# Data-Driven Reduced-Complexity Modeling of Fluid Flows: A Community Challenge

Oliver T. Schmidt*
*University of California San Diego, La Jolla 92093, California*
Aaron Towne†
*University of Michigan, Ann Arbor 48109, Michigan*
Adrian Lozano-Duran‡
*California Institute of Technology, Pasadena 91125, California*
Scott T. M. Dawson§
*Illinois Institute of Technology, Chicago 60616, Illinois*
and
Ricardo Vinuesa¶
*University of Michigan, Ann Arbor 48109, Michigan*

https://doi.org/10.2514/1.J066928

**We introduce a community challenge designed to facilitate direct comparisons between data-driven methods for compression, forecasting, and sensing of complex aerospace flows. The challenge is organized into three tracks that target these complementary capabilities: compression (compact representations for large datasets), forecasting (predicting future flow states from a finite history), and sensing (inferring unmeasured flow states from limited measurements). Across these tracks, multiple challenges span diverse flow datasets and use cases, each emphasizing different model requirements. The challenge is open to anyone, and we invite broad participation to build a comprehensive and balanced picture of what works and where current methods fall short. To support fair comparisons, we provide standardized success metrics, evaluation tools, and baseline implementations, with one classical and one machine-learning baseline per challenge. Final assessments use blind tests on withheld data. We explicitly encourage negative results and careful analyses of limitations. Outcomes will be disseminated through an *AIAA Journal Virtual Collection* and invited presentations at AIAA conferences.**

Downloaded by University of Michigan on October 14, 2026 | http://arc.aiaa.org | DOI: 10.2514/1.J066928

## I. Introduction

DRIVING community progress through well-designed challenges is an established practice in machine learning and has led to major breakthroughs, for example, in image recognition through the ImageNet challenge [1], in natural language understanding through the GLUE benchmark and analysis platform [2], and in protein folding through the CASP series of challenges [3]. In aerospace science and fluid dynamics, such efforts are comparatively scarce, but notable examples include the ML4CFD competition on machine learning for airfoil design [4], the Collaborative Testing Challenge on data-driven turbulence modeling [5], "The Well" challenge, which assembles a broad collection of physics simulations, many of them fluid-dynamics problems, for machine learning [6], and, very recently, a common task framework for machine learning of dynamical systems [7].

This community challenge targets complex aerospace flows across multiple application-driven cases and organizes them into three complementary capabilities: *compression*, *forecasting*, and *sensing*. It pairs standardized evaluation with baseline methods drawn from established statistical tools, dynamical systems, and machine learning. *Compression* provides compact representations that are essential not only for storage but, crucially, also to ensure that other methods remain tractable for large datasets and deployable in near real time. *Forecasting* targets prediction in time, i.e., inferring future flow states from a finite history. *Sensing targets* prediction in space, i.e., inferring unmeasured states or derived quantities from limited measurements. Together, these tasks cover common operational needs in analysis, design, and control. Each challenge type is represented schematically in Fig. 1.

The challenge emphasizes purely data-driven methods. Our goal is to enable fair comparisons using standardized success metrics and common baselines on relevant example data, spanning classical statistical or dynamic approaches and modern machine-learning methods, in a fully open and transparent setting. While each challenge specifies datasets and metrics, participants are completely free in their choice of method so long as it is purely data-driven.

To anchor comparisons, we provide established baseline methods, one classical and one ML method per challenge, paired with diverse flow datasets, largely drawn from a public database specifically constructed for evaluating reduced-complexity models for fluid flows [8]. Each dataset is paired with specific challenges that reflect representative aerospace use cases. An overview of the challenges, datasets, and baseline methods is given in Table 1.

Participants are expected to release their code and undergo blinded evaluation on held-out data by submitting results to the designated point of contact for each challenge. Dissemination will include a Virtual Issue of the *AIAA Journal* and invited talks in special sessions at AIAA conferences. Participation guidelines are detailed in Sec. V, and additional information and periodic updates are available on the challenge webpage: fluids-challenge.engin.umich.edu.

Our objective is to identify competitive methods, distill best practices, and advance community understanding. Novel architectures are welcome, as are careful adaptations or lightweight customizations of established methods. Importantly, submissions that illuminate limitations or failure modes are also valuable contributions to the community.

Received 9 February 2026; accepted for publication 30 March 2026; published online 14 October 2026. Copyright © 2026 by Oliver T. Schmidt, Aaron Towne, Adrian Lozano-Duran, Scott T. M. Dawson, and Ricardo Vinuesa. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. All requests for copying and permission to reprint should be submitted to CCC at www.copyright.com; employ the eISSN 1533-385X to initiate your request. See also AIAA Rights and Permissions https://aiaa.org/publications/publish-with-aiaa/rights-and-permissions/.

*Associate Professor, Department of Mechanical and Aerospace Engineering. Senior Member AIAA (Co-First Authors).
†Associate Professor, Department of Mechanical Engineering. Senior Member AIAA (Co-First Authors).
‡Associate Professor, Graduate Aerospace Laboratories. Senior Member AIAA.
§Associate Professor, Department of Mechanical, Materials, and Aerospace Engineering. Associate Fellow AIAA.
¶Associate Professor, Department of Aerospace Engineering.

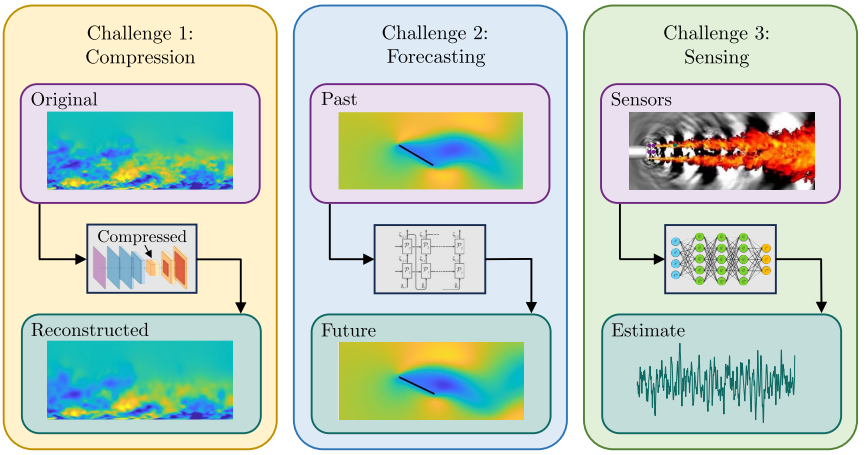



**Fig. 1 Overview of the challenges, which span the core capabilities of compression, forecasting, and sensing. Each challenge type conceptually includes input data (plum boxes) and target output data (teal boxes) and seeks methods to perform the desired task (gray boxes).**

**Table 1 Summary of the three challenges, their subchallenges, datasets, and baseline methods**

| Challenge | Subchallenges | Data | Baseline methods |
|---|---|---|---|
| 1. Compression | 1.1 Compression of 2D turbulent boundary-layer velocity fields | DNS of high-$Re$ turbulent boundary layer | Proper orthogonal decomposition (POD), Convolutional autoencoders (CNN-AE) |
| | 1.2 Compression of 3D turbulent boundary-layer velocity fields | DNS of high-Re turbulent boundary-layer | |
| 2. Forecasting | 2.1 Forecasting of a turbulent cavity flow | Time-resolved PIV of a cavity | Dynamic mode decomposition (DMD), Long short-term memory network (LSTM) |
| | 2.2 Forecasting of pitching airfoils with parametric variations | DNS of pitching airfoil across pitching angles and frequencies | |
| 3. Sensing | 3.1 Sensing of a turbulent jet using sparse measurements | LES of axisymmetric jet | Wiener filter (WF) & Multilayer perceptron (MLP), Linear stochastic estimation (LSE) & convolutional neural network (CNN) |
| | 3.2 Sensing of a turbulent boundary layer from wall stress | DNS of low-$Re$ turbulent boundary layer | |

## II. Challenge 1: Compression

### A. Introduction

Compressing turbulent flow data poses a fundamental challenge due to the chaotic nature of turbulence, its fractal structure, and the wide range of spatial and temporal scales involved. These characteristics make it difficult to reduce the dimensionality of the data without sacrificing critical physical information. Effective compression must therefore strike a balance between a compact representation and the ability to accurately reconstruct key flow features. In practical scenarios, such as real-time monitoring or on-the-fly flow compression, the wall time and computational cost associated with compression and reconstruction are equally important. These considerations motivate the need for novel modeling strategies that optimize not only the compression ratio and reconstruction fidelity but also the efficiency of the process in terms of cost and time.

We pose two representative challenges within the context of a turbulent boundary layer:

i) The first focuses on compressing and reconstructing all three components of the velocity fluctuations on two-dimensional planes extracted from the flow.

ii) The second extends this task to the three-dimensional reconstruction of streamwise velocity fluctuations.

These scenarios reflect different levels of complexity and spatial information. To evaluate performance, we consider a combination of metrics that capture tradeoffs between reconstruction accuracy, measured via normalized mean square error (NMSE), the achieved compression ratio, and the computational cost of both compression and reconstruction.

### B. Baseline Methods

Two baseline methods are considered to illustrate the compression challenge: space-only POD with $N$ modes and a convolutional autoencoder (CNN-AE) with a latent space of dimension $N$.

#### 1. *Proper Orthogonal Decomposition*

Proper orthogonal decomposition (POD) constitutes the optimal linear method for data compression [9]. Here, we use the space-only variant of POD [10,11], in which the spatially discretized flow state $\boldsymbol{q}(t)$ is approximated by $\tilde{\boldsymbol{q}}(t)$ as a linear combination of $N$ orthogonal modes $\boldsymbol{\phi}_i$ and their corresponding time-dependent coefficients $a_i(t)$, i.e.,

$$\tilde{\boldsymbol{q}}(t) = \sum_{i=1}^{N} a_i(t)\boldsymbol{\phi}_i \tag{1}$$

The optimal spatial modes are obtained from the eigendecomposition of the spatial correlation matrix $\boldsymbol{C}_{qq} = E\{\boldsymbol{q}\boldsymbol{q}^*\}$. The temporal coefficients are recovered by projecting the state $\boldsymbol{q}$ onto each mode and are used to reconstruct the state according to Eq. (1). POD has been extensively employed to compress fluid flows, often as a starting point for additional modeling efforts [12–14].

#### 2. *Convolutional Autoencoder (CNN-AE)*

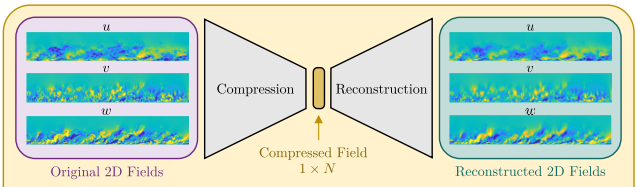
Autoencoders offer a natural nonlinear extension of POD [15,16] and consist of an encoder and decoder. The encoder is a nonlinear function $E_\theta$ that maps the flow state $\boldsymbol{q}$ to a latent representation $\boldsymbol{a} \in \mathbb{R}^N$ that captures the essential features of the input in a reduced-dimensional space, i.e., $\boldsymbol{a} = E_\theta(\boldsymbol{q})$. The decoder reconstructs the flowfield from the latent space using another nonlinear function $D_\theta$, i.e., $\tilde{\boldsymbol{q}} = D_\theta(\boldsymbol{a})$, where $\tilde{\boldsymbol{q}}$ is the reconstructed approximation of the original flow state $\boldsymbol{q}$. During training, the network parameters $\theta$ are optimized to minimize a desired loss function.

Here, we use a convolutional autoencoder (CNN-AE) [17] with a latent space of dimension $N$ to compress and reconstruct the flowfield data. Both the encoder and decoder are built using convolutional neural networks, allowing the model to leverage the

Downloaded by University of Michigan on October 14, 2026 | http://arc.aiaa.org | DOI: 10.2514/1.J066928

spatial correlations inherent in the flowfields for more accurate and efficient representation and reconstruction. The network is trained by minimizing the mean-squared reconstruction error. Autoencoders have become popular in recent years for identifying features and low-dimensional subspaces in fluid flows [18,19].

### C. Challenge 1.1: Compression of 2D Turbulent Boundary-Layer Velocity Fields




**Challenge 1.1:** Given 1000 planar snapshots of the velocity field from a turbulent boundary layer, compress the data to maximize the compression ratio and minimize the total compression-reconstruction time for a target error level.

#### 1. *Data*

This challenge uses data from a direct numerical simulation (DNS) of a zero-pressure-gradient incompressible turbulent boundary layer. In particular, we use data from the BL2 case described in Ref. [8]. The full database contains 7500 three-dimensional flowfields sampled every $\Delta t^+ \approx 4$ and spanning a physical time of 7.4 eddy turnovers. The training dataset for this challenge includes 5000 planar snapshots in the streamwise-wall-normal ($x$–$y$) plane, covering friction Reynolds numbers in the range $Re_\tau \approx 480$–$1020$. Each snapshot contains the streamwise ($u$), wall-normal ($v$), and spanwise ($w$) velocity fluctuations on a $4000 \times 148$ grid. The test set comprises an additional 1000 snapshots drawn from the same simulation.

#### 2. *Baseline Results*

The performance of the methods in the compression challenge is assessed by examining the compression ratio and compression/reconstruction times as functions of the root-normalized-mean-squared error (root-NMSE) obtained using compressor–reconstructor pairs with varying numbers of retained modes/latent-space dimensions ($N$). The normalized mean-squared error (NMSE) is defined as

$$e = \frac{E\{\|\tilde{\boldsymbol{q}}(t) - \boldsymbol{q}(t)\|^2\}}{E\{\|\boldsymbol{q}(t)\|^2\}} \tag{2}$$

where the expectation is a time average over snapshots and the norm constitutes spatial integration over the field.

POD is conducted on the three velocity components for 11 cases, using retained mode counts ranging from $N = 4$ up to 4096, increasing in successive powers of two. The 2D-CNN-AE comprises three channels (each with one velocity component) that are stacked into a 4D array. The encoder consists of two convolutional blocks, each with a 3×3 convolution (16 and 32 filters, respectively), *Re*LU activation, and 2×2 max-pooling, reducing spatial dimensions by 4. The bottleneck uses a fully connected layer to produce a latent vector of size $N$. The decoder uses a fully connected layer to reshape the latent representation, followed by two transposed convolutional layers (4×4, stride 2) and a final 3×3 convolution to reconstruct the 3-channel output, trained with Adam optimization for 100 epochs using a mean-squared-error loss function.

The example results for Challenge 1.1 are shown in Fig. 2, which presents a comparative analysis of compressing and reconstructing the three components of the turbulent velocity fluctuations using POD and CNN-AE. An example of the reconstructed $u$ is shown in Fig. 3.

Figure 2a shows the relationship between the root-NMSE and the dimension of the latent space (or the number of POD modes). This plot illustrates the fundamental tradeoff between dimensionality reduction and reconstruction fidelity. Lower root-NMSE values at smaller latent dimensions reflect a more effective compression method, capable of capturing essential flow features in a compact representation. While latent-space dimensionality is not used as a standalone performance metric (due to method-dependent definitions), we include root-NMSE versus latent-dimension plots for both POD (interpreted as the number of modes) and 2D-CNN-AE, as they offer valuable insights. For both POD and 2D-CNN-AE, the root-NMSE consistently decreases with increasing latent dimension, indicating improved reconstruction accuracy at higher representation capacity. However, to reach a comparable root-NMSE, POD requires significantly larger latent spaces than 2D-CNN-AE. This highlights

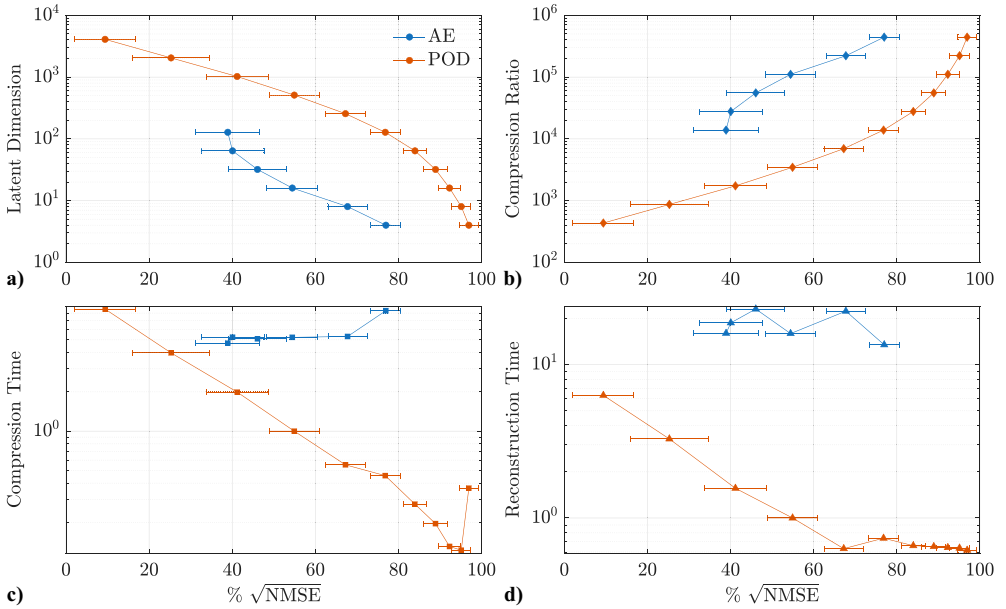



**Fig. 2 Challenge 1.1: Application of compression metrics of success in 2D planes of the three velocity components in a turbulent boundary layer using 2D-CNN-AE and POD. The bars represent the standard deviation of the error. The compression and reconstruction times are normalized using the POD method with 512 modes.**

Downloaded by University of Michigan on October 14, 2026 | http://arc.aiaa.org | DOI: 10.2514/1.J066928

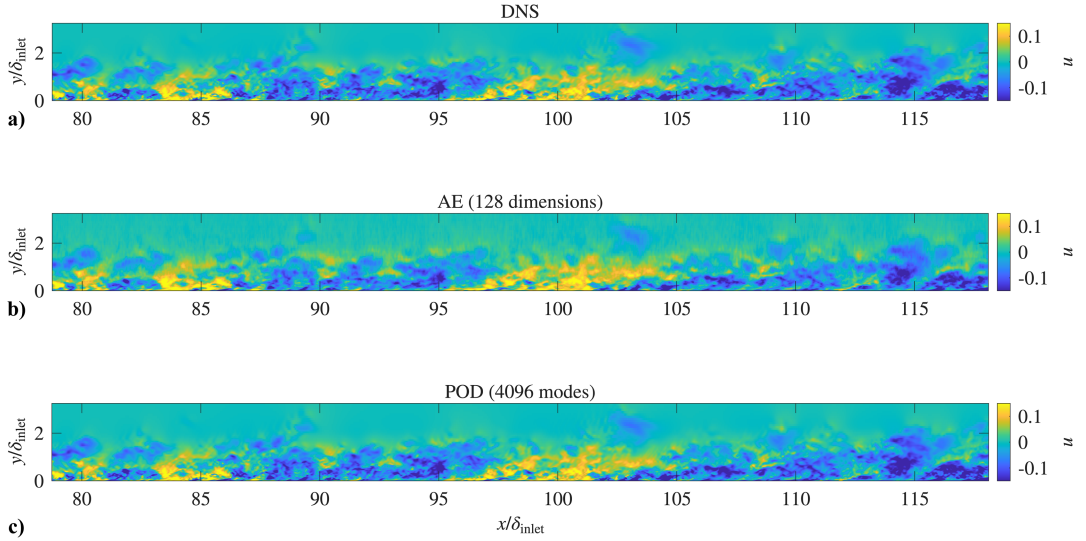



**Fig. 3 Challenge 1.1: Application of compression to 2D fields. Example snapshot of the streamwise velocity field (a), its reconstruction using a 2D-CNN-AE with a latent dimension of 128 (b), and reconstruction using 4096 POD modes (c). The streamwise ($x$) and wall-normal ($y$) coordinates are nondimensionalized by the boundary-layer thickness at the inlet of the simulation ($\delta_{\text{inlet}}$).**

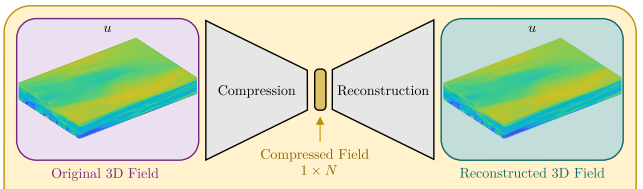
2D-CNN-AE's superior ability to represent the complex, nonlinear structures in turbulent flows, in contrast to the linear basis functions used in POD.

Figure 2b shows the root-NMSE as a function of the compression ratio. In this analysis, we consider only the size of the compressed (latent) representation, excluding the overhead associated with the compression and reconstruction machinery, such as neural network parameters or modal basis functions. As a result, the reported compression ratio reflects the asymptotic limit of *infinite data*, where the size of the compression algorithm itself becomes negligible compared to the size of the data. Both POD and 2D-CNN-AE exhibit a gradual increase in root-NMSE with increasing compression, reflecting the expected tradeoff between data reduction and reconstruction accuracy. For a given root-NMSE, 2D-CNN-AE achieves significantly higher compression ratios than POD, highlighting its efficiency in representing turbulent flows. This result suggests that 2D-CNN-AE can reduce data volume more aggressively without compromising reconstruction fidelity—an important advantage for applications that demand high compression.

Figures 2c and 2d examine root-NMSE as a function of compression and reconstruction time, respectively. To reduce variability introduced by hardware differences, both compression and reconstruction times are reported relative to a baseline method executed on the same hardware used for each submission. In this case, both compression and reconstruction times are normalized by the corresponding values obtained using the POD method with 512 modes. For POD, both times increase as the root-NMSE decreases, reflecting the additional computational effort required to achieve higher accuracy due to the larger number of retained modes. In contrast, the compression and reconstruction times for 2D-CNN-AE are generally higher than those for POD over the range of root-NMSE values considered and remain relatively constant, regardless of the desired accuracy.

In summary, the 2D-CNN-AE outperforms POD in terms of compression ratio, achieving higher reconstruction accuracy for smaller compressed data sizes. However, the 2D-CNN-AE shows signs of saturation as the latent-space dimension increases and never yields errors below 40%. In contrast, POD achieves errors as low as 10%. POD also offers faster compression and reconstruction, outperforming the 2D-CNN-AE in computational efficiency over the root-NMSE range considered. Notably, the compression-reconstruction time for 2D-CNN-AE remains relatively constant as the target root-NMSE decreases, indicating that 2D-CNN-AE may surpass POD in efficiency as well for lower root-NMSE values.

### D. Challenge 1.2: Compression of 3D Turbulent Boundary-Layer Velocity Fields




**Challenge 1.2:** Given 50 volumetric snapshots of the streamwise velocity from a turbulent boundary layer, compress the data to maximize the compression ratio and minimize the total compression-reconstruction time for a target error level.

#### 1. *Data*

This challenge uses data from the same DNS of a zero-pressure-gradient incompressible turbulent boundary layer [8]. The training set consists of 500 volumetric (three-dimensional) snapshots of streamwise velocity fluctuations, extracted from a subdomain of the boundary layer with $Re_\tau \approx 920$–$945$. Each volumetric field is defined on a $256 \times 64 \times 256$ grid, corresponding to the streamwise, wall-normal, and spanwise directions, respectively. The test set includes 50 additional snapshots of the volumetric field.

#### 2. *Baseline Results*

POD is performed on the full 3D streamwise velocity field for four cases, with the number of retained POD modes set to $N = 50$, 100, 200, and 400. The 3D-CNN-AE consists of an encoder with five blocks, each containing a 3D convolution (3×3×3 filters, increasing from 16 to 256 channels), batch normalization, *Re*LU activation, and 2×2×2 average pooling, reducing spatial dimensions to [8, 2, 8]. The bottleneck layer applies a 3×3×3 convolution to produce a latent representation with $N_c$ channels. The decoder mirrors the encoder with five blocks of transposed convolutions (4×4×4, stride 2) to upsample, followed by convolution, batch normalization, and *Re*LU, ending in a final 3×3×3 convolution to output a single-channel reconstructed volume. The 3D-CNN-AE is trained with Adam optimization for 650 epochs using mean-squared-error loss.

The results for Challenge 1.2 are shown in Fig. 4, which presents a comparative analysis of compressing and reconstructing the 3D field for the streamwise velocity fluctuations using POD and 3D-CNN-AE. An example of the 3D reconstructed $u$ is shown in Fig. 5. The

Downloaded by University of Michigan on October 14, 2026 | http://arc.aiaa.org | DOI: 10.2514/1.J066928

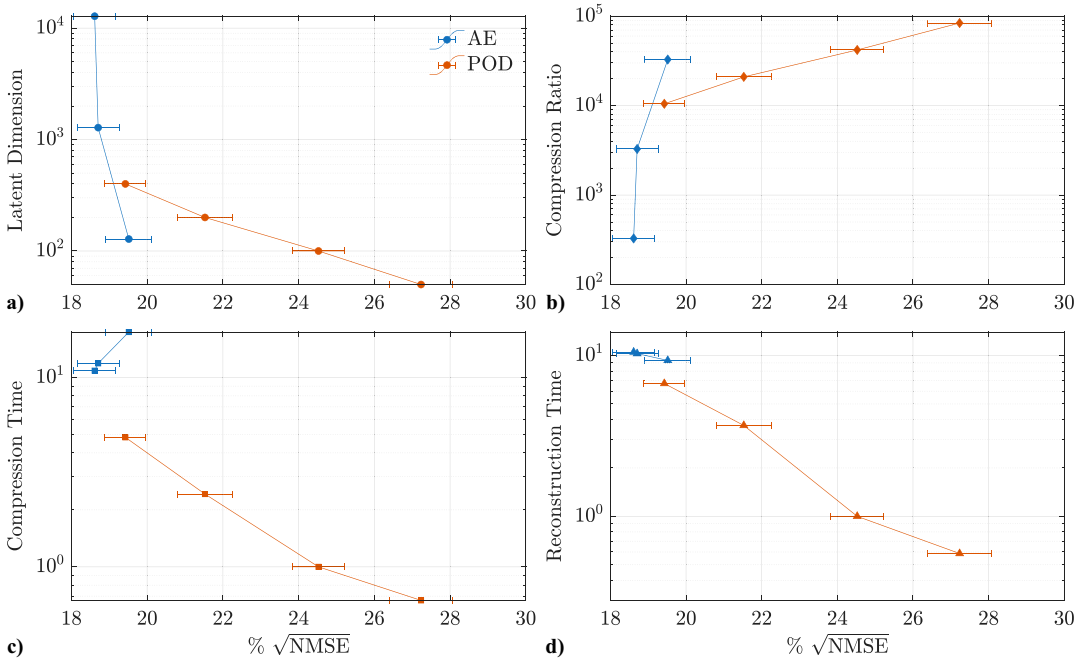



**Fig. 4 Challenge 1.2: Application of compression: metrics of success in 3D fields of the streamwise velocity fluctuations in a turbulent boundary layer using 3D-CNN-AE and POD. The bars represent the standard deviation of the error. The compression and reconstruction times are normalized using the POD method with 100 modes.**

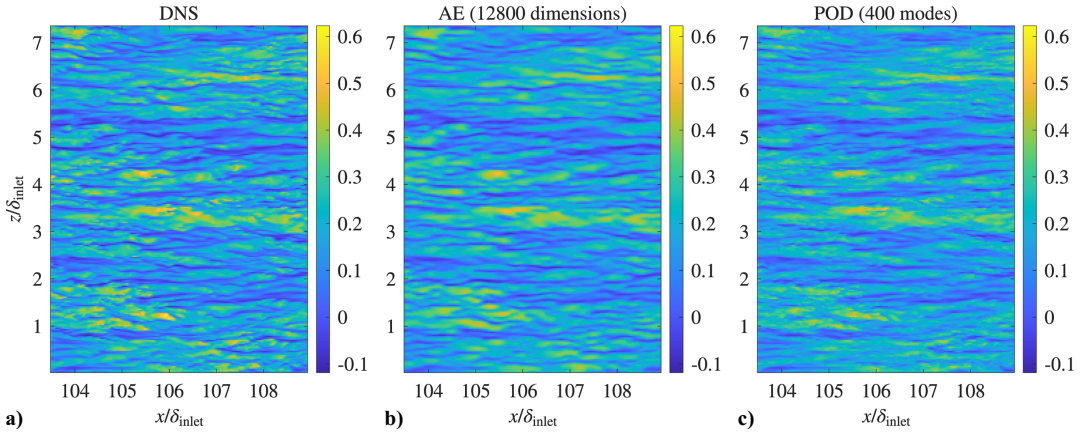



**Fig. 5 Challenge 1.2: Application of compression to 3D fields: Example of a 2D cut extracted from the 3D field of the streamwise velocity fluctuations (a), its reconstruction using a 3D-CNN-AE with a latent dimension of 12,800 (b), and reconstruction using 400 POD modes (c). The streamwise ($x$) and spanwise ($z$) coordinates are nondimensionalized by the boundary-layer thickness at the inlet of the simulation ($\delta_{\text{inlet}}$).**

results for the 3D case exhibit trends that differ from those in the 2D case, and the superiority of the 3D-CNN-AE is less pronounced. Although the 3D-CNN-AE yields slightly lower reconstruction errors, its overall behavior more closely mirrors that of POD. For example, both 3D-CNN-AE and POD achieve reconstruction errors of about 20% at compression ratios on the order of $10^4$.

A clear limitation of POD is its restriction to a maximum of 500 modes due to the method of snapshots. This constraint arises because the number of available POD modes is bounded by the number of training flowfields. In contrast, the 3D-CNN-AE does not suffer from this limitation, as its latent dimension can be increased independently of the number of training samples.

In all cases, the compression-reconstruction time remains on the order of a few seconds for both the 3D-CNN-AE and POD. Interestingly, the reconstruction error of the 3D-CNN-AE does not improve substantially when increasing the latent dimension from 128 to 12,800. This suggests that the performance bottleneck of the current architecture lies not in the latent dimensionality but rather in the convolutional layers.

It is important to note that the 3D-CNN-AE architectures employed in this study were not optimized or specifically tuned for the dataset. Substantial performance gains are likely achievable through careful architectural design and hyperparameter tuning.

## III. Challenge 2: Forecasting

### A. Introduction

The fundamental goal of forecasting in fluid mechanics is to predict the future evolution of a flowfield from a finite sequence of past observations. Forecasting plays a central role in applications where anticipating the flow state is critical, such as flow control, adaptive sensing, and reduced-order modeling.

Forecasting fluid flows poses distinct challenges arising from the complex, multiscale, and often chaotic nature of fluid motion. Even small modeling or measurement errors can grow rapidly in time, limiting the useful prediction horizon. Nevertheless, many flows exhibit coherent spatiotemporal structures or quasi-periodic components that can be exploited for prediction. For example, cavity flows exhibit Rossiter modes associated with feedback loops between the shear layer and the cavity interior, while pitching airfoils display periodic vortex shedding modulated by the prescribed kinematics. These temporal correlations motivate methods that can

Downloaded by University of Michigan on October 14, 2026 | http://arc.aiaa.org | DOI: 10.2514/1.J066928

Downloaded by University of Michigan on October 14, 2026 | http://arc.aiaa.org | DOI: 10.2514/1.J066928

identify and model dominant dynamics in an efficient, data-driven manner.

We pose two challenges that each highlight different capabilities and difficulties for forecasting fluid flows. The first dataset we consider is a turbulent cavity flow (Sec. III.C), characterized by broadband turbulence with dominant resonant tones, providing a stringent test of short-horizon predictability. The second is a laminar pitching airfoil (Sec. III.D), where the flow evolution is strongly coupled to prescribed kinematics, requiring parametric models that incorporate known input signals such as the instantaneous angle of attack. Together, these problems form a representative pair of forecasting benchmarks for reduced-complexity modeling of fluid flows.

### B. Baseline Methods

#### 1. *Dynamic Mode Decomposition*

Dynamic mode decomposition (DMD) approximates a linear operator $\boldsymbol{K}$ that advances the state from one snapshot to the next,

$$\tilde{\boldsymbol{q}}_{k+1} = \boldsymbol{K}\boldsymbol{q}_k \tag{3}$$

where $\boldsymbol{q}_k = \boldsymbol{q}(k\Delta t)$ denotes the snapshot at time $t = k\Delta t$, so that Eq. (3) represents propagation over a single time step. Various algorithms for computing DMD exist. Here, we adopt the exact DMD algorithm, known for its robustness and ability to handle nonsequential data where snapshots are provided in pairs. The process begins with two data matrices,

$$\boldsymbol{Q}_1^{M-1} = [\boldsymbol{q}_1, \boldsymbol{q}_2, \ldots, \boldsymbol{q}_{M-1}], \quad \boldsymbol{Q}_2^{M} = [\boldsymbol{q}_2, \boldsymbol{q}_3, \ldots, \boldsymbol{q}_M] \tag{4}$$

which can either consist of sequential but shifted snapshots or individual snapshot pairs (each pair separated by the same time step, $\Delta t$). The next step is to compute the singular value decomposition (SVD):

$$\boldsymbol{Q}_1^{M-1} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^* \tag{5}$$

Retaining only the $N$ largest singular values (and corresponding columns of $\boldsymbol{U}$ and $\boldsymbol{V}$) yields a rank-$N$ truncation, $\boldsymbol{U}_N$, $\boldsymbol{\Sigma}_N$, and $\boldsymbol{V}_N$, whose components are then used, together with the shifted data matrix, to form the approximation of the propagation matrix:

$$\tilde{\boldsymbol{K}} = \boldsymbol{U}_N^* \boldsymbol{Q}_2^M \boldsymbol{V}_N \boldsymbol{\Sigma}_N^{-1} \tag{6}$$

In the following, we do not explicitly denote the rank truncation and omit the subscript $N$. The eigendecomposition of this matrix,

$$\tilde{\boldsymbol{K}} = \boldsymbol{D}\boldsymbol{\Lambda}\boldsymbol{D}^{-1} \tag{7}$$

is then used to compute the DMD modes and eigenvalues. Given $\boldsymbol{D}$, we proceed by computing the matrix of DMD modes:

$$\boldsymbol{\Phi} = \boldsymbol{Q}_2^M \boldsymbol{V}\boldsymbol{\Sigma}^{-1}\boldsymbol{D} \tag{8}$$

The $i$th column of $\boldsymbol{\Phi}$ corresponds to the $i$th DMD mode, $\boldsymbol{\phi}_i$, while the $i$th diagonal entry of $\boldsymbol{\Lambda}$ in Eq. (7), $\lambda_i$, provides the mode's complex frequency, $\omega_i = \log(\lambda_i)/\Delta t$. Since these frequencies are complex, the temporal dynamics are generally characterized by an oscillatory component and an amplification or damping rate. This distinction makes DMD modes inherently dynamic, unlike POD modes, where temporal dynamics are represented through time-dependent expansion coefficients to describe a specific trajectory. Once $\tilde{\boldsymbol{K}} \approx \boldsymbol{K}$ is determined, one could, in principle, use Eq. (3) to advance the flowfield step by step in time, starting from an initial condition $\boldsymbol{q}_0$. By leveraging the linearity of Eq. (3), the initial condition can be directly propagated to any desired time $t$ by expressing it in the basis of DMD modes and evolving the solution as

$$\tilde{\boldsymbol{q}}(k\Delta t) = \sum_{i=1}^{N} \boldsymbol{\phi}_i e^{\omega_i k\Delta t} b_i \tag{9}$$

where the solution is represented as the sum of the $N$ retained DMD modes, each evolving according to its complex frequency $\omega_i$, starting from the initial amplitude and phase determined by $b_i$. The vector of complex initial amplitudes is obtained by solving the least-squares regression problem

$$\boldsymbol{b} = \boldsymbol{\Phi}^{+}\boldsymbol{q}_0 \tag{10}$$

Here, it is most conveniently expressed using the pseudo-inverse of the DMD mode matrix, $\boldsymbol{\Phi}^{+}$, which can be efficiently obtained via SVD.

#### 2. *Long Short-Term Memory Networks*

Long short-term memory (LSTM) networks [20] are a class of recurrent neural networks (RNNs) specifically designed to learn long-range temporal dependencies in sequential data. They are particularly well-suited for time-series forecasting problems, where the prediction at a given time benefits from information carried forward from many previous time steps. Unlike standard feedforward architectures such as multilayer perceptrons (MLPs, which are memoryless) and vanilla recurrent neural networks (RNNs, which struggle with vanishing gradients), LSTMs use a gated memory cell to retain and selectively update information over long time horizons, improving the learning of long-term temporal dependencies in dynamic forecasts. In addition to the flow state $\boldsymbol{q}_k$, an LSTM maintains a hidden state $\boldsymbol{h}_k$ used for prediction and a cell state $\boldsymbol{c}_k$ that acts as a longer memory. These internal states are updated as

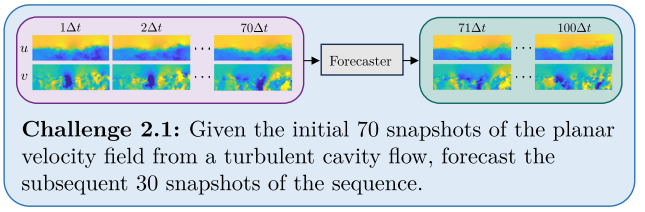
$$(\boldsymbol{h}_{k+1}, \boldsymbol{c}_{k+1}) = f_\theta(\boldsymbol{q}_k, \boldsymbol{h}_k, \boldsymbol{c}_k) \tag{11}$$

and the forecasted flow state is given as

$$\tilde{\boldsymbol{q}}_{k+1} = g_\theta(\boldsymbol{h}_{k+1}) \tag{12}$$

where $f_\theta$ and $g_\theta$ are the LSTM state-update map and read-out map, respectively. The network parameters $\theta$ are optimized during training to minimize a predefined loss function, in our case, the mean-squared error of the forecast. LSTMs have become a standard choice for forecasting complex dynamic systems [21,22], including fluid flows [23,24].

### C. Challenge 2.1: Forecasting of a Turbulent Cavity Flow




**Challenge 2.1:** Given the initial 70 snapshots of the planar velocity field from a turbulent cavity flow, forecast the subsequent 30 snapshots of the sequence.

#### 1. *Data*

This challenge uses experimental data from a high-Reynolds-number flow over an open cavity, obtained using planar time-resolved particle image velocimetry (TR-PIV) by Zhang et al. [25]. This case represents real-world data from a complex, mixed broadband-tonal turbulent flow in the presence of measurement noise. The center plane of the open cavity has a length-to-depth ratio of $L/D = 6$ and a width-to-depth ratio of $W/D = 3.85$. The Reynolds number based on the cavity depth is $Re = \rho U_\infty D/\mu \approx 3.3 \times 10^5$, and the Mach number is $M = U_\infty/c_\infty = 0.6$, where $c_\infty$ is the far-field speed of sound. The sampling rate is $f_s = 16$ kHz, and the field of view is resolved by $154 \times 51$ points. The database comprises 16,000 snapshots of the instantaneous streamwise ($u$) and wall-normal ($v$) velocity. For further details on the experimental setup, refer to Zhang et al. [25].

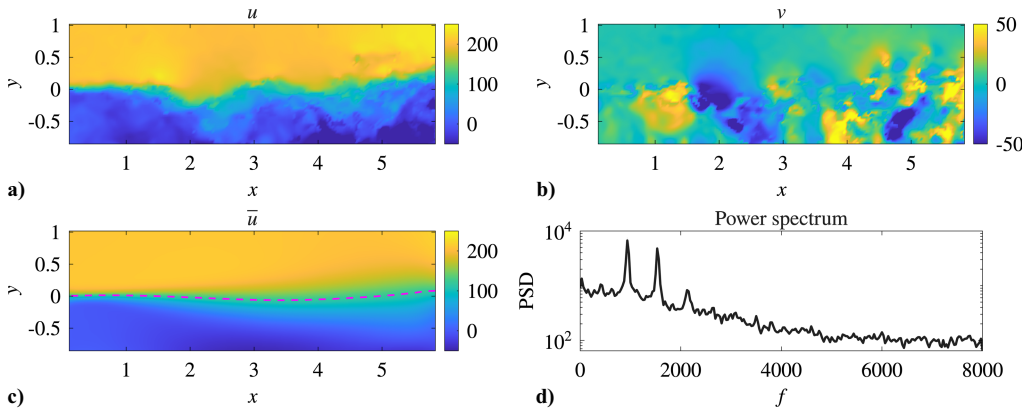



**Fig. 6 Challenge 2.1: Instantaneous velocity field of experimental open cavity flow data by Zhang et al. [25]: a) streamwise velocity, $u$; b) wall-normal velocity, $v$; c) mean streamwise velocity and 120 m/s isoline (dashed magenta line); d) power spectrum at $(x,y) = (5,0)$. Velocities are in m/s.**

Downloaded by University of Michigan on October 14, 2026 | http://arc.aiaa.org | DOI: 10.2514/1.J066928

The instantaneous streamwise and wall-normal velocity fields from a representative snapshot are shown in Figs. 6a and 6b. The flow is evidently turbulent, as confirmed by the broadband-tonal power spectrum in Fig. 6d, computed at $(x, y) = (5, 0)$. Superimposed on the turbulent background are distinct tonal peaks corresponding to Rossiter modes, which arise from a resonance between shear-layer instability waves and upstream-traveling acoustic waves. These acoustic waves originate at the trailing edge and synchronize with the shear-layer instabilities, thereby completing the resonance loop [26].

The training set contains 12,800 snapshots, and the test set includes the remaining 3200 snapshots. This test block is partitioned into 32 sequences of 100 consecutive snapshots. Participants are provided with the first 70 snapshots of every test sequence, and the final 30 snapshots are withheld and must be forecast.

To motivate the challenge, we first estimate the time interval over which realistic predictions can be expected by considering the characteristic time and length scales of the setup. Given a Mach number of 0.6, a cavity length of 0.159 m, a measurement time step of $\Delta t = \frac{1}{f_s} = 6.25 \times 10^{-5}$ s, and assuming a speed of sound of 343 m/s, we estimate that a fluid particle traveling at the freestream velocity would pass through the field of view in approximately 12 time steps. However, the dominant energetic structures in this flow are associated with the Rossiter cycle and are linked to the shear-layer dynamics. As a representative velocity scale for these dynamics, the 120 m/s isoline, shown as the magenta dashed line in Fig. 6c, closely follows the center of the shear layer. Assuming this velocity approximates the convection speed of the relevant structures in the shear layer, we estimate a flow-through time of about $21\Delta t$. This provides a rough upper bound for realistic predictability, suggesting that forecasts beyond ∼20 time steps are likely to degrade significantly. That being said, predictive methods may benefit from the inclusion of low-frequency components present in cavity flows, known as centrifugal modes. To account for their potential influence, we allow for a longer prediction horizon.

### 2. *Baseline Results*

As our classical baseline method, we use DMD, as described in Sec. III.B.1, to forecast the cavity flow data. We use $M = 70$ consecutive snapshots to construct two data matrices according to Eq. (4), for example, $\boldsymbol{Q}_1^{69}$ and $\boldsymbol{Q}_2^{70}$, and build a DMD model that uses the first 70 snapshots to predict the next 30. It is important to note that the DMD model does not "learn" from large datasets; instead, it performs a one-shot forecast based on the data used to compute it. For each of the 32 segments of the testing data, we construct a DMD model following Eqs. (5–8). The initial mode amplitudes $\boldsymbol{b}$ are then computed via Eq. (10) and used in Eq. (9) to evolve the system forward in time by 30 steps. We retain $N = 25$ out of the $M - 1 = 69$ available modes to prevent overfitting.

As our baseline machine-learning method, we use LSTM regression neural networks, as introduced in Sec. III.B.2. We use MATLAB's standard LSTM implementation [27], as it is widely accessible, well-documented, and reproducible. Because direct prediction in physical space is computationally prohibitive, we predict in the space of POD coefficients. Specifically, we perform standard space-only POD for dimensionality reduction and use the POD coefficients as latent variables in the machine-learning model. The space-only POD modes are computed from the data matrix containing the full training dataset, that is, $\boldsymbol{Q}_1^{12,800}$ in the notation of Eq. (4). For brevity, we refer to the full training data matrix as $\boldsymbol{Q}$ and approximate it via the economy SVD, $\boldsymbol{Q} \approx \boldsymbol{\Phi}_N \boldsymbol{\Sigma}_N \boldsymbol{V}_N^T$, where $N = 25$ and $\boldsymbol{\Phi}_N$, $\boldsymbol{\Sigma}_N$, and $\boldsymbol{V}_N$ contain the $N$ largest singular values and their corresponding left and right singular vectors. The left singular vectors, $\boldsymbol{\Phi}$, are the POD modes, whereas the right singular vectors, $\boldsymbol{V}$, provide the temporal POD expansion coefficients $\boldsymbol{A} = \boldsymbol{\Sigma}_N \boldsymbol{V}_N^T$ that allow us to expand the data in terms of the POD modes as $\boldsymbol{Q} \approx \boldsymbol{\Phi}_N \boldsymbol{A}$. Written as the expansion of individual snapshots $\boldsymbol{q}_k = \boldsymbol{q}(k\Delta t)$ at the $k$th time instant as the product of spatial POD modes and time-varying coefficients, $\tilde{\boldsymbol{q}}_k = \sum_{i=1}^{N} a_i(k\Delta t)\boldsymbol{\phi}_i$, we can forecast the flowfield by predicting future expansion coefficients, $\boldsymbol{a}(t + \Delta t)$, $\boldsymbol{a}(t + 2\Delta t), \ldots$, and reconstructing the state with the precomputed modes. In particular, given the $M = 70$ consecutive snapshots of a realization of the flow from the test data, we form the data matrix $\boldsymbol{Q}_1^{70}$ and compute its expansion coefficients by projecting the data onto the modal basis as $\boldsymbol{A}_1^{70} = \boldsymbol{\Phi}_N^H \boldsymbol{Q}_1^{70}$. That is, we expand the new test data sequence into the truncated mode basis obtained from the training data. To make predictions from a segment of $M = 70$ consecutive snapshots, we form the coefficient matrix $\boldsymbol{A}_1^{70}$, use $\boldsymbol{A}_1^{69}$ as the input sequence for the LSTM, and use the shifted $\boldsymbol{A}_2^{70}$ as the target sequence. From the total of 12,800 snapshots, this yields a training sample of 182 realizations of shifted coefficient matrix pairs to train the LSTM network. Note that the network is trained to predict only the next time instant, $\boldsymbol{a}(71\Delta t)$, from the 70 instances provided (of which only 69 are used, as explained below) and is then applied to the shifted sequence that includes its own prediction to produce a total of 30 forecasted snapshots, that is, the remaining expansion coefficients $\boldsymbol{a}(72\Delta t), \ldots, \boldsymbol{a}(100\Delta t)$ required for the challenge. This procedure is often referred to as closed-loop forecasting.

The network is trained using the standard Adam optimizer with a constant learning rate of 0.001 for 200 epochs, with data shuffled each epoch. We set the mini-batch size to 26, so the training data are divided into seven mini-batches per epoch, resulting in a total of 1400 iterations. The state vector includes both velocity components. The LSTM configuration includes a sequence input layer, one LSTM layer with 128 hidden units, and a fully connected output layer. Following best practices, the coefficients are normalized to zero mean and unit variance, and the mean-square error is used as the loss function [27]. The LSTM was trained on data sequences of 70 snapshots, but because the network must be trained from the shifted sequences $\boldsymbol{A}_1^{69}$ and $\boldsymbol{A}_2^{70}$, the effective input sequences have length 69. In principle, the LSTM can learn dependencies spanning the entire 69-step history, but its effective memory length

is determined by what it learns through the gating mechanism. At inference, we initialize the network state with snapshots 2–70 of the testing sequence so that the final input to the network is the true snapshot at time step 70, and the first forecasted step is predicted from the last instance of the testing sequence. This reduces the initial error and subsequent error accumulation. We note this departure from the conventional practice (where, consistent with training, the last entry of the input sequence is predicted) because the cavity data are stationary, allowing the 69-long input sequence to be shifted arbitrarily.

To make quantitative comparisons between the full flowfields predicted by DMD and LSTM, we use the normalized mean-squared error (NMSE),

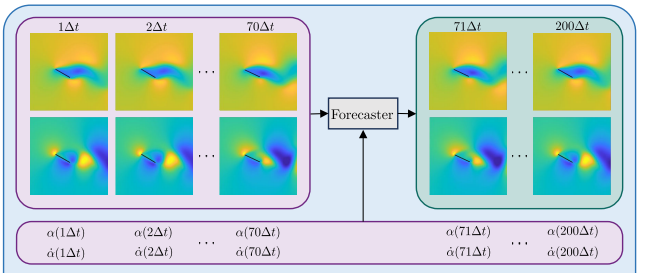
$$e_k = \frac{E\{\|\tilde{\boldsymbol{q}}_k - \boldsymbol{q}_k\|^2\}}{E\{\|\boldsymbol{q}_k\|^2\}} \tag{13}$$

where the expectation is an ensemble average.

The prediction errors for DMD and LSTM are shown in Fig. 7. Individual test-sequence errors (gray curves) display substantial spread for both methods, as reflected by the shaded standard-deviation bands. The mean error increases approximately monotonically at short lead times for both methods and then levels off. The direct comparison in Fig. 7c shows that DMD is slightly more accurate at the first predicted time step, but its error increases more rapidly thereafter, and LSTM attains a lower mean error over the subsequent 16 time steps. However, even the LSTM forecast exceeds an NMSE of one by lead time 13, indicating that the predictive capability of both methods is effectively exhausted beyond this horizon.

Figure 8 shows the $u$- and $v$-velocity components of the reference data (ground truth) and the corresponding DMD and LSTM forecasts at five representative lead times. The most apparent difference between DMD and LSTM is that the DMD forecasts retain much finer turbulent structure, closely resembling the reference data, whereas the LSTM forecasts appear significantly smoother. This is consistent with how the two models are constructed: DMD predictions lie in the linear subspace spanned by the DMD modes, which are themselves built as linear combinations of the snapshot data used to identify the model. By contrast, the LSTM operates in a reduced latent space obtained from a truncated POD representation. Recall that the LSTM is not applied to the raw flowfields but to the POD expansion coefficients; the flow is then reconstructed from a limited number of 25 POD modes, which are optimally averaged spatial structures and therefore inherently smoother than the original data. In other words, we use POD for dimensionality reduction and treat the POD coefficients as latent variables that are advanced in time by the LSTM. In summary, although the DMD forecasts display finer turbulent structure and might therefore be perceived as a more realistic representation of the flow, Fig. 7 shows that the much smoother LSTM forecasts achieve comparable and often lower prediction errors. As a final side note, simply increasing the number of retained POD modes in the LSTM model does not improve accuracy or make the forecasts resemble the DMD predictions or the ground truth more closely. Instead, the higher-dimensional latent space makes training more difficult and convergence less reliable.

### D. Challenge 2.2: Forecasting of Pitching Airfoils with Parametric Variations




**Challenge 2.2:** Given the initial 70 snapshots of vertical velocity from the flow over a periodically pitching airfoil at a set of pitching parameters (base angle of attack and pitching frequency), forecast the subsequent 130 snapshots for in- and out-of-sample pitching parameters.

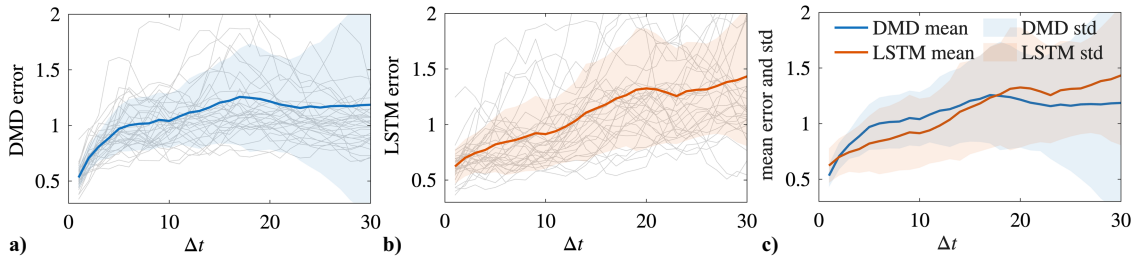



**Fig. 7 Challenge 2.1: Prediction error of the forecasts. Gray lines show the 32 individual test sequences; the thick line is their mean and the shaded band indicates standard deviation. a) DMD; b) LSTM; c) direct comparison of methods (mean NMSE and standard deviations).**

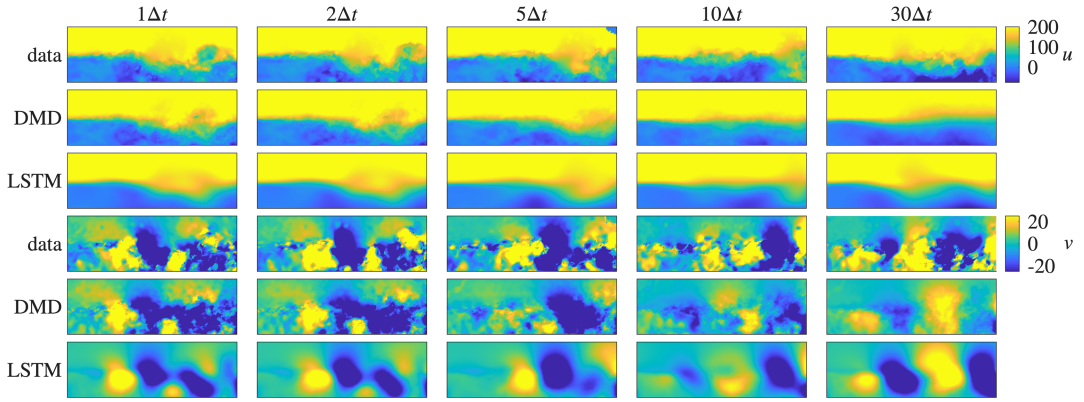



**Fig. 8 Challenge 2.1: Comparison of instantaneous flowfields for ground truth, DMD, and LSTM predictions. The streamwise velocity $u$ is shown on the top rows and the wall-normal velocity $v$ on the bottom rows. Forecast horizons correspond to the frame immediately after the input sequence and to +2, +5, +10, and +30 time steps.**

Downloaded by University of Michigan on October 14, 2026 | http://arc.aiaa.org | DOI: 10.2514/1.J066928

Downloaded by University of Michigan on October 14, 2026 | http://arc.aiaa.org | DOI: 10.2514/1.J066928

### 1. Data

This challenge uses DNS data of the flow over a flat-plate airfoil undergoing periodic pitching motions about its midchord, with the time-varying angle of attack

$$\alpha(t) = \alpha_0 - \alpha_P \sin(2\pi f_P t) \tag{14}$$

The Reynolds number (based on the freestream velocity and chord length) is 100. The simulations utilize an immersed boundary method [28,29], and the simulation parameters and numerical setup are identical to those described for the same configuration in Ref. [8]. Compared to the turbulent cavity flow in the first forecasting challenge, the dynamics in each simulation are relatively simple due to the low, laminar Reynolds number. The distinguishing features of this dataset that make forecasting nontrivial are the fact that it includes continuous actuation (pitching) at a range of different frequencies about base states that can be both stable (for $\alpha_0 = 25$ deg) and unstable with natural vortex shedding (for $\alpha_0 > 27$ deg). The natural vortex shedding frequency is approximately $f_N = 0.24$ for $\alpha_0 = 30$ deg, which falls within the range of pitching frequencies considered.

The dataset consists of the instantaneous streamwise ($u$) and transverse ($v$) velocity, collected in a window surrounding the airfoil, as indicated in Fig. 9a. The data are also downsampled relative to the simulations by a factor of four in the $x$ and $y$ directions, giving each snapshot a size of $60 \times 60$. Data are collected only after initial transients have decayed. For the purposes of developing models for predicting the evolution of this flowfield, we also assume knowledge of the pitching kinematics, characterized by $\alpha(t)$ and $\dot{\alpha}(t)$.

The training dataset consists of 16 different pitching profiles, with two base angles of attack $\alpha_0 \in \{25 \text{ deg}, 30 \text{ deg}\}$ and eight pitching frequencies $f_P \in \{0.05, 0.1, 0.2, 0.25, 0.3, 0.35, 0.4, 0.5\}$, where this frequency is nondimensionalized by the freestream velocity ($U_\infty$) and airfoil chord ($c$). The pitching amplitude is fixed at $\alpha_P = 5$ deg. The dataset used for testing consists of these parameters, as well as 15 additional parameter combinations not featured in the training data, as indicated in Fig. 9b. These are referred to as in-sample and out-of-sample cases, respectively. These additional out-of-sample parameter combinations include cases where either one or both of $\alpha_0$ and $f_P$ differ from the training cases and include a frequency that lies outside of the range used for training. Each case consists of 800 snapshots collected with time step $\Delta t = 0.1$ (nondimensionalized by $c$ and $U_\infty$).

Similar to the turbulent cavity flow example considered in Sec. III.C, we aim to identify models capable of forecasting future velocity fields, given an initial sequence of data. This initial sequence (consisting of 70 time steps of data) includes both the velocity field and airfoil kinematics ($\alpha(t), \dot{\alpha}(t)$), with the airfoil kinematics also provided for the forecasting time steps, as indicated in the Challenge 2.2 schematic. The forecast window consists of 130 time steps of data.

### 2. Baseline Results

We first train a model for the dynamics using DMD with control, a variant of DMD that enables the incorporation of known input signals into the assumed discrete-time linear dynamics. That is, Eq. (3) is modified such that the dynamics are

$$\tilde{\boldsymbol{q}}_{k+1} = \boldsymbol{K}\boldsymbol{q}_k + \boldsymbol{B}\boldsymbol{u}_k \tag{15}$$

This approach was first described in the context of DMD in Ref. [30]. Here, the input signal $\boldsymbol{u}_k$ consists of the instantaneous angle of attack $\alpha(k\Delta t)$ and pitch rate $\dot{\alpha}(k\Delta t)$. To reduce the dimensionality of the model, we project the data onto the leading $r = 20$ POD modes, computed from the concatenation of training data at all pitching frequencies. We emphasize that, while the DMD method used for Challenge 2.1 described in Sec. III.C identifies a separate model for each testing sequence, here we seek a single global model that can be used for different airfoil kinematics.

As in Sec. III.C, we additionally consider a simple LSTM neural network to perform this forecasting task. We similarly divide each training sequence into segments of 70 snapshots, using a 50% overlap between adjacent segments, and train a model to predict the state one time step into the future. The dimensionality of the problem is reduced by projecting the data onto the first 20 POD modes, as was also done for the DMD model. The learning rate, number of epochs, and number of LSTM blocks in the single hidden layer are the same as those used in Sec. III.C. We similarly quantify the error of the DMD and LSTM models using the NMSE, as given by Eq. (13).

Figure 10 shows the error for both the DMD and LSTM models across the forecasting window. The error is partitioned into those cases with parameters used for model training (in-sample) and those with different parameters (out-of-sample). Overall, the LSTM model has a lower total average error for the out-of-sample cases but a slightly larger average error for the in-sample cases (despite the LSTM model yielding a number of in-sample cases with lower error than any cases from the DMD model). The large average error for the out-of-sample cases in the DMD model is largely due to the four cases for which the average DMD model errors grow rapidly, which correspond to the cases where $\alpha_0 \in \{25 \text{ deg}, 30 \text{ deg}\}$ and $f_P \in \{0.23, 0.27\}$, which are all close to the natural vortex shedding frequency of the system. For the in-sample parameter values also used in the training, the average LSTM model error continues to grow over the forecast window, whereas the DMD model error appears to level off after approximately 80 time steps.

Figure 11 compares the flowfield predictions of both the DMD and LSTM models for two airfoil pitching parameter combinations. Figure 11a shows the case where $\{\alpha_0, f_P\} = \{30 \text{ deg}, 0.25\}$, which is included in the training set. This pitching frequency is also close to the natural vortex shedding frequency, and the vortex shedding dynamics are observed to lock on to the pitching. Accordingly, the response is periodic and is accurately captured by both models across the entire forecasting window. Figure 11b shows the case where $\{\alpha_0, f_P\} = \{30 \text{ deg}, 0.27\}$, which also exhibits periodic dynamics with vortex shedding phase-locked to pitching. These airfoil kinematics are not included in the training data, and the accuracy of both models is significantly worse. The DMD model predicts velocity fluctuations that grow too large over the forecasting horizon, though it appears to capture the correct phase of the flowfield fluctuations (most clearly in the $v$ component) across the

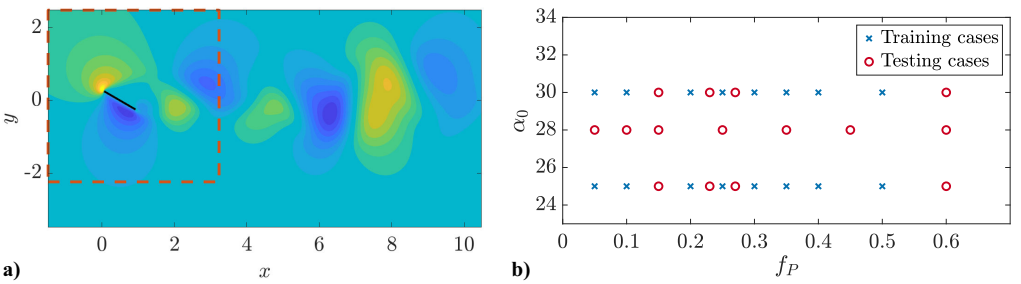



**Fig. 9 Challenge 2.2: a) Example flowfield snapshot (vertical velocity, $v$) for the pitching airfoil dataset. The red dashed box indicates the subregion of the domain used for the benchmark problem. b) Base angles of attack $\alpha_0$ and pitching frequencies used for the training (in sample) and testing (out of sample) datasets.**

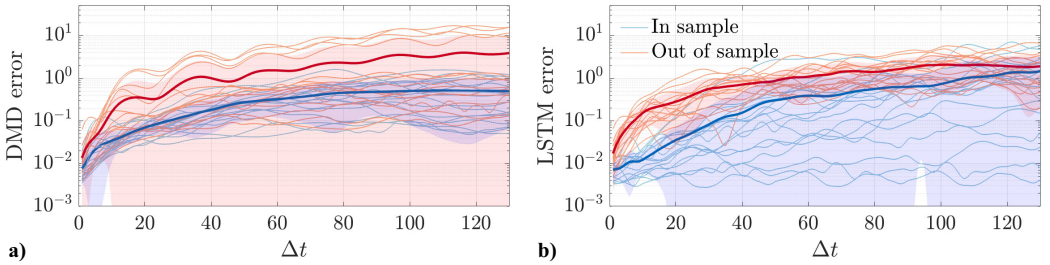



**Fig. 10 Challenge 2.2: Prediction error (NMSE) of the forecasts for Challenge 2.2, using a) DMD and, b) LSTM models. Thin lines show individual testing cases, divided between those parameter combinations included in (blue) and excluded from (red) the training dataset, with thick lines showing the average error across each of these two groups, and shaded bands indicating the standard deviation across all cases in each group. Results are plotted on a logarithmic scale to better differentiate the low-error cases.**

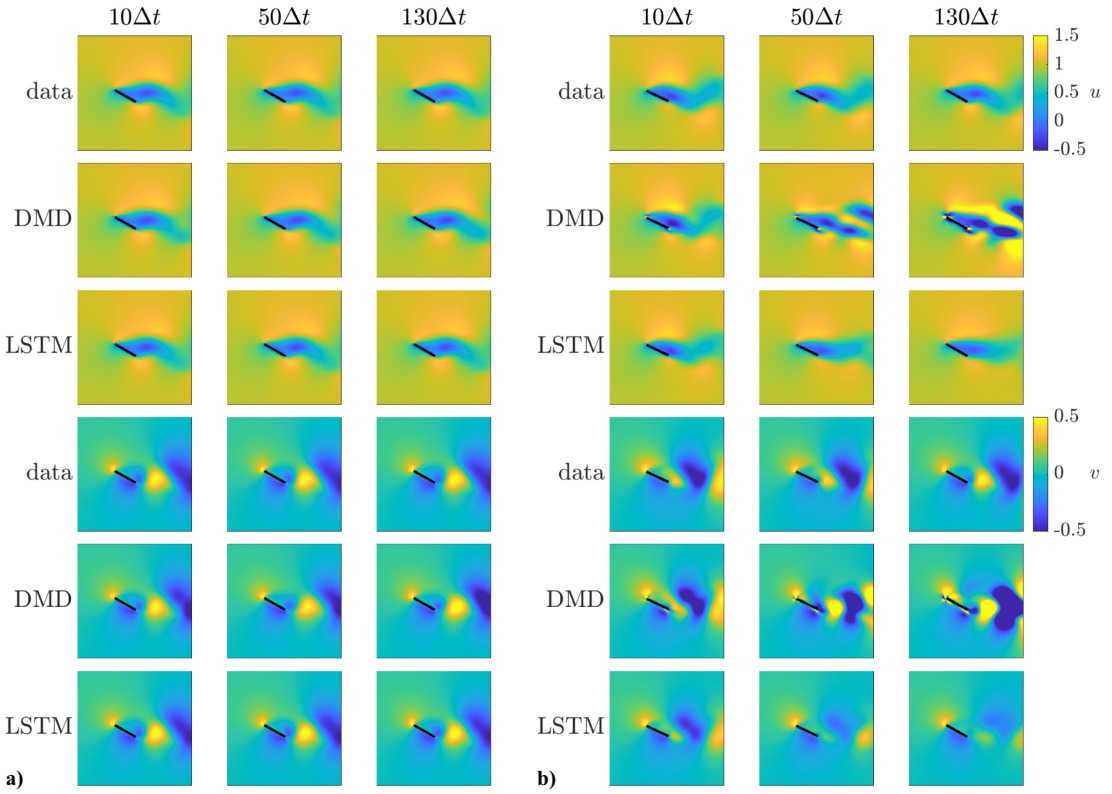



**Fig. 11 Comparison of instantaneous flowfields for ground truth, DMD, and LSTM predictions for Challenge 2.2, for pitching parameters a) $\alpha_0 = 30$ deg and $f_P = 0.25$ and b) $\alpha_0 = 30$ deg and $f_P = 0.27$. The streamwise velocity $u$ is shown on the top rows and the wall-normal velocity $v$ on the bottom rows. Columns denote forecast predictions at 10, 50, and 130 time steps.**

Downloaded by University of Michigan on October 14, 2026 | http://arc.aiaa.org | DOI: 10.2514/1.J066928

forecasting window. Note that this was one of the four cases with growing DMD model error in Fig. 10a. In contrast, the LSTM model predicts velocity field fluctuations that appear to decay over the forecasting window. While also not capturing the correct periodic dynamics, this results in lower error.

## IV. Challenge 3: Sensing

### A. Introduction

The fundamental goal of sensing in fluid mechanics is to reconstruct or estimate the state of a flow from partial, often sparse, measurement data. Sensing is critical for applications where comprehensive direct measurements are impractical or impossible. For example, in an experiment, particle image velocimetry might provide velocity data but not thermodynamic quantities, with limited resolution and a restricted field of view. In practical applications, measurements may be available for only a few probes, such as a few pressure sensors on the surface of an aerodynamic body. Fluid flows, particularly turbulence, pose distinct challenges for sensing. Turbulent flows are chaotic, multiscale, high-dimensional, and nonlinear. These characteristics limit the information about unseen flow states that can be deduced from limited measurements and violate the assumptions underlying many canonical state-estimation methods. Broadly speaking, flow sensing is a key capability for the effective monitoring, control, optimization, and understanding of fluid systems.

Using a turbulent jet and a turbulent boundary layer as representative examples, this challenge seeks new data-driven methods that address these challenges and enable improved flow sensing. In particular, the challenge is designed to probe two limiting cases for the available sensor data. The jet challenge uses measurements that are spatially sparse but temporally resolved, allowing the time history to be exploited to estimate the current targets. The boundary-layer challenge uses measurements that are not time-resolved but are spatially dense on the wall, allowing spatial patterns to be exploited to estimate the instantaneous velocity field away from the wall. Due to these differing limits, we use two sets of baseline methods: a Wiener filter and multilayer perceptron for the jet and linear stochastic estimation and a convolutional neural network for the boundary layer.

Downloaded by University of Michigan on October 14, 2026 | http://arc.aiaa.org | DOI: 10.2514/1.J066928

### B. Baseline Methods

#### 1. Linear Stochastic Estimation

Linear stochastic estimation (LSE) offers the optimal linear regressor for reconstructing a target signal from correlated but potentially noisy measurements [31]. Given a vector of measurements $\boldsymbol{y}$ (e.g., wall quantities such as shear stress or pressure) and corresponding target signals $\boldsymbol{z}$ (e.g., the velocity field), LSE seeks a linear transfer function $\boldsymbol{T}$ such that the instantaneous estimated target

$$\tilde{\boldsymbol{z}} = \boldsymbol{T}\,\boldsymbol{y} \tag{16}$$

minimizes the expected mean-squared error, $E\{\|\tilde{\boldsymbol{z}} - \boldsymbol{z}\|^2\}$. The optimal transfer function is

$$\boldsymbol{T} = \boldsymbol{C}_{zy}(\boldsymbol{C}_{yy} + \boldsymbol{N})^{-1} \tag{17}$$

where $\boldsymbol{C}_{zy}$ is the cross-correlation between the targets and the measurements, $\boldsymbol{C}_{yy}$ is the autocorrelation of the measurements, and $\boldsymbol{N}$ is the autocorrelation of any additive noise within the measurements. In practice, both measurements and targets are often projected onto a low-dimensional basis, such as proper orthogonal decomposition (POD), before applying LSE. Within this framework, the temporal coefficients of the measured quantities are linearly mapped into those of the target field using a transfer function constructed from their empirical correlations. This approach is mathematically equivalent to extended proper orthogonal decomposition (EPOD) [32,33]. LSE and EPOD have been applied successfully in turbulent jets [34], channels [35], as well as other wall-bounded flows [36], and serve here as a linear benchmark against which more advanced nonlinear estimators, such as convolutional neural networks, can be evaluated.

#### 2. Wiener Filter

The Wiener filter [37] constitutes the minimum mean-squared-error linear estimator between wide-sense stationary measurements $\boldsymbol{y}$ and targets $\boldsymbol{z}$. That is, we seek a linear transfer function $T(\tau)$ such that the estimated target

$$\tilde{\boldsymbol{z}}(t) = \int_{-\infty}^{\infty} \boldsymbol{T}(t-\tau)\boldsymbol{y}(\tau)\,\mathrm{d}\tau \tag{18}$$

minimizes the expected mean-squared error. Whereas LSE (in its typical form) uses only current, instantaneous measurements to estimate the current target, the Wiener filter uses the full measurement history. The transfer function that minimizes the error is given in the frequency domain as

$$\hat{\boldsymbol{T}}(\omega) = \boldsymbol{S}_{zy}(\boldsymbol{S}_{yy} + \boldsymbol{N})^{-1} \tag{19}$$

where $\boldsymbol{S}_{zy}$ is the cross-spectral density (CSD) between the targets and the measurements, $\boldsymbol{S}_{yy}$ is the CSD of the measurements, and $\boldsymbol{N}$ is the CSD of additive noise within the measurements. The time-domain transfer function $\boldsymbol{T}(\tau)$ is recovered via an inverse Fourier transform.

The formulation described so far is noncausal in the sense that future measurements are required to estimate the current target, as expressed by the infinite upper bound of the integral in Eq. (18). This is not an issue for postprocessing tasks where all data are simultaneously accessible. In other sensing tasks, such as system monitoring or closed-loop control, future data are not available. Attempting to bypass this requirement by simply truncating the noncausal ($\tau < 0$) parts of the transfer function forfeits the optimality of the estimator.

The optimal causal transfer function can be derived by restricting the integration limits in Eq. (18) to $(-\infty, 0]$ and again minimizing the expected mean-squared error. The transfer function that minimizes the error is

$$\hat{\boldsymbol{T}} = [\boldsymbol{S}_{zy}(\boldsymbol{S}_{yy} + \hat{\boldsymbol{N}})^{-1}_{-}]_{+}(\boldsymbol{S}_{yy} + \hat{\boldsymbol{N}})^{-1}_{+} \tag{20}$$

Here, $[\cdot]_{\pm}$ and $(\cdot)_{\pm}$ correspond to causal and noncausal components as determined by additive and multiplicative Wiener–Hopf decompositions, respectively [38–41]. Wiener filters are closely related to a number of other methods, including resolvent-based estimators and Kalman filters [41,42], and have been applied for flow estimation tasks in a variety of cases, including channels [43], airfoil wakes [44], and jets [45].

#### 3. Multilayer Perceptron

The multilayer perceptron (MLP) [46] is the most basic type of artificial neural network. These networks consist of two or more layers of nodes, with each node connected to every node in both the preceding and following layers. Conceptually, the network represents a static map between an input and output space. Within the context of sensing, the MLP provides a nonlinear map $\boldsymbol{\mathcal{M}}_\theta$ between the measurements $\boldsymbol{y}(t)$ and the targets $\boldsymbol{z}(t)$, i.e.,

$$\tilde{\boldsymbol{z}}(t) = \boldsymbol{\mathcal{M}}_\theta(\boldsymbol{y}(t)) \tag{21}$$

The network parameters $\theta$ are again optimized to minimize a desired loss function, in our case, the mean-squared estimation error.

To use the MLP as an estimator that accounts for measurement history, analogous to the Wiener filter, we redefine the input as a time-delay variable that consists of measurements over a past time interval. We note that time histories can be more elegantly accounted for using RNNs or LSTMs, among other advanced architectures; we use an MLP here to provide the simplest possible nonlinear extension of the Wiener filter, which should, in theory, reproduce the Wiener filter if the network is constructed with linear activation functions. Several examples of applications of MLPs to predict fluid flows are presented in Ref. [47].

#### 4. Convolutional Neural Network

Convolutional neural networks (CNNs) [48] are widely used in applications where the input and output domains share structural similarities, allowing the exploitation of spatial correlations in the data [49]. In the context of flow sensing, structured input might take the form of two-dimensional measurements (e.g., distributions of wall pressure or wall-shear stress), and structured outputs might include flowfields at target wall-normal locations (e.g., planar velocity fields). Similar to the MLP, the CNN provides a nonlinear map $\boldsymbol{\mathcal{C}}_\theta$ between the measurements $\boldsymbol{y}(t)$ and the targets $\boldsymbol{z}(t)$, i.e.,

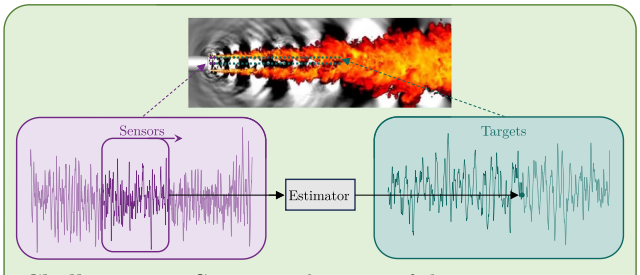
$$\tilde{\boldsymbol{z}}(t) = \boldsymbol{\mathcal{C}}_\theta(\boldsymbol{y}(t)) \tag{22}$$

The distinguishing feature of an CNN is that the map takes the form

$$\boldsymbol{\mathcal{C}}_\theta = \rho_L \circ \kappa_L \circ \cdots \circ \rho_1 \circ \kappa_1 \tag{23}$$

where each $\kappa_\ell$ is a convolutional operator, and each $\rho_\ell$ is a pointwise nonlinearity. That is, a CNN estimates the target $\boldsymbol{z}$ from measurement data $\boldsymbol{y}$ by repeatedly applying local, translation-equivariant convolutions with shared weights. This structure drastically reduces the number of network parameters $\theta$ and leverages the spatial relationship between nearby measurements, making CNNs effective when $\boldsymbol{y}$ has spatial structure. A detailed discussion of CNNs and their applications can be found in Ref. [50], and a comparison with LSE for flow sensing can be found in Ref. [51].

### C. Challenge 3.1: Sensing of a Turbulent Jet Using Sparse Measurements




**Challenge 3.1:** Given time histories of the axisymmetric streamwise velocity from four near-nozzle sensors in a turbulent jet, estimate the streamwise velocity at a series of downstream targets.

Downloaded by University of Michigan on October 14, 2026 | http://arc.aiaa.org | DOI: 10.2514/1.J066928

### 1. Data

This challenge uses large-eddy-simulation (LES) data for a turbulent jet issuing from a contoured convergent-straight nozzle [52]. The jet Mach number is $M_j = U_j/c_j = 0.9$ and the jet is isothermal ($T_j/T_\infty = 1.0$), where $U$ is the mean streamwise velocity, $c$ is the speed of sound, $T$ is the mean temperature, and the subscripts $j$ and $\infty$ refer to the jet exit and freestream conditions, respectively. The Reynolds number is $Re_j = \rho_j U_j D/\mu_j \approx 1 \times 10^6$, where $D$ is the nozzle exit diameter, $\rho$ is density, and $\mu$ is dynamic viscosity. The simulation was conducted using the compressible flow solver *CharLES* [53], which solves the spatially filtered compressible Navier–Stokes equations on unstructured grids using a finite volume method. The grid used here contains approximately 16 million control volumes. The LES methodologies, numerical setup, and comparisons with measurements are described in Ref. [52]. The full database [8] contains 10,000 snapshots of the density, velocity, and pressure fields sampled every 0.2 acoustic time units ($tc_\infty/D$).

The sensor data for this challenge correspond to measurements of the axisymmetric streamwise velocity taken from the LES data at four probe locations, $(x/D, r/D) = (0.25, 0), (0.5, 0), (0.25, 0.5), (0.5, 0.5)$. To mimic the finite size and bandwidth of real sensors, each measurement is defined as a Gaussian-weighted spatial average, which also has the effect of filtering high-frequency components of the true velocity field. The Gaussians are centered at the four aforementioned positions and have variance $\sigma/D = 0.1$. Since we are working with the axisymmetric component of the velocity field, each sensor with $r/D > 0$ would be physically implemented as an azimuthal ring of sensors with sufficient density to isolate the azimuthal mean. These sensor data are to be used to estimate the axisymmetric streamwise velocity at a series of target locations along the jet centerline ($r/D = 0$) and lip line ($r/D = 0.5$). For consistency with the sensors, each target is also defined using the same Gaussian-weighted average. The streamwise locations of the targets span $x/D \in [0.5, 10]$ with spacing $\Delta x/D = 0.25$.

The training data consist of 8000 snapshots of the sensor and target data, while the test data contain the remaining 2000 snapshots. Participants are provided with the sensor test data, while the target test data are withheld and must be estimated. A requirement of this challenge is that methods must be causal in the sense that they require only past and current data to estimate the current state of the targets. To accommodate methods that use an interval of the previous time history to produce the current estimate, the first 200 snapshots of the sensor test data are excluded when computing the estimation error.

### 2. Baseline Results

To construct the causal Wiener filter using a history of 200 snapshots, cross-spectra were computed using Welch's method with 400 snapshots per block, 90% overlap, and a sixth-order sine window [54]. While the data are not explicitly corrupted with noise, we assume that each measurement signal contains white noise at an amplitude of 0.1% to regularize the transfer functions.

The MLP estimator was likewise constructed to minimize the MSE loss function using a history of 200 snapshots of the sensor readings as input. To maximize the available training samples, an overlap of 199 was used to define each temporal block, resulting in 9800 samples. Fifteen percent of these training samples were withheld for validation, giving 8330 training and 1470 validation samples in total. Both the input and target data were mean-subtracted and normalized by their standard deviation prior to training. We considered two different architectures, each with an input layer, three fully connected hidden layers with 512 nodes, and a fully connected output layer. The first MLP uses linear activation functions and therefore should theoretically reproduce the Wiener filter. The second MLP uses a nonlinear (hyperbolic tangent) activation function followed by a 20% dropout layer to enhance convergence. For both MLPs, the minibatch size was 64, the learning rate was 0.0001, and training was performed using an Adam optimizer. The model was found to be highly prone to overfitting, and we therefore use early stopping, i.e., the iteration with the minimum validation loss.

The performance of the estimator is evaluated using the normalized mean-squared error (NMSE),

$$e = \frac{E\{\|\tilde{z}(t) - z(t)\|^2\}}{E\{\|z(t)\|^2\}} \tag{24}$$

where the expectation is a time average. Results for the causal Wiener filter and the two MLPs are shown in Fig. 12 as a function of $x/D$ along the jet centerline (a) and lip line (b). A striking, and perhaps unexpected, feature of these results is that the nonlinear MLP does not outperform the Wiener filter and, in fact, produces higher errors for most targets. This highlights the need for sensing algorithms that better leverage patterns within the available time-series data. Similarly, the linear MLP does not faithfully recover the Wiener filter, i.e., its approximation of the theoretical Wiener filter is less accurate than that obtained using CSDs computed by Welch's method, as described above.

In all cases, the error increases with $x/D$ as the distance between the sensors and targets increases. The error is higher along the lip line than along the centerline; the former is turbulent for all streamwise positions, while the latter is contained within the comparatively laminar potential core up until around $x/D = 5.5$. Indeed, there is a clear slope discontinuity in the centerline error at this location for all three estimators, after which the centerline error quickly reaches similar order-one levels as the lip-line error. The figure additionally shows the standard deviation of each estimate. In all cases, it is very large, indicating that the instantaneous estimated target values can deviate sharply from the true values.

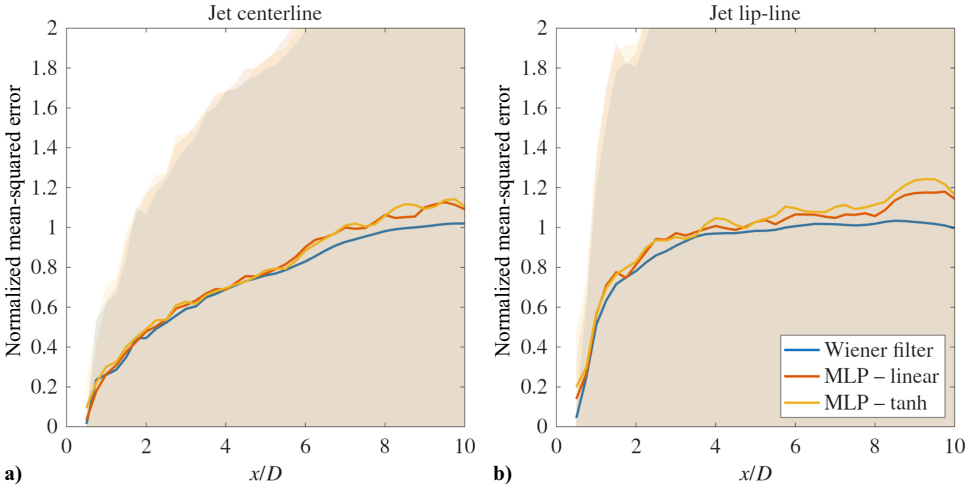
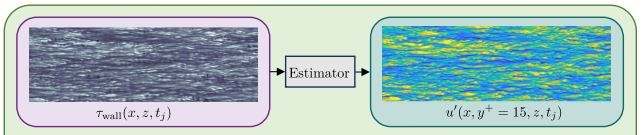



**Fig. 12 Challenge 3.1: Normalized mean-squared error as a function of the streamwise position $x/D$ along the jet a) centerline and b) lip line for a Wiener filter, linear MLP, and nonlinear MLP.**

Downloaded by University of Michigan on October 14, 2026 | http://arc.aiaa.org | DOI: 10.2514/1.J066928

### D. Challenge 3.2: Sensing of a Turbulent Boundary Layer from Wall Stress




**Challenge 3.2:** Given an instantaneous snapshot of the wall shear stress from a turbulent boundary layer, estimate the streamwise velocity at the same instant on a plane away from the wall.

#### 1. Data

This challenge uses data from a direct numerical simulation (DNS) of a zero-pressure-gradient incompressible turbulent boundary layer, specifically the BL1 case described in Ref. [8]. The full database contains 10,000 three-dimensional flowfields sampled every $\Delta t^+ \approx 1.5$ and spanning a physical time of 26 eddy turnovers.

From this database, the sensor data for this challenge correspond to the streamwise wall-shear stress fluctuation $\tau_w$ in a streamwise-spanwise ($x$–$z$) plane at a friction Reynolds number $Re_\tau \approx 400$ and discretized using $507 \times 150$ grid points. The target data consist of the streamwise velocity fluctuation at a wall-normal distance $y^+ = 15$ within the same $x$–$z$ domain, where $y^+ = yu_\tau/\nu$ is the inner-scaled wall-normal coordinate, $u_\tau = \sqrt{\bar{\tau}_w/\rho}$ is the friction velocity, $\bar{\tau}_w$ is the mean streamwise wall-shear stress, and $\rho$ is the fluid density. This is an important problem in experiments, where real-time wall measurements are less intrusive than introducing a probe to measure the flow in the near-wall region, particularly at high Reynolds numbers.

The training data consist of 1000 snapshots of the sensor and target data, while the test data contain an additional 250 snapshots. Participants are provided with the sensor test data, while the target test data are withheld and must be estimated. A requirement of this challenge is that a particular snapshot of the target velocity field is to be estimated using only the snapshot of the wall-shear stress sensor reading at the same instant in time.

#### 2. Baseline Results

The LSE estimator is constructed by using the training data to approximate the cross-correlation between the targets and measurements and the autocorrelation of the measurements. Although the DNS data are not explicitly corrupted with noise, we assume that the sensor data contain white noise at an amplitude of 0.1% to regularize the transfer functions.

**Table 2 Challenge 3.2 error metrics**

| Parameter | LSE | CNN |
|---|---|---|
| Normalized mean-squared error | 0.912 | 0.239 |
| Standard deviation | 1.141 | 0.432 |

The CNN architecture comprises 17 convolutional layers, each with a stride of 1 and the same padding, so the spatial resolution is preserved throughout while only the channel dimension is transformed. The number of filters increases over the first 8 layers from the initial value of 4 up to 32, then symmetrically decreases over the next 8 layers back to 1, forming an hourglass-shaped hierarchy of feature maps that enables multiscale representation and subsequent reconstruction of the target field. All intermediate layers use `tanh` activation to provide nonlinear transformations appropriate for signed flow quantities, whereas the final layer uses a `linear` activation to produce unconstrained regression outputs. Training is performed with an adaptive learning rate schedule that reduces the learning rate when the validation loss plateaus, combined with early stopping and checkpointing, so that the final deployed model is the one that attains the minimum validation loss, thereby limiting overfitting while retaining good generalization performance.

Figures 13a and 13b show one snapshot of the streamwise velocity fluctuation at $y^+ = 15$, i.e., the target to be sensed, and the wall shear stress, i.e., the sensor reading from the DNS. The corresponding estimates of the target velocity field produced by the LSE and CNN, and the difference compared to the DNS, are shown in Figs. 13c–13f. While both estimates show the presence of near-wall streaky structures, the LSE-estimated field significantly underpredicts their amplitude. Additionally, the CNN-estimated field contains more of the small-scale structures present in the DNS field. The NMSE from Eq. (24), the metric both methods were designed to minimize, averaged over the test data, is reported in Table 2, along with its standard deviation.

## V. Participation Guidelines

The objective of this challenge—to facilitate direct comparisons between data-driven methods for compression, forecasting, and

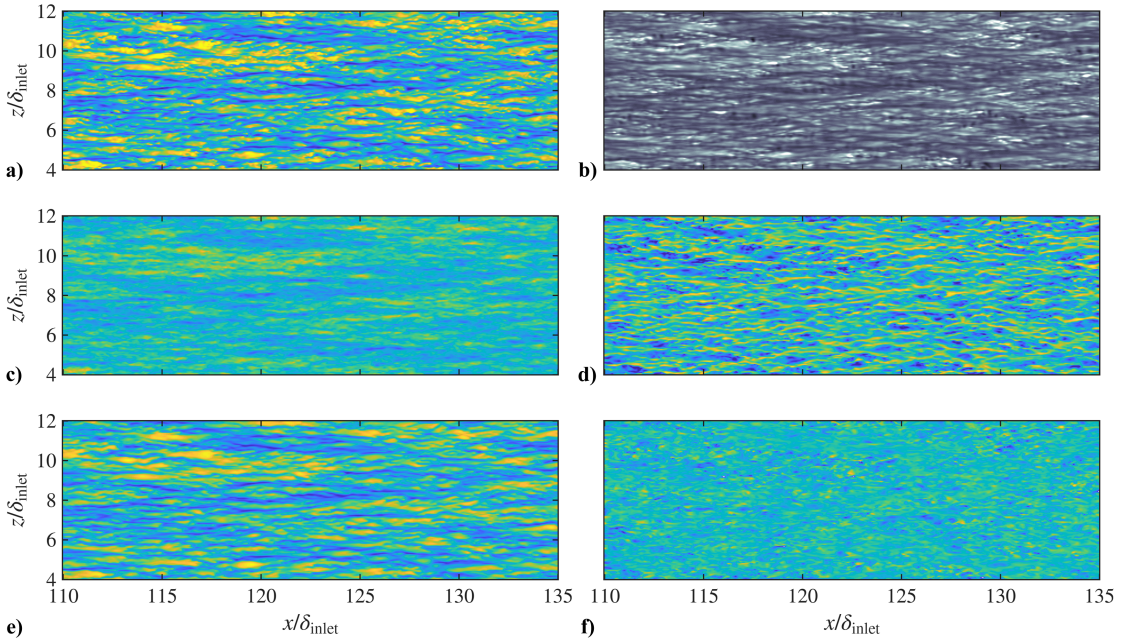



**Fig. 13 Challenge 3.2: A snapshot of the target, sensor, and error fields. a) Streamwise velocity fluctuation at $y^+ = 15$ from DNS; b) the wall shear stress from DNS; c) LSE estimate of the velocity field; d) difference between LSE and DNS velocity fields; e) CNN estimate of the velocity field; f) difference between CNN and DNS velocity fields.**

sensing of turbulent flows—will only be achieved with broad community participation. The challenge is open to anyone who wishes to participate, and we invite you to do so. Participation guidelines are summarized below; additional details, as well as periodic updates, can be found on the challenge webpage: fluids-challenge.engin.umich.edu.

### A. Data

All data required for the challenge are available on the webpage in a series of HDF5 files. Training data are provided for each challenge. Additionally, input data required for testing the methods are provided for the forecasting and sensing challenges, i.e., initial conditions and/or parameters and sensor measurements, respectively. Additional test data are withheld to ensure blind tests, as described in more detail below.

### B. Code

The website also provides example code for the baseline methods for each challenge. In addition to allowing you to reproduce the baseline results included in this paper, this code serves as a template for methods submitted to the challenge. Each baseline method has a main driver script that reads the training data (and the input test data if applicable), executes the method (by calling additional functions in a corresponding utilities directory), and writes out a results file. You are expected to release open-source code for each method submitted to the challenge, following this same structure; code that does not run or is not compliant with the required format will not be accepted. The code can be written in any language; indeed, we used multiple languages (a mix of MATLAB and Python) for the baseline methods to emphasize this point. We also provide an evaluation script for each challenge that will be used to calculate performance metrics for each submitted method, as described below.

### C. Blind Testing

To ensure fair comparisons, we will use blind tests to evaluate each method. The main driver script (described above) for each method must write out a results file containing the same information as included in the example baseline code, e.g., the estimated target values for the sensing challenges. You will send these results files to the relevant challenge point of contact (POC; see the webpage for details). For the forecasting and sensing challenges, the results file strictly contains data, i.e., predictions of the future flow state and targets, respectively. For the compression challenges, the results file will contain the encoder and decoder themselves. The POC will then execute the evaluation function, which reads both the results file and the necessary (withheld) test data and writes a metrics file that will be sent back to you.

### D. *AIAA Journal Virtual Collection* and Invited Presentations

In addition to the open-source code described above, the final deliverable of the challenge will be a Virtual Collection (analogous to a special issue) in the *AIAA Journal*. Specifically, for each challenge you complete, you will submit an article describing the proposed method and reporting its results, as contained in the metrics file supplied by the POC and compared to the provided baseline results. You are also welcome to optionally include results for additional test problems of your choosing that are relevant to a given challenge type. If you apply more than one method to the same challenge, they should be included in a single submission, and the methods should be appropriately compared and contrasted, in addition to comparisons with the baseline methods. If you complete multiple challenges, you should submit multiple articles. Each submission will undergo the standard *AIAA Journal* review process and should follow all *AIAA Journal* guidelines.

Given the expected scope, we anticipate relatively brief articles. The publishability of a submission is not contingent on the performance of the method(s). Indeed, we encourage negative results to achieve a balanced and comprehensive overview of available methods and the current state of the art. Instead, the review process will focus on the quality of the presentation and compliance with the challenge requirements. Finally, at the conclusion of the challenge, we will write an overview article summarizing the submissions to and results of the challenge. Additionally, we will host a special session at the AIAA SciTech and/or Aviation meetings, during which a subset of participants will be invited to present their methods and results.

### E. Administration

The challenge webpage contains, in addition to the data and example codes described above, additional details not included in this summary, including the POCs for each challenge. We request that you indicate your intention to participate by completing the survey on the webpage. In addition to helping us prepare, this will add you to a mailing list to ensure that you receive relevant updates. Finally, the webpage also outlines the challenge timeline and key dates and contains instructions for submitting your results. We look forward to receiving your submissions!

## Acknowledgments

Oliver T. Schmidt was funded in part by Air Force Office of Scientific Research Grant No. FA9550-22-1-0541 in support of Challenge 2.1. Aaron Towne was funded in part by Office of Naval Research Grant No. N00014-22-1-2561 in support of Challenge 3.1. Scott T. M. Dawson was funded in part by Department of Energy Grant No. DE-SC0025597 in support of Challenge 2.2. Adrian Lozano-Duran was funded in part by National Science Foundation Grant No. 2317254 and National Aeronautics and Space Administration Grant No. 80NSSC23K1498 in support of Challenges 1.1 and 1.2. We thank Junoh Jung for providing parts of the Wiener–Hopf code used in Challenge 3.1 and Andrés Cremades for his contributions to the CNN code used in Challenge 3.2.

## References

[1] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L., "ImageNet: A Large-Scale Hierarchical Image Database," *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Inst. of Electrical and Electronics Engineers, New York, 2009, pp. 248–255. https://doi.org/10.1109/CVPR.2009.5206848

[2] Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., and Bowman, S., "GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding," *2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Assoc. for Computational Linguistics, Brussels, Belgium, 2018, pp. 353–355. https://doi.org/10.18653/v1/W18-5446

[3] Moult, J., "A Decade of CASP: Progress, Bottlenecks and Prognosis in Protein Structure Prediction, Sequences and Topology/Nucleic Acids," *Current Opinion in Structural Biology*, Vol. 15, No. 3, 2005, pp. 285–289. https://doi.org/10.1016/j.sbi.2005.05.011

[4] Yagoubi, M., Danan, D., Leyli-Abadi, M., Brunet, J.-P., Mazari, J. A., Bonnet, F., Farjallah, A., Cinnella, P., Gallinari, P., Schoenauer, M., et al., "Neurips 2024 ml4cfd Competition: Harnessing Machine Learning for Computational Fluid Dynamics in Airfoil Design," arXiv preprint arXiv: 2407.01641, 2024. https://doi.org/10.48550/arXiv.2407.01641

[5] Rumsey, C., and Coleman, G., "Data-Driven Turbulence Modeling: Summary and Outcomes of the 2022 NASA Symposium," *High-Fidelity Large-Eddy Simulations (LES) and Direct Numerical Simulations (HiFiLeD) Symposium 2022*, European Research Community on Flow, Turbulence and Combustion (ERCOFTAC), NASA TM 20220014930, 2022.

[6] Ohana, R., McCabe, M., Meyer, L., Morel, R., Agocs, F., Beneitez, M., Berger, M., Burkhart, B., Dalziel, S., Fielding, D., et al., "The Well: A Large-Scale Collection of Diverse Physics Simulations for Machine Learning," *Advances in Neural Information Processing Systems*, Vol. 37, Dec. 2024, pp. 44,989–45,037.

[7] Kutz, J. N., Battaglia, P., Brenner, M., Carlberg, K., Hagberg, A., Ho, S., Hoyer, S., Lange, H., Lipson, H., Mahoney, M. W., et al., "Accelerating Scientific Discovery with the Common Task Framework," arXiv: 2511.04001, 2025. https://doi.org/10.48550/arXiv.2511.04001

Downloaded by University of Michigan on October 14, 2026 | http://arc.aiaa.org | DOI: 10.2514/1.J066928

Downloaded by University of Michigan on October 14, 2026 | http://arc.aiaa.org | DOI: 10.2514/1.J066928


[8] Towne, A., Dawson, S. T., Brès, G. A., Lozano-Durán, A., Saxton-Fox, T., Parthasarathy, A., Jones, A. R., Biler, H., Yeh, C.-A., Patel, H. D., et al., "A Database for Reduced-Complexity Modeling of Fluid Flows," *AIAA Journal*, Vol. 61, No. 7, 2023, pp. 2867–2892. https://doi.org/10.2514/1.J062203

[9] Lumley, J. L., *Stochastic Tools in Turbulence*, Academic Press, New York, 1970.

[10] Sirovich, L., "Turbulence and the Dynamics of Coherent Structures. I. Coherent Structures," *Quarterly of Applied Mathematics*, Vol. 45, No. 3, 1987, pp. 561–571. https://doi.org/10.1090/qam/910462

[11] Towne, A., Schmidt, O. T., and Colonius, T., "Spectral Proper Orthogonal Decomposition and Its Relationship to Dynamic Mode Decomposition and Resolvent Analysis," *Journal of Fluid Mechanics*, Vol. 847, July 2018, pp. 821–867. https://doi.org/10.1017/jfm.2018.283

[12] Berkooz, G., Holmes, P., and Lumley, J. L., "The Proper Orthogonal Decomposition in the Analysis of Turbulent Flows," *Annual Review of Fluid Mechanics*, Vol. 25, No. 1, 1993, pp. 539–575. https://doi.org/10.1146/annurev.fl.25.010193.002543

[13] Rowley, C. W., and Dawson, S. T., "Model Reduction for Flow Analysis and Control," *Annual Review of Fluid Mechanics*, Vol. 49, No. 1, 2017, pp. 387–417. https://doi.org/10.1146/annurev-fluid-010816-060042

[14] Colonius, T., and Towne, A., "Chapter 2—Modal Decomposition," *Data Driven Analysis and Modeling of Turbulent Flows*, edited by K. Duraisamy, Computation and Analysis of Turbulent Flows, Academic Press, New York, 2025, pp. 27–81. https://doi.org/10.1016/B978-0-32-395043-5.00008-5

[15] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning Representations by Back-Propagating Errors," *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, edited by D. E. Rumelhart, and J. L. McClelland, Vol. 1, MIT Press, Cambridge, MA, 1986, pp. 318–362.

[16] Hinton, G. E., and Salakhutdinov, R. R., "Reducing the Dimensionality of Data with Neural Networks," *Science*, Vol. 313, No. 5786, 2006, pp. 504–507. https://doi.org/10.1126/science.1127647

[17] Masci, J., Meier, U., Cireşan, D., and Schmidhuber, J., "Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction," *Artificial Neural Networks and Machine Learning—ICANN 2011, Lecture Notes in Computer Science*, edited by T. Honkela, W. Duch, M. Girolami, and S. Kaski, Vol. 6791, Springer, Berlin, 2011, pp. 52–59. https://doi.org/10.1007/978-3-642-21735-7

[18] Lee, K., and Carlberg, K. T., "Model Reduction of Dynamical Systems on Nonlinear Manifolds Using Deep Convolutional Autoencoders," *Journal of Computational Physics*, Vol. 404, March 2020, Paper 108973. https://doi.org/10.1016/j.jcp.2019.108973

[19] Solera-Rico, A., Sanmiguel Vila, C., Gómez-López, M., Wang, Y., Almashjary, A., Dawson, S. T., and Vinuesa, R., "$\beta$-Variational Autoencoders and Transformers for Reduced-Order Modelling of Fluid Flows," *Nature Communications*, Vol. 15, No. 1, 2024, p. 1361. https://doi.org/10.1038/s41467-024-45578-4

[20] Hochreiter, S., and Schmidhuber, J., "Long Short-Term Memory," *Neural Computation*, Vol. 9, No. 8, 1997, pp. 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

[21] 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, 2018, Paper 20170844. https://doi.org/10.1098/rspa.2017.0844

[22] Lindemann, B., Müller, T., Vietz, H., Jazdi, N., and Weyrich, M., "A Survey on Long Short-Term Memory Networks for Time Series Prediction," *Procedia Cirp*, Vol. 99, Jan. 2021, pp. 650–655. https://doi.org/10.1016/j.procir.2021.03.088

[23] Hasegawa, K., Fukami, K., Murata, T., and Fukagata, K., "Machine-Learning-Based Reduced-Order Modeling for Unsteady Flows Around Bluff Bodies of Various Shapes," *Theoretical and Computational Fluid Dynamics*, Vol. 34, No. 4, 2020, pp. 367–383. https://doi.org/10.1007/s00162-020-00528-w

[24] Rahman, S. M., Pawar, S., San, O., Rasheed, A., and Iliescu, T., "Nonintrusive Reduced Order Modeling Framework for Quasigeostrophic Turbulence," *Physical Review E*, Vol. 100, No. 5, 2019, Paper 053306. https://doi.org/10.1103/PhysRevE.100.053306

[25] Zhang, Y., Cattafesta, L. N., and Ukeiley, L., "Spectral Analysis Modal Methods (SAMMs) Using Non-Time-Resolved PIV," *Experiments in Fluids*, Vol. 61, No. 11, 2020, pp. 1–12. https://doi.org/10.1007/s00348-020-03057-8

[26] Rossiter, J. E., "Wind Tunnel Experiments on the Flow over Rectangular Cavities at Subsonic and Transonic Speeds," RAE TR 64037, Bedford, England, U.K., Oct. 1964.

[27] MathWorks, "Time Series Forecasting Using Deep Learning," 2024, https://www.mathworks.com/help/deeplearning/ug/time-series-forecasting-using-deep-learning.html [retrieved 13 March 2024].

[28] Taira, K., and Colonius, T., "The Immersed Boundary Method: A Projection Approach," *Journal of Computational Physics*, Vol. 225, No. 2, 2007, pp. 2118–2137. https://doi.org/10.1016/j.jcp.2007.03.005

[29] Colonius, T., and Taira, K., "A Fast Immersed Boundary Method Using a Nullspace Approach and Multi-Domain Far-Field Boundary Conditions," *Computer Methods in Applied Mechanics and Engineering*, Vol. 197, Nos. 25–28, 2008, pp. 2131–2146. https://doi.org/10.1016/j.cma.2007.08.014

[30] Proctor, J. L., Brunton, S. L., and Kutz, J. N., "Dynamic Mode Decomposition with Control," *SIAM Journal on Applied Dynamical Systems*, Vol. 15, No. 1, 2016, pp. 142–161. https://doi.org/10.1137/15M1013857

[31] Adrian, R. J., "Stochastic Estimation of Conditional Structure: A Review," *Applied Scientific Research*, Vol. 53, Nos. 3–4, 1994, pp. 291–303. https://doi.org/10.1007/BF00849106

[32] Borée, J., "Extended Proper Orthogonal Decomposition: A Tool to Analyse Correlated Events in Turbulent Flows," *Experiments in Fluids*, Vol. 35, No. 2, 2003, pp. 188–192. https://doi.org/10.1007/s00348-003-0656-3

[33] Encinar, M. P., and Jiménez, J., "Logarithmic-Layer Turbulence: A View from the Wall," *Physical Review Fluids*, Vol. 4, No. 11, 2019, Paper 114603. https://doi.org/10.1103/PhysRevFluids.4.114603

[34] Tinney, C. E., Ukeiley, L. S., and Glauser, M. N., "Low-Dimensional Characteristics of a Transonic Jet. Part 2. Estimate and Far-Field Prediction," *Journal of Fluid Mechanics*, Vol. 615, Nov. 2008, pp. 53–92. https://doi.org/10.1017/S0022112008003601

[35] Guastoni, L., Güemes, A., Ianiro, A., Discetti, S., Schlatter, P., Azizpour, H., and Vinuesa, R., "Convolutional-Network Models to Predict Wall-Bounded Turbulence from Wall Quantities," *Journal of Fluid Mechanics*, Vol. 928, Dec. 2021, Paper A27. https://doi.org/10.1017/jfm.2021.812

[36] Guastoni, L., Balasubramanian, A. G., Foroozan, F., Güemes, A., Ianiro, A., Discetti, S., Schlatter, P., Azizpour, H., and Vinuesa, R., "Fully Convolutional Networks for Velocity-Field Predictions Based on the Wall Heat Flux in Turbulent Boundary Layers," *Theoretical and Computational Fluid Dynamics*, Vol. 39, No. 1, 2025, p. 13. https://doi.org/10.1007/s00162-024-00732-y

[37] Wiener, N., *Extrapolation, Interpolation, and Smoothing of Stationary Time Series*, MIT Press, Cambridge, MA, 1949.

[38] Hopf, E., "Über Die Zusammenhänge Zwischen Linearen Operationen und Verallgemeinerten Harmonischen Funktionen," *Mathematische Annalen*, Vol. 114, No. 1, 1937, pp. 161–186. https://doi.org/10.1007/BF01594171

[39] Noble, B., *Methods Based on the Wiener-Hopf Technique for the Solution of Partial Differential Equations*, Pergamon Press, London, 1958, p. 246.

[40] Daniele, V., and Lombardi, G., "Fredholm Factorization of Wiener–Hopf Scalar and Matrix Kernels," *Radio Science*, Vol. 42, No. 06, 2007, Paper RS6S01:1–19. https://doi.org/10.1029/2007RS003673

[41] Martini, E., Jung, J., Cavalieri, A. V., Jordan, P., and Towne, A., "Resolvent-Based Tools for Optimal Estimation and Control via the Wiener–Hopf Formalism," *Journal of Fluid Mechanics*, Vol. 937, April 2022, Paper A19. https://doi.org/10.1017/jfm.2022.102

[42] Towne, A., Lozano-Durán, A., and Yang, X., "Resolvent-Based Estimation of Space–Time Flow Statistics," *Journal of Fluid Mechanics*, Vol. 883, Jan. 2020, Paper A17. https://doi.org/10.1017/jfm.2019.854

[43] Arun, R., Bae, H. J., and McKeon, B. J., "Towards Real-Time Reconstruction of Velocity Fluctuations in Turbulent Channel Flow," *Physical Review Fluids*, Vol. 8, No. 6, 2023, Paper 064612. https://doi.org/10.1103/PhysRevFluids.8.064612

[44] Jung, J., and Towne, A., "Resolvent-Based Estimation of a Turbulent Wake," arXiv: 2507.18837, 2026.
https://doi.org/10.1017/jfm.2026.11444

[45] Maia, I. A., Jordan, P., Cavalieri, A. V. G., Martini, E., Sasaki, K., and Silvestre, F. J., "Real-Time Reactive Control of Stochastic Disturbances in Forced Turbulent Jets," *Physical Review Fluids*, Vol. 6, No. 12, 2021, Paper 123901.
https://doi.org/10.1103/PhysRevFluids.6.123901

[46] Rumelhart, D. E., Hinton, G. E., and Williams, R. J., "Learning Internal Representations by Error Propagation," Inst. for Cognitive Science, Univ. of California, San Diego, Rept. ICS-8506, La Jolla, CA, 1985.

[47] Srinivasan, P. A., Guastoni, L., Azizpour, H., Schlatter, P., and Vinuesa, R., "Predictions of Turbulent Shear Flows Using Deep Neural Networks," *Physical Review Fluids*, Vol. 4, No. 5, 2019, Paper 054603.
https://doi.org/10.1103/PhysRevFluids.4.054603

[48] LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P., "Gradient-Based Learning Applied to Document Recognition," *Proceedings of the IEEE*, Vol. 86, No. 11, 1998, pp. 2278–2324.
https://doi.org/10.1109/5.726791

[49] Long, J., Shelhamer, E., and Darrell, T., "Fully Convolutional Networks for Semantic Segmentation," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Inst. of Electrical and Electronics Engineers, Boston, MA, 2015, pp. 3431–3440.
https://doi.org/10.1109/CVPR.2015.7298965

[50] Goodfellow, I., Bengio, Y., and Courville, A., *Deep Learning*, MIT Press, Cambridge, MA, 2016.

[51] Nakamura, T., Fukami, K., and Fukagata, K., "Identifying Key Differences Between Linear Stochastic Estimation and Neural Networks for Fluid Flow Regressions," *Scientific Reports*, Vol. 12, No. 1, 2022, p. 3863.
https://doi.org/10.1038/s41598-022-07515-7

[52] Brès, G. A., Jordan, P., Jaunet, V., Le Rallic, M., Cavalieri, A. V. G., Towne, A., Lele, S. K., Colonius, T., and Schmidt, O. T., "Importance of the Nozzle-Exit Boundary-Layer State in Subsonic Turbulent Jets," *Journal of Fluid Mechanics*, Vol. 851, Sept. 2018, pp. 83–124.
https://doi.org/10.1017/jfm.2018.476

[53] Brès, G. A., Ham, F. E., Nichols, J. W., and Lele, S. K., "Unstructured Large Eddy Simulations of Supersonic Jets," *AIAA Journal*, Vol. 55, No. 4, 2017, pp. 1164–1184.
https://doi.org/10.2514/1.J055084

[54] Martini, E., Cavalieri, A. V. G., Jordan, P., and Lesshafft, L., "Accurate Frequency Domain Identification of ODEs with Arbitrary Signals," arXiv preprint arXiv:1907.04787, 2019.
https://doi.org/10.48550/arXiv.1907.04787

S. Brunton
*Associate Editor*

Downloaded by University of Michigan on October 14, 2026 | http://arc.aiaa.org | DOI: 10.2514/1.J066928