Expert Systems with Applications Volume 177, 2021, 114924. https://doi.org/10.1016/j.eswa.2021.114924

Preprint

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

Manuel Lopez-Martin, Soledad Le Clainche, Belen Carro

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 jets dataset.

Index Terms— Computational fluid dynamics; prediction; deep learning; convolutional neural network

* Correspondence: mlopezm@acm.org; Tel.: +34-983-423-980, Fax: +34-983-423-667
The authors declare that there is no conflict of interest regarding the publication of this paper

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. For instance, in nature it is possible to find complex flows describing the wake of flying insects, the movements of some marine animals (Le Clainche, 2019a), the movement of the blood in vessels…. Some examples in the field of engineering

M. Lopez and B. Carro are with the Dpto. TSyCeIT, ETSIT, Universidad de Valladolid, Paseo de Belén 15, Valladolid 47011, Spain; (mlopezm@acm.org).
S. Le Clainche is with the ETSI Aeronáutica y del Espacio, Universidad Politécnica de Madrid, Pl. Cardenal Cisneros 3, 28040, Spain; (soledad.leclainche@upm.es)

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, the pollution and the cost (Marusic et al., 2003). In biological flows, such as blood, turbulence occurs in pathological situations (i.e.: medical implants), triggering negative biological responses such as coronary artery diseases (Ferrari et al., 2006). 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 this reason, 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 modelling 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 $Re^{37/14}$, 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 modelling the 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, Zhang, Kou, Liu & Ye, 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,..) and where the training 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, Bengio & Hinton, 2015). These algorithms are lately applied to a vast number of problems with extremely good results in most of the cases (Dargan et al, 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 analyze 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 shared by the different k-ahead predictions. This embedding is used by the last layers as a common input to create each of the particular predictions. 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, which shows 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 at reducing the computational cost in numerical simulations. In 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 for the first time in this work (to the authors knowledge) 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, this model can achieve excellent prediction performance metrics for the k-ahead predictions of this high-dimensional spatio-temporal problem using a very reduced number of past samples.

The contributions of this work are:

- Novel application of deep learning techniques to the fluid dynamics field
- The proposed model can achieve excellent prediction performance metrics for the k-ahead predictions of a high-dimensional spatio-temporal problem using a very reduced number of past samples.
- It is not based on a prior physical model
- The training and prediction phases require few computational resources compared to other alternative techniques (e.g. DMD, POD)

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, Noack & Koumoutsakos, 2020), although it is recognized that their application is discreet compared to other fields and considering the great promise of benefits suggested by CFD experts (Kutz, 2017; Brunton, Noack & Koumoutsakos, 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, Noack & Koumoutsakos (2020) presents 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 et al. (2018) proposes 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 et al., 2019a; Lee, Yang & Yin, 2017; Cai et al., 2019b) 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 & Iorio (2016) provides a velocity field predictor for steady flows, using a CNN model with an encoding/decoding configuration. White, Ushizima & Farhat (2019) proposes a cluster network to perform simulations in fluid dynamics, the model requires extensive hyper-parameter search and tuning, but is faster than alternative methods based on Gaussian processes. Likewise, Vlachas (2019) offers 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 & Sapsis (2018) presents a solution to model complex dynamical systems under extreme events, using a recurrent neural network (RNN) to help improve a reduced-order model in locations where data is 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, Kutz & Brunton (2017) proposes a generalization of Koopman representation in a linear embedding using a modified deep auto-encoder, the intention is to maintain the physical interpretability of the Koopman approach with the higher efficiency of a deep neural network. All the previous works apply several deep learning

models to fluid dynamics estimation or prediction tasks, but not with the architecture (3D CNN with a low-dimensional intermediate latent space) and objectives (k-ahead velocity-field prediction for a synthetic jet) presented in this work.

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. Most solutions incorporate LSTM networks with only a few proposing exclusively convolutional networks (Mathieu, Couprie & LeCun, 2016; Vukotic, 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 & Li (2017) and Lopez-Martin, Carro & 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 & 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 & Kotamarthi (2019) proposes an alternative to physics-based predictions using an ad-hoc deep learning architecture with fully connected layers. Agrawal et al. (2019) is 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 are comparable or better than SOTA models for short-term predictions.

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. Two main parameters characterize this type of flow, the Reynolds number, defined as $\mathrm{Re}=\frac{UD}{\nu}$, and the Strouhal number, defined as $\mathrm{St}=\frac{fD}{U}$, where U, D, ν and f represent the jet mean momentum velocity (Le Clainche, 2019a), the diameter of the jet nozzle, the kinematic viscosity of the fluid and the piston (or membrane) oscillation frequency, respectively. This article analyses the numerical dataset generated and analysed in (Le Clainche, 2019a), for Re=1000 and St=0.03, with D=1 and U=1. This database has been generated using the solver Nek5000 (Fischer, Lottes & Kerkemeier, n.d.), solving the non-linear incompressible form of Navier-Stokes equations. The solver uses as spatial discretization spectral elements with Gauss-Lobatto-Legendre points of polynomial order Π. 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 (Re=1000, St=0.03) the flow is laminar in the near field, but it transitions to turbulence in the far field.

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.

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. Right: saddle point (S).

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.

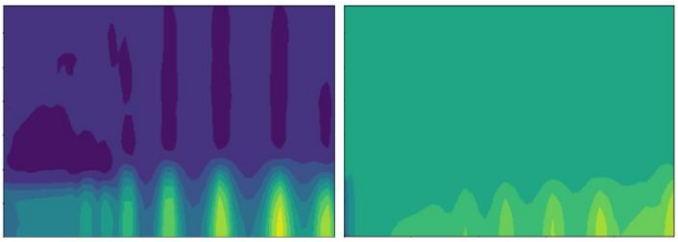

Fig. 2. Contour lines representation of the velocity fields shown in Fig 1 for the two topology patterns: vortex rings (A) and saddle point (B). In both cases, only the streamwise component of the velocity field is shown.

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 x 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 16112 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 11642 time-slices are reserved to train the model, with the following 2054 and 2416 time-slices reserved as validation and test sets, respectively. All the performance metrics of the prediction results 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 surface was performed by taking one of every two consecutive spatial points in both directions (x and y), ending in a surface of 35 x 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.

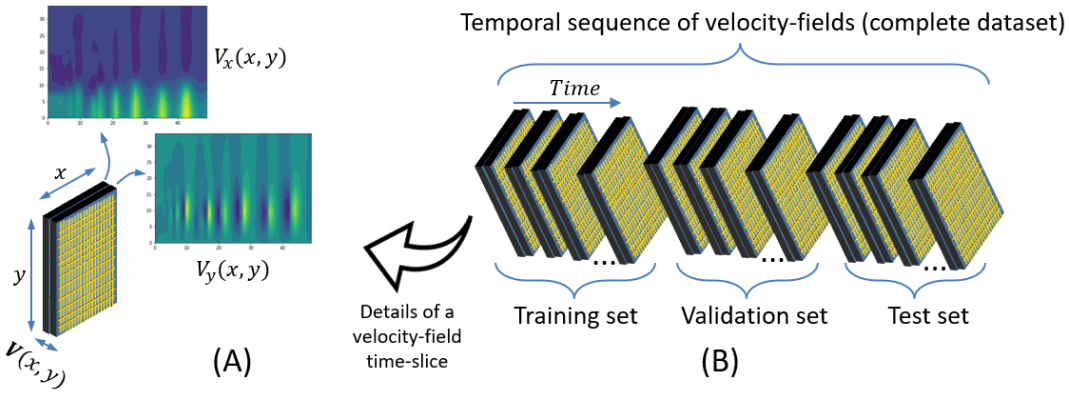

Fig. 3. Data structure to perform the experiments. (A) Details of a 3-dimensional velocity-field time-slice consisting on a surface with the velocity components x and y as parts of the third dimension. (B) Velocity-field time-slices following their temporal sequence. This sequence

forms the complete dataset that is divided along the time dimension into training, validation and test sets.

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 are 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 period and the rolling-window advance can be controlled with a jump length (stride) between consecutive VF-DSs. The initial period 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 period 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. Taking these alternatives into account, the results presented in Section 4 have been obtained with an initial period of 0 (which means we include the transient stage in the training set), a stride value for the training VF-DSs of 2, and a stride value of 1 for the validation and test VF-DSs. The final number of VF-DSs for training, validation and testing are 5813, 2038 and 2400 respectively, assuming that we establish values of 10, 6, 0, 2, 1 and 1 for the parameters: k, p, initial period, stride for the training set, stride for the validation set and stride for the test set, respectively.

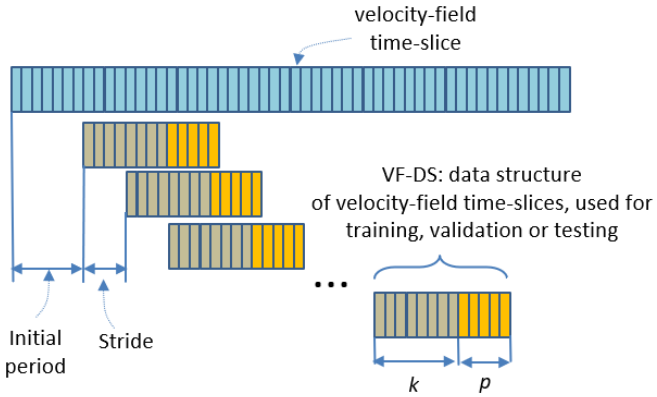

Fig. 4. Arrangement of time-slices of velocity-field into a velocity-field data structures (VF-DS) used for training, validation or testing.

3.2 Model description

The proposed model to perform velocity-field predictions based on the dataset described in section 3.1 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_s). 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, and d) avoid an excessive number of weights (evade overfitting) while maintaining the required model complexity.

The model (Fig 5) is based on a multilayer deep neural network that receives a sequence of *k* velocity-field ($\boldsymbol{V}(x, y)$) 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 x 1 x 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-slides to be predicted. The tensor reshape 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 tensor split 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 the first FC layers (*p* of them) 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 35000 which are finally reshaped into *p* predicted velocity-field ($\widehat{\boldsymbol{V}}(\boldsymbol{x}, \boldsymbol{y})$) time-slices.

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 which are later differentiated with the second FC layer.

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

The details of the model are provided in Table I. The layer configuration in Table I corresponds to the best model whose results are presented in section 4. First column in Table I is the layers order, second column offers the layers details and last column gives the tensor dimensions for the output of each layer. The parameters *k* and *p* appear explicitly in the tensor dimensions 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 stopping was also used if the last 10 epochs do not reduce the loss function on the validation set. The layers description in Table I follows the conventions in (Lopez-Martin et al., 2017): Conv3D/v(x,y,z)(r,s,t)(m) stands for a convolutional layer with v filters where x, y and z are the width, height and depth of the 3D kernel, with a stride of r, s and t on each dimension and SAME padding if m is equal to S or VALID padding if m is equal to V (VALID implies no padding and SAME implies padding that preserves output dimensions). MaxPooling(x,y,z)(r,s,t)(m) stands for a Max Pooling layer where x, y and z are the pool sizes, with a stride of r, s and t on each dimension and SAME padding if m is equal to S or VALID padding if m is equal to V (VALID implies no padding and SAME implies padding that preserves output dimensions). FC(x) stands for a fully connected layer with x nodes. The activation function used for each layer appears at the end of the layer description. Finally, and asterisk (*) in the layer description indicates a repetition of the object (layer or output tensor) the number of times indicated by the associated number.

Each layer description provides also 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 architectural alternatives for the proposed model, with the model 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).

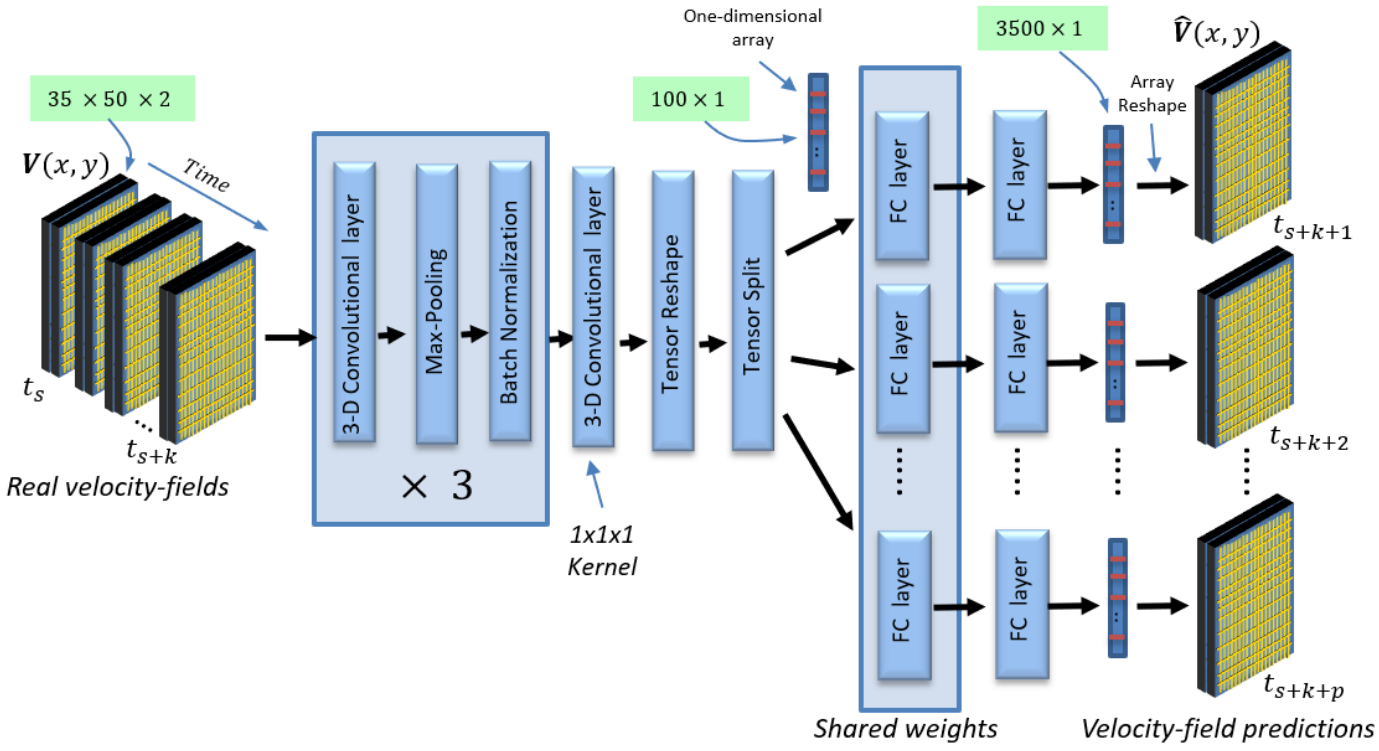

Fig. 5. Proposed deep learning model to perform velocity-field predictions for the *p* future time-slices based on *k* past velocity-field inputs

that start on an arbitrary initial time (t_s)

Layer #	Layer details	Tensor dimensions
1	Input	(bs,k,35,50,2)
2	Conv3D/5(2,2,2)(1,1,1)(V), ReLU	(bs,9,34,49,5)
3	MaxPooling3D(1,2,2)(1,2,2)(V)	(bs,9,17,24,5)
4	BatchNormalization	(bs,9,17,24,5)
5	Conv3D/10(2,2,2)(1,1,1)(V), ReLU	(bs,8,16,23,10)
6	MaxPooling3D(1,2,2)(1,2,2)(V)	(bs,8,8,11,10)
7	BatchNormalization	(bs,8,8,11,10)
8	Conv3D/20(2,2,2)(1,1,1)(V), ReLU	(bs,7,7,10,20)
9	MaxPooling3D(1,2,2)(1,2,2)(V)	(bs,7,3,5,20)
10	BatchNormalization	(bs,7,3,5,20)
11	Conv3D/p(1,1,1)(1,1,1)(V), ReLU	(bs,7,3,5,p)
12	Permute(4,1,2,3)	(bs,p,7,3,5)
13	Reshape(p,105)	(bs,p,105)
14	Split(p)	(bs,105) * p
15	p * FC(80), ReLU -- sharing weigths	(bs,80) * p
16	p * FC(3500), Sigmoid	(bs,3500) * p

Table I. 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.

4. RESULTS

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 11642 consecutive time-slices where the last ones (2416) are reserved for testing. Each time-slice made up of a 3D data volume that provides the x and y velocity-fields over a surface of 70 x 100 point. The resulting test set is segregated into *k*+*p* consecutive time-slices with a rolling-window strategy (section 3.1). The data structures (VF-DS) formed by these consecutive time-slices 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: square 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, $\hat{Y}$ the predicted values, $\bar{Y}$ the mean value of Y and N the total number of scalar values for all predicted time-slices (Hyndman & Koehler, 2006):

$$MSE = Mean((Y - \hat{Y})^2); \quad MAE = Mean(|Y - \hat{Y}|); \quad MAD = Median(|Y - \hat{Y}|)$$

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

$$R^2 = 1 - \frac{\sum_{i=0}^{N}(Y_i - \hat{Y}_i)^2}{\sum_{i=0}^{N}(Y_i - \bar{Y})^2}$$

$$sMAPE = \frac{100}{N}\sum_{i=0}^{N} 2\frac{|Y_i - \hat{Y}_i|}{|Y_i| + |\hat{Y}_i|} \ \%$$

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 II, III and IV 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 ($p = 6$).

Tables II, III and IV present the metrics using a color code to easily show where the best values are. Color coding uses a color palette where the greenest color is for the best result and the reddest for the worst. Color coding is applied across the rows to compare the results for the same metric.

Table II 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 II 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 period 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 III 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 III 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 III corresponds to the average of the rows in Table II. These values are obtained using 10 past time-slices ($k = 10$) as predictors, with no initial period (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 IV presents the forecast metrics with different number of time-slices used as predictors (parameter k), with no initial period (section 3.1). The values in Table IV 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 s 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.

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

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

	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 III. 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.

	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

Table IV. 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.

It is interesting to analyze the evolution of the forecast metrics for different VF-DSs along time and for different time ahead prediction periods. Fig 6 and 7 provides information about this evolution for the first (T0) and last (T5) prediction periods, respectively. Figure 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, Vega & Soria, 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 Fig 6 and 7. 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.

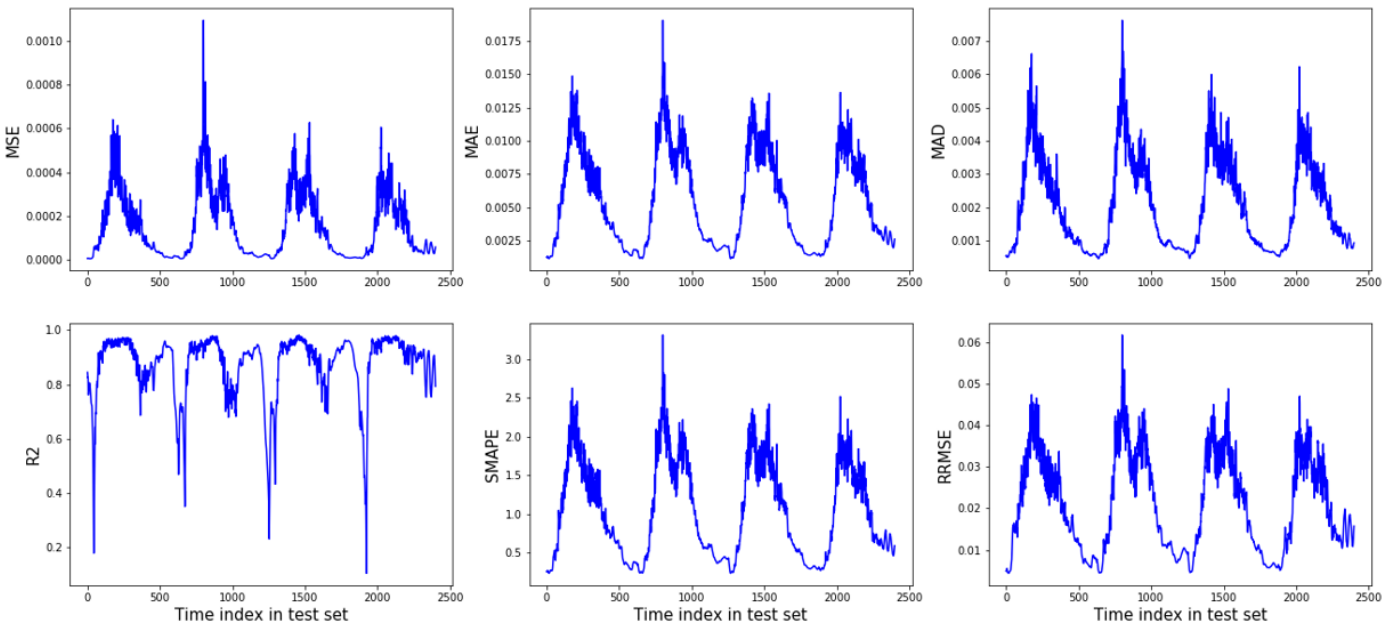

Fig 6. Forecast metrics for the prediction of the first time-ahead period (T0). The charts present evolution of the predictions along the time index of the test set.

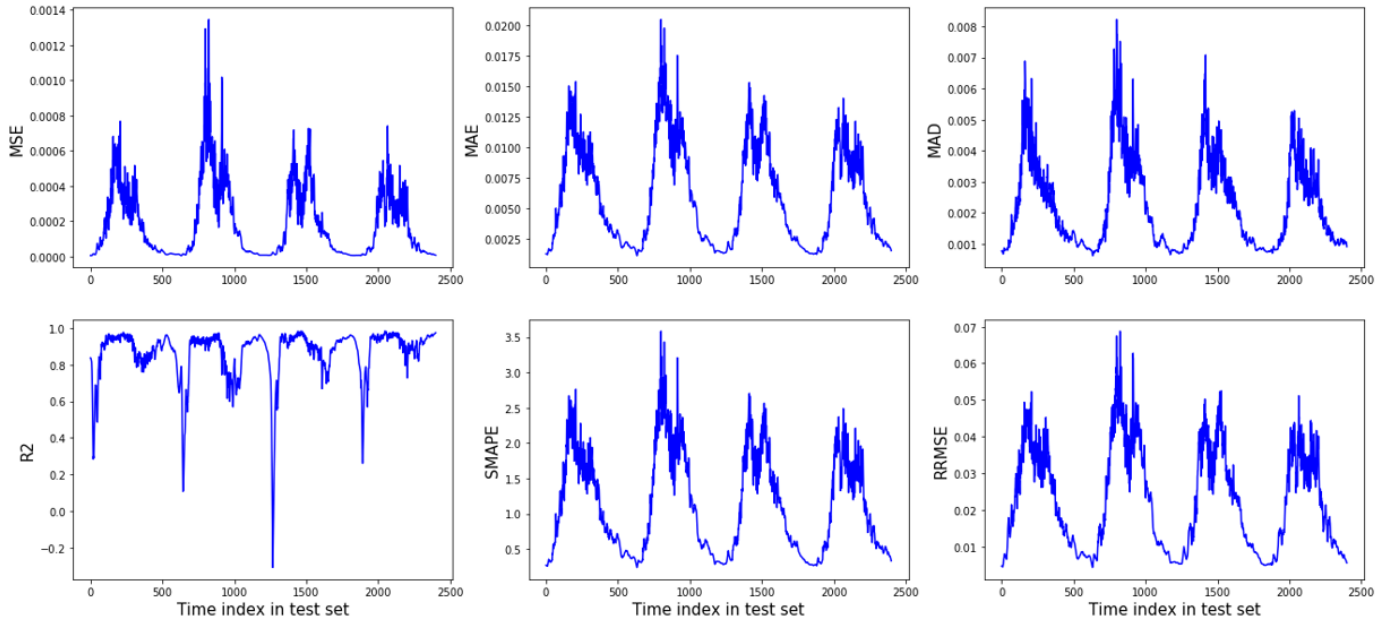

Fig 7. Forecast metrics for the prediction of the last time-ahead period (T5). The charts present evolution of the predictions along the time index of the test set.

In order to provide a visual representation of the quality of predictions, in Fig 8 and 9 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 8 corresponds to a time-slice with high velocity values, representing the injection phase of the jet, and Fig 9 to a different one with low values of velocity, representing the suction phase in the jet flow (see Fig. 1).

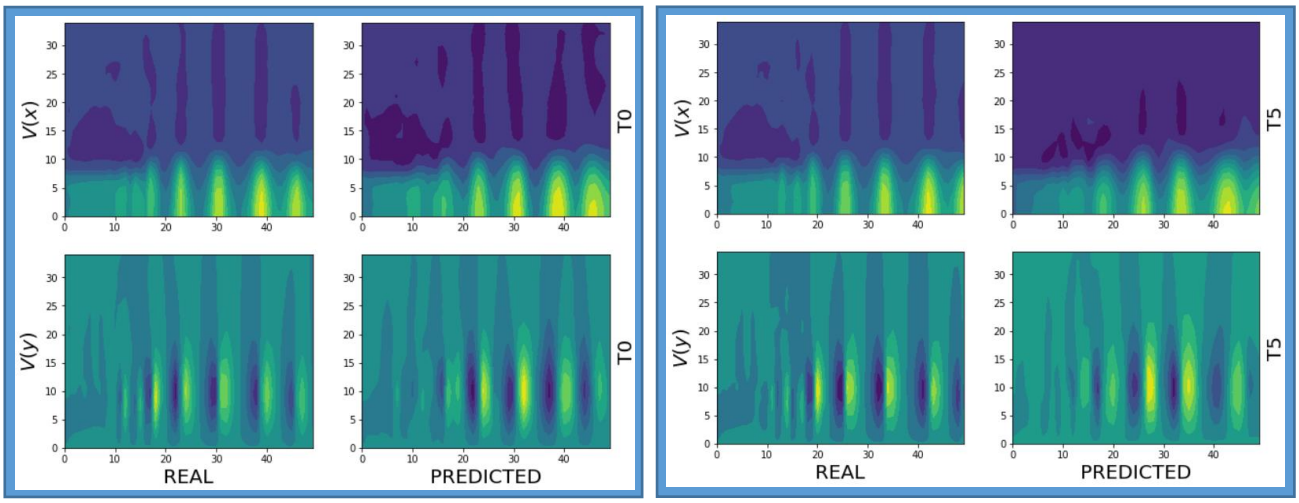

Fig 8. Contour graphs of the components x and y of velocity (rows) for real and predicted velocity fields (columns) when the predictions are made for the first time-ahead period (T0) (left part) and last time-ahead period (T5) (right part). Both predictions correspond to the time index 300 of the test set, which has high velocity values.

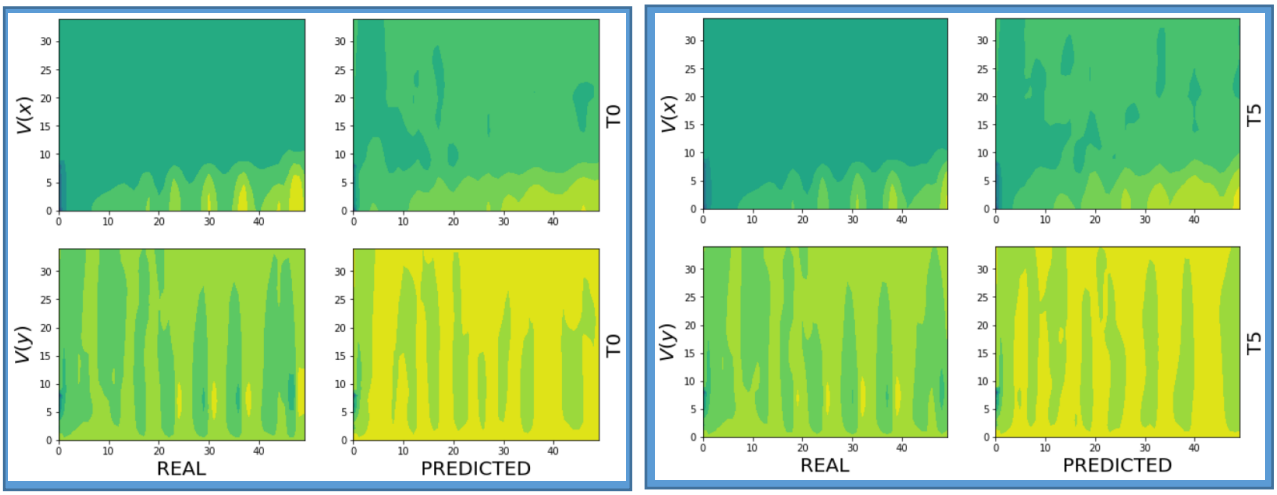

Fig 9. Contour graphs of the components x and y of velocity (rows) for real and predicted velocity fields (columns) when the predictions are made for the first time-ahead period (T0) (left part) and last time-ahead period (T5) (right part). Both predictions correspond to the time index 400 of the test set, which has low velocity values.

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 both forecast errors and the time and 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.

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 16GB of RAM, Intel Xeon 2.3GHz 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 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 1.068 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 jets dataset.

This work is also a proof of concept to assess the suitability of deep 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, Noack & Koumoutsakos, 2020) and to explore ensemble models for multivariate time-series forecasting (Lopez-Martin, Carro & Sanchez-Esguevillas, 2019).

ACKNOLEDGEMENTS

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).

REFERENCES

Abadi, M. et al., (2016). TensorFlow: Large-scale machine learning on heterogeneous distributed systems. arXiv: 1603.04467v2 [cs.DC].

Agrawal S. et al., (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 pp. 216-233. http://dx.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

Cai, S. et al., (2019a) Dense motion estimation of particle images via a convolutional neural network. Exp Fluids 60, 73 (2019). https://doi.org/10.1007/s00348-019-2717-2

Cai S. et al. (2019b) 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, J. Fluid Mech., 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, Annu. Rev. Fluid Mech. 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. Springer-Verlag, Berlin

Dargan, S. et al., (2019) A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning. Arch Computat Methods Eng (2019). https://doi.org/10.1007/s11831-019-09344-w

DeMont, E.& Gosline, J., (1998) Mechanics of jet propulsion in the hydromedusan jellyfish, polyorchis penicillatus, J. Exp.

Biol. 143:347-361

Ferrari, M. et al., (2006) Turbulent flow as a cause for underestimating coronary flow reserve measured by Doppler guide wire, Cardiovasc Ultrasound 14(4).

Fischer, P.F., Lottes, J.W. & Kerkemeier, S.G., Nek5000. Available online: https://nek5000.mcs.anl.gov (accessed on April 2020).

Gao, C., Zhang, W., Kou, J., Liu, Y. & Ye, Z., (2017) Active control of transonic buffet flow, J. Fluid Mech. 824:312-351

Glezer, A. & Amitay, M., (2002) Synthetic jetst, Annu. Rev. Fluid Mech. 34:503-529

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

Hyndman R.J. & Koehler A.B. (2006). Another look at measures of forecast accuracy. Int. J. Forecast. 22 (4) (2006) 679–688, http://dx.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

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, Phys. 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, Exp. Therm. Fluid Sci. 88:336-353

Le Clainche, S., Varas, F. & Vega, J.M., (2017) Accelerating reservoir simulations using POD on the fly, Int. J. Num. Meth. Eng. 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* 2019, *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, Springer, 950:557-566, 2019. DOI: 10.1007/978-3-030-20055-8_53

LeCun Y, Bengio Y, & Hinton G (2015) Deep Learning. Nature 521(7553):436-4, DOI: 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. Exp Fluids 58, 171 (2017). https://doi.org/10.1007/s00348-017-2456-1

Lopez-Martin M. et al., (2017). Network Traffic Classifier with Convolutional and Recurrent Neural Networks for Internet of Things. IEEE Access, vol. 5, pp. 18042-18050, 2017. 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, vol 100, pp 656-673. https://doi.org/10.1016/j.future.2019.05.060

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. et al., (2003) Real Time Feature Extraction for the Analysis of Turbulent Flows, Semantic Scholar, Chapter 13, 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.

Noack, B. R., Morzynski, M. & Tadmor, G., (2011) Reduced-Order Modelling for Flow Control, Springer, New York

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

Park, K.H., et al. (2013) Reduced-order model with an artificial neural network for aerostructural design optimization, J. Aircraft 50:1106-1116.

Pathak J. et al., (2018) Model-Free Prediction of Large Spatiotemporally Chaotic Systems from Data: A Reservoir Computing Approach, Phys. Rev. Lett. 120, 024102, DOI: 10.1103/PhysRevLett.120.024102

Pavlova, A. & Amitay, M., (2006) Electronic cooling using synthetic jet impingement, J. Heat Trans. 128:897-907

Pedregosa F. et al. (2011) Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, vol. 12, pp. 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, vol. 29, no. 9, 2017, pp. 2352–449

Scher S. (2018). Toward data-driven weather and climate forecasting: Approximating a simple general circulation model with deep learning. Geophysical Research Letters, 45, 12,616–12,622. 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, vol 12, number 7, pp. 2797-2809 https://doi.org/10.5194/gmd-12-2797-2019

Schmid, P., (2010) Dynamic mode decomposition of numerical and experimental data, J. Fluid Mech. 656:5-28

Sirovich, L., (1987) Turbulence and the dynamics of coherent structures, Quart. Appl. Math., XLV: 561—590

Vlachas P.R et al. (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. et al., (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): e0197704. https://doi.org/10.1371/journal.pone.0197704

Wang, H. & Menon, S., (2001) Fuel-air mixing enhancement by synthetic microjets, AIAA J., 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, 2019; 12 (10): 4261 https://doi.org/10.5194/gmd-12-4261-2019

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]

Xiaoxiao G., Wei L. & Iorio F., (2016) Convolutional Neural Networks for Steady Flow Approximation. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16). ACM, New York, NY, USA, 481–490. https://doi.org/10.1145/2939672.2939738

Zong, H. & Kotsonis, M., (2018) Formation, evolution and scaling of plasma synthetic jets, J. Fluid Mech. 837:147-181