Bergische Universität Wuppertal

Fakultät für Mathematik und Naturwissenschaften

Institute of Mathematical Modelling, Analysis and Computational
Mathematics (IMACM)

Claudia Drygala, Benjamin Winhart, Francesca di Mare, Hanno
Gottschalk

Generative Modeling of Turbulence

December 6, 2021

http://www.imacm.uni-wuppertal.de

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

Abstract

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 form 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 obtained from large eddy simulations (LES). Two architectures are investigated in detail: we use a deep, convolutional GAN (DCGAN) to synthesise the turbulent flow around a cylinder. We furthermore simulate the flow around a low pressure turbine stator using the `pix2pixHD` architecture for a conditional DCGAN being conditioned on the position of a rotating wake in front of the stator. The settings of adversarial training and the effects of using specific GAN architectures are explained. We thereby show that GAN are efficient in simulating turbulence in technically challenging flow problems on the basis of a moderate amount of training date. GAN training and inference times significantly fall short when compared with classical numerical methods, in particular LES, while still providing turbulent flows in high resolution.

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

1. Introduction

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, with the ratio of largest to smallest structures scaling with the Reynolds number[19]. 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 (high Reynolds numbers) 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, whereby different approaches can be taken. One obvious approach is ML-based improvement of the prediction quality of existing models, also known as ML augmented turbulence modeling. Here, one possibility is to calibrate the empirically determined constants of the respective models for the underlying use case by means of data-driven ML augmentation [9, 14, 15, 73, 67, 66, 75, 74]. Pioneering publications in this area are the works of Ling et al. [42] and Jiang et al. [31] who used deep neural networks (DNNs) to determine the model constants of nonlinear algebraic eddy viscosity models and were thus able to significantly improve the prediction of anisotropic turbulence effects. Another way is the correction of existing models with the help of additional source terms, which were successfully used in [52, 60, 61, 26] for the augmentation of turbulence models and in [71] for the augmentation of transition models.

A completely different approach has been pursued recently, based on the generative adversarial networks (GAN) as introduced by Goodfellow [24], which allow a hierarchical identification and abstraction of features in images by means of deep neural networks (DNN). By the fact that also in the case of turbulent flows there is a complex superposition of different structures and scales suggests that these methods are well suited for learning the physical relationships in such flows. In [35, 36] it was shown that GAN are able to generate synthesizations of 2D flow fields after they have been previously trained based on DNS data. The reproductions even fulfilled some statistical constraints of turbulent flows such as Kolmogorov's $-5/3$ law and the small scale intermittency of turbulence. Using a deep unsupervised learning approach and a combination of a GAN and a recurrent neural network (RNN), Kim & Lee [33] were able to generate high-resolution turbulent inlet boundary conditions at different Reynolds numbers, which show a statistical similarity to real flow fields.

Another application of GAN is the field of super-resolution reconstruction of turbulent flows. With these methods it is possible to synthetically scale up flow fields which are low-resolution or noisy due to the measurement technique used or, in the case of numerical data, due to limited data storage capacity. [20, 21, 43, 11, 70, 68, 62]. These works assume a supervised learning approach, which means that labeled paired datasets of low-resolution and high-resolution images must be available. Here, the low-resolution data sets are usually generated by filtering the high-resolution data sets obtained, for example, from direct numerical simulations (DNS). In many practical situations, however, such high-resolution data sets are usually not available, which to a certain extent limits the

range of applicability. A more general and therefore more practical approach is the unsupervised super-resolution reconstruction method. Here, pairwise data sets are no longer necessary, as Kim & Lee could show by successfully using an unsupervised GAN for the generation of boundary conditions for turbulent flow [33]. Applications of such methods would be e.g. the augmentation or denoising of experimental data sets (PIV) or the derivation of subgrid-scale models for the application in the field of large-eddy simulation (LES).

In our work, we show the possibilty of synthesizing turbulence structures of a similar quality as predicted by of LES with GAN trained from scratch and completely unsupervised. Thus, we are able to produce data with help of the trained generator by only having a noise vector as input. 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 flowpath, e.g. caused by a rotation wake. We also show that introducing generative learning to model turbulences finds its justification in the enormous reduction of computational time compared to LES, while maintaining the resolution. Lastly, besides the practical aspects, we prove, using the mathematical concept of ergodicity, that learning to generate states of chaotic systems using GAN is possible.

Outline. The paper is organised as follows. In section 2 we briefly summarize the concept of ergodicity, discuss the mathematical foundations behind GAN along with the learning theory for deterministic ergodic systems. Also, a survey of modern GAN architectures is given. The hierarchy of datasets used for our experiments, ranging from the Lorenz attractor and the flow around a cylinder to a perioic wake impinging on a low-pressure turbine stator blade, are described in section 3. This is followed by section 4, where we give details on the training of our various GAN models. In section 5 we discuss the results of our numerical experiments. Finally, in section 6 we present the conclusion and an short outlook.

2. Methodology

In this work we apply generative adversarial networks (GAN) to generate typical states of a deterministic chaotic dynamic system. This is made mathematically precise via the notion of ergodicity [55].

2.1. Ergodicity

Let be $(\Omega, \mathcal{A}, \mu)$ a probability space, consisting of a state space Ω, a collection $\mathcal{A}$ of events/subsets of the state space $A \subseteq \Omega$ known as σ-algebra and a probability measure on $\mathcal{A}$ that attributes the probability $\mu(A)$ to the events $A \in \mathcal{A}$. In our context, the state Ω is chosen as the phase space of a dynamic system $\varphi_t : \Omega \to \Omega$, $t \in \mathbb{R}$, that fulfills $\varphi_t \circ \varphi_s = \varphi_{s+t}$ and $\varphi_0(x_0) = x$.

Frequently in this work, we need the concept of an image measure, i.e. the transformation of a measure by a mapping. To this purpose, let $\varphi : \Omega \to \Omega'$ be a mesurable mapping with respect to the σ-algebra $\mathcal{A}$ on Ω and a second sigma

3

algebra $\mathcal{A}'$ on Ω', i.e. for all $A' \in \mathcal{A}'$ we have $\varphi^{-1}(A') = \{x \in \omega | \varphi(x) \in A'\} \in \mathcal{A}$. The image probability measure of μ under φ, denoted by $\varphi_* \mu$, is then defined by

$$\varphi_* \mu(A') = \mu(\varphi^{-1}(A')) \quad \forall A' \in \mathcal{A}'. \tag{1}$$

In the following, without further mention, we assume all mappings to be measurable with respect to suitable σ-algebras.

In the case considered here, Ω is a state space of a dynamic system. A dynamic system with the given state space consists of a collection of mappings $\varphi^t : \Omega \to \Omega$ that fulfill $\varphi^0 = \mathrm{id}_\Omega$ and $\varphi^t \circ \varphi^s = \varphi^{s+t}$, where $\varphi^t \circ \varphi^s(x) = \varphi^t(\varphi^s(x))$. In many cases, like ours, the state of the dynamic system $\varphi_t(x)$ at time $t \in \mathbb{R}$ is obtained as a solution mapping $\varphi_t : \Omega \to \Omega$ associated with a (discretized) ordinary or partial differential equation starting in the initial state $x \in \Omega$. E.g., $\Omega = \mathbb{R}^3$ for the case of the Lorenz attractor or $\Omega = \mathbb{R}^d$ with d a large number of dimensions of the disrcetized state space of the fluid field in the case of the numerical simulation of turbulent fluids.

The probability measure μ is an invariant measure for the dynamic system defined by φ^t, if all solution mappings φ^t are measure preserving with respect to μ, i.e. $\varphi_*^t \mu = \mu$ for all $t \in \mathbb{R}$.

We next turn to the space of physical observables on Ω and define it as space $\mathcal{H}$ of all square-integrable functions $f : \Omega \to \mathbb{R}$, i.e.

$$\mathcal{H} := L^2(\Omega, \mathcal{A}, \mu) = \left\{ f : \Omega \to \mathbb{R} : f \text{ measurable}, \int_\Omega |f|^2 \, \mathrm{d}\mu < \infty \right\}, \tag{2}$$

We next turn to to the notion of ergodicity, which equates the time average of a dynamic system with the ensemble average of its invariant measure. In mathematical notation, ergodicity of the dynamic system φ_t with respect to the invariant measure μ is defined as

$$\lim_{T \to \infty} \frac{1}{T} \int_0^T f \circ \varphi_t(x_0) \, \mathrm{d}t = \int_\Omega f(x) \, \mathrm{d}\mu(x) = \mathbb{E}_{\boldsymbol{x} \sim \mu}[f(\boldsymbol{x})] \; \forall x_0 \in \Omega. \tag{3}$$

Neumann [47] and Birkhoff [7] established quite general conditions, under which ergodicity holds. See also [16, 3] for extensive treatments of discrete and time continuous ergodic systems.

In some of our numerical experiments, we do not consider the entire state-space Ω, but reduce the degrees of freedom using a mapping $\pi : \Omega \to \Omega'$ with Ω' the reduced state space. Let $\pi_* \mu$ be the projected measure. Assuming the ergodictity of the original dynamics φ_t with respect to μ, we see that

$$\lim_{T \to \infty} \frac{1}{T} \int_0^T f \circ \pi \circ \varphi_t(x_0) \, \mathrm{d}t = \int_{\Omega'} f(x') \, \mathrm{d}\pi_* \mu(x') \; \forall x_0 \in \Omega, \tag{4}$$

whenever $f \circ \pi$ is square integrable with respect to μ. This easily follows from the general transformation formula $\int_{\Omega'} f(x') \, \mathrm{d}\, \pi_* \mu(x') = \int_{\Omega'} f \circ \pi(x) \, \mathrm{d}\, \mu(x)$ and (3). Hence, ergodicity remains meaningful on the reduced state space Ω', even if the dynamics φ_t can not be consistently formulated on Ω'.

4

2.2. Mathematical foundations of generative learning for ergodic systems

Generative Adversarial Networks (GAN) consist 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. experiments uniform or 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 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 Fig. 1.

In practice, both the generator φ and the discriminator D are realized by neural networks. The feedback of D to ϕ is transported backwards by back-propagation [58] through the concatenated mapping $D \circ \phi$ in order to train the weights of the neural network ϕ. At the same time, the universal approximation property of (deep) neural networks guarantees that any mappings ϕ and D can be represented with a given precision, provided the architecture of the networks is sufficiently wide and deep, see [23, 5, 56, 64, 30, 76, 65, 32] for qualitative and quantitative results.

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

$$\min_{\phi} \max_{D} \mathcal{L}(D, \phi) \tag{5}$$

with the loss function, also known as binary cross-entropy [28]

$$\mathcal{L}(D, \phi) = \mathbb{E}_{\boldsymbol{x} \sim \mu}[\log(D(\boldsymbol{x}))] + \mathbb{E}_{\boldsymbol{z} \sim \lambda}[\log(1 - D(\phi(\boldsymbol{z})))] . \tag{6}$$

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. As has been observed in [23],

$$\max_{D \in \mathcal{H}_D} \mathcal{L}(D, \phi) = \mathfrak{d}_{\mathrm{JS}}(\mu \| \phi_* \lambda) + \log(4) \tag{7}$$

if the maximum is taken over a sufficiently large hypothesis space $\mathcal{H}_D$ of discriminators. Here, $\mathfrak{d}_{\mathrm{JS}}(\mu \| \phi_* \lambda)$ stands for an information theoretic pseudo distance between the invariant measure μ and the generated measure $\phi_* \lambda$ known as the Jensen-Shannon divergence

$$\mathfrak{d}_{\mathrm{JS}}(\mu \| \phi_* \lambda) = \mathfrak{d}_{\mathrm{KL}} \left(\mu \left\| \frac{\phi_* \lambda + \mu}{2} \right. \right) + \mathfrak{d}_{\mathrm{KL}} \left(\phi_* \lambda \left\| \frac{\phi_* \lambda + \mu}{2} \right. \right), \tag{8}$$

with $\mathfrak{d}_{\mathrm{KL}}(\mu \| \nu) = -\mathbb{E}_{\boldsymbol{x} \sim \mu} \left[\log \left(\frac{f_\nu}{f_\mu}(\boldsymbol{x}) \right) \right]$ the Kulback-Leibler pseudo distance between the measures ν and μ with continuous probability densities f_μ and

5

f_ν, respectively. Note that $\mathfrak{d}_{\mathrm{KL}}(\mu\|\nu) = 0$ holds if and only if $f_\mu(\boldsymbol{x}) = f_\nu(\boldsymbol{x})$ holds with μ-probability one and hence $\mu = \nu$. Consequently, also $\mathfrak{d}_{\mathrm{JS}}(\mu\|\phi_*\lambda)$ measures the distance between μ and $\phi_*\lambda$.
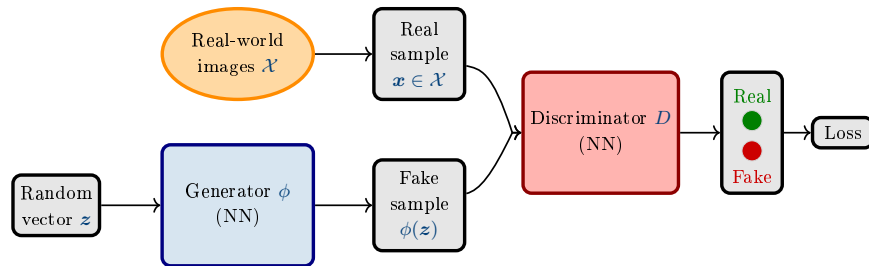

Figure 1: Architecture of the original GAN. According to: [23]. The generator produces fake samples $\phi(\boldsymbol{z}) \sim \phi_*\lambda$ by the random vector $\boldsymbol{z}$ as its input. On the other hand the real-world data $\mathcal{X}$ representing the training data is given. The discriminator gets as input fake as well as real samples and estimates the probability that the given input sample comes from $\mathcal{X}$ than generated by ϕ. Thus, the output of D is a single scalar value per sample in the range of $[0, 1]$. The feedback of the discriminator reaches the generator when the weights of the GAN framework are updated by backpropagation [58] during the training. Since both networks are fully differentiable and trained end-to-end the whole GAN framework can be backpropagated in one go using the same loss function for ϕ and D. The optimum of the problem (5) is reached if the distribution of the real-world data is captured by the generator and the discriminator is not able to distinguish real from fake samples, so $\phi_*\lambda = \mu$ and $D(\cdot) = 1/2$.

2.3. Learning theory for deterministic ergodic systems

In this work, we show that it is possible to model turbulent flows with GAN in practice. In this section we outline a proof that generative learning for deterministic ergodic systems converges in the limit of large observation time T.

As described in section 2.2 μ is the unknown invariant measure encoding the statistical properties of the dynamic system $\varphi_t(x_0)$ with x_0 the initial state. Our goal is to sample from μ but since it is unknown, we want to learn it from the data given by the observed trajectory $\varphi_t(x_0)$. Thus, in context of generative learning a generator $\tilde{\phi}$ is searched for which holds $\tilde{\phi}_*\lambda = \mu$, where λ is, e.g., the Lebesgue measure than corresponds to d-dimensional uniform noise.

Let μ be the invariant measure of the dynamic system $\{\varphi_t\}_{t\in\mathbb{R}}$ acting on the measurable space $([0, 1]^d, \mathcal{B}([0, 1]^d))$ with $\mathcal{B}([0, 1]^d)$ the Borel-σ-algebra and $[0, 1]^d$ the sample space of state configurations with normalized state components in $[0, 1]$. It is assumed that $d\mu(x) = f(x)d\lambda(x)$ with the continuous probability density $f(x) > 0$ in the space of k-times differentiable α-Hölder functions $\mathcal{C}^{k,\alpha}([0, 1]^d, \mathbb{R})$ [1]. If this is not the case, one can easily regularize μ to achieve this. Moreover, we assume $\phi : [0, 1]^d \rightarrow [0, 1]^d$ also lies in the space of k-α-Hölder functions $\mathcal{C}^{k,\alpha}([0, 1]^d, \mathbb{R}^d)$, $k \geq 1$. By the realizability theorem of [5] it follows that $\exists \, \phi_0 \in \mathcal{C}^{k,\alpha}([0, 1]^d, \mathbb{R}^d)$, such that

$$\phi_{0_*}\lambda = \mu. \tag{9}$$

By knowing that $\phi_{0_*}\lambda^{(d)}$ is realizable in the hypotheses space

$$\mathcal{H} = \{\phi \in \mathcal{C}^{k,\alpha}([0,1]^d, \mathbb{R}^d) | \|\phi\|_{\mathcal{C}^{k,\alpha}} \leq K, \|\phi^{-1}\|_{\mathcal{C}^{k,\alpha}}\} \tag{10}$$

for $K > 0$ sufficiently large, our goal is to estimate ϕ_0 by $\hat{\phi}_T \in \mathcal{H}$ based on the data given by the ergodic flow $\boldsymbol{\varphi}_T = \{\varphi_t(x_0)\}_{T \geq t \geq 0}$.

The estimation of ϕ_0 is performed using an empirical loss function $\hat{L}(\phi, D, \boldsymbol{\varphi}_T)$ that is designed to approximate the theoretical loss function (6) and hence minimizing the difference between the measure μ of the ergodic system and the image measure $\phi_*\lambda$ of the synthesized images. Mathematically, we search the generator

$$\hat{\phi}_T \in \operatorname*{arg\,min}_{\phi \in \mathcal{H}} \sup_{D \in \mathcal{H}_D} \hat{\mathcal{L}}(\phi, D, \boldsymbol{\varphi}_T) \tag{11}$$

with the discriminator hypotheses space $\mathcal{H}_D$ such that an optimal choice of D is feasible:

$$\mathcal{H}_D = \left\{ D_{\phi,\phi'} = \frac{f_\phi}{f_\phi + f_{\phi'}} \Big| \phi, \phi' \in \mathcal{H} \right\}. \tag{12}$$

Here, $f_\phi(x) = |\det(D\phi^{-1})(x)|$ stands for the continuous probability density associated with the probability measure $\phi_*\lambda$. We propose

$$\hat{\mathcal{L}}(\phi, D, \boldsymbol{\varphi}_T) = \frac{1}{T} \int_0^T \log(D(\varphi_t(x_0)))\, \mathrm{d}t + \frac{1}{[T]} \sum_{j=1}^{[T]} \log(1 - D(\phi(z_j))) \tag{13}$$

as empirical loss function for the ergodic system where $[-] : \mathbb{R} \to \mathbb{Z}$ denotes the rounding function.

Apparently, in the limit $T \to \infty$ by ergodicity (3), the first term converges to the first term in (6) whereas the second term converges almost surely by the law of large numbers. Therefore, the generator $\hat{\phi}$ that is learned from the empirical loss function (13) for large T will approximately solve the minimax problem (5), which by (7) relates to the Jensen-Shannon distance between the estimated measure $\hat{\phi}_{T_*}\lambda$ and the invariant measure μ of the ergodic system. In particular, we obtain the following:

Theorem 1. *Under the assumptions above it holds almost surely*[1] *that*

$$\lim_{T \to \infty} \mathfrak{d}_{JS}(\mu\|\hat{\phi}_{T_*}\lambda) = 0. \tag{14}$$

Proof. Here we give a sketch of the proof. For a detailed argument in a related situation, see [5]. We introduce the following notation: D_ϕ is the discriminator solving $D_\phi \in \operatorname{arg\,max} \mathcal{L}(\phi, D, \boldsymbol{\varphi}_T)$ and, likewise, $\hat{D}_\phi \in \operatorname{arg\,max} \hat{\mathcal{L}}(\phi, D, \boldsymbol{\varphi}_T)$,

---

[1]w.r.t. the probability measure used for the sampling of the latent noise variables z_j.

where we suppressed the suppressed the T dependence of D and $\hat{D}$ to ease the notation. We obtain the estimate

$$
\begin{aligned}
\mathfrak{d}_{\mathrm{JS}}(\mu\|\phi_*\lambda) &= \mathcal{L}(\hat{\phi}_T, D_{\hat{\phi}_T}) - \log(4) \\
&\leq \hat{\mathcal{L}}(\hat{\phi}_T, D_{\hat{\phi}_T}, \boldsymbol{\varphi}_T) + \sup_{\phi,D}\left|\mathcal{L}(\phi, D) - \hat{\mathcal{L}}(\phi, D, \boldsymbol{\varphi}_T)\right| - \log(4) \\
&\leq \hat{\mathcal{L}}(\hat{\phi}_T, \hat{D}_{\hat{\phi}_T}, \boldsymbol{\varphi}_T) + \sup_{\phi,D}\left|\mathcal{L}(\phi, D) - \hat{\mathcal{L}}(\phi, D, \boldsymbol{\varphi}_T)\right| - \log(4) \\
&\leq \hat{\mathcal{L}}(\phi_0, \hat{D}_{\hat{\phi}_T}, \boldsymbol{\varphi}_T) + \sup_{\phi,D}\left|\mathcal{L}(\phi, D) - \hat{\mathcal{L}}(\phi, D, \boldsymbol{\varphi}_T)\right| - \log(4) \\
&\leq \mathcal{L}(\phi_0, \hat{D}_{\hat{\phi}_T}) + 2\sup_{\phi,D}\left|\mathcal{L}(\phi, D) - \hat{\mathcal{L}}(\phi, D, \boldsymbol{\varphi}_T)\right| - \log(4) \\
&\leq \mathcal{L}(\phi_0, D_{\phi_0}) + 2\sup_{\phi,D}\left|\mathcal{L}(\phi, D) - \hat{\mathcal{L}}(\phi, D, \boldsymbol{\varphi}_T)\right| - \log(4) \\
&= \mathfrak{d}_{\mathrm{JS}}(\mu\|\phi_{0*}\lambda) + 2\sup_{\phi,D}\left|\mathcal{L}(\phi, D) - \hat{\mathcal{L}}(\phi, D, \boldsymbol{\varphi}_T)\right| \\
&= 2\sup_{\phi,D}\left|\mathcal{L}(\phi, D) - \hat{\mathcal{L}}(\phi, D, \boldsymbol{\varphi}_T)\right|
\end{aligned}
\tag{15}
$$

In the first equality we used (7). In the third line, the definition of $\hat{D}_{\hat{\phi}_T}$ was used and in the fourth line we applied (11). In the sixth line, we used the definition of D_{ϕ_0}. In the seventh line, we again used (7) and in the final line we applied (9), which is possible under the given assumptions as proven in [5].

It remains for us to show that the sampling error on the right hand side of (15) vanishes as $T \to \infty$. Note that we can decompose

$$
\begin{aligned}
&\sup_{\phi,D}\left|\mathcal{L}(\phi, D) - \hat{\mathcal{L}}(\phi, D, \boldsymbol{\varphi}_T)\right| \\
&\leq \sup_{\phi,D}\left|\mathbb{E}_{\boldsymbol{x}\sim\mu}[\log(D(\boldsymbol{x}))] - \frac{1}{T}\int_0^T \log(D(\varphi_t(x_0)))\,\mathrm{d}t\right| \\
&+ \sup_{\phi,D}\left|\mathbb{E}_{\boldsymbol{z}\sim\lambda}[\log(1 - D(\phi(\boldsymbol{z})))] - \frac{1}{[T]}\sum_{j=1}^{[T]}\log(1 - D(\phi(z_j)))\right|
\end{aligned}
\tag{16}
$$

The second term on the right hand side vanishes by the uniform law of large numbers, as the hypothesis spaces $\mathcal{H}$ and $\mathcal{H}_D$ can be endowed with $C^{k,\alpha'}$ topologies that are, for $\alpha' < \alpha$, slightly little weaker than the $C^{k,\alpha}$-topology. Nevertheless, the hypothesis spaces under these topologies are compact, see [5] for the details. Consequently, the expression in the first term vanishes by the standard uniform law of large numbers, see e.g. [17].

For the first term, we have already seen that ergodicity implies that the expressions in the absolute value by ergodicity vanish in the limit $T \to \infty$. Also, with respect to the aforementioned $C^{k,\alpha'}$-topologies the hypothesis spaces are compact. Last, it is easy to see that $\frac{1}{T}\int_0^T \log(D(\varphi_t(x_0)))\,\mathrm{d}t$ is equicontinuous

in D wrt. this topology (as $D(x)$ is uniformly lower bounded away from zero in $\mathcal{H}_D$). As for equicontinuous functions, pointwise convergence implies uniform convergence, the first term on the right hand side vanishes as well in the limit $T \to \infty$. □

We note that in practice, the Hölder generators ϕ and discriminators D are replaced by deep neural networks. As such networks possess the universal approximation property, see e.g. [72], one can approximate the Hölder functions to arbitrary precision. Secondly, instead of solving the integral in (13) to compute the loss function, one uses a monte carlo approximation by sampling from the trajectory $\boldsymbol{\varphi}_T$. Theorem 1 remains valid under this replacement, as one can see from one further application of the uniform law of large numbers.

Note however that these theoretical results do not guarantee the success of the numerical experiments. This is mostly due to the fact that the optimization problem (11) is highly non-convex and can not be solved exactly, as e.g. for neural nets this problem is NP-hard [59]. In practice, one rather finds sufficiently good local minima instead of a global optimum. Also, practical issues occur with the choice of the capacity and other elements of architecture of the neural networks.

2.4. Advanced GAN frameworks

After the introduction of the original GAN framework by Goodfellow figure 1, it became apparent that GAN are powerful models which can be applied to a wide variety of tasks by modifying or extending the architecture [51]. In this work three of these modified frameworks are investigated.

Wasserstein GAN (WGAN). The Wasserstein GAN differs from the original GAN mainly in the change of the loss function and thus also in the change of the optimization problem [4]. For the WGAN framework the goal is not to minimize the Jensen Shannon divergence but the Wasserstein distance expressed by the Kantorovich-Rubinstein duality

$$\mathcal{W}(\mu, \phi_* \lambda) = \frac{1}{K} \sup_{\|\psi\|_L \leq K} \left(\mathbb{E}_{\boldsymbol{x} \sim \mu}[\psi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z} \sim \lambda}[\psi(\phi(\boldsymbol{z}))] \right) \tag{17}$$

with the supremum over all K-Lipschitz functions $\psi : \mathcal{C} \to \mathbb{R}$ and $\mathcal{C}$ a compact metric set. Under the satisfaction of certain conditions the authors of [4] showed that the optimization problem

$$\max_{\|\psi\| \leq 1} \left(\mathbb{E}_{\boldsymbol{x} \sim \mu}[\psi(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z} \sim \lambda}[\psi(\phi(\boldsymbol{z}))] \right) \tag{18}$$

has a solution for $K = 1$ and that the gradient of (17) exists.

In practice, the solution of (18) can be approximated by training a neural network $\psi = \psi_w$ parameterized by the weights $w \in \mathcal{W}$ with $\mathcal{W}$ a compact space. This assumption implies that all parameterized functions ψ_w are K-Lipschitz for some $K \geq 1$. To ensure that all weights lie in a compact space and thus the Lipschitz constraint is preserved, the weights are clipped [8] to a certain range after each gradient update in the implementation.

Deep Convolutional GAN (DCGAN). The deep convolutional GAN has the same base architecture as shown in figure 1, but the generator ϕ and the discriminator D are convolutional neural networks (CNNs) [56]. These kind of neural networks are especially in the field of image processing successfully applicable [2, 34]. In order to be able to integrate CNNs into GAN the authors of [56] pointed out which guidelines are to follow to enable a stable training at higher resolution and with deeper architectures.

The stability of the training is ensured by applying batch normalization [29] on the output layer of ϕ and the input layer of D. To work with deeper architectures fully-connected layers [45] should be avoided on top of convolutional features. Finally, the choice of the leaky rectified linear unit (LReLu) activation function [49] for D allows higher resolution modeling. Moreover, the generator captures faster the color space of the distribution μ by applying bounded activation functions in the last layer as the LReLu [49]. Finally, mentionable is that ϕ and D are able to learn their own spatial up- or downsampling by replacing deterministic spatial pooling layers [13] with (fractional-) strided convolutions.

Conditional GAN (cGAN). By conditioning a GAN framework with additional information it is possible to take the control over the data production process performed by the generator ϕ [46]. Thereby, additional information can be represented for example by class labels or semantic segmentation masks [22]. As shown in figure 2 the conditioning can be realized by feeding the supplementary information η to the discriminator D and the generator ϕ as an extra input channel. During training, η is sampled from a data model $\boldsymbol{\eta} \sim \nu$, where ν gives the distribution of η in the data generation process. This extension of the architecture leads to the modified loss function

$$\mathcal{L}_{\mathrm{cond.}}(D, \phi) = \mathbb{E}_{\substack{x \sim \mu \\ \eta \sim \nu}} [\log(D(\boldsymbol{x}|\boldsymbol{\eta}))] + \mathbb{E}_{\substack{z \sim \lambda \\ \eta \sim \nu}} [\log(1 - D(\phi(\boldsymbol{z}|\boldsymbol{\eta})))] \ . \tag{19}$$
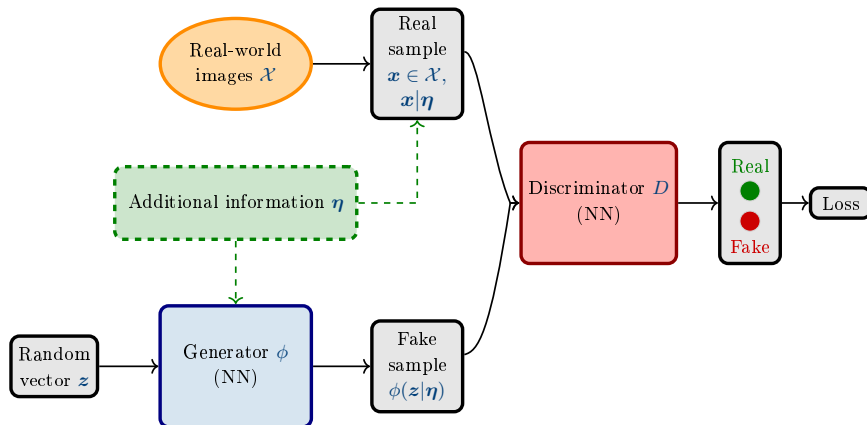

Figure 2: Architecture of a conditional GAN. According to: [46].

10

A special form of the cGAN investigated in this work is the so called `pix2pixHD` introduced by [64]. This conditional adversarial framework allows to generate high-resolution photo-realistic images from semantic segmentation masks. The pix2pixHD framework is based on its former version `pix2pix` [30] whose optimization problem is given as

$$\min_{\phi} \max_{D} \mathcal{L}_{\text{cond.}}(D, \phi, \boldsymbol{x}) \tag{20}$$

with $\mathcal{L}_{\text{cond.}}$ defined as in (19). To improve the photorealism and the resolution of the generated images the architecture was changed by introducing three innovations.

First, a coarse-to-fine generator was implemented. For this, the generator was decomposed into the two sub-networks ϕ_1 having the role of a global generator and ϕ_2 as a local enhancer. By this the global and local information can be aggregated effectively within the generator $\phi = \{\phi_1, \phi_2\}$ for the image synthesis task.

In order for the discriminator to distinguish between generated and real high-resolution images it needs a large receptive field. Therefore, the common discriminator D was replaced by three multi-scale discriminators D_1, D_2 and D_3 which have an identical network architecture, but operate at three different image scales. Hence, the optimization problem (20) extended to

$$\min_{\phi} \max_{D_1, D_2, D_3} \sum_{i=1}^{3} \mathcal{L}_{\text{cond.}}(\phi, D_i) . \tag{21}$$

In particular, a pyramid of images is created during the training by downsampling the input image by factor two and four. Since the discriminator operating on the coarsest scale has the largest receptive field and hence a more global view it is possible to guide the generator producing globally consistent images. Whereas, the discriminator performing on the finest scale is able to make the generator ϕ pay attention to finer details during the data production.

Lastly, a feature matching loss $\mathcal{L}_{FM}$ [64] was added to (21) in order to stabilize the training of the `pix2pixHD` framework. By this, the complete optimization problem is defined as

$$\min_{\phi} \left[\left(\max_{D_1, D_2, D_3} \sum_{i=1}^{3} \mathcal{L}_{\text{cond.}}(\phi, D_i) \right) + \gamma \sum_{i=1}^{3} \mathcal{L}_{FM}(\phi, D_i) \right] \tag{22}$$

with γ the weighting parameter for both terms.

3. Preparation of datasets

The datasets used for generative learning are described below. We proceed from the Lorentz attactor as a simple chaotic system to LES simulations of simple and complex turbulent flows.

11

3.1. Lorenz attractor

The Lorenz attractor is a non-periodic, non-linear and deterministic ergodic system which is given by the system of ordinary differential equations [44] :

$$
\begin{aligned}
\frac{dx}{dt} &= \sigma(y - x) \\
\frac{dy}{dt} &= x(\rho - z) - y \\
\frac{dz}{dt} &= xy - \beta z
\end{aligned}
\tag{23}
$$

By [63] it has been proven that this dynamic system is representing a strange attractor. Within this hydrodynamic system x describes the rate of convection, y is proportional to the temperature variation between ascending and decreasing flow and z represents the distortion rate of the vertical temperature profile from linearity [44].

The physical parameters are given by σ as the Prandtl number, ρ as the relative Rayleigh number and β representing the measure for the cell geometry. In this work we use the classic parameter values $\sigma = 10$, $\rho = 28$ and $\beta = \frac{8}{3}$ [40].

The training data for the generative learning is given by the points of the attractor's trajectory within the three dimensional space computing the system (23) applying the `odeint` routine of the python package `scipy.integrate` which uses the lsoda algorithm [10]. In total $20,000$ data points of $200,000$ trajectories started from different initial points (x_0, y_0, z_0) randomly sampled within the ranges $x_0 \in [-40, 40]$, $y_0 \in [-30, 40]$ and $z_0 \in [0, 50]$.

3.2. LES

The computational fluid dynamics (CFD) results presented in this paper form the basis for GAN training. They were generated using large-eddy simulations (LES). In this approach, the spatially filtered variant of the Navier-Stokes equations is solved, with the computational grid designed to provide a resolution of at least 80% of the turbulent kinetic energy (TKE) of the flow. The effect of smaller turbulent structures, which are not captured by the grid, are represented using semiempirical models, the so-called subgrid scale models [27]. The spatial filter is thus implicitly given by the computational grid. The LES approach is reasonable, because it is the large vortex structures that transport the bulk of the energy [18] while the smaller structures can be considered to be mainly isotropic and homogeneous (not in the close vicinity of solid walls) by the assumption of local isotropy according to Kolmogorov [38], which simplifies their modeling considerably.

3.3. Test-cases & numerical setup

Two different test cases were chosen for training of GAN, which differ in the complexity of the resulting flow field. Both simulations were performed with the commercial flow solver ANSYS Fluent which was set up to solve the incompressible variant of the spatially filtered Navier-Stokes equations. For time

integration, a non-iterative time advancement scheme is used in combination with a fractional step method for pressure-velocity coupling. The advective fluxes are treated by a bounded central scheme in order to introduce as low numerical dissipation as possible to avoid unphysical dampening of small turbulent structures [69].
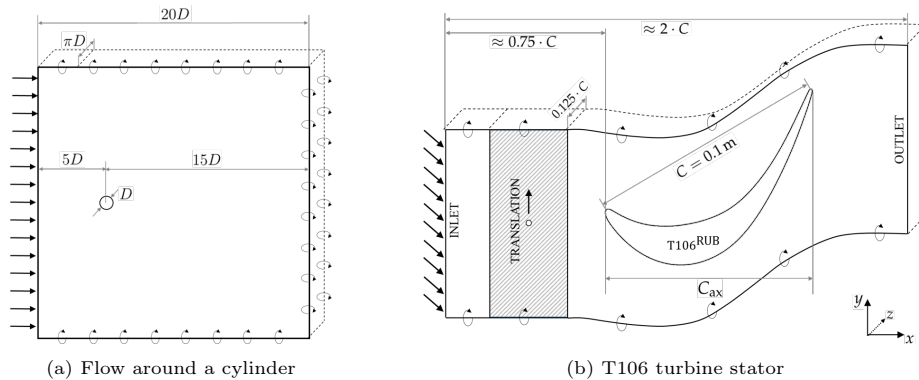

(a) Flow around a cylinder

(b) T106 turbine stator

Figure 3: Numerical domains for both investigated test cases

3.3.1. Flow around a cylinder

The first test case is the flow around a cylinder at Reynolds number 3900. This is a widely used test case, which has been studied in great detail in the literature both experimentally [53, 48, 50] and numerically [53, 6, 39]. The flow field in this case is characterized by a Kármán vortex street, that forms in the wake region of the cylinder and consists of the typical coherent vortex system, where the axis of rotation of the individual vortices is parallel to the axis of the cylinder. A schematic representation of the numerical domain is shown in Fig. 3a. The computational grid consists of a total of 15 million cells. The time step was chosen so that the CFL number was on the order of unity, and the simulation was run for a total of $25,000$ time steps after initial transient effects had disappeared, which corresponds to a total physical time period of approximately 1.45 seconds.

3.3.2. T106 turbine stator under periodic wake impact

The second test case is an academic low-pressure turbine (LPT) stator under periodic wake impact. In this configuration, the wakes, which are comparable to those of the cylinder test case described above, are artificially generated by means of an upstream mounted rotating bar grid. The wakes are convected into the stator passages where deformation occurs as a consequence of the flow turning within the passage. Furthermore, a complex interaction between the wakes and the periodically detaching boundary layer takes place in the rear region of the suction side of the LPT stator, which in total makes this test case an interesting demonstrator for complex turbulent interaction phenomena. A

13

schematic representation of the numerical domain is shown in Fig. 3b. The computational grid consists of a total of approx. 72 million elements. The time step was chosen so that the CFL number was on the order of unity, and the simulation was run for a total of 22,500 time steps after initial transient effects had disappeared, which corresponds to 10 bar passing periods or approx. 1.43×10^{-3} s.

3.4. Data sets and data production

The data sets used for training the GAN were generated by post-processing the transient LES velocity field data. In this process, grayscale images are generated via a projection mapping in the sense of (4). In the case of the flow around a cylinder experiment, the gray scale is showing the distribution of the absolute deviation of the local fluctuating velocity magnitude $c(\xi, t) = \sqrt{u(\xi, t)^2 + v(\xi, t)^2 + w(\xi, t)^2}$ at the location ξ from its time average

$$c'(\xi, t) = |c(\xi, t) - \bar{c}(\xi)|, \quad \bar{c}(\xi) = \frac{1}{T} \int_0^T c(\xi, t) \, \mathrm{d}t. \qquad (24)$$

As the moving wake determines the turbulent flow field in the case of the LPT turbine, time averaging at a fixed point in this case does not make much sense. Therefore, a different representation of the turbulence (or projection mapping) is chosen, which simply depicts the velocity component perpendicular to the image, $w(\xi, t)$. Figure 4 shows an example image for each of the two test cases examined. The gray scale for $w(\xi, t) \approx 0$ is found in the upper left corner of the right panel. Negative values for $w(\xi, t)$ are shown in lighter and positive values in darker grey.

Basic parameters of the generated data sets are summarized in table 1. The time step interval between two successive frames is chosen so that the respective snapshots are sufficiently far apart in time to minimize the correlation between the individual frames.
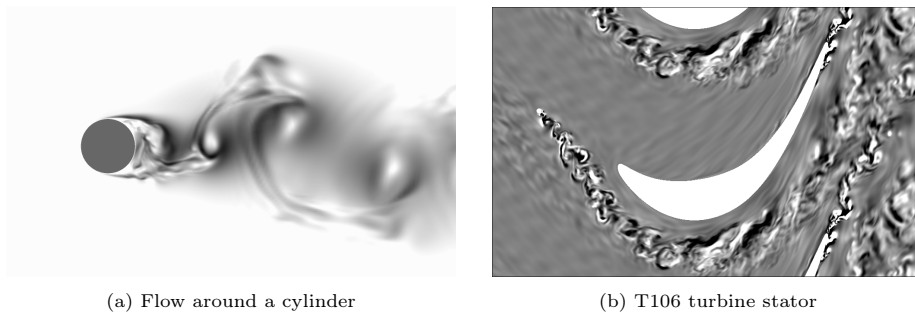

(a) Flow around a cylinder   (b) T106 turbine stator

Figure 4: Example snapshots for both investigated test cases extracted from the LES.

14

Table 1: Summary of the main data set parameters.

	Sampling frequency	Image resolution	Number of files	Total size
Cylinder	68.9 kHz	1000 x 600 px	5, 000	527 MB
Turbine	40.5 kHz	1000 x 625 px	2, 250	700 MB

3.5. Computational cost

At this point, the computational effort of the simulations presented in this paper should be briefly discussed, as this is the main criterion for the applicability of such scale-resolving simulations.

All simulations presented were performed on the in-house High-Performance Computing (HPC) cluster of the Chair of Thermal Turbomachines and Aero Engines, whose main specifications are summarized in table 2.

Table 2: Summary of the main specifications of the HPC cluster.

Partition	Number of nodes	Cores per node	CPU type	RAM	Interconnect
#1	28	28	Intel Xeon "Skylake" Gold 6132 @2.6 GHz	96 GB	Intel Omni-path
#2	8	40	Intel Xeon Scalable Gold 6248 @2.5 GHz	96 GB	Intel Omni-path
TOTAL	**36**	**1104**		**3.4 TB**	

In total 20 computational nodes of the #1 partition of the HPC cluster were allocated in both runs, resulting in a total number of 560 CPU cores. In the case of of the flow around a cylinder, this resulted in a total computation time of about one day for the output run consisting of 25, 000 iterations, which corresponds to about 72 core weeks. In the case of T106 LPT stator, the calculation time was approx. 8 days for the output run consisting of 2, 250 time steps, which corresponds to 10 bar passings, i.e. approx. 640 core weeks.

4. Setup and configuration of GAN training

The implementations details of the training with the GAN frameworks introduced in section 2.4 are summarized for the different datasets in the following. All GAN were set up and trained using the `PyTorch` [54].

4.1. Lorenz attractor

The Lorenz attractor was trained by a original GAN with a discriminator consisting of four fully connected hidden layers [25] with 1024, 512, 256 and 64 neurons. Since the attractor is a deterministic ergodic system [44] Gaussian noise was added to the network of the discriminator as well as to the real input data to regularize the training and hence reduce overfitting [5, 12]. The real data

representing the training data is given by the points of the attractor's trajectory within the three dimensional space as described in section 3.1.

The generator is also given by a fully connected neural network composed of three hidden layers with $256, 512$ and 1024 neurons. Its input is given by a random vector of dimension 100×1 whose elements come from the standard normal Gaussian distribution.

Both neural networks ϕ and D apply the ReLu activation function for the input and hidden layers. The activation of the output layer of the discriminator is given by a sigmoid function and for the generator by a linear function.

The GAN framework was trained for $200,000$ epochs with a batch size of $20,000$. Hence, the trajectory consisting of $20,000$ data points was regarded during one epoch whereby the trajectory started from different randomly sampled initial points (x_0, y_0, z_0) lying in the ranges $x_0 \in [-40, 40]$, $y_0 \in [-30, 40]$ and $z_0 \in [0, 50]$.

The optimization problem was given as in (5). To update the weights of the neural networks ϕ and D the Adam optimizer [37] was applied with the parameter $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a learning rate of 2×10^{-4}. Here, only half of the batch size was used to update the weights of the discriminator.

4.2. Flow around a cylinder

Experiments have been performed on this dataset using the original GAN, WGAN and DCGAN framework. For the original GAN and WGAN the discriminator is given by a fully connected neural network with five layers in total whereby the hidden layers consist of $1024, 512$ and 256 neurons. The generator of both GAN frameworks also consists of five fully connected layers in total with the number of $256, 512$ and 1024 neurons for the hidden layers. In exception of the output layer the Leaky ReLu is applied as activation function. The last layer of the generator is activated by the hyperbolic tangent function. For the original GAN the discriminators last layer is activated by the sigmoid function and the linear activation function is used in case of the WGAN. For the training of the DCGAN the architecture suggested by [41] was adopted.

The three investigated GAN frameworks take images of size $k \times k$ as input. In our experiments we investigated the training with $k \in \{64, 128, 256, 512\}$. We trained all GAN for 200 epochs with a batch size of 20 using $5,000$ images of the dataset. For further investigations the DCGAN training was continued up to epoch $2,000$. The input vector of the generator consists of 100 elements randomly sampled of the standard Gaussian distribution.

For the update of the weights, the Adam optimizer is applied in case of the original GAN and DCGAN with the parameter settings $\beta_2 = 0.5$ and $\beta_2 = 0.999$ and a learning rate of 2×10^{-4} is used. For the WGAN the weight update is realized by the optimizer RMSProp [57] with a learning rate of 5×10^{-5} whereby the weights are clipped to the range $[-0, 01, 0.01]$.

4.3. T106 turbine stator under periodic wake impact

The DCGAN has been also trained for 2000 epochs and $k = 512$ on the whole dataset of the wake disturbed turbine stator-row with the parameter settings
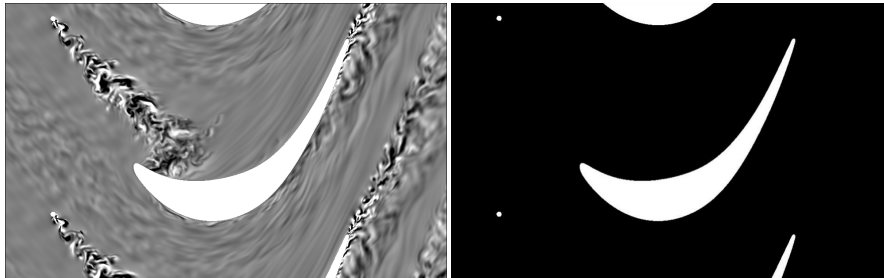
16

Figure 5: An image of the training set (left) and its corresponding binary segmentation mask (right).

described as in section 4.2.

Moreover, the pix2pixHD has been trained as second GAN framework with this dataset. As described in section 2.4 the pix2pixHD is a conditional GAN and hence incorporates additional information to the training. Here, this supplementary information η is given by the binary segmentation masks shown in figure 5. In terms of conditional GAN-learning (19), this corresponds to a uniform distribution $\boldsymbol{\eta} \sim \nu_{\text{unif.}}$ over the y coordinate of the wake. For the experiments with the pix2pixHD the implementation of [64] has been used with small changes. To avoid the appearance of artifacts in the data synthesized by ϕ we replaced the reflection padding with a replication padding and add a replication padding to the global generator before the convolution during the downsampling procedure.

Contrary to the DCGAN framework it is possible to train the pix2pixHD on images of size $k \times k'$, $k \neq k'$. The only important thing to take care of is that k and k' are divisible by 32. For this reason, the images were resized for the training to size $k \times k' = 992 \times 624$, such that the aspect ratio has been preserved.

Since the GAN is trained in a conditioned fashion the binary masks are also needed during the inference. For this reason, the dataset was split into a training- and test set. The training set contains the first 2000 images of the whole dataset and the test set consists of the remaining 250 images.

The pix2pixHD has been trained for 200 epochs with a batch size of 10. Analogous to the DCGAN the weights were updated by the Adam optimizer with the parameter $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a learning rate of 2×10^{-4}.

5. Results of experiments

The results of the numerical experiments are presented and discussed in this section. In the following, we refer to the process of applying a trained generator to the latent random vector z as inference. At inference time, the latent vector also consists of 100 elements sampled from the standard normal distribution.

17

5.1. Lorenz attractor

As described in section 4.1 we trained a original GAN for $200,000$ epochs in order to synthesize three dimensional data points which come from a trajectory of the Lorenz attractor that has converged towards the strange attractor. For consistency, a trajectory of $20,000$ real data points is considered at inference time as in the training. To get a better overview of the results, 500 data points produced by the trained generator ϕ are shown in figure 6. It can be observed that the generated data points are on or close to the true trajectory of the Lorenz attractor. For the points that do not seem to lie directly on the trajectory, it has to be taken into account that the trajectory shown here is also not very dense due to the small number of data points. Considering randomly sampled real data points of a trajectory consisting of one million data points as it must be noted that the distribution is similar to the one of the synthesized data points. Moreover, it can be seen from the rotated figure 7 that, apart from a few outliers, the generated data points are all located in the area of the trajectory in three-dimensional space.
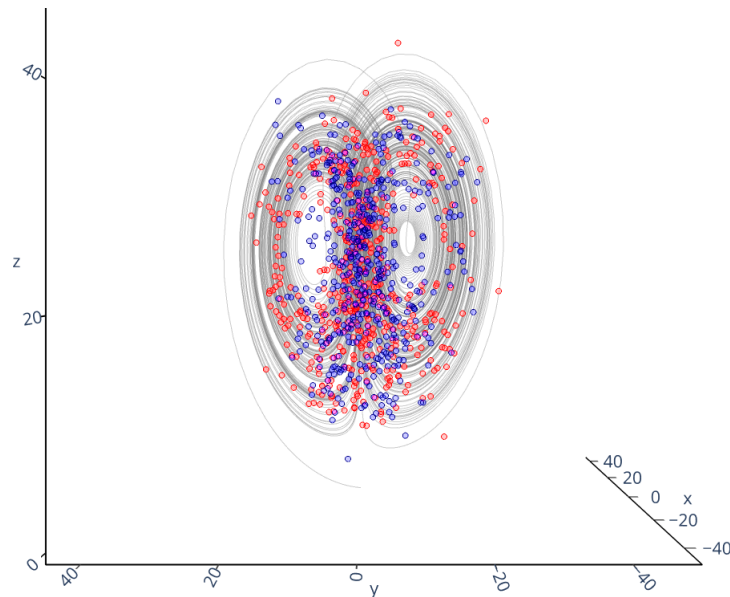
Figure 6: The y-z plane perspective of a trajectory from the Lorenz attractor consisting of $20,000$ data points starting from the initial point $(x, y, z) = (0.1, 0, 0)$ (grey), 500 synthesized data points (blue) and 500 real data points randomly sampled from a trajectory consisting of one million data points (red).

5.2. Flow araound a cylinder

In order to generate the Kármáan vortex street, GAN frameworks with a simpler architecture have been considered first, namely the original GAN and the WGAN. As to observe in figure 8 the trained generators of both GAN are
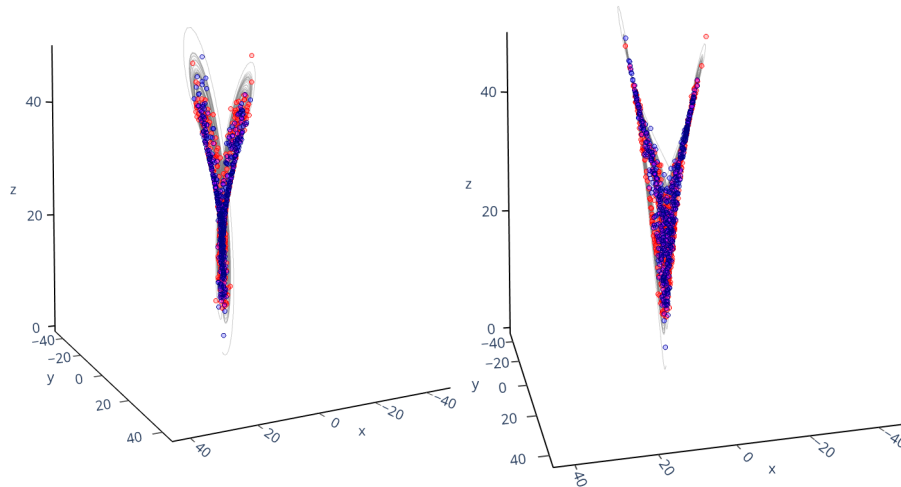
18

Figure 7: Rotated perspective from the trajectory of real data points (grey), synthesized data points (blue) and randomly sampled real data points (red) given in figure 6.

able to position the cylinder in the right place after 200 epochs and that they try to synthesize the wake vortex. However, neither the original GAN nor the WGAN can capture the concrete structure of the vortex street. In addition, it is to observe that the color space has not been learned appropriate by the original GAN such that the generated images are significantly darker than the original images from the LES (see figure 4a). To address these issues, another GAN framework has been considered whose generator and discriminator are represented by convolutional neural networks. As already described in section 2.4, CNNs can be used particularly successfully in image processing. In our experiments, we also found that the DCGAN was able to capture the flow structures after 200 epochs in contrast to the original GAN and the WGAN (see figure 8). To increase the quality of the synthesized images the DCGAN has been further trained out to epoch $2,000$ (see Appendix A for the training progress). Based on figure 9, it can be seen that the images produced by the generator of the DCGAN hardly differ from the real images from the LES after $2,000$ epochs of training.

Finally, it should be mentioned that the networks have been trained on images of size $k \times k$. It has been observed in our experiments that the quality of the generated images have been significantly better with increasing image resolution at inference time. Therefore, we present here the results for the training with images of size 512×512.

5.3. T106 turbine stator under periodic wake impact

Since we got impressive results from the DCGAN for the flow around a cylinder, we trained this GAN framework under the same parameter settings for the second test case. As we observe in figure 10, the LPT stator has been correctly
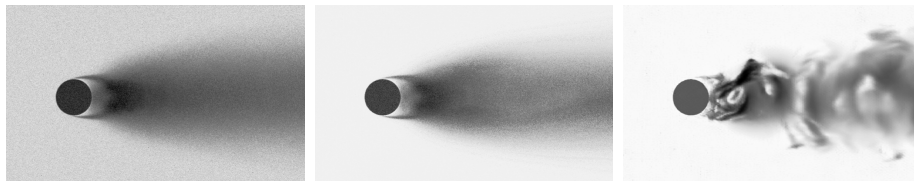
19

Figure 8: Comparison of images synthesized by the generator of a original GAN (left), WGAN (middle) and DCGAN (right) after 200 epochs trained on $5,000$ images of size $k \times k$, $k = 512$.
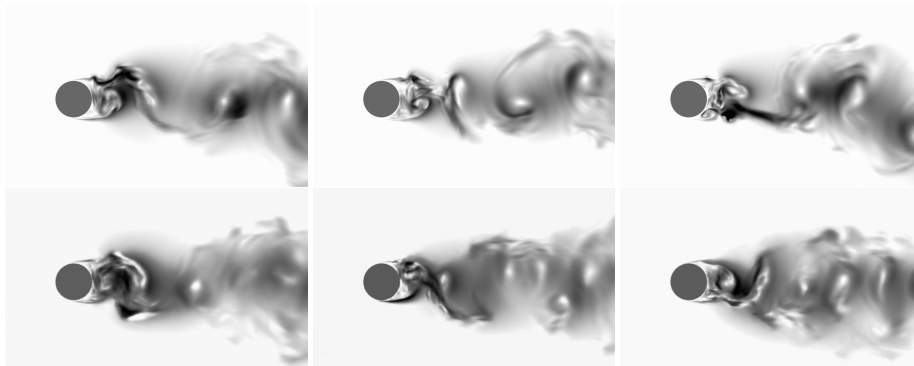

Figure 9: Comparison of images from the LES (top) and synthesized Karman vortex streets produced by the generator ϕ of the DCGAN trained over $2,000$ epochs (bottom).

positioned and the structure of the vortex flows has been also reasonably captured. However, at inference time, the generator has massive problems correctly capturing the position of the cylinder as it periodically slides from bottom to top over time. Especially by direct comparison in figure 11 we can observe, that the structures in the background are not properly captured and the synthesized images are significantly darker than the real images of the LES. To address these problems of the DCGAN we considered the pix2pixHD as another GAN framework. In order to have control over the position of the cylinder at inference, we feed binary segmentation masks shown in figure 5 as additional information η to the GAN framework during training and at inference time (see section 2.4). These masks have the information about the position of the cylinder and the LPT stator. Moreover, we are allowed to generate high resolution images by the pix2pixHD framework such that the structure in the background of the images should also be preserved.

As shown in figure 12, using the generator from pix2pixHD we were able to generate images which again can be hardly qualitatively distinguished from the real image from the LES on a visual level after only 200 epochs (see Appendix A for the training progress). It is also noticeable that the wake vortices do not look identical. Hence, the generator did not simply memorize the structure of the wake vortices at the respective positions and thus variation is given in the synthesized data.
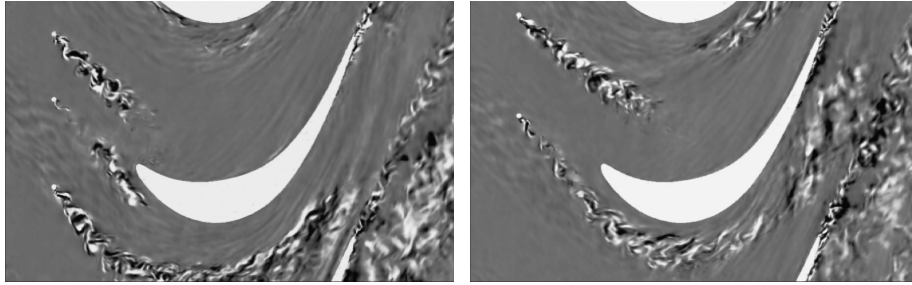
Figure 10: Examples of images synthesized by ϕ of the DCGAN trained on $2,250$ images for $2,000$ epochs.
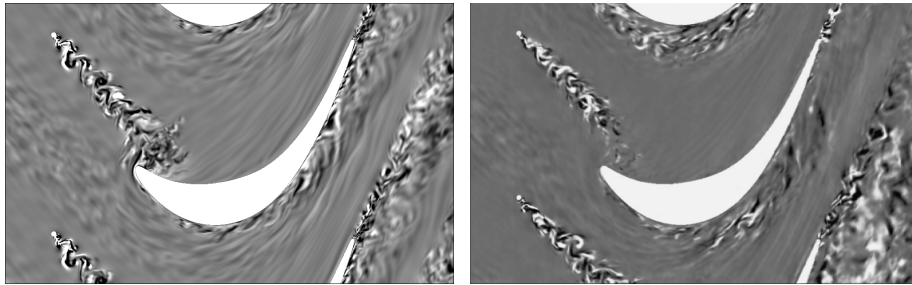

Figure 11: Comparison of a real images from the LES (left) and an images synthesized by ϕ of the DCGAN trained on $2,250$ images for $2,000$ epochs (right).

5.4. Comparison of Computational Costs

Finally, the computational costs of the training and inference performed on a GPU of type Quadro RTX 8000 with 48 GB of the successful GAN frameworks are reported in this section.

The training of the DCGAN with $5,000$ images of the dataset showing the flow around a cylinder has taken 1.5 minutes per epoch. The computational time of pure inference is given by 0.001 seconds per image. Thus, the production of a dataset containing $5,000$ images would take with the beforehand trained generator about 1.67 minutes. This leads to a tremendous amount of time saved compared to one day needed for the generation of the images by the LES.

Since the `pix2pixHD` has a much more complex architecture than the DC-GAN the training of one epoch with $2,000$ images has taken 17 minutes. However, the computational time of pure inference is also given by only 0.01 seconds per image. Hence, the production of $2,250$ images of the LPT stator under periodic wake impact would take about $22.5\,\mathrm{s}$ at inference. Thus, the saved computational time for the data production is very significant in comparison to 8 days for the LES.
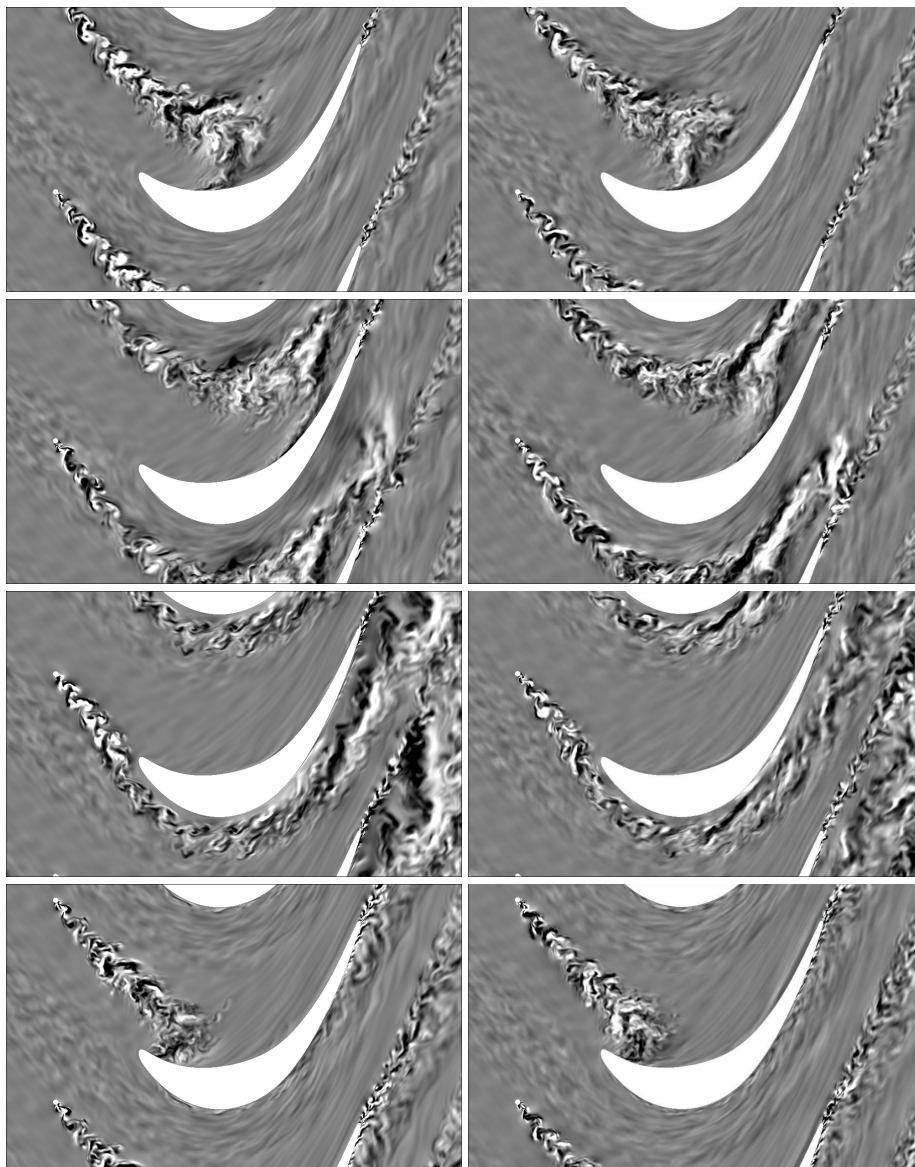
Figure 12: Comparison of images from the LES (left) and synthesized turbulences under periodic wake impact produced by the generator ϕ of the pix2pixHD trained over 200 epochs (right).

6. Conclusion and Outlook

We introduced generative adversarial networks as another way to model turbulence. In doing so, we showed that through generative learning it is possible to synthesize turbulence that matches the quality of LES images on a visual

level while dramatically reducing computational time. Unlike previous work, we trained the GAN from scratch and only require a randomly samples noise vector for the data production in the unconditional case. For training and inference of conditional GAN, we also need binary segmentation masks which can be created manually and do not necessarily need to be obtained by simulations. Using conditional GAN, we have found a solution for generating visually high-quality turbulence when solid objects as the rotation wake change position in space. Thus, we have provided a first approach to generalization with respect to spatial changes. Finally, we have also demonstrated that generative learning of ergodic systems also works at the theoretical level.

So far, we have evaluated our experiments at the visual level. In our future research, numerical experiments are of interest to measure also quantitatively the similarity between the LES images and the images generated by the GAN. Attention will also be given to exploring appropriate methods for making these comparisons. In addition, we have so far 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. 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.

Acknowledgments

Appendix A. Training history of the GAN frameworks

The training progress of the experiments with the DCGAN discussed in section 5.2 is described in figure A.13. Since we trained the GAN framework on images of size 512×512 we also got images of this size as output during the training. It can be observed that the synthesized images already show a quite good quality after 500 epochs. However, on closer inspection, it is noticeable that the structures of the vortex street become finer with an increasing number of training epochs and that the color space is also captured much better after $2,000$. In figure A.14 the development of the synthesized images during the training is illustrated for the pix2pixHD whose results are discussed in section 5.3. Similar to the DCGAN we can observe that the results improve significantly with increasing number of training epochs.
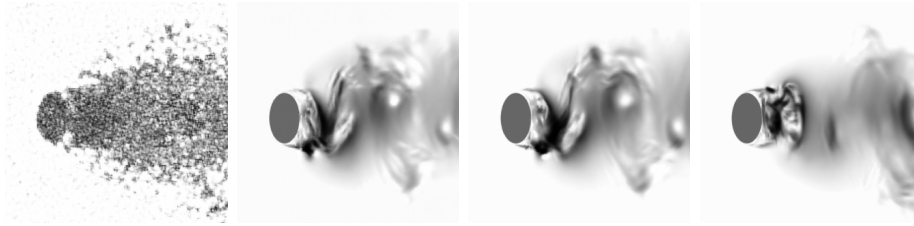
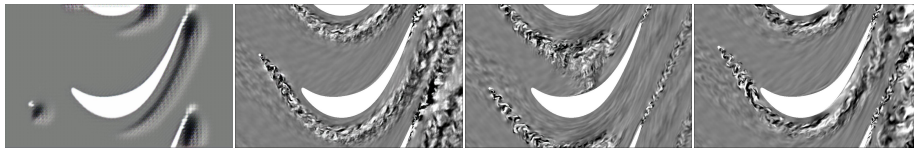Figure A.13: Development of training results after 1, 500, $1,500$ and $2,000$ epochs for the DCGAN.

Figure A.14: Development of training results after 1, 50, 150 and 200 epochs for the `pix2pixHD` framework.

References

[1] Robert A Adams and John JF Fournier. *Sobolev spaces*. Elsevier, 2003.

[2] Saad Albawi, Tareq Abed Mohammed, and Saad Al-Zawi. Understanding of a convolutional neural network. In *2017 International Conference on Engineering and Technology (ICET)*, pages 1–6, 2017.

[3] Hassan Arbabi and Igor Mezić. Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the koopman operator. *SIAM J. Appl. Dyn. Syst.*, 16:2096–2126, 2017.

[4] Martin Arjovsky, Soumith Chintala, and Léon Bottou. Wasserstein generative adversarial networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 214–223. PMLR, 06–11 Aug 2017.

[5] Hayk Asatryan, Hanno Gottschalk, Marieke Lippert, and Matthias Rottmann. A convenient infinite dimensional framework for generative adversarial learning. *arXiv preprint arXiv:2011.12087*, 2020.

[6] Patrick Bruno Beaudan. *Numerical experiments on the flow past a circular cylinder at sub-critical Reynolds number*. PhD thesis, Stanford University, 1995.

[7] George D. Birkhoff. Proof of the ergodic theorem. *Proceedings of the National Academy of Sciences*, 17(12):656–660, 1931.

[8] Xiangyi Chen, Steven Z. Wu, and Mingyi Hong. Understanding gradient clipping in private sgd: A geometric perspective. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 13773–13782. Curran Associates, Inc., 2020.

[9] Sai Hung Cheung, Todd A. Oliver, Ernesto E. Prudencio, Serge Prudhomme, and Robert D. Moser. Bayesian uncertainty analysis with applications to turbulence modeling. *Reliability Engineering & System Safety*, 96(9):1137–1149, 2011. Quantification of Margins and Uncertainties.

[10] The SciPy community. Scipy documentation, 2008-2021. Accessed: 04.12.2021.

[11] Zhiwen Deng, Chuangxin He, Yingzheng Liu, and Kyung Chun Kim. Super-resolution reconstruction of turbulent velocity fields using a generative adversarial network-based artificial intelligence framework. *Physics of Fluids*, 31(12):125111, 2019.

[12] Tom Dietterich. Overfitting and undercomputing in machine learning. *ACM computing surveys (CSUR)*, 27(3):326–327, 1995.

[13] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning, 2018.

[14] W.N. Edeling, P. Cinnella, and R.P. Dwight. Predictive rans simulations via bayesian model-scenario averaging. *Journal of Computational Physics*, 275:65–91, 2014.

[15] Wouter Edeling, Paola Cinnella, Richard Dwight, and Hester Bijl. Bayesian estimates of parameter variability in the $k - \varepsilon$ turbulence model. *Journal of Computational Physics*, 258:73–94, 02 2014.

[16] Tanja Eisner, Bálint Farkas, Markus Haase, and Rainer Nagel. *Operator Theoretic Aspects of Ergodic Theory*. Springer International Publishing, Cham, 2015.

[17] Thomas S Ferguson. *A course in large sample theory*. Routledge, 2017.

[18] J.H. Ferziger and M. Perić. *"Computational Methods for Fluid Dynamics"*. Springer, Berlin, 2008.

[19] Uriel Frisch and Andreĭ Nikolaevich Kolmogorov. *Turbulence: the legacy of AN Kolmogorov*. Cambridge university press, 1995.

[20] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Super-resolution reconstruction of turbulent flows with machine learning. *Journal of Fluid Mechanics*, 870:106–120, May 2019.

[21] Kai Fukami, Koji Fukagata, and Kunihiko Taira. Machine learning based spatio-temporal super resolution reconstruction of turbulent flows, 2020.

[22] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, and Jose Garcia-Rodriguez. A review on deep learning techniques applied to semantic segmentation, 2017.

[23] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Y. Bengio. Generative adversarial networks. *Advances in Neural Information Processing Systems*, 3, 06 2014.

[24] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[25] Martin T Hagan, Howard B Demuth, and Mark Beale. *Neural network design*. PWS Publishing Co., 1997.

[26] Chuangxin He, Yingzheng Liu, and Lian Gan. A data assimilation model for turbulent flows using continuous adjoint formulation. *Physics of Fluids*, 30:105108, 10 2018.

[27] Charles Hirsch. *"Numerical Computation of Internal and External Flows: The Fundamentals of Computational Fluid Dynamics"*. Butterworth-Heinemann , 01 2007.

[28] Yaoshiang Ho and Samuel Wookey. The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling. *IEEE Access*, 8:4806–4813, 2020.

[29] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 448–456, Lille, France, 07–09 Jul 2015. PMLR.

[30] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei Efros. Image-to-image translation with conditional adversarial networks. pages 5967–5976, 07 2017.

[31] Chao Jiang, Junyi Mi, Shujin Laima, and Hui Li. A novel algebraic stress model with machine-learning-assisted parameterization. *Energies*, 13:258, 01 2020.

[32] 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.

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

[34] Phil Kim. *Convolutional Neural Network*, pages 121–147. Apress, Berkeley, CA, 2017.

[35] Ryan King, Peter Graf, and Michael Chertkov. Creating Turbulent Flow Realizations with Generative Adversarial Networks. In *APS Division of Fluid Dynamics Meeting Abstracts*, APS Meeting Abstracts, page A31.008, November 2017.

[36] Ryan King, Oliver Hennigh, Arvind Mohan, and Michael Chertkov. From deep to physics-informed learning of turbulence: Diagnostics, 2018.

[37] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[38] A. N. Kolmogorov. "The Local Structure of Turbulence in Incompressible Viscous Fluid for Very Large Reynolds Numbers". *Proceedings: Mathematical and Physical Sciences*, 434(1890):9–13, 1991.

[39] Arthur G. Kravchenko and Parviz Moin. Numerical studies of flow over a circular cylinder at red=3900. *Physics of Fluids*, 12(2):403–417, 2000.

[40] Nikolay V. Kuznetsov, Timur N. Mokaev, Olga A. Kuznetsova, and Elena V. Kudryashova. The lorenz system: hidden boundary of practical stability and the lyapunov dimension. *Nonlinear Dynamics*, 102:713–732, 2020.

[41] Erik Linder-Norén. Pytorch-gan. https://github.com/eriklindernoren/PyTorch-GAN. Accessed: 12.11.2021.

[42] Julia Ling, Andrew Kurzawski, and Jeremy Templeton. Reynolds averaged turbulence modelling using deep neural networks with embedded invariance. *Journal of Fluid Mechanics*, 807:155–166, 2016.

[43] Bo Liu, Jiupeng Tang, Haibo Huang, and Xi-Yun Lu. Deep learning methods for super-resolution reconstruction of turbulent flows. *Physics of Fluids*, 32(2):025105, 2020.

[44] Edward N Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.

[45] Wei Ma and Jun Lu. An equivalence of fully connected layer and convolutional layer, 2017.

[46] Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.

[47] J. v. Neumann. Proof of the quasi-ergodic hypothesis. *Proceedings of the National Academy of Sciences*, 18(1):70–82, 1932.

[48] C. Norberg. An experimental investigation of the flow around a circular cylinder: influence of aspect ratio. *Journal of Fluid Mechanics*, 258:287–316, 1994.

[49] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning, 2018.

[50] Lawrence Ong and James M. Wallace. The velocity field of the turbulent very near wake of a circular cylinder. *Experiments in Fluids*, 20:441–453, 1996.

[51] Zhaoqing Pan, Weijie Yu, Xiaokai Yi, Asifullah Khan, Feng Yuan, and Yuhui Zheng. Recent progress on generative adversarial networks (gans): A survey. *IEEE Access*, 7:36322–36333, 2019.

[52] Eric J. Parish and Karthik Duraisamy. A paradigm for data-driven predictive modeling using field inversion and machine learning. *Journal of Computational Physics*, 305:758–774, 2016.

[53] Philippe Parnaudeau, Johan Carlier, Dominique Heitz, and Eric Lamballais. Experimental and numerical studies of the flow over a circular cylinder at reynolds number 3900. *Physics of Fluids*, 20(8):085101, 2008.

[54] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[55] Ole Peters. The ergodicity problem in economics. *Nature Physics*, 15:1216–1221, 12 2019.

[56] Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016.

[57] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[58] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[59] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[60] Anand Singh and Karthik Duraisamy. Using field inversion to quantify functional errors in turbulence closures. *Physics of Fluids*, 28:045110, 04 2016.

[61] Anand Singh, Shivaji Medida, and Karthik Duraisamy. Machine-learning-augmented predictive modeling of turbulent separated flows over airfoils. *AIAA Journal*, 55, 08 2016.

[62] Akshay Subramaniam, Man Long Wong, Raunak D Borker, Sravya Nimma-gadda, and Sanjiva K Lele. Turbulence enrichment using physics-informed generative adversarial networks, 2020.

[63] Warwick Tucker. The lorenz attractor exists. *Comptes Rendus de l'Académie des Sciences - Series I - Mathematics*, 328(12):1197–1202, 1999.

[64] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic ma-nipulation with conditional gans. In *2018 IEEE/CVF Conference on Com-puter Vision and Pattern Recognition*, pages 8798–8807, 2018.

[65] Xintao Wang, Ke Yu, Shixiang Wu, Jinjin Gu, Yihao Liu, Chao Dong, Yu Qiao, and Chen Change Loy. Esrgan: Enhanced super-resolution gen-erative adversarial networks. In *Proceedings of the European conference on computer vision (ECCV) workshops*, 2018.

[66] J. Weatheritt and R.D. Sandberg. The development of algebraic stress models using a novel evolutionary algorithm. *International Journal of Heat and Fluid Flow*, 68:298–318, 2017.

[67] Jack Weatheritt and Richard Sandberg. A novel evolutionary algorithm applied to algebraic modifications of the rans stress–strain relationship. *Journal of Computational Physics*, 325:22–37, 2016.

[68] Maximilian Werhahn, You Xie, Mengyu Chu, and Nils Thuerey. A multi-pass gan for fluid flow super-resolution. *Proceedings of the ACM on Com-puter Graphics and Interactive Techniques*, 2(2):1–21, Jul 2019.

[69] Benjamin Winhart, Martin Sinkwitz, Andreas Schramm, Pascal Post, and Francesca di Mare. Large eddy simulation of periodic wake impact on boundary layer transition mechanisms on a highly loaded low-pressure tur-bine blade. In *Turbo Expo: Power for Land, Sea, and Air*, volume 84102, page V02ET41A013. American Society of Mechanical Engineers, 2020.

[70] You Xie, Eric Franz, Mengyu Chu, and Nils Thuerey. Data-driven synthesis of smoke flows with cnn-based feature descriptors. *ACM Transactions on Graphics*, 36(4):1–14, Jul 2017.

[71] Muchen Yang and Zhixiang Xiao. Improving the $k - \omega - \gamma - ar$ transition model by the field inversion and machine learning framework. *Physics of Fluids*, 32, 06 2020.

[72] Dmitry Yarotsky. Error bounds for approximations with deep relu networks. *Neural Networks*, 94:103–114, 2017.

[73] Jincheng Zhang and Song Fu. An efficient bayesian uncertainty quantification approach with application to $k-\omega-\gamma$ transition modeling. *Computers & Fluids*, 161:211–224, 2018.

[74] Weiwei Zhang, Linyang Zhu, Jiaqing Kou, and Yilang Liu. Machine learning methods for turbulence modeling in subsonic flows over airfoils, 06 2018.

[75] Yaomin Zhao, Harshal D. Akolekar, Jack Weatheritt, Vittorio Michelassi, and Richard D. Sandberg. Rans turbulence model development using cfd-driven machine learning. *Journal of Computational Physics*, 411:109413, 2020.

[76] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.