33rd Symposium on Naval Hydrodynamics
Osaka, Japan, 31 May-5 June 2020

Turbulent wake prediction using deep convolutional neural networks

B. Font[1,2], G.D. Weymouth[1,3], V.-T. Nguyen[2] and O.R. Tutty[1]
([1]University of Southampton, Southampton, UK, [2]Institute of High Performance Computing, A*STAR, Singapore, [3]Alan Turing Institute, London, UK)

ABSTRACT

A machine-learning based closure is explored for the prediction of the turbulent wake of flow past a circular cylinder at a high Reynolds number. We show that classic turbulence closures based on the turbulent-viscosity hypothesis are not capable of modelling the non-linear relationship between the mean quantities and the target turbulent fields. Instead, different multiple-input multiple-output auto-encoder convolutional neural networks are explored in this work to develop a data-driven closure. A detailed hyper-parameter study is completed including network architecture, loss functions and input sets, among others. A-priori results show 80% to 90% correlation coefficients between target and predicted turbulent fields of previously unseen data. High correlation coefficients are rapidly achieved by networks with a large number of trainable parameters, whereas smaller networks require more training epochs. The integration of the model in live simulations is theoretically discussed from its stability standpoint as well as preliminary physics-based constraints ideas to provide more stable data-driven closures.

INTRODUCTION AND OBJECTIVES

Complex non-linear dynamical systems such as turbulent flows have pushed researchers to explore new data-driven modelling techniques. The use of machine-learning (ML) methods applied in computational fluid dynamics has increased over the last few years (Kutz, 2017). The high flexibility and dimensionality of deep-learning models offers a feasible solution to the limited applicability range of classical turbulence models based on the Boussinesq's turbulent-viscosity hypothesis.

Recent work has successfully used neural networks for the prediction of turbulent fields. Multilayer perceptron (MLP) neural networks have been used to provide closure for Reynolds-averaged Navier-Stokes (RANS) and Large Eddy Simulation (LES) systems. Ling et al. (2016) used a tensor-based MLP neural network (TBNN) for the prediction of the normalised anisotropy Reynolds stress tensor. The latter was assumed to be recoverable by a combination of the mean rate of strain and mean rate of rotation tensors, and their resulting tensor invariants, which embeds Galilean invariance into the model. Gamahara and Hattori (2017) also used a MLP neural network to model the subgrid scales (SGS) in a LES formulation obtaining similar results to the Smagorinsky model for different sets of input quantities. Maulik et al. (2018) used a neural network to classify points in the domain which required to be modelled and, based on such classification, a blend of different turbulence modelling closures could be derived.

Beyond MLP architectures, convolutional neural networks (CNNs) perform well on the recognition and mapping of spatial patterns between known and target quantities (LeCun et al., 1989). Hidden correlations can be detected by the convolutional kernels (or filters) leading to feature maps, an operation that can be sequentially stacked across multiple layers. An added benefit is the translational invariance of the model intrinsically carried by the convolution operation. CNNs have been used to provide closure for the SGS tensor components (Beck et al., 2019) and for the prediction of turbulent heat transfer (Kim and Lee, 2020), among others. Simulations fully based on recursive flow fields predictions of flow past a circular cylinder at different Reynolds numbers were conducted by Lee and You (2019), where the performance of CNNs and generative adversarial networks (GANs) was assessed as well as the inclusion of physical constraints in the loss function. Positive a-priori results have been reported in all these investigations, however, the error propagation arising from the noisy predictions within an a-posteriori framework is currently a topic under investigation. For an extended review on machine-learning turbulence models the reader is referred to Duraisamy et al. (2019), Brenner et al. (2019), and Brunton et al. (2020).

In this work, incompressible viscous flow past a circular cylinder at a high Reynolds number (Re) is

considered. This is a canonical test case of wall-generated turbulence and flow separation on which classical turbulence models perform poorly (Nguyen and Nguyen, 2016). Its multiple ocean engineering applications such as marine risers, tow and mooring cable systems, tall pillars, etc, makes it an attractive candidate for the investigation of new turbulence modelling techniques. This work investigates a fully data-driven method using CNNs to provide a closure for the spanwise-averaged two-dimensional (2-D) governing equations. The network architecture, input data, loss function and other hyper-parameters are investigated.

The paper is structured as follows: The following section provides information of the dataset generation such as the numerical solver employed, computational details of the test case, and the definition of the turbulent fields aimed to be predicted. The CNN architecture and its relevant parameters are also detailed. Next, a-priori results for training and testing data are both quantitatively and qualitatively shown and analysed. Finally, the inclusion of physics-based constraints in the loss function is theoretically discussed as well as the error propagation of the model when deployed in live simulation.

METHODOLOGY

Dataset generation

A high-fidelity dataset is firstly generated using implicit Large Eddy Simulations (iLES) of incompressible viscous flow past a circular cylinder at $Re = 10^4$. The continuity equation and the Navier-Stokes momentum equations, which can be respectively written in their non-dimensional form as,

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

$$\partial_t \boldsymbol{u} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\nabla p + Re^{-1} \nabla^2 \boldsymbol{u}, \tag{2}$$

where $\boldsymbol{u}(\boldsymbol{x},t)$ is the velocity vector field and $p(\boldsymbol{x},t)$ is the pressure field, are solved in a rectilinear grid using the boundary data immersion method (BDIM) (Weymouth and Yue, 2011). Briefly, BDIM is an immersed boundary method that maps the governing equations for both the fluid and the solid domains into a single meta-equation using the distance to the fluid-solid boundary as a weighting function between the mediums.

Our finite volume solver is second-order accurate in space (quadratic upstream interpolation for convective kinematics, a.k.a. QUICK, scheme) and second-order accurate in time (predictor-corrector scheme). The solver has been recently used for this same test case in Font et al. (2019), where an extensive validation is available. The rectilinear grid is formed with a uniform fine grid for the close and near wake regions and a stretched grid for the far field regions, as depicted in figure 1. A resolution of 90 cells per diameter (D) is used in the uniform domain. A uniform velocity profile is used as the inlet boundary condition, a natural convection condition is used for the outlet boundary, no penetration slip conditions are used for the top and bottom boundaries, a periodic condition is used for the spanwise direction boundaries, and the no-slip condition is enforced on the cylinder boundary.

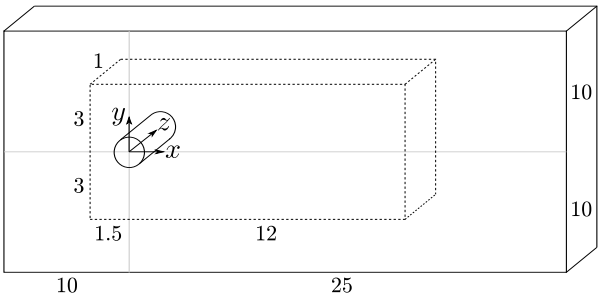


Figure 1: Computational domain. Distances are non-dimensionalised with the cylinder diameter D. A uniform grid (discontinuous lines) is used near the cylinder. A stretched grid (solid lines) is used from the uniform grid to the domain boundaries.

A flow decomposition based on spanwise-averaged and spanwise-fluctuating quantities is considered. In particular, the flow fields are decomposed as $q = \langle q \rangle + q'$, where the prime superscript denotes the spanwise-fluctuating quantity and the angle brackets denote the spanwise average operator along the cylinder span (L_z), which can be written as,

$$Q = \langle q \rangle (x,y,t) = \frac{1}{L_z} \int_0^{L_z} q(x,y,z,t)\, \mathrm{d}z. \tag{3}$$

Applying this decomposition and averaging operation to the governing equations yields the divergence of the spanwise-stress residual tensor ($\nabla \cdot \boldsymbol{\tau}_{ij}^R$) as the unclosed term for the 2-D spanwise-averaged Navier-Stokes equations when spanwise-periodicity is considered. Such term can be written as,

$$\nabla \cdot \boldsymbol{\tau}_{ij}^R = \nabla \cdot \begin{pmatrix} \langle u'u' \rangle & \langle u'v' \rangle \\ \langle u'v' \rangle & \langle v'v' \rangle \end{pmatrix} = \begin{pmatrix} \partial_x \langle u'u' \rangle + \partial_y \langle u'v' \rangle \\ \partial_x \langle u'v' \rangle + \partial_y \langle v'v' \rangle \end{pmatrix}. \tag{4}$$

It is worth noting that the unclosed term $\nabla \cdot \boldsymbol{\tau}_{ij}^R$ needs to account for the discretisation error of the resolved spanwise-averaged quantities in the derived 2-D system, as pointed out in Beck et al. (2019) for an analogous LES formulation. Hence, instead of directly computing $\nabla \cdot \boldsymbol{\tau}_{ij}^R$ from the primitive fluctuating velocity field ($\boldsymbol{u}'$) in the three-dimensional (3-D) system, the unclosed term is computed as the residual ($\mathscr{S}^R$) between the spanwise-averaged 3-D spatial operator $\langle \mathscr{S}(\boldsymbol{u}, p) \rangle$

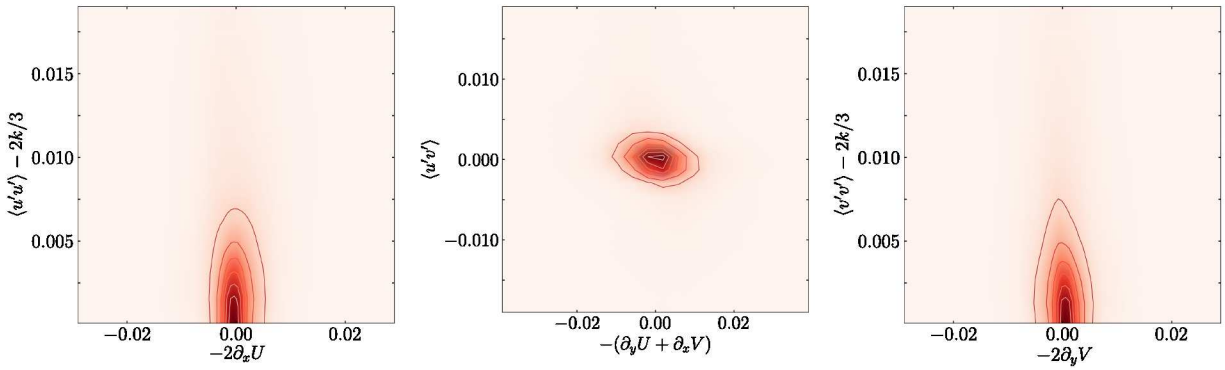


Figure 2: Density plots of resolved and fluctuating quantities computed within the domain $x \in [0,12], y \in [-2,2]$.

and the 2-D spatial operator of the spanwise-averaged quantities $\tilde{\mathscr{S}}(\boldsymbol{U},P)$ as follows,

$$\left\langle \partial_t \boldsymbol{u} + \boldsymbol{u}\cdot\nabla\boldsymbol{u} = -\nabla p + Re^{-1}\nabla^2\boldsymbol{u} \right\rangle, \tag{5}$$

$$\langle \partial_t \boldsymbol{u} \rangle + \left\langle \boldsymbol{u}\cdot\nabla\boldsymbol{u} + \nabla p - Re^{-1}\nabla^2\boldsymbol{u} \right\rangle = 0, \tag{6}$$

$$\partial_t \boldsymbol{U} + \langle \mathscr{S}(\boldsymbol{u},p) \rangle = 0, \tag{7}$$

$$\partial_t \boldsymbol{U} + \boldsymbol{U}\cdot\nabla\boldsymbol{U} = -\nabla P + Re^{-1}\nabla^2\boldsymbol{U} - \nabla\cdot\boldsymbol{\tau}^R_{ij}, \tag{8}$$

$$\partial_t \boldsymbol{U} + \tilde{\mathscr{S}}(\boldsymbol{U},P) = -\nabla\cdot\boldsymbol{\tau}^R_{ij}, \tag{9}$$

now redefining the divergence of the SSR tensor as the residual between $\tilde{\mathscr{S}}$ and $\langle\mathscr{S}\rangle$,

$$\partial_t \boldsymbol{U} + \tilde{\mathscr{S}}(\boldsymbol{U},P) = \tilde{\mathscr{S}}(\boldsymbol{U},P) - \langle \mathscr{S}(\boldsymbol{u},p) \rangle \equiv \mathscr{S}^R(\boldsymbol{u}'), \tag{10}$$

where the spanwise-averaging operator is assumed to commute with the time derivative (∂_t) and spatial derivatives (∂_x, ∂_y). Note that because $\tilde{\mathscr{S}}(\boldsymbol{U},P)$ cancels out in equation 10 , this formulation perfectly recovers the spanwise-averaged flow (equation 7) regardless of the spatial discretisation of the resolved quantities.

Turbulence closure

We now turn the focus on finding a closure based on the resolved quantities for equation 10, which effectively is a 2-D Navier-Stokes equations system with an additional source term. A-priori, not much physical information is available for the turbulent spanwise stresses. These represent the information lost during the spanwise-average operation applied to the governing equations and it is intrinsically different from the Reynolds stresses in the RANS formulation (time-averaging) or the SGS stresses in the LES formulation (filtering). The inclusion of the spanwise stresses in the 2-D governing equation allows to recover the spanwise-averaged solution of the flow. Such solution is logically related to the original 3-D flow, hence it can be argued that the spanwise stresses modify the system dynamics from 2-D to 3-D. This has a strong implication for turbulent flows, where opposite dynamics are present for 2-D (inverse energy cascade) and 3-D (direct energy cascade) systems.

The success of classic turbulence closures, such as the turbulent-viscosity hypothesis, tends to be case-dependent hence the model constants need to be readjusted according to the nature of the turbulent flow. The turbulent-viscosity hypothesis can be written as,

$$-\langle \boldsymbol{u}' \otimes \boldsymbol{u}' \rangle = \nu_t \left(\nabla \otimes \boldsymbol{U} + \boldsymbol{U} \otimes \nabla\right) - \frac{2}{3} k \delta_{ij}, \tag{11}$$

where ν_t is the turbulent viscosity, k is the turbulent kinetic energy (TKE) and δ_{ij} is the Kronecker delta tensor. This turbulence model requires further information on ν_t, an scalar field, to provide a closed system of equations. Even adjusting model constants arising in the definition of ν_t, it is well-known that the turbulent-viscosity hypothesis yields poor results for flows containing strong gradients, such as shear flows. It assumes that the anisotropic turbulent stress tensor components are aligned with the components of the mean rate of strain tensor, an assumption that fails even for simple shear flows (Pope, 2000, p. 94).

For incompressible 2-D flow, the relationship between the turbulent stresses and the mean rate of strain tensor arising in the turbulent-viscosity hypothesis (equation 11) can be written as,

$$\langle u'u' \rangle - 2k/3 = -2\nu_t \partial_x U, \tag{12}$$

$$\langle u'v' \rangle = -\nu_t \left(\partial_y U + \partial_x V\right), \tag{13}$$

$$\langle v'v' \rangle - 2k/3 = -2\nu_t \partial_y V, \tag{14}$$

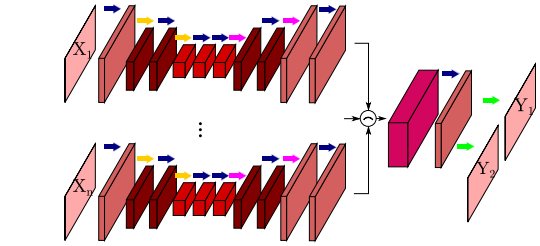
where $k = 0.5\left(\langle u'u' \rangle + \langle v'v' \rangle\right)$. The density plot between turbulent stresses and resolved quantities is displayed in figure 2, where a weak correlation can be observed. Quantitatively, the Pearson correlation coefficient has been computed and averaged over 80

snapshots taken throughout 8 wake cycles. A correlation coefficient of 0.16 and 0.26 is found for the normal streamwise and normal crossflow stresses, respectively. The wake anisotropy might be responsible for the different correlation values among the normal stresses. Additionally, a correlation coefficient of 0.23 is found for the shear stress. These observations are in agreement with the turbulent-viscosity hypothesis weaknesses previously discussed. More importantly, the nature of the spanwise turbulent stresses is different from the Reynolds stresses or the LES residual stresses, adding further complexity to the closure.

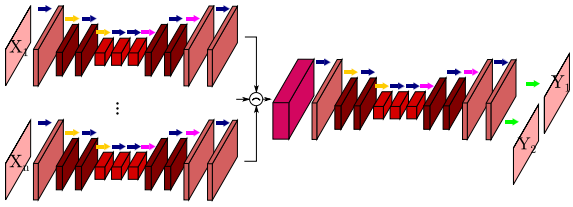
Data-driven model

The above weak correlation between turbulent stress and resolved quantities motivates the development of the proposed data-driven closure, one without a pre-defined turbulent stress vs resolved strain relationship. We focus on CNNs because they offer strong capabilities on finding hidden correlations between local spatial data. We consider different multiple-input multiple-output CNN architectures as shown in figure 3. All of the different CNNs are based on encoding the input information into some lower-dimensional latent space followed by a decoding procedure that maps such latent space into the target outputs (a.k.a. auto-encoders), similarly to SegNet (Badrinarayanan et al., 2015) or U-net (Ronneberger et al., 2015) architectures.

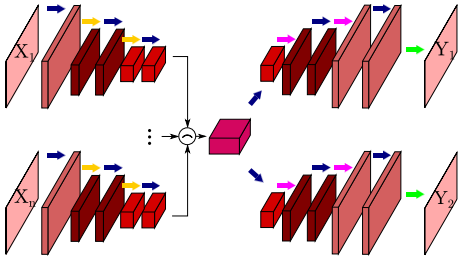
A dataset of 10152 snapshots containing the resolved quantities $\{U,V,P\}$ and unclosed quantities $\left\{\mathscr{S}_x^R,\mathscr{S}_y^R\right\}$ is generated throughout approximately 600 wake cycles. The size of the recorded flow fields is $1216\times 540\times 1$. The data is split 80-20 for training and testing, respectively. The training data is further split 75-25 for actual training (data used for optimization) and validation, respectively. The test dataset is generated after the 600 wake cycles, hence this data is completely hidden from the CNN during the training stage. Only the inner uniform domain is stored during the data generation process. For the training stage, an early-stopping approach is considered as the convergence criteria when the validation loss does not improve for 8 consecutive epochs (dataset iterations). Success is measured by the correlation coefficient between the target and predicted output fields, respectively $\mathscr{S}^R$ and $\mathscr{S}^{R,\mathrm{ML}}$, where the ML superscript denotes the CNN prediction. A mini-batch gradient descend method (4 to 6 samples depending on the GPU memory) using the Adam optimizer (Kingma and Ba, 2014) is employed. The learning rate is set to 10^{-4} with a decay rate of 10^{-5}. The full model is developed under the Keras deep-learning Python library (using the Tensorflow backend) (Chollet et al., 2015).



(a) Model 1 (M1)



(b) Model 2 (M2)

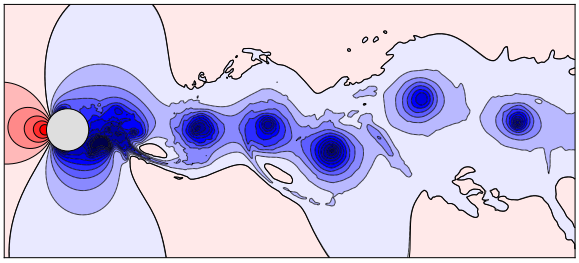


(c) Model 3 (M3)

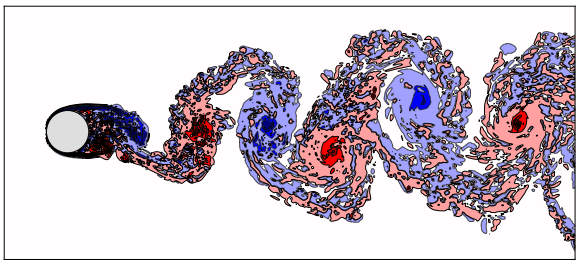
Figure 3: CNN architectures investigated. The input (X_n) and output (Y_n) fields size is $1216\times 540\times 1$. Arrows color indicate layers of: Dark blue: Conv2D + batch normalization + activation function. Yellow: MaxPooling 2x2. Magenta: UpSampling 2x2. Green: Conv2D 1x1 + batch normalization + linear function. The thickness of each block (noted on the last dimension) represents the number of filters. On the encoding blocks, the number of filters is doubled at every convolutional layer, reaching a maximum number of filters just before the decoding block, where the operation is reversed halving the number of filters at every convolutional layer. The merging operator $\frown$ indicates concatenation in the last dimension, hence stacking filters from multiple blocks.

Because of the network weights are initialized randomly and stochastic gradient descent backpropagation is used to optimize the network, there is an irreducible level of randomness in the final results. Hence, the training process is repeated for cases converged into very shallow local minimum errors and only the best results have been reported.

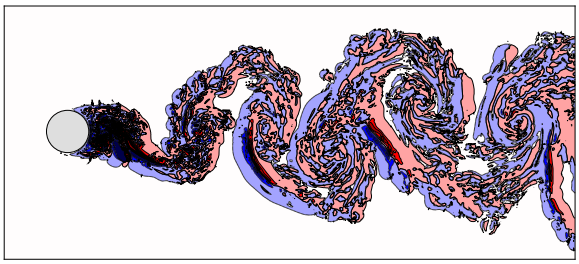
A flow field snapshot of resolved quantities (pressure, vorticity) and a target output ($\mathscr{S}_x^R$) is displayed in figure 4. The complex flow structures at the shear layer roll-up region behind the cylinder can be particularly challenging to model. This can indicate that a large CNN in terms of number of trainable parameters (adding weights $\underline{W}$ and bias $\boldsymbol{b}$) is required. Also, beyond exploring different CNN architectures, we test different loss functions, activation functions, number of filters and kernel size, effect of input data normalisation, and different input sets. A summary of the parametric study is detailed in table 1 and the base-case parameters are shown in table 2.



(a) P



(b) ω_z



(c) $\mathscr{S}_x^R$

Figure 4: Flow snapshot of resolved and target quantities.

Table 1: Parametric study. sse: sum squared error. sae: sum absolute error. ReLU: rectified linear unit. Ω_z: Spanwise-averaged vorticity. d: Distance function to the cylinder.

Architecture	M1, M2, M3
Loss function	sse, sae
Activation function	ReLU, tanh, sigmoid
Max. filters	16, 32, 64
Kernel size	$3\times3,\ 5\times5,\ 7\times7$
Normalize input data	False, True
Input set	$\{U,V,P\}$, $\{\Omega_z,d\}$, $\{\nabla\boldsymbol{U},\nabla P\}$

Table 2: Base-case parameters of the parametric investigation.

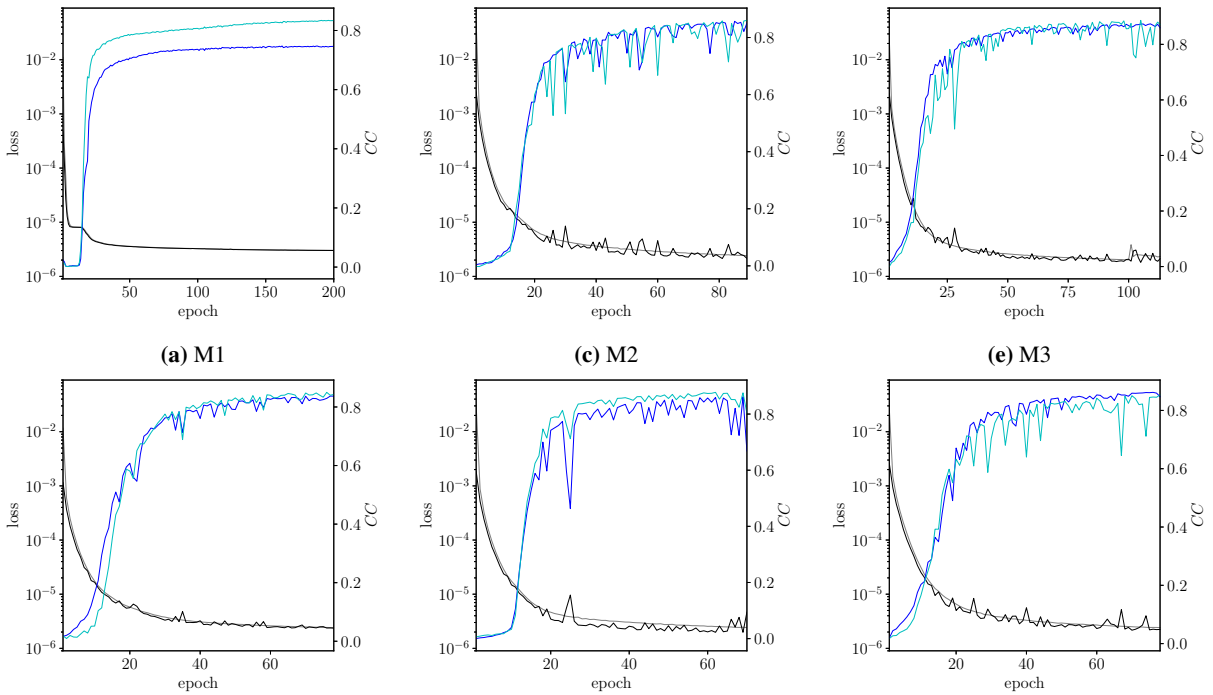
Architecture	M3
Loss function	sse
Activation function	ReLU
Max. filters	64
Kernel size	3×3
Normalize input data	True
Input set	$\{U,V,P\}$

RESULTS

Model architecture and data normalisation

We start investigating the CNN architecture based on three different models, namely M1, M2, and M3 (see figure 3). The base-case parameters are detailed in table 2. We also study the effect of normalizing each input data sample with its own mean and standard deviation: $\hat{x} = (x-\bar{x})/\sigma_x$.

Results for the training stage of each model are displayed in figure 5. It can be noted that normalizing the input data provides faster convergence. In addition, normalized input data presents an advantage in terms of generalizing the ML model to other flow configurations. The correlation coefficient results are provided in table 3. All models provide a high correlation coefficient for the validation and test datasets (hence no over-fitting is detected). The general trend is that accuracy increases with the network size. It is expected that output decoding branches having access to a larger latent space (number of filters encoding information on the encoder output), such as M3, are more flexible and perform better than networks with a smaller latent space. The M3 model with normalized input data is selected for all the subsequent analysis, unless specified otherwise. The choice of the M3



(a) M1 **(c)** M2 **(e)** M3

(b) M1, norm. **(d)** M2, norm. **(f)** M3, norm.

Figure 5: Training for the different model architectures. Top: input data not normalized. Bottom: input data normalized. Legend: grey: fit loss. black: validation loss. dark blue: $\mathscr{S}_x^R$ correlation coefficient (validation dataset). light blue: $\mathscr{S}_y^R$ correlation coefficient (validation dataset).

model is motivated by its training consistency and slightly faster convergence when compared to other models

Table 3: Validation and test data results for different models.

Case	Train. params.	val.: $CC(\mathscr{S}_x^R)$, $CC(\mathscr{S}_y^R)$	test: $CC(\mathscr{S}_x^R)$, $CC(\mathscr{S}_y^R)$	Epochs
M1	258242	0.75, 0.83	0.74, 0.83	192
M1 norm.	258242	0.84, 0.85	0.83, 0.84	70
M2	300194	0.85, 0.86	0.84, 0.84	81
M2 norm.	300194	0.86, 0.88	0.85, 0.88	62
M2 norm. stacked	58538	0.76, 0.77	0.74, 0.76	63
M3	338498	0.87, 0.88	0.86, 0.87	105
M3 norm.	338498	0.86, 0.85	0.85, 0.84	70

Additionally, a simple auto-encoder concatenating the resolved quantities in a single input branch (stacking them in the filter dimension, noted in table 3 as "stacked") has been tested. Even though this model is much smaller than the other ones in terms of number of trainable parameters, it still manages to provide acceptable predictions.

Loss function

Next, the loss function between predicted and target fields is investigated. The same previous base-case parameters are chosen and results are summarised in table 4.

Table 4: Validation and test data results for different loss functions.

Case	Train. params.	val.: $CC(\mathscr{S}_x^R)$, $CC(\mathscr{S}_y^R)$	test: $CC(\mathscr{S}_x^R)$, $CC(\mathscr{S}_y^R)$	Epochs
sse	338498	0.86, 0.85	0.85, 0.84	70
sae	338498	0.86, 0.57	0.83, 0.48	45

It can be observed that using the absolute error yields to a local minima where one of the outputs has not

converged successfully. The squared error function being an easier metric to differentiate (convex parabola) than the absolute error might be causing such difference.

Activation function

Activation functions (h) are used to dictate the output of a neuron based on some input value x (the model input or a previous layer output), a weight and a bias: $a = h(wx + b)$. Activation functions widely used in neural networks models are tested here. We consider the ReLU, tanh, and sigmoid activation functions. Results are displayed in table 5.

It can be observed that the ReLU activation function provides a clear advantage in comparison with the other tested functions. The ReLU function allows for faster trainings of deep neural networks because it preserves the neuron relative intensity across layers (maximum value is not capped) (Nair and Hinton, 2010). This is also due to the fact that the ReLU function helps mitigating vanishing-gradient caveats which arise more naturally when using the sigmoid function.

Table 5: Validation and test data results for different activation functions.

Case	Train. params.	val.: $CC(\mathscr{S}_x^R)$, $CC(\mathscr{S}_y^R)$	test: $CC(\mathscr{S}_x^R)$, $CC(\mathscr{S}_y^R)$	Epochs
ReLU	338498	0.86, 0.85	0.85, 0.84	70
tanh	338498	0.78, 0.76	0.77, 0.76	62
sigmoid	338498	0.81, 0.82	0.80, 0.82	36

Number of filters and kernel size

The number of filters per convolutional layer and the convolutional kernel size is explored. The results for the maximum number of filters per convolutional layer are displayed in table 6, where the number of trainable parameters differs by a factor of 4. The results observed in this particular study are strongly biased by the number of training epochs. It should be noted that the case using 16 filters only converged after almost 400 epochs, whereas the other cases are both under 100 epochs. The slow convergence of the 16 filters might be due to the lack of flexibility (model degrees of freedom) compared to the models with more filters available. Still, the 64 filters case displays correlation coefficients relatively close to the 16 filters case with only 70 epochs of training. Hence it can be argued that increasing the number of filters allows the model to be optimized based on a wider range of spatial features thus achieving faster convergence. Simultaneously, increasing the number of filters does not automatically provide better accuracy.

From a computational cost standpoint, the training time for the 64 maximum filters case is approximately 26 min./epoch, whereas the 32 and 16 filters cases are both around 23 min./epoch (the latter only measured during its first 100 epochs). Hence, the computational cost does not linearly increase with the network size. On the other hand, the computational training time of the 16 filters case gradually increases with every new epoch. For the 378 epochs until convergence, a total of 180 hours of computational time is required, averaging to roughly 28 min./epoch. Therefore, the computational time per epoch increases with the number of epochs, making the 16 maximum filters case less efficient compared to the larger network size cases.

Table 6: Validation and test data results for different maximum number of filters.

Case	Train. params.	val.: $CC(\mathscr{S}_x^R)$, $CC(\mathscr{S}_y^R)$	test: $CC(\mathscr{S}_x^R)$, $CC(\mathscr{S}_y^R)$	Epochs
64	338498	0.86, 0.85	0.85, 0.84	70
32	85154	0.81, 0.72	0.81, 0.71	34
16	21554	0.87, 0.87	0.87, 0.86	386

Table 7: Validation and test data results for different kernel sizes.

Case	Train. params.	val.: $CC(\mathscr{S}_x^R)$, $CC(\mathscr{S}_y^R)$	test: $CC(\mathscr{S}_x^R)$, $CC(\mathscr{S}_y^R)$	Epochs
3×3	338498	0.85, 0.86	0.86, 0.84	70
5×5	937282	0.89, 0.89	0.89, 0.89	101
7×7	1835458	0.87, 0.86	0.86, 0.86	34

The results for the kernel size effect are displayed in table 7. The kernel size is important for the detection of small and large-scale structures. The kernel size to capture the smallest scales in the wake should be, at least, a 3×3 filter since our iLES solver is resolving spatial scales down to grid-level. Structures larger than the grid-level scale might be better identified by slightly larger kernels. However, the non-linear combination of feature maps across the CNN layers allows small kernels to reconstruct such large-scale structures, hence the implied advantage of slightly larger kernels is not as relevant. On the other hand, large kernels will filter out information of the sub-kernel scale structures. This is why the current accepted practice in CNNs is to use either 3×3 or 5×5 kernels and our results agree with that practice.

The 5×5 kernel configuration provides the best prediction among the kernels set, where a compromise between the detection of small and large scales is achieved.

Input data

Last, different input sets are evaluated. Here, we consider the M2 model without normalising the inputs. This is motivated by the fact that the M3 model becomes very large when 6 different inputs are used, as it is the case with the $\{\nabla \boldsymbol{U}, \nabla P\}$ input set. The remaining base-case parameters are employed. By testing different inputs we seek to include features more closely related to the expected outputs so that the regression problem solved by the CNN becomes easier (Gamahara and Hattori, 2017; Beck et al., 2019).

As summarised in table 8, it is found that the resolved quantities set and the gradient set provide the highest correlation between target and predicted closure terms. On the other hand, the vorticity plus the distance function input set does not perform as well. The closure terms are, in their core, convective terms of the spanwise fluctuating velocity field. Hence, it is reasonable to argue that spatial derivatives of the resolved spanwise-averaged velocity field carry more relevant information to the closure terms than the other input sets. Still, the primitive resolved quantities set performs very similarly to the gradient set. This can be an indicator of the network being able to find the gradients on its own. Also, compared to the vorticity input set, it seems that information on the pressure field is important although this has not been tested in direct comparison.

Table 8: Validation and test data results for different input sets.

Case	Train. params.	val.: $CC(\mathscr{S}_x^R)$, $CC(\mathscr{S}_y^R)$	test: $CC(\mathscr{S}_x^R)$, $CC(\mathscr{S}_y^R)$	Epochs
$\{U,V,P\}$	338498	0.86, 0.85	0.85, 0.84	70
$\{\Omega_z, d\}$	255746	0.57, 0.52	0.56, 0.52	110
$\{\nabla \boldsymbol{U}, \nabla P\}$	433538	0.86, 0.87	0.85, 0.86	112

DISCUSSION

The M2 model has provided the highest correlations across the tested models for a given set of base-case parameters, and, on the other hand, the M3 model offered the best consistency while training. Also, normalising the input data allows for faster training convergence without significantly compromising accuracy. The sse loss function, a maximum of 64 filters, and a kernel size of 5×5 proved to be the best possible parameters. Finally, the gradient input set and the primitive resolved quantities set performed similarly providing the highest correlation coefficients.

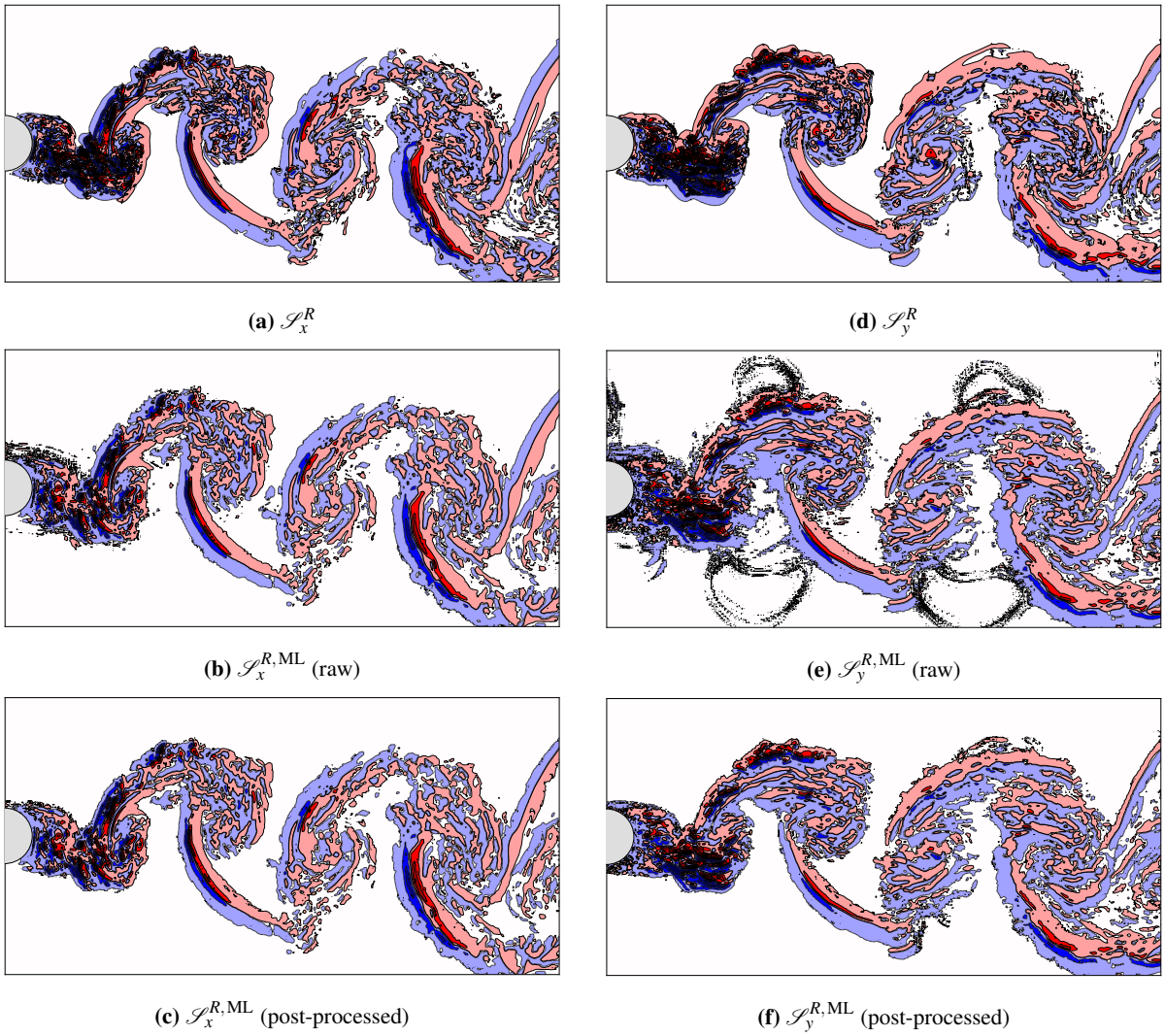
From a qualitative standpoint, predictions and target closure terms are displayed in Figure 6. The M3 model using the normalized primitive resolved quantities input set, a kernel size of 5×5, the sse loss function, and 64 maximum filters has been employed to generate the predicted closure terms (highest correlation values observed in the parametric study: $CC(\mathscr{S}_x^R) = 0.89, CC(\mathscr{S}_y^R) = 0.89$. The predicted fields have been post-processed including a wake mask step constructed from the resolved vorticity field and a Gaussian filter to mitigate the noise in the raw CNN output (note that all the presented correlation values do not include this post-processing step). It can be appreciated that both large and mid-scale structures are well captured by the model. On the other hand, small-scale structures are not so well identified. Similar issues are encountered in Lee and You (2019), where the deep-learning models employed for recursively generating snapshots of flow past a circular cylinder at high *Re* failed to fully reproduce small-scale structures.

Table 9: Summary of best results.

Architecture	M2 (highest correlation) M3 (best training consistency)
Loss function	sse
Activation function	ReLU
Max. filters	64
Kernel size	5×5
Normalise input data	True
Input dataset	$\{U,V,P\}$, $\{\nabla \boldsymbol{U}, \nabla P\}$

With a data-driven model providing almost 90% correlation on the closure terms, the next logical step is to incorporate the model outputs directly in a live 2-D simulation (i.e. direct modelling). However, as discussed in Beck et al. (2019), using the direct model is not likely to produce a stable closure. The error and noise incorporated in the predictions will affect the resolved quantities on the calculation of the next time step which, in turn, will provide a noisy input for the prediction of the next time step unclosed terms yielding to even worse predictions, and so on. The best direct model has been used in a 2-D simulation confirming the instability of the direct model closure.

Efforts to mitigate this issue were made by Beck et al. (2019) by projecting the learned closure terms onto a turbulent-viscosity model to benefit from their stability. In Lee and You (2019), stability improved when physics-based constraints were included into the loss function. In this work, the closure terms are



(a) $\mathscr{S}_x^R$ **(d)** $\mathscr{S}_y^R$

(b) $\mathscr{S}_x^{R,\mathrm{ML}}$ (raw) **(e)** $\mathscr{S}_y^{R,\mathrm{ML}}$ (raw)

(c) $\mathscr{S}_x^{R,\mathrm{ML}}$ (post-processed) **(f)** $\mathscr{S}_y^{R,\mathrm{ML}}$ (post-processed)

Figure 6: Qualitative comparison between a snapshot of target fields (top), CNN raw predictions (middle), and CNN post-processed predictions (bottom). The post-processing remove high-frequency oscillations with a Gaussian filter and applies a wake detection filter constructed from the resolved vorticity field.

residuals from the spanwise-averaging operation and, so far, applicable physical laws are still unknown. Only from the comparison between 2-D and 3-D spanwise-averaged simulations, it can be deduced that such terms are in charge of dissipating energy throughout the turbulent wake, being more important in the near-wake region. Hence, the inclusion of the energy dissipation rate in the loss function as a method to bound the model error will be investigated in the future. In this sense, Long-Short Term Memory (LSTM) neural networks can be used together with a CNN to recognise temporal dynamics on top of spatial patterns, thus facilitating the implementation of time-related constraints. Vlachas et al. (2018) used a LSTM recurrent neural network to capture long-term statistics of very chaotic systems. A similar architecture can be considered in this case to achieve convergence to an invariant measure, i.e. the energy dissipation rate of the closure terms.

CONCLUSIONS

A fully data-driven model to predict the instantaneous turbulent wake of flow past a circular cylinder at $Re = 10^4$ has been proposed. A spanwise average of the Navier-Stokes equations systems has been employed to generate 2-D closure terms accounting for the information lost during the averaging operation. The resulting residual closure terms have been modelled using an auto-encoder CNN. A parametric study of the CNN has been conducted in terms of network architecture, loss function, maximum

number of filters, kernel size and input data. It has been found that architectures embedding a larger number of trainable parameters yield higher correlation coefficients between predicted and target fields compared to smaller networks for the same number of training epochs. The selection of the loss function also has an impact on the model training, where the sse function has showed a better performance than the sae function. Different activation functions have also been investigated, with the ReLU function providing the best result. The number of filters used per convolutional layer and the kernel size has also been parametrised. Results show that high correlation coefficients can be obtained despite the number of filters. Also, an intermediate kernel size (5×5) yields the best results since a compromise between correctly capturing small and large-scale structures is achieved, as displayed in figure 6. Finally, the primitive resolved quantities input set and the gradients set provided similar correlation coefficients, whereas the vorticity plus distance function input set did not perform as well.

It has also been noted that directly including the model predictions in a live simulation results into an unstable closure because of the predictions error recursively propagating into the resolved quantities. Future work is set on the development of a stable closure based on the energy dissipation rate associated to the spanwise stresses. In this sense, LSTM CNNs shall be explored so that the temporal dynamics of the closure terms are included in the model.

ACKNOWLEDGEMENTS

The authors acknowledge the support of the Singapore Agency for Science, Technology and Research (A*STAR) Research Attachment Programme (ARAP) as a collaboration between the A*STAR Institute of High Performance Computing (IHPC) and the Faculty of Engineering and Physical Sciences of the University of Southampton. The authors also acknowledge the use of the IRIDIS High Performance Computing Facility, and associated support services at the University of Southampton, in the completion of this work. G.D.W. acknowledges the support from the Office of Naval Research Global award N62909-18-1-2091.

REFERENCES

Kutz, J. N., "Deep learning in fluid dynamics," Journal of Fluid Mechanics, vol. 814, pp. 1–4, 2017.

Ling, J., Kurzawski, A., and Templeton, J., "Reynolds averaged turbulence modelling using deep neural networks with embedded invariance," Journal of Fluid Mechanics, vol. 807, pp. 155–166, 2016.

Gamahara, M. and Hattori, Y., "Searching for turbulence models by artificial neural network," Physical Review Fluids, vol. 2, no. 5, 2017.

Maulik, R., San, O., Rasheed, A., and Vedula, P., "Subgrid modelling for two-dimensional turbulence using neural networks," Journal of Fluid Mechanics, vol. 858, pp. 122–144, 2018.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D., "Backpropagation applied to handwritten zip code recognition," Neural Computation, vol. 1, no. 4, pp. 541–551, 1989.

Beck, A., Flad, D., and Munz, C.-D., "Deep neural networks for data-driven LES closure models," Journal of Computational Physics, vol. 398, p. 108910, 2019.

Kim, J. and Lee, C., "Prediction of turbulent heat transfer using convolutional neural networks," Journal of Fluid Mechanics, vol. 882, p. A18, 2020.

Lee, S. and You, D., "Data-driven prediction of unsteady flow over a circular cylinder using deep learning," Journal of Fluid Mechanics, vol. 879, pp. 217–254, 2019.

Duraisamy, K., Iaccarino, G., and Xiao, H., "Turbulence modeling in the age of data," Annual Review of Fluid Mechanics, vol. 51, no. 1, pp. 357–377, 2019.

Brenner, M. P., Eldredge, J. D., and Freund, J. B., "Perspective on machine learning for advancing fluid mechanics," Physical Review Fluids, vol. 4, no. 10, 2019.

Brunton, S. L., Noack, B. R., and Koumoutsakos, P., "Machine learning for fluid mechanics," Annual Review of Fluid Mechanics, vol. 52, no. 1, pp. 477–508, 2020.

Nguyen, V.-T. and Nguyen, H. H., "Detached eddy simulations of flow induced vibrations of circular cylinders at high Reynolds numbers," Journal of Fluids and Structures, vol. 63, pp. 103–119, 2016.

Weymouth, G. D. and Yue, D. K. P., "Boundary data immersion method for Cartesian-grid simulations of fluid-body interaction problems," Journal of Computational Physics, vol. 230, no. 16, pp. 6233–6247, 2011.

Font, B., Weymouth, G. D., Nguyen, V.-T., and Tutty, O. R., "Span effect on the turbulence nature of flow past a circular cylinder," Journal of Fluid Mechanics, vol. 878, pp. 306–323, 2019.

Pope, S. B., Turbulent Flows. Cambridge University Press, 2000.

Badrinarayanan, V., Kendall, A., and Cipolla, R., "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," 2015. URL: https://arxiv.org/abs/1511.00561

Ronneberger, O., Fischer, P., and Brox, T., "U-net: Convolutional networks for biomedical image segmentation," 2015. URL: https://arxiv.org/abs/1505.04597

Kingma, A. P. and Ba, J., "Adam: A method for stochastic optimization," 2014. URL: https://arxiv.org/abs/1412.6980

Chollet, F. et al., "Keras," https://keras.io, 2015.

Nair, V. and Hinton, G. E., "Rectified linear units improve restricted boltzmann machines," in Proceedings of the 27th International Conference on International Conference on Machine Learning, ser. ICML'10, no. 9. Madison, WI, USA: Omnipress, 2010, p. 807–814.

Vlachas, P. R., Byeon, W., Wan, Z. Y., Sapsis, T. P., and Koumoutsakos, P., "Data-driven forecasting of high-dimensional chaotic systems with long short-term memory networks," Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences, vol. 474, no. 2213, p. 20170844, 2018.