CS230

Prediction of Velocity Magnitude for Flows around Bluff Bodies on Irregular Geometries

Ali Kashefi
Department of Civil and Environmental Engineering
Stanford University
kashefi@stanford.edu

1 Abstract

A deep learning configuration for prediction of the velocity magnitude field over bluff bodies is presented. Our neural network is based on a modified segmentation track of PointNet to take its advantages over traditional convolutional neural networks (CNNs). Laminar steady-state flows over seven different cross sections are considered as test cases for generating data set. The procedure of choosing hyper-parameters is discussed. Visual comparison between the ground truth and predicted fields is provided. Computed mean square error demonstrates a reliable level of accuracy of the prediction. Our network obtains a considerable rate of speedup for predicting the velocity field in comparison with our regular numerical solver.

2 Introduction

One of the most important contributions of machine learning tools to Computational Fluid Dynamics (CFD) simulations is reducing the computational expenses. Even with high performance computing techniques [Moureau et al., 2011, Nagel et al., 2019] and efficient numerical algorithms [Filelis-Papadopoulos et al., 2014, Kashefi and Staples, 2018]) for accelerating CFD simulations, investigation of geometrical parameters for reaching an optimized design is yet computationally expensive, specifically due to requiring a huge number of iterations for analyzing flow fields. To overcome these issues, a few or all the components of a CFD solver have been replaced by a neural network [Sekar et al., 2019, Guo et al., 2016, Tompson et al., 2017, Thuerey et al., 2019, Bhatnagar et al., 2019].

3 Related Work

To employ artificial neural networks as a replacement of CFD solvers, it is critical to appropriately feed CFD data into a network. Hence, an effective data representation is crucial. The connection of neural networks with Cartesian grids is straightforward. In this case, using two and three-dimensional convolutional neural networks (CNNs) is a regular approach [Fukami et al., 2019, Lapeyre et al., 2019, Kim and Lee, 2020]. Based on this scenario, each vertex of a Cartesian grid corresponds to a pixel of an image processed by a CNN. Nevertheless, using unstructured grids is unavoidable for real-world applications. In contrast with Cartesian grids, the connection of unstructured grids, and consequently scatter CFD data, with neural networks becomes challenging. The approach to connect scatter CFD data to a two or three-dimensional CNN is to use pixelation. Using pixelation, scatter CFD data is projected into a two or three-dimensional Cartesian grid such that they become readable by regular CNNs [Sekar et al., 2019, Guo et al., 2016, Tompson et al., 2017, Thuerey et al., 2019, Bhatnagar et al., 2019, Jin et al., 2018, Zhang et al., 2018, Han et al., 2019, Hui et al., 2020, Hasegawa et al.,

CS230: Deep Learning, Fall 2020, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

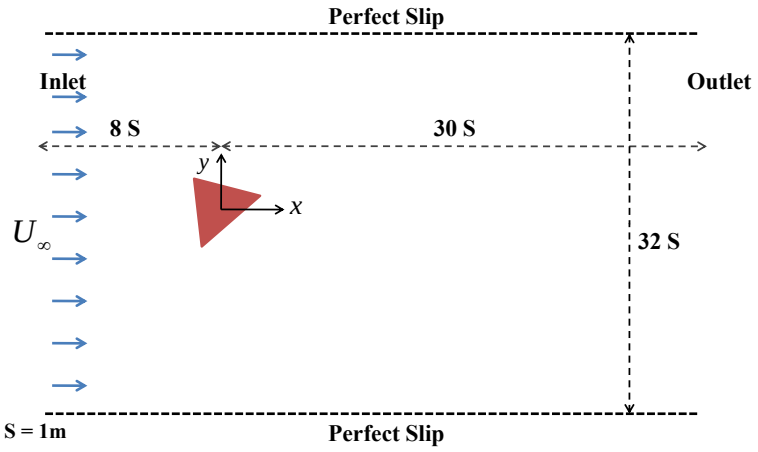

Figure 1: Schematic representation of the physical domain for a cylinder with a triangular cross section

2020]. However, this approach has several shortcoming such as introducing error to the dataset due data interpolation, introducing artificial roughness to previously-smoothed curves in CFD domains, wasting computational capacities of CNNs due to masking interior pixels corresponded to an object inside CFD domains, and etc. To overcome these issues, we present scatter CFD data as a point cloud and consequently use a point-net-based neural network such as PointNet [Qi et al., 2017] for prediction of velocity magnitudes.

4 Dataset and Features

To generated our dataset, first we need to solve the governing equations of fluid motions. Incompressible viscous steady flow is governed by the continuity and momentum equations as follow:

$$\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y} = 0, \tag{1}$$

$$\rho\left(u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y}\right) = -\frac{\partial p}{\partial x} + \mu\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right), \tag{2}$$

$$\rho\left(u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y}\right) = -\frac{\partial p}{\partial y} + \mu\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right), \tag{3}$$

where u and v indicate x and y components of the velocity vector, respectively. p stands for the absolut pressure of the fluid. ρ is the fluid density and μ shows the dynamics viscosity of fluid. Specifically, we are interested in the velocity magnitude (U), which is computed by

$$U = \sqrt{u^2 + v^2}. \tag{4}$$

One may refer to Kashefi and Staples [2018] for further details of the equations such as enforced boundary conditions. The solution of Eqs. 1–3 is function of geometry of the object inside the domain, while they do not have any analytical solutions. The Gmsh [Geuzaine and Remacle, 2009] application is employed for the discretization of the fluid domain by finite volume meshes. OpenFOAM [Weller

et al., 1998] is utilized to obtain the numerical solution of Eqs. 1–3, which is considered as the ground truth. The velocity magnitude is made dimensionless as follows:

$$U^* = \frac{U}{U_\infty}, \tag{5}$$

where U_∞ is the uniform input velocity to the domain (see Fig. 1). In the next stage, we scale U^* in a range of $[0, 1]$ by the following formulation:

$$U^{'} = \frac{U^* - \min(U^*)}{\max(U^*) - \min(U^*)}. \tag{6}$$

We consider seven various geometries: circle [Behr et al., 1995], square [Sen et al., 2011], triangle [Kumar De and Dalal, 2006], rectangle [Zhong et al., 2019], ellipse [Mittal and Balachandar, 1996], pentagon [Abedin et al., 2017], and hexagon [Abedin et al., 2017]. A summary of the geometries is provided in Table 1. A total number of 1875 data is generated. We split the generated data into three categories of training (80%), validation (10%), and test (10%) sets randomly.

Shape	Variation in orientation	Variation in length scale	Number of data
Circle	-	$a = 1$ m	1
Equilateral hexagon	$3°, 6°, \ldots, 60°$	$a = 1$ m	20
Equilateral pentagon	$3°, 6°, \ldots, 72°$	$a = 1$ m	24
Square	$3°, 6°, \ldots, 90°$	$a = 1$ m	30
Equilateral triangle	$3°, 6°, \ldots, 180°$	$a = 1$ m	60
Rectangle	$3°, 6°, \ldots, 180°$	$a = 1$ m; $b/a = 1.2, 1.4, \ldots, 3.6$	780
Ellipse	$3°, 6°, \ldots, 180°$	$a = 1$ m; $b/a = 1.2, 1.4, \ldots, 4.2$	960

Table 1: Generated geometries; For circle, equilateral hexagon, equilateral pentagon, and equilateral triangle, a is defined as the main diameter. For square and rectangle, a is defined as the small side, while b is defined as the large side (if any). For ellipse, a and b are defined as the small and large diameters respectively.

5 Methods

We take the segmentation component of the PointNet architecture [Qi et al., 2017] for our purpose (see Fig. 2). One may refer to Qi et al. [2017] for details of the PointNet architecture. We adjust the network according to our desired application. This adjustments involves two main steps. First, the "sigmoid" activation function is used in the last layer. Second, mean square error (MSE) is used as the loss function. MSE is a suitable loss function ($\mathcal{L}$) for deep learning of computational fluid dynamics and frequently is used in the literature (see e.g., Refs. Sekar et al. [2019], Bhatnagar et al. [2019]). Thus, this function is used in this project and determined as

$$\mathcal{L} = \frac{1}{N}\left(\sum_{i=1}^{N}\left[(U_i^{'} - \tilde{U}_i^{'})^2\right]\right), \tag{7}$$

where $\tilde{U}^{'}$ is the velocity magnitude predicted by our neural network. N indicated the number of points inside the cloud. In this study, we take $N = 1024$. We use the Adam optimizer [Kingma and Ba, 2014]. The learning rate of $\alpha = 5 \times 10^{-4}$ and the hyper-parameters of $\beta_1 = 0.9$, $\beta_2 = 0.999$ are chosen. The batch size of 256 is selected. We stop training after 4000 epochs to avoid over-fitting. The training process takes approximately 10 hours on our available GPU. To set these hyper-parameters, a systematic process has been undertaken. To save space, we present our analysis for the learning rate and the batch size. Table 2 and Table 3 demonstrate a summary of our analysis respectively for the batch size and the learning rate. Note that another important hyper-parameter is the size of the global feature (see Fig. 2), since a relatively higher size leads to a more accurate encoding of the geometrical features of the CFD domain. Based on our numerical experiment, the size of 2048 led to 4.3% reduction in the average error of the test set in comparison with the size of 1024. Notwithstanding this success, the choice of batch size of 256 was impossible due to the memory limitation. On the other hand, the batch size of 128 increased the training time to 16 hours; and that is why we stand with 1024.

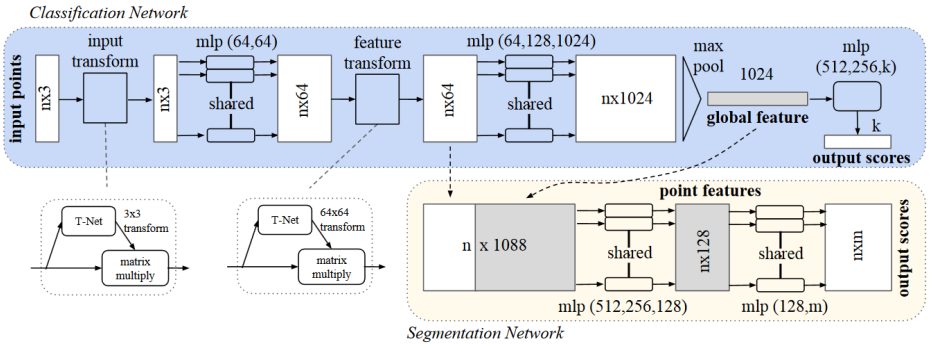

Figure 2: Schematic representation of PointNet; the segmentation branch of this network is used in the current study. This figure is completely taken from Qi et al. [2017].

Batch size	64	128	256
Average loss ($\mathcal{L}$) of the training set	4.86E−5	5.38E−5	8.47E−5
Average loss ($\mathcal{L}$) of the validation set	9.75E−4	8.92E−4	6.83E−4
Average loss ($\mathcal{L}$) of the test set	4.01E−3	2.81E−3	2.17E−3

Table 2: Effect of batch size on the accuracy of training, validation, and test sets

6 Experiments/Results/Discussion

A visual comparison between the ground truth and our network prediction for a few different cross sections selected from the test set is made in Fig. 3. Quantitatively, the average, minimum, and maximum L^2 norm error of the test set are tabulated in Table 4. Accordingly, an excellent to reasonable level of accuracy of prediction is obtained. Next, we report the speedup factor obtained by our neural network to assess the rate of computation accelerating with reference to our traditional computational fluid dynamics solver. The wall time consumed on our available CPU for the simulation of the velocity magnitude for 256 unseen shapes by the CFD solver takes approximately 10105 seconds (nearly 3 hours), while our network predicts the same field for these 256 shapes in 6 seconds on our available GPU. Thus, the a speedup factor of 1683 is achieved. Note that this number is not absolute and strongly depends on the efficiency of our computational resources.

7 Conclusion/Future Work

We proposed a deep learning strategy for the prediction of velocity magnitude in complex geometries. Our neural network was fundamentally based on the segmentation component of PointNet Qi et al. [2017]. By means of this deep learning configuration, potential users are able to preserve the accuracy of CFD data for training. Moreover, they would be able to study the effect of small variations in object geometries, something that is not achievable using pixelation strategies and regular CNNs, unless a super-high resolution input is used, which is computationally expensive by itself. Similarly, boundary smoothness is not destroyed using our approach in contrast with CNN-based algorithms. For our future projects, we investigate the prediction of the velocity vector (u and v) and pressure (p) fields (rather than just simply U). Moreover, we are interested in unsteady flows where objects inside the domain or domain boundaries evolve in time, and thus a dynamic point-cloud deep learning is essential.

8 Contributions

This team has a one member and all the tasks have been carried out by the single author. The basic code of the PointNet architecture has been taken from HERE. This code has been written by the Keras library [Chollet, 2015]. However, the author modified the code for the purpose of solving the regression problem.

Learning rate	$1\times10\text{E}{-4}$	$2\times10\text{E}{-4}$	$5\times10\text{E}{-4}$
Average loss ($\mathcal{L}$)	2.88826E−3	3.38516E−3	2.17772E−3
Maximum loss ($\mathcal{L}$)	6.42315E−2	8.07633E−2	6.88919E−2
Minimum loss ($\mathcal{L}$)	1.66632E−4	5.42080E−4	2.06523E−4

Table 3: Effect of learning rate on the accuracy of prediction of test set; for all the learning rate, the fixed batch size of 256 is chosen. For each learning rate, iterations are continued until a convergence for the validation set is observed.

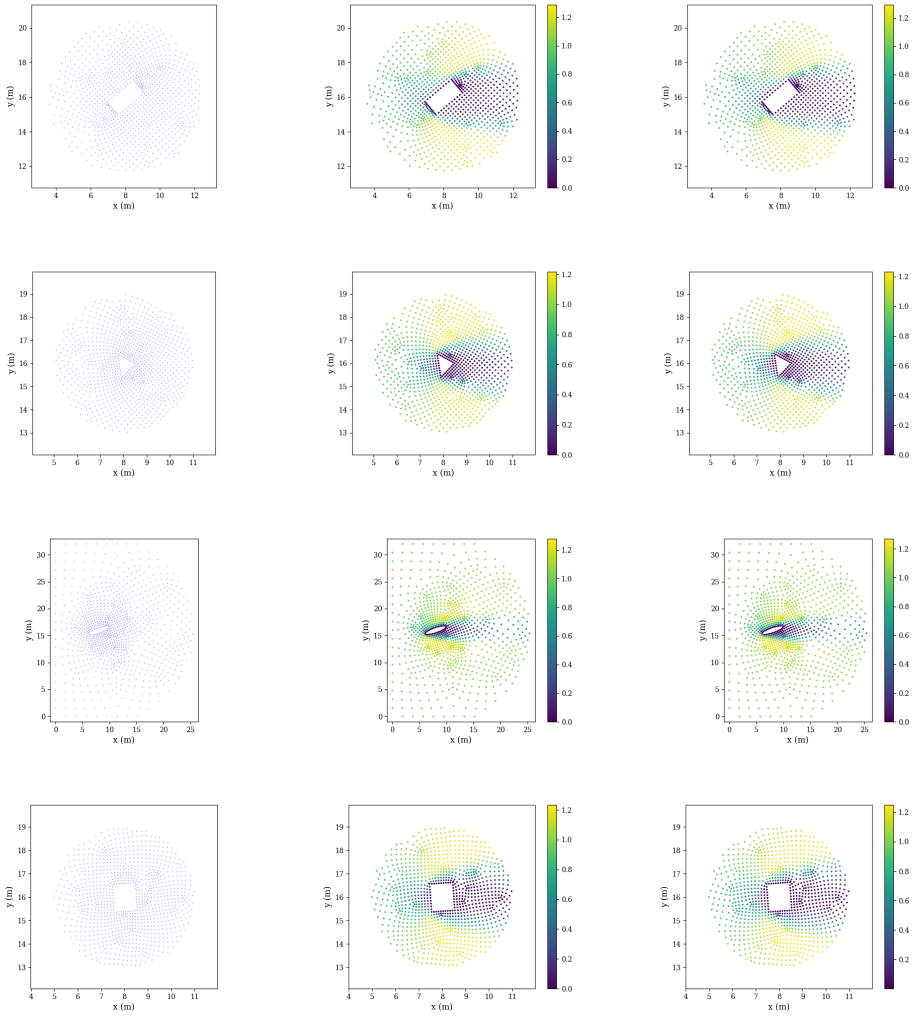

Figure 3: Each row shows a set of input as a point cloud (left), ground truth of the velocity magnitude (center) and the predicted velocity magnitude (right) for a cross section

	L^2 norm
Average	4.66658E−3
Maximum	2.62472E−1
Minimum	1.43709E−2

Table 4: Error analysis of prediction of the velocity magnitude (U) for 256 unseen data (test set)

References

Z. Abedin, N. A. Khan, M. M. Rizia, and M. Q. Islam. Simulation of wind flow over square, pentagonal and hexagonal cylinders in a staggered form. In *AIP Conference Proceedings*, volume

1919, page 020004. AIP Publishing LLC, 2017.

M. Behr, D. Hastreiter, S. Mittal, and T. Tezduyar. Incompressible flow past a circular cylinder: dependence of the computed flow field on the location of the lateral boundaries. *Computer Methods in Applied Mechanics and Engineering*, 123(1-4):309–316, 1995.

S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, and S. Kaushik. Prediction of aerodynamic flow fields using convolutional neural networks. *Computational Mechanics*, 64(2):525–545, 2019.

F. Chollet. Keras. `https://github.com/fchollet/keras`, 2015.

C. K. Filelis-Papadopoulos, G. A. Gravvanis, and E. A. Lipitakis. On the numerical modeling of convection-diffusion problems by finite element multigrid preconditioning methods. *Advances in Engineering Software*, 68:56–69, 2014.

K. Fukami, K. Fukagata, and K. Taira. Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics*, 870:106–120, 2019.

C. Geuzaine and J.-F. Remacle. Gmsh: A 3-d finite element mesh generator with built-in pre-and post-processing facilities. *International journal for numerical methods in engineering*, 79(11): 1309–1331, 2009.

X. Guo, W. Li, and F. Iorio. Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 481–490, 2016.

R. Han, Y. Wang, Y. Zhang, and G. Chen. A novel spatial-temporal prediction method for unsteady wake flows based on hybrid deep neural network. *Physics of Fluids*, 31(12):127101, 2019.

K. Hasegawa, K. Fukami, T. Murata, and K. Fukagata. Cnn-lstm based reduced order modeling of two-dimensional unsteady flows around a circular cylinder at different reynolds numbers. *Fluid Dynamics Research*, 2020.

X. Hui, J. Bai, H. Wang, and Y. Zhang. Fast pressure distribution prediction of airfoils using deep learning. *Aerospace Science and Technology*, page 105949, 2020.

X. Jin, P. Cheng, W.-L. Chen, and H. Li. Prediction model of velocity field around circular cylinder over various reynolds numbers by fusion convolutional neural networks based on pressure on the cylinder. *Physics of Fluids*, 30(4):047105, 2018.

A. Kashefi and A. E. Staples. A finite-element coarse-grid projection method for incompressible flow simulations. *Advances in Computational Mathematics*, 44(4):1063–1090, 2018.

J. Kim and C. Lee. Prediction of turbulent heat transfer using convolutional neural networks. *Journal of Fluid Mechanics*, 882, 2020.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

A. Kumar De and A. Dalal. Numerical simulation of unconfined flow past a triangular cylinder. *International journal for numerical methods in fluids*, 52(7):801–821, 2006.

C. J. Lapeyre, A. Misdariis, N. Cazard, D. Veynante, and T. Poinsot. Training convolutional neural networks to estimate turbulent sub-grid scale reaction rates. *Combustion and Flame*, 203:255–264, 2019.

R. Mittal and S. Balachandar. Direct numerical simulation of flow past elliptic cylinders. *Journal of Computational Physics*, 124(2):351–367, 1996.

V. Moureau, P. Domingo, and L. Vervisch. Design of a massively parallel cfd code for complex geometries. *Comptes Rendus Mécanique*, 339(2-3):141–148, 2011.

W. E. Nagel, D. Kröner, and M. M. Resch. *High Performance Computing in Science and Engineering'18*. Springer, 2019.

C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

V. Sekar, Q. Jiang, C. Shu, and B. C. Khoo. Fast flow field prediction over airfoils using deep learning approach. *Physics of Fluids*, 31(5):057103, 2019.

S. Sen, S. Mittal, and G. Biswas. Flow past a square cylinder at low reynolds numbers. *International Journal for Numerical Methods in Fluids*, 67(9):1160–1174, 2011.

N. Thuerey, K. Weißenow, L. Prantl, and X. Hu. Deep learning methods for reynolds-averaged navier–stokes simulations of airfoil flows. *AIAA Journal*, pages 1–12, 2019.

J. Tompson, K. Schlachter, P. Sprechmann, and K. Perlin. Accelerating eulerian fluid simulation with convolutional networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3424–3433. JMLR. org, 2017.

H. G. Weller, G. Tabor, H. Jasak, and C. Fureby. A tensorial approach to computational continuum mechanics using object-oriented techniques. *Computers in physics*, 12(6):620–631, 1998.

Y. Zhang, W. J. Sung, and D. N. Mavris. Application of convolutional neural network to predict airfoil lift coefficient. In *2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference*, page 1903, 2018.

W. Zhong, L. Deng, and Z. Xiao. Flow past a rectangular cylinder close to a free surface. *Ocean Engineering*, 186:106118, 2019.

9 Appendix

Some relevant figures are listed in this section specifically for those who are less familiar with the area of computatinoal fluid dynamics.

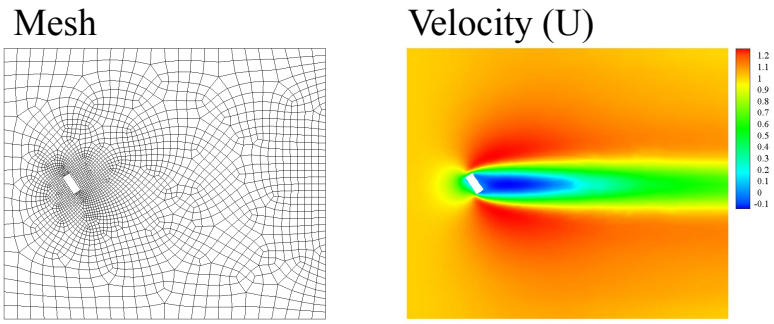

Figure 4: Finite volume mesh and the corresponding velocity magnitude field for a rectangular cross section