Expert Systems With Applications 177 (2021) 114924

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Model-free short-term fluid dynamics estimator with a deep 3D-convolutional neural network

Manuel Lopez-Martin [a,*,1], Soledad Le Clainche [b,2], Belen Carro [a,3]

[a] ETSIT, Universidad de Valladolid, Paseo de Belen 15, Valladolid 47011, Spain
[b] ETSIAE, Universidad Politecnica de Madrid, Plaza Cardenal Cisneros, 3, Madrid 28040, Spain

ARTICLE INFO

Keywords:
Computational fluid dynamics
Prediction
Deep learning
Convolutional neural network

ABSTRACT

Deep learning models are not yet fully applied to fluid dynamics predictions, while they are the state-of-the-art solution in many other areas i.e. video and language processing, finance, robotics . Prediction problems on high-dimensional, complex dynamical systems require deep learning models devised to avoid overfitting while maintaining the required model complexity. In this work we present a deep learning prediction model based on a combination of 3D convolutional layers and a low-dimensional intermediate representation that is specifically designed to forecast the future states of this type of dynamical systems. The model predicts p future velocity-field time-slices (samples) based on k past samples from a training dataset consisting of a synthetic jet in transitional regime. The complexity of this flow is characterized by two topology patterns that are periodically changing, making this flow as a suitable example to test the performance of deep learning models to predict time states in complex flows. Moreover, the wide number of applications of synthetic jets (i.e.: fluid mixing, heat transfer enhancement, flow control), points out this example as a reference for future applications, where modeling synthetic jet flows with a reduced computational effort is needed. This work additionally opens up research opportunities for other areas that also operate with complex and high-dimensional time-series data: future frame video prediction, network traffic forecasting, network intrusion detection .

The proposed model is presented in detail. A comprehensive analysis of the results is provided. The results are based on a strict validation strategy to ensure its generalization. The model offers an average symmetric mean absolute error (sMAPE) and a relative root mean square error (RRMSE) of 1.068 and 0.026 respectively (one order of magnitude improvement over low-rank approximation tools), using 10 past samples and predicting 6 future samples of a two-dimensional velocity field on a 70x50 point matrix associated to a synthetic jet dataset.

1. Introduction

Computational fluid dynamics prediction problems are challenging in terms of complexity of the underlying physical model and computational resources required. Model-free data-driven paradigms are recent promising attempts to address this area without assumptions about the intrinsic physical model. Data-driven models have two main approaches, (i) those that have a concern on the inference of a reduced order model (i.e. dynamic mode decomposition) and, (ii) those that focus exclusively on prediction. Machine learning techniques based on deep neural networks correspond to the latter approach.

Fluid dynamics is linked to complex, high-dimensional systems, and its prediction is particularly challenging, as it requires not only a rich learning model, but also an adequate mapping function that generates an intermediate representation in a low-dimensional feature space (latent space) to reduce its computational needs. Solutions to the prediction problem for fluid dynamics can also be useful for other areas of a similar nature, with complex and high-dimensional time-series data: future frame video prediction, network traffic forecasting, network intrusion detection .

Complex flows, namely flows in transitional and turbulent regime, are present in several engineering, industrial and natural applications.

* Corresponding author.
E-mail addresses: manuel.lopezm@uva.es (M. Lopez-Martin), soledad.leclainche@upm.es (S. Le Clainche), belcar@tel.uva.es (B. Carro).
[1] ORCID: https://orcid.org/0000-0003-3550-560X.
[2] ORCID: https://orcid.org/0000-0003-3605-7351.
[3] ORCID: https://orcid.org/0000-0001-7051-8479.

https://doi.org/10.1016/j.eswa.2021.114924
Received 9 April 2020; Received in revised form 12 January 2021; Accepted 17 March 2021
Available online 20 March 2021

For instance, in nature it is possible to find complex flows describing the movement of the blood in vessels, the wake of flying insects, the movements of some marine animals (Le Clainche, 2019a). In this line, Le Clainche, 2019a proposed a new data-driven approach based on modal decompositions and physical principles to predict the temporal evolution of the flow. Some examples in the field of engineering and the industry include the flow inside heat exchangers, power plants or combustion systems, the flow past transport vehicles (i.e.: cars, aircrafts, submarines), micro- aerial or unmanned aerial vehicles (micro-AVs or UAVs), to name a few.

It is well known that the effect produced by complex flows can be undesirable in some cases. Turbulence increases the drag in several industrial devices and vehicles, producing fatigue loads or structural vibrations and rising the quantity of fuel consumption, pollution and cost. In the same line of modelling industrial turbulent flows, Marusic, Candler, Interrante, Subbareddy, and Moss (2003) proposed a new approach combining selective data storage with data mining tools to analyze the data based on real-time feature extraction. In biological flows, such as blood, turbulence occurs in pathological situations (i.e.: medical implants), triggering negative biological responses such as coronary artery diseases. For instance, Ferrari, Werner, Bahrmann, Richartz, and Figulla (2006) performed doppler wires experiments to measure the coronary flow velocity reserve (CFVR), which remains reduced in some patients after coronary angioplasty. They found an underestimation of the CFVR caused by turbulent flow when the coronary flow velocity was above the velocity of the critical Reynolds number. On the contrary, the effects of turbulence can be positive in some situations, for instance, turbulent flows enhance the fluid mixing properties or the heat transfer. For all these reasons, among others, studying and understanding complex flow behavior is a research topic of high interest.

During the last 20 years, the community has paid special attention to model complex flows. The main drawback lies in the disparate number of spatio-temporal scales involved in the flow motion. Using numerical simulations, it is possible to model complex flows defined in large computational domains, containing a sufficiently high number of grid points to properly solve the spatio-temporal evolution of the flow structures. In particular, the number of degrees of freedom is proportional to $Re^{9/4}$, where Re is the Reynolds number (non-dimensional number comparing viscous and convective terms), which defines the flow complexity and is very high in the case of turbulent flows (at least on the order of millions). Modelling and analyzing complex flows require a great investment in computational resources, computational time and memory, which is proportional to the number of degrees of freedom defining the problem. Hence, big data and complex flows are two equivalent terms, characterized by the volume, veracity and variety of the information contained (Le Clainche, 2019b). During the last years, the continuous search for finding new methods providing high-fidelity low-rank approximations that model flow dynamics has become an important research topic (Le Clainche & Ferrer, 2018; Gao, Zhang, Kou, Liu, & Ye, 2017).

Reduced order models (ROMs) provide low-rank approximations of complex dynamical systems. In fluid dynamics, ROMs provide general approximations of complex flows that can be used (i) to extract spatio-temporal information suitable to understand the underlying physics of the problem solved, (ii) to create powerful tools for flow control (Gao et al., 2017) and optimization (Park et al., 2013) and (iii) to predict different time states, reducing the computational cost (time and memory) in numerical simulations (Le Clainche, Varas, & Vega, 2017) or minimizing the number of information collected in experiments.

In the field of fluid dynamics, depending on the type of data available and the expert knowledge, it is possible to distinguish two types of ROMs. These are:

- *Pre-processed ROMs*. These types of ROMs are based on the Galerkin projection of the full state equations into sub-spaces of smaller dimension, which are defined using a modal basis that can be defined using several techniques, for instance, proper orthogonal decomposition (Noack, Morzynski, & Tadmor, 2011), dynamic mode decomposition (Luchtenburg, Noack, & Schlegel, 2009), proper generalized decomposition (Chinesta, Keunings, & Leygue, 2014), reduced basis methods (Quarteroni, Manzoni, & Negri, 2016), This method solves the full state equations for a reduced time, extracting some relevant information that is then used to create the sub-space basis. The reduced dimension equations are then solved, providing information about the time sate evolutions, with a reduced computational cost, proportional to the reduction in degrees of freedom of the equation solved.
- *Data-driven ROMs*. These types of ROMs extract relevant information from the full state equations to describe the flow as a modal expansion, that can be extrapolated in time. These ROMs are generated using purely data-driven methods, which is advantageous for two main reasons: (i) it is possible to create a model without the need of a priori knowledge of the underlying equations (Le Clainche & Vega, 2017) and (ii) the model can be constructed using either numerical or experimental data (Le Clainche, Vega, & Soria, 2017).

Using machine learning strategies, it is also possible to reduce the dimensionality of the data and to predict state variables in complex dynamical systems. Although these types of methods are not yet fully developed in the field of fluid dynamics, machine learning, and particularly deep learning, is presented as a new powerful tool for data-driven system identification (Brunton, Noack, & Koumoutsakos, 2020).

Deep learning algorithms (LeCun, Bengio, & Hinton, 2015) are based on a sequence of neural network layers with more than 3–4 layers of (usually) non-linear nodes with some layers implementing complex functions (convolutions, recurrence,..). The training of these models is done by optimizing a cost function using some form of gradient descent. They are extremely good in representation learning which is a real advantage to avoid difficult and costly feature engineering (LeCun et al., 2015). These algorithms are recently applied to a large number of problems with extremely good results in most cases (Dargan, Kumar, Ayyagari, & Kumar, 2019).

This article introduces a novel application of deep learning to predict state variables (velocity field) in a complex flow. More specifically, an ad-hoc deep neural network (DNN) architecture is applied to a dataset obtained by modelling a synthetic jet in transitional regime. The proposed DNN model consists of several blocks of 3D convolutional layers (Rawat & Wang, 2017) specifically designed to perform a dimensionality reduction along the spatio-temporal dimensions and with a final stage that creates a low-dimensional vector representation (embedding) that incorporates all the information required for the prediction. The model predicts p future velocity-field time-slices (samples) based on k past samples from the dataset. The embedding is used by the last layers as a common input to create each of the particular predictions (p velocity-field time-slices). The model does not incorporate recurrent layers (e.g., long short-term memory-LSTM) that are a usual component of multivariate time-series prediction but requires longer training and prediction time. The proposed deep learning model is presented in detail with an in-depth analysis of the prediction results showing state-of-the-art (SOTA) performance metrics compared to alternative methods (Le Clainche, 2019a).

The main **goal** of this work is to present a novel application of machine learning in fluid dynamics, to predict the temporal states in a synthetic jet in transitional regime with the aim of reducing time and computational requirements in numerical simulations. In a previous work (Le Clainche, 2019a), a low-rank approximation model (data-driven ROM) based on proper orthogonal decomposition (POD) (Sirovich, 1987) and dynamic mode decomposition (DMD) (Schmid, 2010) was successfully tested in a synthetic jet flow, however these techniques are based on the physical description of the flow and the detection of coherent structures. The model presented in this work is not based on

any physical model and the training and prediction phases require few computational resources compared to other alternative techniques (e.g., POD, DMD). As it is presented below, the proposed model can achieve excellent prediction performance metrics for multi-step ahead predictions of a high-dimensional spatio-temporal problem using a very reduced number of past samples.

The **contributions** of this work are the following: (a) Novel deep learning architecture based on 3D convolutional layers with a low-dimensional intermediate representation applied to the fluid dynamics field. (b) Present a model that achieves excellent prediction performance metrics for multi-step ahead predictions of a high-dimensional spatio-temporal problem using a very reduced number of past samples. (c) Data-driven model not based on a prior physical model. (d) The training and prediction phases require few computational resources compared to other alternative data-driven techniques (e.g., DMD, POD) and alternative deep learning methods (e.g., recurrent neural networks).

The paper is organized as follows: Section 2 summarizes previous works. Section 3 describes the dataset and the proposed model. Section 4 provides a description of the results and Section 5 presents the conclusions.

2. Related works

Machine learning algorithms have been applied to several areas of computational fluid dynamics (CFD) (Brunton et al., 2020), although it is recognized that their application is discreet compared to other fields, especially considering the great promise of benefits suggested by CFD experts (Kutz, 2017; Brunton et al., 2020). This paper tries to address this issue and provides a novel deep learning solution to predict state variables in a complex dynamical system generated by a synthetic jet in transitional regime.

Reviewing the areas where machine learning has been applied to CFD, we can appreciate the growing interest in this line of research: Brunton et al. (2020) present a survey on different application areas, mainly focused on system identification, flow features extraction and dimensionality reduction, flow modeling and control, but not on state flow predictions. Pathak, Hunt, Girvan, Lu, and Ott (2018) propose an echo-state-network based on recurrent neural networks that focuses specifically on predicting the state evolution of a chaotic system. Velocity field estimation for particle image velocimetry (PIV) is proposed in (Cai, Zhou, Xu, & Gao, 2019; Cai, Lian, Gao, Xu, & Wei, 2019; Lee, Yang, & Yin, 2017) using several 2D convolutional neural network (CNN) architectures. These works do not propose a velocity prediction but an estimation of the velocity vectors from a sequence of images. Xiaoxiao, Wei, and Iorio (2016) provide a velocity field predictor for steady flows, using a CNN model with an encoding/decoding configuration. White, Ushizima, and Farhat (2019) propose a cluster network to perform simulations in fluid dynamics, this model requires extensive hyper-parameter search and tuning, but is faster than alternative methods based on Gaussian processes. Likewise, Vlachas, Byeon, Wan, and Sapsis (2019) offer a comparison between forecasting high-dimensional dynamics with Gaussian processes versus the use of a recurrent neural network (LSTM), showing an improvement in forecasting accuracy. In the same line of work, Wan, Vlachas, Koumoutsakos, and Sapsis (2018) present a solution to model complex dynamical systems under extreme events, using a recurrent neural network (RNN) to help improving a reduced-order model in locations where data are available. A model combining convolutional and recurrent layers in an encoder-decoder architecture is presented in (Wiewel, Becher, & Thuerey, 2019) to predict changes of pressure fields over time for fluid flows. Lusch et al. (2017) propose a generalization of Koopman representation in a linear embedding using a modified deep auto-encoder with the intention of maintaining the physical interpretability of the Koopman approach with the higher efficiency of a deep neural network.

Recent research in the medical field employs sophisticated deep learning architectures based on convolutional layers. The work in (Lu, Wang, & Zhang, 2020) presents an interesting solution based on transfer learning using a pre-trained AlexNet network (based on 2D CNN) where the last layers are replaced by an extreme learning machine optimized by chaotic bat algorithm. The method is applied to abnormal brain detection using magnetic resonance image (MRI). In the same application domain, for classification of multiple sclerosis with MRI, (Wang & Zhang, 2020) also propose a transfer learning solution with a DenseNet architecture based on 2DCNN layers. With a different architecture, (Lopez-Martin, Nevado, & Carro, 2020) present a randomized 2D CNN architecture applied to the detection of early stages of Alzheimer's disease based on Magnetoencephalography (MEG) activity.

All previous works apply several deep learning models to fluid dynamics estimation or prediction tasks, but not with the solution (3D CNN with a low-dimensional intermediate representation) and objectives (k-ahead velocity-field prediction for a synthetic jet) presented in the proposed architecture.

The problem addressed in this article is also related to the challenge of future frame video prediction, which is also an open problem in itself. Castelló (2018) provides a comprehensive review of video prediction with an analysis of the challenges posed. Some solutions incorporate LSTM networks with only a few proposing exclusively convolutional networks (Mathieu, Couprie, & LeCun, 2016; Vukotic, Pintea, Raymond, Gravier, & Gemert, 2017). Most solutions are based on smooth frame transitions and focus on specific video sequences (e.g., human actions or poses). It is important to mention the specific nature of the velocity fields produced by a synthetic jet, which are not always smooth and are not related to everyday video sequences.

Considering the application of deep learning to generic multivariate multioutput time-series forecasting, in addition to the application of classic statistics techniques (Borchani, Varando, Bielza, & Larrañaga, 2015) there are many recent works that employ convolutional and recurrent neural networks and ensemble solutions: Huang, Chiang, and Li (2017) and Lopez-Martin, Carro, and Sanchez-Esguevillas (2019) to cite a few.

Connected with the problem of fluid dynamics and time-series prediction, in weather forecasting there are also works exploring the application of deep learning models instead of or in combination with dynamical systems simulation techniques. Scher (2018) and Scher and Messori (2019) present a deep learning model based on different autoencoder architectures using 2D convolutional neural networks to emulate the dynamics of a simple general circulation model. Wang, Balaprakash, and Kotamarthi (2019) propose an alternative to physics-based predictions using an ad-hoc deep learning architecture with fully connected layers. Agrawal et al. (2019) present a recent contribution to rainfall forecasting using a U-Net which is a specific encoder/decoder architecture with 2D convolutional layers. In the referred works, the results obtained with deep learning models are comparable or better than SOTA models for short-term predictions.

As a summary, the proposed architecture presents several intersections with related works: CFD analysis and prediction using deep learning, video frames prediction, multivariate multioutput time-series forecasting; and with similar or related fields: medical image analysis and weather forecast. In these related research activities, there is a common interest in then application of ad-hoc and sophisticated deep neural network architectures based mainly on 2D convolutional and recurrent layers.

3. Methods description

This Section provides a detailed description of the dataset used for the experiments and the proposed model to perform the velocity-field predictions of the fluid flow. The dataset and the proposed model are presented on Sections 3.1 and 3.2 respectively.

3.1. Selected dataset

A synthetic jet, also known as zero-net-mass flux jet (Carter & Soria,

2002) is a fluid stream that is formed by the periodic ejection of vortex rings from a cavity. The cavity contains a piston or a membrane that oscillates with a periodic movement, forcing the flow to periodically leave and to re-enter into the cavity through a small orifice, the jet nozzle (Glezer & Amitay, 2002). This characteristic feature of synthetic jets makes very attractive using these devices for several industrial applications. For instance, some of the most popular applications include active flow control of boundary layer (Cattafesta & Sheplak, 2010), plasma actuators (Zong & Kotsonis, 2018), heat transfer enhancement (Pavlova & Amitay, 2006) and fluid mixing (Wang & Menon, 2001). Moreover, synthetic jets also model natural propulsion systems such as the swimming motion of some marine animals like jellyfish, squids or salps (DeMont & Gosline, 1998).

The flow generated by a synthetic jet with a cylindrical cavity and circular jet nozzle is modelled using numerical simulations. The flow is governed by the non-linear incompressible Navier-Stokes equations, defined as

$$\nabla \cdot \boldsymbol{V} = 0, \tag{1}$$

$$\frac{\partial \boldsymbol{V}}{\partial t} + (\boldsymbol{V} \cdot \nabla \boldsymbol{V}) = -\nabla p + \frac{1}{\mathrm{Re}} \Delta \boldsymbol{V}, \tag{2}$$

where p and $\boldsymbol{V}$ are the non-dimensional pressure and velocity vector ($\boldsymbol{V} = (V_x, V_y)$, with V_x and V_y as the streamwise and normal velocity components) and Re is the Reynolds number, defined as $\mathrm{Re} = \frac{\mathrm{UD}}{\nu}$, where U, D, ν are the jet mean momentum velocity (Le Clainche, 2019a), the diameter of the jet nozzle and the kinematic viscosity of the fluid. Two main parameters characterize this type of flow, the Reynolds number, and the Strouhal number, defined as $\mathrm{St} = \frac{f\mathrm{D}}{\mathrm{U}}$, where f represents the piston (or membrane) oscillation frequency. This article analyses the numerical dataset generated and analysed in (Le Clainche, 2019a), for $\mathrm{Re} = 1000$ and $\mathrm{St} = 0.03$, with $D = 1$ and $U = 1$. This database has been generated using the solver Nek5000 (Fischer et al., n.d.), solving the Navier-Stokes Eqs. (1) and (2). The solver uses as spatial discretization spectral elements with Gauss-Lobatto-Legendre points of polynomial order $\Pi = 8$ (each two-dimensional element is discretized using $\Pi - 1 = 7$ points in each direction, hence it contains 49 grid points). The temporal discretization uses an implicit second order backwards differentiation scheme for the viscous terms and an explicit second order extrapolation scheme for the non-linear terms (Ohlsson, Schlatter, Fischer, & Henningson, 2010). Based on the diagram presented in (Carter & Soria, 2002), at the present conditions ($\mathrm{Re} = 1000$, $\mathrm{St} = 0.03$) the flow is laminar in the near field, but it transitions to turbulence in the far field. The computational domain is formed by a two-dimensional rectangle with length L = 80D and height H = 40D, connected to a small cavity with dimension 5D × 2.5D (the diameter is $D_p = 5D$) by the jet nozzle, as presented in Fig. 1. The boundary conditions imposed are: (1) at the inlet surface, the external surface of the cavity upstream the jet nozzle, Neumann and Dirichlet boundary conditions are imposed for the pressure and for the velocity, defined as $(V_x, V_y) = (V_p \sin(2\pi f t), 0)$, where the streamwise velocity defines the periodic movement of the piston oscillating in time t with frequency $\mathrm{St} = 0.03$, and V_p is the peak amplitude of the piston velocity, $V_p = D_p D/\sqrt{2}$, (2) at the outlet surface zero stress and Dirichlet boundary conditions are imposed for the velocity ($\nabla \boldsymbol{V} \cdot \boldsymbol{n} = 0$, with $\boldsymbol{n}$ = unit normal) and the pressure ($p = 0$), respectively, (3) non-slip boundary conditions, $(V_x, V_y) = (0, 0)$, are imposed in the wall of the jet and left surfaces of the domain, finally, (4) axi-symmetric condition is imposed in the bottom part of the domain.

The flow complexity in synthetic jets is mainly characterized by the two different topologies describing the flow that are periodically changing. As presented in Fig. 1., when the flow is ejected through the orifice (injection phase), a vortex ring is created that travels downstream. On the contrary, a saddle point is identified when the flow is injected through the jet nozzle (suction phase), which separates the flow re-entering into the cavity from the flow that continues moving downstream.

These two topology patterns are identified in the data analysed in this article and predicted using deep neural networks. For simplicity, these patterns will be represented by the different velocity fields, streamwise and radial velocity components, represented as *Vx* and *Vy*, which periodically change from values on the order of magnitude of U (injection phase) to zero (suction phase) as presented in Fig. 2.

A schematic of the dataset used to train/test the prediction model is provided in Fig. 3. The dataset consists of a temporal sequence of velocity-fields, each of them formed by a 3-dimensional volume composed of a surface of 70 × 100 points (x, y dimensions) with the components x and y for the velocity of each point included in the third dimension. The details of a velocity-field in Fig. 3-A corresponds to a time-slice of the complete dataset (Fig. 3-B). The complete dataset is formed by a temporal sequence of 16,112 time-slices each of them corresponding to a velocity field. Each oscillation period (injection + suction phase) of the piston or the membrane inside the cavity upstream the jet nozzle is represented by 624 time-slices, thus the total dataset represents ~25 periods of flow oscillation, modeling the transient and the saturated regime of the numerical simulation. The first 11,642 time-slices are reserved to train the model, with the following 2054 and 2416 time-slices reserved as validation and test sets, respectively. Therefore, the fractions of the entire dataset for the training, validation, and test sets are 72.3%, 12.7%, and 15%, respectively. All prediction performance metrics presented in Section 4 are obtained with the test set exclusively. The validation set is used to determine when to stop training and to choose the best weights for the prediction model.

To facilitate the training of the prediction model and to reduce data volume and memory requirements, a spatial down-sampling of the

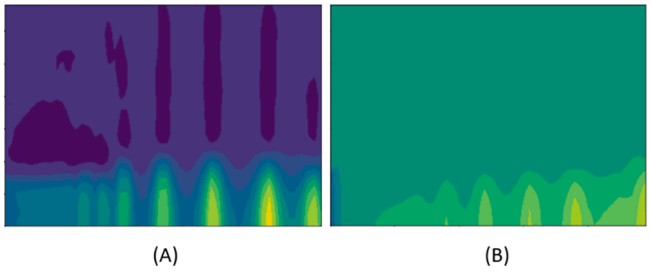

Fig. 2. Contour lines representation of the velocity fields shown in Fig. 1 for the two topology patterns: vortex rings (A) and saddle point (B). In both cases, only the streamwise component of the velocity field is shown.

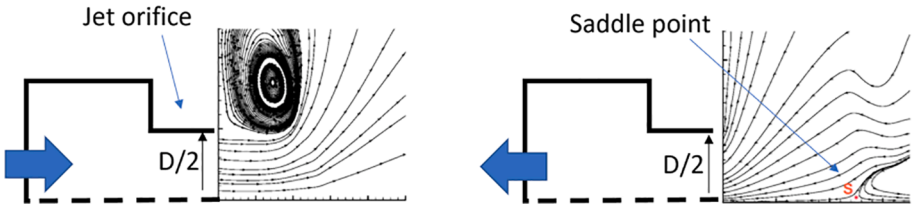

Fig. 1. Streamlines describing the two topology patterns characterizing a synthetic jet with diameter D. The streamlines are represented in half of the jet domain (axi-symmetric flow). The arrows point the flow direction. Left: vortex rings in the injection phase. Right: saddle point in the suction phase (marked with S).

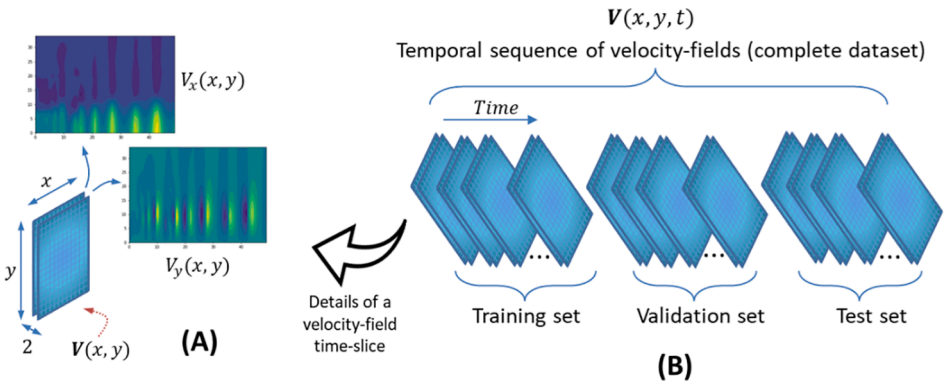

Fig. 3. Data structure to perform the experiments. (**A. Left part**) Details of a 3-dimensional velocity-field time-slice consisting of a surface with the velocity components x and y as parts of the third dimension. (**B. Right part**) Velocity-field time-slices following their temporal sequence. This sequence forms the complete dataset that is divided along the time dimension into training, validation and test sets.

surface was performed by taking one of every two consecutive spatial points in both directions (x and y), ending in a surface of 35 × 50 points. An additional min–max scaling of the components x and y of the velocity-field was performed reducing their value range to the [0–1] interval.

The dataset described in Fig. 3 is used to perform the prediction of p future velocity-field time-slices using the information contained in the previous k time-slices. That implies that the velocity-field time-slices used for training, validation and testing must be aggregated into velocity-field data structures (VF-DS) as shown in Fig. 4, where each VF-DS is formed by $k + p$ consecutive time-slices. The referred VF-DS is created using a rolling-window until the corresponding training, validation or test sets are exhausted (Fig. 4). The sampling of time-slices starts after an initial offset and the forward movement of the rolling-window can be controlled by varying the jump distance between successive VF-DSs (stride). The initial offset is applied only for the training VF-DSs, and the strides applied for the training, validation and test sets can be different. The reason to include an initial offset is the possibility to avoid the irregular behavior of the first time-slices of the simulation, which corresponds to initial transient stage of the numerical simulations.

The results presented in Section 4 have been obtained with an initial offset of 0 (which means we include the transient stage in the training set), a stride value for the training VF-DSs of 2, a stride value of 1 for the validation and test VF-DSs and values for k and p of 10 and 6, respectively. With these parameter values, the final number of VF-DSs for training, validation and testing are 5813, 2038 and 2400 respectively. The parameter values used to form the training, validation and test sets are summarized in Table 1.

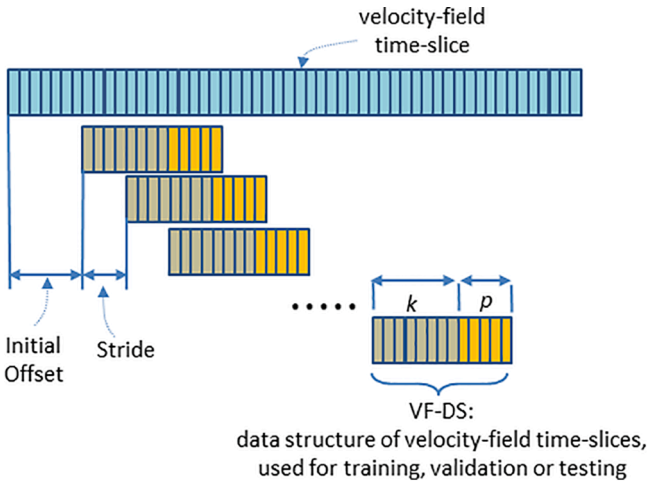

Fig. 4. Arrangement of time-slices of velocity-field into a velocity-field data structure (VF-DS) used for training, validation or testing. The initial offset and stride control the forward advancement of the rolling-window and, k and p are the number of velocity-field time-slices used as predictors and to be predicted, respectively.

3.2. Model description

The proposed model is presented schematically in Fig. 5. The objective of the model is to perform velocity-field predictions for the p future time-slices based on k past velocity-field inputs that start on an arbitrary initial time (t). The results presented in Section 4 are for a value of k and p of 10 and 6, respectively.

The challenges of the model are:

a) Prediction of a high-dimensional spatio-temporal multi-output
b) Big data volumes for the input
c) Need to build input data on the fly due to the inability to accommodate the memory requirements of an alternative solution based on a complete pre-built training set
d) Avoid an excessive number of weights (evade overfitting) while maintaining the required model complexity.
e) Training and prediction phases requiring few computational resources (time and memory)

The model (Fig. 5) is based on a multilayer deep neural network that receives a sequence of k velocity-field ($V(x,y)$) time-slices as input. The model starts with 3 initial blocks each formed by a 3D convolutional layer (Rawat & Wang, 2017), a max pooling layer (Lee, Gallagher, & Tu, 2015) and a batch normalization layer (Ioffe & Szegedy, 2015). It follows an additional 3D convolutional layer with a kernel of size 1 × 1 × 1 that performs an averaging along the remaining spatio-temporal dimensions which allows us to control the final depth dimension of the feature space that we chose to be the same as the number p of time-slices to be predicted. Following the 1x1 convolutional layer, there is a tensor reshape that performs an exchange of the first for the last dimension, and flattens all dimensions except the new first dimension, resulting in a 2D matrix where the first dimension is equal to p. The resulting matrix is applied to a tensor split that creates p vectors by breaking the previous 2D matrix by rows. Each of these vectors is the input to p fully connected (FC) layers all sharing weights. The outputs from these layers (p of them)

Table 1
Summary of the parameter values used to form the training, validation and test sets.

	Stride length				
Initial offset	**Training**	**Validation**	**Test**	**k**	**p**
0	2	1	1	10	6

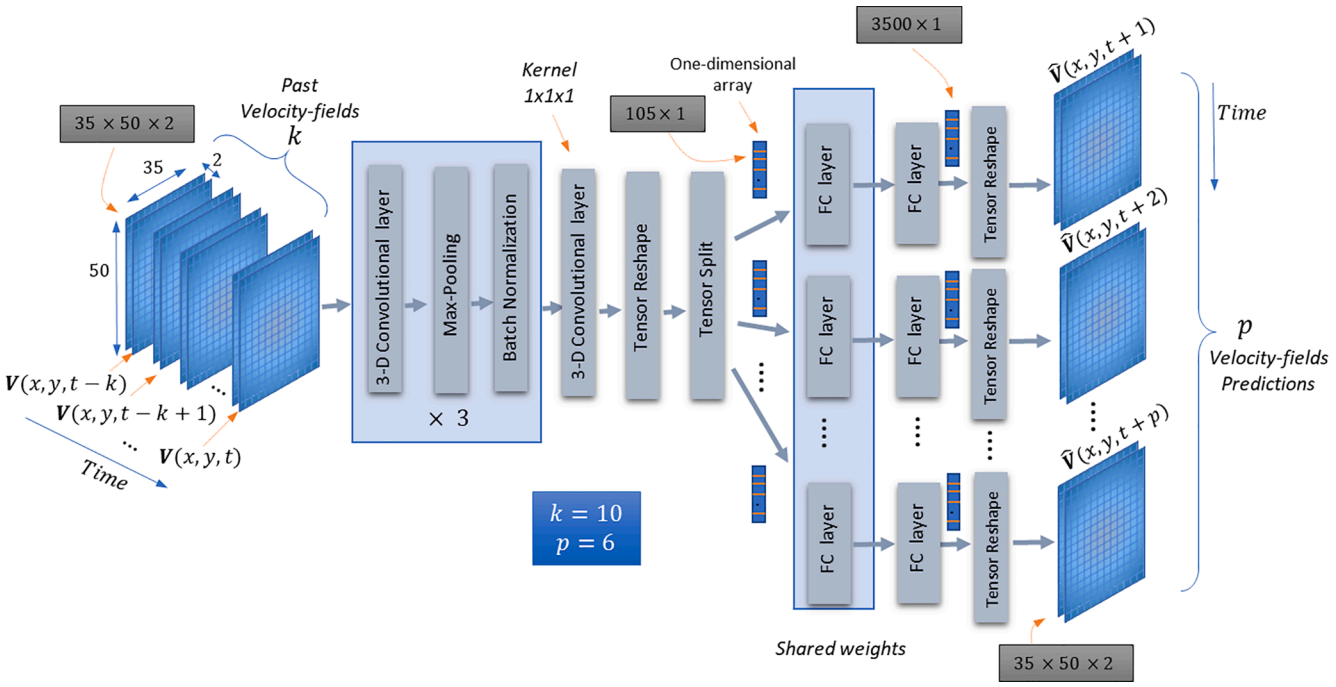

Fig. 5. Proposed deep learning model to perform velocity-field predictions for the p future time-slices based on k past velocity-field inputs that start on an arbitrary initial time (t).

are given as inputs to p subsequent FC layers with independent weights this time. The final output from the model are p vectors of length 35,000 which are finally reshaped into p predicted velocity-field time-slices ($\widehat{V}(x,y)$).

The first part of the model (the part before the FC layers) provides a representation learning that integrates, into p low-dimensional vectors, all the required information to perform the predictions. The weight sharing of the first FC layers is important to avoid overfitting and to force the model to learn the commonalities between all p predictions with a following FC layer creating the necessary differences between each of the p predictions.

The model is trained with a loss function based on the mean squared error between the ground-truth and predicted velocity-fields for all the p predicted outputs, and the optimization is done with batch Stochastic Gradient Descent (SGD) with Adam.

The mean squared error (MSE) used to train the proposed model is presented in Eq. (3), considering that: We represent the generic predicted velocity-field time-slices as $\widehat{V}(x,y)$ and the ground-truth velocity-field time-slices as $V(x,y)$, and any of them has the structure shown in Fig. 3-A. To represent a velocity-field time-slice for a specific time (t) we use $V(x,y,t)$ and $\widehat{V}(x,y,t)$. The velocity-field has two components $V = (V_x, V_y)$. The flow section is a rectangle of dimensions H × W, and $\|V\|$ represent the norm of the vector V. The MSE loss defined in Eq. (3) is for a specific $t$corresponding to a single VF-DS defined in Section 3.1. We consider the time (t) as a discretized sequence of indices.

The details of the model are provided in Table 2. The layer configuration in Table 2 corresponds to the best model whose results are presented in Section 4. The table presents the architecture description with a row to describe each layer. The first column in Table 2 is the layers order. The next columns present the following information, in order: (a) If the layer contains parallel branches and its number, in case of no parallel branches the number is 0. (b) Description of the layer. (c) The number of kernels or neurons of the layer. The number of kernels is applicable for convolutional/maxpooling layers and the number of neurons for fully connected layers. (d) The kernel size for convolutional/maxpooling layers. (e) The stride for convolutional/maxpooling layers. (f) The padding for convolutional/maxpooling layers with a V standing for VALID padding or and S for SAME padding. VALID implies no padding and SAME implies padding that preserves output dimensions. (g) The activation function of the layer. (h) In case of a layer with branches if the branches share their weights. (i) The number of weights in each layer or layer group (in case of branches). (j) The output dimensions for each layer or layer group (in case of branches). We have maintained the first index in the output dimensions for the batch size, while replacing the actual values of k and p in the expressions. The parameters k and p appear explicitly above the last column, together with the parameter bs corresponding to the batch size.

A batch is a set of VF-DSs (not necessarily consecutive) used to perform a training round. Each training round update weights averaging the contributions from each element of the batch. An epoch is the number of training rounds needed to pass all VF-DSs in the training set. The particular parameter values used to train the proposed model are $k = 10$, $p = 6$, $bs = 5$ and a number of epochs equal to 70. An early

$$MSE_{Loss}(t) = \frac{1}{2pHW}\sum_{\tau=t+1}^{t+p}\sum_{\substack{x=1\cdots W\\ y=1\cdots H}}\|V(x,y,\tau)-\widehat{V}(x,y,\tau)\|^2 = \frac{1}{2pHW}\sum_{\tau=t+1}^{t+p}\sum_{\substack{x=1\cdots W\\ y=1\cdots H}}[(V_x(x,y,\tau)-\widehat{V}_x(x,y,\tau))^2+(V_y(x,y,\tau)-\widehat{V}_y(x,y,\tau))^2] \quad (3)$$

Table 2
Details of the layers of the proposed model. The parameters *k*, *p* and *bs* are, respectively, the number of time-slice predictors, the number of predicted time-slices and the batch size.

Layer #	Parallel branches	Layer details	Number Kernels/ Neurons	Kernel size	Stride	Padding	Activation	Shared weights	Number Weights	**k = 10, p = 6 batch size (bs) = 5** Output dimensions
0	0	Input								bs × k × 35 × 50 × 2 bs × 10 × 35 × 50 × 2
1	0	Conv3D	5	2 ×2 ×2	1	V	ReLU	No	85	bs × 9 × 34 × 49 × 5
2	0	MaxPooling3D	1	1 ×2 ×2	1	V	None		0	5 × 9 × 17 × 24 × 5
3	0	BatchNormalization						No	20	bs × 9 × 17 × 24 × 5
4	0	Conv3D	10	2 ×2 ×2	1	V	ReLU	No	410	5 × 8 × 16 × 23 × 10
5	0	MaxPooling3D	1	1 ×2 ×2	1	V	None		0	bs × 8 × 8 × 11 × 10
6	0	BatchNormalization						No	40	bs × 8 × 8 × 11 × 10
7	0	Conv3D	20	2 ×2 ×2	1	V	ReLU	No	1620	bs × 7 × 7 × 10 × 20
8	0	MaxPooling3D	1	1 ×2 ×2	1	V	None		0	bs × 7 × 3 × 5 × 20
9	0	BatchNormalization						No	80	bs × 7 × 3 × 5 × 20
10	0	Conv3D	p	1 ×1 ×1	1	V	ReLU	No	126	bs × 7 × 3 × 5 × p bs × 7 × 3 × 5 × 6
11	0	Permute(4,1,2,3)							0	bs × 6 × 7 × 3 × 5
12	0	Reshape(p,105)							0	bs × 6 × 105
13	p	Split(p)							0	(bs × 105) * 6
14	p	Fully Connected	80				ReLU	Yes	8480	(bs × 80) *6
15	p	Fully Connected	3500				Sigmoid	No	283,500 * 6	(bs × 3500) * 6
							Total Weights =		**1,711,861**	

stopping was also used if the last 10 epochs do not reduce the loss function on the validation set.

Each layer description also provides the activation function employed. All layers apply a ReLU activation function with the exception of the last layer with a sigmoid function. The data to be predicted is scaled in the range [0–1], which is consistent with the output values of the sigmoid function. Other activation functions (e.g., linear) have been tried, with sigmoid activation providing the best results. We have also considered other configurations for the architecture of the proposed model, with the configuration presented in Fig. 5 providing the best results. In particular, different kernel sizes and number of convolutional layers were considered, as well as not using shared weights for the layers after the tensor split (Fig. 5).

To tune the network weights, we have used mini-batch gradient descent with early-stopping as an implicit regularization mechanism and best solution search. We have applied early stopping using the validation set to choose the best configuration. Early stopping is based on computing the validation loss (in our case the mean squared error) at the end of each epoch. If the validation loss at the end of a certain number of previous epochs does not obtain any reduction, the training process stops and the weights corresponding to the best validation loss are used as final weights. The waiting period (number of previous epochs used to compare any decrease in the validation loss) is called the patience value.

As a summary of the training parameters for the neural network in this work: (a) We have used Adam as the optimization method. The parameters used are α(learningrate): 0.001, β_1 0.9, β_2: 0.999 and ε: 1e−8, which are the default values proposed in (Kingma & Ba, 2014). (b) We have used mini-batch gradient descent with a batch size of 5, using 70 epochs for training with an early stopping using the past 10 epochs (patience value) to decide when to stop. (c) We have used as predictors a number of 10 (*k*) past velocity-field time-slices to predict 6 (*p*) future velocity-field time-slices. (d) Considering the considerable size of the dataset, instead of using the full training, validation and test sets, we have used a data-loader in charge of randomly building the data batches on-the-fly, requiring much less computer memory.

As regularization mechanisms we have used: (a) Early stopping (as described above), (b) weights sharing, (c) batchnormalization and (d) employ a model based on convolutional layers since they are extremely efficient in reusing weights. We have not used for regularization: dropout, activation or weights regularization (L2 regularization). The reason for not using these regularization methods is that they need to adjust additional hyper-parameters (e.g., dropout rate, regularization coefficient for L2) adding complexity to the training process.

The proposed model (Fig. 5) shows a good balance between good prediction performance with small computational and memory requirements. This is achieved by a combination of: (a) the 3D convolutional and maxpooling layers that reuse the kernel weights along all tensor dimensions with the maxpooling layer further reducing the output dimensions of the tensors, (b) the weights sharing of the last layers, and (c) the final tensor split into as many tensors as the number of predicted outputs.

To arrive to this architecture, we have investigated a series of alternative architectures presented schematically in Fig. 6, with each architecture identified with a letter (A–F). The first architecture (A) corresponds to the proposed model. Some architectures have as input a single tensor containing all k velocity-field time-slices (A,B,F). The rest

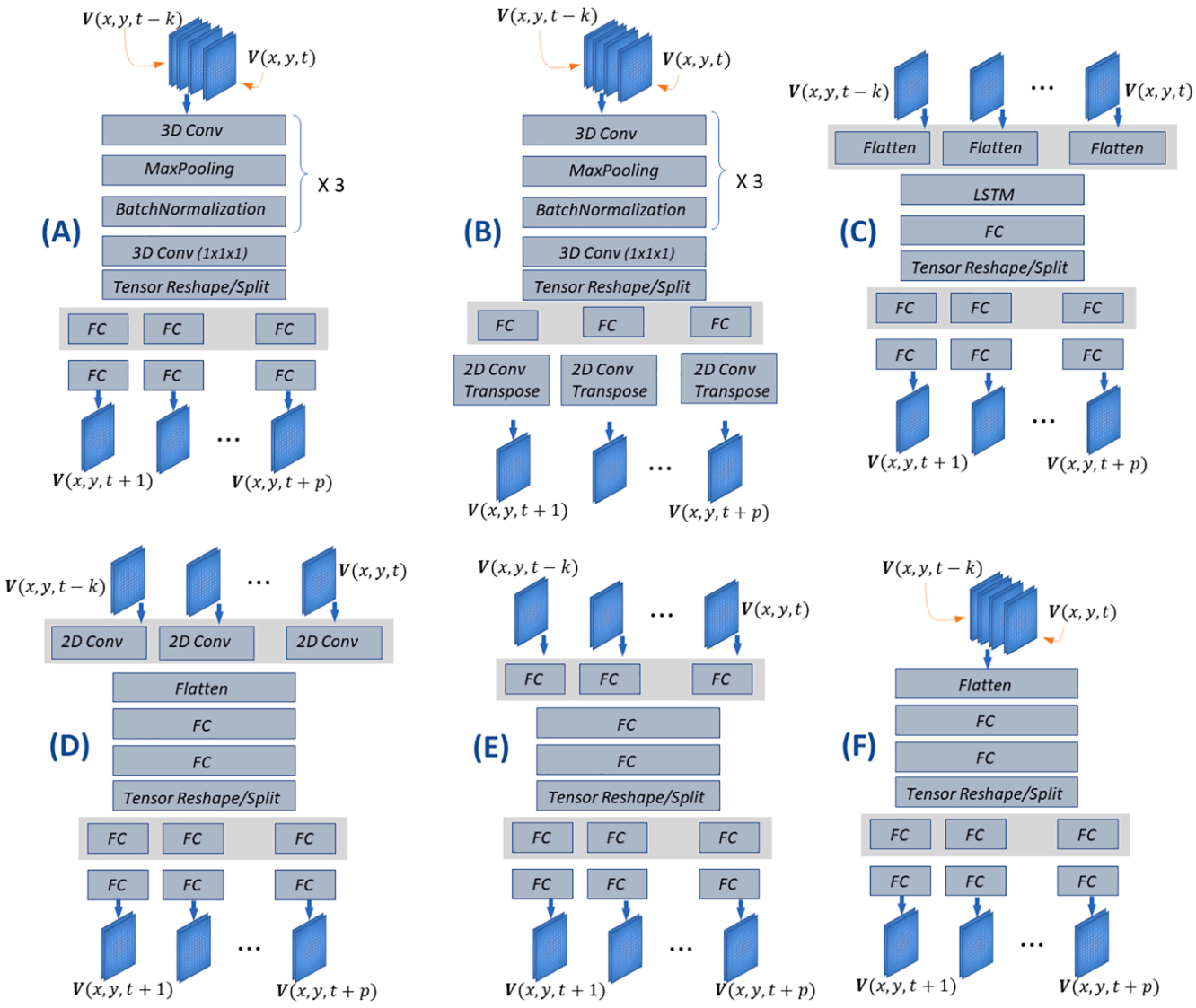

Fig. 6. Diagram of the different architectures considered for this problem. Each architecture is identified with a letter (A-F). The first architecture (A) corresponds to the proposed model. Some architectures have as input a single tensor containing all k velocity-field time-slices (A,B,F). The rest have a time-series input associated to each specific velocity-field time-slice (C,D,E). Two architectures (C,F) flatten their respective inputs.

have a time-series input associated to each specific velocity-field time-slice (C,D,E). Two architectures (C,F) flatten their respective inputs. In Section 4.2 are provided the prediction performance results of all these architectures, with the proposed model having the best results. A description of the different architectures is provided in Table 3. The layers description in Table 3 follows a simplified version of the convention used in (Lopez-Martin, Carro, Sanchez-Esguevillas, & Lloret, 2017): Conv3D/v(x,y,z) stands for a convolutional layer with v filters where x, y and z are the width, height and depth of the 3D kernel. MaxPooling(x,y,z) stands for a Max Pooling layer where x, y and z are the pool sizes. FC(x) stands for a fully connected layer with x nodes (number of hidden layer neurons). The convolutional and maxpooling layers all have a stride of 1 and a VALID padding. The activation function used for all layers is ReLU except the last layer which has a sigmoid activation. The letters WS at the end of a layer indicate that the layer implements weights sharing. Finally, the sign (×) in the layer description indicates a repetition of the object (layer or output tensor) the number of times indicated by the associated number.

The last three columns in Table 3 show if the architecture provides spatial and/or sequential inductive bias and a category of the size (number of weights) of the architecture. The exact number of weights for each architecture along with its prediction performance metrics are shown in Table 7. We can see that architectures A and B have the smallest size and are the only ones that incorporate both spatial and sequential inductive bias. Inductive bias expresses the set of assumptions and restrictions incorporated in a model about its data generation process or its solutions space (Cohen & Shashua, 2016). All machine learning algorithms have some type of inductive bias that provides the ability to generalize to test samples that were not seen at training time. Inductive bias allows a model to be data efficient and generalize to new data (Cohen, Sharir, Levine, Tamari, Yakira, & Shashua, 2017). In Table 3 the columns spatial and sequential inductive biases indicate the capacity of the architecture to be locality-aware in the case of spatial inductive bias or time-aware in the case of sequential inductive bias. Both biases made the models to be invariant to some type of data translation or transformation, in the case of spatial to spatial translation and in the case of sequential to time translation. Spatial inductive bias is found in convolutional layers and sequential inductive bias in recurrent (e.g., LSTM) layers (Mitchell, 2017). The 3D convolutional layer, with its 3D kernels, has the ability to locally explore both spatial and time dimensions, offering locality and sequential inductive biases with a reduced number of weights. We can observe how the other solutions fail

Table 3
Description of the different architectures considered for this problem. The model Id corresponds to the letter identifying each architecture in Fig. 6. The last three columns show if the architecture provide spatial and sequential inductive bias and a category of the size (number of weights) of the architecture. The exact number of weights for each architecture along with its prediction performance metrics are shown in Table 7.

Model Id.	Architecture	Inductive Bias		
		Spatial	Sequential	Size
A Proposed model*	Conv3D/5(2,2,2) + MaxPool(1,2,2) + BatchNorm + Conv3D/10(2,2,2) + MaxPool(1,2,2) + BatchNorm + Conv3D/20 (2,2,2) + MaxPool(1,2,2) + BatchNorm + Conv3D/6(1,1,1) + Split × 6 + [1FC(80)WS + 1FC(3500)] × 6	Yes	Yes	Medium
B	Conv3D/5(2,2,2) + MaxPool(1,2,2) + BatchNorm + Conv3D/10(2,2,2) + MaxPool(1,2,2) + BatchNorm + Conv3D/20 (2,2,2) + MaxPool(1,2,2) + BatchNorm + Conv3D/6(1,1,1) + Split × 6 + [1FC(850)WS + Conv2DTrans/2(2,2)]x 6	Yes	Yes	Small
C	LSTM (400) + 1FC(600) + Split × 6 + [1FC(100)WS + 1FC(3500)] × 6	No	Yes	Large
D	Split × 10 + Conv2D/20(4,4) × 10 + Flatten + 1FC(80) + 1FC(600) + Split × 6 + [1FC(100)WS + 1FC(3500)] × 6	Yes	No	Very Large
E	Split × 10 + 1FC(400) × 10 + 1FC(600) + Split × 6 + [1FC(100)WS + 1FC(3500)] × 6	No	No	Large
F	Flatten + 1FC(300) + 1FC(600) + Split × 6 + [1FC(100)WS + 1FC(3500)] × 6	No	No	Very Large

to provide some of these biases, creating more complex solutions with worse prediction performance. Architecture B is similar to the proposed solution, but replacing the last layer with a 2D transposed convolutional layer that reverses the function of a 2D convolutional layer (Wojna, Ferrari, Guadarrama, Silberman, Chen, Fathi, & Uijlings, 2017). Architecture B provides the smallest model size, but with worse results than the proposed solution (Table 7). In this case, the proposed solution seems to be a good balance between size/complexity and prediction performance.

4. Results

In the following Sections we present the results obtained with: (a) the proposed model (Section 4.1), (b) the results obtained with the alternative architectures presented in Section 3.2 which allow us to decide the best architecture for this problem (Section 4.2), (c) the results obtained when trying a flow with different characteristics and geometry (Section 4.3), and (d) the results obtained with different random initializations of the weights and the evolution of the training and validation losses during training (Section 4.4).

4.1. Proposed model

In this section we present the prediction performance results obtained with the proposed model presented in Section 3.2. All the results presented are based on the dataset described in Section 3.1 using exclusively the test set. The dataset consists of 11,642 consecutive time-slices where the last ones (2416) are reserved for testing. Each time-slice is made up of a 3D data volume that provides the x and y velocity-fields over a surface of 35 × 50 point. The test set is separated into blocks of $k + p$ consecutive time-slices forming a VF-DS data structure (Section 3.1). The VF-DS data structures are the basis for all test results. Parameters k and p correspond to the number of time-slices used as predictors and the number of predicted time-slices, respectively. For the results presented here, we will use a value of k equal to 10 and p equal to 6.

Considering the difficulties of this multivariate multi-output regression problem we will use several forecast metrics to ensure several points of view when analyzing the results. The forecast metrics applied are: mean squared error (MSE), mean absolute error (MAE), median absolute error (MAD), coefficient of determination (R^2), relative root mean squared error (RRMSE) and symmetric mean absolute percentage error (sMAPE). The definition of these metrics is the following, considering Y as the ground-truth values, $\widehat{Y}$ the predicted values, $\overline{Y}$ the mean value of Y and N the total number of scalar values for all predicted time-slices (Hyndman & Koehler, 2006):

$$MSE = Mean(\left(Y - \widehat{Y}\right)^2) \tag{4}$$

$$MAE = Mean(\left|Y - \widehat{Y}\right|) \tag{5}$$

$$MAD = Median(\left|Y - \widehat{Y}\right|) \tag{6}$$

$$RRMSE = \frac{\sqrt{\sum_{i=0}^{N}(Y_i - \widehat{Y}_i)^2}}{\sqrt{\sum_{i=0}^{N}(Y_i)^2}} \tag{7}$$

$$R^2 = 1 - \frac{\sum_{i=0}^{N}(Y_i - \widehat{Y}_i)^2}{\sum_{i=0}^{N}(Y_i - \overline{Y})^2} \tag{8}$$

$$sMAPE = \frac{100}{N}\sum_{i=0}^{N} 2\frac{\left|Y_i - \widehat{Y}_i\right|}{|Y_i| + \left|\widehat{Y}_i\right|}\% \tag{9}$$

The values associated to Y_i and $\widehat{Y}_i$ correspond to each component of the tensors $V(x,y,\tau)$ and $\widehat{V}(x,y,\tau)$, respectively; with the following range of parameter values: $x = 1\cdots W, y = 1\cdots H, \tau = t+1\cdots t+p$, $t = 1\cdots T_{test}$. Where T_{test} is the number of VF-DSs in the test set, W is 35, H is 50, and N is the total number of components in the p predicted tensors (V) for the total number (T_{test}) of VF-DSs used in the test phase, i.e. $N = 2 \times W \times H \times p \times T_{test}$. The number 2, in the previous expression for N, corresponds to the two components (x,y) of the velocity-field. In summary, the above metrics are computed by flattening the complete tensor of results for the p predictions of the entire test set, and associating the values of Y_i or $\widehat{Y}_i$ (ground-truth or predicted) to each component of the flattened tensor.

All metrics have values greater than zero with no upper limit, except R^2 that has an upper limit of 1 with no lower limit and sMAPE which has an upper limit of 200%. In all cases, the smaller the value, the better the result, except the R^2 metric, where the relationship is the opposite. R^2 provides an indication of the variance explained by the model. A value of 1 corresponds to a perfect fit of the model to the real data, a value of zero indicates a prediction as good as always predicting the mean value, and a negative value a worse prediction than choosing the mean (dummy predictor). The other metrics (MSE, MAE, MAD, sMAPE and RRMSE) are error metrics, they are always positive, with a value of zero corresponding to the best result.

The RRMSE and SMAPE will be considered as particularly important, since they calculate the ratio between the prediction error and a value related to the actual quantity to be predicted. Tables 4–6 provide the prediction metrics obtained under different scenarios with the proposed model (Section 3.2), all the results are based on the prediction of 6 future values ($p = 6$).

Table 4 gives the forecast metrics for each time ahead period. Each column provides the metrics for each period separately. The values are averages for all VF-DSs in the test set, i.e., the values in Table 4

Table 4
Forecast performance metrics for each time ahead period. The values are averages for all velocity-field data structures (VF-DSs) in the test set. The best value in each row is marked in bold.

	Time ahead prediction						
	T0	**T1**	**T2**	**T3**	**T4**	**T5**	***Average***
MSE	0.00015	**0.00015**	0.00018	0.00018	0.00021	0.00020	*0.00018*
MAE	0.00554	**0.00548**	0.00594	0.00596	0.00633	0.00620	*0.00591*
MAD	0.00159	**0.00155**	0.00178	0.00173	0.00176	0.00174	*0.00169*
R2	0.94750	**0.94793**	0.93852	0.93882	0.92758	0.93072	*0.93851*
SMAPE	1.04692	**1.03693**	1.12294	1.12642	1.18470	1.16433	*1.11371*
RRMSE	0.02487	**0.02477**	0.02691	0.02685	0.02921	0.02857	*0.02687*

Table 5
Forecast performance metrics considering different number of VF-DSs used for training (different size of the training set). The values are averages for all VF-DSs in the test set and the 6 predictions made per VF-DS. The best value in each row is marked in bold.

	Number of VF-DSs used for training				
	5813	**4443**	**3074**	**2046**	**1019**
MSE	0.00017	0.00018	**0.00017**	0.00021	0.00042
MAE	0.00571	0.00591	**0.00566**	0.00642	0.00931
MAD	0.00155	0.00169	**0.00153**	0.00183	0.00305
R2	0.94018	0.93851	**0.94318**	0.92932	0.85614
SMAPE	1.07541	1.11371	**1.06836**	1.21030	1.74598
RRMSE	0.02651	0.02687	**0.02584**	0.02870	0.04100

Table 6
Forecast performance metrics considering different number of predictors (time-slices) used per prediction. The values are averages for all VF-DSs in the test set and the 6 predictions made per VF-DS. The best value in each row is marked in bold.

	Number of time-slices used as predictors (*k*)				
	624	**300**	**100**	**60**	**10**
MSE	0.00035	0.00031	0.00032	0.00035	**0.00017**
MAE	0.00759	0.00694	0.00743	0.00762	**0.00571**
MAD	0.00151	**0.00142**	0.00167	0.00166	0.00155
R2	0.88084	0.88202	0.89410	0.88349	**0.94018**
SMAPE	1.39427	1.28302	1.37588	1.41089	**1.07541**
RRMSE	0.03764	0.03544	0.03588	0.03740	**0.02651**

correspond to an average of the prediction metrics for each time ahead period for the 2400 VF-DSs used for testing (Section 3.1). These values are obtained using 10 past time-slices ($k = 10$) as predictors, with no initial offset and 5813 VF-DSs used for training (Section 3.1). As expected, we can see that the best predictions are for the first two periods, but the metrics are not much reduced for the rest.

Table 5 offers the evolution of the forecast metrics when changing the number of consecutive VF-DSs used for training. The sequence of VF-DSs used for training always start from the beginning of the training set. The reduction in the number of training VF-DSs is done by selecting only the initial part of the training set. The values in Table 5 are averages for all VF-DSs in the test set and along the 6 predictions made per VF-DS, that means that the first column in Table 5 corresponds to the average of the rows in Table 4. These values are obtained using 10 past time-slices ($k = 10$) as predictors, with no initial offset (Section 3.1). We can see that the results remain quite similar until the number of elements used for training falls below 2000. This makes sense since this number of elements is quite a small number considering the prediction difficulties and considering that the first part of the training set represent the transient stage of the numerical simulation, where the dynamics is mixed-up with a large number of transient modes, introducing noise and masking the real solution, which is then presented in the saturated regime.

Table 6 presents the forecast metrics with different number of time-slices used as predictors (parameter k) (Section 3.1). The values in Table 6 are averages for all VF-DSs in the test set and the 6 predictions made per VF-DS. These values are obtained using 5813 training VF-DSs, with a number of test VF-DSs between 1785 and 2400, depending on the value of the parameter k. The results here are quite interesting since the best results are obtained with a very small k, indicating that a longer past period used to extract the predictors does not provide any improvement in the model's ability to make short-term predictions.

It is interesting to analyze the evolution of the forecast metrics for different VF-DSs along time and for different time ahead prediction periods. Figs. 7 and 8 provides information about this evolution for the first (T0) and last (T5) prediction periods, respectively. Fig. 6 presents one chart per metric with each graph showing the value of a forecast metric versus the time index of the VF-DS used for the test, that is, an index of zero corresponds to the first VF-DS extracted from the test set. The last index corresponds to the last VF-DS obtained from the test set. All test VF-DSs are extracted following the temporal sequence. It is important to appreciate the periodic nature of all charts, which is repeated every 624 time-slices similarly to the fundamental frequency of the simulations (St = 0.03). It is also possible to identify some other periodic events, whose fundamental frequencies are the high order harmonics of this fundamental frequency (i.e.: St = 0.06, 0.09…), which is in good agreement with the non-linear dynamics driving the flow (Le Clainche, 2019a; Le Clainche et al., 2017). Additionally, the time evolution of the charts is completely similar regardless of the time ahead period used for prediction, as can be seen from the similarities between the corresponding charts in Figs. 7 and 8. It is also important to mention that information about the fundamental frequency of 640 was not inserted into the model at any stage (training or validation), for example, into the number of time-slices used as predictors, which is 10, or the rest of hyperparameters used for training. This demonstrates that the model can follow fundamental flow properties in addition to achieving excellent predictions.

In order to provide a visual representation of the quality of predictions, in Figs. 9 and 10 are shown different velocity fields contour graphs for the prediction of the first and last time-ahead period. The figures present separate graphs for the streamwise and radial components of the velocity fields, and with the real and predicted fields adjacent to each other. Fig. 9 corresponds to a time-slice with high velocity values, representing the injection phase of the jet, and Fig. 10 to a different one with low values of velocity, representing the suction phase in the jet flow (see Fig. 1).

4.2. Comparison with alternative architectures

This Section provides a comparison of the proposed model with the different alternative deep learning architectures which were presented in Section 3.2 (Fig. 6 and Table 3), as well as a comparison with approaches that focus on the inference of a reduced order model (e.g., DMD), and a comparison in terms of computational resources between the flow solver and the proposed model.

Table 7 presents the comparison between the proposed model and the alternative deep learning architectures presented in Section 3.2. The

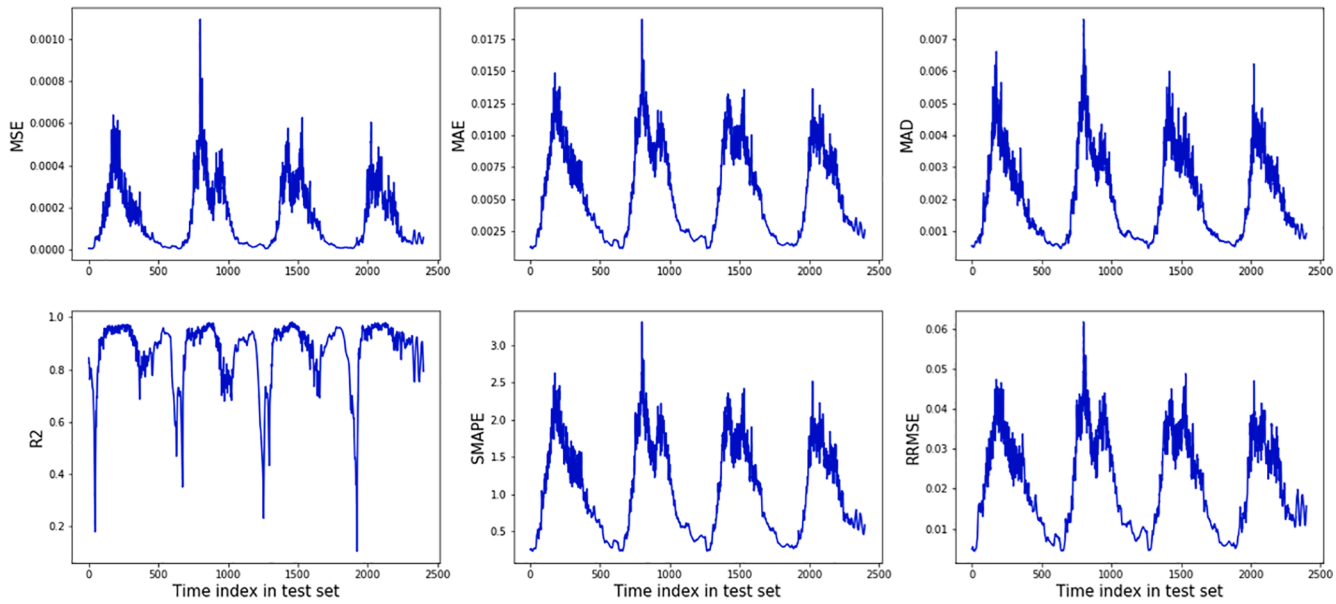

Fig. 7. Forecast metrics for the prediction of the first time-ahead period (T0). The charts present evolution of the predictions along the time index of the test set.

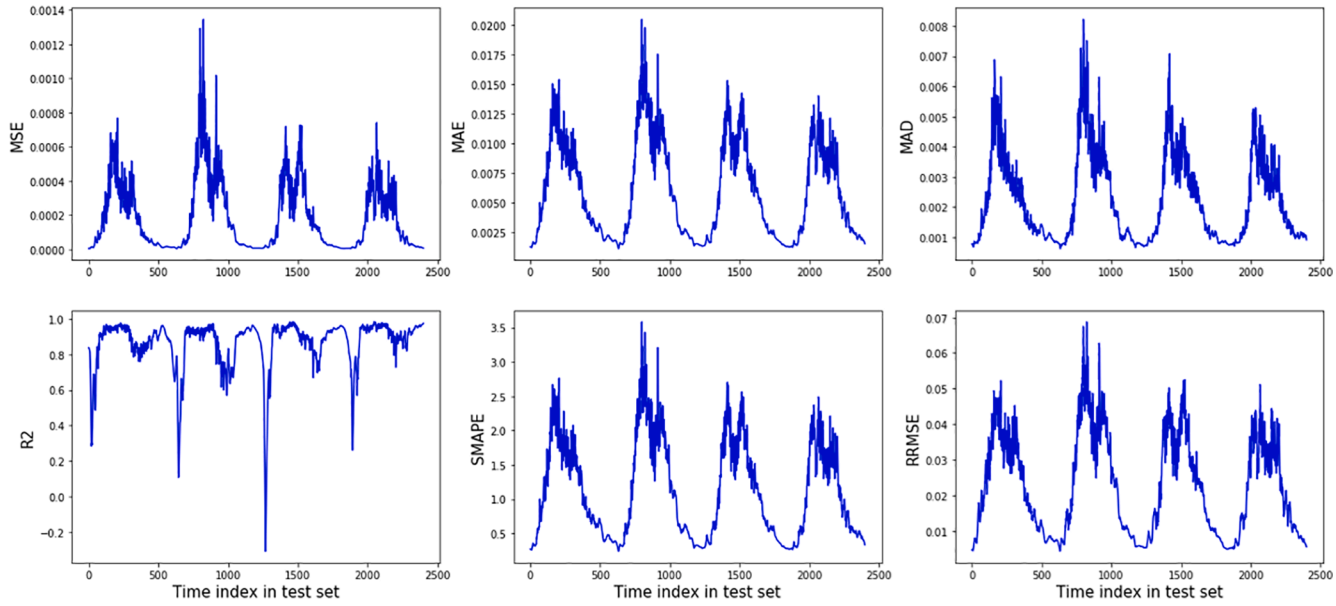

Fig. 8. Forecast metrics for the prediction of the last time-ahead period (T5). The charts present evolution of the predictions along the time index of the test set.

models are identified with a letter corresponding with the identifiers used in Fig. 6 and Table 3. The architecture A corresponds with the proposed model. We can observe how the proposed model presents the best prediction results in almost all metrics and with an ample difference. The training and prediction times are second best. With also the second smallest number of weights, considering that the objective is to have good generalization (small error in the test set) with as few parameters (weights) as possible to avoid overfitting.

As anticipated in the introduction, the proposed deep learning model is a pure predictive model and, as such, offers excellent predictive results in terms of forecast errors and time/resources required to perform model training and predictions. Other approaches that focus on the inference of a reduced order model and on understanding the intrinsic properties of fluid dynamics (e.g., DMD, POD) have more difficulties in the prediction task. In other words, creating a data-driven ROM based on DMD (Le Clainche, 2019a) using ~ 3100 time-slides in the training, the RRMS error of this predictions in the near and far fields are ~ 0.2 and ~ 0.3, respectively, while using the present deep learning model, this error is ~ 0.02 in the entire flow field (more than one order of magnitude smaller). Nevertheless, DMD or POD offer insight on the fluid behavior that deep learning models cannot provide. Thus, both approaches can be seen as complementary and both are valuable tools in the difficult problem of fluid dynamics analysis.

It is interesting to make a comparison of the time and computational resources required between the Nek5000 solver and the proposed model to generate the dataset used in this work: (a) The time required by the solver to generate the 13,696 time slices for the training and validation sets was $\sim$60 h, using 24 processors. (b) The time required to generate the 2416 time slices for the test set by the flow solver was $\sim$12 h using 24 processors (the time to generate this data set is smaller than the time required to generate the training data, because at this stage, the flow is saturated, in the first part of the training, the flow is transitory). (c) The

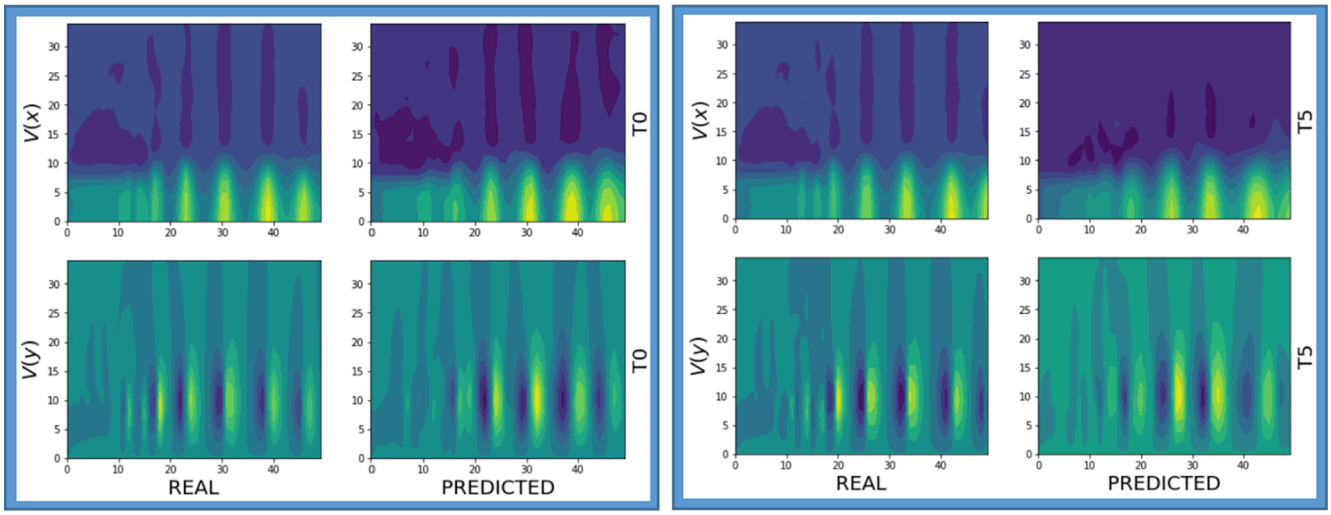

Fig. 9. Contour graphs of the components x and y of velocity (rows) for real and predicted velocity fields (columns) when the predictions are made for the first time-ahead period (T0) (**left part**) and last time-ahead period (T5) (**right part**). Both predictions correspond to the time index 300 of the test set, which has high velocity values.

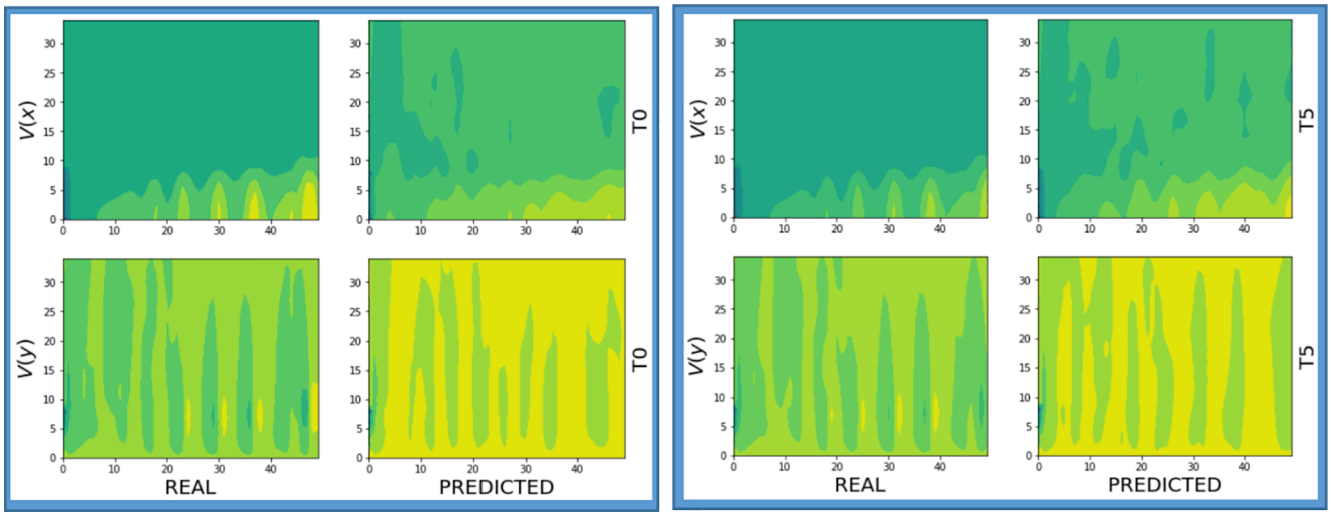

Fig. 10. Contour graphs of the components x and y of velocity (rows) for real and predicted velocity fields (columns) when the predictions are made for the first time-ahead period (T0) (**left part**) and last time-ahead period (T5) (**right part**). Both predictions correspond to the time index 400 of the test set, which has low velocity values.

Table 7
Comparison of the different architectures applied to the problem. The architectures are described in Table 3 and Fig. 6 and identified with a letter (A-F). The first architecture corresponds to the proposed model. The best value in each column is marked in bold.

Model Id.	Num Weights	Average Performance Metric						Training Time (min)	Test Time (min)
		MSE	MAE	MAD	R2	SMAPE	RRMSE		
A Proposed model*	1,711,861	**0.00018**	**0.00591**	0.00169	**0.93851**	**1.11371**	**0.02687**	14.045	0.090
B	**92,637**	0.00049	0.00940	0.00235	0.83225	1.77750	0.04398	16.253	**0.065**
C	8,191,280	0.00033	0.00750	0.00158	0.88711	1.38700	0.03647	17.090	0.155
D	25,822,420	0.00029	0.00692	**0.00151**	0.89963	1.27661	0.03439	24.346	0.200
E	5,510,080	0.00062	0.00997	0.00174	0.78900	1.84303	0.04986	**4.118**	0.135
F	12,389,980	0.00055	0.00970	0.00168	0.81169	1.78740	0.04711	15.344	0.137

time to train the neural network using 13,696 time slices for the training and validation sets was 14.045 min with one single processor (Table 7 and 8). (d) The time to make predictions with the neural network for the 2416 time slices of the test set was 0.090 min with one single processor (Table 7 and 8). Considering these results, the time and cost savings that occur when using the proposed model are clear, compared to continuing to use the solver to generate the predictions.

4.3. Results with different flows

The proposed model was chosen based on the good results obtained with the dataset shown in Section 3.1, but the spirit of the work was that the model were useful for any type of flow, based on the implicit hypothesis that considers that intrinsically the methods applied for video prediction (which are currently a challenging and difficult problem) using deep learning could be extended to CFD. In this regard, this work

Table 8
(Cylinder configuration) Forecast performance metrics for each time ahead period. The values are averages for all velocity-field data structures (VF-DSs) in the test set.

	Time ahead prediction						
	T0	**T1**	**T2**	**T3**	**T4**	**T5**	***Average***
MSE	0.00001	0.00002	0.00002	0.00002	0.00001	0.00002	0.00002
MAE	0.00261	0.00367	0.00346	0.00308	0.00253	0.00384	0.00320
MAD	0.00203	0.00334	0.00279	0.00244	0.00193	0.00347	0.00267
R2	0.99982	0.99970	0.99969	0.99974	0.99982	0.99965	0.99974
SMAPE	0.76248	1.00708	0.92370	0.85607	0.69621	1.01106	0.87610
RRMSE	0.00572	0.00732	0.00732	0.00666	0.00560	0.00767	0.00672

and similar works show that this hypothesis can be considered as plausible, but we realize that it is difficult to demonstrate it in a physically principled way since most of the results obtained by a neural network are empirical. Nevertheless, and in order to further sustain this hypothesis, we have applied the same proposed model (Section 3.1) to the prediction of the x and y velocity-fields components of a different type of flow.

The new flow corresponds to the two-dimensional wake of a circular cylinder at Re = 100 (defined as in Section 3.1, Re $= \frac{UD}{\nu}$, where U and D now correspond to the free stream velocity and to the cylinder diameter). At these flow conditions, the solution is periodic (see more details in Barkley & Henderson, 1996). As for the synthetic jet case, numerical simulations have been carried out using the solver Nek5000 (Fischer et al., n.d.) to solve the incompressible non-dimensional form of Navier-Stokes Eqs. (1) and (2). The dimensions of the computational domain are defined as in the literature (Barkley & Henderson, 1996), with non-dimensional sizes in the normal direction $L_y = \pm 15D$, and in the streamwise direction, $L_x = 15D$ upstream the cylinder and $L_x = 50D$ downstream the cylinder. The computational domain is formed by 600 rectangular elements, each one discretized using the polynomial order $\Pi = 9$. A grid independence study has been previously carried out to validate the solution presented with the literature. A set of 150 snapshots, equi-spaced in time with time interval 0.2, is collected when the flow is saturated (simulation converged in time), to perform the present test.

The new dataset used for this flow is quite different to the original dataset used in this work (Section 3.1). It is much smaller with 152 time-slices divided into 97, 24 and 30 time-slices for training, validation and test, respectively. Which corresponds to a fraction of 64,2%, 15,8% and 20% for the training, validation and test sets. The values for the parameters k, p and batch size (bs) have been maintained to 10, 6 and 5, respectively. To train the model we have also used 70 epochs with early-stopping and a patience value of 10 epochs. With all these configuration parameters, we have obtained the prediction performance metrics shown in Table 8 with a visual indication on the prediction quality shown in Fig. 11 using contour graphs of the x and y velocity-fields for the time-ahead prediction corresponding to the first and last predicted time-slice.

From the results in Table 8, we can conclude that the model provides excellent prediction results for this alternative flow. The results are even better than those presented in Table 4 for the dataset in Section 3.1. This can be justified considering that in this case the flow is less complex than in the synthetic jet. Although both test cases represent a periodic solution, the synthetic jet presents a greater spatial flow complexity, as previously introduced in Fig. 1, where it was possible to distinguish two completely different topology patterns that were periodically changing in the flow. On the contrary, in the circular cylinder, it is possible to only identify the periodic oscillations of the cylinder wake, as presented in Fig. 11, that shows the streamwise and normal velocity fields at two different time instants.

4.4. Training robustness

It is interesting to show that the proposed model is robust under different random weight initialization in order to ensure that the good performance results of the model are robust and not due to a lucky random outcome. Table 9 presents the average performance results for 10 training runs of the proposed model with random weight initializations. The table also presents the average performance for the 10 runs with the standard deviation and coefficient of variation (CV) which is the standard deviation divided by the mean. We can see how the results are quite stable with a CV of less than 10% for the most important metrics, with training time being the most variable result due to the early-stopping criteria used to adjust training. Early stopping is random

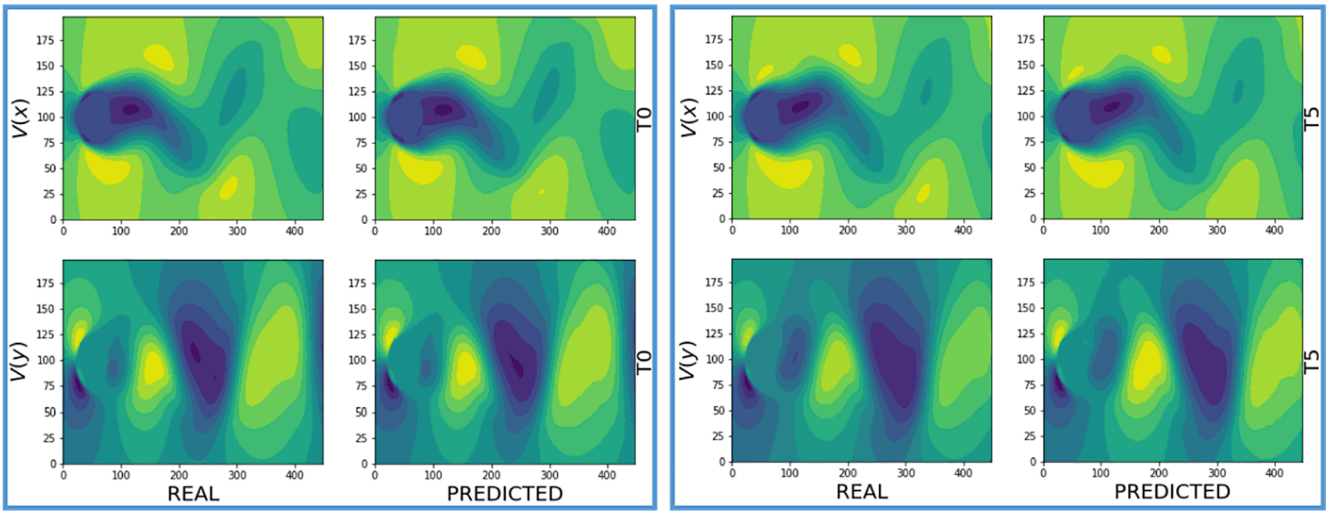

Fig. 11. (Cylinder configuration) Contour graphs of the components x and y of velocity (rows) for real and predicted velocity fields (columns) when the predictions are made for the first time-ahead period (T0) (**left part**) and last time-ahead period (T5) (**right part**).

Table 9
Different results for different training runs (10) with random initialization of weights for the architecture of the proposed model (Table 2). At the bottom of the table are the mean (column-wise) and standard deviation (column-wise) and coefficient of variation (CV) which is the standard deviation (Std Dev) divided by the mean.

RUN Id.	MSE	MAE	MAD	R2	SMAPE	RRMSE	Training Time (min)	Test Time (min)
RUN #1	0.00016	0.00566	0.00160	0.94408	1.07018	0.02565	11.661	0.094
RUN #2	0.00014	0.00504	0.00135	0.95263	0.94983	0.02361	23.161	0.115
RUN #3	0.00021	0.00631	0.00173	0.92879	1.18991	0.02880	9.223	0.106
RUN #4	0.00018	0.00573	0.00148	0.93822	1.08080	0.02682	9.880	0.091
RUN #5	0.00021	0.00651	0.00188	0.92782	1.22668	0.02907	6.940	0.097
RUN #6	0.00015	0.00528	0.00142	0.94888	0.99318	0.02447	15.707	0.104
RUN #7	0.00017	0.00559	0.00151	0.94263	1.05066	0.02595	10.216	0.098
RUN #8	0.00023	0.00669	0.00184	0.92070	1.25636	0.03055	6.624	0.099
RUN #9	0.00020	0.00610	0.00162	0.93083	1.13654	0.02847	8.945	0.092
RUN #10	0.00017	0.00566	0.00159	0.94309	1.06791	0.02589	11.239	0.098
Mean	*0.00018*	*0.00586*	*0.00160*	*0.93777*	*1.10220*	*0.02693*	*11.360*	*0.100*
Std Dev	*3.005E-05*	*5.337E-04*	*1.748E-04*	*1.030E-02*	*9.919E-02*	*2.219E-03*	*4.878*	*7.158E-03*
CV (%)	*16.57%*	*9.11%*	*10.92%*	*1.10%*	*9.00%*	*8.24%*	*42.94%*	*7.19%*

in nature and produces quite different training times depending on the time evolution of the validation loss.

Fig. 12 shows the evolution of the MSE loss for the train set (in solid-blue) and validation set (in dotted-green) during training for the proposed model with the two datasets used for the experiments: the synthetic jet dataset (Section 3.1) (Fig. 12, left) and the cylinder dataset (Section 4.3) (Fig. 12, right). We observe in Fig. 12 the need for at least 40–50 epochs to achieve convergence. In both cases, we can observe a smooth training with different evolutions depending on the dataset but arriving, in both cases, to a stable training. The fewer number of training samples adds an initial difficulty to the training for the second dataset with a final convergence.

We implemented the deep learning models in python using Tensorflow/Keras (Abadi et al., 2016) and the scikit-learn python package (Pedregosa et al., 2011) to calculate the performance metrics. To perform the computation, a Linux-Ubuntu system with 25 GB of RAM, Intel Xeon 2.3 GHz and GPU was used.

5. Conclusion

This article introduces a novel application of machine learning to fluid dynamics. Deep neural networks have been used to predict the evolution of the velocity field in a complex flow. More specifically, these algorithms have been applied to analyze the flow modeling a synthetic jet in transitional regime. Synthetic jets are complex flows formed by the periodic oscillation of a piston or membrane in a cavity, which is periodically forcing to leave and re-enter the flow into the cavity through a jet nozzle. Hence, the net mass flux of the jet is zero, but the streamwise mean momentum is not zero, bringing these devices to be useful in a wide range of industrial and natural applications: fluid mixing, heat transfer enhancement, flow control, natural propulsion systems (modeling the swimming motion of marine animals). The complexity and multiple applications of this flow makes it a suitable and interesting example to test the performance of machine learning for temporal forecasting in fluid dynamics. The dataset analyzed has been obtained numerically and contains information regarding the transient and saturated region of a numerical simulation.

Deep neural networks are presented as a new powerful tool for data-driven system identification. We propose a novel architecture based on 3D convolutional layers specifically designed to accommodate the learning needs to predict future velocity fields of a complex fluid flow. The architecture is based on the hypothesis that a low-dimensional intermediate representation could be used as the basis for all k-ahead velocity fields prediction. This inductive bias has been introduced considering the good results obtained by low-rank approximation solutions (e.g., DMD, POD….).

The proposed model is analyzed in detail, providing prediction performance metrics from different points of view, showing that the model is suitable for providing SOTA prediction results. The model offers an average symmetric mean absolute error (sMAPE) and a relative root mean square error (RRMSE) of 1.114 and 0.027 respectively (one order of magnitude improvement over low-rank approximation tools), using 10 past samples and predicting 6 future samples of a two-dimensional velocity field on a 35x50 point matrix associated to a synthetic jet dataset. Additionally, to validate the present model, another example has been presented to predict the two dimensional wake of a circular cylinder at Re = 100, with excellent results as well (sMAPE: 0.876 and RRMSE: 0.007) which helps to sustain its applicability to different types of flow.

This work is also a proof of concept to assess the suitability of deep

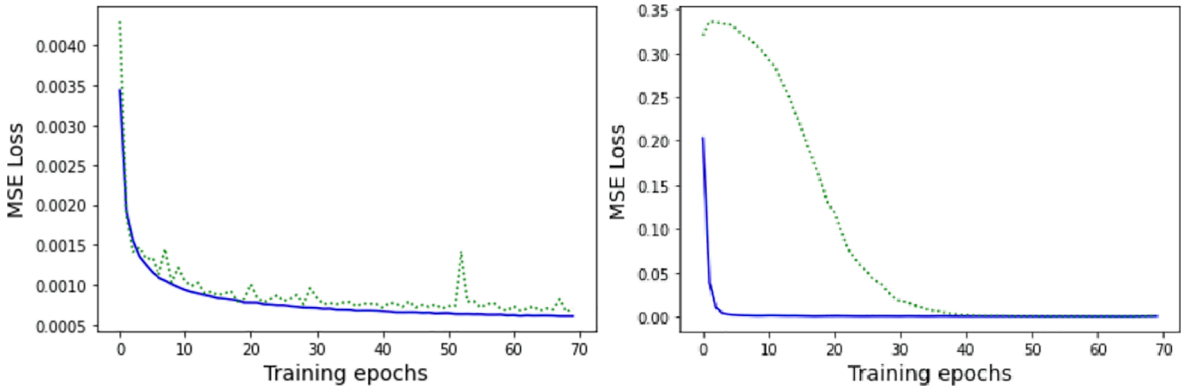

Fig. 12. Evolution of the MSE loss for the train set (solid-blue) and validation set (dotted-green) during training for the proposed model applied to two different datasets: synthetic jet dataset (**Left chart**) and cylinder dataset (**Right chart**).

learning models to make predictions about complex high-dimensional time-series data. In particular, we show that a 3D convolutional network together with an architecture that exploits a low-dimensional intermediate representation is suitable for this task and opens up interesting research opportunities for other areas that also operate with complex and high-dimensional time-series data: future frame video prediction, network traffic forecasting, network intrusion detection. This allows us to validate the algorithms and methods developed from previous research activities on network intrusion detection, extending the scope of our research to other fields such as fluid dynamics.

As future lines of work, it would be highly interesting to continue with this line of research, particularly in the areas of generative models and creation of synthetic jet flows, while preserving their fundamental physical properties (Brunton et al., 2020) and to explore ensemble models for multivariate time-series forecasting (Lopez-Martin et al., 2019). It is also important to further investigate the behavior of the model in fully turbulent flows, or under a full range of different flow types, which remains as an open topic for future research.

CRediT authorship contribution statement

Manuel Lopez-Martin: Investigation, Methodology, Software, Writing - original draft. **Soledad Le Clainche:** Investigation, Formal analysis, Data curation, Writing - original draft. **Belen Carro:** Funding acquisition, Resources, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

The preparation of the article and the study of algorithms were funded with grant RTI2018-098958-B-I00 from Proyectos de I + D + i «Retos investigación», Programa Estatal de I + D + i Orientada a los Retos de la Sociedad, Plan Estatal de Investigación Científica, Técnica y de Innovación 2017–2020. Spanish Ministry for Science, Innovation and Universities; the Agencia Estatal de Investigación (AEI) and the Fondo Europeo de Desarrollo Regional (FEDER).

Appendix A. Supplementary data

Supplementary data to this article can be found online at https://doi.org/10.1016/j.eswa.2021.114924.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo. E., Chen Z., Citro. C., et al. (2016). TensorFlow: Large-scale machine learning on heterogeneous distributed systems. arXiv: 1603.04467v2 [cs.DC].

Agrawal, S., Barrington, L., Bromberg, C., Burge, J., Gazen, C., Hickey, J. (2019). Machine Learning for Precipitation Nowcasting from Radar Images, arXiv: 1912.12132v1 [cs.CV] 11 Dec 2019.

Borchani, H., Varando, G., Bielza, C., & Larrañaga, P. (2015). A survey on multi-output regression. *Data Mining and Knowledge Discovery., 5*(5), 216–233. https://doi.org/10.1002/widm.1157

Brunton, S.L., Noack, B.R. & Koumoutsakos, P. (2020) Machine Learning for Fluid Mechanics. arXiv:1905.11075v3 [physics.flu-dyn] 4 Jan 2020.

Barkley, D., & Henderson, R. D. (1996). Three-dimensional Floquet stability analysis of the wake of a circular cylinder. *Journal of Fluid Mechanics, 322*, 215–241. https://doi.org/10.1017/S0022112096002777

Cai, S., Zhou, S., Xu, C., & Gao, Q. (2019). Dense motion estimation of particle images via a convolutional neural network. *Experiments in Fluids, 60*, 73. https://doi.org/10.1007/s00348-019-2717-2

Cai, S., Lian, J., Gao, Q., Xu, C., & Wei, R. (2019). Particle image velocimetry based on a deep learning motion estimator. *IEEE Transactions on Instrumentation and Measurement*. https://doi.org/10.1109/TIM.2019.2932649

Carter, J. E., & Soria, J. (2002). The evolution of round zero-net-mass-flux jets. *Journal of Fluid Mechanics, 472*, 167–200.

Castelló, J. S. (2018). *A Comprehensive survey on deep future frame video prediction.* Universitat de Barcelona.

Cattafesta, L. M., & Sheplak, M. (2010). Actuators for active flow control. *Annual Review of Fluid Mechanics, 43*, 247–272.

Chinesta, F., Keunings, R., & Leygue, A. (2014). *The Proper Generalized Decomposition for Advanced Numerical Simulations, Springer Briefs in Applied Sciences and Technology.* Berlin: Springer-Verlag.

Cohen, N., Sharir, O., Levine, Y., Tamari, R., Yakira, D., & Shashua, A. (2017). Analysis and Design of Convolutional Networks via Hierarchical Tensor Decompositions. Retrieved from http://arxiv.org/abs/1705.02302.

Cohen, N., & Shashua, A. (2016). Inductive Bias of Deep Convolutional Networks through Pooling Geometry. Retrieved from http://arxiv.org/abs/1605.06743.

Dargan, S., Kumar, M., Ayyagari, M. R., & Kumar, G. (2019). A survey of deep learning and its applications: A new paradigm to machine learning. *Archives of Computational Methods in Engineering*. https://doi.org/10.1007/s11831-019-09344-w

DeMont, E., & Gosline, J. (1998). Mechanics of jet propulsion in the hydromedusan jellyfish, polyorchis penicillatus. *The Journal of Experimental Biology, 143*, 347–361.

Ferrari, M., Werner, G. S., Bahrmann, P., Richartz, B. M., & Figulla, H. R. (2006). Turbulent flow as a cause for underestimating coronary flow reserve measured by Doppler guide wire. *Cardiovascular Ultrasound, 14*(4). https://doi.org/10.1186/1476-7120-4-14

Fischer, P.F., Lottes, J.W. & Kerkemeier, S.G., Nek5000. Available online: <https://nek5000.mcs.anl.gov> (Accessed January 2021).

Gao, C., Zhang, W., Kou, J., Liu, Y., & Ye, Z. (2017). Active control of transonic buffet flow. *Journal of Fluid Mechanics, 824*, 312–351.

Glezer, A., & Amitay, M. (2002). Synthetic jets. *Annual Review of Fluid Mechanics, 34*, 503–529.

Huang, C., Chiang, C., & Li, Q. (2017). A study of deep learning networks on mobile traffic forecasting. In *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)* (pp. 1–6). https://doi.org/10.1109/PIMRC.2017.8292737

Hyndman, R. J., & Koehler, A. B. (2006). Another look at measures of forecast accuracy. *International Journal of Forecasting, 22*(4), 679–688. https://doi.org/10.1016/j.ijforecast.2006.03.001

Ioffe S. & Szegedy C., (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift.. https://arxiv.org/abs/1502.03167.

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. arXiv: 1412.6980 [cs.LG].

Kutz, J. (2017). Deep learning in fluid dynamics. *Journal of Fluid Mechanics, 814*, 1–4. https://doi.org/10.1017/jfm.2016.803

Le Clainche, S., & Vega, J. M. (2017). Higher order dynamic mode decomposition to identify and extrapolate flow patterns. *Physics of Fluids, 29*, 08412.

Le Clainche, S., Vega, J. M., & Soria, J. (2017). Higher order dynamic mode decomposition for noisy experimental data: Flow structures on a zero-net-mass-flux jet. *Experimental Thermal and Fluid Science, 88*, 336–353.

Le Clainche, S., Varas, F., & Vega, J. M. (2017). Accelerating reservoir simulations using POD on the fly. *International Journal for Numerical Methods in Engineering, 28*, 79–100.

Le Clainche, S., & Ferrer, E. (2018). A reduced order model to predict transient flows around straight bladed vertical axis wind turbines. *Energies, 11*, 566–578.

Le Clainche, S. (2019a). Prediction of the optimal vortex in synthetic jets. *Energies, 12*, 1635. https://doi.org/10.3390/en12091635

Le Clainche, S. (2019b). *An introduction to some methods for soft computing in fluid dynamics, SOCO 2019-advances in intelligent systems and computing* (pp. 557–566). Springer. https://doi.org/10.1007/978-3-030-20055-8_53

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature, 521*(7553), 436–444. https://doi.org/10.1038/nature14539

Lee, C. -Y., Gallagher, P. W., & Tu, Z. (2015). Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. https://arxiv.org/abs/1509.08985.

Lee, Y., Yang, H., & Yin, Z. (2017). PIV-DCNN: Cascaded deep convolutional neural networks for particle image velocimetry. *Experiments in Fluids, 58*, 171. https://doi.org/10.1007/s00348-017-2456-1

Lopez-Martin, M., Carro, B., Sanchez-Esguevillas, A., & Lloret, J. (2017). Network traffic classifier with convolutional and recurrent neural networks for internet of things. *IEEE Access, 5*, 18042–18050. https://doi.org/10.1109/ACCESS.2017.2747560

Lopez-Martin, M., Carro, B., & Sanchez-Esguevillas, A. (2019). Neural network architecture based on gradient boosting for IoT traffic prediction. *Future Generation Computer Systems, 100*, 656–673. https://doi.org/10.1016/j.future.2019.05.060

Lopez-Martin, M., Nevado, A., & Carro, B. (2020). Detection of early stages of Alzheimer's disease based on MEG activity with a randomized convolutional neural network. *Artificial Intelligence in Medicine, 107*, Article 101924. https://doi.org/10.1016/j.artmed.2020.101924

Lu, S., Wang, S.-H., & Zhang, Y.-D. (2020). Detection of abnormal brain in MRI via improved AlexNet and ELM optimized by chaotic bat algorithm. *Neural Computing and Applications*. https://doi.org/10.1007/s00521-020-05082-4

Luchtenburg, D. M., Noack, B. R., & Schlegel, M. (2009). *An introduction to the POD Galerkin method for fluid flows with analytical examples and MATLAB source codes.* Berlin Institute of Technology. Report 01.

Lusch B., Kutz J.N. &, Brunton S.L. (2017) Deep learning for universal linear embeddings of nonlinear dynamics. arXiv:1712.09707 [math.DS].

Marusic, I., Candler, G. V., Interrante, V., Subbareddy, P. K., & Moss, A. (2003). Real time feature extraction for the analysis of turbulent flows. *Semantic Scholar*. https://doi.org/10.1007/978-1-4615-1733-7-13

Mathieu M., Couprie C. & LeCun Y., (2016). Deep multi-scale video prediction beyond mean square error. arXiv:1511.05440v6 [cs.LG] 26 Feb 2016.

Mitchell, B. R., 2017. The spatial inductive bias of deep learning. Johns Hopkins University. Retrieved from <https://dspace-prod.mse.jhu.edu/handle/1774.2/40864>. (Accessed January 2021).

Noack, B. R., Morzynski, M., & Tadmor, G. (2011). *Reduced-order modelling for flow control*. New York: Springer.

Ohlsson, J., Schlatter, P., Fischer, P. F., & Henningson, D. S. (2010). Direct numerical simulation of separated flow in a three-dimensional diffuser. *Journal of Fluid Mechanics, 650*, 307–381.

Park, K. H., Jun, S. O., Baek, S. M., Cho, M. H., Yee, K. J., & Lee, D. H. (2013). Reduced-order model with an artificial neural network for aerostructural design optimization. *Journal of Aircraft, 50*, 1106–1116. https://doi.org/10.2514/1.C032062

Pathak, J., Hunt, B., Girvan, M., Lu, Z., & Ott, E. (2018). Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical Review Letters, 120*, Article 024102. https://doi.org/10.1103/PhysRevLett.120.024102

Pavlova, A., & Amitay, M. (2006). Electronic cooling using synthetic jet impingement. *Journal of Heat Transfer, 128*, 897–907.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., et al. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research, 12*, 2825–2830. arXiv:1201.0490 [cs.LG].

Quarteroni, A., Manzoni, A., & Negri, F. (2016). *Reduced basis methods for partial differential equations*. Springer.

Rawat, W., & Wang, Z. (2017). Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation, 29*(9), 2352–2449.

Scher, S. (2018). Toward data-driven weather and climate forecasting: Approximating a simple general circulation model with deep learning. *Geophysical Research Letters, 45*, 12616–12622. https://doi.org/10.1029/2018GL080704

Scher, S., & Messori, G. (2019). Weather and climate forecasting with neural networks: Using general circulation models (GCMs) with different complexity as a study Ground. *Geoscientific Model Development, 12*(7), 2797–2809. https://doi.org/10.5194/gmd-12-2797-2019

Schmid, P. (2010). Dynamic mode decomposition of numerical and experimental data. *Journal of Fluid Mechanics, 656*, 5–28.

Sirovich, L. (1987). Turbulence and the dynamics of coherent structures. *Quarterly of Applied Mathematics, XLV*, 561–590.

Vlachas, P. R., Byeon, W., Wan, Z.Y., Sapsis, T.P., & Koumoutsakos. P. (2019). Data-Driven forecasting of high-dimensional chaotic systems with long-short term memory networks. arXiv:1802.07486v5 [physics.comp-ph] 19 Sep 2019.

Vukotic V., Pintea, S. -L., Raymond, C., Gravier, G., & Gemert, J. V. (2017). One-Step Time-Dependent Future Video Frame Prediction with a Convolutional Encoder-Decoder Neural Network, arXiv:1702.04125 [cs.CV].

Wan, Z. Y., Vlachas, P., Koumoutsakos, P., & Sapsis, T. (2018). Data-assisted reduced-order modeling of extreme events in complex dynamical systems. *PLoS ONE, 13*(5), Article e0197704. https://doi.org/10.1371/journal.pone.0197704

Wang, H., & Menon, S. (2001). Fuel-air mixing enhancement by synthetic microjets. *AIAA Journal, 39*, 2308–2319.

Wang, J., Balaprakash, P., & Kotamarthi, R. (2019). Fast domain-aware neural network emulation of a planetary boundary layer parameterization in a numerical weather forecast model. *Geoscientific Model Development, 12*(10), 4261. https://doi.org/10.5194/gmd-12-4261-2019

Wang, S.-H., & Zhang, Y.-D. (2020). DenseNet-201-based deep neural network with composite learning factor and precomputation for multiple sclerosis classification. *ACM Transactions on Multimedia Computing, Communications, and Applications, 16*(2s). https://doi.org/10.1145/3341095

White C., Ushizima D. & Farhat C. (2019) Fast Neural Network Predictions from Constrained Aerodynamics Datasets. arXiv:1902.00091 [physics.comp-ph].

Wiewel S., Becher M. & Thuerey N. (2019) Latent-space Physics: Towards Learning the Temporal Evolution of Fluid Flow. arXiv:1802.10123 [cs.LG].

Wojna, Z., Ferrari, V., Guadarrama, S., Silberman, N., Chen, L.-C., Fathi, A., & Uijlings, J. (2017). The Devil is in the Decoder: Classification, Regression and GANs. Retrieved from http://arxiv.org/abs/1707.05847.

Xiaoxiao, G., Wei, L., & Iorio, F. (2016). Convolutional neural networks for steady flow approximation. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16)* (pp. 481–490). New York, NY, USA: ACM. https://doi.org/10.1145/2939672.2939738.

Zong, H., & Kotsonis, M. (2018). Formation, evolution and scaling of plasma synthetic jets. *Journal of Fluid Mechanics, 837*, 147–181.