Synthetic Turbulent Inflow Generation for Simulations of Turbulent Wall-bounded Flows

Hang Song Ahmed Elnahhas
Department of Mechanical Engineering, Stanford University, Stanford, CA 94305

Abstract

An efficient and realistic synthetic turbulent inflow generation is highly demanded in the simulations of wall bounded turbulent flows. In this work, a generative variational auto-encoder (VAE) architecture is applied to generate the three components of the velocity field of an incompressible turbulent channel flow at a grid resolution of 385×512*. The training data is from direct numerical simulations. The VAE network is designed based on convolutional neural networks, and the computational code is implemented using PyTorch. The results show a qualitative agreement with realistic data. Detailed analysis shows that the generated flow fields are more accurate at large scales although significant improvements can be made.*

1. Introduction

1.1. Background & Motivation

Turbulent wall-bounded flows are the fundamental building blocks of most engineering flows of interest, and are frequently found in external aerodynamics applications, as well as geophysical flows. Direct numerical simulations of the Navier-Stokes equations, which govern fluid flow, allow us to probe the internal dynamics of these flows at unprecedented levels of detail. In the case of boundary layers, due to their spatially-developing nature, we can either start the simulation at the fictitious origin of the flow, or further downstream, given that we have a realistic inflow condition at that location. While the first option is possible, it is beyond current supercomputing capabilities to advance such a simulation to the asymptotic parameter regime, $\mathrm{Re} \to \infty$, where Re is the Reynolds number of the flow, and we are therefore limited to the second option. However, an unrealistic inflow condition placed in the middle of the flow leads to readjustment zones that become increasingly longer the farther downstream that inflow condition is placed. In essence, making the second option also prohibitively expensive in reaching asymptotic parameter regimes [8, 10].

1.2. Problem Statement

If a more realistic turbulent structure can be provided at the inflow boundary, the readjustment regions are expected to be significantly reduced, allowing more of the simulation domain to be used for analysis, extending the possible parameter regime. In an effort to accomplish this task, we propose to use generative computer vision deep learning models to generate this inflow condition.

1.3. Inputs & Outputs

We use turbulent channel flow simulation data to train our generative models [7]. Channel flows are used because they are closely related to turbulent boundary layers but with more available data due to the aforementioned limitations. The inputs to our generative model, a Variational Auto-Encoder (VAE) [6], are cross-sectional planes of the channel flow field. Each plane contains three velocity components, namely the streamwise, wall-normal, and spanwise velocity components, which can be thought of as input channels. The output of our VAE are generated planes of the same three velocity components. Both input and output data sizes are $3 \times 385 \times 512$ for single instance.

2. Related Work

2.1. Baseline Turbulent Inflow Generation

The current traditional approach of generating turbulent inflows for wall-bounded flows is that of Lund et al. [8], where turbulent cross-sectional flow fields downstream of the simulation inlet are recycled and rescaled in their wall-normal extent. This approach is shown by Sillero et al. [10] to work well for low Re numbers but not for larger values due to growing decorrelation length scales of the flow.

2.1.1 Flavors of VAE Models

The original VAE model is introduced by Kingma and Welling [6]. This work illustrates the mechanism of VAE network and the design of the loss function based on the variation lower bound, which is described in section 3. An advancement introduces a hyper-parameter (β) that weighs

the two components of the loss-function in [6], known as the disentangled VAE (or β-VAE), and the results have been found to better decompose the transformations applied to an image [3]. Finally, a VAE model which includes residual parameterizations and spectral regularization is shown to be the current state-of-the-art as of 2020 [11].

2.2. Attempts at Generative Models for Turbulence

More recently, deep neural network models have been emerging as methods of generating turbulent inflows. Examples include an encoder-decoder network [1], and a generative adverserial network (GAN) [2] combined with recurrent neural networks (RNN) for turbulent time dependent turbulent channel flow generation [5].

[1] utilizes a convolutional network encoder and decoder without a variational parameterization, and is mostly concerned with advancing an input plane one time-step into the future. Therefore, a turbulent flow field seed is required. [5] utilizes more modern deep learning architectures but has not tested the output planes as a boundary conditions for spatially developing simulations. The networks of both [1] and [5] are relatively deep. Furthermore, they do not perform their generation at sufficiently large Re of interest. Between the two, the work of [5] is more state-of-the-art.

Our goal is to utilize a different algorithm, namely VAE, to train synthetic turbulence generators that don't require realistic seeds, to be eventually implemented into a running calculation as a boundary condition. Examples where VAEs were used include generative models for the cosmic microwave background radiation field, which is similar to our goal of generating a turbulent inflow [12].

3. Methods

The VAE model maps the generated variable $\boldsymbol{x}$ to a latent random variable $\boldsymbol{z}$. The training process is to learn the joint distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x}, \boldsymbol{z})$ to obtain the maximum likelihood $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. According to Bayes' rule, the data likelihood

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}) p_{\boldsymbol{\theta}}(\boldsymbol{z}) / p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \tag{1}$$

where the posterior distribution $p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})$ is intractable, and an encoder process parametrized by $\boldsymbol{\phi}$ is introduced for an approximated posterior $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$.

3.1. Loss function

The loss function is formulated based on maximizing the logarithmic likelihood, $\log(p_{\boldsymbol{\theta}}(\boldsymbol{x}))$.

$$\log(p_{\boldsymbol{\theta}})(\boldsymbol{x}) = \mathcal{L} + \mathbb{D}_{KL}\left(p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \| q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right) \tag{2}$$

where $\mathbb{D}_{KL}(\cdot\|\cdot)$ denotes the Kullback–Leibler (KL) divergence, and $\mathcal{L}$ is the evidence lower bound [6].

$$\mathcal{L} = \mathbb{E}_{\boldsymbol{z}}\left[\log(p_{\boldsymbol{\theta}}(\boldsymbol{x}|\boldsymbol{z}))\right] - \mathbb{D}_{KL}(q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x}) \| p_{\boldsymbol{\theta}}(\boldsymbol{z})) \tag{3}$$

Due to the intractibility of the second term on the right-hand-side of Eq(2), maximizing $\mathcal{L}$ is applied as an variational method by considering $\mathbb{D}_{KL}\left(p_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x}) \| q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})\right) \succeq 0$.

The loss function contains two parts: the generative loss and latent loss. The former is to minimize the discrepancy between the generated and reference data, which is indicated by the pixel-by-pixel L2 error.

$$L_G = \|\boldsymbol{x} - \widehat{\boldsymbol{x}}\|_2^2 \tag{4}$$

where L_G is the generative loss function, $\boldsymbol{x}$ and $\widehat{\boldsymbol{x}}$ are the reference and generated data respectively. The latter is designed to minimize the difference between the modeled posterior distribution and the true posterior distribution, which is indicated by $\mathbb{D}_{KL}(q_{\phi}(\boldsymbol{z}|\boldsymbol{x}) \| p_{\boldsymbol{\theta}}(\boldsymbol{z}))$ assuming $p_{\boldsymbol{\theta}}(\boldsymbol{z}) \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

$$\widetilde{L}_L = -\frac{1}{2}\sum_{j=1}^{C_L}\left(1 + \log(\sigma_j^2) - \mu_j^2 - \sigma_j^2\right) \tag{5}$$

where σ_j and μ_j are the standard deviation and mean of $q_{\boldsymbol{\phi}}(\boldsymbol{z}|\boldsymbol{x})$.

3.2. Architecture of neural network

The structure of the VAE network used for this study is shown in Fig.1. The encoder network consists 7 convolution layers. The latent size is 512. The mean and log-variance vectors are obtained from by two individual fully connect layers from the output of the encoder network. Data through each layer is activated using the Leaky Rectified Linear Unit (Leaky ReLU) function ($\alpha = 0.01$). The latent vector is obtained via a reparameterize process using a random vector $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$. The final latent vector size is 512. Compared to the input data size $N \times 3 \times 385 \times 512$, where N is batch size, the compression ratio through the encoder is approximately 1155 per each image in the batch. The decoder network consists of a fully connected layer and 5 transposed convolution layers for upsampling and intervened with 5 convolution layers. The dimensions of the deconvolution layers are mirrored from the encoder network structure.

3.3. Enforcing Flow Field Periodicity

Since the input data is periodic in the spanwise direction, the VAE model needed to retain this periodicity while encoding and decoding the data. To do so, circular padding was used and its results are shown in 5.

3.4. Code implementation

All code including data pre-/post-processing, VAE model, utility functions for training and test are implemented by the students. The source code and implementation history are documented on GitHub. The code is implemented using PyTorch as the primary library. Numpy

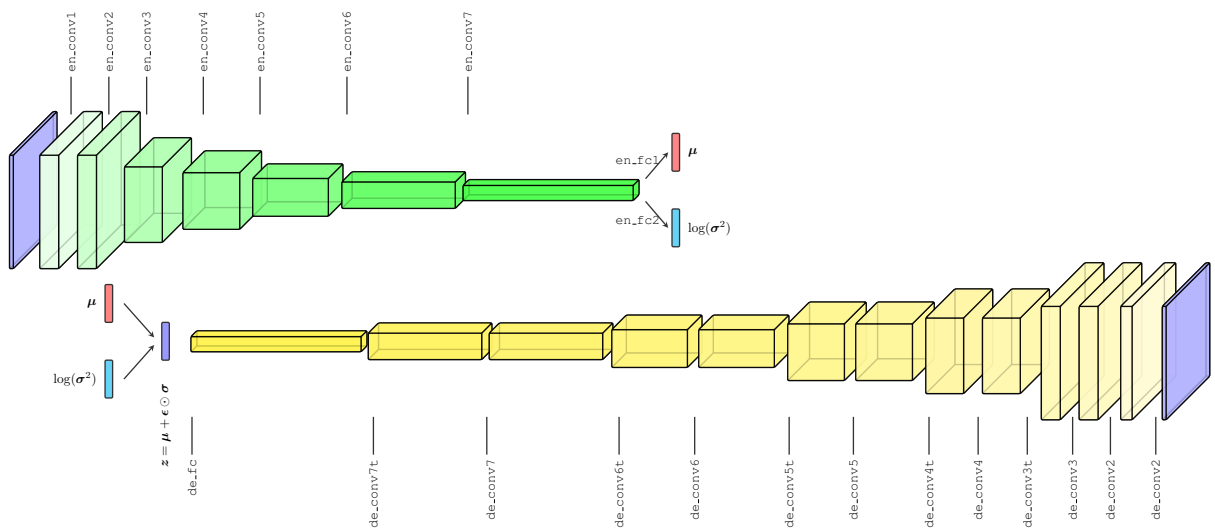


Figure 1. Model architecture

Layer	Module	Filters	Stride	Activation
en_conv1	conv	$8 \times (3 \times 3)$	1	LeakyReLU ($\alpha = 0.01$)
en_conv2	conv	$8 \times (3 \times 3)$	1	LeakyReLU ($\alpha = 0.01$)
en_conv3	conv	$16 \times (3 \times 3)$	2	LeakyReLU ($\alpha = 0.01$)
en_conv4	conv	$32 \times (3 \times 3)$	2	LeakyReLU ($\alpha = 0.01$)
en_conv5	conv	$64 \times (3 \times 3)$	2	LeakyReLU ($\alpha = 0.01$)
en_conv6	conv	$128 \times (3 \times 3)$	2	LeakyReLU ($\alpha = 0.01$)
en_conv7	conv	$256 \times (3 \times 3)$	2	LeakyReLU ($\alpha = 0.01$)
en_fc1	linear	49152×512	-	null
en_fc2	linear	49152×512	-	null
de_fc	linear	512×49152	-	null
de_conv7t	deconv	$128 \times (2 \times 2)$	2	LeakyReLU ($\alpha = 0.01$)
de_conv7	conv	$128 \times (3 \times 3)$	1	LeakyReLU ($\alpha = 0.01$)
de_conv6t	deconv	$64 \times (2 \times 2)$	2	LeakyReLU ($\alpha = 0.01$)
de_conv6	conv	$64 \times (3 \times 3)$	1	LeakyReLU ($\alpha = 0.01$)
de_conv5t	deconv	$32 \times (2 \times 2)$	2	LeakyReLU ($\alpha = 0.01$)
de_conv5	conv	$32 \times (3 \times 3)$	1	LeakyReLU ($\alpha = 0.01$)
de_conv4t	deconv	$16 \times (2 \times 2)$	2	LeakyReLU ($\alpha = 0.01$)
de_conv4	conv	$16 \times (3 \times 3)$	1	LeakyReLU ($\alpha = 0.01$)
de_conv3t	deconv	$16 \times (2 \times 2)$	2	LeakyReLU ($\alpha = 0.01$)
de_conv3	conv	$16 \times (3 \times 3)$	1	Tanhshrink
de_conv2	conv	$8 \times (3 \times 3)$	1	Tanhshrink
de_conv1	conv	$3 \times (3 \times 3)$	1	Instance Row Normalization

Table 1. Architecture of network used in this work.

and Matplotlib are used for data pre-/post-processing and visualization. H5py is used for loading the raw data for preprocessing.

3.5. Training

The final training setup is the following: The optimization is based on Adam algorithm as a built-in optimizer in PyTorch. The learning rate is 0.0003 while other parameters remain default. The model was trained over 12 epochs with the batch size varied between $N \in \{4, 20, 50\}$.

3.6. Qualitative & Quantitative Analysis Methods

To quantitatively determine whether our generated results are capturing the correct behavior of wall-bounded turbulent flows, we analyze the reference and generated flow fields using three metrics. Firstly, we report the joint probability density function between the the streamwise and wall-normal velocities at various wall-normal distances. Secondly, The average cross-correlation between the two velocities is also reported. Thirdly, the energy spectrum, or power spectral density, or the streamwise velocity component along the spanwise direction is quantified. The details of these methods and their importance to turbulent flow analysis are described in [9]. To qualitatively judge the generative model, we visually contrast the flow fields with the reference data.

4. Dataset and Features

4.1. Dataset description

We used existing high-fidelity simulations of turbulent channel flow [7]. This was done because the data is readily available and it is possible to extract multiple examples at the same Re from each snapshot of the simulation due to the homogeneity of the flow in the streamwise direction. In total, 1264 planes were extracted each with three velocity components which we treat as three channels. Only 8 planes were extracted from each snapshot (there are 512 per snapshot) to ensure some spatial decorrelation is present. Similarly, the snapshots extracted were spaced 100 timesteps apart for temporal decorrelation. The channel flow was in equilibrium at $\mathrm{Re}_\tau \approx 950$. Compared to the works of [1] and [5], this is a much larger and therefore harder Re_τ to generate due to the presence of smaller scales. The velocity fields included both mean and fluctuating components.

4.2. Preprocessing

The goal of this VAE model is to predict the spatial structure of the turbulent flow, and not its mean statistics. Therefore, the post-processing of the data involved the removal of the wall-normal dependent mean component. Furthermore, each wall-normal location (each horizontal row of pixels) is normalized by its wall-normal dependent standard deviation. The final images fed into the model network have zero mean and unity standard deviation for each row of the image.

4.3. Examples of the Flow Fields

Examples of the flow fields are shown below in figure 2, without removing and normalizing by the wall-normal mean and standard deviation, respectively. Notice the presence of smaller scales closer to the top and bottom walls, and larger scales towards the center of the channel.

5. Experiments, Results and Discussion

5.1. Hyperparameter Tuning

Once the architecture reported in 3 was finalized, the following hyperparameters were varied. Firstly, the ADAM optimizer was chosen and not changed as it showed successful convergence of both the generative and latent loss of the network after some manipulation of the remaining hyperparameters. Secondly, the learning rate was chosen to be 0.0003 after attempts at a larger learning rate lead to blow-ups during training. Thirdly, the Xavier initialization gain parameter was varied. It was found that a larger gain parameter did not lead to convergence and it was kept at the default value of 1.0. Fourthly, the minibatch N proved to be crucial. While training with $N = 4$, it was observed that the latent loss of the network continued to converge annealment of the learning rate from $0.001 \rightarrow 2 \times 10^{-5}$. However, the generative loss was plateauing. The generated flow fields were blurry. By increasing the the batch size to 20 and then finally 50, the generative loss started to converge. Fifthly, the size of the latent vector was changed from 256 to 512 with better convergence achieved. The final learning set-up did not include learning rate annealing and its details are reported in section 3.5. Finally, tuning of the factor that controls the weighting between the generative loss and latent loss, β, is attempted. Compared to a reference value, $\beta = 0.1$, a larger value of β significantly contributes to the convergence of the latent loss. However, the convergence of the generative loss is not sensitive to the value of β. For this work, the reference value was used.

5.2. Visual Comparison at $\mu = 0$ & $\sigma = I$

The most basic VAE reparameterization trick imposes $p_{\boldsymbol{\theta}}(\boldsymbol{z}) \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$. Figures 2 and 3 visually contrast the reference and generated velocity field for this default parameterization. It is evident that the model generates reasonable large scale turbulent structures in the middle of the domain. However, the generation gets worse closer to the wall when it comes to generating the finer structures. In fact, the failure to generate finer structures is observed everywhere in the domain, but it most problematic near the wall as they are the most dominant features. Another issue is the extreme correlation between the streamwise, wall-normal, and spanwise velocities which is not observed in the reference data. The quantitative accuracy of the generated fields are reported in the subsequent subsections.

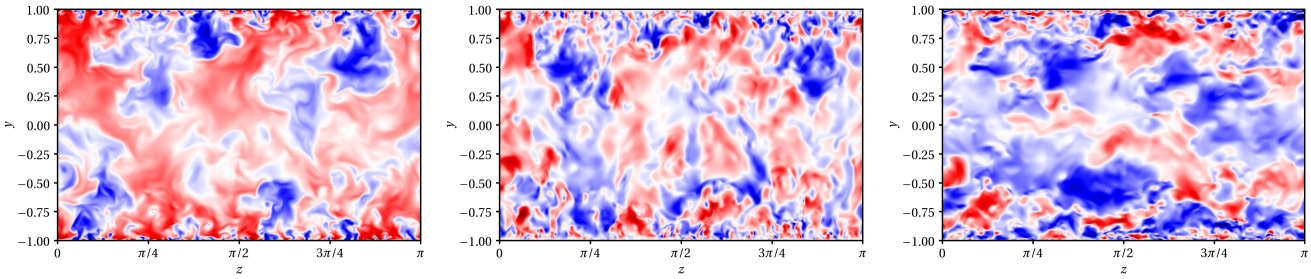
5.3. Enforced Periodicity

One of the key features of a high Re turbulent flow is its power law decaying energy spectra. While there are methods of measuring a representation of these spectra in non-periodic flows, our reference data is periodic, which allows us to use Fourier transforms in its computation. To compare the reference data to the generated one using this metric, we first have to verify that the periodicity of the generated data is enforced through the circular padding employed by the model. Figure 4 shows the same generated velocity field next to a copy of itself translated by its width. It is evident that the periodicity is enforced and we can proceed towards more quantitative analysis.

5.4. Quantitative Analysis - Energy Spectra

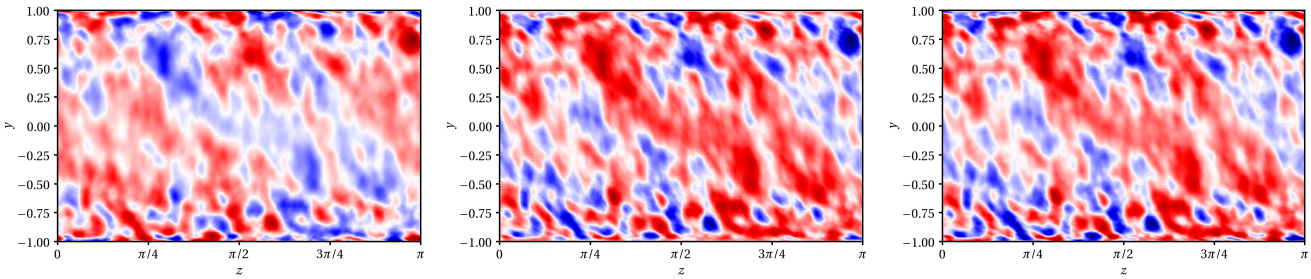
Figure 5 compares the spanwise energy spectrum of the streamwise velocity fluctuation u' as a function of the wall-normal distance normalized by both the channel half-height, y/h, which is a normalization representative of the large scales of the flow, and the wall-normal distance normalized by the viscous length scale, y^+, a normalization representative of the small scales of the flow.

As was visually gleaned from figure 3, the spectral plots illustrate that closer to the wall, the generated flow field is missing lots of the small scales of the flow, leading to a poor representation of the flow. However, as the distance from the wall is increased, the energy spectra of the reference and generated velocity field start to align at the small wavenumbers, corresponding to the large scales, reaching almost perfect alignment near the center of the channel. The large wavenumbers, corresponding to small scales, are not well represented at any distance from the wall as was observed qualitatively earlier. As such, it is evident that the model is capable of capturing the large scales but not the small scales. This is inline with the previous experience of VAE models and their tendency to blur the generated images.



(a) Streamwise velocity example. (b) Wall-normal velocity example. (c) Spanwise velocity example.

Figure 2. Representative examples of the reference turbulent channel flow fields at Re_τ.



(a) Streamwise velocity example. (b) Wall-normal velocity example. (c) Spanwise velocity example.

Figure 3. Representative examples of the generated turbulent channel flow fields at Re_τ.

5.5. Quantitative Analysis - Velocity Component Cross-correlation

Beyond spectral content, the different velocity components, or image channels, need to be correlated in a certain way to transport momentum efficiently. Figure 6 illustrates scatter plots between the streamwise velocity, u', and the wall-normal velocity, v', representing the joint probability density function between the two components.

It appears that closer to the wall, the model is capable of capturing the correct correlation trends between the two velocity components, up to $y/h \approx 0.5$. However, while the trend is accurate, the spread of the data is much narrower in the generated field, which is inline with the visual observation of the generated velocities being much more correlated than the reference velocities in figures 3 and 2, respectively. As we traverse further away from the wall, $y/h > 0.5$, the model retains the same correlation trends and does not decorrelate like the true velocity field. In fact, the spread is reduced. The mean cross-correlation of these joint probability density functions are reported in table 2. Due to the reduced spread, the mean increases beyond $y/h > 0.5$. It appears that the spectra get more accurate closer to the centerline, but the cross-correlations do not. The explanation for this is currently unknown.

y/h	y^+	$\rho^{\mathrm{g}}_{(u',v')}$	$\rho^{\mathrm{r}}_{(u',v')}$
0.012	11.26	-0.737	-0.413
0.043	40.22	-0.387	-0.171
0.076	71.09	-0.405	-0.215
0.114	106.7	-0.405	-0.215
0.202	188.4	-0.475	-0.203
0.310	290.0	-0.552	-0.242
0.507	473.6	-0.775	-0.441
1.000	940.0	-1.815	0.056

Table 2. The cross-correlation $\rho_{(\alpha,\beta)}$ between the streamwise, u', and the wall-normal, v', velocities as a function of the wall-normal distance normalized by the channel half-height y/h and normalized by the viscous length scale y^+. $\rho^r_{(u',v')}$ and $\rho^g_{(u',v')}$ are the cross-correlations of the reference and the generated data, respectively.

5.6. Variation of the Reparameterizaion Sampling Statistics

The performance of the trained VAE is further evaluated by sampling the random noise over a non-standard Gaussian distribution function. The evaluation was conducted by individually varying μ and $\log(\sigma^2)$ from -1 to 1 respectively. Figure 7 illustrates the effect of this variation of parameters. It appears that the generated fields only vary in their details but the overall structure remains the same with the afore-

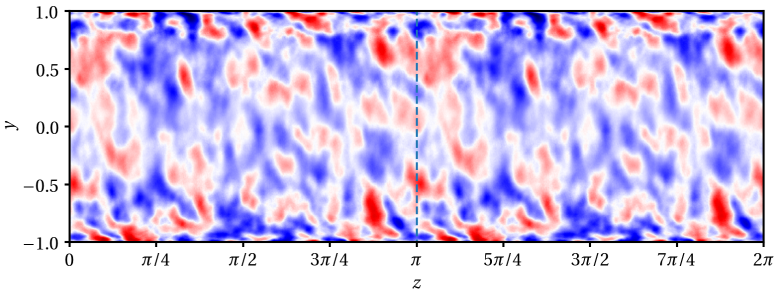
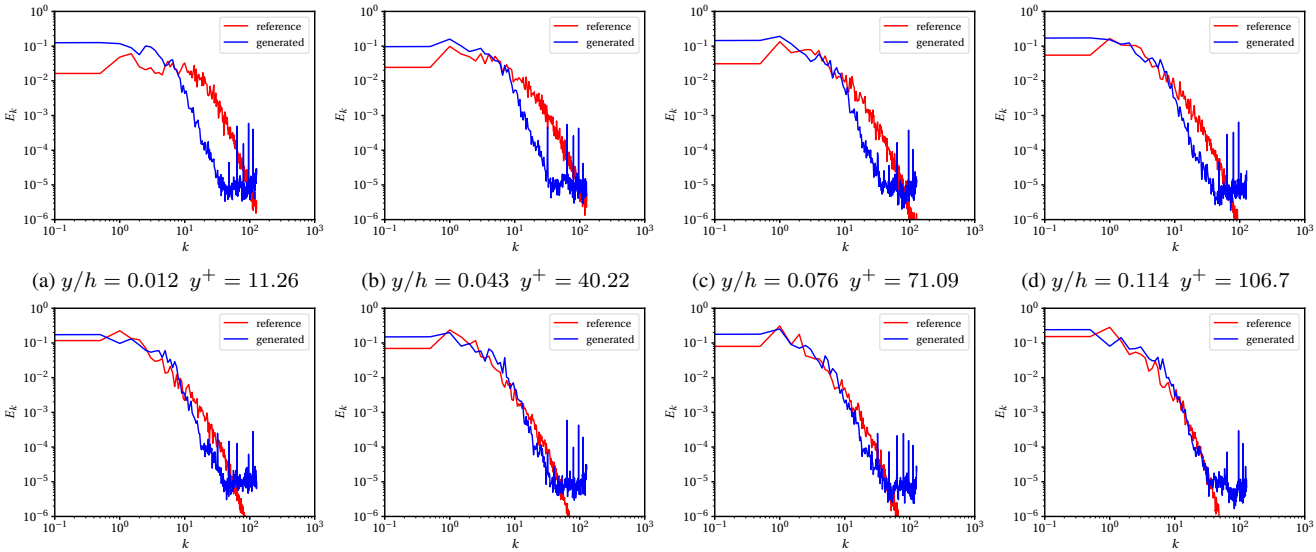


Figure 4. A concatenated velocity field showing the periodicity enforced by circular padding. The dashed line represents demarcation between the original and the laterally translated copy of the same field.



(a) $y/h = 0.012$ $y^+ = 11.26$ (b) $y/h = 0.043$ $y^+ = 40.22$ (c) $y/h = 0.076$ $y^+ = 71.09$ (d) $y/h = 0.114$ $y^+ = 106.7$

(e) $y/h = 0.202$ $y^+ = 188.4$ (f) $y/h = 0.310$ $y^+ = 290.0$ (g) $y/h = 1.000$ $y^+ = 940.0$ (h) $y/h = 1.000$ $y^+ = 940.0$

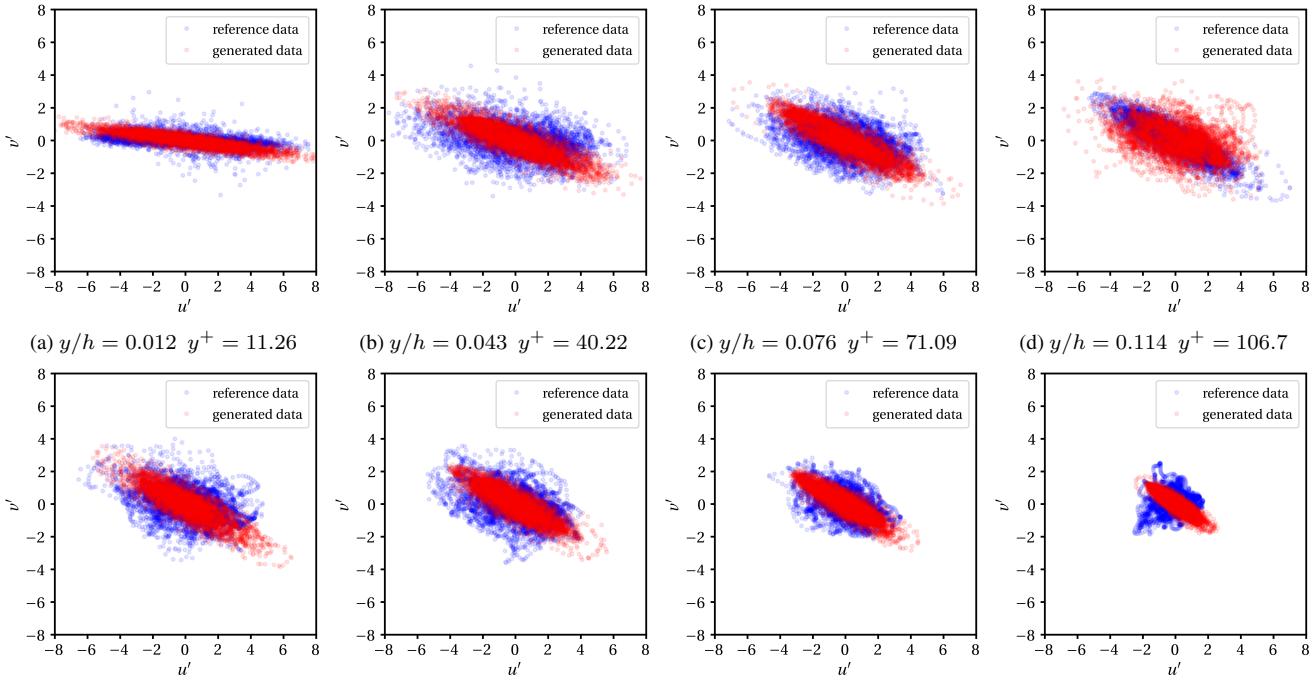
Figure 5. Energy spectra, or power spectral density of the streamwise velocity, u', as a function of the spanwise wavenumber and the wall-normal distance.

mentioned quantitative analysis for $\mu = 0$ and $\sigma = 1$ still applicable.

6. Conclusions and Future Work

In this study, we aimed to generate cross-sectional turbulent fields to be used as boundary conditions for high Re_τ spatially varying turbulent boundary layer simulations. If successful, the proposed application would open the possibility for larger high-fidelity calculations [10]. Furthermore, we aimed for the generator to not require an initial inflow seed. The simplest of such generative models is the Variational Auto-Encoder (VAE) [3,6,11], which we implemented from scratch using PyTorch.

Due to its ubiquitous availability, channel flow data was used for training the VAE model [7]. Both qualitative and quantitative analyses were conducted to measure the fidelity of the generated fields. The qualitative comparison between the reference and the generated fields indicated that the final model is capable of capturing the large scales of the flow in the center of the domain, but was incapable of representing the finer scales, both far and close to the wall. This is inline with the blurring effect commonly attributed to VAE models. This observation was verified using spectral quantitative analysis illustrated in Figure 5. Cross-correlation analysis between the velocity components, or image channels, showed that near the wall, the correlation was captured well, but its fidelity deteriorated as the center of the channel was approached. This contradictory result, of capturing the scales but their correlation correctly near the center of the channel, but the reverse being true close to the wall is currently not understood. It was found that deeper VAE models, with larger latent vector sizes and large batch sizes perform best. This is conceptually aligned with the multi-scale nature of turbulence that was in full effect at the high



(a) $y/h = 0.012$ $y^+ = 11.26$ (b) $y/h = 0.043$ $y^+ = 40.22$ (c) $y/h = 0.076$ $y^+ = 71.09$ (d) $y/h = 0.114$ $y^+ = 106.7$

(e) $y/h = 0.202$ $y^+ = 188.4$ (f) $y/h = 0.310$ $y^+ = 290.0$ (g) $y/h = 0.507$ $y^+ = 473.6$ (h) $y/h = 1.000$ $y^+ = 940.0$

Figure 6. Scatter plots representing the joint probability density function between the streamwise, u', and wall-normal, v', velocities as a function of the wall-normal distance.

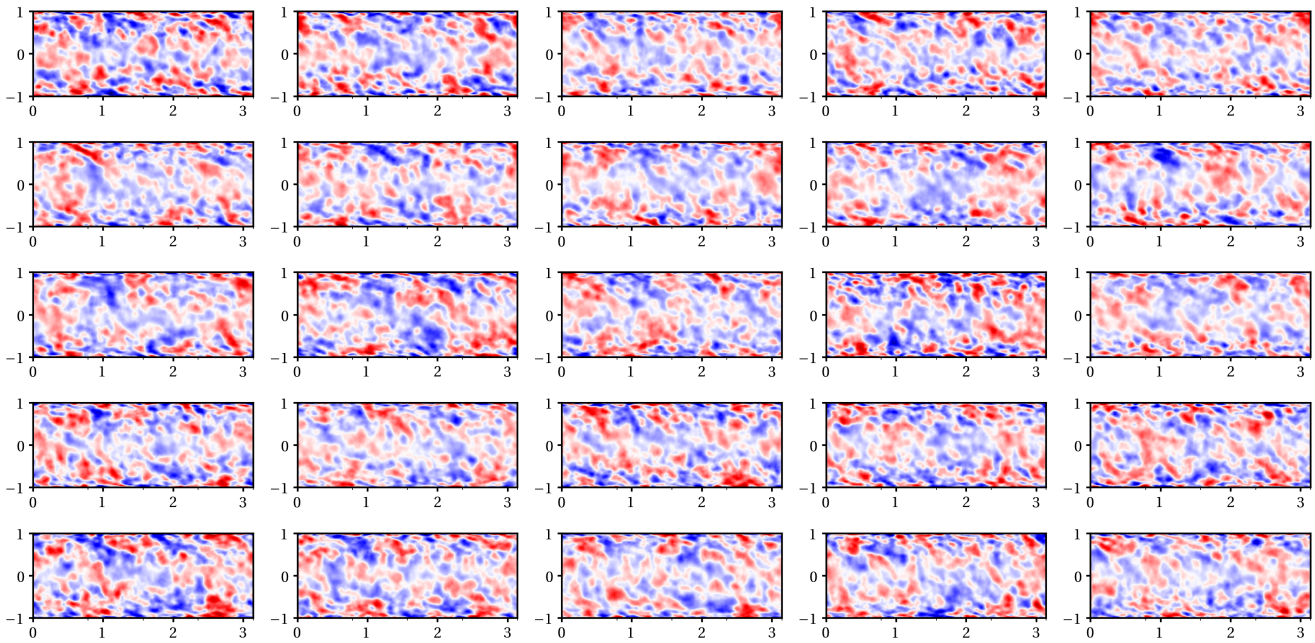


Figure 7. Variation of $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ in the parameterization, $p_{\boldsymbol{\theta}}(\boldsymbol{z}) \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$. Rows represent, from top to and bottom, $\mu \in \{-1, -0.5, 0, +0.5, +1\}$. Columns represent, from left to right, $\log(\sigma^2) \in \{-1, -0.5, 0, +0.5, +1\}$.

Re_τ attempted. Finally, we believe that the field sizes were much larger than what is currently considered state-of-the-art; 385×512 versus 256×256 as stated by [11].

Overall, the potential for a VAE model to represent

wall-bounded turbulent flow at large Re_τ is non-negligible. However, some architectural changes are necessary. In particular, as is expected with a highly multiscale system, a single network architecture is most likely incapable of capturing the flow at any Re_τ. However, a potential work around is a hierarchical generative model, where a portion is responsible for generating a single scale, or perhaps only a few scales, and another portion is responsible for assembling those scales multiple times in space with that number being Re_τ dependent. If given more time and resources, a network of this type will be attempted. Another possibility, which does not involve the design of a novel architecture, is the use of GANs [2], and in particular, StyleGANS [4], which have been showed to generate extremely fine-scale features by using the latent space distribution to influence each convolutional layer.

7. Acknowledgements

The authors are grateful to Mr. Charles Lin and Mr. Joshua Dong for the discussion about the generative neural networks.

8. Contributions

A.E. and H.S. designed the project. A.E. and H.S. designed and trained the neural networks. A.E. and H.S. designed and implemented the evaluation framework. A.E. and H.S. wrote the paper.

References

[1] Kai Fukami, Yusuke Nabae, Ken Kawai, and Koji Fukagata. Synthetic turbulent inflow generator using machine learning. *Physical Review Fluids*, 4(6):064603, 2019. 2, 3

[2] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. 2, 8

[3] Irina Higgins, Loic Matthey, Xavier Glorot, Arka Pal, Benigno Uria, Charles Blundell, Shakir Mohamed, and Alexander Lerchner. Early visual concept learning with unsupervised deep learning. *arXiv preprint arXiv:1606.05579*, 2016. 2, 6

[4] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4401–4410, 2019. 8

[5] Junhyuk Kim and Changhoon Lee. Deep unsupervised learning of turbulence for inflow generation at various reynolds numbers. *Journal of Computational Physics*, 406:109216, 2020. 2, 3

[6] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 1, 2, 6

[7] Adrián Lozano-Durán and Javier Jiménez. Effect of the computational domain on direct simulations of turbulent channels up to re τ= 4200. *Physics of Fluids*, 26(1):011702, 2014. 1, 3, 6

[8] Thomas S Lund, Xiaohua Wu, and Kyle D Squires. Generation of turbulent inflow data for spatially-developing boundary layer simulations. *Journal of computational physics*, 140(2):233–258, 1998. 1

[9] Stephen B Pope and Stephen B Pope. *Turbulent flows*. Cambridge university press, 2000. 3

[10] Juan A Sillero, Javier Jiménez, and Robert D Moser. One-point statistics for turbulent wall-bounded flows at reynolds numbers up to $\delta^+ \approx 2000$. *Physics of Fluids*, 25(10):105102, 2013. 1, 6

[11] Arash Vahdat and Jan Kautz. Nvae: A deep hierarchical variational autoencoder. *Advances in Neural Information Processing Systems*, 33:19667–19679, 2020. 2, 6, 7

[12] Kai Yi, Yi Guo, Yanan Fan, Jan Hamann, and Yu Guang Wang. Cosmo vae: Variational autoencoder for cmb image inpainting. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020. 2