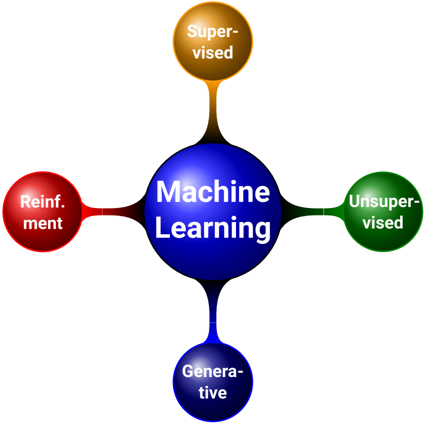
Generative AI and Mechanical Engineering

Hanno Gottschalk (Math+, TU Berlin)

CASES, September 10 2024

The Task of Generative Learning

- **Supervised learning:** Learn a unknown conditional distribution $f(y|x)$ from pairs of data (Y_i, X_i).
- **Unsupervised learning:** Learn a unknown distribution $f(x)$ from data X_j.
- **Reinforcement learning:** Learn a policy from rewards
- **Generative learning:** Learn how to generate new samples from a unknown distribution $f(x)$ on the basis of data $X_j \sim f(x)$.

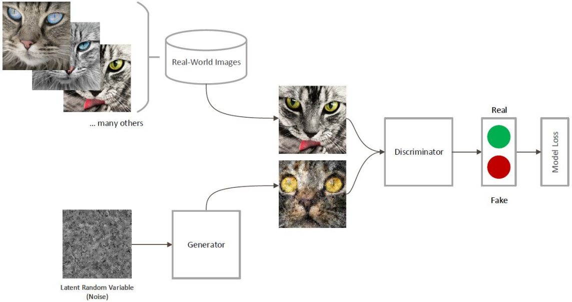


The Challenge of Modern Generative Learning

From NVIDEA Style GAN, `thispersondoesnotexist.com`, Karras, Laine, Alia 2018

- **Challenge:** $X_j \in \mathbb{R}^d$ is high dimensional, $d \sim 10^6$.
- X_j could be images, a text messages, spoken language...
- Sampling from (log-) densities with MCMC is feasible, but need a density

Generative Adversarial Learning



- Min-Max two player game $\hat{\phi} \in \arg\min_{\phi \in \mathcal{H}} \sup_{D \in \mathcal{H}_D} \hat{L}(\phi, D, \{X_j\})$
- With $X_j \sim \mu$, $U_j \sim \lambda^{(d)}$, the empirical loss is defined as

$$\hat{L}(\phi, D, \{X_j\}) = \frac{1}{2n} \sum_{j=1}^{n} [\log(D(X_j)) + \log(1 - D(\phi(U_j))]$$

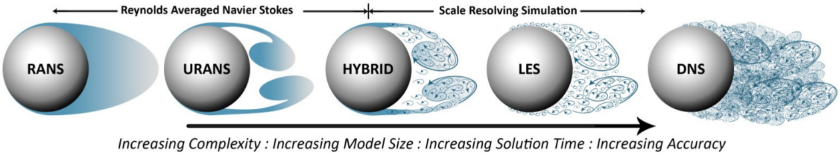
Turbulence in Fluid Dynamics

- Turbulent flow is **chaotic** by nature
- Computational Fluid Dynamics (CFD): Simulation of turbulences by numerically solving Navier-Stokes equations:

$$
\begin{aligned}
\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) &= 0 \\
\frac{\partial (\rho u)}{\partial t} + \nabla \cdot [\rho u \otimes u] &= -\nabla p + \nabla \cdot \tau + \rho f \\
\frac{\partial (\rho e)}{\partial t} + \nabla \cdot ((\rho e + p) u) &= \nabla \cdot (\tau \cdot u) + \rho f \cdot u + \nabla \cdot \dot{q} + r
\end{aligned}
$$

- But: Turbulent flow develops ever smaller and faster structures which are hard to resolve numerically

Simulation of Turbulence



Comparison of turbulence modeling approaches. Source: (J. Hart 2016)

- Modeling turbulences challenging in practice but highly technical relevant
- LES/DNS: Simulation of large or even all scales of turbulence but: Enormous computational costs
- RANS/URANS: Struggle in simulation of fine details in vortex flow but: Computational costs are sustainable $\Rightarrow$ mostly used in industry

Ergodicity: What Does "Chaotic" Actually Mean?

- Vortices always look different but in the long run their statistical properties are determined **(ergodicty)**:

$$\lim_{T\to\infty} \frac{1}{T} \int_0^T f \circ \varphi_t(x_0)\,\mathrm{d}t = \int_\Omega f(x)\,\mathrm{d}\mu(x)\ \forall x_0 \in \Omega. \tag{1}$$

  $\Rightarrow$ The time average of a dynamical system equates with the ensemble average of its invariant measure.
- Probability measure μ on the flow configurations x encodes the statistical properties of the chaotic flow/dynamics $\varphi_t(x_0)$
- Goal: Sampling from the **unknown** measure $\mu \Rightarrow$ Can we **learn** μ **from data**?

Modeling Turbulence by GAN

Our contribution:

- GANs as another possibility to model turbulence
- Proof that GANs do converge for ergodic learning problems
- Generation of images that are as good as the output of the LES while requiring significantly less computation effort
- Numerical demonstration of generalization over changes in geometry
- Proof that physical quantities that characterize turbulence converge

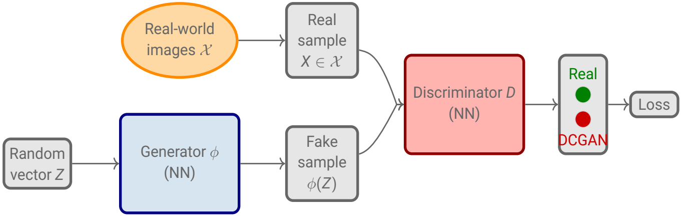
Lorenz Attractor

- Less complex **deterministic** ergodic system given by the system of ordinary differential equations

$$\frac{dx}{dt} = \sigma(y - x)$$
$$\frac{dy}{dt} = x(\rho - z) - y$$
$$\frac{dz}{dt} = xy - \beta z$$

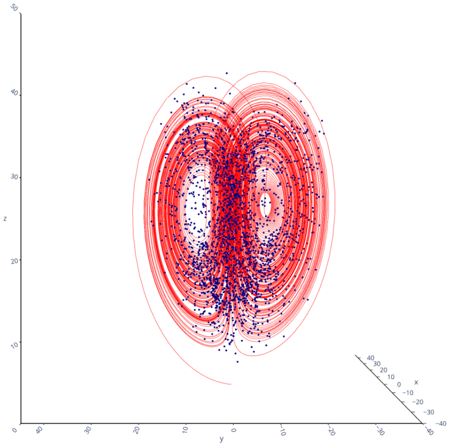
with $\sigma = 10$, $\beta = \frac{8}{3}$ and $\rho = 28$.

Lorenz Attractor - Training



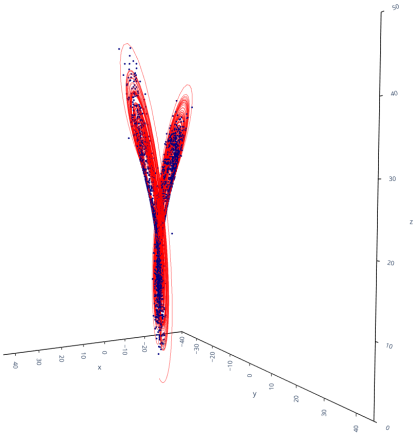
- Vanilla GAN for 200 000 epochs with batch size 20 000
- Trajectory started per epoch from randomly sampled initial point (x_0, y_0, z_0)
- Gaussian noise added to the network of discriminator and its real input data
- ϕ and D (deep) fully connected neural networks
- $U \in \mathbb{R}^{100}, U \sim U(0, 1)$

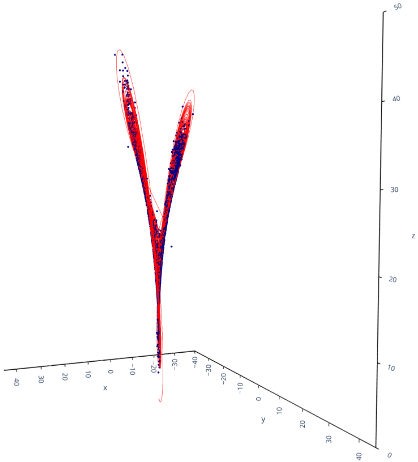
Lorenz Attractor - Results I



Trajectory consisting of 20 000 data points (red) and 3 000 synthesized data points (blue)

Lorenz Attractor - Results III





Rotated perspective from the trajectory of real data points (red) and the synthesized data points (blue).

Karman Vortex Street - Data

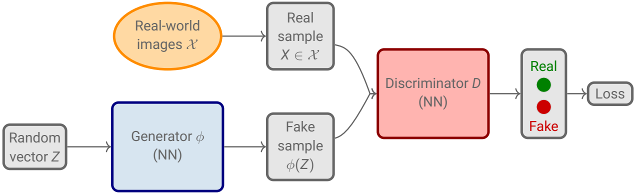
- Dataset: $5,000$ images produced by LES
- Images: $w \times h = 1,000 \times 600$

Example of the dataset.

Karman Vortex Street - Training and Inference I

Training

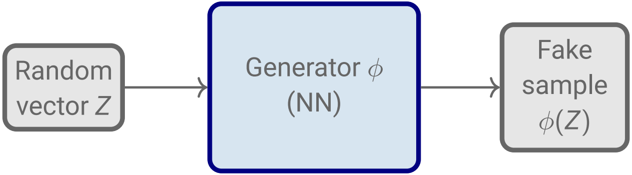
- GAN frameworks:
  - Vanilla GAN
  - Wasserstein GAN
  - Deep convolutional GAN
- Epochs: 200
- Batch size: 20
- Input:
  - $5,000$ LES images
  - Image size: $k \times k$, $k \in \{64, 128, 256, \mathbf{512}\}$
  - Noise vector $Z \in \mathbb{R}^{100}, Z \sim N(0,1)$



Karman Vortex Street - Training and Inference II

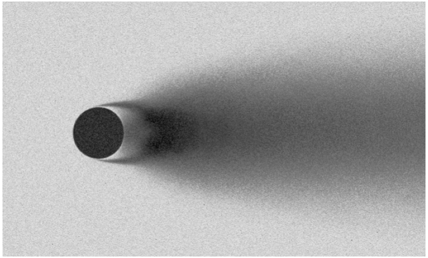
Inference

- Trained generator ϕ
- Input: Noise vector $Z \in \mathbb{R}^{100}, Z \sim N(0,1)$



Karman Vortex Street - Vanilla GAN

LES image

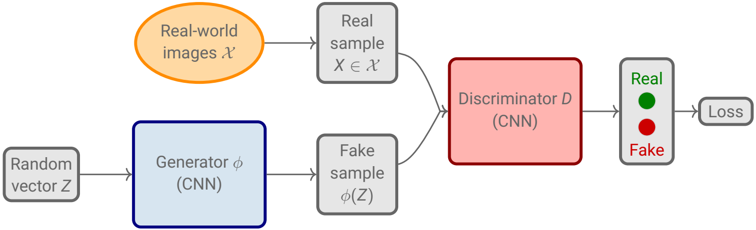


Fake image

Experiment settings:

- Architecture of ϕ and D: Fully connected NN with 5 layers
- Optimizer: Adam
- Learning rate: $2 \cdot 10^{-4}$

Deep Convolutional GAN (DCGAN)



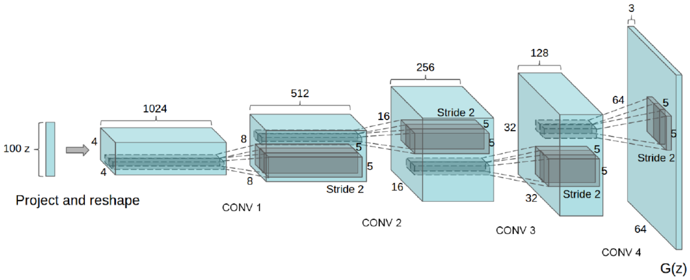
Architecture of DCGAN.

- Generator and discriminator are convolutional neural networks (CNNs)
- CNNs expecially successful and applicable in field of image processing

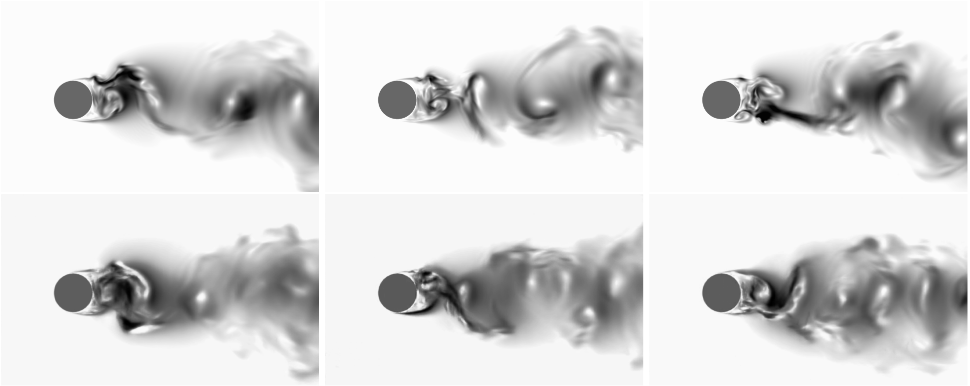
Deep Convolutional Generative Adversarial Network II

Guidelines to follow for **stable training** at **higher resolution** and **deeper architectures**:

- Stability: Apply batch normalization on the output layer of ϕ and the input layer of D
- Deeper architectures: Avoid fully-connected layers on top of convolutional features
- Higher resolution modeling: Leaky Rectified Linear Unit (ReLu) activation function for D
- ϕ and D learn own spatial up- or downsampling by by replacing deterministic spatial pooling layers with (fractional-) strided atrous convolutions (Radfort et al 2016)



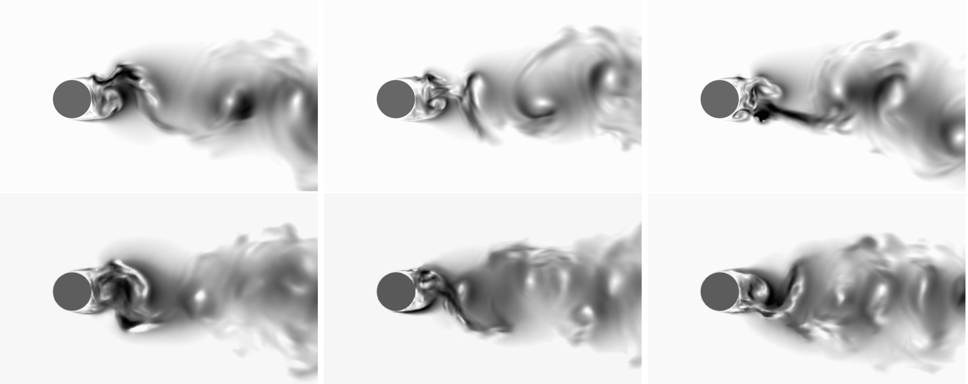
Karman Vortex Street - Inference results after $2,000$ Epochs



Karman Vortex Street - Inference results after $2,000$ Epochs

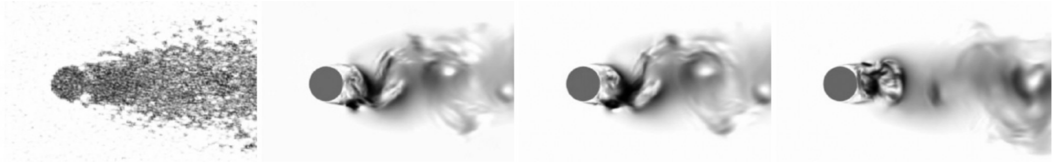
LES

Fake



Comparison of LES (top) and fake images (bottom) produced by generator ϕ trained for 2 000 epochs.

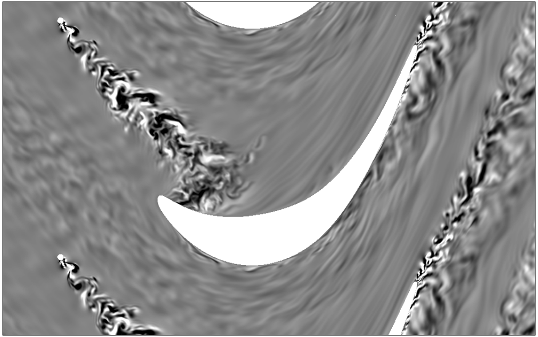
Sampling Results for the Karman Street



- Results of the DCGAN after 1, 500, 1000, 1500 and 2000 Epochs of training

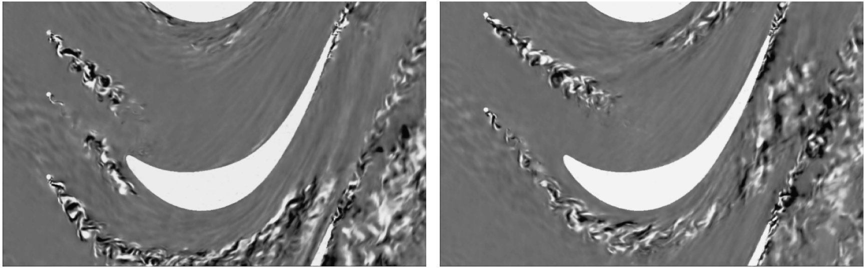
LPT Stator Under Periodic Wake Impact (LPT Stator) - Data

- Dataset: $2,250$ images produced by LES
- Images: $w \times h = 1,000 \times 625$



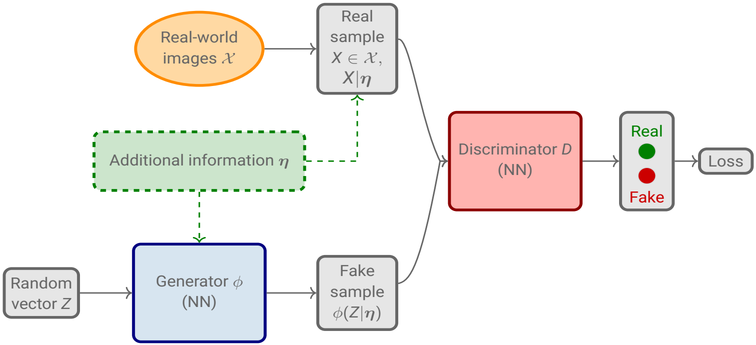
Example of the dataset.

LPT Stator - DCGAN: Results



Examples of images generated by ϕ trained for 2 000 epochs with 2 250 images.

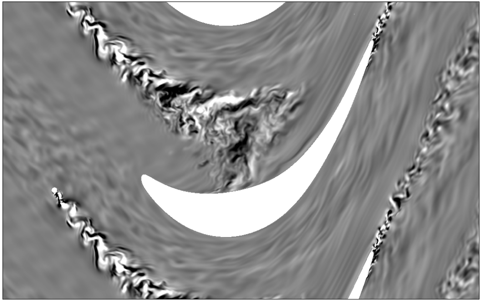
Conditional GAN (cGAN)



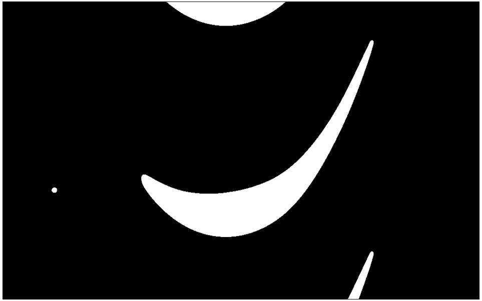
- Take control over the data production process by conditioning GAN framework
- Extension of loss function:

$$\mathcal{L}_{cond}(D,\phi) = \mathbb{E}_{\substack{X\sim\mu \\ \eta\sim\nu}}[\log(D(X|\eta))] + \mathbb{E}_{\substack{Z\sim\lambda \\ \eta\sim\nu}}[\log(1 - D(\phi(Z|\eta)))] \tag{2}$$

LPT Stator - Conditional Training



LES image



Binary segmentation mask

High-Resolution Image Synthesis with Conditional GANs

- Special form of cGAN
- Generation of **high-resolution photo-realistic** images by **conditioning** the input of the adversarial network on the corresponding **semantic label maps**
- We use the NVIDEA `pix2pixHD` cGAN architecture, whose optimization problem is given as

$$\min_{\phi} \max_{D} \mathcal{L}_{cond}(D, \phi) \tag{3}$$

High-Resolution Image Synthesis with Conditional GANs

- Introduction of **three innovations** for **improvement** of **photorealism** and **resolution** of the synthesized images:
  1. Coarse-to-fine generator: Decomposition of the generator into two sub-networks $\Rightarrow \phi = \{\phi_1, \phi_2\}$
  2. Multi-scale discriminators: Three discriminators D_1, D_2 and D_3 with same architecture but operating on different scales $\Rightarrow$ Modification of loss function:

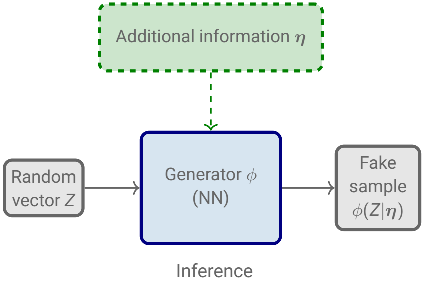
$$\min_{\phi} \max_{D_1,D_2,D_3} \sum_{i=1}^{3} \mathcal{L}_{cond}(\phi, D_i) \tag{4}$$

  3. Feature matching loss $\mathcal{L}_{FM}$: Stabilize training
- Loss function in total:

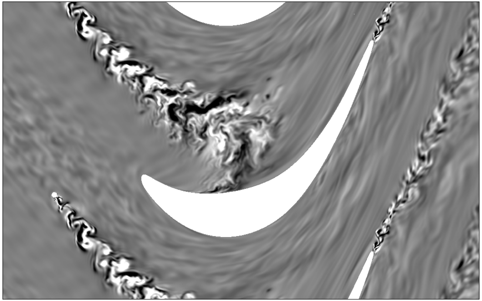
$$\min_{\phi} \left[\left(\max_{D_1,D_2,D_3} \sum_{i=1}^{3} \mathcal{L}_{cond}(\phi, D_i) \right) + \gamma \sum_{i=1}^{3} \mathcal{L}_{FM}(\phi, D_i) \right] \tag{5}$$

LPT-Stator - Experiment Settings

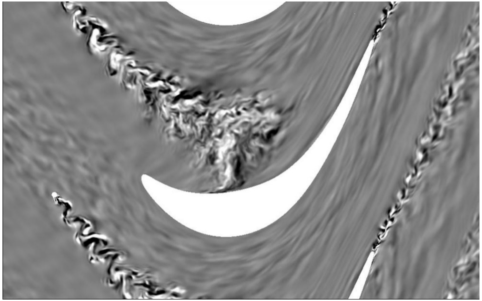
- Architecture: Adopted from original authors with small changes to avoid artifacts
- Epochs: 200
- Batch size: 10
- Input: $2,000$ LES images, noise vector and masks of size $k \times k' = 992 \times 624$
- Optimizer: Adam
- Learning rate: $2 \cdot 10^{-4}$



LPT Stator - Inference Results I

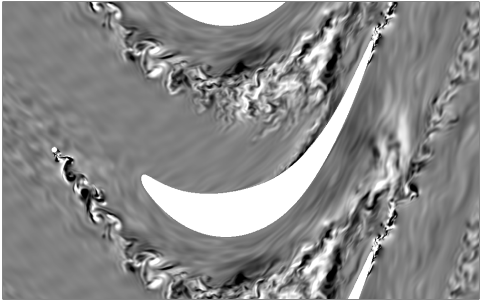


LES image

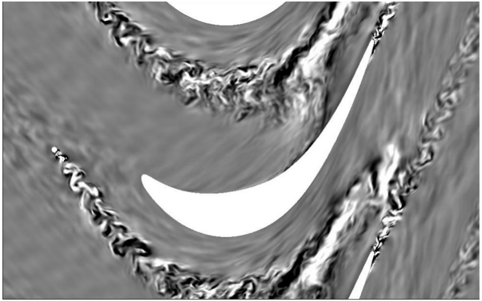


Fake image

LPT Stator - Results II

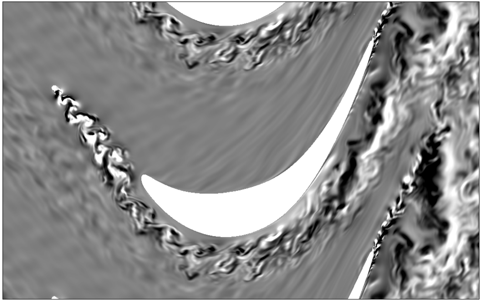


LES image

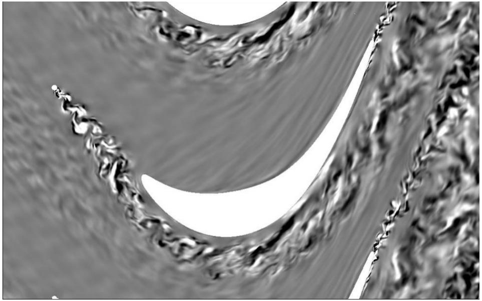


Fake image

LPT Stator - Results III

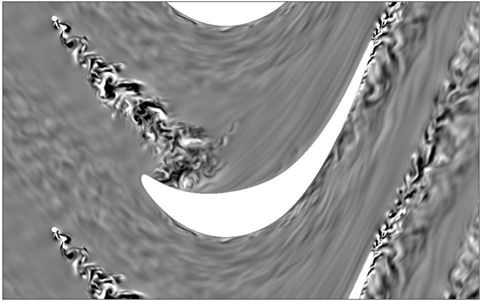


LES image

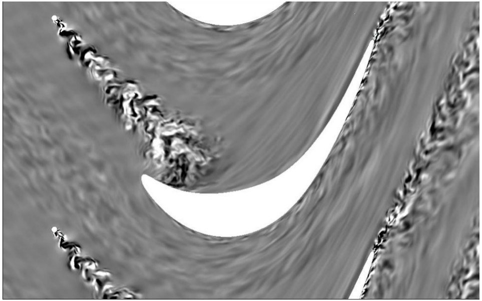


Fake image

LPT Stator - Results IV



LES image

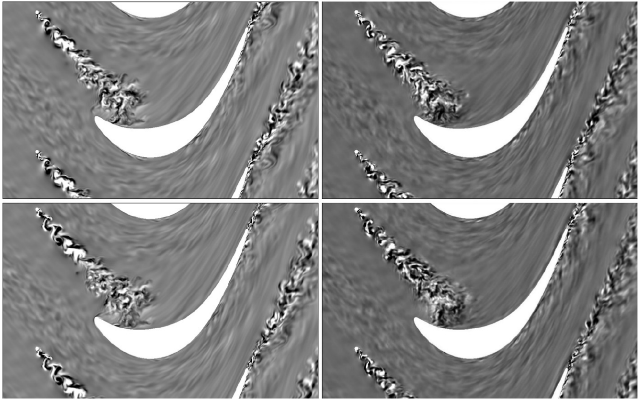


Fake image

Computational Costs

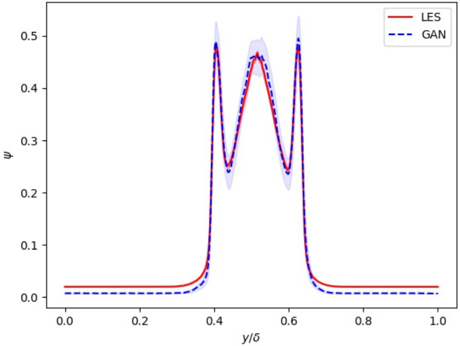
- **LES**:
  - Performed on 560 CPU cores of the CPU type Intel Xeon "Skylake" Gold 6132 @2.6 GHz
  - Karman vortex street (5 000 images): 72 core weeks $\hat{=}$ 1 days
  - LPT Stator (2250 images): 640 core weeks $\hat{=}$ 8 days
- **GAN-Training**:
  - Performed on GPU of type Quadro RTX 8000 with 48 GB
  - Karman vortex street (DCGAN, 2 000 epochs): 1.5 min/epoch
  - LPT Stator (pix2pixHD, 200 epochs): 17 min/epoch
- **GAN-Inference**:
  - Performed on GPU of type Quadro RTX 8000 with 48 GB
  - Karman vortex street (DCGAN): 0.001 sec/image $\Rightarrow$ 5 seconds
  - LPT Stator (pix2pixHD): 0.01 sec/image $\Rightarrow$ 22.5 seconds

Genaralization for Unseen Geometry Configurations



- Repeat experiment with 5% of the images ommited around specific wake position
- During inference, position the wake at exactly the middle of that position
- results demonstrates generatization capability over unseen changes in geometry

Statistical Similarity vs Phsics



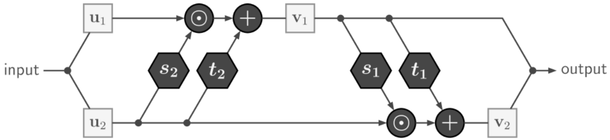
- Measure strength of turbulence (variation of $c = |u|$ as a function of ξ

$$\mathrm{Var}[c(\xi)] = \lim_{T\to\infty} \frac{1}{T} \int_0^\infty |c(\xi,t) - \bar{c}(\xi)|^2 \, \mathrm{d}t$$

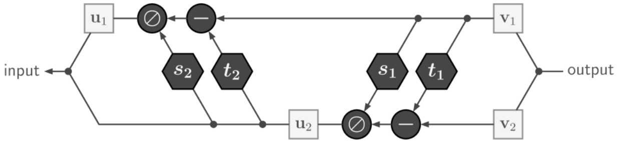
Invertible Neural Networks - INN

Forward Direction ():

Ardizonne et al., 2018

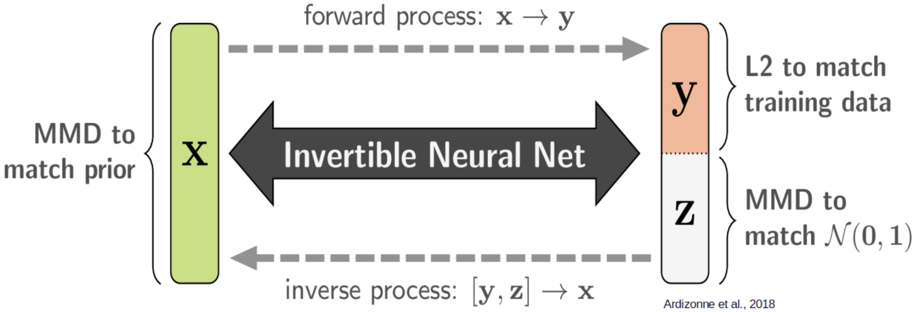


Backward Direction ():



- Generative learning relates to inverse design

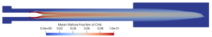
Dimension Matching

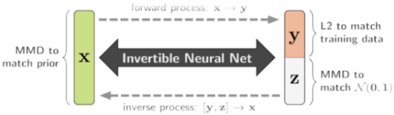


Ardizonne et al., 2018

- In Design, usually there is a dimensioal mismatch between many input- and a few output parameters
- This can be used to generate design alternatives

Case study for a Multi Fuel Burner





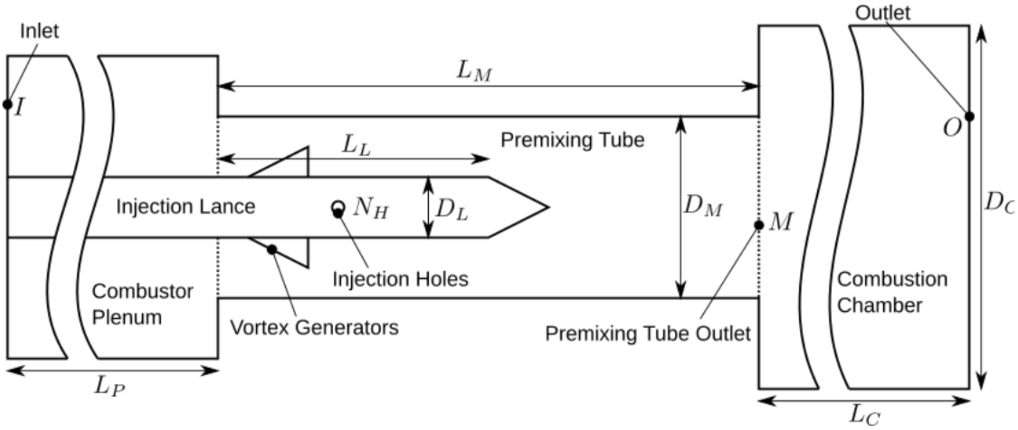
Desired performance values (labels)

generative design tool

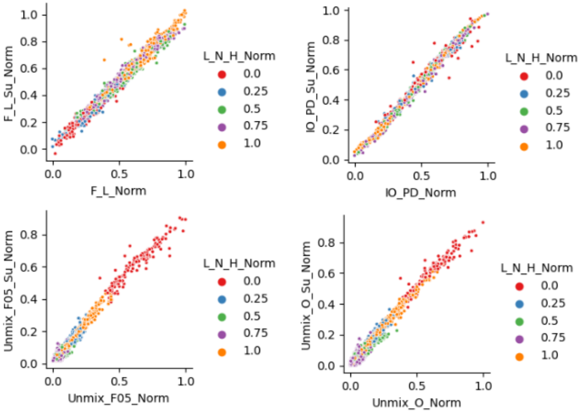
Multiple alternatives of geometry parameters yielding performance values

MATH+

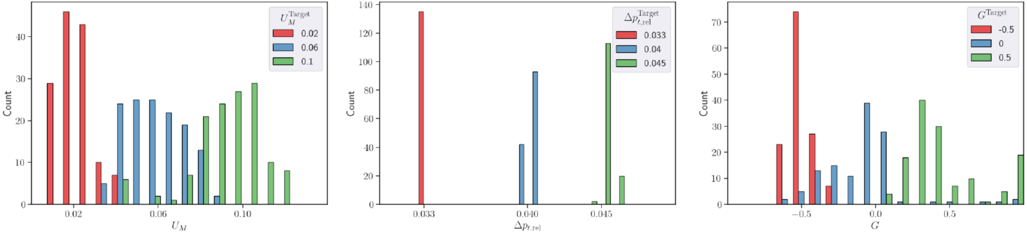
Design Parameters



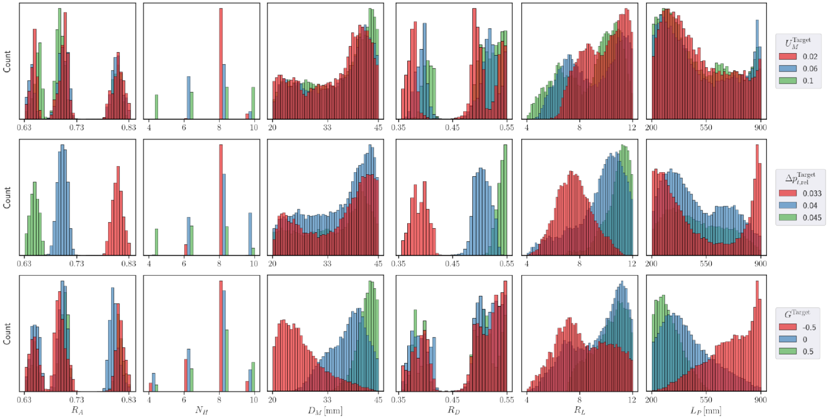
Trained INN - Forward Mode



Trained INN - Backward Mode



Trained INN - Design Diversity



Summary

- Generative learning can be useful for applications beyond speech generation and computer vision
- Further data models, like deterministic chaos, can be combined with generative adversarial learning.
- Considerable speedup can be obtained during inference by GAN
- INN can help in inverse design

1. H. Asatryan, G., M. Lippert, M. Rottmann, A Convenient Infinite Dimensional Framework for Generative Adversarial Learning, Electronic Journal Of Statistics 2023, arXiv:2011.12087
2. A. Mütze, M. Rottmann, G., Semi-supervised domain adaptation with CycleGAN guided by a downstream task loss, VISAPP2023
3. C. Drygala, B. Winhard, F. di Mare, G., Generative Modeling of Turbulence, Physics of Fluids 2021 (editor’s pick)
4. C. Drygala, F. di Mare, G., Generaization capabilities of conditional GAN for turbulent flow, EUROGEN 2023.
5. R. Chan, S. Penquitt, H. Glotschalk, LU-Net: Invertible Neural Networks Based on Matrix Factorization, IJCNN 2023.
6. P. Krüger, H. Gottschalk, B. Werdelmann and W. Krebs, Generative design of a gas turbine combusteor using INN, ASME TurboExpo 2024 and J. of Turmomachinery 2024.