



---

*Research article*

## Unsteady-state turbulent flow field predictions with a convolutional autoencoder architecture

Álvaro Abucide<sup>1</sup>, Koldo Portal<sup>2</sup>, Unai Fernandez-Gamiz<sup>2</sup>, Ekaitz Zulueta<sup>1,\*</sup> and Iker Azurmendi<sup>1,3</sup>

<sup>1</sup> Automatic Control and System Engineering Department, University of the Basque Country UPV/EHU, Nieves Cano 12, 01006 Vitoria-Gasteiz, Spain

<sup>2</sup> Nuclear Engineering and Fluid Mechanics Department, University of the Basque Country UPV/EHU, Nieves Cano 12, 01006 Vitoria-Gasteiz, Spain

<sup>3</sup> CS Centro Stirling S. Coop., Avda. Álava 3, 20550 Aretxabaleta, Spain

\* **Correspondence:** Email: [ekaitz.zulueta@ehu.eus](mailto:ekaitz.zulueta@ehu.eus).

**Abstract:** Traditional numerical methods, such as computational fluid dynamics (CFD), demand large computational resources and memory for modeling fluid dynamic systems. Hence, deep learning (DL) and, specifically Convolutional Neural Networks (CNN) autoencoders have resulted in accurate tools to obtain approximations of the streamwise and vertical velocities and pressure fields, when stationary flows are considered. The novelty of this paper consists of predicting the future instants from an initial one with a CNN autoencoder architecture when an unsteady flow is considered. Two neural models are proposed: The former predicts the future instants on the basis of an initial sample and the latter approximates the initial sample. The inputs of the CNNs take the signed distance function (SDF) and the flow region channel (FRC), and, for the representation of the temporal evolution, the previous CFD sample is added. To increment the amount of training data of the second neural model, a data augmentation technique based on the similarity principle for fluid dynamics is implemented. As a result, low absolute error rates are obtained in the prediction of the first samples near the shapes surfaces. Even in the most advanced time instants, the prediction of the vortices zone is quite reliable. 62.12 and 9000 speed-up ratios are achieved by the predictions of the first and second neural models, respectively, compared to the computational cost regarded by the CFD simulations.

**Keywords:** computational fluid dynamics; unsteady flow; deep learning; CNN; data augmentation

**Mathematics Subject Classification:** 76F99, 65Z05

---

## 1. Introduction

The study of turbulence in fluids has been a significant research topic for many years due to its impact on a wide variety of applications. Experimentation with turbulent flows has improved the understanding of the behavior of turbulences and has allowed the design of more efficient systems. Unfortunately, experiments with turbulent flows are usually too costly and impractical. Computational fluids dynamics (CFD) simulations provide detailed enough results when constraints appear in the physical experiments. This is principally possible thanks to the exponential advances in computational resources. Nonetheless, CFD has its drawbacks. For example, when fine meshing is required or a complex geometry is analyzed, the computational resources demanded by CFD become prohibitive. The generation of the mesh and the closure model requires the user's influence, which may limit the quality of the CFD simulation. To solve these drawbacks, CFD problems have been solved with deep learning techniques [1]. In this case, unsteady flows are predicted by a convolutional neural network (CNN) architecture.

Aerodynamic shape optimization (ASO) is one of the issues in which DL has been recently involved. Qin et al. [2] proposed an artificial neural network (ANN) and deep reinforcement learning (DRL) to reach aerodynamic optimization in a blade profile. Qiu et al. [3] employed a novel optimization strategy based on the proper orthogonal decomposition (POD) method to reduce the design variables of the transonic airfoil RAE2822 and the transonic wing ONERA M6.

Two main objectives of DL in CFD applications are the reduction of computational costs and the direct calculation of some fluid features. For example, Hanna et al. [4] and Bao et al. [5] focused on diminishing the computation time in the modeling and simulation of coarse meshes. Tlaes et al. [6] reduced the computational costs by applying a clustering technique as a mesh adaptation sensor. Guo et al. [7] predicted non-uniform steady laminar flow fields around bluff body objects with low computational costs and demonstrated the capacity of generalization of the CNNs to the rapid approximation of the flow field. Ribeiro et al. [8] and Kashefi et al. [9] obtained estimations of the streamwise and vertical velocities and pressure fields for slight modifications of a series of diversely shaped geometries, using an autoencoder (AE) and a PointNet architecture, respectively. Moreover, Murata et al. [10] applied a CNN auto-encoder for the application of modal decomposition.

Instead of the estimation of the flow fields, other works focus on different purposes. Some studies focus on turbulence modeling, such as the tensor basis neural network (TBNN) architecture proposed by Ling et al. [11] to predict the Reynolds stress tensor, which provided more accurate results than the ones obtained with Reynolds-Averaged Navier-Stokes (RANS) models. Lee and You [12] used a Generative Adversarial Network (GAN) architecture to generate a surrogate model to predict the shedding of non-stationary laminar vortices on a circular cylinder. Liu et al. [13] and Deng et al. [14] applied a CNN architecture for detecting impacts and vortices, respectively.

The difficulty in the analysis of fluid dynamics features when a 3D-shaped geometry or turbulent flows are considered remarkably increases. This implies that most of the works study 2D geometries and laminar flows. The works of Guo et al. [7] and Nowruzzi et al. [15] accomplished accurate solutions with 3D geometries, assuming large computational costs. Mohan et al. [16] developed a DL framework to reduce the geometry and, consequently, the computational costs. Some studies obtained quality and computational-efficient results, regarding turbulent flows. For example, Portal-Porrás et al. [17] developed different CNN structures to predict the coupled velocity fields with turbulent flows. Abucide-Armas et al. [18] conceived a data augmentation technique to reduce

the amount of CFD simulations required to train the network and obtained low error rates for turbulent flows and variable input velocities to the domain. Thuerey et al. [19] predicted the streamwise and vertical velocities and pressure fields of the RANS-based Spalart-Allmaras turbulence model on airfoils, and Fang et al. [20] employed a TBNN to analyze the specific case of turbulent channel flow.

CFD applications frequently need to know the time evolution of the different parameters and features of a fluid. Many of the aforementioned works are based on the behavior of the fluid in a specific state. The temporal evolution of the latent vectors is obtained by a nonlinear autoencoder mixed with sparse identification of nonlinear dynamics (SINDy) in the works of Champion et al. [21] and Fukami et al. [22] and with a Gaussian process by Maulik et al. [23]. Other studies employ recurrent neural networks (RNN) to analyze the flow properties on a time-based approach. For example, Agostini [24] predicted the streamwise velocity field with an AE and CNN framework. King et al. [25], Maulik et al. [26] and Gonzalez and Balajewicz [27] predicted some flow properties on time-based approaches.

This paper aims to exploit the CFD simulated data to train different versions of a CNN architecture. The data contains the simulations of turbulent fluid streamwise and vertical velocities and the pressure fields around different-shaped geometries. First, the CNN purpose is to predict the future states of a fluid given an initial CFD obtained state and then it predicts the futures states based on the previously predicted sample. In addition, the CNN is trained to obtain the initial state based on the corresponding geometric information. The current work focuses on the prediction of the velocity and pressure when unsteady flows are considered.

## 2. Materials and methods

### 2.1. Numerical simulations

The Unsteady Reynolds Navier-Stokes (URANS) approach was adopted for the current simulations since they are usually employed when long term periodical oscillations in a turbulent flow are investigated. The URANS equations are obtained by the following procedure. The Navier-Stokes equations for incompressible flow are time-filtered according to Eq (1):

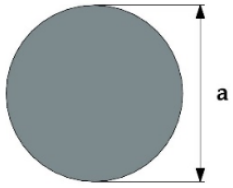
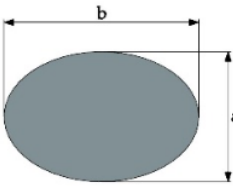
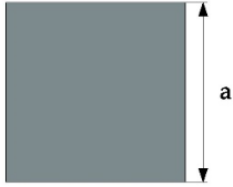
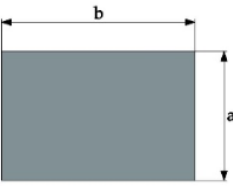
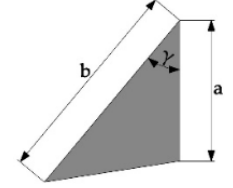
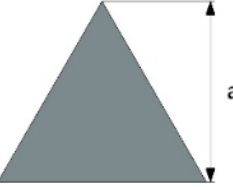
$$\frac{\delta \langle u_i \rangle}{\delta t} + \frac{\delta}{\delta x_j} (\langle u_j u_i \rangle) = -\frac{1}{\rho} \frac{\delta \langle p \rangle}{\delta x_i} + \nu \frac{\delta^2 \langle u_i \rangle}{\delta x_k^2}. \quad (1)$$

Then, the turbulent stress tensor  $\tau_{ij} = \langle u_i \rangle \langle u_j \rangle - \langle u_j u_i \rangle$  is introduced, which gives the final URANS equation:

$$\frac{\delta \langle u_i \rangle}{\delta t} + \frac{\delta}{\delta x_j} (\langle u_i \rangle \langle u_j \rangle) = -\frac{1}{\rho} \frac{\delta \langle p \rangle}{\delta x_i} + \frac{\delta \langle p \rangle}{\delta x_j} + \nu \frac{\delta^2 \langle u_i \rangle}{\delta x_k^2}. \quad (2)$$

More detailed explanations of the URANS approach are given, e.g. in [28]. Star-CCM+ commercial code was used to run the CFD simulations. The CFD code employs finite volume methods to convert the continuous systems of equations into a set of discrete algebraic equations. The following geometries are considered: circle, ellipse, square, rectangle, triangle and equilateral triangle. Table 1 contains the descriptions of the geometries where the sketch indicates the orientation with angle  $0^\circ$ .

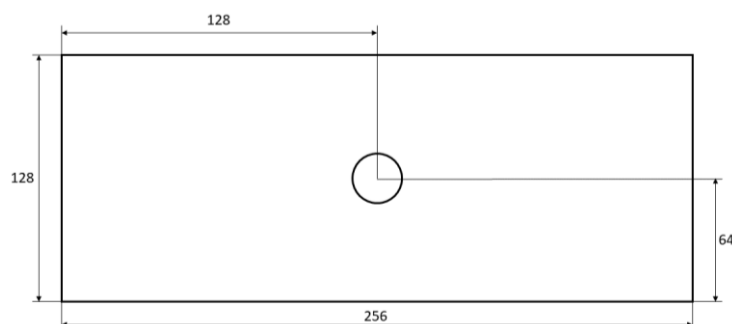
Table 1. Geometric data.

Shape	Sketch	Orientation	Scale
Circle		-	$a = 0.02 \text{ m}$
Ellipse		$60^\circ$	$a=0.02 \text{ m};$ $b=0.04 \text{ m}$
Square		$21^\circ$	$a=0.02 \text{ m}$
Rectangle		$114^\circ$	$a=0.02 \text{ m};$ $b=0.04 \text{ m};$
Triangle		$276^\circ$	$a=0.02 \text{ m}$ $b=0.035 \text{ m}$ $\gamma=80^\circ$
Equilateral triangle		$105^\circ$	$a=0.02 \text{ m}$

Overall, 500 samples of each geometry are simulated, giving a total of 3000 samples. For each sample, the vertical and streamwise velocities and the pressure fields are computed. Each simulation lasted 0.3 seconds with a time-step of  $2 \cdot 10^{-4}$  s. Data was collected after each time step, once the flow was fully developed, at  $t=0.2$  s. The time intervals were chosen to be small enough to capture the vortex shedding. The robustness of the solution is ensured with an upwind scheme [29,30] which discretizes the convective terms. The RANS-based  $k-\omega$  shear stress transport (SST) turbulence model introduced by Menter [31] was used to model the turbulence. The studies of Rajani et al. [32] and Rahman et al. [33] have successfully computed unsteady state simulations applied to similar cases. All the simulations were converged until a satisfactory residual convergence was achieved for the velocity and pressure quantities.

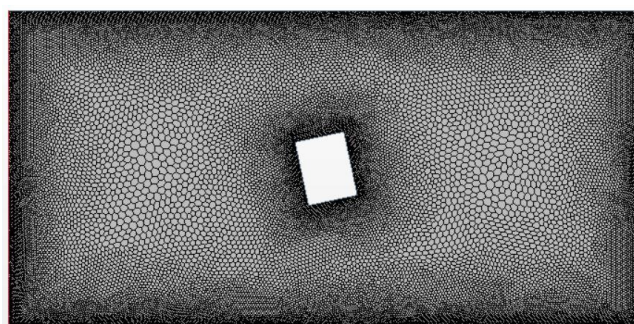
The numerical domain is composed of a rectangular two-dimensional computational domain with

the shape located in the center of the domain; see Aramendia et al. [34]. The left and right sides of the domain were set as the inlet and outlet, respectively. Top and bottom sides were set as no-slip walls, as well as the shape edge. Figure 1 shows a detailed view of the computational domain and its dimensions.



**Figure 1.** Numerical domain (not to scale).

The precision of, the convergence of and the time required to attain the solution of a CFD analysis strongly depend on the design and construction of a high-quality grid. Within this domain, a mesh of two-dimensional polyhedral cells was generated. The cell density is greater around the shape and on the domain walls. In addition, a volumetric control was designed to refine the mesh around the body to maintain the  $y^+$  value below 1. Figure 2 illustrates the mesh distribution around a square shape.



**Figure 2.** Example of the mesh distribution around the square.

With regards to the fluid, incompressible turbulent unsteady air is considered. The density ( $\rho$ ) of the selected fluid is equal to  $1.18415 \text{ kg/m}^3$ , and its dynamic viscosity ( $\mu$ ) is equal to  $1.85508 \cdot 10^{-5} \text{ Pa}\cdot\text{s}$ . Both magnitudes are assumed to be constant. The velocity at the inlet ( $u$ ) is  $5 \text{ m/s}$ . Consequently, the CFD numerical simulations have been run within a range of  $Re$  of  $6380$ - $12760$ . The CFD simulation data was interpolated into a  $79 \times 172$  grid to be manageable by the CNN [35].

In order to verify sufficient mesh resolution, the Richardson extrapolation method [36] was applied to the mean drag coefficient for the case of the circle, with  $Re = 6,380$ . This method consists of estimating the value of a parameter when the mesh size tends to zero (in other words, the mesh quantity tends to infinity) from a minimum of three meshes. Therefore, a coarse mesh (around 16,000 cells) a medium mesh (around 25,000 cells) and a fine mesh (around 44,000 cells) were designed. As summarized in Table 2, the convergence condition ( $R$ ), which should be between 0 and 1 to ensure

monotonic convergence, is fulfilled. Additionally, the drag coefficient obtained with the fine mesh is close to the estimated value. Moreover, the results have been compared with the experimental data reported by Roshko et al. [37] for the same case, showing fairly accurate results.

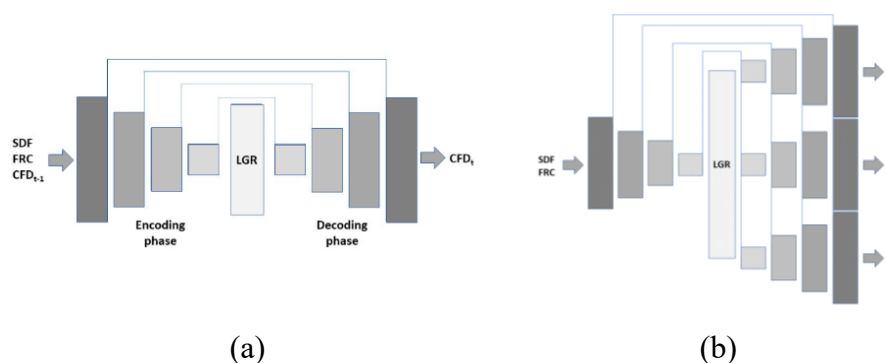
**Table 2.** Mesh verification and comparison with experimental data for the case of a cylinder.

Mesh Resolution			Richardson Extrapolation			Experimental
Coarse	Medium	Fine	RE	p	R	
0.796	0.835	0.858	0.907	0.681	0.566	0.91

## 2.2. Convolutional neural network architecture

CNNs are a type of neural network proposed by LeCun et al. [38] which are extremely efficient in the identification of patterns in a group of images on the pixel level. The digital images are, in essence, matrices. Therefore, the vertical and downstream velocities and pressure fields are also matrices, which enables the CNN architecture to identify the patterns.

Ronneberger et al. [39] proposed the U-Net architecture for medical images segmentation. Ribeiro et al. [8] demonstrated the adaptability of the U-Net to accurately predict the coupled velocity and pressure fields, providing the three solutions all at once. The CNN architecture consists of an encoder network that compresses the geometric input data to obtain a condensed version of the data, which facilitates the CNN in detecting the relevant patterns. This reduced version of the geometrical information is called latent geometry representation (LGR). Subsequently, the LGR is mapped into the data space, through a decoder network. The decoder network can take one, two or three decoders, depending on the number of variables studied. Figure 3 shows two simplified diagrams of the network architecture for one and three decoders.



**Figure 3.** Simplified diagrams of the U-Net. a) 1 decoder and b) 3 decoders.

The encoder-decoder network architecture is now described in detail. The encoder part is constituted of 4 or 5 decoder blocks, depending on the case, which, in turn, contain 3 convolution layers. The convolution layer parameters are the following:

- Filters: 8, 16, 32 and 32 or 8, 16, 16, 32 and 32 for each encoder block, respectively.
- Kernel size: 3, 5 or 7, depending on the model.

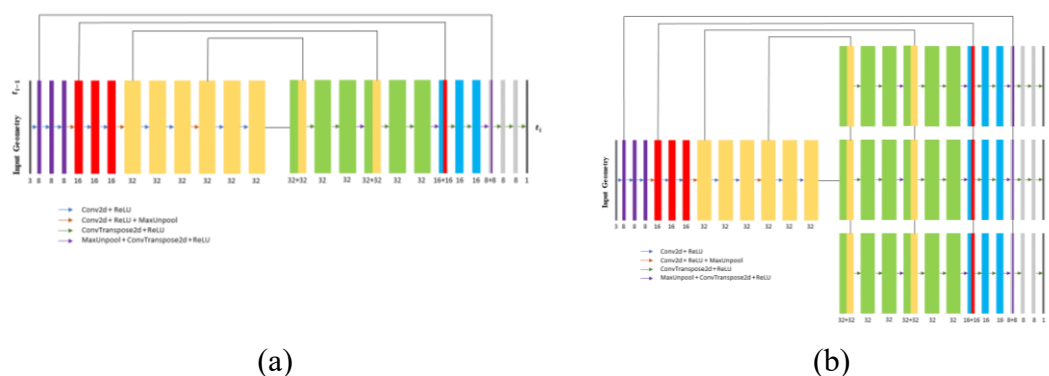
Each encoder block is formed by the following layers:

- A convolution layer with an equal number of filters at its input as the output of the previous block, and the output with the corresponding number of filters plus a ReLU layer. In the first encoder block, the number of filters is equal to the number of inputs of the CNN.
- A convolution layer with an equal number of filters in its input and output and a ReLU layer.
- Lastly, the same process as the previous point plus a max-pooling layer.

Each decoder block is formed by the following layers:

- A deconvolution layer with the double number of filters compared to the corresponding, and an output equal to the corresponding number of filters plus a ReLU layer.
- A deconvolution layer with the input and output equal to the number of filters plus a ReLU layer.
- A max-pooling layer before the deconvolution layer, with the corresponding number of filters for input and output, and a ReLU layer. In the last decoder block, the max-pooling is not considered, and the deconvolution takes a unique output, which corresponds to the evaluated variable.

In this study, two variants of CNN are analyzed. In the first one, the future instants of the downstream and vertical velocities and pressure fields are predicted. In fluid dynamics, the state of a fluid at time instant  $t$  is strongly dependent on its previous state,  $t-1$ . The fluid transition between the states is of great importance. In terms of a neural network, this dependency is implemented with an input to the net, representing the  $t-1$  state. Due to the addition of the previous instant of the CFD sample to the inputs of the CNN, each variable must be analyzed separately. The simultaneous analysis of the three variables would be erroneous, due to mixing information between the variables. The second variant focuses on the prediction of the first sample, needed for the prediction of the following states. In this case, three decoders are implemented, and the CNN inputs regard solely the geometric information. Figure 4 represents the two architectures analyzed with an [8, 16, 32, 32] filter configuration.



**Figure 4.** Diagrams of the complete architecture of the CNN for an 8, 16, 32, 32 filter configuration. a) 1 decoder and b) 3 decoders.

The pressure field can be determined by solving the Poisson equation, see Eq (3). Nevertheless, since the prediction of the velocity through a neural network implies the appearance of an error, if the pressure is calculated with Eq (3), the error in the pressure is carried from the predictions of the velocities made by the neural model.

$$\Delta p = -\rho \nabla \cdot (v \cdot \nabla v) = -\rho \text{Tr}((\nabla v)(\nabla v)). \quad (3)$$

Physics-informed neural networks (PINNs) compose a type of neural networks that provide the prediction of the solution of a nonlinear partial differential equation (PDE) [40–43]. The loss function of these neural networks is formed by the residual of the PDE. The methodology used in this work could be converted to a PINN methodology by employing Eq (2) as the PDE needed for PINN.

Nowadays, ChatGPT is a popular tool generating code. The following lines contain a ChatGPT prompt that describes the network architecture in order to generate the Python code of the neural network [44,45]: Develop a Python code for the prediction of the velocity and pressure fields around the following geometries in two dimensions: circle, ellipse, square, rectangle, triangle and equilateral triangle. Specifically, use convolutional neural networks to encode geometric features and then use deconvolutional neural networks to decode the outputs. Input is 79 by 172 images, and the output has the same size.

### 2.3. Convolutional neural network inputs

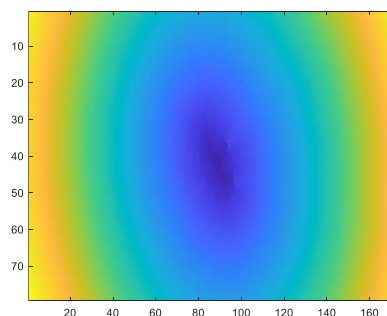
Following the studies of Guo et al. [7], the signed distance function was considered as an input to the CNN. The SDF is a function that measures the distance between any point in the grid and the nearest boundary of a closed geometry shape. The sign of the value depends on whether the point is inside (negative) or outside (positive) the closed geometry. This function provides smaller values than the typical binary representation. The mathematical expression of this function is given by Eq (4).

$$SDF(x) = \begin{cases} d(x, \vartheta\Omega) & \text{if } x \in \Omega \\ -d(x, \vartheta\Omega) & \text{if } x \in \Omega^c \end{cases} \quad (4)$$

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

$$d(x, \vartheta\Omega) := \inf_{y \in \vartheta\Omega} d(x, y), \quad (5)$$

where  $\inf$  denotes the infimum. Grid positions inside the interior of the geometry are assigned negative distances. Figure 5 shows an example of the SDF of an ellipse.

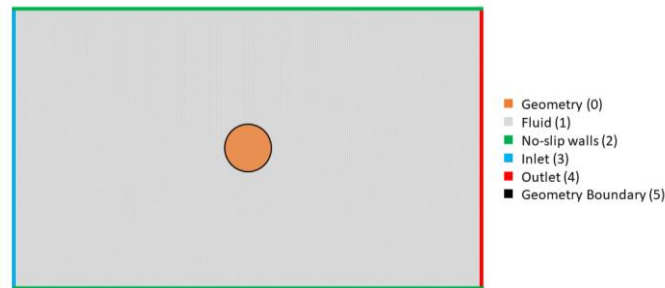


**Figure 5.** SDF of an ellipse.

The flow region channel (FRC) represents a multi-class channel that contains information about



the boundary conditions of the domain. The information is organized into 5 categories: 0 for the geometry, 1 for the free flow region, 2 for the no-slip walls, 3 for the inlet and 4 for the outlet. Figure 6 shows a schematic view of the FRC.



**Figure 6.** Diagram of an FRC (not to scale).

The SDF and FRC represent the inputs of the second variant of the CNN. For the first variant, which predicts the future states of the fluid, a third input is added. The CFD simulations represent the third input. In particular, the previous instant of the current CFD-analyzed sample is selected as the input. The testing of the CNN is initialized with a random sample of the CFD simulations, and in the next predictions, the previous prediction is the sample used as the current reference for the new prediction.

#### 2.4. Training parameters

AdamW is the optimizer selected for the training of CNN. It is based on the Adam algorithm, which updates the gradient vector and the squared gradient using an exponential moving average [46].  $\beta_1$  and  $\beta_2$  represent the forgetting factors for the gradients and second moments of the gradients, respectively, and their values were both set to 0.5. AdamW improves regularization by decoupling the weight decay from the gradient-based update [47]. For the hyper-parameter search, the CNN is trained for every possible combination of the values given by Tables 3 and 4 for the net which predicts the future states and the net which predicts the initial state, respectively.

**Table 3.** Set of parameters considered for the hyper-parameter search for the net which predicts the future states.

Parameter	Values		
Filters	8, 16, 32, 32		
Kernel size	3	5	7
Loss function	L1-norm		
Learning rate	0.001	0.0001	
Weight decay	0.005		
Batch size	32	64	128
Training-test ratio	0.7-0.3		
Number of epochs	2000		

**Table 4.** Set of parameters considered for the hyper-parameter search for the net which predicts the initial state.

Parameter	Values			
Filters	8, 16, 32, 32		8, 16, 16, 32, 32	
Kernel size	3	5	7	9
Loss function	L1-norm			
Learning rate	0.001		0.0001	
Weight decay	0.005			
Batch size	32	64		128
Training-test ratio	0.7-0.3			
Number of epochs	1000		2000	

### 2.5. Data augmentation

The training of the CNN which predicts the initial state of the streamwise and vertical velocities and the pressure fields needs many data samples to obtain quality results. To increment the amount of data, a data augmentation technique is applied.

Data augmentation is a well-known technique applied in DL applications. This technique consists of generating realistic synthetic data to increase the data quantity in the learning process. Typical data augmentation applies geometric transformations and perturbations to the original data. Nevertheless, this procedure cannot be implemented in this study. To solve this drawback, a data augmentation technique based on the similarity theory for fluid dynamics is applied [18]. The Reynolds number is calculated by Eq (6).

$$Re = \frac{u_{\infty} D \rho}{\mu}. \quad (6)$$

Assuming that the Reynolds number is kept constant in each case, the new input velocity can be calculated with Eq (7). The fluid and boundary conditions remain constant, and the density and dynamic viscosity have no influence on the velocity; therefore, with slight modifications in the shape size, the amount of input data considerably increases.

$$u_{\infty i}^* = \frac{D_1}{D_i} u_{\infty 1}. \quad (7)$$

The values of the novel fields are given by Eqs 8–10.

$$\hat{u}_{xi}(\hat{x}, \hat{y}) = \frac{u_{xi}}{u_{\infty i}^*} \left( \frac{x_i}{D_i}, \frac{y_i}{D_i} \right), \quad (8)$$

$$\hat{u}_{yi}(\hat{x}, \hat{y}) = \frac{u_{yi}}{u_{\infty i}^*} \left( \frac{x_i}{D_i}, \frac{y_i}{D_i} \right), \quad (9)$$

$$\hat{p}_i(\hat{x}, \hat{y}) = \frac{p_{xi}}{u_{\infty i}^{*2} \rho} \left( \frac{x_i}{D_i}, \frac{y_i}{D_i} \right), \quad (10)$$

where  $\hat{u}_{xi}$ ,  $\hat{u}_{yi}$  and  $\hat{p}_i$  represent the new coupled velocities and the pressure fields, and  $x_i/D_i$  and  $y_i/D_i$  represent the new coordinates inside the domain. As the similarity theory establishes, the domain size changes proportionately with shape size. This is shown by Eq (11), where the equivalence between two concrete points of the grid is provided.

$$\hat{x} = \frac{x_1}{D_1} = \frac{x_2}{D_2}. \quad (11)$$

### 3. Results

#### 3.1. CNN that predicts the future states

Table 3 indicates the values of the hyper-parameters selected for the training of the neural models. In Tables 5–7 the best neural models ordered by minimum mean error are provided. Since an independent model is generated for each of the three variables, different combinations of the parameters can be selected. Every training was conducted with 2000 epochs and an [8, 16, 32] filter configuration. The criterion followed to pick the adequate model is based on the mean and maximum error given by the tests of each neural model. The training and test sets were the same in every trained neural model. In the cases of the vertical velocity and the pressure, the minimum mean and maximum error correspond to the same neural model. However, for the streamwise velocity, the two former models with the least mean error provide excessively large maximum errors. The results of the third and fourth models are relatively similar, each one providing less mean error and bigger maximum error, and vice versa. The fourth model is the chosen one, due to having the least maximum error. The selection is based on the influence of the large errors in the last samples predicted. Hence, the IDs of the chosen models are 15, 5 and 12 for the streamwise and vertical velocities and pressure, respectively.

**Table 5.** 10 best models for the prediction of the future states of the streamwise velocity field.

ID	Kernel size	Lr	Batch size	Training duration (h)	Mean error $v_x$ (m/s)	Maximum error $v_x$ (m/s)
10	3	0.0001	32	1.70	0.1413	33.4453
13	3	0.0001	64	1.35	0.1561	43.7074
12	7	0.0001	32	2.83	0.1715	15.6595
15	7	0.0001	64	2.60	0.2053	14.8938
4	3	0.001	64	1.37	0.2188	2705.84
11	5	0.0001	32	1.67	0.2217	21.416
3	7	0.001	32	2.74	0.2510	73.7285
14	5	0.0001	64	1.51	0.2847	48.7618
6	7	0.001	64	2.59	0.3729	1848.65
5	5	0.001	64	1.50	0.5728	39398.8

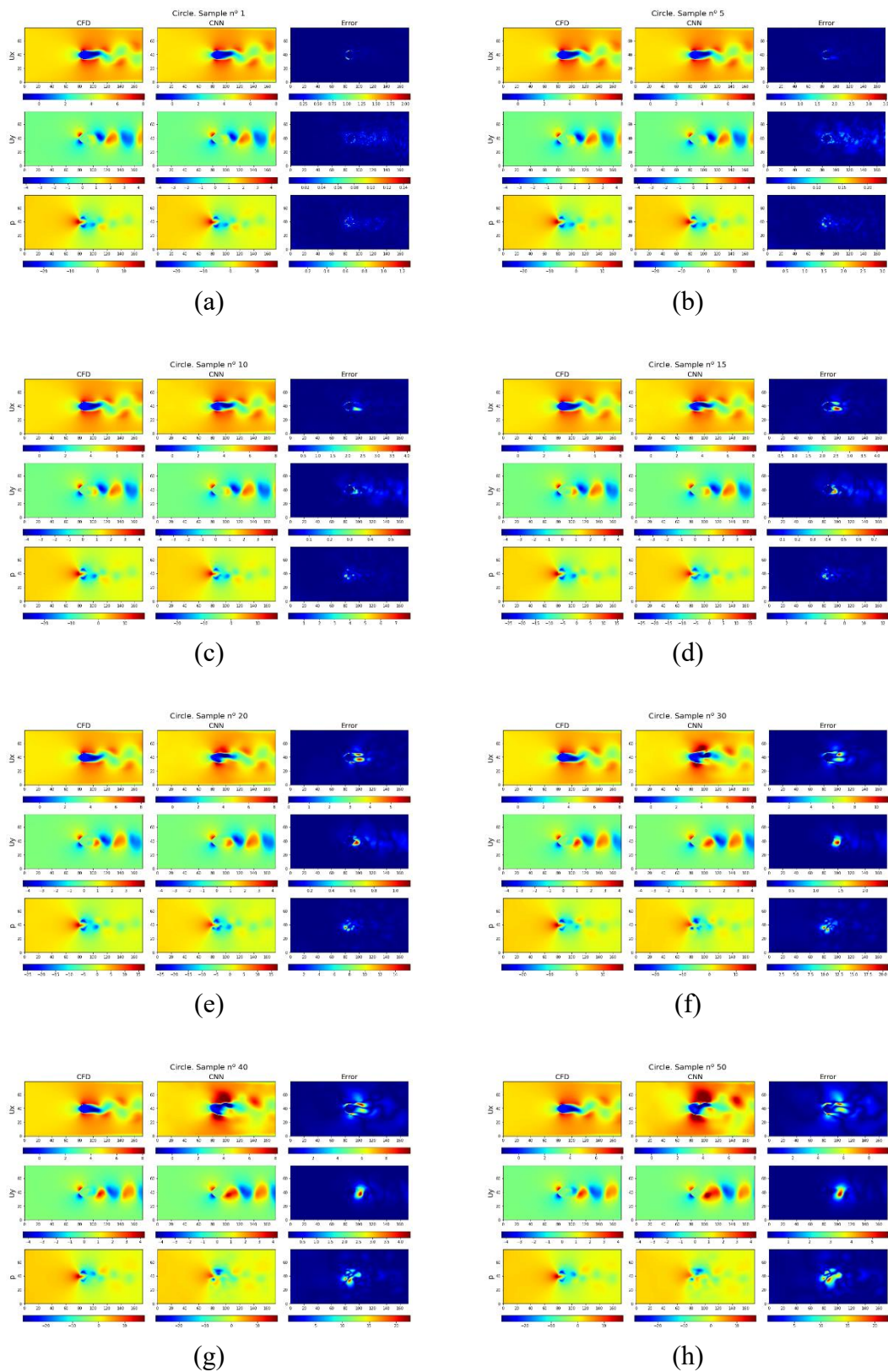
**Table 6.** 10 best models for the prediction of the future states of the vertical velocity field.

ID	Kernel size	Lr	Batch size	Training duration (h)	Mean error $v_y$ (m/s)	Maximum error $v_y$ (m/s)
5	5	0.001	64	1.45	0.0585	9.7446
15	7	0.0001	64	2.55	0.0651	10.5586
12	7	0.0001	32	2.72	0.0651	32.4277
2	5	0.001	32	1.56	0.0674	21.7514
14	5	0.0001	64	1.46	0.0976	18.6669
7	3	0.001	128	1.40	0.0992	35.5557
9	7	0.001	128	2.69	0.0999	15.4869
1	3	0.001	32	1.59	0.1016	802.375
8	5	0.001	128	1.49	0.1094	90.4429
16	3	0.0001	128	1.41	0.1358	34.6491

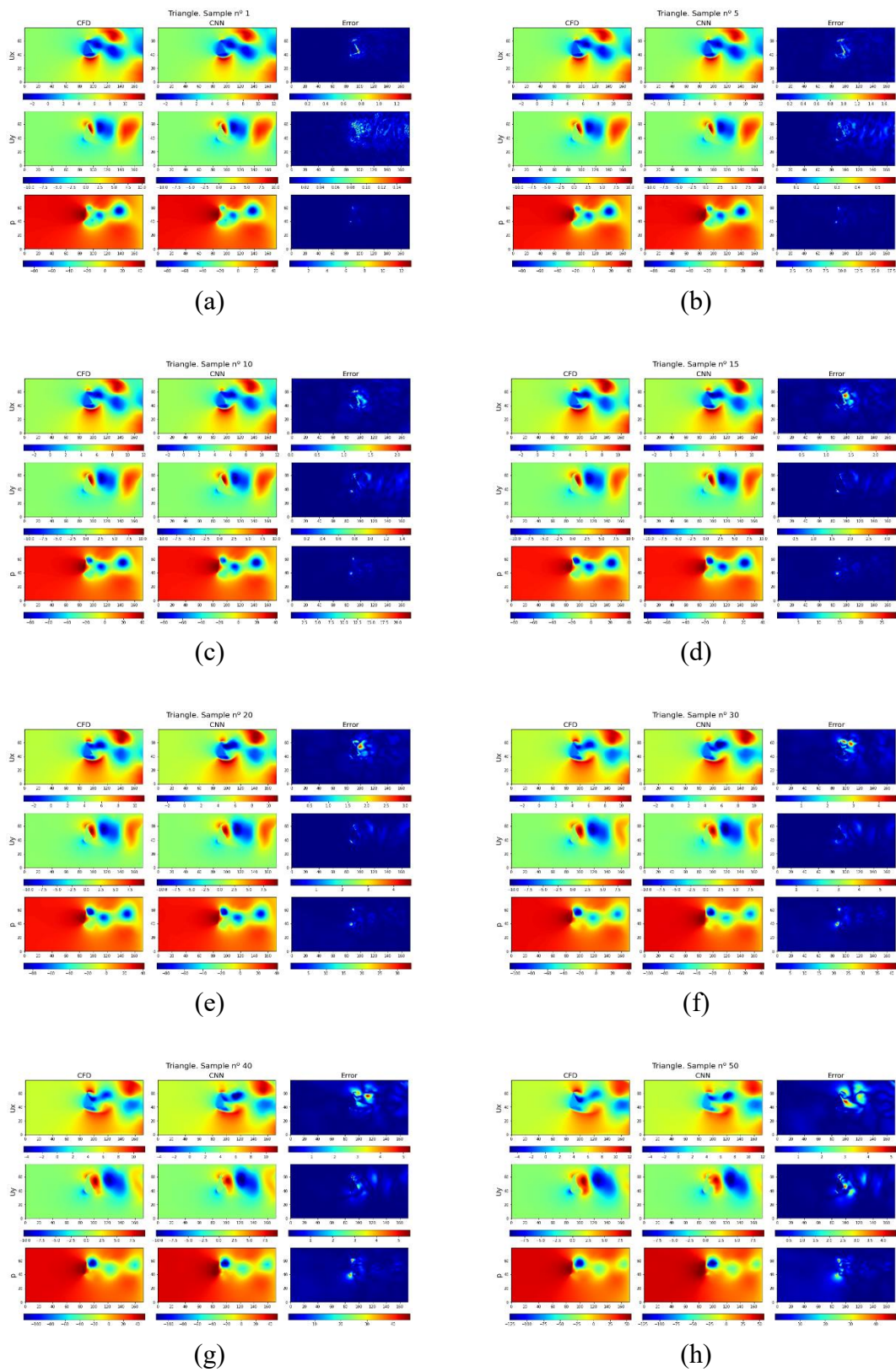
**Table 7.** 10 best models for the prediction of the future states of the pressure field.

ID	Kernel size	Lr	Batch size	Training duration (h)	Mean error p (Pa)	Maximum error p (Pa)
12	7	0.0001	32	2.72	0.7139	112.09
4	3	0.001	64	1.24	0.8828	132.73
11	5	0.0001	32	1.57	0.9542	119.92
1	3	0.001	32	1.58	0.9748	466.44
14	5	0.0001	64	1.46	0.9933	117.70
5	5	0.001	64	1.44	1.0869	356.00
15	7	0.0001	64	2.54	1.0900	212.32
7	3	0.001	128	1.40	1.1218	549.21
18	7	0.0001	128	2.71	1.1887	108.50
2	5	0.001	32	1.55	1.3195	196.38

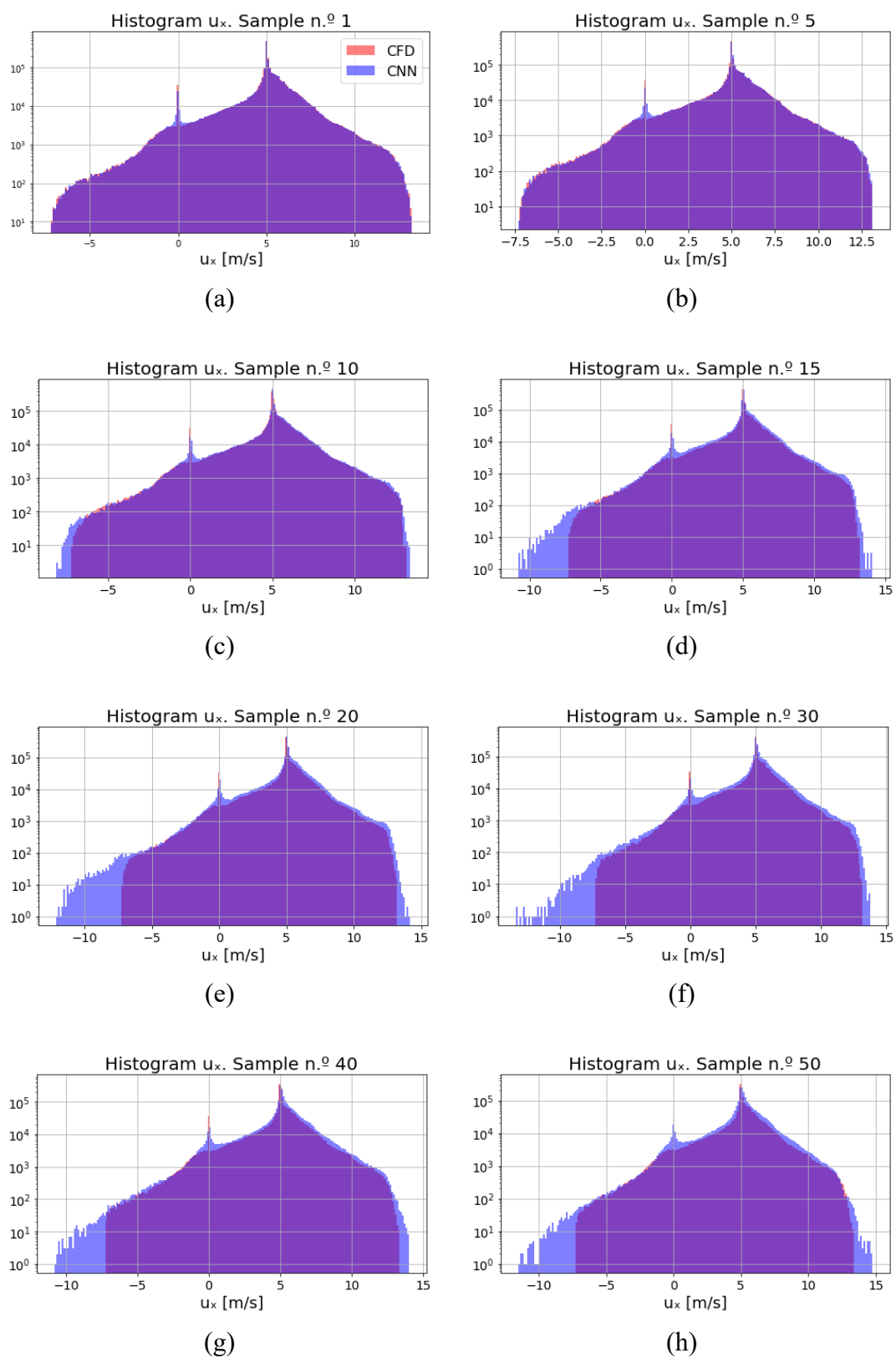
Figure 7 and Figure 8 show graphically the comparison of results obtained by the CNN and the CFD ground-truth data. Table 8 contains the quantitative values of the arithmetic mean and the standard deviation of the CFD simulations and the CNN tests. The biggest variations appear in the wake due to being the zone where the turbulences and velocity gradients are more notorious. Conversely, in the contours of the geometries, the effect of the boundary layer and its separation lead to the appearance of errors. The deviations between the CFD results and the CNN predictions appear in the aforementioned zones when the predictions advance through time. Moreover, Figure 9, Figure 10 and Figure 11 show the histograms for the three variables, where the values given by the CFD are compared with the CNN test ones. The absolute error in the predictions of the three fields raises over the samples predicted, due to the dependency on the initial state. The appearance of an excessively high error in a certain point of the mesh has a direct effect on the precision of the next predictions on that point. In the first twenty samples predicted, the absolute error is relatively small; however, in the rear zone of the geometry contour, high absolute errors appear, which are accumulated during the predictions. The vortices are well predicted, even in the most advanced samples.



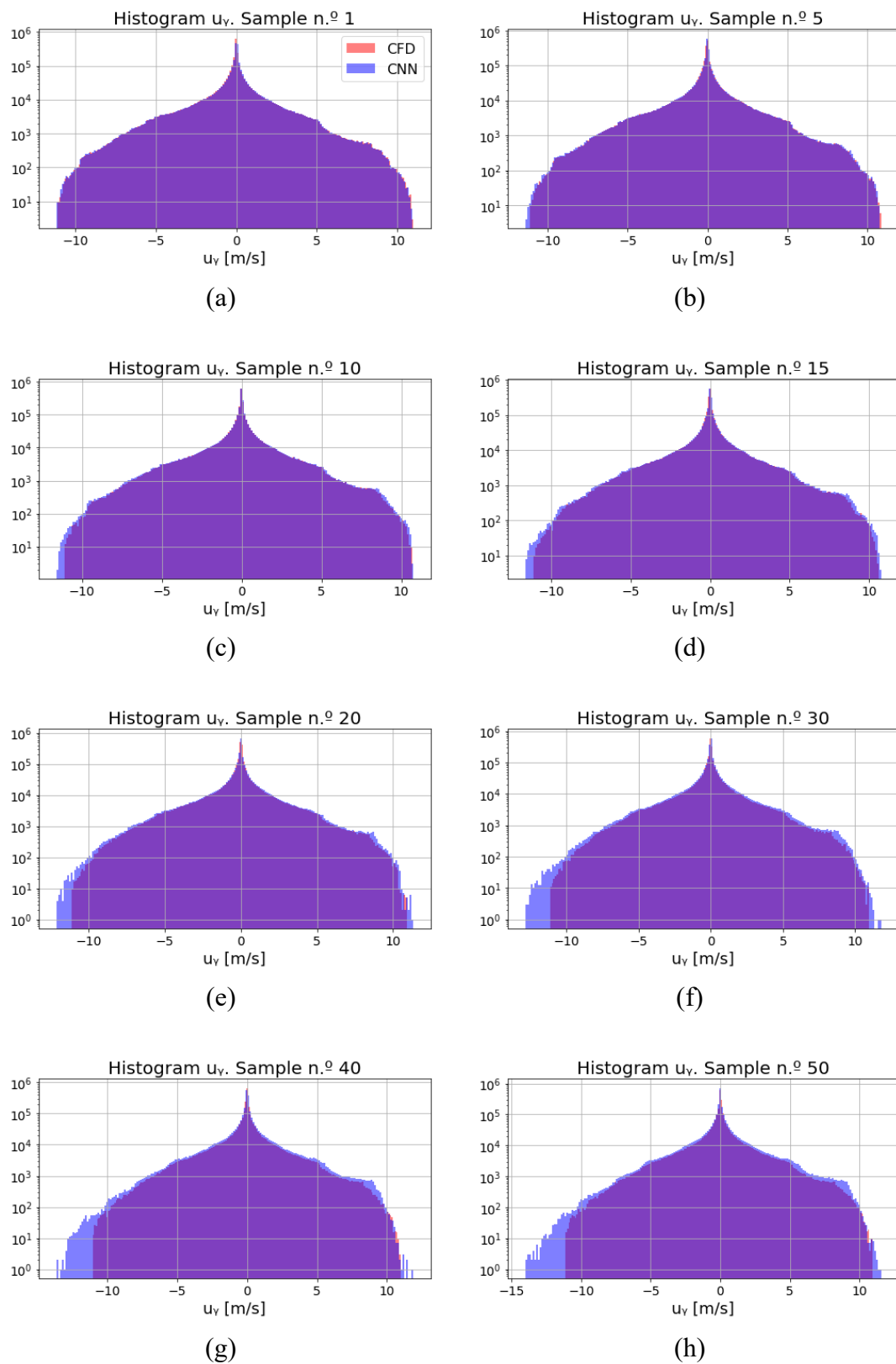
**Figure 7.** Predictions for the circular shaped geometry for the sample number: a) 1, b) 5, c) 10, d) 15, e) 20, f) 30, g) 40 and h) 50.



**Figure 8.** Predictions for the triangle shaped geometry for the sample number: a) 1, b) 5, c) 10, d) 15, e) 20, f) 30, g) 40 and h) 50.

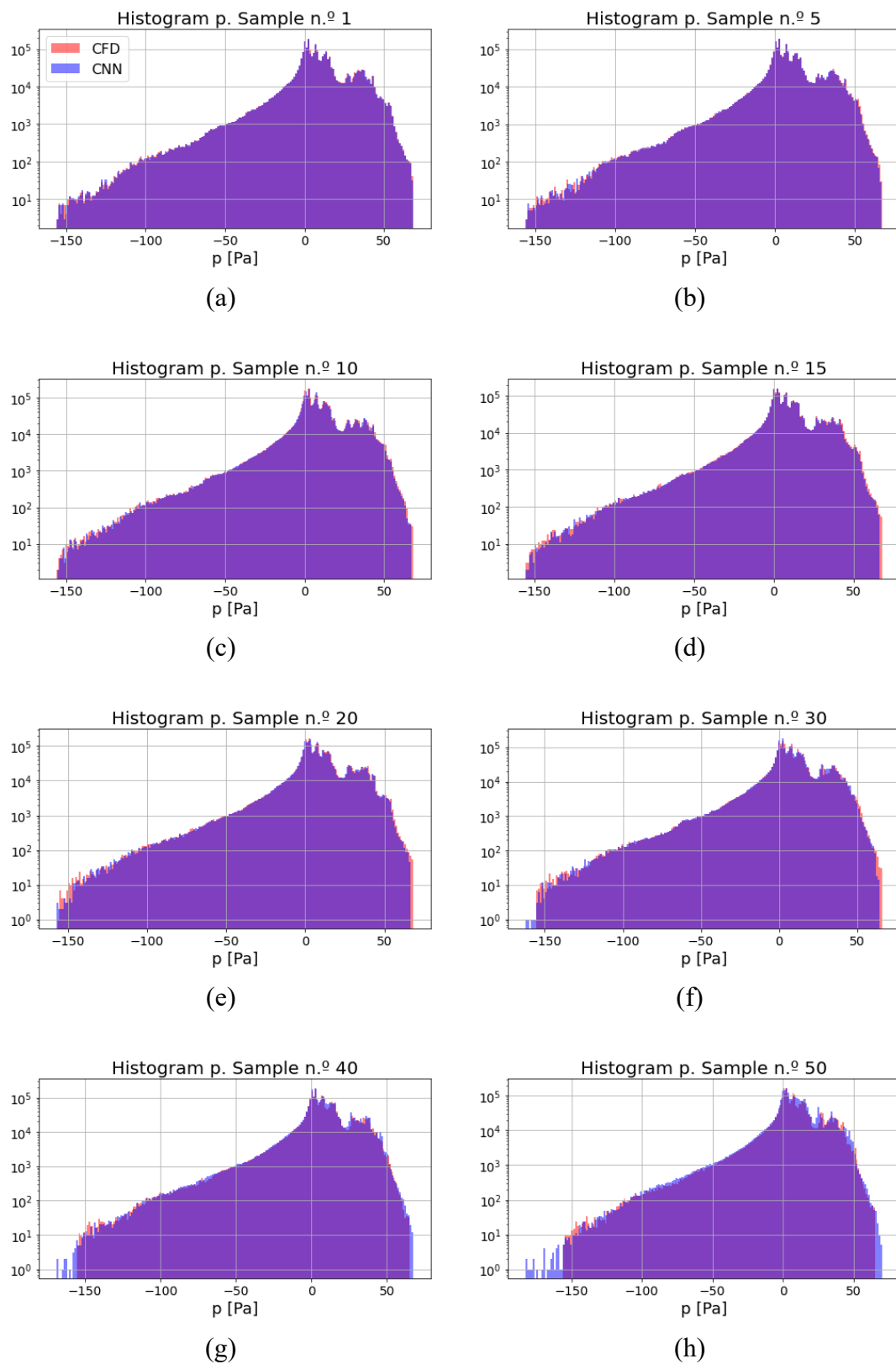


**Figure 9.** Data distribution of the CFD simulations and CNN tests for the streamwise velocity field for the sample number: a) 1, b) 5, c) 10, d) 15, e) 20, f) 30, g) 40 and h) 50.



**Figure 10.** Data distribution of the CFD simulations and CNN tests for the vertical velocity field for the sample number: a) 1, b) 5, c) 10, d) 15, e) 20, f) 30, g) 40 and h) 50.





**Figure 11.** Data distribution of the CFD simulations and CNN tests for the pressure field for the sample number: a) 1, b) 5, c) 10, d) 15, e) 20, f) 30, g) 40 and h) 50.

**Table 8.** Arithmetic mean and standard deviation obtained by the CFD simulations and CNN tests.

Method	CFD			CNN		
	$u_x$ (m/s)	$u_y$ (m/s)	$p$ (Pa)	$u_x$ (m/s)	$u_y$ (m/s)	$p$ (Pa)
Arithmetic mean ( $\mu$ )	5.0283	-0.0538	8.4882	5.0413	-0.0452	8.5360
Standard deviation ( $\sigma$ )	1.9109	1.7205	17.0865	1.9940	1.7725	16.9417

### 3.2. CNN that predicts the initial state

The CNN was trained for every combination of the hyper-parameters provided by Table 4, and Tables 9–11 show the 10 best-trained neural models for the streamwise and vertical velocities and the pressure, respectively. Analyzing the mean and maximum errors in each case, the following combination provides the best results:

- Filters combination: 8, 16, 16, 32 and 32
- Kernel size: 3
- Lr: 0.001
- Batch size: 32
- Number of epochs: 2000

This combination is selected due to being the best model for the streamwise velocity and the pressure and the second best for the vertical velocity. Every model has been run with the same training and test sets.

Figure 12 shows a prediction with the selected neural model for every analyzed geometry. The accuracy of the results is evaluated quantitatively in Table 12, where the arithmetic mean and standard deviation of the CFD simulations and CNN tests are compared. Moreover, the value distribution obtained by both methods are compared by the histograms of Figure 13.

### 3.3. Computational cost analysis

In this section, the computational time is required to calculate the streamwise and vertical velocities and the pressure fields with the CFD software and with the predictions of the neural models. The training duration of the models for the prediction of the future states is 2.60, 1.45 and 2.72 hours for the streamwise and vertical velocities and the pressure, respectively. This gives a total of 6.77 hours of training. The prediction of 50 instants lasts 1.76, 3.77 and 6.06 seconds, respectively. For the neural model for the prediction of the initial sample, the training took 19.66 minutes, and the predictions take a mean of 0.08 seconds.

Table 13 shows the comparison between the duration of the predictions of the neural models and the CFD simulations. In the case of the predictions of the future samples, the time needed to obtain 50 samples of the three variables is compared. For the initial sample, the time needed for a unique prediction is compared. Both training and testing were carried out using an NVIDIA Quad RTX 6000 GPU and an Intel Xeon Gold 5120 CPU was used for the CFD simulations.

**Table 9.** 10 best models for the prediction of the initial state of the streamwise velocity field.

Filters combination	Kernel size	Lr	Batch size	N° epochs	Training duration (min)	Mean error $v_x$ (m/s)	Maximum error $v_x$ (m/s)
8, 16, 16, 32, 32	3	0.001	32	2000	19.66	0.0528	2.4436
8, 16, 32, 32	5	0.001	32	2000	22.33	0.0550	2.5234
8, 16, 16, 32, 32	5	0.001	32	2000	21.60	0.0678	3.0944
8, 16, 32, 32	3	0.001	32	2000	19.45	0.0698	3.2475
8, 16, 32, 32	7	0.001	32	1000	13.84	0.0705	2.5926
8, 16, 32, 32	9	0.0001	32	2000	37.20	0.0707	2.9895
8, 16, 16, 32, 32	5	0.001	32	1000	10.88	0.0746	2.9101
8, 16, 32, 32	7	0.0001	32	2000	27.85	0.0793	4.3594
8, 16, 32, 32	3	0.001	128	2000	17.02	0.0811	4.0057
8, 16, 16, 32, 32	9	0.0001	32	2000	34.19	0.0831	3.7460

**Table 10.** 10 best models for the prediction of the initial state of the vertical velocity field.

Filters combination	Kernel size	Lr	Batch size	N° epochs	Training duration (min)	Mean error $v_x$ (m/s)	Maximum error $v_x$ (m/s)
8, 16, 32, 32	7	0.0001	32	2000	27.85	0.0281	2.2092
8, 16, 16, 32, 32	3	0.001	32	2000	19.66	0.0288	2.6517
8, 16, 32, 32	5	0.001	32	2000	22.33	0.0292	2.4026
8, 16, 32, 32	7	0.001	32	1000	13.84	0.0310	2.4049
8, 16, 32, 32	3	0.001	32	2000	19.45	0.0333	3.9388
8, 16, 32, 32	5	0.001	64	2000	20.31	0.0343	2.6450
8, 16, 16, 32, 32	5	0.001	32	2000	21.60	0.0353	2.0954
8, 16, 32, 32	9	0.0001	32	2000	37.20	0.0365	2.2067
8, 16, 32, 32	5	0.001	32	1000	11.19	0.0371	1.8295
8, 16, 16, 32, 32	7	0.001	32	1000	12.88	0.0397	4.2255

**Table 11.** 10 best models for the prediction of the initial state of the pressure field.

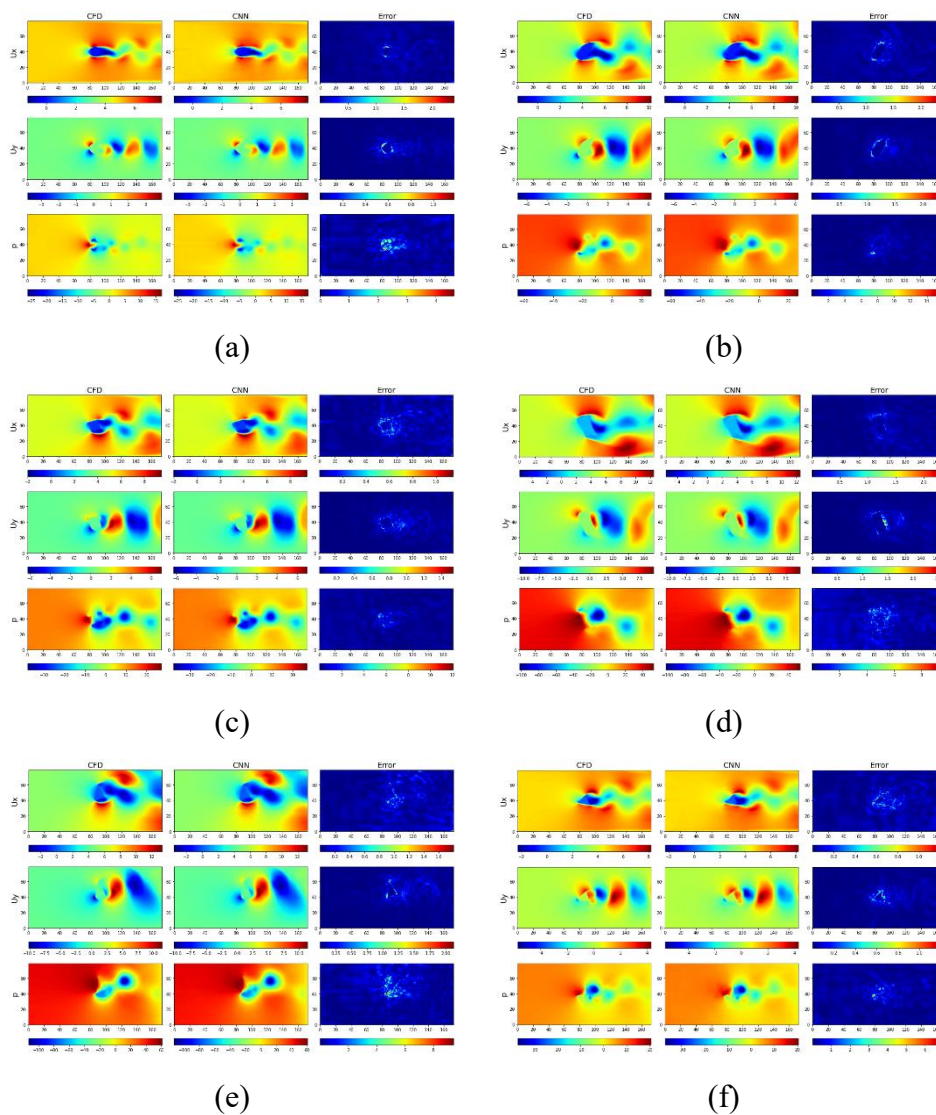
Filters combination	Kernel size	Lr	Batch size	N° epochs	Training duration (min)	Mean error $v_x$ (m/s)	Maximum error $v_x$ (m/s)
8, 16, 16, 32, 32	3	0.0001	32	2000	19.66	0.2474	16.1953
8, 16, 32, 32	7	0.001	32	1000	13.84	0.2604	18.4768
8, 16, 16, 32, 32	5	0.001	32	2000	22.33	0.2643	16.4145
8, 16, 16, 32, 32	9	0.001	64	2000	32.99	0.2736	19.4038
8, 16, 16, 32, 32	5	0.001	32	2000	21.60	0.2770	22.1191
8, 16, 32, 32	5	0.001	64	2000	20.31	0.2814	17.5238
8, 16, 32, 32	3	0.001	32	2000	19.45	0.3048	17.0342
8, 16, 32, 32	7	0.0001	32	2000	27.85	0.3080	23.0767
8, 16, 16, 32, 32	9	0.0001	32	2000	34.19	0.3269	27.6613
8, 16, 32, 32	5	0.001	32	1000	11.19	0.3314	19.8795

**Table 12.** Arithmetic mean and standard deviation obtained by the CFD simulations and CNN tests.

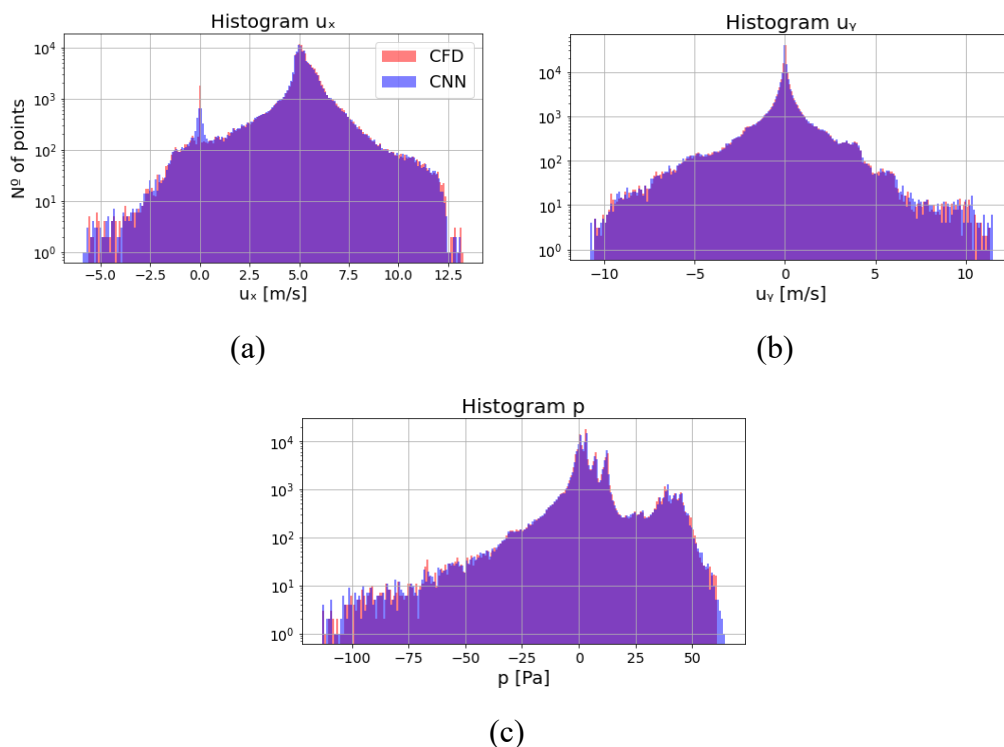
Method	CFD			CNN		
	$u_x$ (m/s)	$u_y$ (m/s)	p (Pa)	$u_x$ (m/s)	$u_y$ (m/s)	p (Pa)
Arithmetic mean ( $\mu$ )	5.0674	-0.0644	5.9574	5.0414	-0.0628	6.0475
Standard deviation ( $\sigma$ )	1.6867	1.5074	15.0737	1.6797	1.5142	15.0853

**Table 13.** Comparison between the calculation time required by the CFD simulations and the predictions of the neural models for the streamwise and vertical velocities and the pressure.

Net	CFD time (s)	Prediction time (s)	Speedup
Future states of the fields	720.00	11.59	62.12
Initial state	720.00	0.08	9000.00



**Figure 12.** Predictions of the initial state for the a) circle, b) cylinder, c) square, d) rectangle, e) triangle and f) equilateral triangle.



**Figure 13.** Data distribution of the CFD simulations and CNN tests for the a) streamwise velocity, b) vertical velocity and c) pressure.

#### 4. Conclusions

The CFD simulations are of great utility for the study of turbulent fluids when they face concrete geometries. The computational costs of these simulations are frequently too high, and their precision depends also on the mesh and turbulence model generation. These issues imply the use of DL techniques to approximate the CFD results and reduce their computational costs. In the current work, a U-Net structure was applied to predict the streamwise and vertical velocities and the pressure fields downstream of a series of different geometries. A turbulent fluid was analyzed, and CFD unsteady simulations were conducted. The predictions were accomplished using a time-based approach, predicting the immediate future sample based on its dependency on its previous state. Until the twentieth sample, the predictions are relatively reliable. The absolute errors are higher in the streamwise velocity than in the vertical velocity and the pressure. The vortices are well predicted, giving the more inexact results in the back of the contour of the geometries. The data augmentation technique employed is efficient to increase the number of samples required by the training, avoiding running extra CFD simulations for each variation in the size of the geometries. Hence, the trained neural model accurately predicts the streamwise and vertical velocities and pressure fields. With respect to the reduction of the computational cost, the neural model which predicts the future states is 62.12 times faster than the CFD simulations, and the model for the initial sample is 9000 times faster thanks to the data augmentation applied. The limitation of the proposed method is based on the necessity to obtain the CFD data and the computational time and resources required by the training process. In addition, the current CNN

model has been tested in the five proposed geometries. Therefore, the model is considered generalizable for similar geometries and shapes. For future works, this U-Net structure can be applied to simulate and predict the fields of more aerodynamic shapes, such as airfoils or wings. The addition of historical inputs, such as multiple previous time steps, may be considered, too. Furthermore, to attain aerodynamic optimization, DL techniques can be applied to predict the best shape of an airfoil with an added gurney flap.

### Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

### Acknowledgments

The authors were supported by the government of the Basque Country through the research grant ELKARTEK KK-2021/00014 BASQNET (Estudio de nuevas técnicas de inteligencia artificial basadas en Deep Learning dirigidas a la optimización de procesos industriales) and IT1514-22. K. P.-P. was supported by the INVESTIGO program of the Basque Country 2022.

The authors are grateful for the support provided by the SGIker of UPV/EHU.

### Conflict of interest

Unai Fernandez-Gamiz and Ekaitz Zulueta are the Guest Editors of special issue "Artificial Intelligence for Fluid Mechanics and its Engineering Applications" for AIMS Mathematics. Unai Fernandez-Gamiz and Ekaitz Zulueta were not involved in the editorial review and the decision to publish this article.

All authors declare no conflicts of interest in this paper.

### References

1. S. L. Brunton, B. R. Noack, P. Koumoutsakos, Machine learning for fluid mechanics, *Annu. Rev. Fluid Mech.*, **52** (2020), 477–508. <https://doi.org/10.1146/annurev-fluid-010719-060214>
2. S. Qin, S. Wang, L. Wang, C. Wang, G. Sun, Y. Zhong, Multi-objective optimization of cascade blade profile based on reinforcement learning, *Appl. Sci.*, **11** (2021), 106. <https://doi.org/10.3390/app11010106>
3. Y. Qiu, J. Bai, N. Liu, C. Wang, Global aerodynamic design optimization based on data dimensionality reduction, *Chinese J. Aeronaut.*, **31** (2018), 643–659. <https://doi.org/10.1016/j.cja.2018.02.005>
4. B. N. Hanna, N. T. Dinh, R. W. Youngblood, I. A. Bolotnov, Coarse-grid computational fluid dynamic (CG-CFD) error prediction using machine learning, preprint paper, 2017. <https://doi.org/10.48550/arXiv.1710.09105>
5. H. Bao, J. Feng, N. Dinh, H. Zhang, Computationally efficient CFD prediction of bubbly flow using physics-guided deep learning, *Int. J. Multiphase Flow*, **131** (2020), 103378. <https://doi.org/10.1016/j.ijmultiphaseflow.2020.103378>

6. K. Tlales, K. E. Otmani, G. Ntoukas, G. Rubio, E. Ferrer, Machine learning adaptation for laminar and turbulent flows: applications to high order discontinuous Galerkin solvers, preprint paper, 2022. <https://doi.org/10.48550/arXiv.2209.02401>
7. X. Guo, W. Li, F. Iorio, Convolutional neural networks for steady flow approximation, In: *Proceedings of the Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2016, 481–490. <https://doi.org/10.1145/2939672.2939738>
8. M. D. Ribeiro, A. Rehman, S. Ahmed, A. Dengel, DeepCFD: Efficient steady-state laminar flow approximation with deep convolutional neural networks, preprint paper, 2020. <https://doi.org/10.48550/arXiv.2004.08826>
9. A. Kashefi, D. Rempe, L. J. Guibas, A point-cloud deep learning framework for prediction of fluid flow fields on irregular geometries, *Phys. Fluids*, **33** (2021), 027104. <https://doi.org/10.1063/5.0033376>
10. T. Murata, K. Fukami, K. Fukagata, Nonlinear mode decomposition with convolutional neural networks for fluid dynamics, *J. Fluid Mech.*, **882** (2020), A13. <https://doi.org/10.1017/jfm.2019.822>
11. J. Ling, A. Kurzawski, J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, *J. Fluid Mech.*, **807** (2016), 155–166. <https://doi.org/10.1017/jfm.2016.615>
12. S. Lee, D. You, Prediction of laminar vortex shedding over a cylinder using deep learning, preprint paper, 2017. <https://doi.org/10.48550/arXiv.1712.07854>
13. Y. Liu, Y. Lu, Y. Wang, D. Sun, L. Deng, F. Wang, et al., A CNN-based shock detection method in flow visualization, *Comput. Fluids*, **184** (2019), 1–9. <https://doi.org/10.1016/j.compfluid.2019.03.022>
14. L. Deng, Y. Wang, Y. Liu, F. Wang, S. Li, J. Liu, A CNN-based vortex identification method, *J. Vis.*, **22** (2019), 65–78, <https://doi.org/10.1007/s12650-018-0523-1>
15. H. Nowruzzi, H. Ghassemi, M. Ghiasi, Performance predicting of 2D and 3D submerged hydrofoils using CFD and ANNs, *J. Mar. Sci. Technol.*, **22** (2017), 710–733. <https://doi.org/10.1007/s00773-017-0443-0>
16. A. Mohan, D. Daniel, M. Chertkov, D. Livescu, Compressed convolutional LSTM: An efficient deep learning framework to model high fidelity 3D turbulence, preprint paper, 2019. <https://doi.org/10.48550/arXiv.1903.00033>
17. K. Portal-Porrás, U. Fernández-Gamiz, A. Ugarte-Anero, F. Zulueta, A. Zulueta, Alternative artificial neural network structures for turbulent flow velocity field prediction, *Mathematics*, **9** (2021), 1939. <https://doi.org/10.3390/math9161939>
18. A. Abucide-Armas, K. Portal-Porrás, U. Fernández-Gamiz, E. Zulueta, A. Teso-Fz-Betoño, A data augmentation-based technique for deep learning applied to CFD simulations, *Mathematics*, **9** (2021), 1843. <https://doi.org/10.3390/math9161843>
19. N. Thuerey, K. Weißenow, L. Prantl, X. Hu, Deep learning methods for Reynolds-averaged Navier–Stokes simulations of airfoil flows, *AIAA J.*, **58** (2020), 25–36, <https://doi.org/10.2514/1.J058291>
20. R. Fang, D. Sondak, P. Protopapas, S. Succi, Deep learning for turbulent channel flow, preprint paper, 2018. <https://doi.org/10.48550/arXiv.1812.02241>

21. K. Champion, B. Lusch, J. N. Kutz, S. L. Brunton, Data-driven discovery of coordinates and governing equations, *Proc. Natl. Acad. Sci. USA*, **116** (2019), 22445–22451. <https://doi.org/10.1073/pnas.1906995116>
22. K. Fukami, T. Murata, K. Zhang, K. Fukagata, Sparse identification of nonlinear dynamics with low-dimensionalized flow representations, *J. Fluid Mech.*, **926** (2021), A10. <https://doi.org/10.1017/jfm.2021.697>
23. R. Maulik, T. Botsas, N. Ramachandra, L. R. Mason, I. Pan, Latent-space time evolution of non-intrusive reduced-order models using Gaussian process emulation, *Phys. D Nonlinear Phenom.*, **416** (2021), 132797. <https://doi.org/10.1016/j.physd.2020.132797>
24. L. Agostini, Exploration and prediction of fluid dynamical systems using auto-encoder technology, *Phys. Fluids*, **32** (2020), 067103. <https://doi.org/10.1063/5.0012906>
25. R. King, O. Hennigh, A. Mohan, M. Chertkov, From deep to physics-informed learning of turbulence: Diagnostics, preprint paper, 2018. <https://doi.org/10.48550/arXiv.1810.07785>
26. R. Maulik, B. Lusch, P. Balaprakash, Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders, *Phys. Fluids*, **33** (2021), 037106. <https://doi.org/10.1063/5.0039986>
27. F. J. Gonzalez, M. Balajewicz, Deep convolutional recurrent autoencoders for learning low-dimensional feature dynamics of fluid systems, preprint paper, 2018. <https://doi.org/10.48550/arXiv.1808.01346>
28. G. Iaccarino, A. Ooi, P. A. Durbin, M. Behnia, Reynolds averaged simulation of unsteady separated flow, *Int. J. Heat Fluid Flow*, **24** (2003), 147–156. [https://doi.org/10.1016/S0142-727X\(02\)00210-2](https://doi.org/10.1016/S0142-727X(02)00210-2)
29. S. Osher, S. Chakravarthy, Upwind schemes and boundary conditions with applications to Euler equations in general geometries, *J. Comput. Phys.*, **50** (1983), 447–481, [https://doi.org/10.1016/0021-9991\(83\)90106-7](https://doi.org/10.1016/0021-9991(83)90106-7)
30. Siemens Software, 2023. Available from: <https://www.plm.automation.siemens.com/global/en/>.
31. F. R. Menter, Two-equation eddy-viscosity turbulence models for engineering applications, *AIAA J.*, **32** (1994), 1598–1605. <https://doi.org/10.2514/3.12149>
32. B. N. Rajani, A. Kandasamy, S. Majumdar, Numerical simulation of laminar flow past a circular cylinder, *Appl. Math. Model.*, **33** (2009), 1228–1247. <https://doi.org/10.1016/j.apm.2008.01.017>
33. M. M. Rahman, M. M. Karim, M. A. Alim, Numerical investigation of unsteady flow past a circular cylinder using 2-D finite volume method, *J. Nav. Arch. Mar. Engg.*, **4** (1970), 27–42. <https://doi.org/10.3329/jname.v4i1.914>
34. I. Aramendia, U. Fernandez-Gamiz, E. Zulueta Guerrero, J. Lopez-Guede, J. Sancho, Power control optimization of an underwater piezoelectric energy harvester, *Appl. Sci.*, **8** (2018), 389. <https://doi.org/10.3390/app8030389>
35. S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, S. Kaushik, Prediction of aerodynamic flow fields using convolutional neural networks, *Comput. Mech.*, **64** (2019), 525–545. <https://doi.org/10.1007/s00466-019-01740-0>
36. L. F. Richardson, J. A. Gaunt, VIII. The deferred approach to the limit, *Philos. Trans. Royal Soc. London. Series A Containing Papers Math. Phys. Char.*, **226** (1927), 299–361. <https://doi.org/10.1098/rsta.1927.0008>
37. A. Roshko, Vortex shedding from circular cylinders at low Reynolds numbers, *J. Fluid Mech.*, **46** (1971), 749–756. <https://doi.org/10.1017/S002211207100082X>



38. Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, *Proc. IEEE*, **86** (1998), 2278–2324. <https://doi.org/10.1109/5.726791>
39. O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, In: *Navab, N., Hornegger, J., Wells, W., Frangi, A. (eds) Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015. MICCAI 2015. Lecture Notes in Computer Science*, **9351** (2015), 234–241. [https://doi.org/10.1007/978-3-319-24574-4\\_28](https://doi.org/10.1007/978-3-319-24574-4_28)
40. M. Raissi, P. Perdikaris, G. E. Karniadakis, Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations, *J. Comput. Phys.*, **378** (2019), 686–707. <https://doi.org/10.1016/j.jcp.2018.10.045>
41. A. Kashefi, T. Mukerji, Physics-informed PointNet: A deep learning solver for steady-state incompressible flows and thermal fields on multiple sets of irregular geometries, *J. Comput. Phys.*, **468** (2022), 111510. <https://doi.org/10.1016/j.jcp.2022.111510>
42. X. Jin, S. Cai, H. Li, G. E. Karniadakis, NSFnets (Navier-Stokes flow nets): Physics-informed neural networks for the incompressible Navier-Stokes equations, *J. Comput. Phys.*, **426** (2021), 109951. <https://doi.org/10.1016/j.jcp.2020.109951>
43. A. Kashefi, T. Mukerji, Prediction of fluid flow in porous media by sparse observations and physics-informed PointNet, *Neural Networks*, **167** (2022), 80–91. <https://doi.org/10.1016/j.neunet.2023.08.006>
44. A. Kashefi, T. Mukerji, Chatgpt for programming numerical methods, *J. Mach. Learn. Model. Comput.*, **4** (2023), 1–74. <https://doi.org/10.1615/JMachLearnModelComput.2023048492>
45. V. Kumar, L. Gleyzer, A. Kahana, K. Shukla, G. E. Karniadakis, MyCrunchGPT: A chatGPT assisted framework for scientific machine learning, *J. Mach. Learn. Model. Comput.*, **2023**. <https://doi.org/10.1615/JMachLearnModelComput.2023049518> 2023
46. D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, preprint paper, 2014. <https://doi.org/10.48550/ARXIV.1412.6980>
47. I. Loshchilov, F. Hutter, Decoupled weight decay regularization, preprint paper, 2017. <https://doi.org/10.48550/ARXIV.1711.05101>



AIMS Press

© 2023 the Author(s), licensee AIMS Press. This is an open access article distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/4.0>)