SliceGAN Revisited - Carbon Fibers

J. Ijpma†, R. Jense†, H. Lindstedt‡ and A. Sharma†

† Faculty of Electrical Engineering, Mathematics & Computer Science
‡Faculty of Applied Sciences

SliceGAN

SliceGANs main task is to resolve the incompatibility between the dimensions of the Generator, producing 3D volumes and the Discriminator, classifying 2D images. It does this by adding a slicing step before passing the generated volume on to three discriminators Discriminators. Each of the Discriminators tries to distinguish real and generated slices corresponding to either the x, y or z axis of the material. In case the material is isotropic (meaning it looks the same from all three directions), one Discriminator suffices. A Wasserstein loss is used during training.

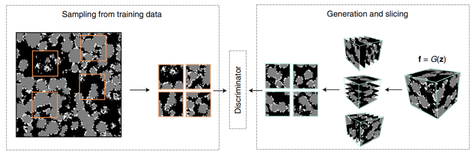

Fig. 1 | SliceGAN training procedure. First, real images are sampled from a 2D training sample. Second, a fake volume **f** is generated and sliced along *x*, *y* and *z*. This yields a compatible pair of datasets, which can both be fed to a 2D discriminator.

Fig. 1: SliceGAN training procedure from [1]

Data & Preprocessing

Although SliceGAN can handle many types o structures, our project revolved around carbon fibers. The data used in this experiment is a glass fibre-epoxy composite specimen retrieved from [2], shown in figure 2. The data is obtained from spectroscopy, and the darkness of the grey scale pixels correspond to the density in the specimen. To be able to work with this data we cropped the images, and since the 3D structure of the fibres is not isotropic we concatenated the cropped slices.

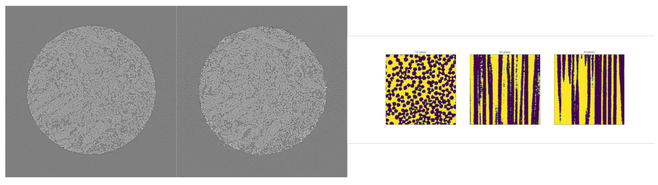

Fig. 2: Data sample

Next, we turned the data into a binary format and cleaned up some artifacts. One of the issues we faced while pre-processing is that- while deciding the threshold to separate the two channels present in our image (fiber and medium), we fail to find a value that accurately does this without any overlap or noise. This becomes a huge issue for us while running CircleNet as the circular structure of the fibers isn't extracted ideally, for it to be recognized and mimicked. Due to this, we apply water-shedding to segment these fiber sub-sections and ensure that these circular sub-sections are separated so that they may be identified and recreated more accurately.

PINN & CircleNet

An interesting field within Deep Learning is *Physically Informed Neural Networks*, or PINN. These networks incorporate a special term in their loss function, with the aim of better capturing some underlying laws of the problem aiming to model some physical behaviour.
We note that an issue with SliceGan producing carbon structures is that the circularity of fibers in the generated volume's cross-sections is not perfect. Rather, they are a bit elliptic and/or distorted compared to our original data.
For implementing this feature in our network, we add a custom loss term in the generator training, aptly dubbed- *Circularity Loss*. This loss term ensures that the number of circles with $circularity > circularitythreshold$ for each sub-image, is of a similar order when comparing our real and generated data. This loss punishes the generator if the criterion is not fulfilled on the generated sub-image by using MSE as its loss.
We call this circle recognising entity *CircleNet*, a network trained only to recognise circles. To ensure that CircleNet is tailored exclusively for our situation, we train it on the real carbon data (which we aim to recreate) by computing ground truth labels with the help of *opencv* libraries.
One issue we faced while implementing this is that the real data our network is trained on also contains a lot of artefacts and distortions that affects our learning capability for modular nodes with high circularity. We apply water-shedding to solve this as mentioned in Data & Preprocessing Section. A sub-image (without water-shedding) which our CircleNet was trained on and classified can be seen in fig 3.

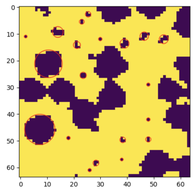

Fig. 3: Cross section of 3D volume along the x-direction as being seen by the CircleNet during training (without water-shedding)

Noise Seed Distributions

One of the things we have to consider when working with a GAN is that it needs a random seed to be able to generate different volumes when we run it.
In the original paper, these seeds are drawn from every statisticians best friend: $N(0, 1)$. We wanted to experiment and see whether it would make a difference to use some other distributions. Some of the results can be seen below. Clearly, Noise from a Cauchy Distribution works much better than that from an exponential distribution.

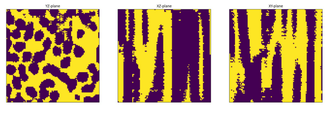

Fig. 4: Cross section of 3D volume generated with Cauchy Noise

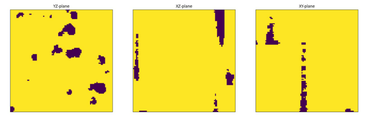

Fig. 5: Cross section of 3D volume generated with exponential Noise

Hyper-parameter tuning

SliceGAN has many hyperparameters that could be configured. For this project, we looked into the following parameters:

- Beta1 for the Adam Optimizer
- Beta2 for the Adam Optimizer
- Different noise distributions fed to the Generator

The Discriminator losses for the different beta1 and beta2 values can be found in figure 6.

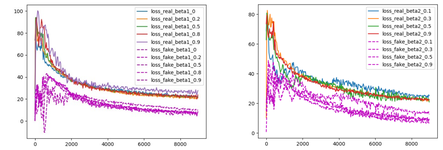

Fig. 6: Discriminator losses for real and fake data for different beta1 and beta2 values

Conclusion

In this project, we carry out a lot of tasks to evaluate and extend on the given SliceGAN model. We first pre-process the data in various ways for the different modules. Testing was conducted on both grayscale and binarized versions of the real data, however, due to time and space complexity issues, binarized was preferred. There is a minute increase in the quality of structures produced after adding the CircleNet but a lot of its hyper-parameters could be varied and tweaked to better assess its impact. As clearly seen, the noise distribution makes a substantial difference as well. The noteworthy distributions were Cauchy, which was the most similar to our real image after Normal, and Uniform, the fibers for which seem to have molded together in the generation. We also test our framework by tweaking different GAN hyper-parameters. This is important to carry out as some strange empirically-driven choices have been made by the original authors while setting these values.

References

[1] Steve Kench and Samuel J Cooper. "Generating 3D structures from a 2D slice with GAN-