Generative Modeling of Turbulence

Claudia Drygala, Hanno Gottschalk

University of Wuppertal, School of Mathematics and Natural Sciences, IMACM & IZMD
{drygala,hanno.gottschalk}@uni-wuppertal.de

Benjamin Winhart, Francesca di Mare

Ruhr University Bochum, Department of Mechanical Engineering, Chair of Thermal Turbomachines and Aero Engines
{benjamin.winhart,francesca.dimare}@ruhr-uni-bochum.de

Extended abstract of work published in Physics of Fluids 34, 035114 (2022) [1].

Keywords: Generative adversarial networks, Turbulence modeling, Ergodicity, Karman vortex street, LPT stator

Turbulent flows are characterized by unsteadiness, chaotic-like flow states and high degree of non-linearity. The structures involved exhibit a wide range of spatial and temporal scales. In order to capture all scales of fluid motion directly, very fine computational meshes and time steps are required, which makes the computational effort in the case of engineering-relevant problems impossible to accomplish in reasonable time despite the rapidly increasing computer performance. To circumvent this problem, closures are used, which allow to model the structures that cannot be captured by the coarser numerical meshes. However, this advantage in computation time is paid for with a modeling error, which can be considerable depending on the chosen approach and the underlying flow case.

Recent developments in the field of machine learning (ML), which are largely driven by increased computational power as well as the availability of exceptionally large data sets, make it possible to address this issue.

We present a mathematically well founded approach for the synthetic modeling of turbulent flows using generative adversarial networks (GAN). Based on the analysis of chaotic, deterministic systems in terms of ergodicity, we outline a mathematical proof that GAN can actually learn to sample state snapshots from the invariant measure of the chaotic system. Based on this analysis, we study a hierarchy of chaotic systems starting with the Lorenz attractor and then carry on to the modeling of turbulent flows with GAN. As training data, we use fields of velocity fluctuations in two different settings obtained from large eddy simulations (LES) (see figure 1).

GAN consist basically of two mappings - a generator $\phi : \Lambda \to \Omega$ and a discriminator $D : \Omega \to [0, 1]$. Here Λ is a space of latent variables endowed with a probability measure λ that is easy to simulate, e.g. Gaussian noise. The generator ϕ transforms the noise measure λ to the image measure $\phi_*\lambda$. The goal of adversarial learning is, to learn a mapping ϕ from the feedback of the discriminator D, such that D is not able to distinguish synthetic samples from $\phi_*\lambda$ from real samples from the target measure μ. However, the discriminator D is a classifier that is trained to assign real data a high probability of being real and synthetic data a low probability. If ϕ has been so well trained, that even the best discriminator D can not distinguish between samples from μ and $\phi_*\lambda$, generative learning is successful, see also figure 2. In practice, both the generator φ and the discriminator D are realized by neural networks. The training of GAN is organized as a two-player minimax game between D and ϕ. Mathematically, it is described by the min-max optimization problem

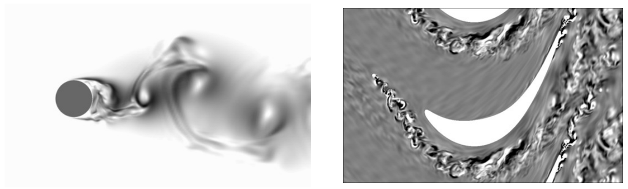
Figure 1: Example snapshots for both investigated settings extracted from the LES. Left: Flow around a cylinder (5, 000 images), right: LPT stator (2, 250 images).

$$\min_{\phi} \max_{D} \mathcal{L}(D, \phi)$$

with the loss function

$$\mathcal{L}(D, \phi) = \mathbb{E}_{\boldsymbol{x}\sim\mu}[\log(D(\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{z}\sim\lambda}[\log(1 - D(\phi(\boldsymbol{z})))] \ .$$

Here, the expected value is denoted by $\mathbb{E}$, the random variable $\boldsymbol{x}$ with values in Ω follows the distribution μ of the real world data and the latent random variable $\boldsymbol{z}$ with values in Λ follows the distribution of the noise measure λ.

In our work two architectures are investigated in detail. We use a deep convolutional GAN (DCGAN) to synthesize the turbulent flow around a cylinder. By conditioning a GAN framework with additional information

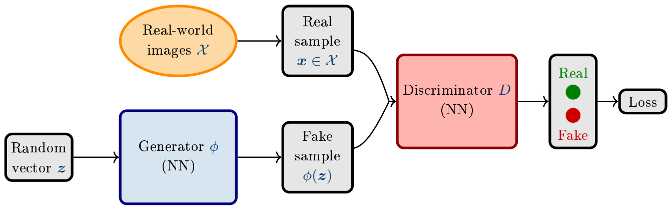
Real-world images $\mathcal{X}$
Real sample $\boldsymbol{x} \in \mathcal{X}$
Discriminator D (NN)
Real
Fake
Loss
Random vector $\boldsymbol{z}$
Generator ϕ (NN)
Fake sample $\phi(\boldsymbol{z})$

Figure 2: Architecture of the original GAN.

it is possible to take the control over the data production process performed by the generator ϕ. The `pix2pixHD` is an extended version of such a conditional GAN framework which allows us to generate high-resolution photo-realistic images from semantic segmentation masks. We simulate the flow around a low pressure turbine (LPT) stator using the `pix2pixHD` architecture being conditioned on the position of a rotating wake in front of the stator (see figure 3). The settings of adversarial training and the effects of using these specific GAN architectures are explained in detail.

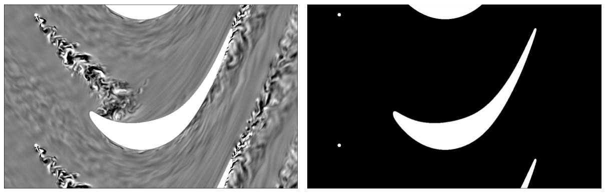

Figure 3: An image of the training set (left) and its corresponding binary segmentation mask (right).

By the investigation of physics-based metrics we show that the statistical properties of GAN-generated and LES flow agree excellently as to observe in figure 4 for the flow around a cylinder. Thus, we are able to produce realistic turbulence by GAN trained from scratch, completely unsupervised and by only having a noise vector as input at inference time in the unconditional case. For training and inference of the conditional GAN, we also additionally use binary segmentation masks but these can be created manually and do not need to be obtained by simulations.

Others than in previous works, the ultimate goal of our research efforts is to devise a structural recognition workflow for a generalised, case-independent synthetisation of turbulent structures which can be carried out independently from a specific configuration. Moreover, we show by investigation of conditional GAN that generators of synthetic turbulent flows can learn to cope with changes of the geometry of the flow path, e.g. caused by a rotation wake. This remains true even if certain positions of the wake are not included into the training data.

Lastly, we show that GAN are efficient in simulating turbulence in technically challenging flow problems on the basis of a moderate amount of training data. GAN training and inference time significantly fall short when compared with classical numerical methods, in particular

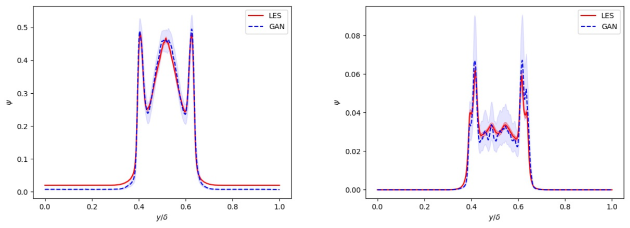

Figure 4: Comparison of the mean pixel values (left) and the statistical fluctuation of the deviation from mean velocities (right) for 5,000 images of flow around a cylinder along the y-axis over a small grid of 12 pixels in x-direction immediately after the wake. The blue and red shaded areas indicate the 95% confidence intervals of the variance for the respective curves. Both data sets were normalized before evaluation.

LES, while still providing turbulent flows in high resolution (see table 1).

So far, we have ignored the physics involved. Therefore, the next step is to feed the GAN with physical parameters so that turbulent flows can also be captured by the GAN in a physically correct manner and hence improve the results regarding the statistical properties even more. Regarding the numerical experiments we will also pay attention to exploring and developing further appropriate evaluation methods. Having provided a first approach to generalization in terms of changes in turbulence space, in future work we will also consider how generalization can be realized in terms of geometries and further boundary conditions.

	LES	GAN-Training	GAN-Inference
Machine	560 CPU cores of Intel Xeon "Skylake" Gold 6132 @2.6 GHz	GPU Quadro RTX 8000 with 48 GB	
Flow around cylinder (DCGAN)	72 core weeks $\hat{=}$ 1 day (for 5,000 images)	1.5 min/epoch ($\approx$ 2 days for 2,000 epochs)	0.001 sec/image ($\approx$ 5 sec for 5,000 images)
LPT stator (`pix2pixHD`)	640 core weeks $\hat{=}$ 8 days (for 2,250 images)	17 min/epoch ($\approx$ 2 days for 200 epochs)	0.01 sec/image ($\approx$ 22.5 sec for 2,250 images)

Table 1: Comparison of computational time.

Acknowledgments C.D. and H.G. thank Hayk Asatryan, Tobias Riedlinger and Matthias Rottmann for discussions and useful advice. The authors also thank Pascal Post for valuable hints for the literature research. We also thank two anonymous referees for hints that helped to improve this paper.

References

[1] C. Drygala, B. Winhart, F. di Mare, and H. Gottschalk. Generative modeling of turbulence. *Physics of Fluids*, 34(3):035114, 2022. `https://doi.org/10.1063/5.0082562`.