Emulating Spatio-Temporal Realizations of Three-Dimensional Isotropic Turbulence via Deep Sequence Learning Models

Mohammadreza Momenifar[1], Enmao Diao[2], Vahid Tarokh[2], Andrew D. Bragg[1],

[1]Department of Civil and Environmental Engineering, [2]Department of Electrical and Computer Engineering
Duke University
Durham, NC, 27701, USA
mohammadreza.momenifar@duke.edu, enmao.diao@duke.edu, vahid.tarokh@duke.edu, andrew.bragg@duke.edu

Abstract

We use a data-driven approach to model a three-dimensional turbulent flow using cutting-edge Deep Learning techniques. The deep learning framework incorporates physical constraints on the flow, such as preserving incompressibility and global statistical invariants of velocity gradient tensor. The accuracy of the model is assessed using statistical and physics-based metrics. The data set comes from Direct Numerical Simulation of an incompressible, statistically stationary, isotropic turbulent flow in a cubic box. Since the size of the dataset is memory intensive, we first generate a low-dimensional representation of the velocity data, and then pass it to a sequence prediction network that learns the spatial and temporal correlations of the underlying data. The dimensionality reduction is performed via extraction using Vector-Quantized Autoencoder (VQ-AE), which learns the discrete latent variables. For the sequence forecasting, the idea of Transformer architecture from natural language processing is used, and its performance compared against more standard Recurrent Networks (such as Convolutional LSTM). These architectures are designed and trained to perform a sequence to sequence multi-class classification task in which they take an input sequence with a fixed length (k) and predict a sequence with a fixed length (p), representing the future time instants of the flow. Our results for the short-term predictions show that the accuracy of results for both models deteriorates across predicted snapshots due to autoregressive nature of the predictions. Based on our diagnostics tests, the trained Conv-Transformer model outperforms the Conv-LSTM one and can accurately, both quantitatively and qualitatively, retain the large scales and capture well the inertial scales of flow but fails at recovering the small and intermittent fluid motions.

Introduction

Turbulence is a complex dynamical system which is strongly high-dimensional, multi-scale, non-linear, non-local and chaotic with a broad range of correlated scales that vary over space and time. Such features make high-fidelity spatio-temporal simulation of turbulence extremely challenging and often impossible (particularly for large domains, unsteady flows, complex boundary conditions) due to limitations of computational power and numerical schemes. From Computational Fluid Dynamics (CFD) perspective, turbulent flows can be simulated by solving Navier-Stokes (NS) equations in three flavors of Direct Numerical Simulation (DNS), Large Eddy Simulation (LES) and Reynolds Averaged Navier-Stokes Simulation (RANS) depending on the application and expected accuracy. In DNS, full-scale resolution is achieved by solving NS on a domain with extremely fine spatial and temporal grids so that all the scales down to the dissipation range are resolved. The LES is based on low-pass spatial filtering of NS equations in which the small scales of turbulent flow are modeled and large unsteady motions, corresponding to the most energetic scales, are resolved. The main goal of modeling in LES is to build a more accurate and universal closure model for the residual (sub-grid) stress tensor. RANS models are derived by taking a temporal average of NS equations, resulting in simpler steady equations, and assuming a linear relationship between Reynolds stresses and mean strain rate. In RANS modeling, the focus is on constructing the Reynolds-stress tensor closure.

The aforementioned CFD techniques are purely physics based where NS equations are solved using different differentiation and integration schemes. However, the high-fidelity simulations using DNS and LES are computationally prohibitive and are limited to simplified turbulent flows to gain detailed insight into the physics of turbulence. Furthermore, RANS cannot simulate important characteristics of turbulence such as unsteadiness and intermittency due to its time averaging nature. Therefore, a major pursuit and fundamental task of turbulence community is to develop new modeling techniques to characterize dynamics of turbulence that evolve over time and space while varying over a broad range of spatio-temporal scales.

Recently with growing the availability of high-fidelity data and computational power data-driven modeling has gained huge interest and been introduced as a competitive alternative to conventional numerical simulations. Among these data-driven methods, Machine Learning (ML) techniques, particularly Deep Learning (DL) models, have received more attention due to their ability to capture complex interactions and achieve outstanding performance across a wide range of applications in information technology, healthcare and engineering to name a few.

In this research our main objective is to bring fresh

perspective to the classic problem of turbulence modeling by exploring and utilizing modern deep learning techniques to develop accurate and efficient turbulence models. More specifically, the primary goal is to develop a data-driven modeling framework to simulate high-fidelity three-dimensional turbulent flow realizations that respect turbulence characteristics without solving the full NS equations.

Background

In fluid dynamics community, particularly computational fluid dynamics of turbulent flows, there have been some attempts in recent years to utilize deep learning tools into turbulence modeling and generate predictive models. These endeavors have been mainly centered around developing Reynolds stress closures and subgrid-scale (SGS) models for RANS and LES simulations respectively (Maulik et al. 2019; Ling, Kurzawski, and Templeton 2016; Beck, Flad, and Munz 2019), super-resolution reconstruction (enhancing the resolution) of coarse flow fields (Fukami, Fukagata, and Taira 2018), turbulence data compression (Glaws, King, and Sprague 2020), and augmenting existing turbulence models with physics-informed machine learning (Wu, Xiao, and Paterson 2018; Wang and Raj 2017).

The first step of modeling dynamics of turbulence is to identify potential deep learning models which are well-suited for handling non-linear spatial and sequential data. Indeed, we seek models that can capture spatial and temporal dependencies of turbulent flow field. Given the grid/pixel-based discretizations of our computational domain, we can draw an analogy between turbulence data and image data where three components of velocity field represent RGB (red, green, blue) color channels and instead of two dimensional spatial grids we have three-dimensional ones. Therefore, we can benefit from modern computer vision approaches in our modeling framework, particularly Convolutional Neural Networks (CNN) (Khan et al. 2018).

The capability of CNNs in extracting spatial distribution of data (exploiting the correlations between the adjacent input data points) could make them a suitable architecture for a variety of physics-based applications. In the past few years there have been multiple studies that utilized CNNs in the context of spatio-temporal modeling of turbulence (Wang et al. 2020; Li et al. 2020) and found promising results. However, these works have been limited to 2D turbulent flows which are far less complicated than 3D turbulence both in terms of physics and computational tractability. The only relevant works aligned with our objective are the recent studies of (Mohan et al. 2019, 2020; Mohan, Livescu, and Chertkov 2020) that their findings, though encouraging but seem suspicious.

In (Mohan et al. 2020), the dynamic mapping of ScalarHIT dataset, containing three components of velocity field of isotropic turbulence and two passive scalar advected with the flow, was modelled via a integration of Convolutional Autoencoder neural network (CAE) and convolutional LSTM, so called Compressed Convolutional LSTM (CCLSTM). In this architecture, flow snapshots are first transformed to a low-dimensional latent-space, using a pre-trained CAE, serving as input sequence of the sequence learning model. This LSTM model is trained to predict an output sequence representing the future snapshots of flow in the latent space and then this output sequence is transformed to original dimension of flow field. This CC-LSTM model is autoregressive (cycling prediction), meaning that previous predictions are fed as the input for the next predictions to generate future temporal realizations. Their CAE has a compression ratio of 125 and the temporal spacing between sampled snapshots, or sampling rate, was $\omega = 0.09\,\tau$, where τ represents eddy turnover time.

The static reconstruction results show that their CAE model has retained large and inertial scales but fails drastically at capturing small scales, as demonstrated in the tails of the PDF of velocity gradient and large wave numbers in energy spectra (Figure 10 in (Mohan et al. 2020)). For the dynamic mapping, they predicted several time instants over range $\tau = [3.4.5]$. Their (dynamic) results (Figure 16 in (Mohan et al. 2020)) show that their model performs even better than the static reconstructions and can fully capture large and inertial scales of flow and generate stable flow realizations over this $1.5\ \tau$ prediction horizon. These results are really suspicious as we observe that the tails of PDF of velocity gradient, which their compression model failed to retain, are almost perfectly recovered and remained stable during dynamic prediction. The authors conjectured that their model adds some artifacts to the predicted snapshots that result in mimicking these intermittent regions of the PDFs. More importantly, their claim (and results) that the error of their temporal predictions remains marginal with increasing prediction horizon is in clear contrast with the findings reported in the recent spatio-temporal modeling of two-dimensional turbulence(Wang et al. 2020).

The same authors in a recent work (Mohan, Livescu, and Chertkov 2020), proposed another spatio-temporal model, called Wavelet-CLSTM, in which the compression is performed via wavelet transformation as opposed to CAE. Their results again indicate that such a model can emulate temporal evolution of flow field such that statistics of large and intermediate scales are retained stable with increasing prediction horizon. While it was clear that their model has large discrepancies in recovering the PDF of velocity gradient, they reported that these PDFs solely test the smallest scales of flow and so such deviations are expected since the loss of information (during wavelet transform) happens at small scales. However, from turbulence literature we know that PDFs of velocity gradient tensor also provide rich multi-scale information and they do not only represent small scales of flow.

Method

In this research, we leverage existing high-fidelity DNS data to emulate spatio-temporal evolution of three-dimensional turbulence with a less memory and computation costs compared to existing flow solvers. More importantly, we make this framework physics-informed through embedding a calibration inside model by infusing as much as our prior knowledge of data and turbulence as possible into training process. Hierarchical superposition of complex structures in deep convolutional layers resemble cascade nature of turbulence

and hence it is worth exploring whether the techniques utilized in computer vision tasks could be applicable to turbulent flow field. Therefore, the convolutional layers would be the core part of the deep learning architecture of this study. We probe and validate this framework by employing high-fidelity DNS data from three-dimensional statistically stationary isotropic turbulence.

The high-fidelity DNS data come from solving the NS equations on a fine-grid three-dimensional mesh. In this study, we start with our smallest dataset which represents a fully resolved statistically stationary isotropic turbulent flow with $R_\lambda = 90$ simulated on a cubic domain with 128 grid point in each direction. At each of these grid points, we have three components of velocity field (u, v, w) and nine components of velocity gradient tensor defined as $A_{ij} = \nabla \boldsymbol{u} = \partial u_i / \partial x_j$, in which u_i is a component of the fluid velocity, and x_i is a spatial coordinate.

Since the size of the dataset is memory intensive, similar to the conceptual design of (Mohan et al. 2019), we first generate a low-dimensional representation of the velocity data and then pass it to a sequence prediction network that learns the positional and temporal correlations of the underlying data. Therefore, our framework will be composed of two separate models where one serves as a compression engine and the other performs prediction. In the deployment phase, once both engines are trained and we want to test the performance of our framework, a sequence of input data is first compressed via the down-sampling part of the compression network, then the prediction network takes this collection of reduced representations and outputs a sequence of future time steps in the latent space and finally this sequence is mapped to original size representation via the upsampling part of the compression engine.

Vector-Quantized Autoencoder (VQ-AE) The dimensionality reduction is performed via extraction using a variant of autoencoder (AE) network, called Vector-Quantized Autoencoder (VQ-AE) architecture (Momenifar et al. 2021; Van Den Oord, Vinyals et al. 2017; Razavi, van den Oord, and Vinyals 2019), which encodes the input data in a discrete latent space and can effectively use the capacity of latent space by conserving important features of data that usually span many dimensions in data space (such as objects in images) and reducing entropy (putting less focus on noise) (Van Den Oord, Vinyals et al. 2017). Mathematically speaking, in a vector quantization (VQ) operation, m-dimensional vectors in $\mathbb{R}^m$ are mapped into a finite set of codewords/vectors $\mathcal{Y} = \{e_i : i = 1, 2, .., K\}$ with a fixed size D or $e_i \in \mathbb{R}^D$, where K represents the size of the codebook. Compared to a conventional autoencoder, a Vector-Quantized Autoencoder has an additional Vector-Quantizer module. The encoder (E) serves as a non-linear function that maps input data (x) to a vector $E(x)$. The quantizer modules takes this vector and outputs an index (k) corresponding to the closest codeword in the codebook to this vector (e_k):

$$\text{Quantize}(E(x)) = e_k,\ k = \arg\min_j \parallel E(x) - e_j \parallel_2 . \quad (1)$$

Codeword index k is used for the integer representation of the latent space, and e_k serves as the input of decoder (D) which operates as another non-linear function to reconstruct the input data. The Vector-Quantizer module brings two additional terms in the loss function, namely codebook loss and commitment loss, to align the encoder output with the vector space of the codebook. The entire VQ-AE loss is defined as:

$$\mathcal{L}(x, D(e)) = \underbrace{\parallel x - D(e) \parallel_2^2}_{reconstruction\ loss} + \underbrace{\parallel sg\{E(x)\} - e \parallel_2^2}_{codebook\ loss} + \underbrace{\beta \parallel sg\{e\} - E(x) \parallel_2^2}_{commitment\ loss}. \quad (2)$$

As noted earlier, preserving small-scale properties of the turbulent flow was a challenge for prior compression models. Here we add appropriate constraints in order to capture these more faithfully. More details on the properties of these physics-based constraints can be found in (Momenifar et al. 2021). By adding these constraints as regularization terms to the VQ-AE loss function gives the overall loss function (OL) given below

$$\text{Overall Loss (OL)} = \text{VQ-AE loss} + \alpha \times \text{VGC} + \gamma \times \text{OC} \quad (3)$$

$$\text{Velocity Gradient Constraint (VGC)} = \underbrace{MSE(A_{ij}, \widehat{A_{ij}})}_{i=j} + a \times \underbrace{MSE(A_{ij}, \widehat{A_{ij}})}_{i \neq j} \quad (4)$$

$$\begin{aligned}\text{Other Constraints (OC)} = {} & MAE(\langle S_{ij} S_{ij} \rangle, \langle \widehat{S_{ij} S_{ij}} \rangle) + \\ & MAE(\langle R_{ij} R_{ij} \rangle, \langle \widehat{R_{ij} R_{ij}} \rangle) + \\ & MAE(\langle S_{ik} S_{kj} S_{ij} \rangle, \langle \widehat{S_{ik} S_{kj}} S_{ij} \rangle) + \\ & MAE(\langle S_{ij} \omega_i \omega_j \rangle, \langle \widehat{S_{ij} \omega_i \omega_j} \rangle),\end{aligned} \quad (5)$$

Convolutional LSTM Vanilla RNN is known to suffer from gradient vanishing and explosion problems, and it fails to capture long term dependencies among sequential data. Therefore, Long-Short Term Memory (LSTM) is later proposed by (Hochreiter and Schmidhuber 1997) as a solution to these drawbacks by introducing memory state and multiple gating mechanism. As formulated below, LSTM has an internal memory cell c_t to store the long term information. It also requires four gates to control the information flow from input, hidden state and the memory cell. The input gate i_t determines how much information of input and hidden state should be remembered by the memory cell. The forget gate f_t determines how much long term memory should be saved for the next time step. g_t contains the information of current input and previous hidden state and the output gate o_t controls the information flows into the next hidden state. Depending on the switching of gates, LSTM can represent

long-term and short-term dependencies of sequential data simultaneously.

$$i_t = \sigma(W_{xi}x_t + b_{xi} + W_{hi}h_{t-1} + b_{hi}) \quad (6)$$
$$f_t = \sigma(W_{xf}x_t + b_{xf} + W_{hf}h_{t-1} + b_{hf}) \quad (7)$$
$$g_t = \tanh(W_{xg}x_t + b_{xg} + W_{hg}h_{t-1} + b_{hg}) \quad (8)$$
$$o_t = \sigma(W_{xo}x_t + b_{xo} + W_{ho}h_{t-1} + b_{ho}) \quad (9)$$
$$c_t = f_t * c_{t-1} + i_t * g_t \quad (10)$$
$$h_t = o_t * \tanh(c_t) \quad (11)$$

Apart from LSTM mentioned above, there still exist many other variations of RNN. Many of them are designed to specialize on specific type of sequential data (Cho et al. 2014; Jozefowicz, Zaremba, and Sutskever 2015; Tai, Socher, and Manning 2015; Greff et al. 2016; Kent and Salem 2019; Diao, Ding, and Tarokh 2019). Because we focus on three-dimensional turbulence data, we use Convolutional LSTM (Conv-LSTM)(Xingjian et al. 2015) to model the dependence between the spatial and temporal information. Conv-LSTM replace linear transformation in vanilla LSTM with Convolutional Neural Networks (CNN) and its general formulation remains the same.

Convolutional Transformer Transformer (Vaswani et al. 2017) architecture has been introduced with the purpose of offering parallel computation (by avoiding recursion that consequently reduces training time) and reducing performance drop due to long-term dependency issues. The transformer model introduces the self-attention unit, which accounts for similarity scores between elements of a sequence, and positional embeddings, a unit that replaces the recurrence. These innovative units can capture sequential relationship between different items of a sequence and consequently allow the Transformer network to process input sequences as a whole rather than element by element, which is typical in recurrent neural networks. This characteristic, processing all the items in an input sequence simultaneously, enables the Transformer model not to rely on previous hidden states for preserving correlations with previous elements in sequence (no backpropagation through time), hence eliminates the risk of forgetting past information with increasing sequence length.

We formulate the transformer block below. Input x is transformed into query, key, and value with three separate weight matrix W_q, W_k, and W_v respectively. We calculate self-attention α by taking the dot product of the query with the key with a scaling factor $\frac{1}{\sqrt{d_k}}$ where d_k is the size of hidden representations. The self-attention is used to attend the value with a linear transformation parameterized by weight matrix W_u. We use a feed-forward block parameterized by weight matrix W_1 and W_2 to further model the attended output. We standardize temporal information with LayerNorm (Ba, Kiros, and Hinton 2016) function. The residual connection is also used to avoid the issue of gradient vanishing.

$$Q(x) = W_q * x, K(x) = W_k * x, V(x) = W_v * x \quad (12)$$
$$\alpha = \text{softmax}\left(\frac{\langle Q(x), K(x)\rangle}{\sqrt{d_k}}\right) \quad (13)$$
$$u' = W_u * \alpha V(x) \quad (14)$$
$$u = \text{LayerNorm}\ (x + u') \quad (15)$$
$$z' = W_2 * \text{ReLU}(W_1 * u) \quad (16)$$
$$z = \text{LayerNorm}(u + z') \quad (17)$$

This block can be stacked multiple times for better performance. Similar to Conv-LSTM, we replace linear transformation in vanilla Transformer with convolution $*$ and its general formulation remains the same.

Computational Details

As mentioned earlier, our framework consists of two deep learning models, one for compression and the other for sequence learning, which are trained separately. Our compression model is a VQ-AE, proposed in our recent work ((Momenifar et al. 2021)). We design our VQ-AE network so that it can downsample original data by a scaling factor of $SF = 2$. With $K = 512$ representing the size of the codebook and mapping three velocity components into one in the discrete latent space, we can achieve $\frac{3\times32}{1\times9} \times (SF)^3$ reduction in bits, corresponding to 85. Indeed, an input data of shape $(3, 128, 128, 128)$ is compressed to $(1, 64, 64, 64)$. One can find more details regarding hyper-parameters, model architecture and training in (Momenifar et al. 2021).

For the sequence learning model, we designed and trained two radically different sequence learning models, convolutional LSTM (Conv-LSTM) and convolutional Transformer (Conv-Transformer). These architectures are designed and trained to take an input sequence with length k and predict a sequence with length p, representing the next p realizations of the system (sequence to sequence multi-class classification). Mathematically, it means mapping $[X_{t-k}, ..., X_{t-2b}, X_{t-b}, X_t]$ to $[X_{t+b}, X_{t+2b}, ..., X_{t+p}]$, where b is the sampling interval (time span between observations), $X \in \mathbb{R}^{H\times W\times D\times C}$ in which C is the number of channels in the data (corresponding with the number of velocity components), H , W and D represent the height, width and depth of data. Here the architectures used in this study are briefly summarized.

For the both sequence learning models, the input sequence of velocity field is first transformed to a low-dimensional discrete latent space, via the pre-trained encoder of the VQ-AE model, and then passes through an embedding layer to transform data to continuous space (indeed, each integer is represented with a codeword which is a vector in real space). For the Convolutional LSTM model, this continuous latent space sequence is fed to a LSTM block consisting of three LSTM cells/layers. Afterwards the output of LSTM block passes through a series of linear layers to expand the channel dimension from codeword size to the codebook one. Then we need to find, for each grid point in three-dimensional domain, the class (index) with the highest probability in this

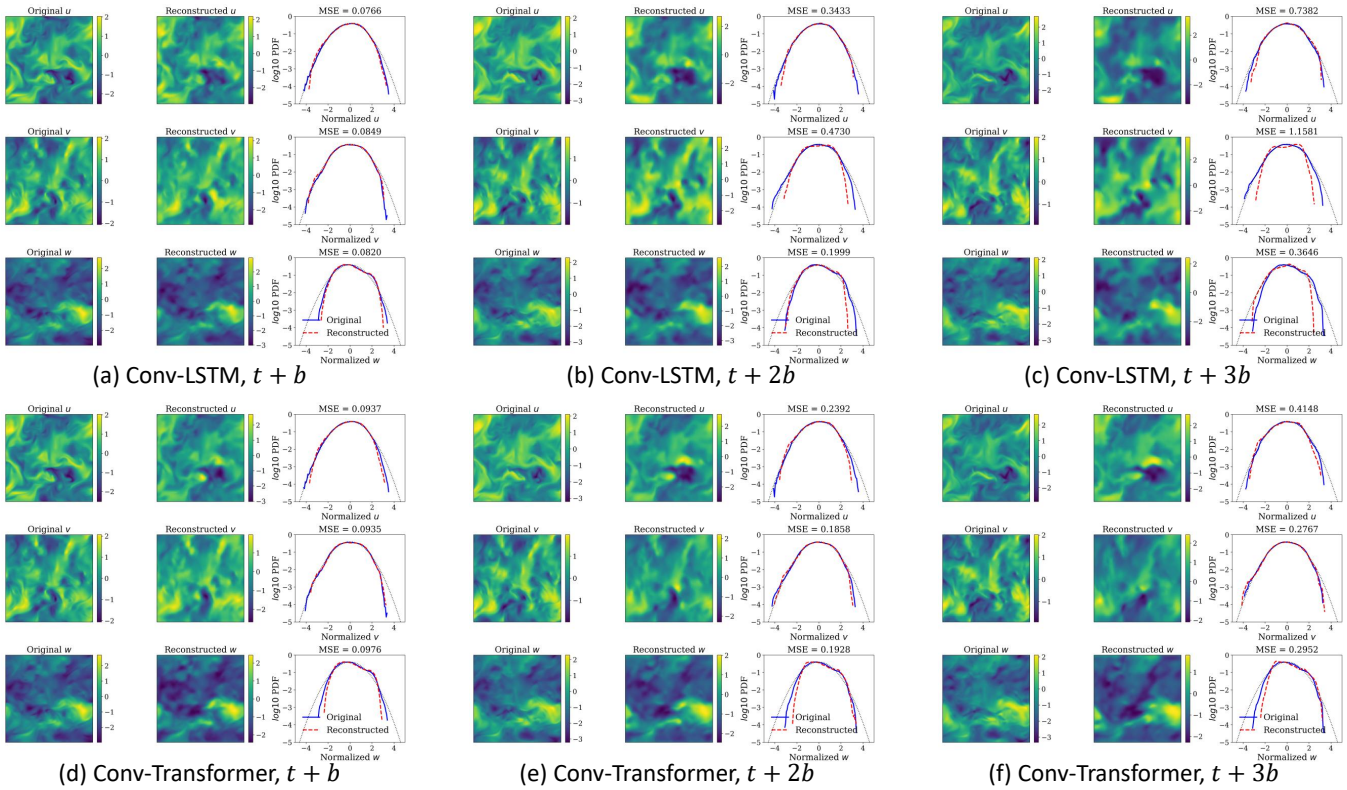

Figure 1: Predictions of velocity field using (a-c) Conv-LSTM and (d-f) Conv-Transformer models.

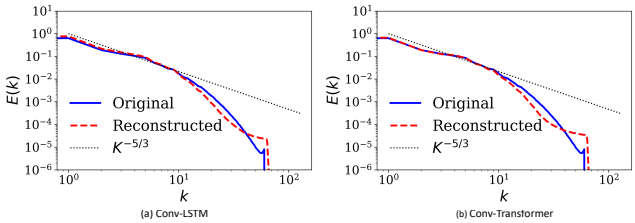

Figure 2: Predictions of turbulent kinetic energy (TKE) using (a-c) Conv-LSTM and (d-f) Conv-Transformer models.

multi-class classification task. This step is performed by returning the index of the largest element in the channel dimension and then we apply the cross-entropy loss to compute the discrepancy between the true and predicted output sequence. Finally, the predicted output sequence is transformed to the original dimension of velocity field through the pre-trained decoder of the VQ-AE model. The same pipeline was used for the Transformer model, where we designed its architecture following the instructions in (Vaswani et al. 2017) and adjusted the network for image data (the Transformer model was originally proposed for text data).

In our dataset we have 80 snapshots equally spaced in time, from $t = 3\,T_L - 7\,T_L$, where T_l denotes large eddy turn over time. Indeed, these snapshots cover a time span of $4T_l$ corresponding to a sampling interval of $b = 0.05\,T_L$. We use the first 40 snapshots as training set and the rest as test set. We trained our framework for 300 epochs with batch size $= 1$ using the Adam optimizer (Kingma and Ba 2014) with learning rate = 0.001, along with a learning rate scheduler. It should be noted that our focus has been on the general characteristics of the framework rather than achieving the best configurations, hence there might be room for improvements by tuning the proposed hyper-parameters.

We implemented this framework in PyTorch using the CUDA environment, and trained it on one NVIDIA Pascal P100 GPU. The performance of the predicted velocity field is assessed not only via the conventional error measurement methods such as MSE and mean absolute error (MAE) but also using rigorous physics-based metrics relevant to the analysis of turbulence such as the probability density functions (PDFs) of the filtered velocity gradient tensor and its invariants, the turbulent kinetic energy spectra and the joint PDF of $Q - R$ plane (detailed in (Momenifar et al. 2021)).

Experiments

Our VQ-AE model offers a compression ratio of 85 and recovers all the flow characteristics up to second order statistics of the velocity gradient tensor, with small discrepancies at the smallest scales. We also found that the embedding physics constraints in the loss function can noticeably improve the quality of the reconstructed small scales of the flow. Interested readers can found details on our data compression engine in (Momenifar et al. 2021).

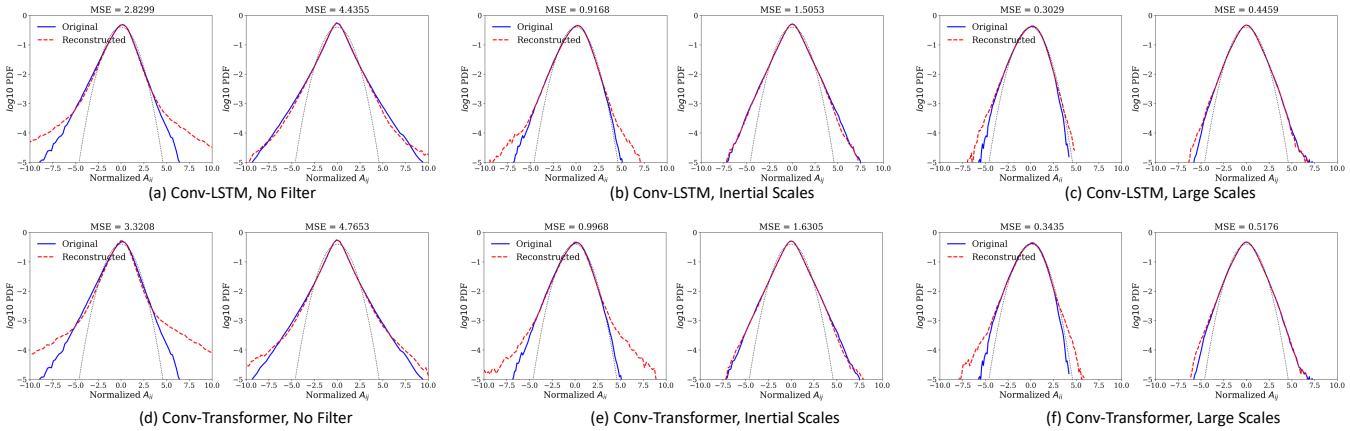

Figure 3: Predictions PDFs of velocity gradient tensor using (a-c) Conv-LSTM and (d-f) Conv-Transformer models on small (a,d), inertial (b,e), and large (d,f) scales.

Table 1: Simulation parameters for the DNS study of isotropic turbulence (arbitrary units). N is the number of grid points in each direction, $Re_\lambda \equiv u'\lambda/\nu$ is the Taylor micro-scale Reynolds number, λ is the Taylor micro-scale, $\mathscr{L}$ is the box size, ν is the fluid kinematic viscosity, ϵ is the mean turbulent kinetic energy dissipation rate, l is the integral length scale, $\eta \equiv \nu^{3/4}/\epsilon^{1/4}$ is the Kolmogorov length scale, $u' \equiv \sqrt{(2k/3)}$ is the fluid r.m.s. fluctuating velocity, k is the turbulent kinetic energy, u_η is the Kolmogorov velocity scale, $T_L \equiv l/u'$ is the large-eddy turnover time, $\tau_\eta \equiv \sqrt{(\nu/\epsilon)}$ is the Kolmogorov time scale, $\kappa_{\max} = \sqrt{2}N/3$ is the maximum resolved wavenumber.

Parameter	N	Re_λ	$\mathscr{L}$	ν	ϵ	l	l/η	u'	u'/u_η	T_L	T_L/τ_η	$\kappa_{\max}\eta$
Value	128	93	2π	0.005	0.324	1.48	59.6	0.984	4.91	1.51	12.14	1.5

As mentioned earlier, our sequence learning models have been trained to take an input sequence with a fixed length (k) and predict a sequence with a fixed length (p), representing the future time instants of the flow. The spacing between the flow snapshots or sampling interval (b), is arbitrary but is constrained with the input sequence. Therefore, feeding an input sequence where all the flow realizations are $0.05T_L$ apart would result in an output sequence with a similar sampling interval $b = 0.05T_L$. In our experiments, we trained our models with sequences where $\omega = 0.05T_L$ and therefore they are well-suited to predict temporal realizations that are $0.05T_L$ apart. Furthermore, we can also roll out autoregressively (also known as cyclic prediction) and feed predicted sequence as input sequence to the model and generate flow realizations over a larger prediction horizon.

Given the size of the compressed data in discrete latent space, $(1, 64, 64, 64)$, and our available GPU memory, we were able to train sequence learning models with $k = p = 3b$. It is worth mentioning that one can train models with much larger k, p with a more compressed data , corresponding to a larger SF. However, this would come at the cost of losing prediction accuracy as the information content of compressed data with $SF = 4$ is less richer that $SF = 2$. In what follows, we present our results during inference for short-term prediction, where the sampling interval in the test data is the same as the training data ($b = 0.05T_L$).

During short-term predictions the models receive input sequence of $[X_{t-2b}, X_{t-b}, X_t]$ and generate sequence of $[\widehat{X_{t+b}}, \widehat{X_{t+2b}}, \widehat{X_{t+3b}}]$, where $\widehat{X}$ represents prediction of X. It is worth pointing out that we obtained robust results over all test data and the following results are for the test case where $t = 6.8T_L$. In all of the diagnostics tests, we observe that the accuracy of results deteriorates from the first to the third predicted snapshots. This is quite expected as the error propagates from the first predictions to the next ones due to autoregressive nature of the predictions. Indeed, during the inference we obtain $\widehat{X_{t+b}}$, $\widehat{X_{t+2b}}$, and $\widehat{X_{t+3b}}$ based on $[X_{t-2b}, X_{t-b}, X_t]$, $[X_{t-b}, X_t, \widehat{X_{t+b}}]$, and $[X_t, \widehat{X_{t+b}}, \widehat{X_{t+2b}}]$, respectively.

In Figure 1, we evaluate the performance of our Conv-LSTM and Conv-Transformer models in reconstructing 2d snapshots (randomly sampled) of the velocity field, as well as the PDFs of the velocity components (where the statistics are based on the full 3d domain) across the predicted snapshots. The results show that for the first prediction time step both models capture reasonably well, Conv-LSTM is slightly better than Conv-Transformer, the instantaneous spatial structure of the flow and the statistical properties of the velocity field. Although the accuracy of reconstructed snapshots decreases for the next predictions due to the error propagation, we clearly observe that the quality of predicted snapshots using Conv-Transformer is much better.

The turbulent kinetic energy (TKE) spectra of the predicted time instants are shown in Figure 2 for the Conv-LSTM and Conv-Transformer models. While the reconstruction quality decreases for the Conv-LSTM model from the first to the third predictions, our Conv-Transformer model accurately captures the large and inertial scales of flow, both quantitatively and qualitatively, with significant loss of information for the smallest scales.

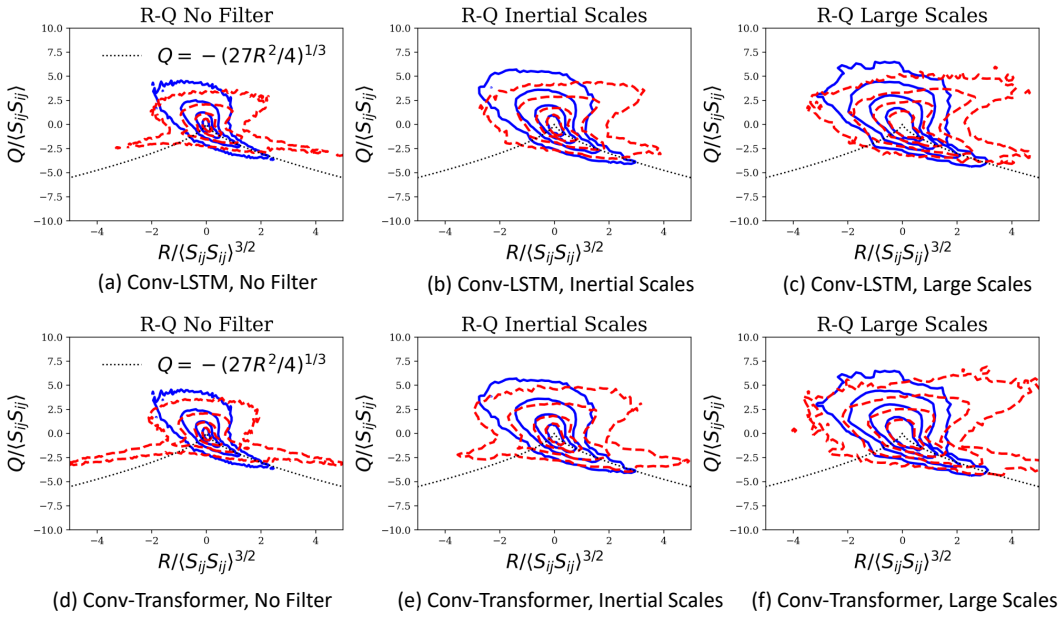

Figure 4: Predictions of R-Q using (a-c) Conv-LSTM and (d-f) Conv-Transformer models on small (a,d), inertial (b,e), and large (d,f) scales.

The PDFs of the longitudinal and transverse components of the velocity gradient tensor for different filtering lengths are shown in Figure 3 across different predictions of the Conv-LSTM and Conv-Transformer models. The results illustrate that our models across all the predicted time instants can accurately capture the body of these PDFs, but fail at retaining heavy tails and skewness. Our results show that the Conv-Transformer model has a better performance than Conv-LSTM one. Furthermore, we observe that the quality of the results improves as we move from inertial to large scales, indicating that loss of information mainly occurred at small scales.

The capability of our models in reconstructing the joint-PDF of the Q and R invariants of velocity gradient tensor across different predictions and flow scales are shown in Figure 4 for the Conv-LSTM and Conv-Transformer models. Our models struggle to capture the behavior of the Q, R PDFs and can only capture some of the most frequent characteristics of the flow (interior contours of the PDF). Such large discrepancies in recovering these joint PDFs may seem surprising given that our models were able to accurately capture the body of the PDFs of velocity gradient tensor. However, the invariants Q, R depend not only upon the properties of the individual velocity gradient components, but also upon more subtle features such as the geometric alignments between the strain-rate and vorticity fields which our models fail to capture from the data.

In summary, we can conclude that the Conv-Transformer model outperforms the Conv-LSTM model and can better retain statistics of flow field across the entire prediction horizon. The outperformance of Transformer model can be heavily attributed to its ability to process input sequences as a whole rather than element by element which is typical in LSTM model. Such a characteristic enables Transformer to capture more faithfully the temporal correlations between sequence elements.

we also tried to test our models, which have been trained for short-term prediction, on challenging tasks of varying sampling interval and long-term predictions. Not shown here, but our results indicate that the quality of reconstructed flow fields across all the predicted snapshots deceases for the varying sampling interval inference. Furthermore, we observed that during cyclic prediction (to generate long-term predictions), error (from one prediction to the next one) propagates significantly and accumulation of such errors repeatedly result in flow snapshots that do not retain important characteristics of turbulence.

Conclusions

In this study, we aim at exploring the feasibility of emulating temporal and spatial patterns of three-dimensional isotropic turbulence, purely from data, via modern deep learning approaches. Since our data size is memory intensive, we first generate a low-dimensional representation of the velocity data and then pass it to a sequence prediction network that learns the spatio-temporal correlations of the underlying data. Our results show that the accuracy of results for both models deteriorates across predicted snapshots due to autoregressive nature of the predictions. One potential idea to improve the paper is by including explicit multi-resolution structure in the latent code generation (Jia et al. 2019).

Based on our diagnostics tests, the Conv-Transformer model outperforms the Conv-LSTM one and can accurately, both quantitatively and qualitatively, retain the large scales, capture well the inertial scales of flow but fail at recovering the small and intermittent fluid motions. The outperfor-

mance of Conv-Transformer model can be heavily attributed to its ability to process input sequences as a whole rather than element by element (which is typical in LSTM model). However, both models neglect important characteristics of the joint-PDF of the Q and R invariants of velocity gradient tensor indicating that our models struggle to capture more subtle features of velocity field such as the geometric alignments between the strain-rate and vorticity fields.

Acknowledgements

This work was supported by National Science Foundation (NSF) under Grant No. ACI-1548562 and the Office of Naval Research (ONR) under Grant No. N00014-18-1-2244.

References

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

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

Cho, K.; Van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Diao, E.; Ding, J.; and Tarokh, V. 2019. Restricted recurrent neural networks. In *2019 IEEE International Conference on Big Data (Big Data)*, 56–63. IEEE.

Fukami, K.; Fukagata, K.; and Taira, K. 2018. Super-resolution reconstruction of turbulent flows with machine learning. *arXiv preprint arXiv:1811.11328*.

Glaws, A.; King, R.; and Sprague, M. 2020. Deep learning for in situ data compression of large turbulent flow simulations. *Physical Review Fluids*, 5(11): 114602.

Greff, K.; Srivastava, R. K.; Koutník, J.; Steunebrink, B. R.; and Schmidhuber, J. 2016. LSTM: A search space odyssey. *IEEE transactions on neural networks and learning systems*, 28(10): 2222–2232.

Hochreiter, S.; and Schmidhuber, J. 1997. Long short-term memory. *Neural computation*, 9(8): 1735–1780.

Jia, X.; Wang, M.; Khandelwal, A.; Karpatne, A.; and Kumar, V. 2019. Recurrent Generative Networks for Multi-Resolution Satellite Data: An Application in Cropland Monitoring. In *IJCAI*.

Jozefowicz, R.; Zaremba, W.; and Sutskever, I. 2015. An empirical exploration of recurrent network architectures. In *International Conference on Machine Learning (ICML)*, 2342–2350.

Kent, D.; and Salem, F. M. 2019. Performance of Three Slim Variants of The Long Short-Term Memory (LSTM) Layer. *arXiv preprint arXiv:1901.00525*.

Khan, S.; Rahmani, H.; Shah, S. A. A.; and Bennamoun, M. 2018. A guide to convolutional neural networks for computer vision. *Synthesis Lectures on Computer Vision*, 8(1): 1–207.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Li, Z.; Kovachki, N.; Azizzadenesheli, K.; Liu, B.; Bhattacharya, K.; Stuart, A.; and Anandkumar, A. 2020. Fourier Neural Operator for Parametric Partial Differential Equations. *arXiv preprint arXiv:2010.08895*.

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

Maulik, R.; San, O.; Rasheed, A.; and Vedula, P. 2019. Subgrid modelling for two-dimensional turbulence using neural networks. *Journal of Fluid Mechanics*, 858: 122–144.

Mohan, A.; Daniel, D.; Chertkov, M.; and Livescu, D. 2019. Compressed convolutional LSTM: An efficient deep learning framework to model high fidelity 3D turbulence. *arXiv preprint arXiv:1903.00033*.

Mohan, A. T.; Livescu, D.; and Chertkov, M. 2020. Wavelet-powered neural networks for turbulence. In *ICLR 2020 Workshop on Integration of Deep Neural Models and Differential Equations*.

Mohan, A. T.; Tretiak, D.; Chertkov, M.; and Livescu, D. 2020. Spatio-temporal deep learning models of 3D turbulence with physics informed diagnostics. *Journal of Turbulence*, 21(9-10): 484–524.

Momenifar, M.; Diao, E.; Tarokh, V.; and Bragg, A. D. 2021. Dimension Reduced Turbulent Flow Data From Deep Vector Quantizers. *arXiv preprint arXiv:2103.01074*.

Razavi, A.; van den Oord, A.; and Vinyals, O. 2019. Generating diverse high-fidelity images with vq-vae-2. In *Advances in Neural Information Processing Systems*, 14866–14876.

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*.

Van Den Oord, A.; Vinyals, O.; et al. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, 6306–6315.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in neural information processing systems*, 5998–6008.

Wang, H.; and Raj, B. 2017. On the origin of deep learning. *arXiv preprint arXiv:1702.07800*.

Wang, R.; Kashinath, K.; Mustafa, M.; Albert, A.; and Yu, R. 2020. Towards physics-informed deep learning for turbulent flow prediction. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 1457–1466.

Wu, J.-L.; Xiao, H.; and Paterson, E. 2018. Physics-informed machine learning approach for augmenting turbulence models: A comprehensive framework. *Physical Review Fluids*, 3(7): 074602.

Xingjian, S.; Chen, Z.; Wang, H.; Yeung, D.-Y.; Wong, W.-K.; and Woo, W.-c. 2015. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, 802–810.