Predictions of turbulent shear flows using deep neural networks

P. A. Srinivasan,[1,2,3] L. Guastoni,[1,3] H. Azizpour,[2,3] P. Schlatter,[1,3] and R. Vinuesa[1,3,*]

[1]*Linné FLOW Centre, KTH Mechanics, SE-100 44 Stockholm, Sweden*
[2]*School of Electrical Engineering and Computer Science, KTH, SE-100 44 Stockholm, Sweden*
[3]*Swedish e-Science Research Centre (SeRC), SE-100 44 Stockholm, Sweden*

In the present work, we assess the capabilities of neural networks to predict temporally evolving turbulent flows. In particular, we use the nine-equation shear flow model by Moehlis *et al.* [New J. Phys. **6**, 56 (2004)] to generate training data for two types of neural networks: the multilayer perceptron (MLP) and the long short-term memory (LSTM) networks. We tested a number of neural network architectures by varying the number of layers, number of units per layer, dimension of the input, and weight initialization and activation functions in order to obtain the best configurations for flow prediction. Because of its ability to exploit the sequential nature of the data, the LSTM network outperformed the MLP. The LSTM led to excellent predictions of turbulence statistics (with relative errors of 0.45% and 2.49% in mean and fluctuating quantities, respectively) and of the dynamical behavior of the system (characterized by Poincaré maps and Lyapunov exponents). This is an exploratory study where we consider a low-order representation of near-wall turbulence. Based on the present results, the proposed machine-learning framework may underpin future applications aimed at developing accurate and efficient data-driven subgrid-scale models for large-eddy simulations of more complex wall-bounded turbulent flows, including channels and developing boundary layers.

I. INTRODUCTION

Artificial neural networks are computational frameworks for learning specific tasks solely from examples, and they are a popular method within the general area of machine learning. They were inspired by their biological counterparts, which are organized as hierarchical layers of neurons as discovered by Hubel and Wiesel [1]. Although in existence for several decades, the recent success of deep neural networks[1] (DNNs) can be attributed to the increased computational power (mainly through graphics processing units or GPU) and the generation of large-scale data sets which benefit the training of the overparametrized DNNs. As discussed in the recent review by Duraisamy *et al.* [2], there have been several studies aimed at using machine learning to model turbulent flows, especially in the context of Reynolds-averaged Navier-Stokes (RANS) simulations [3,4]. Other applications to modeling the near-wall region of turbulent flows were reported by Milano and Koumoutsakos [5], while Lapeyre *et al.* [6] and Beck *et al.* [7] have documented the possibility of using machine learning in designing subgrid-scale (SGS) models for large-eddy simulation (LES). Machine learning has also been applied to flow control [8,9], development of low-dimensional models [10], generation of inflow conditions [11], or structure identification in two-dimensional

---

*[*]rvinuesa@mech.kth.se

[1]In this paper, the terms artificial neural networks, deep neural networks, and neural networks are used interchangeably.

decaying turbulence [12], and as discussed by Kutz [13] DNNs will progressively be more widely used in fluid mechanics in the coming years.

In this study, we assess the feasibility of using DNNs to predict the temporal dynamics of simple turbulent flows. If such predictions are satisfactory, then DNNs could be employed, for instance, as SGS models in LES of more complex cases. As a simplified yet fairly complete model for wall-bounded turbulence, we consider the nine-equation model of a shear flow between infinite parallel free-slip walls subjected to a sinusoidal body force developed by Moehlis *et al.* [14]. In this model, nine Fourier modes $\mathbf{u}_j(\mathbf{x})$ represent the mean profile, streamwise vortices, streaks and their instabilities, as well as the coupling among them. The instantaneous velocity fields in spatial coordinates $\mathbf{x}$ and time t are then given by superposition of these nine modes as

$$\tilde{\mathbf{u}}(\mathbf{x}, t) = \sum_{j=1}^{9} a_j(t)\mathbf{u}_j(\mathbf{x}). \tag{1}$$

Through Galerkin projection, a system of nine ordinary differential equations (ODEs) for the nine-mode amplitudes $a_j(t)$ is obtained. In this work, we considered a model Reynolds number based on channel full height $2h$ and laminar velocity U_0 at a distance of $h/2$ from the top wall equal to $\mathrm{Re} = 400$, and we used U_0 and h as velocity and length scales, respectively. The Fourier base modes $\mathbf{u}_j(\mathbf{x})$ are independent of the model Reynolds number, which defines the viscosity and therefore the dissipation of the problem. The ODE model was then employed to produce time series of the nine amplitudes, to be used as training and validation data sets. Note that, as discussed by Moehlis *et al.* [14], the lifetime of the turbulent state is highly sensitive to the initial conditions and domain size. We produced more than 10 000 different turbulent time series with a time span of 4000 time units (where the time unit is defined in terms of U_0 and h) and considered a constant time step between samples of 1 time unit. The following initial conditions were employed for the Fourier modes: $(a_1^0, a_2^0, a_3^0, a_5^0, a_6^0, a_7^0, a_8^0, a_9^0) = (1, 0.07066, -0.07076, 0, 0, 0, 0, 0)$, as in the work by Kim [15] (here the superscript 0 denotes a value at $t = 0$). The value of a_4^0 was randomly perturbed around 0, and if the generated series reached a fixed point or a periodic orbit then it was discarded. According to Moehlis *et al.* [14], higher model Reynolds numbers lead to longer lifetimes of the perturbation with respect to the laminar profile (a result consistent with the lower dissipation). However, in this study we consider only time series that are turbulent over the whole time span, and for this reason the present analysis would not depend on the particular model Reynolds number under consideration. We considered a domain with $L_x = 4\pi$, $L_y = 2$, and $L_z = 2\pi$, where x, y, and z are the streamwise, wall-normal, and spanwise coordinates. In this study, we assess flow predictions obtained with two different types of neural network and use the machine learning software framework developed by Google Research called TENSORFLOW [16].

This paper is organized as follows: In Sec. II, we describe the characteristics of the multilayer perceptron (MLP) and report the results obtained with this type of network; in Sec. III, we provide a description of the long short-term (LSTM) networks and discuss their performance both in terms of turbulence statistics and dynamical behavior of the system; and finally, in Sec. IV, we summarize the main conclusions of the study.

II. PREDICTIONS BASED ON MULTILAYER PERCEPTRONS (MLPS)

First, we consider multilayer perceptrons (MLPs) [17], which are the most basic type of artificial neural network. As shown schematically in Fig. 1, they consist of two or more layers of nodes (also called neurons), with each node connected to all nodes in the preceding and succeeding layers. Neural network training is based on the back-propagation algorithm [18], which operates layer by layer and updates the parameters in each layer's nodes to improve the predictions on the training data using gradient-following methods. If X and Ψ are input and output spaces and each pair of vectors $(\boldsymbol{\chi}, \boldsymbol{\psi})$ is a training example or sample, the objective of the neural network is to find the mapping $f : X \to \Psi$ using the training set such that a loss function $L(f(\boldsymbol{\chi}); \boldsymbol{\psi})$ is minimized. For
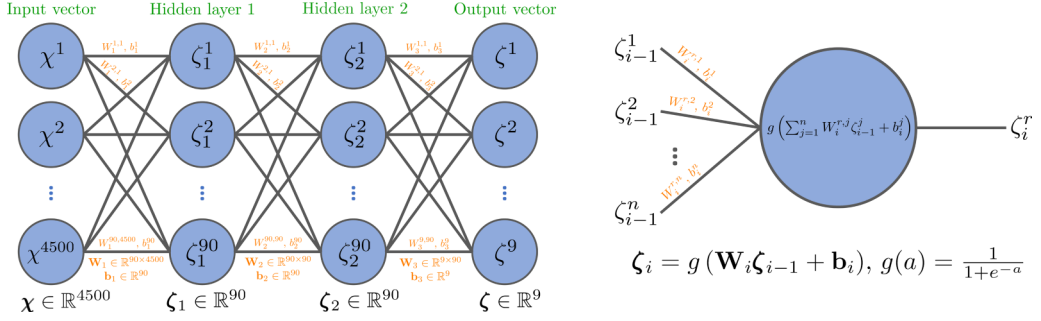
FIG. 1. (Left) Schematic representation of a multilayer perceptron (MLP) with two hidden layers, i.e., $l = 2$, where each circle in the diagram depicts a single neuron. The MLP takes as input a d-dimensional vector $\boldsymbol{\chi}$, then produces hidden n-dimensional vectors $\boldsymbol{\zeta}_i$ for each of the two hidden layers, and finally generates an output vector $\boldsymbol{\zeta}$ with m dimensions. Note that the dimension of the input depends on the number of previous temporal values used for the prediction p. In this example, $p = 500$, $d = 9p = 4500$, $m = 9$, and $n = 90$, which implies that 90 neurons are used for each of the hidden layers. The transformation at a certain layer i is performed with the model parameters given by the weight matrix $\mathbf{W}_i$. The $\mathbf{W}_i$ values are determined through learning using a gradient-following optimization technique on training data. (Right) Detail of a neuron, which takes a vector as input and generates a scalar output by performing a vector dot product and a nonlinear operation given by the activation function g.

an MLP, the mapping $f(\boldsymbol{\chi})$ can be decomposed into a sequence of simple transfer functions, i.e., linear matrix transform followed by an element-wise nonlinear function called activation function (e.g., sigmoidal). Figure 1 summarizes the basic structure of an MLP, where l is the number of hidden layers, n is the number of neurons per hidden layer (which in our case will be constant for all the hidden layers), and p is the number of previous values used to predict the next one. The evaluation of $\boldsymbol{\zeta} = f(\boldsymbol{\chi})$ is done using Algorithm 1, which involves the following parameters for each layer i: g_i are the activation functions (which filter the contributing units in each layer), whereas the weight matrices $\mathbf{W}_i$ and biases $\mathbf{b}_i$ perform a linear transformation from one hidden layer space to the next. The goal of training an MLP is to determine $\mathbf{W}_i$ and $\mathbf{b}_i$ using training data, for the given activation functions. The back-propagation algorithm used to update the weights and biases involves the use of gradient descent, for which a so-called learning rate (step size) needs to be defined. The learning rate can be manually set to a constant value or allowed to decay over time, since an excessively large learning rate may prevent convergence. For all the training runs in this work, a variant of the stochastic gradient descent algorithm called adaptive moment estimation (Adam) [19] is used. Adam has an adaptive learning rate method which is commonly used to train deep networks.

The updates for the weights and biases in the algorithm depend on the choice of the loss function, and a standard choice for regression problems is the mean squared error. Since training a neural

---

Algorithm 1. Compute MLP output.

Input: $\boldsymbol{\chi}$
Output: $\boldsymbol{\zeta} = f(\boldsymbol{\chi})$
set $\boldsymbol{\zeta}_0 \leftarrow \boldsymbol{\chi}$
for $i \leftarrow 1$ **to** $l + 1$ **do**
    $\mathbf{h}_i \leftarrow \mathbf{W}_i \boldsymbol{\zeta}_{i-1} + \mathbf{b}_i$
    $\boldsymbol{\zeta}_i \leftarrow g_i(\mathbf{h}_i)$
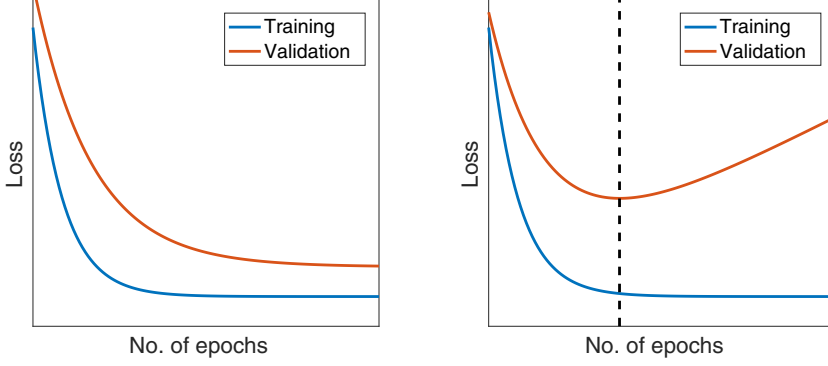return $\boldsymbol{\zeta} \leftarrow \boldsymbol{\zeta}_{l+1}$

FIG. 2. Schematic representation of the training and validation losses as a function of the number of epochs. (Left) Adequate fitting, where the validation loss does not increase with the number of epochs, and (right) overfitting beyond the dashed vertical line (which indicates the point for early stopping).

network is essentially fitting a mathematical function to data, it inherently has a risk of overfitting. An overfitted network is one that has achieved very low losses and performs well on the training set but poorly on new data. To overcome this problem, we use a regularization technique which penalizes too large parameters based on their L_2 norm. It is also common to set aside a small portion of the training set as a validation set which will not be used during training. While training, one complete pass through the training set is called an epoch. At the end of each epoch, the model is evaluated using the validation set and a validation loss is calculated using the loss function. A common method to avoid overfitting is called early stopping, whereby the training is stopped when the validation loss starts increasing. This method is illustrated in Fig. 2. Other relevant decisions to be made when training deep learning models are the strategy for weight initialization and the choice of activation functions. Since the MLP architecture has a large number of weights, the training algorithm seeks to find a local minimum in a high-dimensional space, and the rate of convergence of this search, among other factors, depends on the initial values of these weights. Although randomly selecting initial weights based on a uniform distribution is a natural choice, there are a few other options that may perform better. For instance, since training a neural network involves a forward pass coupled with back-propagation, it is generally better to prevent successive enlargement or shrinkage of layer outputs. Hence, initializations based on a suitable normal distribution perform better than those based on uniform distributions. A truncated normal distribution, which bounds the random variable both above and below, is a commonly used distribution for most learning tasks. Other variations based on the normal distribution, showing better convergence properties than the truncated normal, were proposed by Glorot and Bengio [20] and He *et al.* [21]. Regarding the activation function, the most widely used is the hyperbolic tangent, although other alternatives (see, for instance, the work by LeCun *et al.* [22]) such as the rectified linear unit (ReLU) [23], the Leaky ReLU [24], and the exponential linear unit (eLU) [25] are also reported in the literature.

For the predictions of the near-wall turbulence model using MLPs, we consider the mean squared error as loss function and define the validation loss as

$$L(f(\boldsymbol{\chi}); \boldsymbol{\psi}) = \frac{1}{2m} \sum_{j=1}^{m} |\psi^j - f(\chi^j)|^2, \tag{2}$$

where $\boldsymbol{\psi}$ and $f(\boldsymbol{\chi})$ are the m-dimensional vectors containing the true and predicted values, respectively. We use 20% of the training data as a validation set and consider early stopping to avoid overfitting. The various initialization strategies were compared in the study by Srinivasan [26], and the one by Glorot and Bengio [20] was observed to outperform all the others. Therefore, that specific initialization is employed in this study. Srinivasan [26] also compared the various

TABLE I. Relative errors obtained when predicting a training data set ε_1 (%).

p	$l = 1$	$l = 2$	$l = 3$	$l = 4$	$l = 5$
10	23.34	33.66	25.37	26.01	23.81
50	28.28	20.59	17.66	16.77	22.95
100	18.97	13.58	12.76	17.17	14.42
500	65.21	3.32	6.80	1.43	3.09

activation functions. The hyperbolic tangent provided the best agreement with reference data, and therefore it was employed for the MLP predictions reported below. The next step to designing an MLP architecture is to decide the number of hidden layers l, the number of units per hidden layer n, and the dimensionality of the input d, which is the number of predicted quantities (nine in our case) multiplied by the number of previous values from each amplitude a_j used to predict the next one, denoted by p. We will start by assessing the value of p required for the predictions, and to do it we recall that, as indicated in Fig. 2, a properly trained neural network must be able to first reproduce the training data (which has been seen by the network during training), before attempting to reproduce the validation data (not seen during training). We train the MLP with a single time series of 4000 time units for several epochs and then predict the same data set. For this test, we maintain a constant number of units per hidden layer of $n = 90$ and vary the values of p and l. We define the relative error in the prediction of amplitude a_1 (over N_s samples) as

$$\varepsilon_1 = \frac{1}{(N_s - p)a_{1,\text{lam}}} \sum_{j=p+1}^{N_s} \left| a_{1,\text{tra}}^j - a_{1,\text{pred}}^j \right|, \tag{3}$$

where "tra" and "pred" denote data from the training data set (i.e., seen by the network during training) and data predicted by the neural network (NN), respectively, and $a_{1,\text{lam}} = 1$ is the value for laminar flow. The values of ε_1 for the various cases are summarized in Table I, which shows that, for the data under consideration in this study, it is necessary to consider a value of $p = 500$ in order to obtain acceptable training losses, i.e., below 10% when at least two hidden layers are considered.

Based on the results discussed above, we will consider in all the MLP cases a value of $p = 500$, which leads to a reasonably large input dimension of $d = 4500$, while the output dimension is $m = 9$. In order to select the rest of parameters, five different MLP architectures with various values of n and l were evaluated, as summarized in Table II. After using 1000 data sets (each of them consisting of a time series spanning 4000 time units) to train the MLP, the predicted mode amplitudes were employed to reconstruct the velocity fields. These fields were used to calculate the mean velocity profile $\overline{u}(y)$ and the streamwise velocity fluctuations $\overline{u^2}(y)$, based on averaging over the two periodic directions x and z, in time over the 4000 time units of a time series, and then performing an ensemble average over 500 different time series, which were sufficient to ensure statistical convergence [26]. These profiles were compared with reference statistics obtained from the Moehlis et al. [14] model, also based on 500 time series which had not been seen by the MLP during training. We define the

TABLE II. Summary of MLP architectures and their performance using 1000 training data sets.

Architecture	l	n	$E_{\overline{u}}$ [%]	$E_{\overline{u^2}}$ [%]	Validation loss
MLP1	4	45	1.84	24.91	3.96×10^{-5}
MLP2	3	90	10.96	36.16	4.38×10^{-5}
MLP3	4	90	7.00	29.04	3.90×10^{-5}
MLP4	5	90	3.21	18.61	3.84×10^{-5}
MLP5	4	180	5.87	27.85	3.99×10^{-5}

Algorithm 2. Compute the output sequence of an RNN.

Input: Sequence $\boldsymbol{\chi}_1, \boldsymbol{\chi}_2, \ldots \boldsymbol{\chi}_p$
Output: Sequence $\boldsymbol{\zeta}_1, \boldsymbol{\zeta}_2, \ldots \boldsymbol{\zeta}_p$
set $\mathbf{h}_0 \leftarrow 0$
for $t \leftarrow 1$ **to** p **do**
 $\mathbf{h}_t \leftarrow g_h(\mathbf{W}_{h\chi}\boldsymbol{\chi}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h)$
 $\boldsymbol{\zeta}_t \leftarrow g_\zeta(\mathbf{W}_{\zeta h}\mathbf{h}_t + \mathbf{b}_\zeta)$

---

relative error between the model and the MLP prediction (denoted by the subindices "mod" and "pred," respectively) for the mean flow as

$$E_{\overline{u}} = \frac{1}{2\max(\overline{u}_{\text{mod}})} \int_{-1}^{1} |\overline{u}_{\text{mod}} - \overline{u}_{\text{pred}}| dy, \tag{4}$$

where the normalization with the maximum of $\overline{u}$ is introduced to avoid spurious error estimates close to the centerline where the velocity is 0. This error is defined analogously for $\overline{u^2}$, and both are reported in Table II for each of the architectures, together with the validation loss defined above. These results show that in general MLPs with higher numbers of hidden layers l lead to better predictions of the amplitudes and also of the turbulence statistics, whereas increasing n does not always lead to better predictions. The best MLP architecture among the ones under study was MLP4, with five hidden layers and a total of 90 neurons per layer. Although the agreement with the reference mean velocity profile obtained with MLP4 is acceptable (with a relative error of 3.21%), the streamwise velocity fluctuation profile exhibits a much larger deviation with 18.61% relative error. Given the relatively large size of the network, and particularly the high input dimension, we explored alternative DNNs which are able to better exploit the sequential nature of the data, as discussed below.

III. PREDICTIONS BASED ON LONG SHORT-TERM MEMORY (LSTM) NETWORKS

Although MLPs are frequently used in practice, their major limitation is that they are designed for point prediction as opposed to time-series prediction, which might require a context-aware method. Nevertheless, MLPs provide a solid baseline in machine-learning applications and thereby help verifying the need for a more sophisticated network architecture or lack thereof. An alternative to the MLP is the recurrent neural network (RNN), which is a type of neural network specifically suited to learn from sequential data such as time series. These networks have a much closer resemblance to biological neural networks, since they also learn the temporal dynamics of the input sequence. As opposed to MLPs, RNNs map a sequence of data points to another sequence using recurrent connections in time as outlined in Algorithm 2. A standard RNN is a neural network containing a single hidden layer with a feedback loop. The output of the hidden layer in the previous time instance is fed back into the hidden layer along with the current input. An RNN can be thought of as a very deep MLP with one layer for each instance in the input sequence where all layers share the parameters. This sequential feeding of the input data makes the temporal relationship apparent to the network. An RNN is parametrized by three weight matrices ($\mathbf{W}_{h\chi}$, $\mathbf{W}_{hh}$, and $\mathbf{W}_{\zeta h}$) and two biases ($\mathbf{b}_h$ and $\mathbf{b}_\zeta$). In Algorithm 2, g_h and g_ζ are the hidden and output activation functions, respectively. The loss function of an RNN for a single training example is the sum over p time steps of the squared error. On the other hand, the weights and biases are computed by back-propagation through time [27]. RNNs are particularly difficult to train, especially for sequences with long-range temporal dependencies. Since the back-propagation algorithm uses the chain rule to compute the weight corrections, small gradients in the later layers restrict the rate of learning of earlier layers. For long sequences, this gradient can gradually vanish, a fact that prevents the neural network from learning any further. This is called the problem of vanishing gradients. Long short-term memory (LSTM)

TABLE III. Summary of LSTM architectures and their performance using different numbers of training data sets.

Architecture	p	n_1	n_2	Number of data sets	$E_{\overline{u}}$ [%]	$E_{\overline{u^2}}$ [%]	Validation loss
LSTM1	10	90		100	2.36	14.73	2.0×10^{-8}
LSTM1	10	90		1000	0.83	3.44	8.5×10^{-9}
LSTM1	10	90		10 000	0.45	2.49	5.2×10^{-9}
LSTM2	10	90	90	100	1.94	6.82	2.4×10^{-8}
LSTM3	25	90		100	3.53	18.28	7.4×10^{-8}

networks developed by Hochreiter and Schmidhuber [28] use a gating mechanism to actively control the dynamics of the recurrent connections and thereby mitigate the vanishing gradient issue. Thanks to the same gating mechanism, LSTMs can also model longer temporal dependencies than standard RNNs. The LSTM will be the RNN employed in this study.

The impact of the considered LSTM architecture on the computed turbulence statistics is summarized in Table III, where the relative errors $E_{\overline{u}}$ and $E_{\overline{u^2}}$ are calculated as above based on ensembles of 500 time series spanning 4000 time units each. Note that the LSTM1 and LSTM3 architectures have a single layer, while in LSTM2 we consider a second one, and in all the cases we have 90 units per layer. In this first assessment, we only considered 100 data sets for training, and it can be observed how even using much less training data, the LSTM1 and LSTM2 networks outperform all the MLPs discussed above, and the LSTM3 architecture yields results similar to those of the best MLP, i.e., MLP4. Furthermore, the LSTM1 network leads to relative errors lower than MLP4 using $p = 10$, i.e., an input dimension 50 times lower than that in all the MLPs. The mean velocity profile $\overline{u}$, together with the streamwise velocity fluctuations $\overline{u^2}$ and the Reynolds shear stress $\overline{uv}$ (which is associated with turbulent transport), are shown in Fig. 3 for the three LSTM networks trained with 100 data sets. These are compared with the reference profiles obtained from the nine-equation model. This figure, together with the results in Table III, indicates that it would be possible to further improve the predictions from the LSTM1 network by considering a second layer as in the LSTM2 architecture. This significantly improves the prediction of the streamwise velocity fluctuation profile, reducing the relative error by about a factor of 2. Nevertheless, increasing the dimension of the input to $p = 25$ appears to have a detrimental effect on the results, in particular when predicting the fluctuations. We observed that the deviations in the $\overline{u^2}$ predictions are induced by a subset of the predicted time series, which leads to the different behaviors of the fluctuations
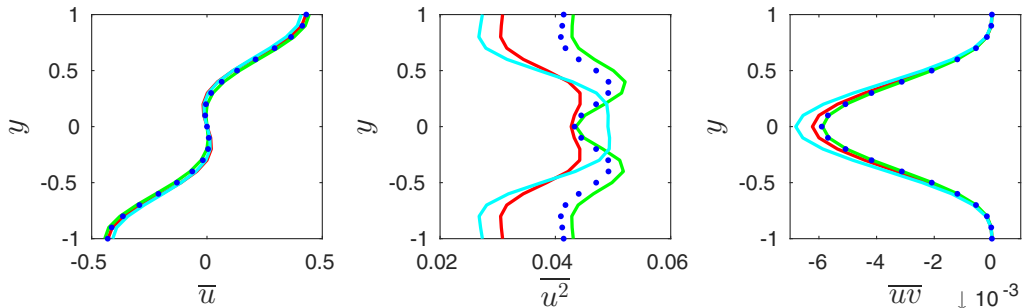

FIG. 3. Turbulence statistics corresponding to (left) streamwise mean profile, (middle) streamwise velocity fluctuations, and (right) Reynolds shear stress. Blue dots are used for the reference nine-equation model, and colored lines denote predictions using the following architectures: (red) LSTM1, (green) LSTM2, and (cyan) LSTM3. The networks were trained with 100 data sets.

observed in Fig. 3. Therefore, the use of long input sequences for predictions using the LSTM network may lead to higher errors in the predictions, especially in higher order moments (note that the prediction of the mean flow is in reasonably good agreement with the reference).

The mean velocity profile and the Reynolds shear stress are connected, as can be shown from a streamwise momentum balance. This was used by Vinuesa *et al.* [29] in turbulent channel flows (where such a balance reveals that the total shear stress is a linear function) to define a criterion for convergence of turbulence statistics. In the present case [14], the total shear stress $1/\text{Re}\, d\overline{u}/dy - \overline{uv}$ is given by a cosine of amplitude $2\sqrt{2}\pi/(4\text{Re})$. The total shear stress computed from the nine-equation model, averaged over 500 data sets, has a deviation of around 2% with respect to this value, a fact that supports the statistical convergence of our results. Interestingly, the predictions from the LSTM1 and LSTM2 networks (trained with 100 data sets) shown in Fig. 3 exhibit 4.7% and 2.8% deviation with respect to the reference amplitude, which suggests that these neural networks reproduced the link between the viscous and turbulent stresses. On the other hand, the LSTM3 network shows a significantly larger deviation of 43.5%, despite its reasonably good prediction of the mean velocity profile. We conjecture that it could be possible to further improve the predictions of neural networks by explicitly including the physical constraints present in the flow when designing the network.

Furthermore, we also assessed the effect of increasing the number of data sets used for training and calculated the deviations with respect to the reference turbulence statistics as summarized in Table III. Although in order to limit the computational cost of the training we considered the smaller LSTM1 architecture with 1000 and 10 000 training data sets, even lower error levels would be expected if the LSTM2 architecture was trained with larger amounts of data. The data in this table allow a direct comparison between the performance of the LSTM1 and MLP4 architectures, both trained with 1000 data sets: While the LSTM network leads to relative errors of 0.83% and 3.44% in the mean and the fluctuations, the MLP exhibits significantly higher errors of 3.21% and 18.61%, respectively. The LSTM1 results are in even better agreement with the reference statistics when 10 000 training data sets are employed for training, with relative errors in the mean velocity and the streamwise fluctuations of 0.45% and 2.49%, respectively. The excellent agreement between the turbulence statistics from the nine-equation model and the LSTM network can be observed in Fig. 4, where in addition to the mean streamwise velocity $\overline{u}$ and the streamwise velocity fluctuations $\overline{u^2}$, we show the Reynolds shear stress $\overline{uv}$, the wall-normal fluctuations $\overline{v^2}$, as well as the skewness $\overline{u^3}$ and the flatness $\overline{u^4}$. The agreement of all the statistics with the reference data is very good, and even higher order moments like the skewness and the flatness exhibit low relative errors, i.e., 1.01% and 2.57%, respectively. Also note that the amplitude of the cosine defining the total shear stress shows a deviation of 3.1% with respect to the value expected from the streamwise momentum balance, a fact that suggests that this network also reproduces the connection between the viscous and turbulent stresses. It is interesting to note that the present results were obtained with a loss function based on minimizing the error in the instantaneous values of the mode coefficients, but other studies such as the one by King *et al.* [30] propose employing turbulence-specific loss functions; these have the potential of further improving the flow predictions. In any case, the present results highlight the excellent predicting capabilities of the LSTM network, which could potentially be further improved by considering a larger network.

One of the most relevant features of the nine-equation model by Moehlis *et al.* [14] is the fact that it contains the features present in the near-wall cycle of wall-bounded turbulence [31], namely the streamwise vortices, the streaks, their instabilities, and the nonlinear interactions among the various structures. Instantaneous velocity fields obtained from the LSTM1 architecture are shown in Fig. 5, where it can be observed that the network exhibits the streamwise vortices convecting near-wall fluid toward the opposing wall, forming high- and low-speed streaks when the flow is lifted up and pulled down toward the wall, respectively. These streaks become unstable due to a sinusoidal modulation in the spanwise direction, which is visible on the vertical midplane. The close resemblance between the LSTM flow fields and the ones reported by Moehlis *et al.* [14] indicates
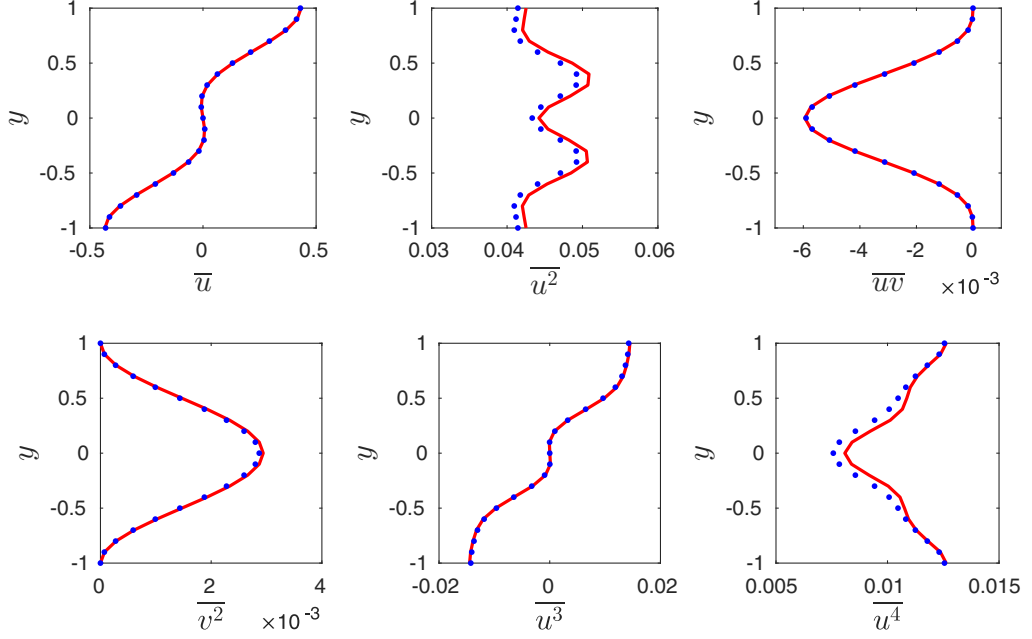
FIG. 4. Turbulence statistics corresponding to (top left) streamwise mean profile, (top middle) streamwise velocity fluctuations, and (top right) Reynolds shear stress; (bottom left) wall-normal velocity fluctuations, (bottom middle) skewness, and (bottom right) flatness. Blue dots are used for the reference nine-equation model and red lines for the predictions using the LSTM1 network trained with 10 000 data sets.

that the LSTM network reproduces the underlying physics in the nine-equation model, a fact that further justifies the excellent agreement of the turbulence statistics.

An additional assessment of the flow behavior can be obtained in terms of the fluctuating vorticity components. The three components of the instantaneous vorticity were computed by calculating the spatial derivatives of the Fourier modes $\mathbf{u}_j(\mathbf{x})$ and using the corresponding time coefficients $a_j(t)$. In Fig. 6, we show the root-mean-square (rms) vorticity fluctuations in the three spatial directions, obtained from the model and the LSTM1 network trained with 10 000 data sets. It is interesting to note that the nine-equation model exhibits a number of characteristic features from wall-bounded turbulence, such as the location of the relative maxima in the streamwise vorticity
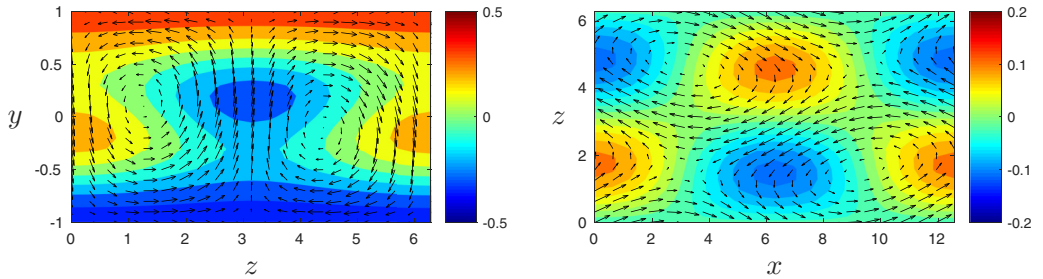

FIG. 5. Instantaneous velocity fields reconstructed from the mode amplitudes for the LSTM1 architecture. The colored contours represent the velocity component perpendicular to the visualized plane, and the vectors are the velocity components in the plane. The left panel is averaged in the streamwise direction, and the right panel shows the flow in the vertical midplane.
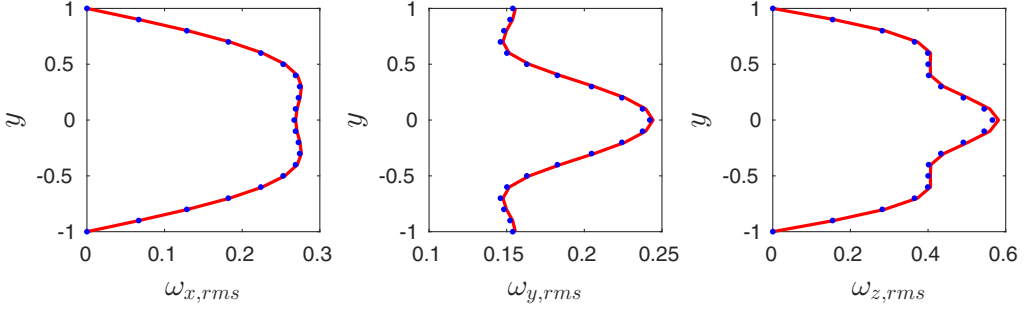
FIG. 6. (Left) Streamwise, (middle) wall-normal, and (right) spanwise root-mean-square vorticity fluctuations obtained from (blue dots) the reference nine-equation model and (red lines) the LSTM1 network. Note that the network was trained with 10 000 data sets.

fluctuation profile $\omega_{x,\text{rms}}$, approximately coinciding with the center of the near-wall streamwise vortices (see Fig. 5) [32], i.e., $y \simeq 0.25$ from the centerline. Regarding the LSTM predictions, they are in excellent agreement in the three cases, as in the other turbulence statistics reported in this study, with relative errors of 0.6%, 0.7%, and 1.3% for the streamwise, wall-normal, and spanwise fluctuating vorticities, respectively.

It is important to note that although the LSTM1 network yields excellent predictions of the turbulence statistics and the predicted fields exhibit the most relevant flow structures, it shows deviations in the instantaneous predictions of the mode amplitudes. This can be observed in Fig. 7, where the amplitudes of the nine coefficients are predicted for a single time series using the LSTM1 network (trained with 10 000 data sets) and are compared with the reference data from the nine-equation model. As shown in Table III, we consider a value $p = 10$ for this network, which implies that 10 points are used for the prediction. This is evident in the figure, where the 10 first points from each coefficient are identical in the LSTM1 and reference curves. Beyond the first 10 data points, small differences start to emerge in the predicted time series, leading to progressively larger errors and thus different temporal evolutions in all the coefficients. Note, however, that despite the instantaneous differences in the values of the coefficients, the qualitative behavior of the reference and predicted curves is similar. In fact, certain features appear to be reproduced by the LSTM1 network, such as, for instance, the large peaks in a_1 observed at $t \simeq 1000$ and 1600, which are observed with small shifts in time and different amplitudes in the predicted series. Similar observations can be made for the rest of coefficients regarding the most prominent features of the various time series.

In order to quantify the deviation between the reference and predicted series discussed above, we define the relative error in the prediction of the amplitude $a_1(t)$ in a single time series consisting of N_s samples as

$$e_1 = \frac{1}{(N_s - p)a_{1,\text{lam}}} \sum_{j=p+1}^{N_s} \left| a_{1,\text{mod}}^j - a_{1,\text{pred}}^j \right|, \tag{5}$$

where also here "mod" and "pred" denote data from the nine-equation model and predictions from the NN, respectively, and $a_{1,\text{lam}} = 1$ is the value for laminar flow. Note that the difference between Eqs. (3) and (5) lies in the data used as a reference: Whereas in the former we use the training data (i.e., data seen by the network during training), in the latter we use new data which has not been employed for training the network. Despite the very low validation loss obtained when considering the LSTM1 network trained with 10 000 data sets (5.2×10^{-9}), the relative error in the instantaneous prediction of a_1 was relatively high ($e_1 = 13.08\%$) and those of the other modes were of comparable magnitude. The reason for this discrepancy lies in the nature of the nine-equation
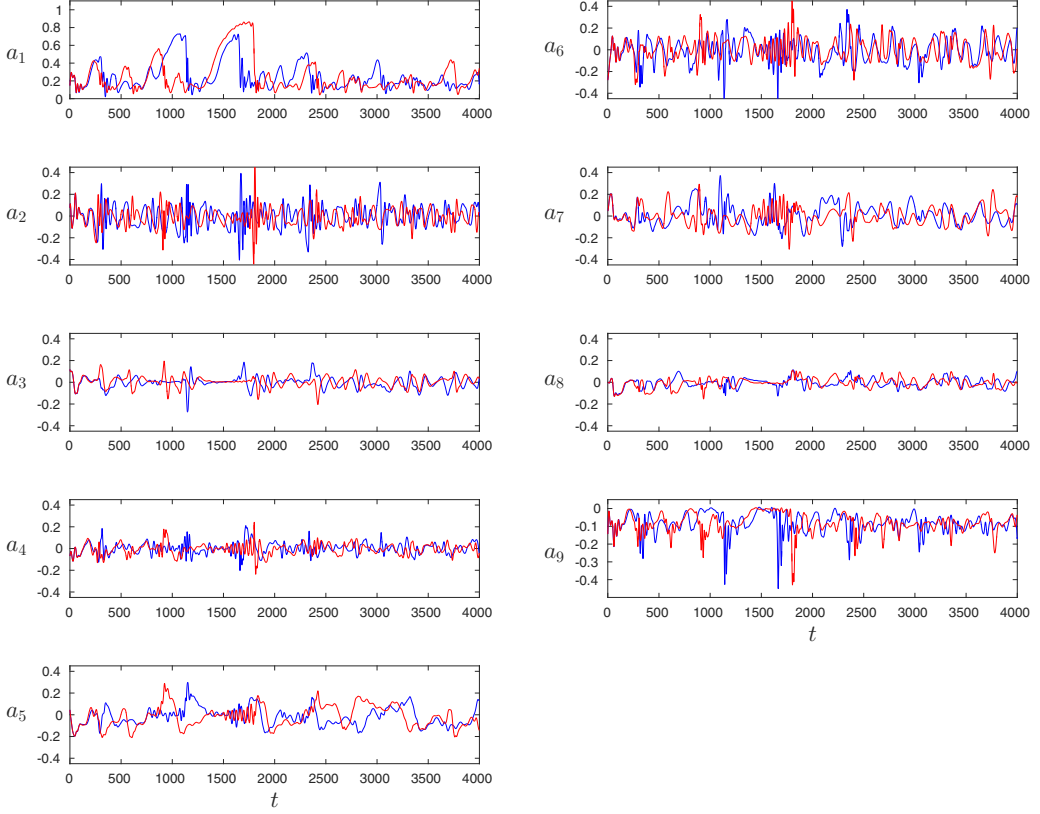
FIG. 7. Temporal evolution of the nine coefficients for a particular time series, from (blue) the nine-equation model and (red) the LSTM1 network trained with 10 000 data sets.

model, which is a dynamical system in which very small perturbations lead to completely different temporal evolutions of the modes. Thus, even if the network is able to reproduce the behavior of the flow very accurately, the instantaneous values differ due to the chaotic nature of the system. In order to further analyze the quality of the predictions, we assessed whether the dynamic behavior of the predicted flow was consistent with the reference model. To this end, we first consider the Poincaré map defined as the intersection of the flow state with the hyperplane $a_2 = 0$ on the $a_1 - a_3$ space (with the additional condition $da_2/dt < 0$). This Poincaré map is a lower-dimensional representation of the system which allows us to assess whether the correlation between amplitudes a_1 and a_3 (which are the amplitudes of the modes containing the base laminar profile and the streamwise vortices [14]) is adequately reproduced by the LSTM network. In Fig. 8 (left), we show the probability density function (pdf) of the Poincaré maps constructed from the 500 time series obtained from the LSTM1 prediction and the reference nine-equation model. The similarity between both maps shows that the LSTM network effectively captures the correlation between both amplitudes, suggesting that the interaction between modes $\mathbf{u}_1$ and $\mathbf{u}_3$ is adequately represented by the network. Additionally, we analyzed the rate of separation among trajectories in the nine-equation model and the LSTM predictions through their Lyapunov exponents. Given two time series 1 and 2, we define the separation of these trajectories as the Euclidean norm in nine-dimensional space,

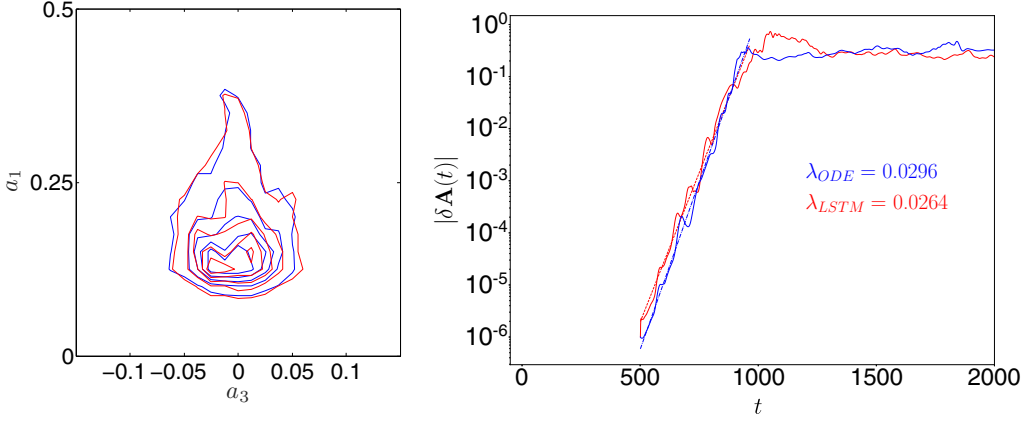$$|\delta\mathbf{A}(t)| = \left\{ \sum_{i=1}^{9} [a_{i,1}(t) - a_{i,2}(t)]^2 \right\}^{1/2}, \tag{6}$$

FIG. 8. (Left) Probability density function of the Poincaré maps, where the intersection with the $a_2 = 0$ plane (with $da_2/dt < 0$) is shown. (Right) Ensemble-averaged divergence of instantaneous time series after a perturbation with $|\delta \mathbf{A}_0| = 10^{-6}$ is introduced at $t_0 = 500$, showing initial exponential growth and the value of the Lyapunov exponent (dashed lines added to illustrate the obtained slope). In both panels, blue and red denote reference model and LSTM prediction, respectively.

and denote the separation at $t = t_0$ as $|\delta \mathbf{A}_0|$. The initial divergence of both trajectories can be assumed as

$$|\delta \mathbf{A}(t')| = \exp(\lambda t')|\delta \mathbf{A}_0|, \tag{7}$$

where λ is the so-called Lyapunov exponent and $t' = t - t_0$. Here, we introduce a perturbation with norm $|\delta \mathbf{A}_0| = 10^{-6}$ (which corresponds to the order of magnitude of the accuracy of the LSTM1 architecture) at $t_0 = 500$, where all the coefficients are perturbed, and analyze its divergence with respect to the unperturbed trajectory. In Fig. 8 (right), we show the evolution of $|\delta \mathbf{A}(t)|$ with time for both the nine-equation model and the LSTM prediction, ensemble averaged over 10 initial conditions. The rates of divergence from both reference and prediction are very similar, and the estimated values of λ are almost identical: 0.0296 for the reference and 0.0264 for the LSTM. After the initial period of divergence, both the nine-equation model and the network saturate at $t \simeq 1000$. This again supports the claim that the LSTM prediction reproduces the dynamical behavior of the original nine-equation model, and the discrepancy in instantaneous predictions is due to the chaotic nature of the system.

IV. CONCLUDING REMARKS

In this study, we have illustrated the potential of neural networks to predict the temporal dynamics of a low-order model of a turbulent flow. We used the nine-equation model by Moehlis *et al.* [14] to train a number of MLP and LSTM networks, using several architectures, and obtained flow predictions from all of them. While both types of neural network are able to capture the most relevant flow structures present in the flow, the LSTM yields excellent agreement with the reference data, both in terms of turbulence statistics and dynamical behavior of the system. Note that we employed the Moehlis *et al.* [14] model of the near-wall cycle due to its simplicity and flexibility to produce large amounts of training data, thus facilitating the training and testing of the various network architectures. After this initial assessment of the suitability of using neural networks to predict simplified turbulent flows, the next step will be to extend the present machine-learning framework to cases where the governing equations are *not* known, e.g., the LES equations arising due to nontrivial (spatial and temporal) filtering or the generation of (instantaneous) inflow and boundary conditions.

The present study is aimed at assessing the feasibility of neural network predictions on a low-order representation of near-wall turbulence. Therefore, its scope is limited to the model by Moehlis *et al.* [14], where the basis functions were fixed *a priori* and the network prediction pertained only to the time coefficients. Despite this limitation, the considered system can be viewed as a representative model for the near-wall cycle of wall-bounded turbulence assuming that the flow is properly scaled in inner units for which the near-wall cycle (disregarding outer-layer influences) is universal. On the other hand, the present methods could be extended to more challenging flows such as channels and boundary layers, including advancing from a low-order model to a higher order representation of the flow dynamics. This step will necessitate the definition of proper basis functions in addition to the temporal dynamics, which could be achieved based on, e.g., Fourier or POD (proper orthogonal decomposition) methods.

Using a standard workstation [Intel(R) Core(TM) i7-4930K CPU at 3.4 GHz], the LSTM1 training with 10 000 time series required around 70 h. After training the network, it takes around 12 min to produce 500 time series, which is the amount of data required to obtain converged statistics. On the other hand, on the same workstation it takes around 6 min to produce the same amount of data by integrating the nine-equation model by Moehlis *et al.* [14]. Thus, after the initial investment required to train the network, the computer time necessary to predict the flow is only around twice as large as that of resolving a nine-equation model. Furthermore, neural networks, once trained, can be accurately summarized into smaller and more computationally efficient networks which will reduce the time complexity of prediction significantly [33]. Consequently, we consider that the computational cost of evaluating the neural network is sufficiently low to constitute an efficient alternative for predicting instantaneous variables as, e.g., in SGS models. Since the network designs discussed here can also be employed for the prediction of other types of flows, this study may serve as a guideline for the most suitable strategies to perform such predictions. Other natural extensions of the present work include the use of deep neural networks to generate inflow conditions for spatially developing flows (see the study by Jarrin *et al.* [34]) and also to set off-wall boundary conditions in high-Reynolds-number wall-bounded turbulence simulations (as discussed in the work by Mizuno and Jiménez [35]). Because a significant fraction of the computational cost is employed to resolve the near-wall region, the use of adequate off-wall boundary conditions may allow us to achieve very high Reynolds numbers in turbulence simulations. These two applications highlight the potential relevance of the methods described in this study.

[1] D. H. Hubel and T. N. Wiesel, Receptive fields, binocular interaction, and functional architecture in the cat's visual cortex, J. Physiol. **160**, 106 (1962).

[2] K. Duraisamy, G. Iaccarino, and H. Xiao, Turbulence modeling in the age of data, Annu. Rev. Fluid Mech. **51**, 357 (2019).

[3] J. Ling, A. Kurzawski, and J. Templeton, Reynolds averaged turbulence modelling using deep neural networks with embedded invariance, J. Fluid Mech. **807**, 155 (2016).

[4] J.-L. Wu, H. Xiao, and E. Paterson, Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework, Phys. Rev. Fluids **3**, 074602 (2018).

[5] M. Milano and P. Koumoutsakos, Neural network modeling for near wall turbulent flow, J. Comput. Phys. **182**, 1 (2002).

[6] C. J. Lapeyre, A. Misdariis, N. Cazard, D. Veynante, and T. Poinsot, Training convolutional neural networks to estimate turbulent sub-grid scale reaction rates, Combust. Flame **203**, 255 (2019).

[7] A. D. Beck, D. G. Flad, and C.-D. Munz, Deep neural networks for data-driven turbulence models, arXiv:1806.04482.

[8] C. Lee, J. Kim, D. Babcock, and R. Goodman, Application of neural networks to turbulence control for drag reduction, Phys. Fluids **9**, 1740 (1997).

[9] N. Gautier, J.-L. Aider, T. Duriez, B. R. Noack, M. Segond, and M. Abel, Closed-loop separation control using machine learning, J. Fluid Mech. **770**, 442 (2015).

[10] M. Shimizu and G. Kawahara, Construction of low-dimensional system reproducing low-Reynolds-number turbulence by machine learning, arXiv:1803.08206.

[11] K. Fukami, Y. Nabae, K. Kawai, and K. Fukagata, A synthetic turbulent inflow generator using machine learning, arXiv:1806.08903.

[12] J. Jiménez, Machine-aided turbulence theory, J. Fluid Mech. **854**, R1 (2018).

[13] J. N. Kutz, Deep learning in fluid dynamics, J. Fluid Mech. **814**, 1 (2017).

[14] J. Moehlis, H. Faisst, and B. Eckhardt, A low-dimensional model for turbulent shear flows, New J. Phys. **6**, 56 (2004).

[15] L. Kim, Transient growth for a sinusoidal shear flow model, Master's thesis, University of California, Santa Barbara, CA, 2005.

[16] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, Tensorflow: A system for large-scale machine learning, in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI '16)* (USENIX Association, 2016), Vol. 16, pp. 265–283.

[17] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning internal representations by error propagation, Tech. Rep. No. ICS-8506, California University at San Diego La Jolla Institute for Cognitive Science, San Diego, CA, 1985 (unpublished).

[18] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, Learning representations by back-propagating errors, Nature (London) **323**, 533 (1986).

[19] D. P. Kingma and L. J. Ba, Adam: A method for stochastic optimization, in *Proceedings of the International Conference for Learning Representations* (Informatics Institute, Ithaca, NY, 2015).

[20] X. Glorot and Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS) 2010, Chia Laguna Resort, Sardinia, Italy* (JMLR W&CP, 2010), Vol. 9, pp. 249–256.

[21] K. He, X. Zhang, S. Ren, and J. Sun, Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, in *Proceedings of the IEEE International Conference on Computer Vision, 11-18 December 2015, Araucano Park, Las Condes, Chile* (IEEE, Piscataway, NJ, 2015), pp. 1026–1034.

[22] Y. LeCun, Y. Bengio, and G. Hinton, Deep learning, Nature (London) **521**, 436 (2015).

[23] V. Nair and G. E. Hinton, Rectified linear units improve restricted Boltzmann machines, in *Proceedings of the 27th International Conference on Machine Learning (ICML-10), Haifa, Israel* (Omni Press, 2010), pp. 807–814.

[24] A. L. Maas, A. Y. Hannun, and A. Y. Ng, Rectifier nonlinearities improve neural network acoustic models, in *Proceedings of the 30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013* (JMLR W&CP, 2013), Vol. 28, p. 3.

[25] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, Fast and accurate deep network learning by exponential linear units (ELUs), arXiv:1511.07289v5.

[26] P. A. A. Srinivasan, Deep learning models for turbulent shear flow, Master's thesis, KTH Royal Institute of Technology, Stockholm, Sweden, 2018.

[27] P. J. Werbos, Backpropagation through time: What it does and how to do it, Proc. IEEE **78**, 1550 (1990).

[28] S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Comput. **9**, 1735 (1997).

[29] R. Vinuesa, C. Prus, P. Schlatter, and H. M. Nagib, Convergence of numerical simulations of turbulent wall-bounded flows and mean cross-flow structure of rectangular ducts, Meccanica **51**, 3025 (2016).

[30] R. King, O. Hennigh, A. Mohan, and M. Chertkov, From deep to physics-informed learning of turbulence: Diagnostics, arXiv:1810.07785.

[31] J. M. Hamilton, J. Kim, and F. Waleffe, Regeneration mechanisms of near-wall turbulence structures, J. Fluid Mech. **287**, 317 (1995).

[32] J. Kim, P. Moin, and R. Moser, Turbulence statistics in fully developed channel flow at low Reynolds number, J. Fluid Mech. **177**, 133 (1987).

[33] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, Quantized neural networks: Training neural networks with low precision weights and activations, J. Mach. Learn. Res. **18**, 1 (2018).

[34] N. Jarrin, S. Benhamadouche, D. Laurence, and R. Prosser, A synthetic-eddy-method for generating inflow conditions for large-eddy simulations, Int. J. Heat Fluid Flow **27**, 585 (2006).

[35] Y. Mizuno and J. Jiménez, Wall turbulence without walls, J. Fluid Mech. **723**, 429 (2013).

[36] P. A. Srinivasan, H. Azizpour, P. Schlatter, and R. Vinuesa, DeepTurbulence, https://github.com/KTH-Nek5000/DeepTurbulence.