mathematics

MDPI

Article

A Data Augmentation-Based Technique for Deep Learning Applied to CFD Simulations

Alvaro Abucide-Armas [1], Koldo Portal-Porras [2], Unai Fernandez-Gamiz [2], Ekaitz Zulueta [1,*] and Adrian Teso-Fz-Betoño [1]

[1] Automatic Control and System Engineering Department, Faculty of Engineering of Vitoria-Gasteiz, University of the Basque Country UPV/EHU, Nieves Cano 12, 01006 Vitoria-Gasteiz, Spain; aabucide002@ehu.eus (A.A.-A.); ateso001@ehu.eus (A.T.-F.-B.)

[2] Nuclear Engineering and Fluid Mechanics Department, Faculty of Engineering of Vitoria-Gasteiz, University of the Basque Country UPV/EHU, Nieves Cano 12, 01006 Vitoria-Gasteiz, Spain; koldo.portal@ehu.eus (K.P.-P.); unai.fernandez@ehu.eus (U.F.-G.)

* Correspondence: ekaitz.zulueta@ehu.eus

Abstract: The computational cost and memory demand required by computational fluid dynamics (CFD) codes simulations can become very high. Therefore, the application of convolutional neural networks (CNN) in this field has been studied owing to its capacity to learn patterns from sets of input data, which can considerably approximate the results of the CFD simulations with relative low errors. DeepCFD code has been taken as a basis and with some slight variations in the parameters of the CNN, while the net is able to solve the Navier–Stokes equations for steady turbulent flows with variable input velocities to the domain. In order to acquire extensive input data to the CNN, a data augmentation technique, which considers the similarity principle for fluid dynamics, is implemented. As a consequence, DeepCFD is able to learn the velocities and pressure fields quite accurately, speeding up the time-consuming CFD simulations.

Keywords: data augmentation; computational fluids dynamics; convolutional neural networks; DeepCFD

Citation: Abucide-Armas, A.; Portal-Porras, K.; Fernandez-Gamiz, U.; Zulueta, E.; Teso-Fz-Betoño, A. A Data Augmentation-Based Technique for Deep Learning Applied to CFD Simulations. *Mathematics* **2021**, *9*, 0. https://doi.org/10.3390/math9161843

Academic Editor: Stefania Cherubini

Received: 29 June 2021
Accepted: 29 July 2021
Published: 4 August 2021

1. Introduction

For many years, turbulence in fluids has been a popular research topic thanks to its impact on a wide variety of applications. To study this phenomenon, CFD simulations are a frequently used tool, as they avoid the need for experimental tests, reducing costs. Nonetheless, CFD simulations generally require the repeated manipulation of thousands, or even millions, of numbers. As a consequence, the computational resources required to conduct these simulations are very high, becoming prohibitive in cases where the geometry is very complex or fine meshing is required. Despite exponential advances in computing over the last few decades, which enable increasingly detailed and sophisticated CFD problem solving, it remains a limitation for product development in a wide range of applications, such as aerodynamic design optimization and fluid–structure interaction [1]. This fact, in addition to the growth in recent years in the development of artificial intelligence, has resulted in many authors using deep learning (DL) techniques to obtain an approximation of the results of CFD simulations.

In terms of deep learning techniques applied to CFD, two types of approaches can be highlighted. The first approach consists of improving the results obtained using a coarse mesh, in order to reduce the computational time of the simulations. For example, Bao et al. [2] applied a physically driven approach to improve the modelling and simulation capability of a coarse mesh, and Hanna et al. [3] designed a DL algorithm to predict and decrease the error of the results obtained on a coarse mesh.

The second approach involves the direct calculation of the desired fluid characteristics. Guo et al. [4] created a convolutional neural network (CNN) to predict stationary

flow fields around solid objects, obtaining slightly inaccurate, but very fast predictions. Ribeiro et al. [5], taking the CNN of Guo et al. [4] as a benchmark, created a very accurate CNN for predicting velocity and pressure fields of stationary fluids around simple shaped obstacles, reducing the computational cost between three and five orders of magnitude in comparison with CFD simulations. Kashefi et al. [6] designed an ANN to obtain the same fields of the previously mentioned studies, and different velocity and pressure fields were obtained with slight modifications of the geometry, which is an essential parameter for optimizing the design.

There are other studies where predictions of other, more specific flow characteristics have been obtained. For example, Ling et al. [7] modelled Reynolds stress tensors with Reynolds-averaged Navier–Stokes (RANS) turbulence modelling using a deep neural network (DNN), achieving a remarkable improvement of the results obtained in CFD simulations. Lee and You [8] predicted the shedding of non-stationary laminar vortices on a circular cylinder using a generative adversarial network (GAN). Liu et al. [9] and Deng et al. [10], using DL-based techniques, designed methods for detecting impacts and vortexes, respectively.

While most studies are focused on the predictions of laminar flows, some authors have created diverse neural networks to predict turbulent flows. Fang et al. [11], by means of DL techniques, predicted turbulent flows on a channel, and Thuerey et al. [12] approximated the velocity and pressure fields of the RANS-based Spalart–Allmaras turbulence model on airfoils with a CNN.

Despite the existence of some studies analysing three-dimensional domains, such as Guo et al. [4] and Nowruzi et al. [13], most studies of this type are focused on 2D geometries, owing to the limited computational domain available for 3D geometries [14]. To avoid this problem, Mohan et al. [14] developed a DL-based infrastructure that reduces the geometry in order to subsequently analyse the characteristics of the flow.

Deep learning usually needs a large amount of data to properly train the large number of weights of a CNN; see the studies of Hartling et al. [15] and Fromm et al. [16]. It is common to apply data augmentation techniques when large data are required, such as in the studies of Taylor and Nitschke [17] and Sajjad et al. [18]. The most common data augmentation techniques in CNN networks are rotations, for example, Okafor et al. [19]; translations, for instance Gupta et al. [20]; and other geometric transformations, such as white noise perturbations. All these data augmentation techniques are well modelled for image processing; however, those techniques are not suitable in fluid mechanics analysis, as 2D outputs and inputs are not images. They typically are 2D flow fields of pressure and velocity or SDF matrices.

In the current work, the proposed CNN learns the velocities and pressure fields for variable input velocities to the computational domain. The study starts with 61 CFD simulations for a circular cylinder geometry. In order to improve the training of the net, a novel technique of data augmentation is proposed based on fluid dynamics similarity principle. This technique generates different synthetics cases to increase the training and validation data, keeping the Reynolds number constant.

2. Materials and Methods

2.1. Numerical Simulations

The incompressible two-dimensional Navier–Stokes [21] equations for mass (1) and momentum (2) conservation read as follows:

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

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

where u is the velocity field (with x and y components for two-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 follows:

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

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

where g represents the gravitational acceleration and ν is 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 [5].

Numerical simulations were conducted in the present study in order to obtain samples for training, validating, and testing the evaluated network. To conduct these simulations, Star-CCM+ [22] commercial code was used.

Regarding the numerical domain, a rectangular two-dimensional computational domain with a circular cylinder inside is considered; see Aramendia et al. [23]. The left side of the domain is set as inlet, and the right side as outlet. No-slip conditions are assigned to the circular cylinder and slip conditions to the top and bottom sides. The diameter of the circle (D) is equal to 10 mm, 20 mm, and 30 mm, depending on the case, and the rest of the domain is normalized with the diameter. The dimensions of the domain are 10D $\times$ 25D, and the center of the circular cylinder is located at 5D from the inlet and from both slip walls. A detailed view of the computational domain and its dimensions is provided in Figure 1.

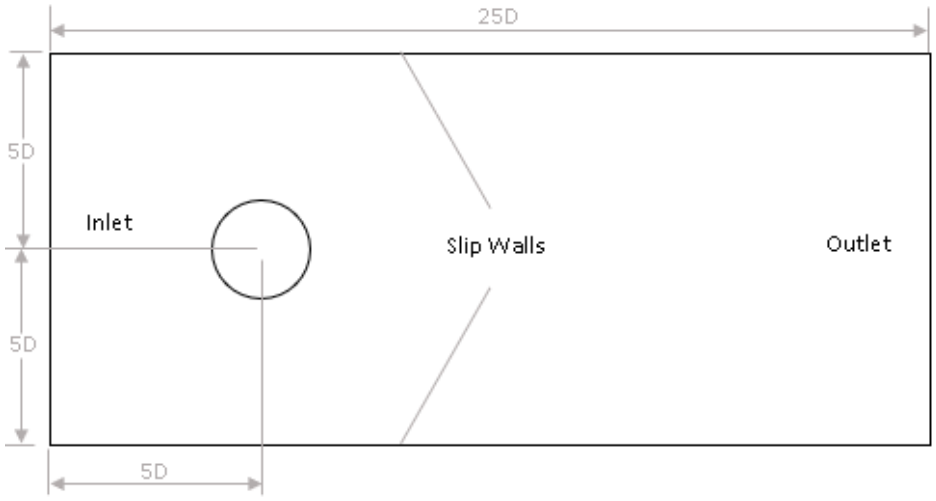


Figure 1. Computational domain (not to scale).

With this domain, a mesh that consists of two-dimensional polyhedral cells is generated. Most of the cells are located downstream the cylinder, in order to capture the vortexes in the wake behind the circle. Additionally, a volumetric control has been designed to refine the mesh around the body, as illustrated in Figure 2, in order to keep the y+ value below 1.

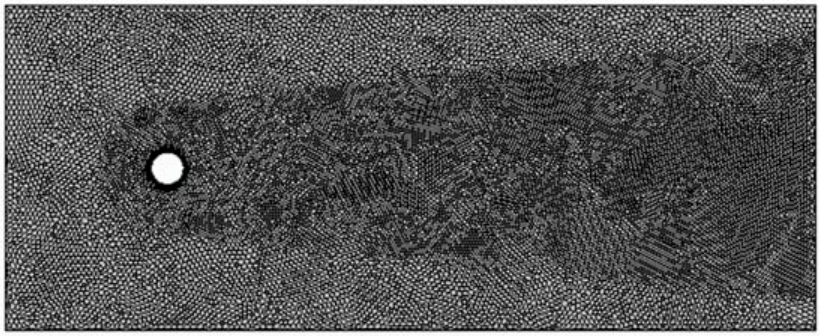


Figure 2. Mesh distribution around the cylinder.

With regards to the fluid, incompressible turbulent unsteady natural gas is considered. The density (ρ) of the selected fluid is equal to 0.7336 kg/m^3 and its dynamic viscosity (μ) is equal to $1.19 \cdot 10^{-5}$ Pa·s. These magnitudes are assumed to be constant. The velocity at the inlet (u_∞) ranges between 5 ft/s and 100 ft/s, with an interval of 5 ft/s between samples, except for D = 10 mm, which has one extra case of inlet velocity of 105 ft/s. This means that, considering the three studied diameters, a total of 61 simulations were carried out.

The numerical solution was reached by applying Menter's k-ω SST [24] RANS-based turbulence model for unsteady state flow in a finite volume flow solver. Upwind scheme [25] was used to discretize the convective terms, ensuring the robustness of the solution. Similar cases have been applied in the studies of Mahbubar et al. [26] and Rajani et al. [27]. All the simulations were converged until a satisfactory residual convergence was achieved on the velocities, pressure, and turbulence quantities.

Finally, after running the simulations, a linear interpolation is implemented, which fits the domain into a grid of 79 × 172. This step is necessary to adjust the results of the simulations to the selected neural network.

2.2. CNN Architecture

A CNN is a non-linear universal approximator of a function. Thus, a CNN can be taken as a non-linear regression. The synaptic weights are the parameters of the non-linear approximator. In addition, many data are usually required in order to obtain a well-posed non-linear regressor.

In the present study, the CNN proposed by Ribeiro et al. [5], denominated as DeepCFD, is applied. This network uses a U-Net architecture [28], which is a special case of an encoder–decoder network. This network architecture basically consists of reducing the input data to a latent geometry representation (LGR) by means of convolution layers, and subsequently, extending the LGR again using transposed convolutions layers, obtaining the velocity and pressure fields of the flow. In this case, a single encoder is used to obtain the LGR, and one decoder is used for each variable, as shown in Figure 3. Python 3.9.6 [29] software was used to train and test the proposed network.

The encoding part is structured in four encoder blocks. Each encoding block is composed of three convolution layers, with a kernel size of 5. Each layer has its respective ReLU (rectifier linear unit) layer, and the last convolution layer of each block contains a max pooling layer. The number of filters of each block, from the outermost to the innermost, is equal to 8, 16, 32, and 32. As mentioned above, the network Contains three decoder parts, one for each variable (u_x, u_y, and p). In these decoder parts, the reverse process of the encoding part process is carried out. Figure 4 provides a detailed view of the architecture of DeepCFD.

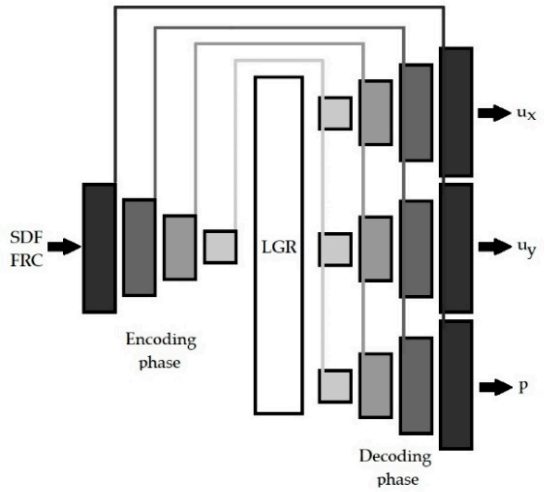


Figure 3. U-Net with three decoders.

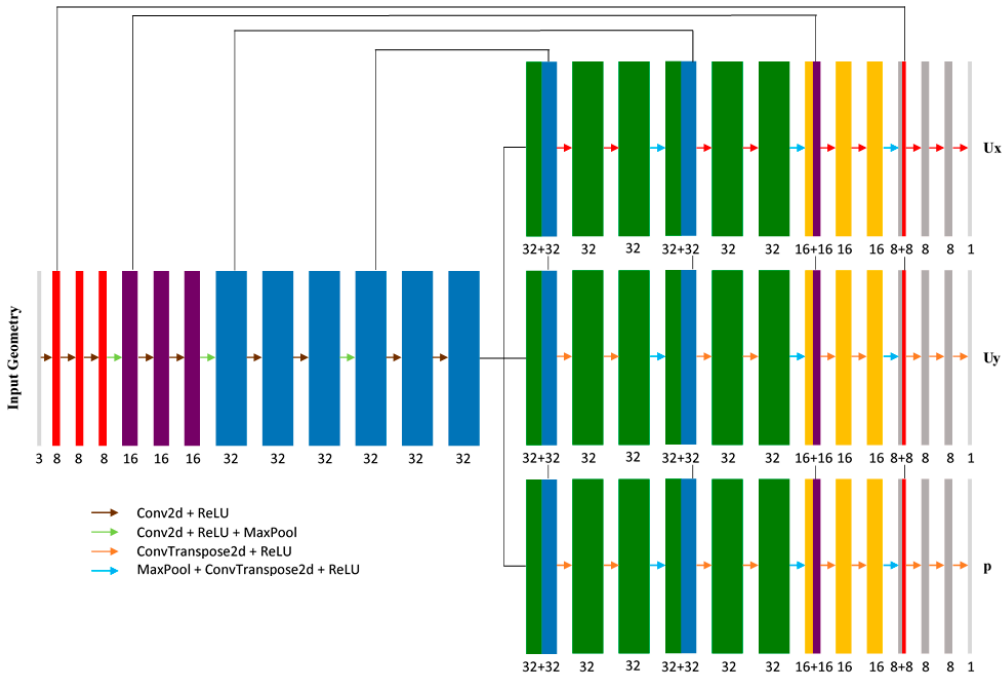


Figure 4. DeepCFD architecture.

2.3. Training Parameters

DeepCFD is trained with a loss function that combines the errors of the three outputs, velocity component fields, and pressure field, as proposed by Ribeiro et al. [5]. However, for this work, in each experiment, the type of loss function utilized for each field was modified in order to seek the lowest possible error rates.

AdamW is the optimizer selected for training the DeepCFD network. This algorithm is based on the adaptative moment estimation (Adam), which updates the gradient vector and the squared gradient using the exponential moving average [30]. Given the parameters $w^{(t)}$ and a loss function L^t, where t indicates the current training iteration. Assigning t = 0 for the first iteration, the Adam parameters' update is given by the following expressions:

$$m_w^{(t+1)} \leftarrow \beta_1 m_w^{(t)} + (1-\beta_1)\nabla_w L^t, \tag{5}$$

$$v_w^{(t+1)} \leftarrow \beta_2 v_w^{(t)} + (1-\beta_2)\left(\nabla_w L^t\right)^2, \tag{6}$$

$$\hat{m}_w = \frac{m_w^{(t+1)}}{1-\beta_1^{t+1}} \tag{7}$$

$$\hat{v}_w = \frac{v_w^{(t+1)}}{1-\beta_2^{t+1}} \tag{8}$$

$$w^{t+1} \leftarrow w^{(t)} - \eta \frac{\hat{m}_w}{\sqrt{\hat{v}_w} + \epsilon} \tag{9}$$

where ϵ is a small scalar (e.g., 10^{-8}) used to prevent division by 0 and β_1 (e.g., 0.9) and β_2 (e.g., 0.999) are the forgetting factors for gradients and second moments of gradients, respectively. Squaring and square-rooting is done elementwise [31].

AdamW is an updated version of the Adam optimizer, which improves regularization by decoupling the weight decay from the gradient-based update [32]. For this work, the learning rate is set at 0.001, the weight decay at 0.005, the batch size at 64, and the training ratio at 0.9.

2.4. DeepCFD Inputs

In this network, two different input layers are considered. These inputs are a signed distance function (SDF) of the geometry and the flow region channel (FRC).

SDF is a function that measures the distance between any point in the grid and the nearest boundary of a closed geometry shape. The sign of each value depends on whether the point is inside (negative) or outside (positive) the shape [4]. The mathematical expression of this function is given by Expression (10).

$$SDF(x) = \begin{cases} d(x,\ \vartheta\Omega)\ if\ x \in \Omega \\ -d(x,\ \vartheta\Omega)\ if\ x \in \Omega^c \end{cases}, \tag{10}$$

where Ω is a subset of a metric space, X, with metric, d, and $\vartheta\Omega$ is the boundary of Ω. For any x $\in$ X:

$$d(x,\ \vartheta\Omega) := \inf_{y \in \vartheta\Omega} d(x,y), \tag{11}$$

where inf denotes the infimum. Grid positions inside the interior of the obstacle (Ω^c) are assigned negative distances [5]. Figure 5 shows the SDF generated for this work with a MATLAB [33] code, which allows selecting the position and size of the geometry as well as the size of the grid.

Figure 5. Signed distance function (SDF) of a circle.

The FRC input layer is a multi-class channel with information about the boundary conditions of the domain. In this case, the first column, which refers to the input, is set with the input velocity of each case. The upper and bottom walls and the outlet are replaced by

free flow regions, as slip conditions are employed in this work for the limits of the domain. Figure 6 displays and schematic view of the FRC layer.

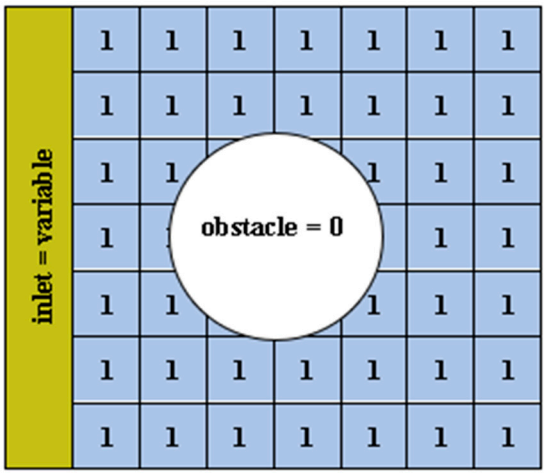


Figure 6. Diagram of the multi-class labelling of flow regions.

2.5. Data Augmentation

Data augmentation is a well-known technique in conventional deep learning. The technique consists of generating realistic synthetic data to increase the data quantity in the learning process. Deep learning usually applies data augmentation with simple geometric transformations and perturbations added to real data. In the current study, simple geometric transformation or perturbation to generate realistic synthetic data cannot be applied. The data augmentation technique of the current work profits from Reynolds number properties for equivalent fluid-dynamics systems.

The training of DeepCFD for the circle geometry CFD simulations is incapable of learning all the patterns of the velocities and pressure fields, principally the vortex shedding. In order to fix this issue, a data augmentation technique is implemented, which consist of enlarging the number of input cases of the CNN from 61 to 610.

The data augmentation is performed taking into account the similarity theory for fluid dynamics. For any simulation, if the Reynolds number is kept constant in each case, the new input velocity of the domain as well as the new velocities and pressure fields can be calculated. The Reynolds number is calculated by Equation (12).

$$Re = \frac{u_\infty D \rho}{\mu} \tag{12}$$

Assuming that the Reynolds number is equal in all the cases, the new input velocity of each case ($u^*_{\infty i}$) can be calculated with Equation (13). Because, in all the cases, the fluid and the boundary conditions are constant, the density and dynamic viscosity have no influence on the velocity. Consequently, with slight modifications in the value of the diameter, the input data of the CNN notoriously increase.

$$u^*_{\infty i} = \frac{D_1}{D_i} u_{\infty 1} \tag{13}$$

The expressions of the velocity and pressure fields of each different case are given by Equations (14)–(16):

$$\hat{u}_{xi}\left(\hat{x}, \hat{y}\right) = \frac{u_{xi}}{u^*_{\infty i}}\left(\frac{x_i}{D_i}, \frac{y_i}{D_i}\right) \tag{14}$$

$$\hat{u}_{yi}\left(\hat{x}, \hat{y}\right) = \frac{u_{yi}}{u^*_{\infty i}}\left(\frac{x_i}{D_i}, \frac{y_i}{D_i}\right) \tag{15}$$

$$\hat{p}_i\left(\hat{x},\hat{y}\right) = \frac{p_{xi}}{u^{*2}_{\infty i}\rho}\left(\frac{x_i}{D_i},\frac{y_i}{D_i}\right) \tag{16}$$

where $\hat{u}_{xi}$, $\hat{u}_{yi}$, and $\hat{p}_i$ are the new velocity and pressure fields, and $\frac{x_i}{D_i}$ and $\frac{y_i}{D_i}$ represent the relative coordinates inside the domain.

Following this process, the fields of each magnitude are calculated for any new diameter. As the similarity theory establishes, the domain size changes proportionally with the diameter. This is shown by Equation (17), where an example for points 1 and 2 of the grid is provided.

$$\hat{x} = \frac{x_1}{D_1} = \frac{x_2}{D_2} \tag{17}$$

Knowing all the previous information, and using a simple code that conducts an interpolation of the original fields, the number of data samples is easily increased. With this data augmentation method, the necessity of performing time- and resource-demanding CFD simulations for each new sample disappears.

3. Results

In order to reach the best results, different approaches have been conducted trying to reduce the error to the possible minimum rates. Table 1 shows a group of characteristics of each of the experiments performed.

Table 1. Features of the simulation.

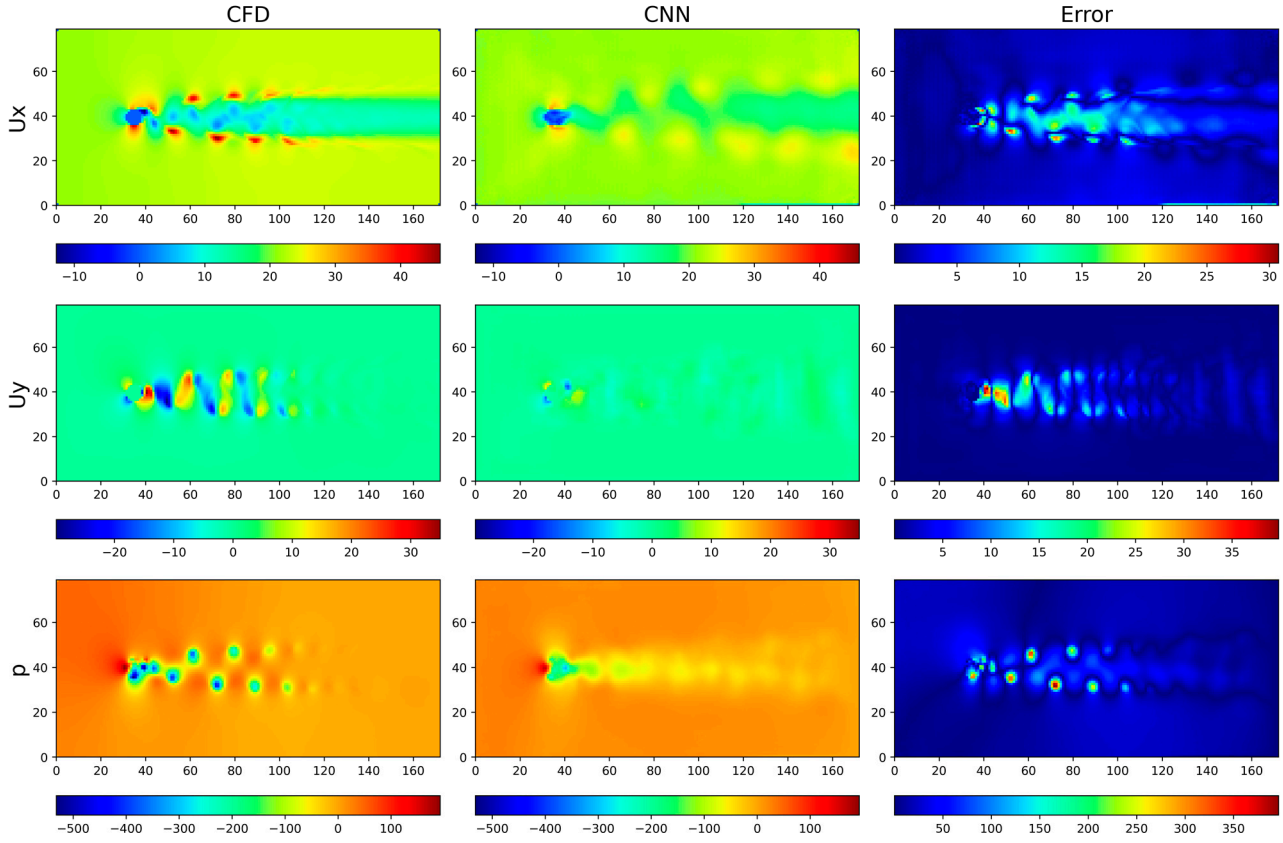
Feature	61 Cases	305 Cases	610 Cases	k = 0.01	L1-Norm
Number of samples	61	305	610	610	60
Number of epochs	10,000	1000	1000	1000	1000
Loss function	L2-norm for u_x L2-norm for u_y L1-norm for p	L2-norm for u_x L2-norm for u_y L1-norm for p	L2-norm for u_x L2-norm for u_y L1-norm for p	L2-norm for $u_x \times k$ L2-norm for u_y L1-norm for $p \times k$	L1-norm for u_x L1-norm for u_y L1-norm for p

3.1. Experiment: Data Augmentation

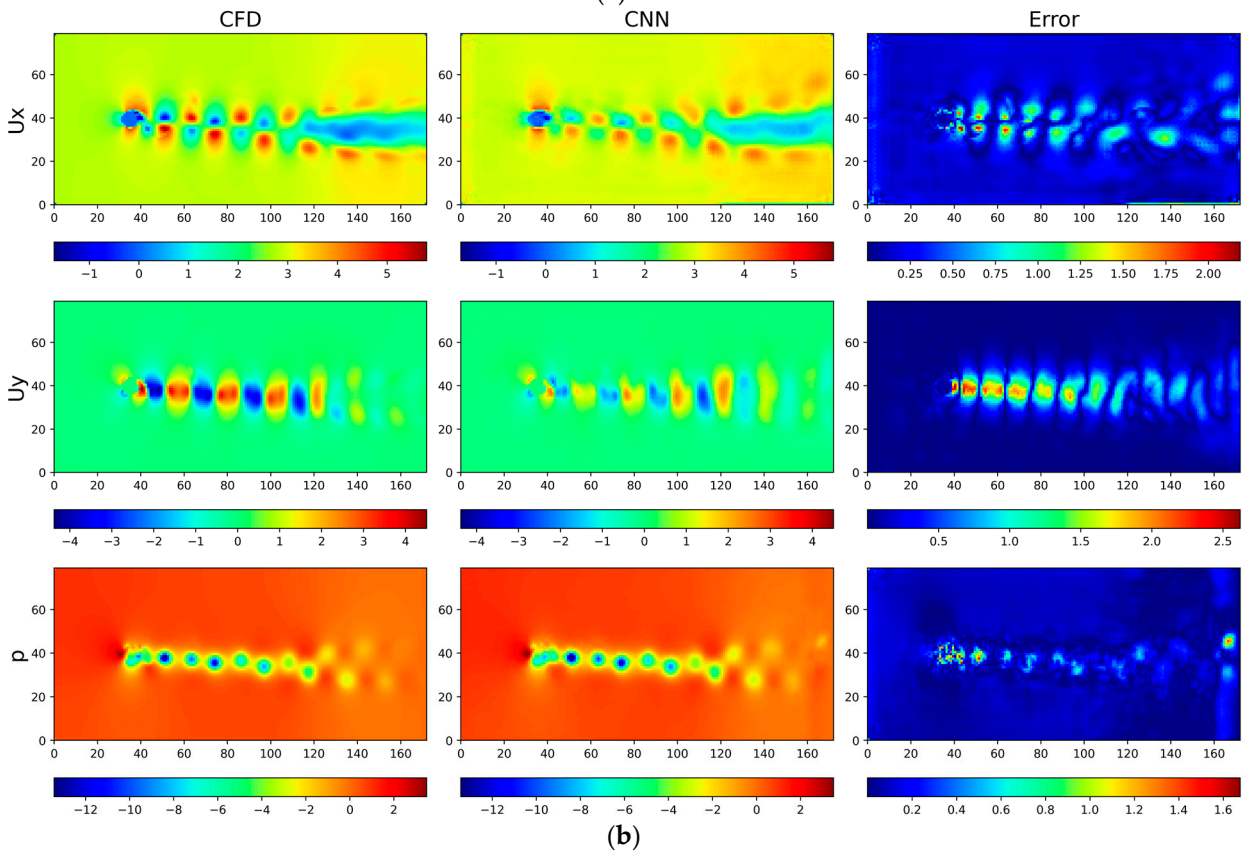
Firstly, DeepCFD has been trained with 61 (no data augmentation), 305, and 610 cases, as Figure 7 shows. Without data augmentation, despite significantly learning the points close to the geometry, the CNN does not perform quality results through the vortex shedding. Incrementing the data to 305 cases improves the results given by training the neural network solely for 61 samples. With 610 cases, the CNN achieves a decent learning of the patterns, especially for the pressure field, which is close to a zero error rate in the majority of points of the domain. Nevertheless, the enhancement in the output of the velocity fields is not precise enough.

3.2. Experiment: Modify the Loss Function

Secondly, a multiplying constant was utilized to reduce or increase the value of each loss function. After trying with different values of the multiplying constant, a value of 0.01 provided the best results in this experiment, as Figure 8 shows. Nonetheless, the error decreases uniquely at a low rate for the x component of the velocity and remains invariable for the other two outputs.



(**a**)



(**b**)

Figure 7. *Cont.*

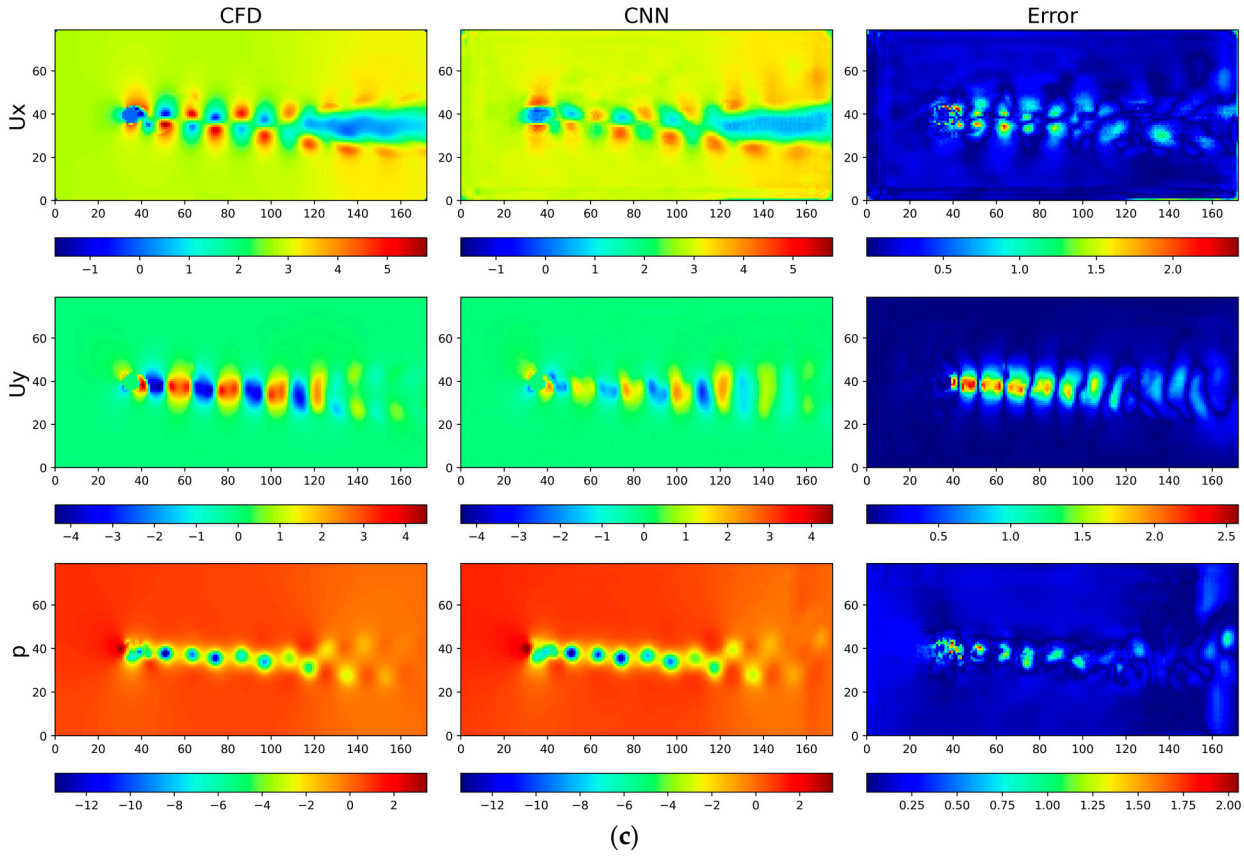


Figure 7. (**a**) Results with 61 samples (no data augmentation); (**b**) results with 305 samples; (**c**) results with 610 cases.

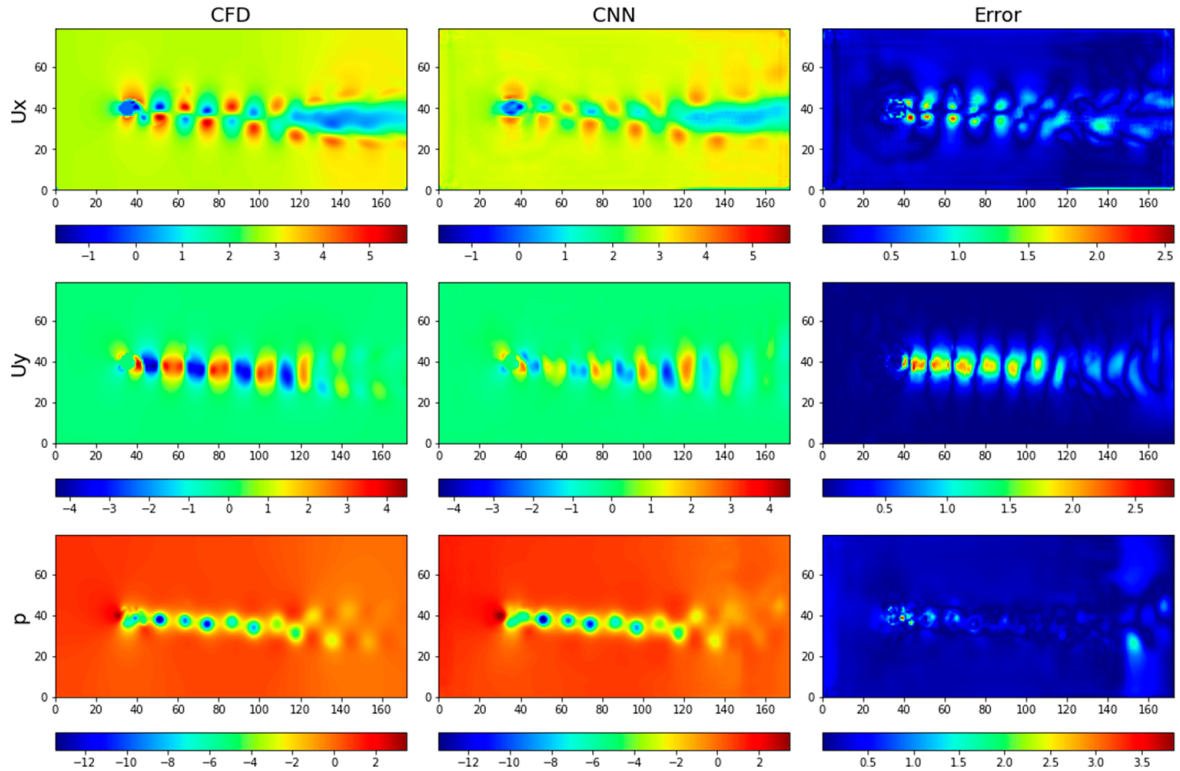


Figure 8. Test with the loss functions of the x component of the velocity and the pressure multiplied by k = 0.01.

3.3. Experiment: L1-Norm for the Three Magnitudes

With this last experiment, the error in the three outputs decreases practically to zero in most points of the grid. This error rates were reached after changing the loss function of the velocity components. In the code of Ribeiro et al. [5], a mean squared error function was utilized for the velocity magnitude, whereas for the pressure, a mean absolute error was employed. Their experiments showed considerably better convergence when L2-norm was applied to the velocity fields and L1-norm for the pressure. However, in the current conditions, the mean absolute error has proven itself the best type of loss function for the three magnitudes. In comparison with Figure 7c, where a L2-norm for velocity fields was used, Figure 9 shows that there is a void error rate in almost every point in each of the three grids.

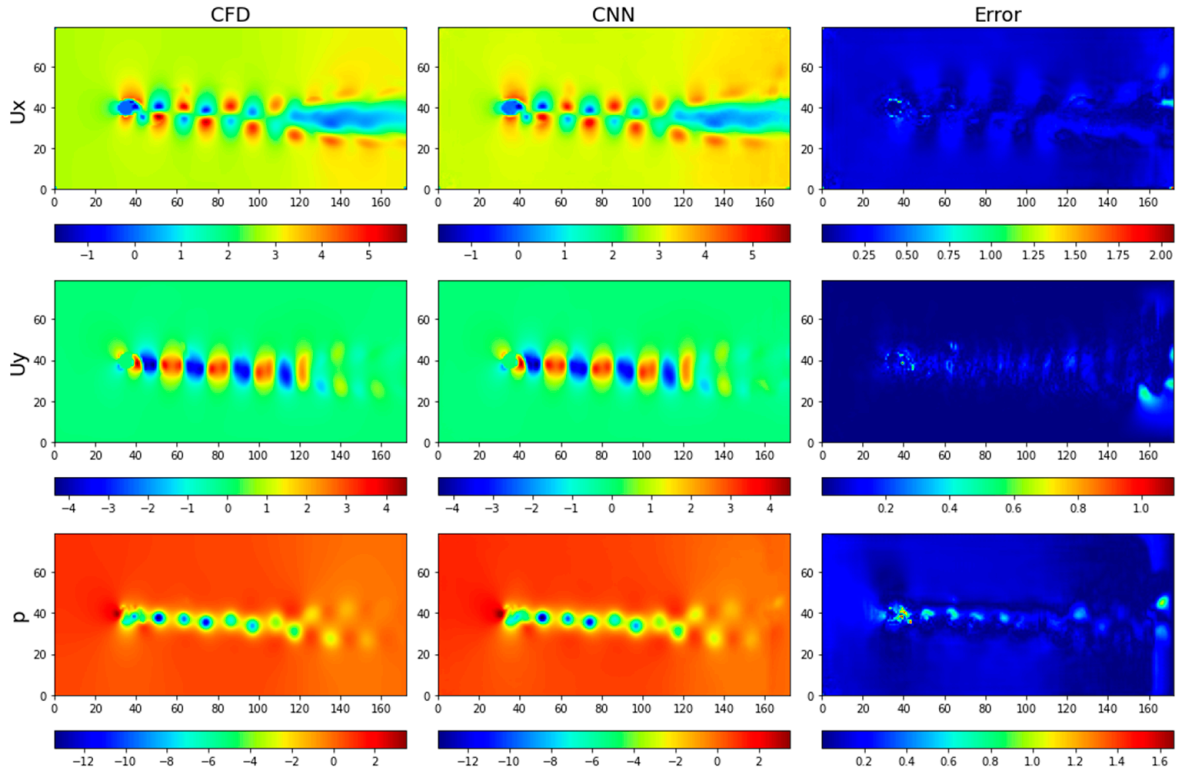
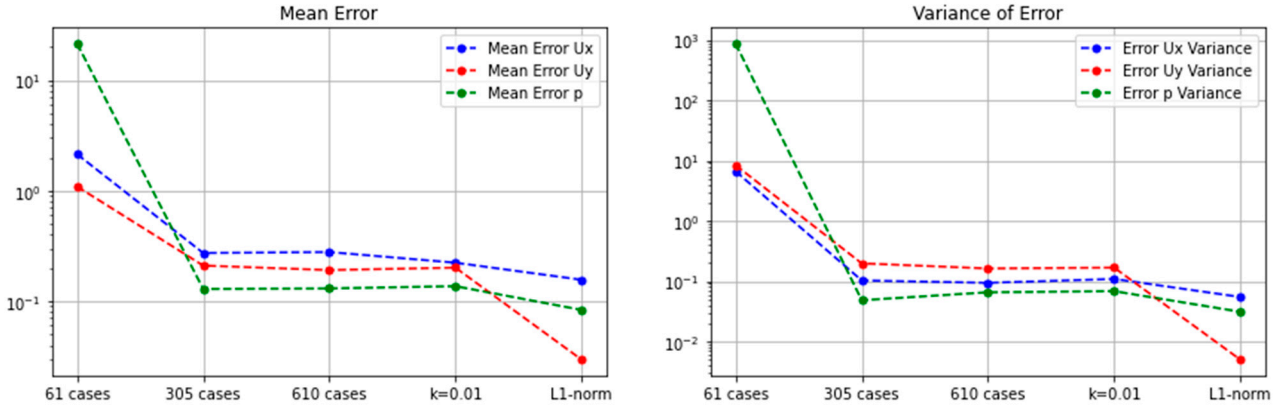


Figure 9. Test with mean absolute error as the loss function of each of the fields.

3.4. Error Analysis

Figure 10 shows the evolution of the error in the five cases simulated. The plots confirm that the data augmentation outperforms the simulation for only 61 samples. Experiment with k = 0.01 allows a reduction in the u_x error with a practically unnoticeable variation in the other two magnitudes. Additionally, the use of an L1-norm for the three magnitudes involves a considerable decrease in the mean error and variance of error for the three outputs. For instance, the data augmentation, with 610 cases, reduces the mean error by 87.0%, 85.5%, and 99.4% for the u_x, u_y, and pressure magnitudes, respectively, in comparison with the simulation with 61 samples. The difference between 610 and 305 samples is an 8.9% decrease in the vertical velocity component, but for the horizontal velocity and pressure, the mean error rises by 1.9% and 0.96%, respectively. With k = 0.01, the mean error is a 19.9% less for u_x than with 610 samples, but increases by 5.2% and 4.9% for u_y and pressure, respectively. Applying a, L1-norm for the three outputs achieves a decrease of the mean error by 43.9%, 84.5%, and 36.0% in comparison with the 610 cases example.



(**a**) (**b**)

Figure 10. Error rates for the five simulations. (**a**) Mean error; (**b**) variance of error.

4. Conclusions

CNN-based techniques usually need large amounts of input data in order to produce accurate results; therefore, the implementation of data augmentation is regarded as a reasonable solution for many cases. The data augmentation technique developed in this study allows the creation of multiple synthetic samples from an original CFD simulation, thus avoiding executing time-consuming CFD simulation for each new generated sample. This technique reduces the time employed for the generation of the input data of the CNN and improves the training. In this case, generating each of the 61 initial samples required 3 h, which means a total of 183 h for generating the 61 samples. For its part, for generating the other 549 samples (the ones from data augmentation), 4 h was required.

Moreover, the results of the current work, specifically the ones given by the L1-norm for the three magnitudes experiment, confirm the hypothesis of a CNN being able to learn rapidly the axial and streamwise velocity and pressure fields formed behind a 2D circular cylinder, for turbulent flow and variable input velocities with low error rates.

Author Contributions: Conceptualization, A.A.-A.; methodology, K.P.-P. and A.A.-A.; software, E.Z. and A.T.-F.-B.; validation, A.A.-A., E.Z. and U.F.-G.; formal analysis, K.P.-P. and A.A.-A.; investigation, A.T.-F.-B. and A.A.-A.; resources, U.F.-G. and E.Z.; data curation, A.A.-A.; writing—original draft preparation, A.A.-A. and K.P.-P.; writing—review and editing, K.P.-P. and U.F.-G.; visualization, A.A.-A. and E.Z.; supervision, U.F.-G.; project administration, U.F.-G. and E.Z.; funding acquisition, U.F.-G. and E.Z. All authors have read and agreed to the published version of the manuscript.

Funding: The authors were supported by the government of the Basque Country through research grants ELKARTEK 21/10: BASQNET: Estudio de nuevas técnicas de inteligencia artificial basadas en Deep Learning dirigidas a la optimización de procesos industriales.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available on request from the corresponding author.

Acknowledgments: The authors are grateful for the support provided by SGIker of UPV/EHU. This research was developed under the frame of the Joint Research Laboratory on Offshore Renewable Energy (JRL-ORE). E-CLEDER research team is also acknowledged.

Conflicts of Interest: The authors declare no conflict of interest.

Nomenclature

ANN	Artificial neural network
CFD	Computational fluid dynamics
CNN	Convolutional neural network
DL	Deep learning
DNN	Deep neural network
FRC	Flow region channel
GAN	Generative adversarial network
LGR	Latent geometry representation
RANS	Reynolds-averaged Navier–Stokes
ReLU	Rectifier linear unit
SDF	Signed distance function
Adam	Adaptative moment estimation
β_1	Forgetting factor for gradients
β_2	Forgetting factor for second moment gradients
D	Diameter of the circle
F	Body forces
K	Multiplying constant for the loss functions
L^t	Loss function
P	Fluid density
p	Pressure
Re	Reynolds number
T	Current training iteration
T	Stress tensor
M	Fluid dynamic viscosity
u_x	Horizontal velocity
u_y	Vertical velocity
u_∞	Inlet velocity to the domain

References

1. Anderson, J.D., Jr. Basic Philosophy of CFD. In *Computational Fluids Dynamics*; Springer: Rhode-Saint-Genèse, Belgium, 2009; pp. 3–15.
2. Bao, H.; Feng, J.; Dinh, N.; Zhang, H. Computationally efficient CFD prediction of bubbly flow using physics-guided deep learning. *Int. J. Multiph. Flow* **2020**, *131*, 103378. [CrossRef]
3. Hanna, B.N.; Dinh, N.T.; Youngblood, R.W.; Bolotnov, I.A. Coarse-Grid Computational Fluid Dynamic (CG-CFD) Error Prediction Using Machine Learning. *arXiv* **2017**, arXiv:1710.09105.
4. Guo, X.; Li, W.; Iorio, F. Convolutional Neural Networks for Steady Flow Approximation. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; ACM: San Francisco, CA, USA, 2016; pp. 481–490.
5. Ribeiro, M.D.; Rehman, A.; Ahmed, S.; Dengel, A. DeepCFD: Efficient Steady-State Laminar Flow Approximation with Deep Convolutional Neural Networks. *arXiv* **2020**, arXiv:2004.08826.
6. Kashefi, A.; Rempe, D.; Guibas, L.J. A point-cloud deep learning framework for prediction of fluid flow fields on irregular geometries. *Phys. Fluids* **2021**, *33*, 027104. [CrossRef]
7. Ling, J.; Kurzawski, A.; Templeton, J. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *J. Fluid Mech.* **2016**, *807*, 155–166. [CrossRef]
8. Lee, S.; You, D. Prediction of Laminar Vortex Shedding over a Cylinder Using Deep Learning. *arXiv* **2017**, arXiv:1712.07854.
9. Liu, Y.; Lu, Y.; Wang, Y.; Sun, D.; Deng, L.; Wang, F.; Lei, Y. A CNN-based shock detection method in flow visualization. *Comput. Fluids* **2019**, *184*, 1–9. [CrossRef]
10. Deng, L.; Wang, Y.; Liu, Y.; Wang, F.; Li, S.; Liu, J. A CNN-based vortex identification method. *J. Vis.* **2018**, *22*, 65–78. [CrossRef]
11. Fang, R.; Sondak, D.; Protopapas, P.; Succi, S. Deep Learning for Turbulent Channel Flow. *arXiv* **2018**, arXiv:1812.02241.
12. Thuerey, N.; Weißenow, K.; Prantl, L.; Hu, X. Deep Learning Methods for Reynolds-Averaged Navier–Stokes Simulations of Airfoil Flows. *AIAA J.* **2020**, *58*, 25–36. [CrossRef]
13. Nowruzi, H.; Ghassemi, H.; Ghiasi, M. Performance predicting of 2D and 3D submerged hydrofoils using CFD and ANNs. *J. Mar. Sci. Technol.* **2017**, *22*, 710–733. [CrossRef]
14. Mohan, A.; Daniel, D.; Chertkov, M.; Livescu, D. Compressed Convolutional LSTM: An Efficient Deep Learning Framework to Model High Fidelity 3D Turbulence. *arXiv* **2019**, arXiv:1903.00033.
15. Hartling, S.; Sagan, V.; Sidike, P.; Maimaitijiang, M.; Carron, J. Urban Tree Species Classification Using a WorldView-2/3 and LiDAR Data Fusion Approach and Deep Learning. *Sensors* **2019**, *19*, 1284. [CrossRef] [PubMed]

16. Fromm, M.; Schubert, M.; Castilla, G.; Linke, J.; McDermid, G. Automated Detection of Conifer Seedlings in Drone Imagery Using Convolutional Neural Networks. *Remote Sens.* **2019**, *11*, 2585. [CrossRef]
17. Taylor, L.; Nitschke, G. Improving Deep Learning Using Generic Data Augmentation. *arXiv* **2017**, arXiv:1708.06020.
18. Sajjad, M.; Khan, S.; Muhammad, K.; Wu, W.; Ullah, A.; Baik, S.W. Multi-grade brain tumor classification using deep CNN with extensive data augmentation. *J. Comput. Sci.* **2019**, *30*, 174–182. [CrossRef]
19. Okafor, E.; Schomaker, L.; Wiering, M.A. An analysis of rotation matrix and colour constancy data augmentation in classifying images of animals. *J. Inf. Telecommun.* **2018**, *2*, 465–491. [CrossRef]
20. Gupta, A.; Venkatesh, S.; Chopra, S.; Ledig, C. Generative Image Translation for Data Augmentation of Bone Lesion Pathology. In Proceedings of the International Conference on Medical Imaging with Deep Learning, London, UK, 8–10 July 2019; pp. 225–235.
21. Çengel, Y.A.; Cimbala, J.M. *Fluid Mechanics: Fundamentals and Applications*, 4th ed.; McGraw-Hill Education: New York, NY, USA, 2018; ISBN 978-1-259-69653-4.
22. STAR-CCM+ V2019.1. Available online: https://www.plm.automation.siemens.com/ (accessed on 2 June 2020).
23. Aramendia, I.; Fernandez-Gamiz, U.; Guerrero, E.Z.; Lopez-Guede, J.M.; Sancho, J. Power Control Optimization of an Underwater Piezoelectric Energy Harvester. *Appl. Sci.* **2018**, *8*, 389. [CrossRef]
24. Menter, F.R. Two-equation eddy-viscosity turbulence models for engineering applications. *AIAA J.* **1994**, *32*, 1598–1605. [CrossRef]
25. Osher, S.; Chakravarthy, S. Upwind schemes and boundary conditions with applications to Euler equations in general geometries. *J. Comput. Phys.* **1983**, *50*, 447–481. [CrossRef]
26. Rahman, M.; Karim, M.; Alim, A. Numerical Investigation of Unstedady Flow Past a Circular Cylinder Using 2-D Finite Volume Method. *J. Nav. Archit. Mar. Eng.* **2007**, *4*, 27–42. [CrossRef]
27. Rajani, B.N.; Kandasamy, A.; Majumdar, S. Numerical Simulation of Laminar Fllow Past a Circular Cylinder. *Appl. Math. Model.* **2009**, *33*, 1228–1247. [CrossRef]
28. Ronneberger, O.; Fischer, P.; Brox, T. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Medical Image Computing and Computer-Assisted Intervention—MICCAI 2015*; Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F., Eds.; Lecture Notes in Computer Science; Springer International Publishing: Cham, Switzerland, 2015; Volume 9351, pp. 234–241. ISBN 978-3-319-24573-7.
29. Python. 2019. Available online: https://www.python.org/ (accessed on 20 January 2021).
30. Han, R.; Wang, Y.; Zhang, Y. A novel spatial-temporal prediction method for unsteady wake flows based on hybrid deep neural network. *Phys. Fluids* **2019**, *31*, 127101. [CrossRef]
31. Kingma, D.; Ba, J. Adam: A Method for Stochastic Optimization. In Proceedings of the International Conference on Learning Representations, San Diego, CA, USA, 7–9 May 2015.
32. Loshchilov, I.; Hutter, F. Decoupled Weight Decay Regularization. In Proceedings of the International Conference on Learning Representations, New Orleans, LA, USA, 6–9 May 2019.
33. MATLAB. Available online: https://es.mathworks.com/products/matlab.html (accessed on 9 June 2021).