Advanced D. L. for Physics

Exercise 5

Exercise 5 – *GAN Project*

The last exercise is a free project during which you will turn the autoencoder network from Exercise 2 into a network with adversarial training, i.e., Generative Adversarial Network (GAN). GAN is one of the most powerful deep neural network architectures, and the goal of this exercise is to give you first hand experience in this area.

The only requirements for your final version are as follows:

- The networks should contain generator+discriminator trained in an adversarial manner.
- The generator should output a 2D velocity field of resolution 32^2 or larger.

We will give further hints below how to get started and how to extend the network.

Deliverables by July 16th:

- 10 or more generated example velocity fields as PNG images
- An in-class presentation of max. 5 minutes describing your final approach and earlier tests. This presentation will take place on July 16th at 15:15 in room 02.13.010.
- After the presentation, send a zip file with code, the 10 examples, and your presentation as PDF to adl.phys.ex@gmail.com. As usual, submit a zip file with the following naming scheme: "**xx_lastname1_lastname2_lastname3.zip**". The subject of the email should be [ADL Exercise05 GroupXX]. Hand in your file by **July 16th before midnight 23:59** (email arrival time will count)**.**

How to get started

- We recommend that you first set up and train a GAN that takes an array of random values as input and produces a velocity output. You can start with the Ex2 code and follow an example such as this one to turn it into a GAN: https://github.com/aymericdamien/TensorFlow-Examples/blob/master/examples/3_NeuralNetworks/gan.py
- Note that you will need to separate the variables into 2 sets corresponding to the generator and discriminator so that they can be trained individually (without affecting the other network). See the link above (lines 107-115) for examples.
- We highly recommend that you save the generator's output (e.g., as images) in intervals during your training. This is crucial to investigate the training process.
- In addition, keep the balance of D&G in view: check the amount of real and fake samples correctly classified by D.
- For a first network, we recommend that you use ca. 4 conv layers (plus one fully connected one for D) with up to 8 feature maps and a similar size (in terms of weights) for D and G.

- Since you will likely be using deconvolutions for the generator, consider the following article: https://distill.pub/2016/deconv-checkerboard/. We recommend using the resizing, e.g., with max. de-pooling, plus convolution approach described there to prevent the generator from producing checkerboard artifacts.

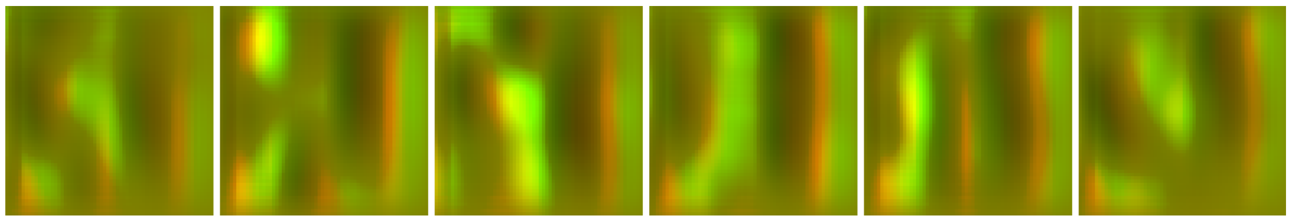
Possible Extensions

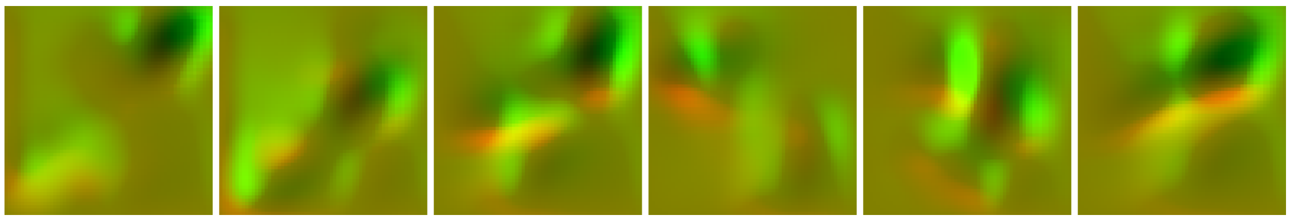
Once you have this basic version working, you can look into these extensions. While they are not mandatory, we will give out a prize for the best GAN project and we will take into account the complexity of the architecture as well as the quality of the generated results.

As improvements to your GAN, consider, e.g.,:

- Making your GAN conditional by providing it with a low-res input instead of a random input for the generator (use at least a 4x smaller data set). If you do this, also show the low-res inputs with your outputs.
- You can experiment with the divergence-free constraint (from the lecture) in the loss function.
- Try other GAN architectures, such as the Wasserstein GAN (https://arxiv.org/pdf/1701.07875.pdf) or BEGAN (https://arxiv.org/pdf/1703.10717.pdf).
- Do *not* try to extend your network to 3D. Typically, this will require huge amounts of data and long training times. Rather try to improve your results in 2D.

Output examples


Several examples from a basic network without extensions.


Examples from an improved network with preconditioning and divergence in the generator loss.