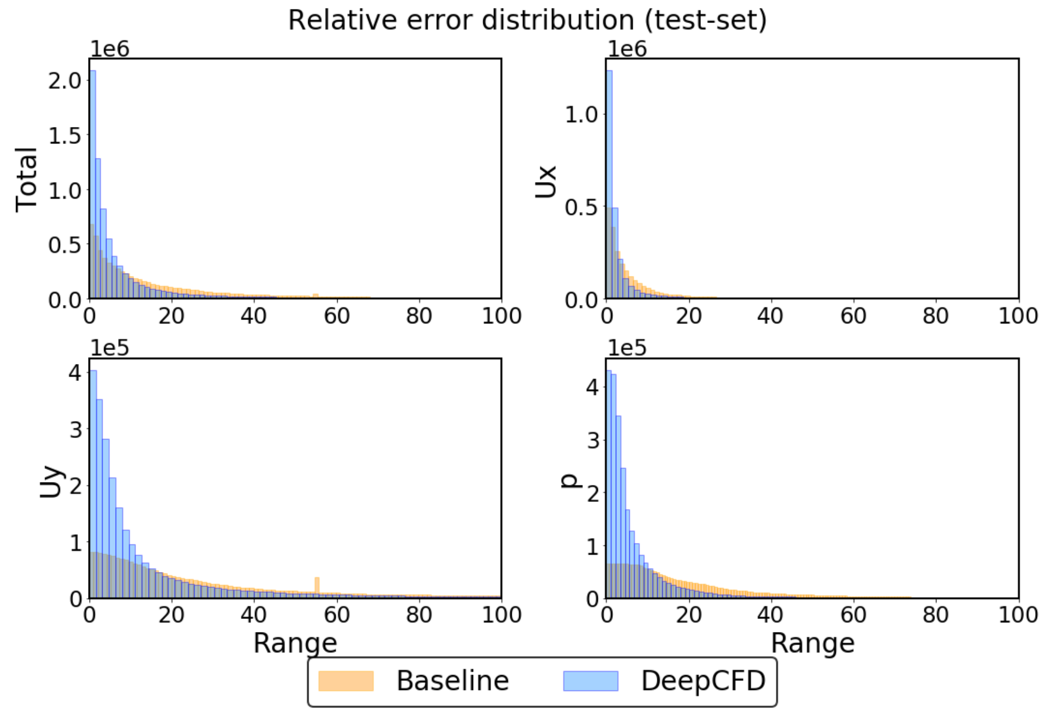

Figure 11: Relative error distribution for predictions using baseline (orange) and DeepCFD (blue) models on 295 test samples.

the case of the machine learned approach, approximations can be generated using both CPUs and GPUs. Therefore, the time taken for DeepCFD to make a prediction was evaluated on the same CPU used to generate the ground-truth CFD data, as well as on an Nvidia Geforce RTX-2080 Ti GPU. Moreover, the time results were averaged from 1000 different runs using three different batch sizes: 1, 10, and 100. Average and standard deviation of run times, as well as speedup values are provided.

For a fair evaluation, only the CPU-CPU time comparison can be considered a speedup advantage, since GPU reference time measurements are not available. Moreover, due to the characteristics of the pressure-velocity coupling, CFD solution times may vary considerably, thus the average solution time was considered. Even in this case, up to three orders of magnitude speedup was obtained for all batch sizes used. Increasing the batch-size improved the performance only marginally for the CPU runs. However, because

Table 3: Run time and speedup comparisons.

	CFD (CPU)	
Batch size	Time (s)	Speedup
1	52.51 ± 15.27	-
	DeepCFD (CPU)	
Batch size	Time (s)	Speedup
1	$4.77 \times 10^{-2} \pm 7.15 \times 10^{-4}$	1.10×10^{3}
10	$3.57 \times 10^{-2} \pm 5.44 \times 10^{-4}$	1.47×10^{3}
100	$3.50 \times 10^{-2} \pm 7.07 \times 10^{-4}$	1.50×10^{3}
	DeepCFD (GPU)	
Batch size	Time (s)	Speedup
1	$4.57 \times 10^{-3} \pm 1.03 \times 10^{-4}$	1.15×10^{4}
10	$6.81 \times 10^{-4} \pm 1.03 \times 10^{-4}$	7.71×10^{4}
100	$1.02 \times 10^{-4} \pm 1.03 \times 10^{-4}$	5.14×10^{5}

machine learning models can be easily run on GPUs, the GPU evaluation times were also considered. In that scenario, prediction times can vary considerably depending on the batch size, starting with 4 orders of magnitude speedup with batch size 1 and up to 5 orders of magnitude with 100 samples.

5. Conclusions and Future Work

In this paper, we proposed a new and efficient way for approximating non-uniform steady laminar flow CFD calculations. Previous works that addressed this problem could provide only a solution for the velocity field, but we showed that an U-Net architecture can be employed to provide complete solutions of coupled velocity and pressure fields. The ground-truth CFD data was created with an extensively validated numerical solver and a workflow for generating training samples for a channel flow around randomly shaped objects was developed. After comprehensive hyper-parameter search, considering 4 different architectures in 108 parameter configurations, we found that the proposed DeepCFD model (U-Net architecture using separate decoders) outperformed all other models, including the baseline (Autoencoder with 3 decoders). The proposed DeepCFD model with optimum parameters was then used in additional experiments in order to perform further analyses in both qualitative and quantitative terms.

Relevant discussion about the model's accuracy and performance in comparison to the baseline and the standard CFD approach was provided. At a cost of low error rates, a speedup of up to 3 orders of magnitude can be achieved (CPU-CPU), or even of up to 5 orders of magnitude (GPU-CPU). Finally, we provide code and dataset as supplementary material in order to contribute with further expansion of the field of data-driven models for CFD.

For future work, we intend to extend the 2D methodology used here for 3D flow configurations, as well as expand the dataset in terms of number of samples and variability for more complex generalization in real flow conditions. Furthermore, since most flows of interest for engineering are turbulent, we intend to incorporate the model developed here to recurrent neural networks architectures in order to take the time dependency into account. Most efforts in this field concentrate on developing closure terms for governing equations from global turbulent quantities, but the ability of CNNs in learning spatial characteristics of the flow can also be leveraged in such future investigations. Finally, we also intend to incorporate physical constraints to the neural network training procedure, so that the network prediction is bounded within the physical constraints of the problem.

References

[1] R. Swanson, S. Langer, Steady-state laminar flow solutions for naca 0012 airfoil, Computers & Fluids 126 (2016) 102 – 128. doi:`https://doi.org/10.1016/j.compfluid.2015.11.009`.

[2] K.-J. Bathe, H. Zhang, A mesh adaptivity procedure for cfd and fluid-structure interactions, Computers & Structures 87 (2009) 604 – 617. doi:`https://doi.org/10.1016/j.compstruc.2009.01.017`, fifth MIT Conference on Computational Fluid and Solid Mechanics.

[3] S. B. Pope, Turbulent Flows, Cambridge University Press, 2000. doi:`10.1017/CBO9780511840531`.

[4] K. Foli, T. Okabe, M. Olhofer, Y. Jin, B. Sendhoff, Optimization of micro heat exchanger: Cfd, analytical approach and multi-objective evolutionary algorithms, International Journal of Heat and Mass Transfer 49 (2006) 1090 – 1099.

[5] C. S. Fernandes, R. P. Dias, J. M. Nóbrega, J. M. Maia, Laminar flow in chevron-type plate heat exchangers: Cfd analysis of tortuosity, shape factor and friction factor, Chemical Engineering and Processing: Process Intensification 46 (2007) 825 – 833. doi:`https://doi.org/10.1016/j.cep.2007.05.011`, selected Papers from the European Process Intensification Conference (EPIC), Copenhagen, Denmark, September 19-20, 2007.

[6] P. Talukdar, C. R. Iskra, C. J. Simonson, Combined heat and mass transfer for laminar flow of moist air in a 3d rectangular duct: Cfd simulation and validation with experimental data, International Journal of Heat and Mass Transfer 51 (2008) 3091 – 3102.

[7] G. D. Portwood, P. P. Mitra, M. D. Ribeiro, T. M. Nguyen, B. T. Nadiga, J. A. Saenz, M. Chertkov, A. Garg, A. Anandkumar, A. Dengel, et al., Turbulence forecasting via neural ode, arXiv preprint arXiv:1911.05180 (2019).

[8] X. Guo, W. Li, F. Iorio, Convolutional neural networks for steady flow approximation, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 16, Association for Computing Machinery, New York, NY, USA, 2016, p. 481490.

[9] T. Q. Chen, Y. Rubanova, J. Bettencourt, D. K. Duvenaud, Neural ordinary differential equations, in: Advances in neural information processing systems, 2018, pp. 6571–6583.

[10] M. Raissi, P. Perdikaris, G. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, Journal of Computational Physics 378 (2019) 686 – 707. doi:`https://doi.org/10.1016/j.jcp.2018.10.045`.

[11] F. Sarghini, G. de Felice, S. Santini, Neural networks based subgrid scale modeling in large eddy simulations, Computers & Fluids 32 (2003) 97 – 108. doi:`https://doi.org/10.1016/S0045-7930(01)00098-6`.

[12] J. Ling, A. Kurzawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, Journal of Fluid Mechanics 807 (2016) 155166. doi:`10.1017/jfm.2016.615`.

[13] H. F. S. Lui, W. R. Wolf, Construction of reduced-order models for fluid flows using deep feedforward neural networks, Journal of Fluid Mechanics 872 (2019) 963994. doi:10.1017/jfm.2019.358.

[14] J. Tompson, K. Schlachter, P. Sprechmann, K. Perlin, Accelerating eulerian fluid simulation with convolutional networks, 2016. arXiv:1607.03597.

[15] M. Dias Ribeiro, G. D. Portwood, P. Mitra, T. Mihn Nyugen, B. T. Nadiga, M. Chertkov, A. Anandkumar, D. P. Schmidt, A data-driven approach to modeling turbulent decay at non-asymptotic reynolds numbers, Bulletin of the American Physical Society (2019).

[16] H. G. Weller, G. Tabor, H. Jasak, C. Fureby, A tensorial approach to computational continuum mechanics using object-oriented techniques, Computers in Physics 12 (1998) 620–631. doi:10.1063/1.168744.

[17] S. V. Patankar, Numerical heat transfer and fluid flow, Series on Computational Methods in Mechanics and Thermal Science, Hemisphere Publishing Corporation (CRC Press, Taylor & Francis Group), 1980.

[18] M. Z. Afzal, S. Capobianco, M. I. Malik, S. Marinai, T. M. Breuel, A. Dengel, M. Liwicki, Deepdocclassifier: Document classification with deep convolutional neural network., in: ICDAR, IEEE Computer Society, 2015, pp. 1111–1115. Relocated from Tunis, Tunisia.

[19] V. Sekar, M. Zhang, C. Shu, B. C. Khoo, Inverse design of airfoil using a deep convolutional neural network, AIAA Journal 57 (2019) 993–1003. doi:10.2514/1.J057894.

[20] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016. http://www.deeplearningbook.org.

[21] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical image computing and computer-assisted intervention, Springer, 2015, pp. 234–241.