18th Brazilian Congress of Thermal Sciences and Engineering
November 16–20, 2020, (Online)

ENC-2020-0223
Application of Convolutional Neural Networks for Surrogate Models of Unsteady Flows

Hugo F. S. Lui
William R. Wolf
Universidade Estadual de Campinas, UNICAMP, 13083-970 Campinas, SP, Brazil
hugo.slui@gmail.com
wolf@fem.unicamp.br

Abstract. *In this work, we present a numerical methodology for construction of surrogate models of fluid flows which combine data-driven system identification and convolutional neural networks. The framework is implemented in a context similar to that of the sparse identification of non-linear dynamics (SINDy) algorithm with some modifications regarding the regression step. The approach presented in this work allows us to obtain an ODE for each flow variable at each mesh point. This should be beneficial for flow control approaches since every flow state can be modified by a control law. The method is tested for an unsteady compressible flow past a cylinder at low Reynolds number. Results demonstrate that the current methodology provides accurate reconstructions of the high fidelity model.*

Keywords: *Surrogate models, Deep Learning, Computational Fluid Dynamics*

1. INTRODUCTION

High performance computing allows the use of high fidelity numerical simulations of turbulent flows over large-scale industrial problems. The results from these simulations certainly improve the understanding of complex physical phenomena such as drag reduction, heat transfer, noise generation, mixing enhancement to name a few. Such simulations may lead to discretization consisting of millions or billions of degree of freedom in order to resolve the energetically relevant spatial and temporal scales. On the other hand, these simulations have a high computational cost, and require the acquisition and treatment of large datasets. Consequently, the use of surrogate models is particularly appealing in the context of optimization analyzes and studies of flow control due to their inherent reduction in the computational costs compared to large-scale simulations. Frangos *et al.* (2010) categorize surrogate models into two classes: reduced-order models (ROMs) and data-fit models.

The construction of ROMs for fluid flows is an active area of research and a broad variety of methodologies for the development of ROMs is available. In general, modal analysis techniques, such as proper orthogonal decomposition (POD) (Lumley, 1967; Sieber *et al.*, 2016; Ribeiro and Wolf, 2017; Towne *et al.*, 2018; Lui and Wolf, 2019) has been combined with Galerkin projection methods or machine learning techniques for construction of ROMs. Such model reduction methods are summarized in a review by Taira *et al.* (2017). Recently, a sequel to this previous paper was published by Taira *et al.* (2019), in which the applications and perspectives of flow modal decomposition techniques are discussed.

Reduced order models developed based on Galerkin projection (Rowley *et al.*, 2004; Carlberg *et al.*, 2011, 2017) are directly related to the physics of the problem and employ POD to rewrite the Navier–Stokes equations as sets of dynamical systems for the evolution of the POD temporal modes. On the other hand, ROMs developed based on data-driven methods such as POD-SINDy (Brunton *et al.*, 2016) and POD-DNN (Lui and Wolf, 2019) use POD to reduce the dimensionality of the model. Then, the regression step is performed by sparse regression or a deep neural network (DNN) in order to obtain a system of ordinary differential equations (ODEs) representing the dynamics of the POD temporal modes. Recently, Yu *et al.* (2019) proposed a data-driven method for construction of ROMs combining POD and dynamic mode decomposition (DMD) (Schmid, 2010; Tu *et al.*, 2014). Neither approaches have a direct connection with the physics of the problem since these techniques learn from data.

Data-fit models are typically generated by learning the input-output relationship in the high-fidelity model from the simulation data. In literature, Gaussian processes have been highly used as surrogate for complex dynamical systems (Parussini *et al.*, 2017; Perdikaris *et al.*, 2015; Kennedy and O'Hagan, 2001). However, unsteady flow problems require methods capable of addressing strong nonlinearities and high dimensionality that may not be present in traditional data-fit models (Sun *et al.*, 2020). In this way, deep learning is a natural candidate for the development of data-fit surrogate models of large-scale dynamical systems, typical of numerical simulations of unsteady flows. A discussion of the application of deep learning in fluid dynamics is provided by Kutz (2017).

In this work, we present a numerical methodology for construction of data-fit surrogate models of fluid flows through combination of data-driven system identification and regression analysis using convolutional neural networks (CNNs). The current framework is implemented in a context similar to the SINDy method (Brunton *et al.*, 2016) with some modifications regarding to the regression step and dimensionality reduction, which is not applied here. We describe the numerical technique employed and present results obtained by the current methodology for a compressible flow past a cylinder.

2. Formulation

The equations governing the unsteady motion of a compressible viscous fluid are known as the Navier-Stokes equations. This set of equations forms a coupled system of nonlinear partial differential equations (PDEs) describing the conservation of mass, momentum, and energy for a fluid. Using the method of lines one can first approximate the spatial derivatives producing a system of ordinary differential equations (ODEs). In the most general notation, we can consider each mesh point as a dynamical system of form (1)

$$\frac{d\mathbf{q}}{dt} = \mathbf{F}(\mathbf{q}, t) \ , \tag{1}$$

where $\mathbf{F}$ is a nonlinear operator, $\mathbf{q} = [\rho, \rho u, \rho v, \rho w, E]^T$ is the vector of flow variables, ρ is the density, u, v, w are the x, y, z components of the velocity vector, E is the total energy per unit of volume and t is the time.

Recently, Brunton *et al.* Brunton *et al.* (2016) introduced the SINDy algorithm which identifies ODEs from data. In order to determine the function $\mathbf{F}$, we follow the ideas from the SINDy algorithm with some modifications regarding to the regression step. First, we collect snapshots of the flow variables which will be our training data. For now, let us consider a two-dimensional flow. The data set is then arranged into a matrix $\mathbf{Q}$ as

$$\mathbf{Q}(t_1) = \begin{bmatrix} \mathbf{q}(x_1, y_1, t_1) & \mathbf{q}(x_1, y_2, t_1) & \dots & \mathbf{q}(x_1, y_{N_y}, t_1) \\ \mathbf{q}(x_2, y_1, t_1) & \mathbf{q}(x_1, y_2, t_1) & \dots & \mathbf{q}(x_2, y_{N_y}, t_1) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{q}(x_{N_x}, y_1, t_1) & \mathbf{q}(x_{N_x}, y_2, t_1) & \dots & \mathbf{q}(x_{N_x}, y_{N_y}, t_1) \end{bmatrix}_{N_x \times N_y \times N_{var} \times N_T} \ , \tag{2}$$

where N_x is the number of grid points in the x-direction, N_y is the number of grid points in the y-direction, N_{var} is the number of flow variables and N_T is the number of snapshots. Hence, $\mathbf{Q} = [\mathbf{Q}(t_1) \dots \mathbf{Q}(N_T)]$.

Because of the high dimensionality of the input data $\mathbf{Q}$, most of the methodologies for the development of ROMs employ POD as a strategy to reduce the dimension of the dynamical system. In the current framework, the convolutional layers are used for feature extraction from the input flow fields obtained from high fidelity simulations, and then the fully connected (FC) layers are utilized to obtain an ODE for each flow variable in each mesh point. Therefore, dimensionality reduction via POD is not needed like in (Lui and Wolf, 2019).

Next, we compute the derivative $\frac{d\mathbf{q}}{dt}$ numerically using the data $\mathbf{q}(t)$ for each mesh point. The numerical scheme employed for the temporal derivatives is a 10th-order accurate compact scheme (Lele, 1992) which provides high spectral resolution being non-dissipative and low-dispersive. The derivative $\dot{\mathbf{q}}(t)$ is then obtained as

$$\delta_1 \dot{\mathbf{q}}_{i-2} + \delta_2 \dot{\mathbf{q}}_{i-1} + \dot{\mathbf{q}} + \delta_2 \dot{\mathbf{q}}_{i+1} + \delta_1 \dot{\mathbf{q}}_{i+2} = \delta_3 \frac{\mathbf{q}_{i+3} - \mathbf{q}_{i-3}}{6h} + \delta_4 \frac{\mathbf{q}_{i+2} - \mathbf{q}_{i-2}}{4h} + \delta_5 \frac{\mathbf{q}_{i+1} - \mathbf{q}_{i-1}}{2h} \ . \tag{3}$$

in the equation above, h is the time step between the snapshots. The coefficients of the numerical scheme are set as $\delta_1 = 1/20$, $\delta_2 = 1/2$, $\delta_3 = 1/100$, $\delta_4 = 101/150$ and $\delta_5 = 17/12$ for a 10th-order discretization. The system of equations (3) written for each grid point can be solved as a pentadiagonal linear system of equations for the unknown derivatives $\dot{\mathbf{q}}(t)$. The boundary data points can be computed from the interior points following Olson (2012). The derivatives $\dot{\boldsymbol{a}}(t)$ are then arranged into a matrix $\dot{\mathbf{Q}} = \left[\dot{\mathbf{Q}}(t_1) \dots \dot{\mathbf{Q}}(N_T)\right]$

$$\dot{\mathbf{Q}}(t_1) = \begin{bmatrix} \dot{\mathbf{q}}(x_1, y_1, t_1) & \dot{\mathbf{q}}(x_1, y_2, t_1) & \dots & \dot{\mathbf{q}}(x_1, y_{N_y}, t_1) \\ \dot{\mathbf{q}}(x_2, y_1, t_1) & \dot{\mathbf{q}}(x_2, y_2, t_1) & \dots & \dot{\mathbf{q}}(x_2, y_{N_y}, t_1) \\ \vdots & \vdots & \ddots & \vdots \\ \dot{\mathbf{q}}(x_{N_x}, y_1, t_1) & \dot{\mathbf{q}}(x_{N_x}, y_2, t_1) & \dots & \dot{\mathbf{q}}(x_{N_x}, y_{N_y}, t_1) \end{bmatrix}_{N_x \times N_y \times N_{var} \times N_T} \ . \tag{4}$$

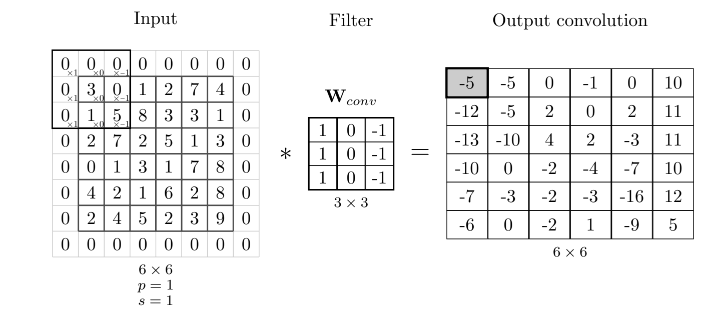
Now, we can set up a regression problem to find the weights $\mathbf{W}$ and biases $\mathbf{b}$ that determine the function $\mathbf{F}$ presented in Eq. 1

$$\dot{\mathbf{Q}} = \mathbf{W}\Theta(\mathbf{Q}) + \mathbf{b} \ , \tag{5}$$

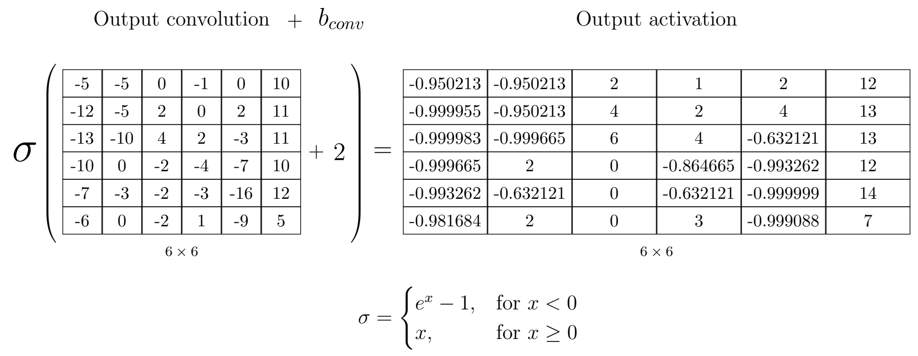
where $\Theta(\mathbf{Q})$ is the matrix of features. In the SINDy algorithm, we need to construct a library $\Theta(\mathbf{Q})$ consisting of candidate functions. For instance, $\Theta(\mathbf{Q})$ may be composed of constant, polynomial, exponential and trigonometric terms.

However, in many cases it is difficult to know a priori the adequate set of features appearing in $\boldsymbol{\Theta}(\mathbf{Q})$. Hence, we use deep learning to circumvent the problem of finding the features which represent the dynamics of the system. Therefore, this methodology can discover not only $\mathbf{W}$ and $\mathbf{b}$ but also $\boldsymbol{\Theta}(\mathbf{Q})$.

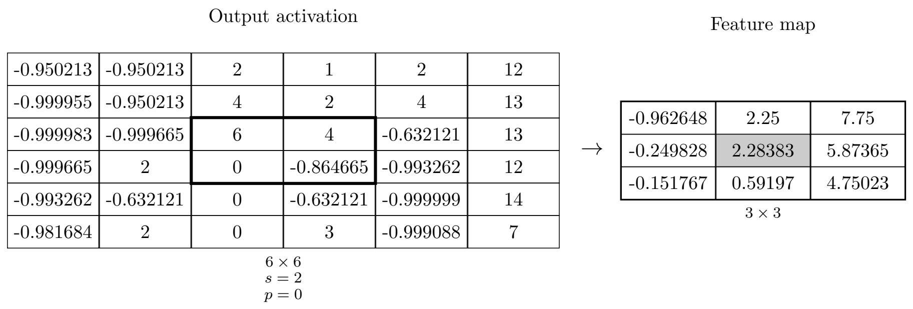
Deep learning methods are feature learning algorithms that can find a proper set of features using multiple processing layers that typically are connected in a hierarchical structure, such that higher layer features are defined in terms of lower layer features (Bengio, 2009; Goodfellow *et al.*, 2016). The objective of a deep learning method is to learn appropriate parameters ($\mathbf{W}$, $\mathbf{b}$) and features $\boldsymbol{\Theta}(\mathbf{Q})$ that provide a mapping of the input data to the output data that minimizes a loss function. This function measures the difference between the predictions and the desired output. Therefore, deep learning methods allow the learning of complex functions through mapping the input to the output directly from a given data. In the current work, we use a convolutional neural network (CNN) to learn the weights $\mathbf{W}$, biases $\mathbf{b}$ and features $\boldsymbol{\Theta}(\mathbf{Q})$. The CNN calculation procedure used is this work is summarized in the following steps:



(a) Example of a convolution operation.


Output convolution + b_{conv}
Output activation
6×6
$\sigma = \begin{cases} e^x - 1, & \text{for } x < 0 \\ x, & \text{for } x \geq 0 \end{cases}$

(b) Example of an activation function operation.



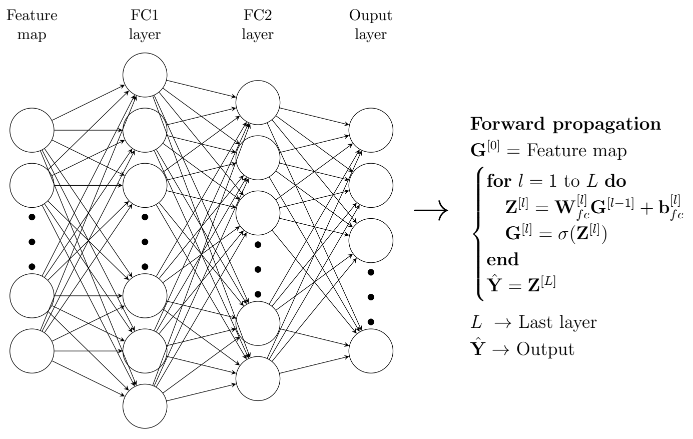
(c) Example of an average pooling operation.

Figure 1. Example of a convolutional neural network layer.

1. The input of the CNN is the matrix $\mathbf{Q}$ with size of (N_x, N_y, N_{var}, N_T) where N_T can be also referred as the number of training examples. The desired output is matrix $\dot{\mathbf{Q}}$ with the same size as the input data. Due to GPU memory limitations, we split our training data into mini-batches of size n_{batch}.

2. The convolution weights $\mathbf{W}_{conv}$ and the fully-connected weights $\mathbf{W}_{fc}$ are initialized using Xavier initialization (Xavier and Yoshua, 2010). The convolution biases $\mathbf{b}_{conv}$ and the fully-connected biases $\mathbf{b}_{fc}$ are initialized to zero. We also initialize the parameters related to the ADAM optimizer (Kingma and Ba, 2014) to zero.

3. Forward propagation in the convolutional layers is performed using the operations shown in figure 1 . A convolutional layer is composed by three types of layer: convolution, activation, and pooling.



```
Forward propagation
$\mathbf{G}^{[0]}$ = Feature map
for $l = 1$ to $L$ do
    $\mathbf{Z}^{[l]} = \mathbf{W}^{[l]}_{fc}\mathbf{G}^{[l-1]} + \mathbf{b}^{[l]}_{fc}$
    $\mathbf{G}^{[l]} = \sigma(\mathbf{Z}^{[l]})$
end
$\hat{\mathbf{Y}} = \mathbf{Z}^{[L]}$

$L$ → Last layer
$\hat{\mathbf{Y}}$ → Output
```

Figure 2. Example of fully connected layers.

(a) The convolution layer extracts spatial features from the input map using a set of n_c learnable filters ($\mathbf{W}_{conv}$). The convolution process consists of sliding a filter of size $f \times f$ across the entire input map using a certain stride s. To maintain dimensions of the output the same as those from input after convolution operations, we pad p the input map with zeros around the border.

(b) Next, an activation layer is used to introduce non-linearity into the network. There are several available activation functions such as sigmoid, hyperbolic tangent (tanh), rectified linear unit (ReLU) (Nair and Hinton, 2010), exponential linear unit (ELU) (Clevert *et al.*, 2015), to name a few. In this work, we employ ELU as activation function based on previous results obtained from DNNs (Lui and Wolf, 2019).

(c) Finally, a pooling layer is utilized for reducing the spatial size of the output activation map. A pooling window of size $f \times f$ is slided over the entire output activation map using a certain stride s to generate an output feature map. The most common pooling operations are max pooling and average pooling. Fukami *et al.* (2019) showed that average pooling is more robust than max pooling for fluid flow problems and, therefore, we use average pooling in our calculations.

4. Next, a feature map is generated by flattening the last pooling layer which is the input layer for the fully connected layers. Then, forward propagation in the fully connected layers is performed as shown in figure 2. The feature map provides the initial information that then propagates to all hidden units at each layer through calculation of a set of linear and nonlinear operations which produce the predicted output $\hat{\dot{\mathbf{Q}}}$. One should note that we also reshape the predicted output into a 4-dimensional data structure same as the input data $\mathbf{Q}$. For all hidden layers except the last one, the activation function $G^{[l]}$ is set as the ELU. For the last layer, we employ a identity function.

5. The cost function is then computed using the following equation

$$J = \frac{1}{2N_T}\left[\sum_{i=1}^{N_x}\sum_{j=1}^{N_y}\sum_{k=1}^{N_T}\sum_{m=1}^{N_{var}}(\hat{\dot{Q}}_{i,j,k,m} - \dot{Q}_{i,j,k,m})^2 + \lambda\sum_{l=1}^{L}\sum_{k=1}^{n^{[l]}}\sum_{j=1}^{n^{[l+1]}}(W^{[l]}_{fc_{j,k}})^2\right], \tag{6}$$

where $n^{[l]}$ is the number of hidden units for layer l, L is the number of fully connected layers, and λ is the regularization parameter.

6. Gradients of the cost function with respect to each parameter are computed using back-propagation. Further details regarding this algorithm can be found in Goodfellow *et al.* (2016).

7. Finally, the parameters ($\mathbf{W}_{conv}$, $\mathbf{b}_{conv}$, $\mathbf{W}_{fc}$, $\mathbf{b}_{fc}$) are updated using the ADAM optimizer (Kingma and Ba, 2014) as well the parameters related to the optimizer.

8. The previous steps are repeated until the convergence criteria (number of epochs) is satisfied. One epoch means that the training algorithm has seen all mini-batches.

n the current work, the open source machine learning framework Tensorflow (Abadi and et. al, 2016) is used for training the CNN. Once we have learned the parameters $\mathbf{W}_{fc}$ and $\mathbf{b}_{fc}$ of our model (5), we can use them to obtain the flow field predictions $\hat{\mathbf{q}}$ given the initial conditions $\mathbf{q}(0)$

$$\frac{d\hat{\mathbf{q}}(t)}{dt} = \mathbf{W}_{fc}^{[L]}\mathbf{G}^{[L-1]} + \mathbf{b}_{fc}^{[L]} = \mathbf{F}(\hat{\mathbf{q}}(t)) \, , \tag{7}$$

where L is the index of the last layer, $\mathbf{G}^{[L-1]}$ is the activation function matrix of the penultimate layer, $\mathbf{W}_{fc}^{[L]}$ is the weight matrix of the last fully connected layer, and $\mathbf{b}_{fc}^{[L]}$ is the bias vector of the last fully connected layer. This system of coupled ODEs is integrated using an explicit 5 stage 4th-order Runge-Kutta scheme derived by Kennedy *et al.* (2000). The values of $\mathbf{G}^{[L-1]}$ depend on the parameters from previous layers. Thus, for each stage of the 4th-order Runge Kutta, we need to perform forward propagation for both convolutional and fully connected layer to obtain $\boldsymbol{F}(\hat{\boldsymbol{q}}(t))$. The approach presented in this work allows us to obtain an ODE for each flow variable at each mesh point. This should be useful for application of flow control at specific mesh points.

The performance of the CNN depends on the selection of hyperparameters such as the learning rate α, regularization parameter λ, number of convolutional layers n_{conv}, number of filters $n_c^{[l]}$, filter size $f^{[l]}$, stride $s^{[l]}$, padding $p^{[l]}$, number of fully connected layer L, number of hidden units for each fully connected layer $n^{[l]}$ and batch size n_{batch}. Indeed, finding an optimal set of hyperparameters which minimizes the cost function is a difficult task given the considerable number of hyperparameters involved. The CNN architecture is based on works of Fukami *et al.* Fukami *et al.* (2019) and Murata *et al.* Murata *et al.* (2019) with modifications. The hyperparameters related to the fully connected layers are obtained via random search and further details regarding this process can be found in Lui and Wolf (2019). To report the performance of each model from a set of candidates, we compute the L_2 norm error over the entire dataset as

$$Error(t) = \frac{\|\mathbf{q}(\mathbf{x},t) - \hat{\mathbf{q}}(\mathbf{x},t)\|_{L_2}}{\|\mathbf{q}(\mathbf{x},t)\|_{L_2}} \, . \tag{8}$$

the best candidate model is the one with the lowest time averaged L_2 norm error.

3. Results

This section presents results obtained by the approach proposed here which combines data-driven system identification and convolutional neural networks. The current method is tested on the reconstruction of an unsteady two-dimensional compressible flow past a cylinder. For this case, we show the ability and limitations of the method to identify the flow dynamics.

3.1 Numerical methods for flow simulations

For the following investigation of the flow past a cylinder, the system dynamics are modeled by the compressible Navier-Stokes equations written in the contravariant form on a general curvilinear system. The numerical scheme for the spatial discretization of the equations is a sixth-order accurate compact scheme (Nagarajan *et al.*, 2003) implemented on a staggered grid. A sixth-order compact scheme is also employed for interpolation and filtering (Lele, 1992) of fluid properties on the different nodes of the staggered configuration.

The time integration of the fluid equations is carried out by the fully implicit second-order scheme of Beam and Warming (Beam and Warming, 1978) in the near-wall region in order to overcome the time step restriction due to the usual near-wall fine-grid numerical stiffness. A third-order Runge-Kutta scheme is used for time advancement of the equations in flow regions far away from solid boundaries. No-slip adiabatic wall boundary conditions are applied along the solid surfaces and characteristic plus sponge boundary conditions are applied in the far field locations. The numerical tool has been previously validated for several simulations of unsteady compressible flows (Wolf *et al.*, 2012; Ramírez and Wolf, 2015).

3.2 Surrogate model

3.2.1 Compressible flow past a cylinder

The numerical simulations are conducted for Reynolds and Mach numbers $Re = 150$ and $M = 0.4$, respectively. These non-dimensional parameters are computed based on freestream quantities. The grid configuration consists of a body-fitted O-grid of with 421 × 751 points in the streamwise and wall-normal direction, respectively. The flow is recorded for 1120 snapshots with non-dimensional time steps of $h = 0.05$. These snapshots are collected after an initial transient period of the simulation is discarded.

The surrogate model is obtained following the procedure described in section 2. The training data comprises the first 512 snapshots and the remaining data is employed as the test set. We extract a portion of the computational domain and, thus, the number of grid points used for training is 420 × 360. The flow variables utilized in this problem are density, x and y momentum components, and pressure p. The CNN architecture used in this case is shown in figure 3. The numbers of fully connected layers and hidden units for each layer were obtained via random search. The CNN is made up of three convolution layers and three fully connected layers. The set of parameters employed in the generation of candidate models is listed in table 1 and the set of hyperparameters used on the best candidate model is presented in table 2 .

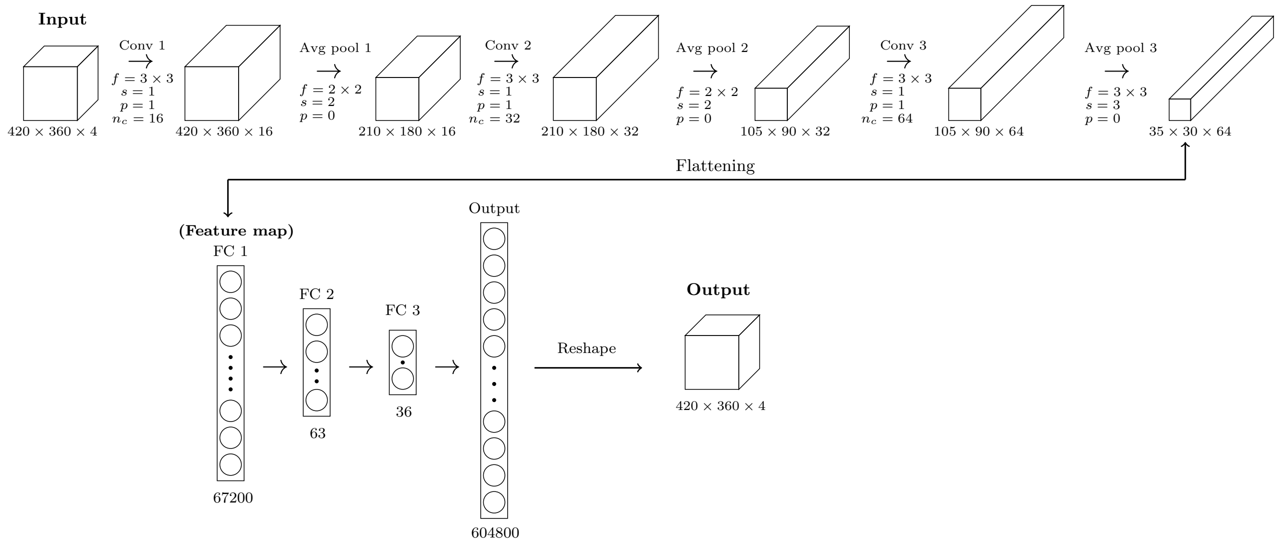


Figure 3. CNN architecture used for the cylinder case.

Table 1. Model generation parameters for the cylinder case.

n_{models}	$n_{layers_{min}}$	$n_{layers_{max}}$	$n_{hidden_{min}}$	$n_{hidden_{max}}$	λ_{min}	λ_{max}
150	3	6	24	96	10^{-6}	10^{-4}

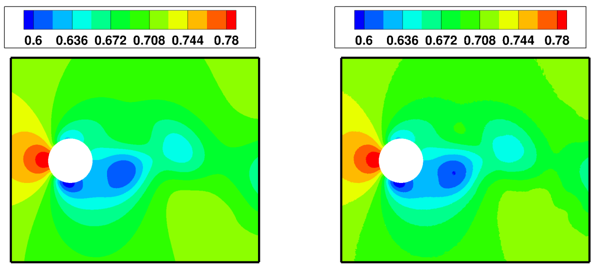
Table 2. Set of hyperparameters used on the best surrogate candidate model for the cylinder case.

σ	α	λ	n_{batch}	n_{epochs}	β_1	β_2	ϵ
ELU	0.0003	7.24253×10^{-5}	32	100	0.9	0.999	1.0×10^{-8}

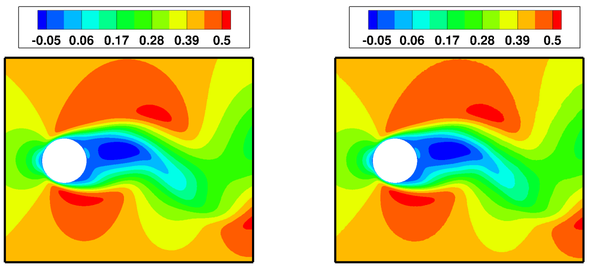
Figures 4 (a) to (d) show contours of pressure, x and y momentum, and z-vorticity, respectively, along the cylinder at time $t = 410$, which is beyond the training window. The snapshots allow a qualitative comparison of results between the high fidelity and surrogate models. For the current Reynolds number, the flow develops a typical von Kármán vortex street along the cylinder wake. The periodical pattern of the vortex shedding can be observed in the figures, in particular for the contours of z-vorticity and y-momentum. One can observe from the figures that the computation of the flow using the proposed framework exhibits a good agreement with the high fidelity model.

In order to show a more qualitative evaluation of the model reconstructions, the x-momentum and pressure time histories are presented for the high fidelity and surrogate models in Figs. 5 and 6, respectively. The figures on the left column show results for a probe located just behind the cylinder, close to the surface, at $(x,y) = (0.55,-0.06)$. The cylinder has radius 0.5 and its center is positioned in the origin of the Cartesian system. On the right column, results are obtained for a probe downstream the cylinder wake, at $(x,y) = (1.1,-0.06)$. Results are shown for both the training window period and beyond. In general, one can observe that the surrogate model accurately reproduces the high fidelity model results during and beyond the training window.

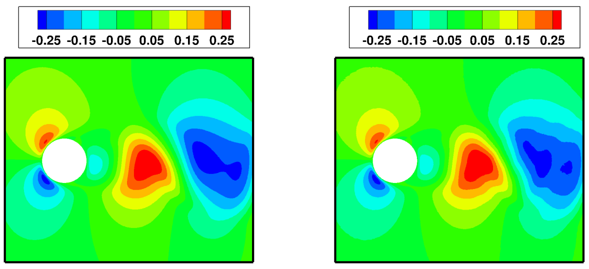
Figure 7 presents the evolution of the L_2 error norm as a function of time. One can see in this figure that the error values remain low throughout the entire time history for all variables except y-momentum, which displays a larger relative error. Probably, the filters $\mathbf{W}_{conv}$ are failing to learn some of the spatial features of the y-component of momentum. Pressure and density have a similar spatial pattern and, as a result, most of the parameters might have been related to these fields. This can explain the small differences between the high fidelity and surrogate model results in the wake region for y-momentum and z-vorticity fields.



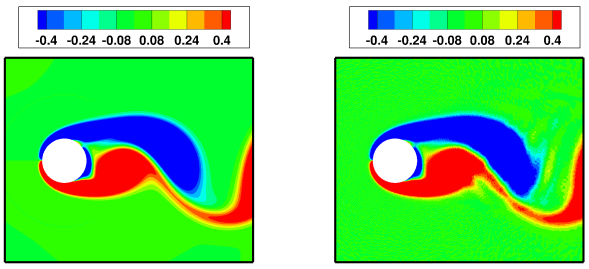
(a) Contours of pressure, t = 410.



(b) Contours of x-momentum, t = 410.

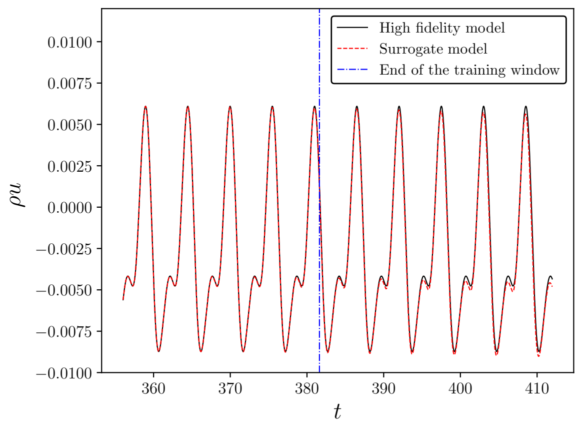


(c) Contours of y-momentum, t = 410.

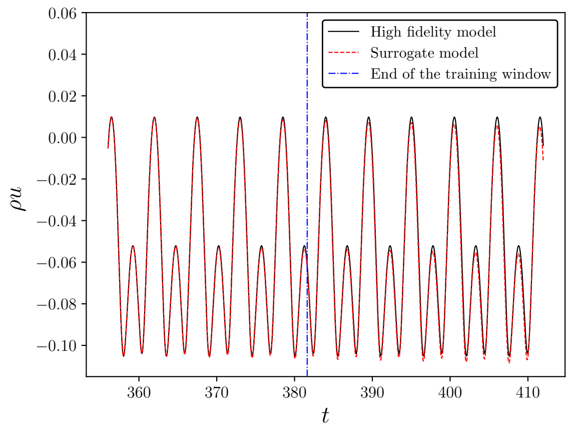


(d) Contours of z-vorticity, t = 410.

Figure 4. Comparison between the high fidelity model (left) and surrogate model (right) for a compressible flow past a cylinder.



(a) Probe positioned just behind the cylinder.



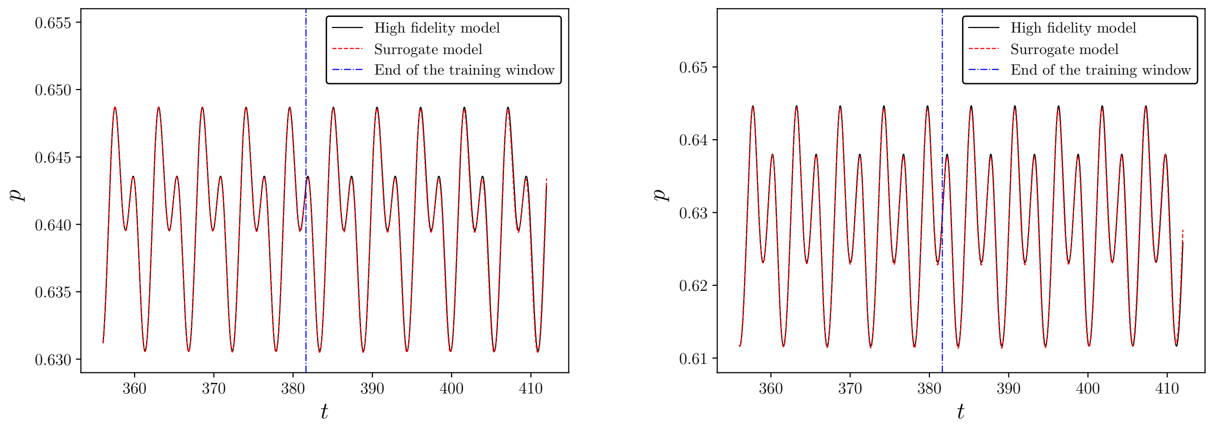
(b) Probe away from the cylinder, along the wake.

Figure 5. Time history of x-momentum for probes located at different positions.

4. Conclusions and additions to the final paper

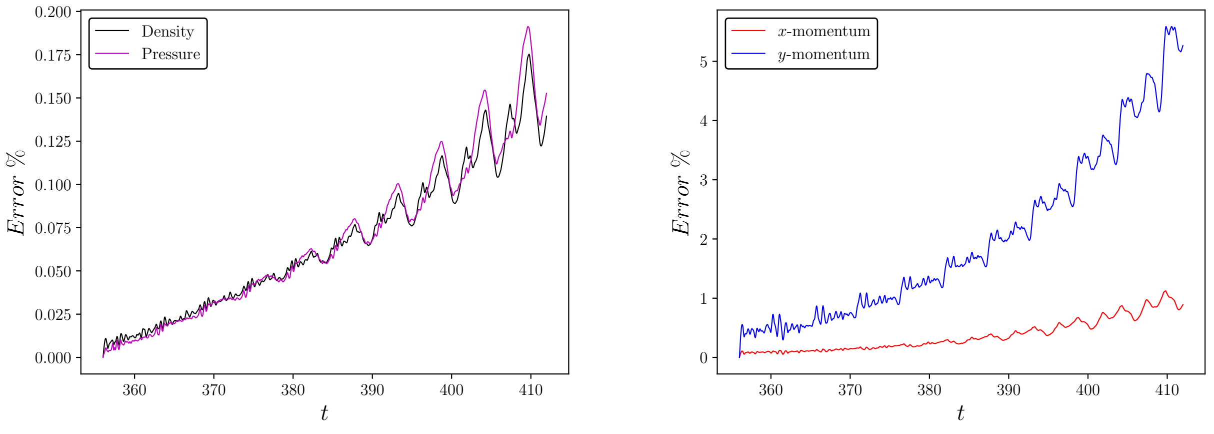
We present a methodology for constructing surrogate models for fluid flows combining data-driven system identification and regression analysis via convolutional neural networks (CNNs). The framework is implemented in a context similar to that of the sparse identification of non-linear dynamics (SINDy) algorithm with some modifications regarding to the regression step. The approach presented in this work allows us to obtain an ODE for each flow variable at each mesh point. The method is tested for a problem involving nonlinear dynamical systems: the compressible flow past a cylinder at low Reynolds number. For this case, the numerical simulations are performed using a high-order compact finite difference flow solver. Then, the high fidelity results are used as the input data for the construction of surrogate models.

Results demonstrate that the current methodology provides accurate reconstructions of the high fidelity model. The method presents stable solutions even beyond the training window of the CNN. The surrogate model is able to capture



(a) Probe positioned just behind the cylinder. (b) Probe away from the cylinder, along the wake.

Figure 6. Time history of pressure for probes located at different positions.



(a) Density and pressure. (b) Momentum components.

Figure 7. Temporal evolution of the relative error for the cylinder case.

most of the dynamics of vortex shedding. The analyses of the L_2 error indicate that some of the convolutional filters are failing to learn certain spatial features of the y-component of momentum. This can explain the slight differences between the high fidelity and surrogate model results in the wake region for z-vorticity fields and y-momentum. Calibration techniques should improve the performance of the surrogate model. We expect that the current methodology can be further improved for applications in flow control and optimization.

5. ACKNOWLEDGEMENTS

The authors acknowledge the financial support received from Fundação de Amparo à Pesquisa do Estado de São Paulo, FAPESP, under Grants No. 2013/08293-7, No. 2013/07375-0 and No. 2019/26196-5, and from Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, under Grants No. 304335/2018-5 and 407842/2018-7. We also thank CEPID-CeMEAI for providing the computational resources used in this work. We gratefully acknowledge Tulio Ricciardi for providing the high fidelity cylinder data employed in the current work.

6. REFERENCES

Abadi, M. and et. al, 2016. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". *CoRR*, Vol. abs/1603.04467.

Beam, R.M. and Warming, R.F., 1978. "An implicit factored scheme for the compressible navier-stokes equations". *AIAA Journal*, Vol. 16, No. 4, pp. 393–402.

Bengio, Y., 2009. "Learning deep architectures for ai". *Found. Trends Mach. Learn.*, Vol. 2, No. 1, pp. 1–127.

Brunton, S.L., Proctor, J.L. and Kutz, N.J., 2016. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". *Proceedings of the National Academy of Sciences*, Vol. 113, No. 15, pp. 3932–3937.

Carlberg, K., Bou-Mosleh, C. and Farhat, C., 2011. "Efficient non-linear model reduction via a least-squares petrov–galerkin projection and compressive tensor approximations". *International Journal for Numerical Methods in Engineering*, Vol. 86, No. 2, pp. 155–181.

Carlberg, K., Barone, M. and Antil, H., 2017. "Galerkin v. least-squares petrov–galerkin projection in nonlinear model reduction". *Journal of Computational Physics*, Vol. 330, pp. 693 – 734.

Clevert, D., Unterthiner, T. and Hochreiter, S., 2015. "Fast and accurate deep network learning by exponential linear units (elus)". *CoRR*, Vol. abs/1511.07289.

Frangos, M., Marzouk, Y., Willcox, K. and van Bloemen Waanders, B., 2010. *Surrogate and Reduced-Order Modeling: A Comparison of Approaches for Large-Scale Statistical Inverse Problems*, John Wiley & Sons, Ltd, chapter 7, pp. 123–149.

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

Goodfellow, I., Bengio, Y. and Courville, A., 2016. *Deep Learning*. The MIT Press.

Kennedy, C.A., Carpenter, M.H. and Lewis, R., 2000. "Low-storage, explicit runge–kutta schemes for the compressible navier–stokes equations". *Applied Numerical Mathematics*, Vol. 35, No. 3, pp. 177 – 219.

Kennedy, M.C. and O'Hagan, A., 2001. "Bayesian calibration of computer models". *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, Vol. 63, No. 3, pp. 425–464.

Kingma, D.P. and Ba, J., 2014. "Adam: A method for stochastic optimization". *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Kutz, J., 2017. "Deep learning in fluid dynamics". *Journal of Fluid Mechanics*, Vol. 814, p. 1–4.

Lele, S.K., 1992. "Compact finite difference schemes with spectral-like resolution". *Journal of Computational Physics*, Vol. 103, No. 1, pp. 16–42.

Lui, H.F.S. and Wolf, W.R., 2019. "Construction of reduced-order models for fluid flows using deep feedforward neural networks". *Journal of Fluid Mechanics*, Vol. 872, p. 963–994.

Lumley, J.L., 1967. "The structure of inhomogeneous turbulence". In *Atmospheric Turbulence and Wave Propagation*, Nauka, Moscow, pp. 166–178.

Murata, T., Fukami, K. and Fukagata, K., 2019. "Nonlinear mode decomposition with machine learning for fluid dynamics".

Nagarajan, S., Lele, S.K. and Ferziger, J.H., 2003. "A robust high-order compact method for large eddy simulation". *Journal of Computational Physics*, Vol. 191, No. 2, pp. 392–419.

Nair, V. and Hinton, G.E., 2010. "Rectified linear units improve restricted boltzmann machines". In *Proceedings of the 27th International Conference on International Conference on Machine Learning*. Omnipress, USA, ICML'10, pp. 807–814.

Olson, B.J., 2012. *Large-eddy simulation of multi-material mixing and over-expanded nozzle flow*. Ph.D. thesis, Stanford University.

Parussini, L., Venturi, D., Perdikaris, P. and Karniadakis, G., 2017. "Multi-fidelity gaussian process regression for prediction of random fields". *Journal of Computational Physics*, Vol. 336, pp. 36 – 50.

Perdikaris, P., Venturi, D., Royset, J.O. and Karniadakis, G.E., 2015. "Multi-fidelity modelling via recursive co-kriging and gaussian&#x2013;markov random fields". *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Vol. 471, No. 2179, p. 20150018.

Ramírez, W. and Wolf, W.R., 2015. "Effects of trailing edge bluntness on airfoil tonal noise at low reynolds numbers". *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, pp. 1–12.

Ribeiro, J.H.M. and Wolf, W.R., 2017. "Identification of coherent structures in the flow past a naca0012 airfoil via proper orthogonal decomposition". *Physics of Fluids*, Vol. 29, No. 8, p. 085104.

Rowley, C.W., Colonius, T. and Murray, R.M., 2004. "Model reduction for compressible flows using POD and galerkin projection". *Physica D: Nonlinear Phenomena*, Vol. 189, No. 1–2, pp. 115 – 129.

Schmid, P.J., 2010. "Dynamic mode decomposition of numerical and experimental data". *Journal of Fluid Mechanics*, Vol. 656, p. 5–28.

Sieber, M., Paschereit, C.O. and Oberleithner, K., 2016. "Spectral proper orthogonal decomposition". *Journal of Fluid Mechanics*, Vol. 792, p. 798–828.

Sun, L., Gao, H., Pan, S. and Wang, J.X., 2020. "Surrogate modeling for fluid flows based on physics-constrained deep learning without simulation data". *Computer Methods in Applied Mechanics and Engineering*, Vol. 361, p. 112732.

Taira, K., Brunton, S.L., Dawson, S.T.M., Rowley, C.W., Colonius, T., McKeon, B.J., Schmidt, O.T., Gordeyev, S., Theofilis, V. and Ukeiley, L.S., 2017. "Modal analysis of fluid flows: An overview". *AIAA Journal*, Vol. 47, pp. 1–28.

Taira, K., Hemati, M., Brunton, S., Sun, Y., Duraisamy, K., Bagheri, S., Dawson, S. and Yeh, C.A., 2019. "Modal analysis of fluid flows: Applications and outlook". *AIAA Journal*, Vol. 58, pp. 1–25.

Towne, A., Schmidt, O.T. and Colonius, T., 2018. "Spectral proper orthogonal decomposition and its relationship to dynamic mode decomposition and resolvent analysis". *Journal of Fluid Mechanics*, Vol. 847, p. 821–867.

Tu, J.H., Rowley, C.W., Luchtenburg, D.M., Brunton, S.L. and Kutz, N.J., 2014. "On dynamic mode decomposition: Theory and applications". *Journal of Computational Dynamics*, Vol. 1, pp. 391–421.

Wolf, W.R., Azevedo, J.L.F. and Lele, S.K., 2012. "Convective effects and the role of quadrupole sources for aerofoil aeroacoustics". *Journal of Fluid Mechanics*, Vol. 708, pp. 502–538.

Xavier, G. and Yoshua, B., 2010. In Y.W. Teh and M. Titterington, eds., *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. PMLR, Chia Laguna Resort, Sardinia, Italy, Vol. 9 of *Proceedings of Machine Learning Research*, pp. 249–256.

Yu, M., Huang, W.X. and Xu, C.X., 2019. "Data-driven construction of a reduced-order model for supersonic boundary layer transition". *Journal of Fluid Mechanics*, Vol. 874, p. 1096–1114.

7. RESPONSIBILITY NOTICE

The authors Hugo F. S. Lui and William R. Wolf are the only responsible for the printed material included in this paper.