RESEARCH ARTICLE | MAY 22 2023

Reduced-order modeling of fluid flows with transformers

AmirPouya Hemmasian ; Amir Barati Farimani

Physics of Fluids 35, 057126 (2023)
https://doi.org/10.1063/5.0151515

Physics of Fluids ARTICLE pubs.aip.org/aip/pof

Reduced-order modeling of fluid flows with transformers

Cite as: Phys. Fluids **35**, 057126 (2023); doi: 10.1063/5.0151515
Submitted: 22 March 2023 · Accepted: 3 May 2023 ·
Published Online: 22 May 2023

View Online Export Citation CrossMark

AmirPouya Hemmasian and Amir Barati Farimani[a)]

AFFILIATIONS

Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, USA

[a)]Author to whom correspondence should be addressed: barati@cmu.edu

ABSTRACT

Reduced-order modeling (ROM) of fluid flows has been an active area of research for several decades. The huge computational cost of direct numerical simulations has motivated researchers to develop more efficient alternative methods, such as ROMs and other surrogate models. Similar to many application areas, such as computer vision and language modeling, machine learning and data-driven methods have played an important role in the development of novel models for fluid dynamics. The transformer is one of the state-of-the-art deep learning architectures that has made several breakthroughs in many application areas of artificial intelligence in recent years, including but not limited to natural language processing, image processing, and video processing. In this work, we investigate the capability of this architecture in learning the dynamics of fluid flows in a ROM framework. We use a convolutional autoencoder as a dimensionality reduction mechanism and train a transformer model to learn the system's dynamics in the encoded state space. The model shows competitive results even for turbulent datasets.

I. INTRODUCTION

The physics of fluid flows is generally governed by nonlinear partial differential equations (PDEs) with no known analytical solution. That is why computational methods have been the dominant approach in studying fluid mechanics among scientists and engineers.[1] There are countless applications for fluid mechanics motivating researchers and industries to invest in advancing the computational tools for studying them, such as aerodynamics,[2–5] air conditioning,[6–8] bio-fluids,[9–11] and heat transfer.[12–14] These classical approaches, however, come with a well-known trade-off between computational cost and accuracy. If traditional computational fluid dynamics (CFD) methods can be accelerated or replaced with faster and more efficient approaches, it can lead to a great step forward in any application relying on fluid flow simulations.

With the emergence of powerful data-driven methods and machine learning algorithms, as well as the increase in data available from experiments and numerical simulations, a new line of research now focuses on data-driven approaches to move toward this goal.[15–19] Some remarkable examples of such methods are spatiotemporal super-resolution of flow data,[20,21] modal decomposition and analysis,[22–24] turbulence modeling,[19,25,26] reduced-order modeling (ROM) of fluid flows,[27–30] and flow control.[31,32]

These techniques can also be accompanied by traditional numerical methods in order to accelerate them.[33,34] However, such models are still based on traditional CFD methods with the same trade-off, but faster with the aid of machine learning. Data-driven models can also leverage analytical knowledge about the physics of the system,[35,36] but these methods face serious challenges when dealing with high-dimensional data and higher-order derivatives.[37] Neural operators are another novel class of methods that use deep learning to learn the solution operator of partial differential equations.[38–40] These methods have a different perspective as they treat the problem as finding the mapping between infinite-dimensional function spaces and have achieved outstanding performance and have desirable properties such as being mesh-agnostic.

This work focuses on reduced-order modeling (ROM), which is basically approximating the evolution of physical systems in time in terms of coherent patterns and structures.[27] A reduced-order model generally consists of a dimensionality reduction mechanism and a dynamical model in the reduced state space. Proper orthogonal decomposition (POD) is a primary example of a dimensionality reduction algorithm in fluid dynamics, which is basically a linear projection of the high-dimensional state onto a low-dimensional subspace. Machine learning and deep learning have made significant contributions to the development of efficient and nonlinear dimensionality reduction techniques,[41,42] especially convolutional autoencoders (CAEs).[24,28,43] Looking at the success of CAEs in dimensionality reduction and feature extraction for fluid flows, we choose to utilize

one in our model. There are numerous choices for the dynamical model component in the literature including classical linear models, such as dynamic mode decomposition (DMD),[44,45] sparse Identification of Nonlinear Dynamics (SINDy),[46] and diverse types of neural network architectures.[28,30,47–53] In this work, we choose a transformer model as the dynamic component.

The transformer architecture,[54] with the attention mechanism[55] at its core, has achieved remarkable success in many research areas, such as natural language processing,[56,57] computer vision,[58,59] and molecular dynamics.[60] These accomplishments have inspired many researchers to utilize the transformer architecture to learn spatial relations[61–64] as well as temporal evolution[65–67] of physical systems. This work falls under the first category because the self-attention mechanism is utilized to learn how different regions in the spatial domain affect each other over time.

The classical approach of reduced-order modeling handles the spatial and temporal flow of information in two separate steps, making it a data-driven separation of variables. These approaches usually assume that the system consists of independent stationary spatial modes that change magnitude over time. That is why methods like POD might struggle with patterns that move in the spatial domain, such as a traveling wave. Ironically, there are many scenarios where fluid flows demonstrate such a behavior, including the famous Karman vortex street in the flow behind a cylinder, even though it can be modeled with these methods.

Reduced Order Model with transformER (ROMER) proposes a new perspective for reduced-order modeling of fluid flows. Instead of compressing the spatial information completely into a single vector with no explicit positional information about the presence of the patterns, our method encodes the state of the system as a set of feature vectors while maintaining the spatial order. The dynamical behavior of the system is, then, modeled as the interaction of these feature vectors together based on their values and their locations in the domain, which is done by the transformer architecture. This content-based modeling approach has been gaining popularity in the computer vision literature[68,69] and has shown promising results. We believe that utilizing a content-dependent mechanism like attention has great potential in modeling and processing fluid flow dynamics.

ROMER succeeds to learn the dynamics in the latent space with competitive performance to the well-known Fourier Neural Operator[39] and powerful convolutional networks, such as ResNet and U-Net. This is despite the fact that the input of our model is only the current time step, while the other models take the last ten time steps as their input. Moreover, our model is not trained in a recurrent setting, and the loss does not backpropagate through time. This leads to faster training (about ten times faster than FNO-2D on a GPU) but a larger error in the long time horizon. The model still exhibits strong performance in turbulent datasets with shorter time horizons.

The main drawbacks of this work to be investigated in the future are error accumulation over time as well as the lack of interpretability and insight into the physics of the system. The error accumulation is a typical drawback of any autoregressive model and can be improved by recurrent training and backpropagation of error through time but with a huge increase in the training cost. The issue of interpretability is also considered a common challenge for most deep learning algorithms. However, the authors believe in the potential of finding meaningful and intuitive representations learned by the transformer architecture as has been the case in areas, such as natural language processing and computer vision.

II. METHODOLOGY

As mentioned before, a reduced-order model is composed of two main components: dimensionality reduction and a dynamical model. In this work, we use a convolutional autoencoder for the former and a transformer architecture for the latter. We will now discuss the architecture and motivation of each model in detail.

A. The convolutional autoencoder

The architecture of the convolutional autoencoder is illustrated in Fig. 1(a) including the details about its components. As shown in the figure, an autoencoder is composed of two sub-models: the encoder and the decoder, each consisting of several blocks. In our model, each encoder block consists of a convolutional layer, a pooling layer, and a nonlinear activation function. The convolutional layer is the main component responsible for extracting features and important information from the input. The main purpose of the pooling layer is to reduce the spatial dimension of the tensors for the sake of memory

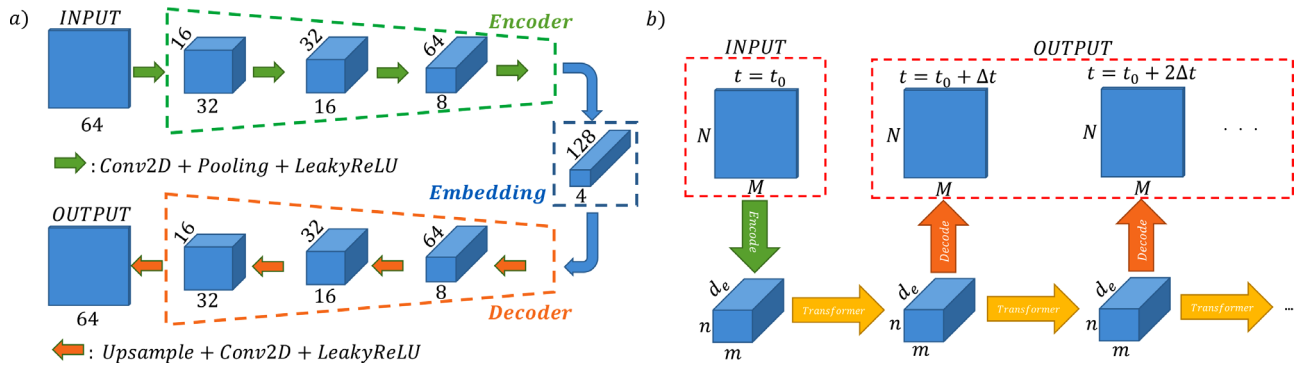

FIG. 1. (a) The architecture of the convolutional autoencoder for $d_{embed} = 128$. All Conv2D layers use kernel size 3, stride 1, and same padding. Average pooling and the bilinear method are used for the pooling and the upsampling layers, respectively. (b) The schematic mechanism of ROMER. The encoder maps the input onto the embedding space, in which the transformer propagates the system through time. The original state of the system at any time can be obtained approximately using the decoder.

and computational efficiency. Finally, the nonlinear activation function is necessary to learn a nonlinear featurization of the system. The output of the encoder is called the bottleneck of the autoencoder or the embedding tensor. We will learn the time dynamics of the system in this embedding space. The embedding tensor has three dimensions: two spatial dimensions and one feature dimension. The spatial resolution of the embedding tensor is much coarser than the original data, which can be thought of as an embedding of a certain region in the input frame, known as the receptive field.

The second sub-model, the decoder, basically is the reverse mechanism of the encoder and is composed of several decoder blocks. Each decoder block of our model consists of an upsampling layer, a convolutional layer, and a nonlinear activation function (except the final block). The output of the decoder will not exactly match the input of the encoder, but the autoencoder is trained so that the output is as close as possible to the input. The metric that we use in this work to measure the difference between the input and output of the autoencoder is the root mean square error (RMSE).

After training the autoencoder, we can use the encoder to obtain an embedding tensor from the original state of the system and use the decoder to reconstruct the original state. As mentioned before, we train a transformer model to learn the time dynamics of the model in the embedding space, as shown in Fig. 1(b). We will now explain the mechanism of the transformer model.

B. The transformer architecture

A transformer model consists of several transformer layers as shown in Fig. 2(a). The embedding tensor first needs to be flattened in the spatial dimensions and arranged as a set of vectors to be fed to the transformer. We denote each of these feature vectors as x_i, where i is the index of the corresponding vector. This will discard the spatial information, but we will later address this issue using positional embedding. The first and most important block in a transformer layer is the multi-head self-attention block, which will be explained later in detail. This is the component that learns the interactions and relations among the input vectors. After this layer, each output vector is added to the input, modeling an incremental change. Then, each vector is passed through an identical feed-forward neural network, and the output is again added to the input, modeling an incremental change independent from the other vectors.

Figure 2(b) provides a simple illustration of the attention mechanism. For each input vector x_i, three different vectors are obtained by applying a linear layer: query (q_i), key (k_i), and value (v_i) with dimensions d_k, d_k, and d_v, respectively. For all the inputs including x_i itself, we calculate the attention weights of the ith vector to other vectors (j) called a_{ij} using the following equations:

$$\tilde{a}_{ij} = \frac{q_i . k_j}{\sqrt{d_k}}, \tag{1}$$

$$a_{ij} = \frac{e^{\tilde{a}_{ij}}}{\sum_j e^{\tilde{a}_{ij}}}. \tag{2}$$

In Eq. (1), we take the inner product of q_i with every k_j to obtain $\tilde{a}_{ij}$. This is basically a similarity metric between q_i and k_j and represents how much x_i can be affected by x_j. The inner product is, then, divided by $\sqrt{d_k}$ for the sake of magnitude normalization. Then, we apply a softmax function to obtain a_{ij}. Because of the property of the softmax function, a_{ij} are positive values that sum up to 1, so they can be interpreted as a set of weights to be used in a weighted average. Finally, the effect of all the vectors on x_i is denoted as z_i calculated by taking a weighted average over the value vectors weighted by a_{ij} as shown in Eq. (3). In the figure, the red boxes in the figure denote the index j meaning that this operation is performed using all the input vectors,

$$z_i = \sum_j a_{ij} v_j. \tag{3}$$

There are some interesting similarities between POD and this mechanism since both rely on linear projections and linear combinations. The important difference is that the attention mechanism calculates the vectors dynamically based on the value of all the feature vectors and the correlating patterns that are detected by key and query vectors, as opposed to POD that extracts the dominant modes first using the whole dataset.

A multi-head self-attention layer is basically composed of several parallel self-attention layers as shown in Fig. 2(c). First, each

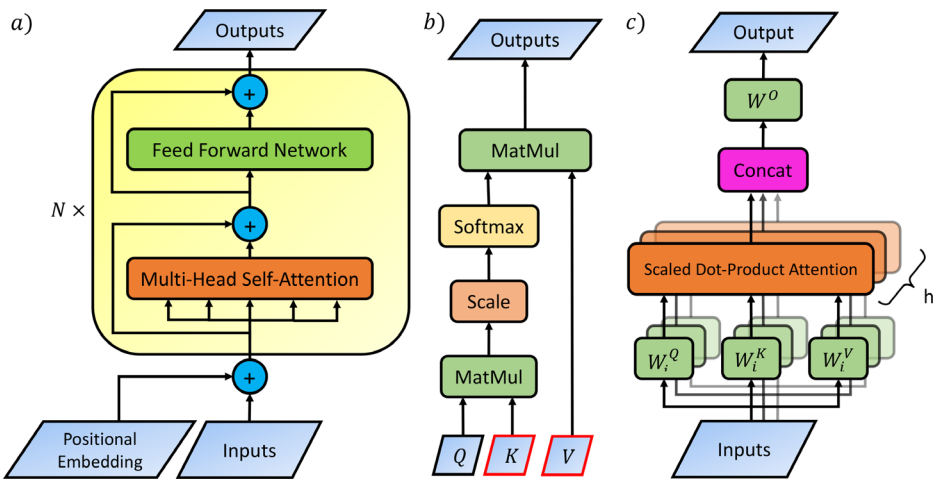

FIG. 2. The architecture of a transformer model.[54] In this work, $N = 6$, $h = 8$, and the other parameters are set as the default values in PyTorch. (a) The transformer model; (b) scaled dot-product attention; and (c) multi-head self-attention.

embedding vector is split into equal-length vectors, each being passed through a different self-attention layer. Finally, the outputs of the self-attention layers are concatenated together and passed to a final linear layer to obtain output vectors with the same size as the embedding vectors, so that they can be passed to the next layers. The weights of the linear layers are different in each head as shown in the figure, denoted by the index i. This mechanism can be summarized in the following equations:

$$MultiHead(X) = Concat(head_i, \ldots, head_h)W^O, \tag{4}$$

$$head_i = Attention(XW_i^Q, XW_i^K, XW_i^V). \tag{5}$$

There is still an important component left in the transformer model, which is the positional embedding. As mentioned before, the spatial information is discarded when the spatial dimensions of the embedding tensor are flattened. This issue is mitigated by adding a positional embedding vector to each input vector before it is passed to the transformer model. In the case of periodic boundary conditions, the positional embedding also should have a periodic nature. Hence, we use trigonometric functions as the primary positional features. Suppose the spatial dimension of the embedding tensor is $N_x \times Ny$. We define the positional feature vector for embedding vector x_{ij} as

$$\left\{ \sin\left(\frac{2\pi k}{N_x} i\right), \cos\left(\frac{2\pi k}{N_x} i\right), \sin\left(\frac{2\pi l}{N_y} j\right), \cos\left(\frac{2\pi l}{N_y} j\right) \right\} \tag{6}$$

for all values of $k = 1, 2, \ldots, N_x - 1$ and $l = 1, 2, \ldots, N_y - 1$. We, then, pass this to a trainable feed-forward neural network to obtain the final positional embedding with the same dimension as the embedding vectors, which is, then, added to the original embedding vectors before they are passed to the transformer model.

III. EXPERIMENTS AND DISCUSSION

In order to evaluate the proposed architecture, we use the three simulation datasets of the 2 D Navier–Stokes equations from the Fourier Neural Operator (FNO) paper[39] with details provided in Table I. There are two main reasons for this choice: First, these datasets are used to benchmark neural operators, which are a very powerful and novel class of algorithms, and accurate predictions for such datasets mean that the model can learn complex dynamics in turbulent regimes. Second, the simple geometry and periodic boundary conditions make these datasets ideal for the evaluation of our model since the flow field is evolving by itself and not affected by any enforced boundary condition or geometry. For each dataset, the convolutional autoencoder is first trained on the training dataset independently. After training, the parameters of the autoencoder are frozen, and the encoder and the decoder are used as shown in Fig. 1(b). Although the overall mechanism of our model uses the transformer recurrently, we train it in a simple supervised learning scheme, meaning that the training data for the transformer consist of input–output pairs s_t, s_{t+1}, where s_t is the embedding tensor at time t. This greatly reduces the computational cost of training compared to a recurrent training scheme where the error backpropagates through time. The CAE and the transformer in ROMER each take about 6 s per epoch to train, while FNO-2D takes more than 2 min. For both the autoencoder and transformer, we set the MSE loss as the objective function and use the Adam optimizer[70] with an initial learning rate of 0.001 to optimize the network parameters. We also utilize a learning rate scheduler to adjust the learning rate when the learning curve reaches a plateau. The patience and factor of the scheduler in PyTorch[71] are set to 5 and 0.2, respectively. The networks are trained until convergence.

TABLE I. Comparison of ROMER with benchmarks from the FNO paper.[39] 1000 training samples and 200 test samples are used in all experiments.

	Data 1	Data 2	Data 3	Parameters
ν	1e-3	1e-4	1e-5	—
$T(T_{pred})$	50 (40)	30 (20)	20 (10)	—
FNO-3D	0.86%	19.18%	18.93%	6 558 537
FNO-2D	1.28%	15.59%	15.56%	414 517
U-Net[72]	2.45%	20.51%	19.82%	24 950 491
TF-Net[73]	2.25%	22.53%	22.68%	7 451 724
ResNet[74]	7.01%	28.71%	27.53%	266 641
ROMER	7.70%	19.71%	17.45%	990 497

Unlike the other models included in the benchmark that take the last ten time steps as input to predict the next time step(s), our model only requires the current state of the system as input, which is similar to traditional numerical approaches, such as finite difference methods. If a certain model truly encapsulates the dynamics of the system, the current state of the system should be sufficient to predict the future. Models that take the state at several time steps as the input are in fact receiving some information about the time dynamics of the system as well.

The results shown in Table I demonstrate the capability of ROMER in learning the time-stepping dynamics of fluid flows even in turbulent regimes. For dataset 1 which is in the laminar regime and has the longest time horizon to predict, the accumulation of error over time is the main problem. That is why FNO-3D has the best performance by predicting all the next 40 time steps in one forward pass. However, the other datasets with turbulent behavior present a more challenging task. It is observed that ROMER has a competitive performance compared to FNO-2D in such scenarios. Even though error accumulation over time is a problem for all models except FNO-3D, ROMER and FNO-2D achieve the most accurate results. A qualitative evaluation of ROMER is also provided in Fig. 3 for dataset 3, which is most turbulent and challenging. As observed in the plots, the predictions closely match the ground truth.

Next, we are going to investigate the effect of the hyperparameters on the performance of the model on dataset 3. The other training settings are the same as before. The results can be found in Table II. As expected, smaller embedding dimensions lead to lower expressive powers and higher error, as well as a smaller model size. Increasing the embedding dimension would increase the capacity of the model and decrease the error but with the cost of a huge model size and parameter count. The number of heads, on the other hand, does not have a straightforward relationship with the performance and does not affect the model size. We can see that the best performance is for the model that uses four heads with a small margin, but decreasing the number of heads to two results in a higher error than eight heads. The number of layers also has a similar role as the embedding dimension, meaning that more layers usually result in higher

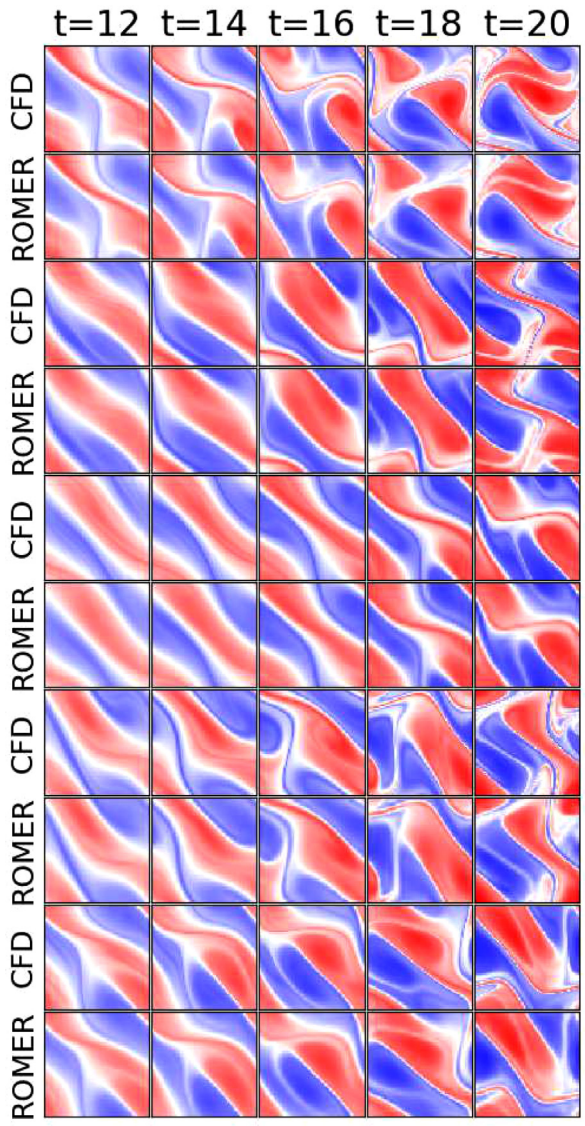

FIG. 3. Comparison between the output of ROMER and ground truth for some test samples from dataset 3.

accuracy but a larger model size. The choice of $N = 6$ seems to be a reasonable middle ground based on our experiments.

At last, we investigate the spatial resolution of the embedding tensor, meaning how much should the encoder downsample the data in the spatial domain for the transformer. If the encoder has three blocks, the spatial resolution of the embedding tensor will be 8×8, while $d_{emebd} = 128$ similar to the default setting. This means that the encoder is even more expressive than the default setting since the dimension of the embedding tensor will be of shape $8 \times 8 \times 128$. Now, the transformer has to learn the relationship among more region pairs in the domain. Despite larger embedding tensor and more potential relations for the transformer to learn, the final error does not seem to differ noticeably. In fact, increasing the embedding resolution to 16×16 leads to much worse performance. This means that the best choice is to let the autoencoder handle the processing of the smaller-scale features and patterns and leave the processing of the dynamics on the larger scale to the transformer model.

TABLE II. The relative RMSE on test data from dataset 3 with different choices of hyperparameters. Other hyperparameters are the same as before in each experiment.

	Value	Error	CAE params	Transformer params
Default	—	17.45%	194 177	796 320
d_{embed}	64	21.67%	48 705	201 696
	256	15.60%	775 425	3 165 216
h	2	18.63%	194 177	796 320
	4	16.83%	194 177	796 320
N	4	19.60%	194 177	532 384
	8	17.51%	194 177	1 060 256
res_{embed}	8×8	19.01%	185 217	801 216
	16×16	33.13%	148 865	812 544

ROMER is different from other reduced-order models in two main aspects. First, the total number of dimensions in the embedding tensor is not very compact compared to the typical dimension of the reduced dimensionality of other ROMs. The focus is compressing the spatial information into the channel dimension, so the dimensionality reduction is mostly about the spatial resolution rather than the sheer number of elements in the embedding tensor. Second, one can think of ROMER as a model that detects spatial correlations and patterns on the fly and calculate the next state according to the patterns that it detects at the moment. That is basically the functionality of the attention mechanism. The scaled dot-product attention is simply a correlation detection mechanism among the input vectors, while the softmax helps with numerical stability. While other reduced-order models first extract a compact representation based on the commonly observed patterns in the data, such as it is done with POD, ROMER utilizes the self-attention mechanism to detect the important correlating patterns at each time and proceed with the calculations accordingly, performing a dynamic or on-the-fly reduced-order modeling, or ROMing!

IV. CONCLUSION

This paper introduces ROMER, a hybrid architecture composed of a convolutional autoencoder and a transformer for reduced-order modeling of fluid flows in a novel perspective.

- Unlike many of the previous works in the literature, the input of this model is solely the current state of the system rather than a concatenation of the states at several consecutive time steps, making it conserve the Markov property of a physical system like traditional numerical solvers.
- ROMER achieves a competitive error magnitude compared to the Fourier Neural Operator and powerful convolutional neural networks (CNNs) like U-Net and ResNet.
- This work demonstrates the capability of the transformer architecture in capturing the spatial relations between different regions in the fluid flow.

There are several challenges to be addressed in future works before ROMER is applicable to more general settings. The current instance of the model is specific to a certain resolution, a simple geometry (rectangular domain), and periodic boundary conditions. In order to develop a mesh-agnostic model, the model will lean toward neural operators as that is the focus of such approaches. In order to develop similar models for more complex geometries and boundary conditions, the positional embedding and the mechanism of data compression and feature extraction need to be modified. If the state of the system can be compressed as a set of feature vectors in the spatial domain, the transformer architecture can be utilized to learn the time dynamics of those vectors. There are also several options on how to utilize the transformer architecture to process such data, which can be inspired by models with a transformer-based backbone in computer vision.

Another important aspect to be investigated is the sampling requirements for such algorithms. As opposed to traditional methods like DMD which rely on more concrete theoretical foundations, deep learning methods use stochastic gradient-based optimization and a complex functional form, which makes it difficult to obtain straightforward rules regarding sampling and convergence. Therefore, such requirements are investigated empirically based on the problem at hand. Exploring the sampling issue for such methods is a potential line of future work that is of high importance for practical applications where data collection is a major challenge.

ACKNOWLEDGMENTS

This work was supported by the National Science Foundation under Grant No. 1953222.

AUTHOR DECLARATIONS

Conflict of Interest

The authors have no conflicts to disclose.

Author Contributions

AmirPouya Hemmasian: Methodology (equal); Software (equal); Writing – original draft (equal). **Amir Barati Farimani:** Funding acquisition (equal); Supervision (equal); Writing – review & editing (equal).

DATA AVAILABILITY

The code and data used in this study are openly available at https://github.com/BaratiLab/ROMER, Ref. 75.

NOMENCLATURE

Abbreviations

CAE	Convolutional autoencoder
CFD	Computational fluid dynamics
CNN	Convolutional neural network
DMD	Dynamic mode decomposition
FNO	Fourier neural operator
MSE	Mean squared error
POD	Proper orthogonal decomposition
RMSE	Root mean squared error
ROM	Reduced-order model(ing)

Variables

a_{ij}	Attention weight of x_i to x_j
d_k	Dimension of the query and key vectors
d_v	Dimension of the value vectors
k_i	Key vector of the feature vector x_i
q_i	Query vector of the feature vector x_i
s_t	Embedding tensor at time step t
v_i	Value vector of the feature vector x_i
x_i	Feature vector of index i
z_i	Vector containing the effect of all feature vectors on x_i calculated by attention

REFERENCES

[1]C. Pozrikidis and D. Jankowski, *Introduction to Theoretical and Computational Fluid Dynamics* (Oxford University Press, New York, 1997), Vol. 675.
[2]D. R. Chapman, "Computational aerodynamics development and outlook," AIAA J. **17**, 1293–1313 (1979).
[3]H. Wang, F. Min, Z. Xie, J. Li, J. Dai, and Y. Yang, "Quantitative study of the control of hypersonic aerodynamics using millisecond pulsed discharges," Phys. Fluids **34**, 021701 (2022).
[4]Q. Zhou, M. M. Alam, S. Cao, H. Liao, and M. Li, "Numerical study of wake and aerodynamic forces on two tandem circular cylinders at Re = 103," Phys. Fluids **31**, 045103 (2019).
[5]J. Shang, Q. Zhou, M. M. Alam, H. Liao, and S. Cao, "Numerical studies of the flow structure and aerodynamic forces on two tandem square cylinders with different chamfered-corner ratios," Phys. Fluids **31**, 075102 (2019).
[6]H. Liu, S. He, L. Shen, and J. Hong, "Simulation-based study of COVID-19 outbreak associated with air-conditioning in a restaurant," Phys. Fluids **33**, 023301 (2021).
[7]L. Wu, X. Liu, F. Yao, and Y. Chen, "Numerical study of virus transmission through droplets from sneezing in a cafeteria," Phys. Fluids **33**, 023311 (2021).
[8]M. Abuhegazy, K. Talaat, O. Anderoglu, and S. V. Poroseva, "Numerical investigation of aerosol transport in a classroom with relevance to COVID-19," Phys. Fluids **32**, 103311 (2020).
[9]B. Mahapatra and A. Bandopadhyay, "Numerical analysis of combined electroosmotic-pressure driven flow of a viscoelastic fluid over high zeta potential modulated surfaces," Phys. Fluids **33**, 012001 (2021).
[10]B. Kada, A. A. Pasha, Z. Asghar, M. W. S. Khan, I. B. Aris, and M. S. Shaikh, "Carreau–Yasuda fluid flow generated via metachronal waves of cilia in a micro-channel," Phys. Fluids **35**, 013110 (2023).
[11]M. W. Saeed Khan, N. Ali, and O. A. Bég, "Thermal entrance problem for blood flow inside an axisymmetric tube: The classical Graetz problem extended for Quemada's bio-rheological fluid with axial conduction," Proc. Inst. Mech. Eng., Part H **236**, 848–859 (2022).
[12]H. Jin, Y. Wang, H. Wang, Z. Wu, and X. Li, "Influence of Stefan flow on the drag coefficient and heat transfer of a spherical particle in a supercritical water cross flow," Phys. Fluids **33**, 023313 (2021).
[13]N. Ali, M. W. S. Khan, and M. Sajid, "The Graetz–Nusselt problem for the curved channel using spectral collocation method," Phys. Scr. **96**, 055204 (2021).
[14]M. W. S. Khan, Z. Asghar, N. Ali, and W. Shatanawi, "Thermal entry problem for vócadlo fluid model bounded within passive tube and channel with axial conduction and viscous dissipation: A Graetz–Nusselt problem," Chin. J. Phys. **81**, 219–232 (2023).
[15]M. A. Mendez, A. Ianiro, B. R. Noack, and S. L. Brunton, *Data-Driven Fluid Mechanics: Combining First Principles and Machine Learning* (Cambridge University Press, 2023).

[16]R. Vinuesa and S. L. Brunton, "Enhancing computational fluid dynamics with machine learning," Nat. Comput. Sci. **2**, 358–366 (2022).
[17]R. Vinuesa and S. L. Brunton, "The potential of machine learning to enhance computational fluid dynamics," arXiv:2110.02085 (2021).
[18]S. L. Brunton, "Applying machine learning to study fluid mechanics," Acta Mech. Sin. **37**, 1718–1726 (2021).
[19]A. Beck and M. Kurz, "A perspective on machine learning methods in turbulence modeling," GAMM-Mitteilungen **44**, e202100002 (2021).
[20]K. Fukami, K. Fukagata, and K. Taira, "Super-resolution analysis via machine learning: A survey for fluid flows," arXiv:2301.10937 (2023).
[21]D. Shu, Z. Li, and A. B. Farimani, "A physics-informed diffusion model for high-fidelity flow field reconstruction," J. Comput. Phys. **478**, 111972 (2023).
[22]K. Taira, M. S. Hemati, S. L. Brunton, Y. Sun, K. Duraisamy, S. Bagheri, S. T. Dawson, and C.-A. Yeh, "Modal analysis of fluid flows: Applications and outlook," AIAA J. **58**, 998–1022 (2020).
[23]K. Fukami, T. Nakamura, and K. Fukagata, "Convolutional neural network based hierarchical autoencoder for nonlinear mode decomposition of fluid field data," Phys. Fluids **32**, 095110 (2020).
[24]T. Murata, K. Fukami, and K. Fukagata, "Nonlinear mode decomposition with convolutional neural networks for fluid dynamics," J. Fluid Mech. **882**, A13 (2020).
[25]K. Duraisamy, G. Iaccarino, and H. Xiao, "Turbulence modeling in the age of data," Annu. Rev. Fluid Mech. **51**, 357–377 (2019).
[26]K. Stachenfeld, D. B. Fielding, D. Kochkov, M. Cranmer, T. Pfaff, J. Godwin, C. Cui, S. Ho, P. Battaglia, and A. Sanchez-Gonzalez, "Learned simulators for turbulence," in *International Conference on Learning Representations* (2022).
[27]D. J. Lucia, P. S. Beran, and W. A. Silva, "Reduced-order modeling: New approaches for computational physics," Prog. Aerosp. Sci. **40**, 51–117 (2004).
[28]R. Maulik, B. Lusch, and P. Balaprakash, "Reduced-order modeling of advection-dominated systems with recurrent neural networks and convolutional autoencoders," Phys. Fluids **33**, 037106 (2021).
[29]S. Fresca, L. Dede, and A. Manzoni, "A comprehensive deep learning-based approach to reduced order modeling of nonlinear time-dependent parametrized PDEs," J. Sci. Comput. **87**, 61 (2021).
[30]P. Pant, R. Doshi, P. Bahl, and A. Barati Farimani, "Deep learning for reduced order modelling and efficient temporal evolution of fluid simulations," Phys. Fluids **33**, 107101 (2021).
[31]T. Duriez, S. L. Brunton, and B. R. Noack, *Machine Learning Control-Taming Nonlinear Dynamics and Turbulence* (Springer, 2017), Vol. 116.
[32]F. Ren, H.-B. Hu, and H. Tang, "Active flow control using machine learning: A brief review," J. Hydrodyn. **32**, 247–253 (2020).
[33]D. Kochkov, J. A. Smith, A. Alieva, Q. Wang, M. P. Brenner, and S. Hoyer, "Machine learning–accelerated computational fluid dynamics," Proc. Natl. Acad. Sci. **118**, e2101784118 (2021).
[34]O. Obiols-Sales, A. Vishnu, N. Malaya, and A. Chandramowliswharan, "CFDNet: A deep learning-based accelerator for fluid simulations," in *Proceedings of the 34th ACM International Conference on Supercomputing* (Association for Computing Machinery, 2020), pp. 1–12.
[35]G. Karniadakis, Y. Kevrekidis, L. Lu, P. Perdikaris, S. Wang, and L. Yang, "Physics-informed machine learning," Nat. Rev. Phys. **3**, 422–440 (2021).
[36]M. Raissi, P. Perdikaris, and G. Karniadakis, "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations," J. Comput. Phys. **378**, 686–707 (2019).
[37]A. Krishnapriyan, A. Gholami, S. Zhe, R. Kirby, and M. W. Mahoney, "Characterizing possible failure modes in physics-informed neural networks," in *Advances in Neural Information Processing Systems*, edited by M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan (Curran Associates, Inc., 2021), Vol. 34, pp. 26548–26560.
[38]L. Lu, P. Jin, G. Pang, Z. Zhang, and G. E. Karniadakis, "Learning nonlinear operators via deeponet based on the universal approximation theorem of operators," Nat. Mach. Intell. **3**, 218–229 (2021).
[39]Z. Li, N. B. Kovachki, K. Azizzadenesheli, B. Liu, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Fourier neural operator for parametric partial differential equations," in *International Conference on Learning Representations* (2021).
[40]N. Kovachki, Z. Li, B. Liu, K. Azizzadenesheli, K. Bhattacharya, A. Stuart, and A. Anandkumar, "Neural operator: Learning maps between function spaces," arXiv:2108.08481 (2021).
[41]B. Lusch, J. N. Kutz, and S. L. Brunton, "Deep learning for universal linear embeddings of nonlinear dynamics," Nat. Commun. **9**, 4950 (2018).
[42]S. Pan, S. L. Brunton, and J. N. Kutz, "Neural implicit flow: A mesh-agnostic dimensionality reduction paradigm of spatio-temporal data," arXiv:2204.03216 (2022).
[43]H. Eivazi, H. Veisi, M. H. Naderi, and V. Esfahanian, "Deep neural networks for nonlinear model order reduction of unsteady flows," Phys. Fluids **32**, 105104 (2020).
[44]N. Takeishi, Y. Kawahara, and T. Yairi, "Learning Koopman invariant subspaces for dynamic mode decomposition," in *Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS'17* (Curran Associates, Inc., Red Hook, NY, 2017), pp. 1130–1140.
[45]C. Y. Li, Z. Chen, X. Lin, A. U. Weerasuriya, X. Zhang, Y. Fu, and T. K. Tse, "The linear-time-invariance notion to the Koopman analysis: The architecture, pedagogical rendering, and fluid–structure association," Phys. Fluids **34**, 125136 (2022).
[46]S. L. Brunton, J. L. Proctor, and J. N. Kutz, "Discovering governing equations from data by sparse identification of nonlinear dynamical systems," Proc. Natl. Acad. Sci. **113**, 3932–3937 (2016).
[47]P. Wu, J. Sun, X. Chang, W. Zhang, R. Arcucci, Y. Guo, and C. C. Pain, "Data-driven reduced order model with temporal convolutional neural network," Comput. Methods Appl. Mech. Eng. **360**, 112766 (2020).
[48]O. San, R. Maulik, and M. Ahmed, "An artificial neural network framework for reduced order modeling of transient flows," Commun. Nonlinear Sci. Numer. Simul. **77**, 271–287 (2019).
[49]H. F. Lui and W. R. Wolf, "Construction of reduced-order models for fluid flows using deep feedforward neural networks," J. Fluid Mech. **872**, 963–994 (2019).
[50]S. Pawar, S. E. Ahmed, O. San, and A. Rasheed, "Data-driven recovery of hidden physics in reduced order modeling of fluid flows," Phys. Fluids **32**, 036602 (2020).
[51]M. Wang, S. W. Cheung, W. T. Leung, E. T. Chung, Y. Efendiev, and M. Wheeler, "Reduced-order deep learning for flow dynamics. The interplay between deep learning and model reduction," J. Comput. Phys. **401**, 108939 (2020).
[52]J. Morton, A. Jameson, M. J. Kochenderfer, and F. Witherden, "Deep dynamical modeling and control of unsteady fluid flows," in *Advances in Neural Information Processing Systems*, edited by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Curran Associates, Inc., 2018), Vol. 31.
[53]S. Bhatnagar, Y. Afshar, S. Pan, K. Duraisamy, and S. Kaushik, "Prediction of aerodynamic flow fields using convolutional neural networks," Comput. Mech. **64**, 525–545 (2019).
[54]A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. U. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, edited by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Curran Associates, Inc., 2017), Vol. 30.
[55]D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," arXiv:1409.0473 (2014).
[56]J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," arXiv:1810.04805 (2018).
[57]T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems* (Curran Associates, Inc., 2020), Vol. 33, pp. 1877–1901.
[58]A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," in *International Conference on Learning Representations* (2021).
[59]K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu *et al.*, "A survey on vision transformer," IEEE Trans. Pattern Anal. Mach. Intell. **45**, 87–110 (2023).
[60]K. Tunyasuvunakool, J. Adler, Z. Wu, T. Green, M. Zielinski, A. Zídek, A. Bridgland, A. Cowie, C. Meyer, A. Laydon, S. Velankar, G. Kleywegt, A.

Bateman, R. Evans, A. Pritzel, M. Figurnov, O. Ronneberger, R. Bates, S. Kohl, and D. Hassabis, "Highly accurate protein structure prediction for the human proteome," Nature **596**, 590–599 (2021).
[61]S. Cao, "Choose a transformer: Fourier or Galerkin," in *Advances in Neural Information Processing Systems*, edited by A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan (MIT Press, 2021).
[62]G. Kissas, J. H. Seidman, L. F. Guilhoto, V. M. Preciado, G. J. Pappas, and P. Perdikaris, "Learning operators with coupled attention," arXiv:2201.01032 (2022).
[63]Y. Shao, C. C. Loy, and B. Dai, "SiT: Simulation transformer for particle-based physics simulation," in The International Conference on Learning Representations, 2022.
[64]Z. Li, K. Meidani, and A. B. Farimani, "Transformer for partial differential equations' operator learning," arXiv:2205.13671 (2022).
[65]N. Geneva and N. Zabaras, "Transformers for modeling physical systems," Neural Networks **146**, 272–289 (2022).
[66]X. Han, H. Gao, T. Pffaf, J.-X. Wang, and L.-P. Liu, "Predicting physics in mesh-reduced space with temporal attention," arXiv:2201.09113 (2022).
[67]R. R. Torrado, P. C. T. Ruiz, L. Cueto-Felgueroso, M. C. Green, T. Friesen, S. F. Matringe, and J. Togelius, "Physics-informed attention-based neural network for solving non-linear partial differential equations," arXiv:2105.07898 (2021).
[68]P. Ramachandran, N. Parmar, A. Vaswani, I. Bello, A. Levskaya, and J. Shlens, "Stand-alone self-attention in vision models," in *Advances in Neural Information Processing Systems* (MIT Press, 2019), Vol. 32.
[69]H. Zhao, J. Jia, and V. Koltun, "Exploring self-attention for image recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (IEEE, 2020), pp. 10076–10085.
[70]D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv:1412.6980 (2014).
[71]A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, edited by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Curran Associates, Inc., 2019), Vol. 32, pp. 8024–8035.
[72]O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Proceedings on the 18th International Conference on Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015* (Springer, 2015), pp. 234–241.
[73]R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu, "Towards physics-informed deep learning for turbulent flow prediction," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, KDD '20* (Association for Computing Machinery, New York, NY, 2020), pp. 1457–1466.
[74]K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (IEEE, 2015).
[75]A. Hemmasian (2023). "Reduced-order modeling of fluid Flows with Transformers," https://github.com/BaratiLab/ROMER