

Design and Evaluation of Generative Deep Learning Models for Electrochemical Device Microstructure Image Augmentation

by

Daulet Tazhbagambetov

Submitted to the Department of Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

at the

NAZARBAYEV UNIVERSITY

April 2026

© Nazarbayev University 2026. All rights reserved.

Author
Department of Computer Science
11.04.2026

Certified by.....
Iliyas Tursynbek
Dr.
Thesis Supervisor

Accepted by
Elizabeth Arkhangelsky
Dean, School of Engineering and Digital Sciences

Design and Evaluation of Generative Deep Learning Models for Electrochemical Device Microstructure Image Augmentation

by

Daulet Tazhbagambetov

Submitted to the Department of Computer Science
on 11.04.2026, in partial fulfillment of the
requirements for the degree of
Master of Science in Computer Science

Abstract

The issue of making more efficient batteries is ever present in today's world. Researchers often have a problem with having a variety of samples to study the inner structure of the batteries. At the same time, the development of generative AI became rapid. It was important to consider the integration of such systems into the energy sector to use its benefits by having opportunities of saving material production and increase the learning efficiency. An effective tool turned out to be Generative Adversarial Network (GAN) that is able to mimic the physical structure of the models to recreate the CT slice scans of the model, to produce an efficient output for further battery improvement. The main concept is to train the model based on the real segmented CT scan dataset, to then generate a 3D model that looks accurate both visually and in terms of physical and chemical properties. The correctness will be evaluated with Tau factor that is responsible for tortuosity calculation, and generated model will be vigorously compared with the original dataset on a pixel level.

Thesis Supervisor: Iliyas Tursynbek
Title: Dr.

Acknowledgments

Contents

1	Introduction	11
1.1	Motivation	13
1.2	Main Objectives	15
2	Related works	17
2.1	Groundwork for GAN	17
2.2	Further GAN development	18
2.3	GAN development challenges and issues to note	19
2.4	Further important GAN aspects for consideration	21
3	Methodology	25
3.1	Dataset	25
3.2	System Configuration and Strategy	28
3.3	System Architecture of C-GAN	28
3.3.1	Data Preprocessing and Data Loading	29
3.3.2	Data Models	30
3.3.3	Model Training	34
3.3.4	Image Generation and Simulation of Physical Properties	35
3.4	Overview	36
4	Results	43
4.1	Visual Assessment	44
4.2	Physical Properties	46

4.3	Model Assessment	48
5	Conclusion	55
5.1	Addressing Key Objectives	55
5.2	Future Work	56

List of Figures

1-1	Energy consumption levels [1]	12
1-2	An example of GAN architecture [2]	14
1-3	Dataset that is used for this work obtained from a paper by Tang et al. [3]	15
2-1	A way to compare the readings of Generated and Real data [4] . . .	19
2-2	Predicted Values for GAN [5]	20
2-3	Example of generated 3D samples [5]	21
2-4	Comparison of various GAN images generated [5]	22
3-1	Dataset structure [3]	26
3-2	Architecture of the system	37
3-3	Overview of the Architecture	38
3-4	Training Cycle	41
4-1	Generated Dataset Morphology	43
4-2	The elements change between slices	44
4-3	Carbon fibers extracted and vertical and horizontal view of the generated sample	45
4-4	Physical properties of the samples	47
4-5	Comparison of layers	51
4-6	L2 Norm Across Layers	51
4-7	Sparsity of the Data	52
4-8	Feature Normalization Trajectory	52

4-9	Recreation from Ground Truth	53
4-10	Residual Map and Phase Attribution	53

List of Tables

3.1	Preprocessing Workflow for Experimental Microstructure Tomography.	29
3.2	Summarized Architecture of the RefinerGenerator	32
3.3	Summarized Architecture of the Discriminator	33
3.4	Kinetic and Morphological Synthesis Parameters.	36
3.5	Multi-Objective Loss Functions for Refiner GAN Training.	40
4.1	3D Volume Phase Composition (Slices 80–150)	46
4.2	Structural Analysis of the Generator: Layer Magnitude and Sparsity.	48
4.3	Evolution of Feature Complexity via Batch Normalization Variance. .	48
4.4	Distribution of Trainable Parameters across Functional Model Groups.	49
4.5	Weight Statistics Summary for Generator Checkpoint	50
4.6	Slice-wise Evaluation Metrics with Global Summary	51

Chapter 1

Introduction

The problem of efficient energy production is ever present in the technological world. Historically, the problem of effective usage of energy sources was primarily handled with fossil fuels, which is known to have severe environmental impact. The examples include ozone layer thinning and increase of pollution level in air. This also produces consequential problems with health and ecological disasters. Several approaches were proposed for mitigation. The primary ones were to have renewable energy source. However, it is known that those sources are not efficient enough to fully replace the traditional fossil fuel-based energy consumption. It is mainly due to the logistics. For example, the water system of a dam requires specific terrain and water source that is not readily available everywhere and the power that is derived from such single source is not able to provide for the whole country, as it operates mainly locally. Another example is wind energy, which is also very weather dependent. Similarly, solar power requires too much space being dedicated to solar panels, and it is not ready to do the work of conserving electrical energy since it is also weather dependent.

All those approaches have strong potential, but ultimately need refinement and logistical improvement. However, the proposed method in this article is mainly dedicated to the development of Lithium Ion batteries to make them more efficient.

This has a broader impact rather than giving a momentary solution to all energy source problems. Hydrogen was proposed as an efficient solution to this problem, however, it did not gain mass application to broader energy sector. It has enough

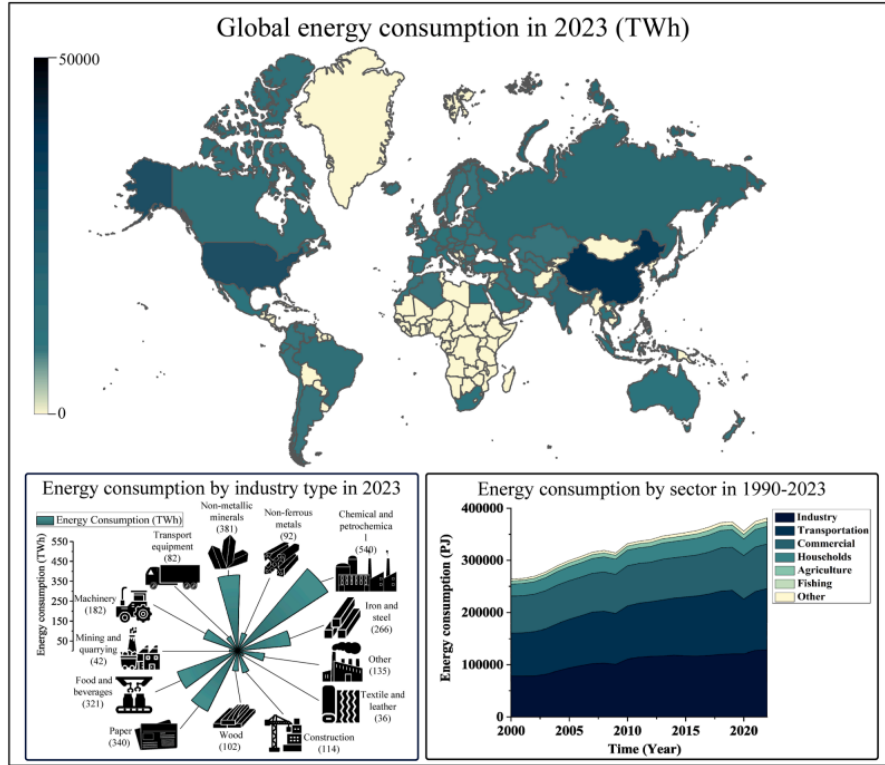


Figure 1-1: Energy consumption levels [1]

density for a more advantageous production, nevertheless, it did not have a lasting effect. It was still partly implemented by gasification methods that was found in natural reserves of hydrogen, which allowed to have more enrichment, hence having gas as another fuel source. Hydrogen technologies remain one of the continued developments in energy studies.

On this occasion, fuel cells gained attention due to its potential for efficient energy production. It is mainly based on chemical reactions that produce energy that also have no environmental impact. Fossil fuels are known to have immediate effects like carbon emissions and pollution, but fuel cells operate on their inner structure that would allow for high energy output. To be specific, PEMFCs (Proton Exchange Membrane Fuel Cells) currently among the research trends deeply tied to the Computer Science topic, specifically with Machine Learning and Generative AI. They are known to be highly enduring and outside pressure of weather or many of physical influence make the damage more difficult to do, which makes them more sustain-

able. Nevertheless, it is known that they have their structure able to change based on more extreme temperature changes which also could be used for energy advantage to change the structure for more efficiency and customization.

PEMFCs are particularly useful for transportation and non-complicated infrastructure that is similar to the regular infrastructure for usual fuels. There are some drawbacks such as material cost which is a significant problem for mass distribution. A link between computer science and fuel cell chemical subject is usage of Generative AI to address the problem of mass production difficulties. Scientists will need more cells for samples to work on development and improvement for greater optimization. For that reason, this work is going to address the problem from the computer science-centered approach.

This thesis is studying a method of integration of generative AI into the battery development. The main aim is to train the system based on the CT images of the slices, and to find a GAN-based training that will implicitly learn the behavior visually, and hence will simulate the physical capabilities. The physical part is going to be assessed with tools like tau factor for tortuosity and general behavior of visible continuity between the slices of the generated 3D dataset.

1.1 Motivation

In recent years, Generative AI and Computer Vision has become a significant part of regular life. It has made contributions to numerous sectors of technology. Chemical sector is no exception. As Fuel Cell is a relatively new topic, it is proposed as a promising improvement to existing battery inner microstructure. It all comes with challenges of difficulties to point out the anisotropic qualities of the inner microstructure. It is a challenge to represent them within the regular computer vision code. Several visual qualities must be adapted based on the dataset. The existing models so far were mostly concentrating on 2D recreation of single slice which is a good starting point, but doesn't fully cover significant 3D slices. With more advancements in terms of computational power, it became possible to train more complicated models

to learn on more specific qualities of the structures, to adapt with novel methods mainly based on PyTorch.

The existing models were successful in assessing the performance and sometimes even predicting the inner structure of the specific elements. However, the topic of generating synthetic data remains a novel one. CNN is good for implicit learning by capturing visual qualities and deriving the changes in pixels implicitly. Which also in turns trains the model. However, this existing method is too simplistic and weak, since it doesn't account for correlations and doesn't produce realistic outputs in terms of physical qualities.

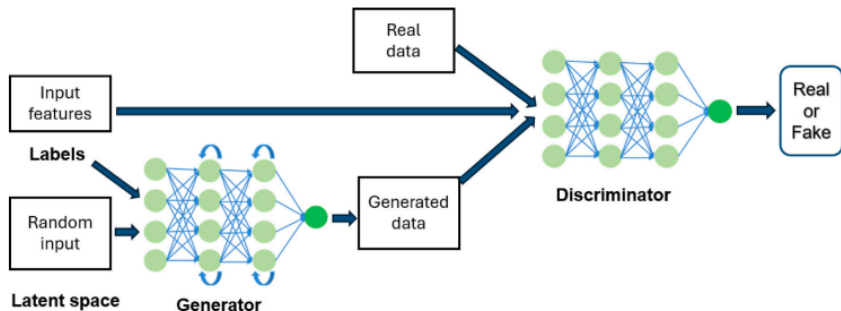


Figure 1-2: An example of GAN architecture [2]

This is where a technology of Generative Adversarial Networks (GAN) come in. It does not have extensive usage in chemical sector, just as in fuel cells too. The architecture of GAN is strong for fuel cells in particular because of general structure of critic and generator. It has a reinforcement learning of multiple Conv2D or Conv3D convolutional layers that are rigorously studying the pixelated structure of the dataset and try to produce the fake image. This is the work of the generator, then the discriminator is going to study for errors and repeatedly dismiss it until the fake one looks so accurate that discriminator is not able to distinguish between fake and real one.

GAN architecture allows for more reinforced and implicit learning to retain the chaotic/anisotropic quality of fuel cell. Since scientists do not have unlimited access to many new CT images of the fuel cells for more energy applications, GAN is designed to make them synthetic data for research and rigorous development. Therefore primary

motivation is to provide convenience for fuel cell studies by making realistic datasets that are fully synthetic and generated with a help of GAN technology.

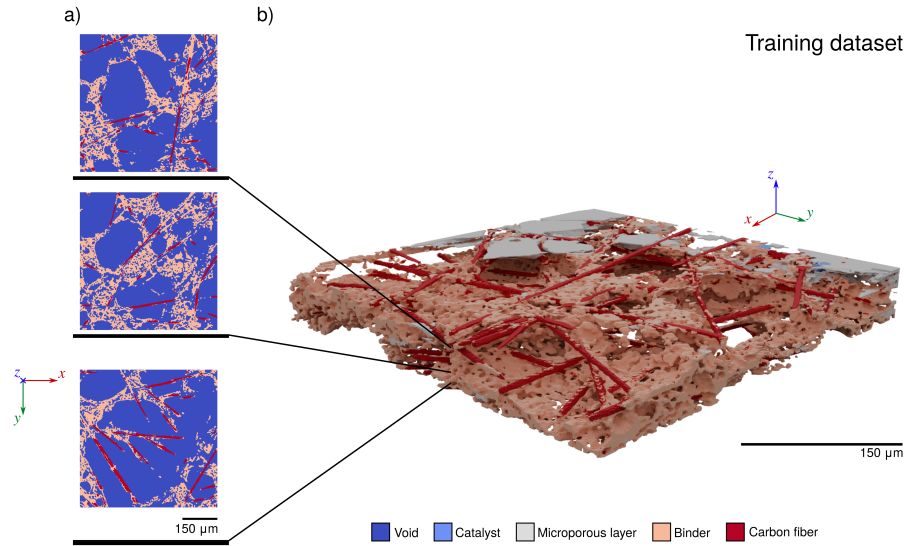


Figure 1-3: Dataset that is used for this work obtained from a paper by Tang et al. [3]

1.2 Main Objectives

The main aim of this thesis is to combine the GAN framework into the fuel cell battery field of knowledge. On top of that, it must be realistic enough in terms of physical properties to be used extensively in scientific projects.

1. Development of GAN-based and physically realistic framework

The main objective is to create a highly efficient and physically accurate model to provide an effective solution for science works to use by Conditional Generative Adversarial Network (C-GAN).

2. Digitally represent the physical behavior of carbon fiber and binder structures

The model must learn on the existing dataset to have the properties studied and replicated accurately during generation part for further research. It should avoid overfitting since every generated sample must be unique.

3. Ensuring a link between carbon fiber and binder

While binder is known to be anisotropic and complex to predict due to its foamy visuals, it is important for it to follow the direction of carbon as it cannot develop chaotically. In the same way, the directions for carbon fiber will be firmly articulated to follow a pattern for accuracy.

4. Replicating the Z-Axis Continuity while still being 2.5D

Even with the limitation of this work being that it operates not fully in 3D, it still replicates this behavior by consistently linking all generated slices to ensure Z axis continuity. This is about proper visual representation of carbon fiber and binder.

Chapter 2

Related works

2.1 Groundwork for GAN

The main topic for this research is to analyze the generated images of fuel cells of carbon fibers. This research is aimed at producing those images by studying GAN (Generative Adversarial Networks).

The concept of two neural networks competing each other was vaguely researched as early as 90s, since there was an increasing interest in improvement of the AI system to boost its learning rate [6]. The concept was relatively simple, however, it has laid a groundwork for further enhancement of the adversarial models concept itself. One of the key ideas of this thesis were to make every other neuron to learn independently. It is necessary to enforce this behavior because they will expand the scope of learning unique features. It has started with the simpler algorithm which was reliant on encoding and decoding. One model is responsible for encoding and the other one for decoding. Progressively, the encoder must make it more and more difficult for the decoder to solve the problem. Hence the learning rate becomes increasingly efficient.

Those systems were designed to create internal tension of competition. One will generate some encoding that is initially easy to solve, and the "generator" stand-in will decode it easily. By making it more complicated each time, it is progressively more difficult to solve, hence the learning rate becomes much more steep. The core of the model is avoiding being too predictable. It enforces the algorithm to find

more efficient paths to be versatile and avoid a stalemate. By enforcing variety, it has to come up with new ways to confuse the decoder. Therefore, it might learn new techniques implicitly. It also makes it more efficient by reducing all unnecessary repetitions with much more emphasis on unique cases and thorough testing that GAN thrives on.

2.2 Further GAN development

The specific adversarial network as we know today was invented in 2014 by Goodfellow et al. [7]. It has formally introduced the concepts of Generator and Discriminator that are now commonly used as the bigger part of the GAN architecture. This novel research was initially directed towards the molecular behavior with a big question of learning based on the existing knowledge to create (generate) new dataset. The skeleton of the model was about discriminator evaluating whether the generated one looked realistic enough. At the beginning stages, it is very apparent for the critic to easily know the difference. It is important to find balance so that the generator is not being crushed by discriminator. Frequently, there could be mode collapse if the generated images stay fake with no improvement or the result is too far off.

The first research by Morizet et al. [4] were concentrating on improving the performance of experimental tests on fuel cells of hydrogen. While it is not directly related to carbon cells, it is still worth to study their approach on enhancing the GAN model that will analyze the images and produce a smoother output. One of the main advantages is that this approach will optimize the research of fuel cells by providing significant optimization. The general structure of GAN will work via generator part that trains on images and tries to make a fake copy of them while the discriminator will distinguish between those two.

It is important to note that the failure rate must converge after the learning process. Additionally, this research mentioned several types of GAN like WGAN, it shows that there are many approaches to consider to find the best suitable one for the scanning.

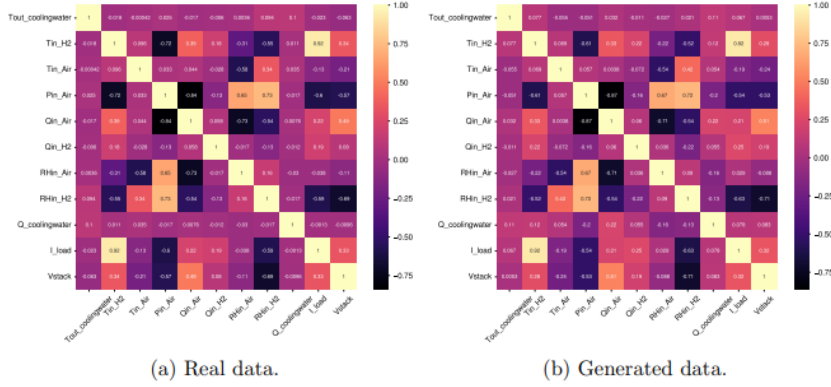


Figure 2-1: A way to compare the readings of Generated and Real data [4]

Another research by Qiu et al. [8] is describing the properties of battery samples of how they are affecting the overall performance. Their aim was to present generated images for research enhancement as it optimizes the overall architecture. The main was concerned with SOC values and estimation as the final report revealed that it has a significant effect on performance. Counting the absolute error allowed to look at the problem more precisely.

2.3 GAN development challenges and issues to note

There is a difficulty of studying the exact patterns of the fuel cells, as it is not predictable for establishing the forms of it. Kharat et al. [9] argues that thermal analysis might help to analyze the system more precisely as there could be numerous similarities between the samples. They are behaving differently when there is a temperature change. Liu et al. [10] discusses the recreation of images from the 2D samples into 3D structures.

This approach is helpful in terms of studying the whole sample from different angles as it allows seeing the various properties and ways for enhancement. Another study by Liu et al. [11] was concentrated on studying the error and faults in designs of fuel cells by using GAN. This approach provides tools for seeing the faulty properties of cells, and take those into account when creating a full structure. GAN samples would be more enhanced and the training will go deeper. Ma et al. [12] provided a

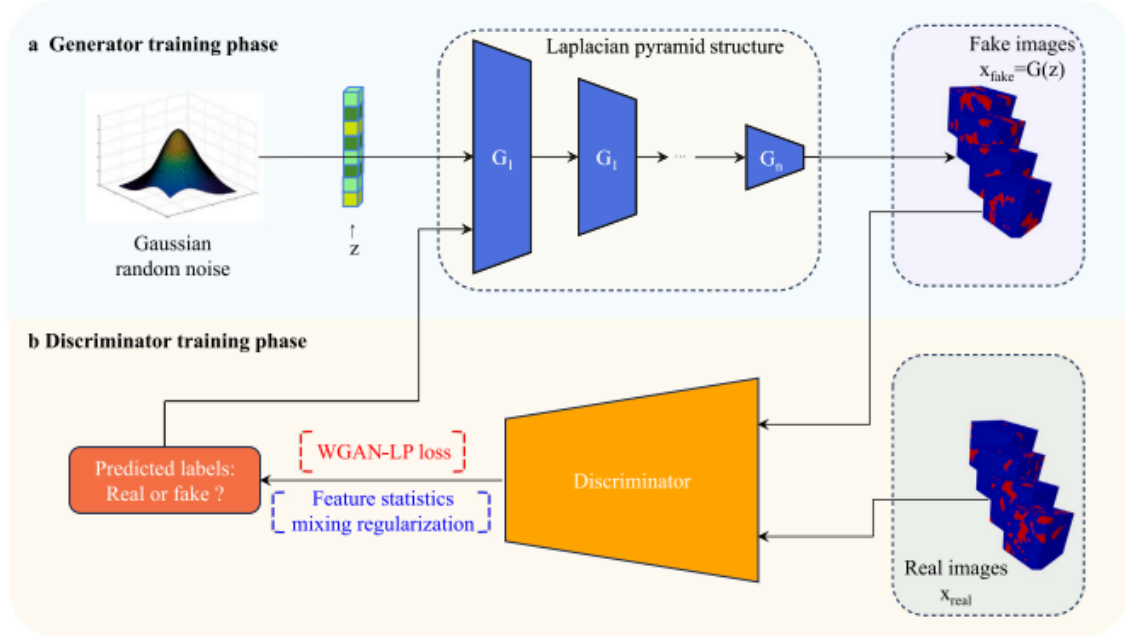


Figure 2-2: Predicted Values for GAN [5]

physical analysis which provides a mathematical modelling for further accuracy as the generated images must have a basis being very similar on ground truth images of fuel cells.

A study by Niu et al. [13] analyzed the hydrogen structure with their catalyst cells. Studying various layers helps to understand the basic structure of carbon fibers as well and gives a broader picture of the approach regarding the usage of GAN. Wang et al. provides a deeper understanding of deep learning algorithms [14]. Kench et al. is focused on analyzing optimization of battery structure [15]. Bai et al. provided a more robust and novel approach in analyzing the fuel cells [16].

Testasecca et al. discussed the creation of digital twin for the research as its recreation is one of the goals of my research [17]. Niu et al. provided different approaches for modelling of the fuel cells as it also analyzed restoration of images. Wang et al [18]. Umme et al. studied the structure of porous materials as it is highly relevant for the current research [19].

Machmud et al. studied the microstructure and 3D visualization which is helpful in understanding of deep learning algorithms for fuel cells [20]. Yang et al. studied

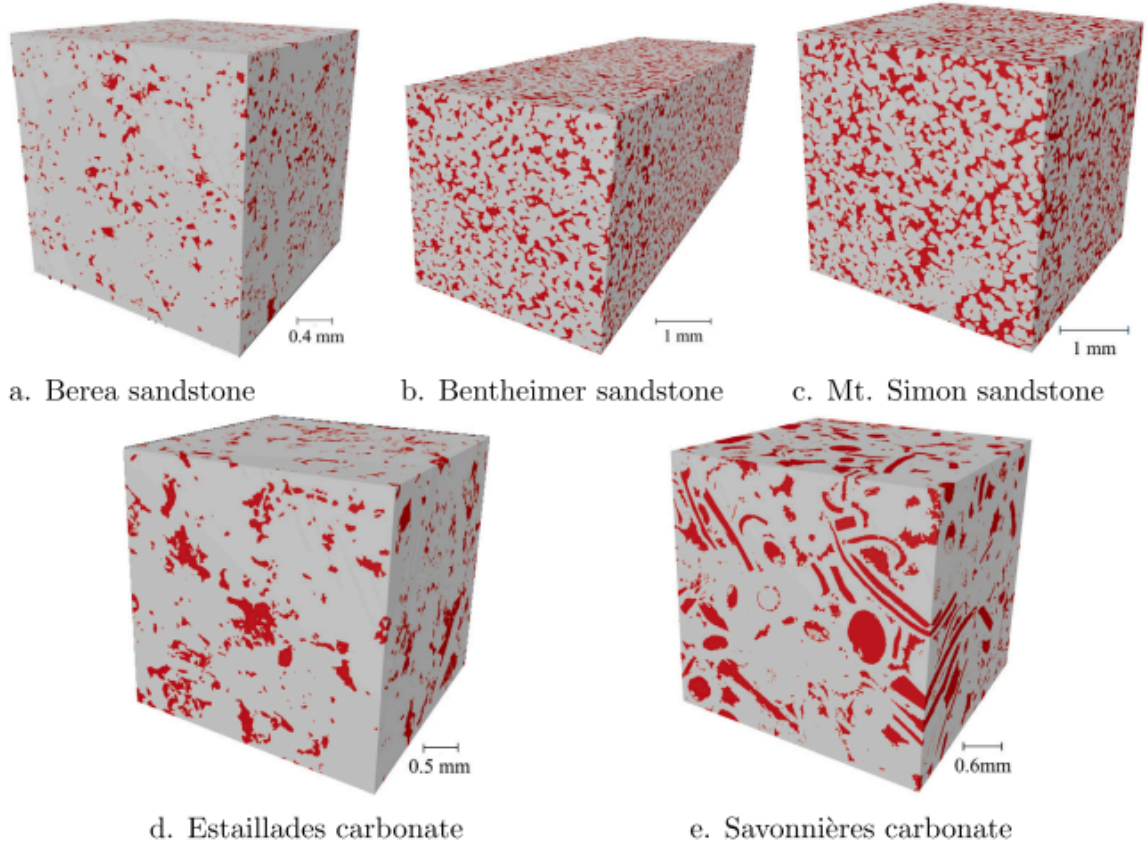


Figure 2-3: Example of generated 3D samples [5]

the microstructure segmentation as it is important to correctly label the microstructure when it will be recreated with a GAN model [21]. PEMFCs in particular has a distinct quality of degradation that severely effects the performance similarly to regular batteries. It significantly reduces its longevity [22].

2.4 Further important GAN aspects for consideration

There are also problems with thinning layer of the binder, which also affects the quality of carbon fiber. Intensity and porosity play a crucial role in efficiency of the batteries. It is not possible to make them fully tortuous, otherwise, it might lead to explosions and overheat, conversely, it might lead to worse degradation even if applied

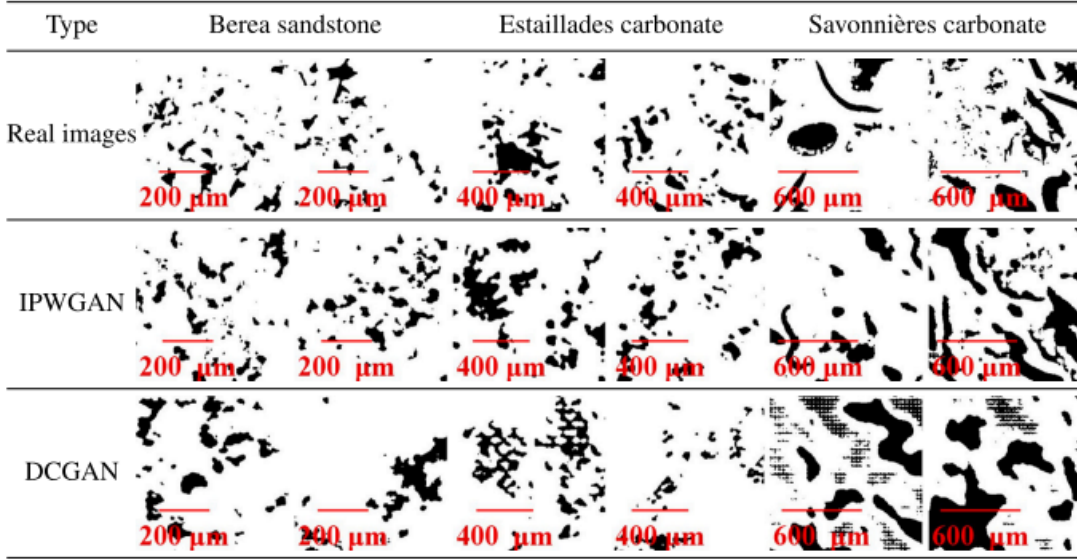


Figure 2-4: Comparison of various GAN images generated [5]

closer to the limit of the material. Henceforth, a balanced approach of GAN gives us general idea how the system might be improved by generational models. The main difficulty is that degradation cannot be replicated strictly in mathematical terms as it is a completely random unit that makes a very unpredictable trajectory to work on.

Another layer of complication is correlation with time, as the degradation process is already quite complicated, it is also has to be represented with a function that is related to passing of time. This random occurrences of quality drop are also considerable research gaps in generating PEMFC slices [23].

A novel approach was proposed by using self-supervised learning along with generative adversarial network [1]. The defining characteristic is to use learning that is highly supported by highlighting contrasting differences which serves the self-supervision.

Another research was proposed to characterize the pores in a structured way. StyleGAN was deemed a good approach since it is mainly based on specific visual cues of the original that could be simulated [24].

Methodology includes separate parts of GAN architecture that is based specifically on Wasserstein-GAN which was developed to address the instability of GAN systems.

This is especially crucial for the carbon fiber dataset. The difficulty of the system is that slices for PEMFC were mostly generated as 2D, 3D was not as common. Therefore, an issue of z axis continuity should be addressed fully. I.e the setback of failing continuity severely damages the physical properties [24].

A useful research also considered nuclear reactor slices that were later mimicked with a help generative network [2].

A possibly useful method is combination of GAN architecture for several more layers of convolution [5].

Chapter 3

Methodology

3.1 Dataset

The dataset for this work was derived from the article by Tang et al. [3]. It is open source, therefore, it is free to use for scientific work. The dataset represents Proton Exchange Membrane Fuel Cell (PEMFC) assembly. It has three main layers. Gas Diffusion Layer (GDL), Membrane and Catalyst Layer. This work only focuses on GDL layer as it contains the necessary parts with carbon fiber and binder for the model to learn its physical properties.

However, it gets increasingly complicated when considered with membrane and catalyst layer they differ heavily even at basic visual level meaning that it will have highly diverse pixel distribution. GDL is uniform enough to replicate the visual differentiation and simulate the physical properties.

The images were created using microCT and the article's aim was to make CNN-based segmentation and compare it to the manual segmentation that typically takes longer to do with image resolution of about 900 nm. The images were grey, black and white initially, then UResNet architecture is used to train the dataset to learn on segmentation. The learning rate is increased by examples of manual segmentation. It was necessary to divide the dataset samples to get the ground truth for comparison which was a helpful approach in this work.

The segmentation as demonstrated above, was divided into 3 classes: Void, Carbon

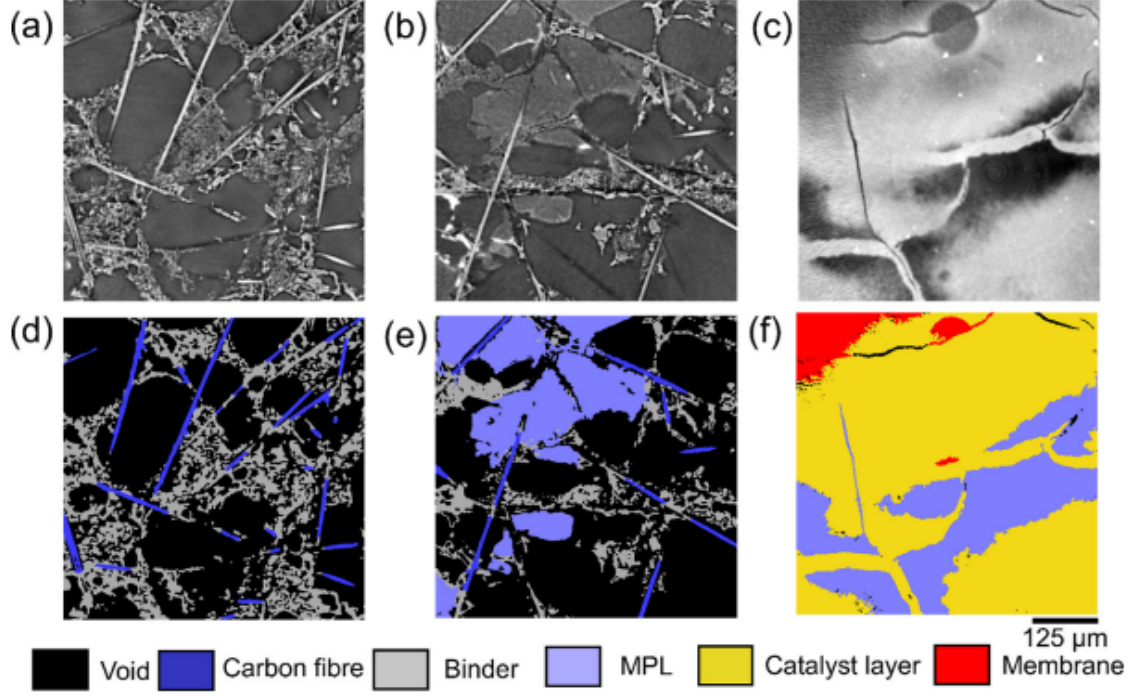


Figure 3-1: Dataset structure [3]

fiber, Binder, and MPL. However, for thesis, a significant streamlining was applied to enhance the learning process. I have only focused on Carbon fiber and Binder. A common way to artificially increase the dataset is using augmentation that was also used in this work by turning and rotating it by 90° , 180° , 270° which will create more 2D images that are helpful to study pixel by pixel [3]. The loss was calculated by the following formula:

$$\text{Cross Entropy Loss} = - \sum_{k=1}^N (p_k \log q_k) \quad (3.1)$$

3.1 is the loss function to train the CNN. It is used to measure difference between generated distance of pixels and actual ones from the dataset. p_k is the probability of the ground truth, while q_k is predicted probability. There is a logarithmic penalty, i.e. if the model is almost nearing 1 in terms of probability, then the loss will be 0, hence being accurate to the original value. Conversely, if the loss is very high, it forces the model to adapt and realize that it is going in wrong direction during training [3].

$$\text{Pixel Accuracy} = \frac{\sum_{i=0}^N p_{ii}}{\sum_{i=0}^N \sum_{j=1}^N p_{ji}} \quad (3.2)$$

3.2 equation acts as a more generalized one to calculate for the whole 3D volume. While it might not be specific, it still is helpful for the model if it is not fundamentally going wrong like quickly identifying empty space. Denominator indicates a sum of all predicted class instances of i divided by sum of pixels in dataset [3].

$$\text{Phase Accuracy} = \frac{p_{ii}}{p_{ii} + p_{ji}} \quad (3.3)$$

3.3 is particularly helpful in dealing with separation into stages. This is because the separate stages act with different physical behavior. As mentioned, the most complex one is carbon fiber that is acting in a straight line, while binder is more anisotropic. It is necessary consider the materials in isolation before building the relationship of mutual dependence [3].

Denominator p_{ji} is used for correctly identified pixels, while p_{ii} is about erroneous ones. It is more about identifying those that were incorrectly identified as correct ones, hence false positives [3].

This was also made more difficult in distinguishing carbon fiber from binder since the original greyscale dataset, it is easy to confuse them. Therefore, such algorithm is also a good basis for more defined and segmented material for training that will be used in this work.

$$\text{mIoU} = \frac{1}{N+1} \sum_{i=0}^N \frac{p_{ii}}{\sum_{j=0}^N p_{ij} + \sum_{j=0}^N p_{ji} - p_{ii}} \quad (3.4)$$

3.4 is more of a standard formula used in training models to see how much overlap is between generated and real data. Primary usage is to avoid overfitting. There is a common problem of the data being too small for training, therefore, the model might just find a loophole to simply copy the exact behavior so that the loss is minimal and accuracy is as high as possible.

Here, intersection is p_{ii} , while their union is represented in numerator, which

is sum of $p(ji)-p(ii)$ and $p(ij)$. It means to find the total area that is taken by both predicted data and real data calculations.

Overall, all these metrics and losses will be particularly helpful in creating a model that will generate structures from scratch, which is a direct development of this dataset that moves on from being merely a segmentation.

3.2 System Configuration and Strategy

The work for recreating images and training the model requires considerable computational power. In general terms, the physical parameters that are numerical are abundant, therefore would need significant RAM usage. The next part is training the model and generating the images. This is more computationally costly. Development of GAN was done using RTX 5080 with 16GB Memory.

There are two big parts of GAN. For Generator, it uses U-Net structure to skip connection and to be allowed to have a larger batch size. Therefore there was no need to crop the original dataset, and full computational power was used to generate images of original size. The size of the individual image is 656 x 656. The part of the dataset that was need was only from GDL, not other parts. Therefore there were about 80 such individual slices.

A significant advantage is that the training is not fully 3D, rather being a stack of 2D images that will have a continuity and eventually stacked together in the end to then simulate a 3D image. Therefore, there was no need to over-complicate the training, instead focusing on physical properties and pixel values alone. It was built on Python 3.10 with CUDA 12.6 to ensure full utilization of GPU. The all-encompassing frameworks that form a basis is PyTorch 2.11.0 with CUDA 12.6 support.

3.3 System Architecture of C-GAN

The core part of this work is C-GAN - Conditional Generative Network. By adding more specific features into the training of the model, it becomes conditions that

generator must meet to satisfy the discriminator. The baseline was derived from Tang et al. paper [3]. U-net is a good base for the generator as it explicitly learns on the pixel structure of the dataset where it implicitly learns the patterns. While the discriminator is mainly based on PatchGAN architecture with basis provided by Shrivastava et al. [25].

3.3.1 Data Preprocessing and Data Loading

A significant part of the project is to prepare the data for training part. It is not efficient for the model to learn on readily available data, it must be standardized so that the features and patterns are identified much faster and more efficiently.

The initial part is preprocessing by looking through the original tif file dataset. As mentioned, it is not relevant to focus on membrane and catalyst layers, therefore, exactly the training will be around slices 80 to 150, hence about 70 slices in total. An important feature is that they are not anisotropic, i.e. they follow visible smooth continuity. That is why it is not as simple to use a variety of datasets, because the CT slices have a high chance of being too different from one another to make the model find a consensus. This will be the most representative region to make a consistent pattern basis for GAN model.

Table 3.1: Preprocessing Workflow for Experimental Microstructure Tomography.

Pipeline Stage	Operation	Technical Implementation / Parameters
Acquisition	Stack Loading	<code>tiffimage.imread</code> (Support for 2D/3D inputs)
Segmentation	Z-Axis Slicing	Selection of frames [80:150] (Region of Interest)
Alignment	Spatial Padding	Zero-padding to multiples of 16 (U-Net compatibility)
Normalization	Label Mapping	Unique pixel values $\rightarrow$ Categorical Indices $\{0, \dots, n\}$
Vectorization	Tensor Casting	NumPy to <code>torch.long</code> conversion
Offloading	Device Transfer	Migration to CUDA/GPU for accelerated training

The dataset should be prepared for Generator part. Since Generator is using U-Net Architecture, it starts from the smaller part of the image to gradually grow upon it. Therefore, downsampling is important to use in this case. It is better for the downsampling to be a multiple of 16. It will also calculate how many pixels are

missing to reach the next sample, by adding 0 if it is missing within the pixel value to maintain the shape without cropping.

The most critical part of the preprocessing that will be true for the rest of the training as a backbone, is usage of color classes for each of the fuel cell elements.

There are 3 elements that are at the focus in this work. The primary one is Carbon Fiber, Binder and Void. Conditionally, they were given values of 0 (Void), Binder (4), and Carbon Fiber (6). For this to be used by neural networks, they need to be enumerated into indexes for easier identification and learning process. This will be critical for Cross-Entropy loss and One-Hot encoding. Finally, it needs vectorization to index every single pixel with further moving it into GPU training directly with torch function.

3.3.2 Data Models

There are several convolutional layers within both generator and discriminator. Refiner Generator uses Stochastic U-Net generator, while discriminator is using Spectral-Normalized PatchGAN.

Refiner Generator

Generator consists of 4 downsampling stages which is as a whole called as Encoding, while the rest of 4 upsampling stages is called Decoding. They are connected via symmetric skip.

Below is the formula of Encoder-Decoder logic:

$$E_i = \sigma (\text{BN} (\text{Conv2d}_{k=4, s=2, p=1} (E_{i-1}))) \quad (3.5)$$

where σ denotes the *LeakyReLU* activation, k the kernel size, s the stride, and p is the padding.

This acts as a visual generalizer to see the whole picture before going into more detail. Every single slice has elaborate structures that are initially difficult to replicate. However, this acts to squint the image for the beginning to then add more and

more detail later as the learning starts and goes on.

Starting with E, it is necessary to look at the previous image as a ground logic for this work to maintain 3D continuity to then scan for structures in it, in this case, it's carbon fiber and binder. BN which is Batch Norm is necessary to keep the calculations strictly in range so that it is not going to make up irrelevant pixel placing. It ensures that the network stays stable and is not going to collapse under generator working without bounds.

σ is like a decision maker to decide during the process if a certain feature worth keeping or not. Therefore, it is somewhat like a filter so that the pixel are not acting chaotically. Overall, the main purpose of this stage is to shrink the image to deduce most important parts to keep the grander picture.

$$\text{Bottleneck} = \text{ReLU}(\text{Conv2d}_{k=3, s=1, p=1}(E_4 \oplus \text{Tile}(z))) \quad (3.6)$$

In the formula above, $\oplus$ denotes assigning unique features for more accurate visual representation. Bottleneck represents the smallest point in the picture, hence unique features are added around it more and more like a concatenation. E is a summarized version of a fuel cell, similarly to the previous formula deciding a bigger picture after simplification.

A critical role of this part is avoiding overfitting for this model. It will be useless if it just decides that certain properties are enough to meet, hence there is no need to develop it further and it can simply meet accuracy by copying the specific parts of the dataset. In a broader sense, it will also ensure that one of the main objectives met to make the model generate completely unique outputs every time it is called.

$$D_i = \text{ReLU}(\text{BN}(\text{Conv2d}(\text{Upsample}(D_{i-1}) \oplus E_{n-i}))) \quad (3.7)$$

Lastly, this part is more layered. It is exactly the reverse process of encoding, here, it is decoding. After getting a general picture, it should be more detailed with more pixel scale, which utilizes upsampling instead of downsampling. $\oplus$ again used for previous image values, then upsampled, went through Conv2D convolutional layer

to then filtered through BN, and finally, ReLU. ReLU role here is to skip connection and act as a memory bridge. It is crucial for smooth behavior of both carbon fiber and binder. So the generator system is essentially, shrinking, knowing the broader picture, and upscaling with a more detailed scale.

Table 3.2: Summarized Architecture of the RefinerGenerator

Stage	Block Type	Layers	Channels (C)	Resolution
Encoder	Contracting	4	$6 \rightarrow 512$	$128 \rightarrow 8$
Bottleneck	Latent Fusion	1	$512 + z \rightarrow 512$	8×8
Decoder	Expanding	4	$512 \rightarrow 32$	$8 \rightarrow 128$
Skip	Connectivity	3	—	—
Output	Prediction	1	$32 \rightarrow 6$	128×128

Discriminator

As mentioned, a basis for Discriminator was PatchGAN. The main function of this model is to act as a critic. Generator works to generate images based on the dataset, while Discriminator’s job is to compare with the existing data, so that it is going to distinguish between real and fake ones. The focus here is to make an elaborate learning process to make generator able to create the slices being so accurate in terms of physics and visuals, that Discriminator is not going to be able to tell the difference. Hence the error is going to get lower and lower each time.

One of the important aspects of Discriminator is to find balance. It is a common problem for GAN to make Discriminator too powerful that it immediately shuts down anything that Generator is creating, hence too much effect on loss that Generator is not able to create anything. This is only bound by how many epochs are given for training, but the real danger is that there could be mode collapse that generator simply never gets the main aspects of the ground data to recreate it and Discriminator is not helping.

A balanced way for Discriminator to operate is to find proper values to consider from the dataset. In this case, it should be bound to the pixel values that were received from the preprocessing stage.

Given that the dataset that needs to be recreated is highly dependent on smooth visual features, Discriminator will look most at the neighboring pixels so that there is no visible discontinuity that breaks the smooth features.

$$D_i = \sigma (\text{IN} (\text{SN} (\text{Conv2d}_{k=4, s=2, p=1} (D_{i-1})))) \quad (3.8)$$

The formula above shows how the convolutional layers are processing the pixel values. As its name suggests, PatchGAN looks into the parts of the image (patches), to judge based on the locality.

The overall logic is about finding the correct placement of the nearest pixel by downsampling and using it as a grid for judging the generator output. Those layers in the formula are calculated with the current grid, then it goes to Conv2D layer, and then the normalization instances start.

SN stands for Spectral Normalization. It is necessary to create bounds for the weights so that they do not exceed the accepted values. While IN stands for Instance Normalization, as its name suggests is about the individual features of the slice to be normalized into a standard for better judgement and preserving the uniqueness. σ is LeakyReLU which handles the stabilization of data during the convolution layer concatenation.

$$O \in \mathbb{R}^{1 \times N \times N}, \quad O = \text{Conv2d}_{k=4, s=1, p=0} (D_{\text{last}}) \quad (3.9)$$

This formula will output the final score of accuracy to then have the final output on judging the Generator output. It will output a feature map where each N x N value represent the grid that was taken to assess the realistic features.

Table 3.3: Summarized Architecture of the Discriminator

Block Type	Channels ($C_{in} \rightarrow C_{out}$)	Resolution	Key Operations
Data Ingestion	6	128×128	Real or Fake Class Map
Feature Extraction	$6 \rightarrow 64$	$128 \rightarrow 64$	SN, Conv2D, LReLU
Deep Encoding	$64 \rightarrow 256$	$64 \rightarrow 16$	SN, IN, Conv2D, LReLU
Judgment Head	$256 \rightarrow 1$	$16 \rightarrow 13$	SN, Conv2D, Linear

3.3.3 Model Training

This step is using the augmentation from the preprocessing that involved turning the image into different positions so that the model does not learn limited number of paths for carbon fiber.

Initially, the image must be broken to start with fake looking one as a starting point for Generator to have a point to develop on.

$$\text{shifted_fibers} = \text{Roll}(\text{fiber_mask}, (s_y, s_x), (1, 2)) \quad (3.10)$$

GAN is generally starting with a chaotic image for generator, as it gets every possible pixel value first to then gradually understand the underlying physical properties that are represented pixel-wise. It is used to generate foam, to replicate the physical behavior of binder, and then overlay this behavior over binder itself. This is because the pixels are randomly shifted to neighboring. However, it was also important for it to not be flickering, therefore, the range of randomness was not too high to make it as smooth as possible.

Next is the stage where Losses get introduced.

$$\mathcal{L}_D = \mathbb{E}[\text{ReLU}(1 - D_{\text{real}})] + \mathbb{E}[\text{ReLU}(1 + D_{\text{fake}})] \quad (3.11)$$

Here, Discriminator is shown both fake and real versions. It will train it to distinguish between the two. If the score is more than 1 for real image, and the fake image is less than -1, the loss will be zero, meaning high accuracy.

Discriminator acts on its main function to assess the generator output, while generator is going to recreate the fake image so well that eventually, Discriminator is not going to tell the difference. Below is the formula for Hybrid Loss.

$$\mathcal{L}_G = -\lambda_{\text{adv}} \mathbb{E}[D_{\text{fake}}] + \lambda_{\text{CE}} \text{CE}(G_{\text{output}}, \text{Truth}) \quad (3.12)$$

λ_{adv} and λ_{CE} are weighting coefficients, as λ_{adv} is for making it look like real image, while being fake. While λ_{CE} assesses the physical consistency. CE denotes the

cross-entropy loss.

3.3.4 Image Generation and Simulation of Physical Properties

This final training stage is the most crucial one in terms of enforcing physical properties into the generated stack by applying several independent parameters for smoothness across all slices to simulate a 3D image.

The model has the real sample of slices as a starting point to identify each element separately. The preprocessing involved labeling them into classes, so it uses label and regionprops to distinguish a fiber from the image. It is important to note that the fibers are looking like lines, therefore, it is necessary to see if there are sudden changes in pixel values, it denotes an end of the line, or different edge.

$$\vec{v}_{t+1} = (\vec{v}_t + \vec{a}_t) (1 - \gamma) \quad (3.13)$$

To set the directions (dy, dx), angles are also determined for each individual fiber. They all move at uniform speed smoothly within the image. This creates a simulation of real carbon fibers that move according to their physical properties in a stack of 2D images that compose a 3D image.

In terms of binder, it was important to assign it the foamy structure with particular smoothness and no flickering. It also should be noted that they do not occupy the whole image randomly, they follow the path of carbon fibers by being around it. This affects the porosity and tortuosity directly.

$$\mathcal{B}_{\text{mask}} = \{p \in \Omega \mid \text{dist}(p, \mathcal{F}) \leq \delta_{\text{max}}\} \quad (3.14)$$

This is a continuation point for subsequent slices because they will not be created from scratch, i.e. noise, but use this starting slice as a reference point to then make the carbon fiber travel across it and make binder go around it. This is particularly helpful in computational cost compared to training the model in full 3D volume.

One of the common problems with existing GAN methods is that they were not able to address the problem of Z continuity as many of the samples were 2D images

Table 3.4: Kinetic and Morphological Synthesis Parameters.

Category	Parameter	Operational Value / Range
Kinematics	Fiber Wander Strength	0.05 – 0.15
	Global Drag Coeff.	0.02 (2%)
	Fiber Speed	12.0 – 18.0 voxels/frame
Morphology	Binder Proximity Limit	EDT $\leq$ 15 voxels
	Macro Threshold	0.65 – 0.75
	Morph Rate (α)	0.1 (Linear Interpolation)
AI Refinement	Z-Interpolation	16 Frames (Linear)
	Smoothing Kernel	5×5 Gaussian ($\sigma = 1.5$)
	Min Object Size	60 voxels (Area Removal)

rather than fully generated 3D images. This problem is addressed directly with Kinematic engine in this work to simulate physical properties and make the computational cost more efficient. As shown in Table 4.4, Morphology here is about the foamy visual qualities of the model.

Then finally it ends with AI refinement of preserving the Z axis continuity and final smoothing kernel. More strict rules are added to contain the binder.

distance_transform_edt is responsible for binder to only be allowed to exist within a specified area, so that the model will not collapse without clear bounds.

3.4 Overview

The overall system is structured by having a dataset of a tif file that has about 500 slices of carbon fuel cells. They have a chaotic structure and linear-like shapes represent the carbon fibers. It is important to replicate those shapes as accurately as possible, because they will be assessed for physical and chemical accuracy. It begins with processing stage that is concerned with processing the images, i.e., changing the scale and partitioning. This way data will be transformed for the next stage. Each is meant to give details for GAN architecture to further have abilities to create structures from scratch. The overall structure has 4 distinct stages and each contributes for further enhancement and accuracy.

After fully changing the scale of the slices, the edge detector is going to proactively search for patterns for generator. Instead of chaotically search for RGB (working from chaos into distinct patterns), it uses one-hot encoding which restricts the generating image into a restricted number of channels that are controlled by a probability factor. However, to perform it, the processing must include discovery of classes. The dataset itself has 4 classes: Binder, NPL, Pores, and Carbon Fiber. Since there is restriction to specific colors, there will be a detection of the whole image and each pixel corresponding to a class will be assigned with its own probability. This gives a specialized palette that consists of these colors.

Pipeline Architecture for Image Generation of Fuel Cells

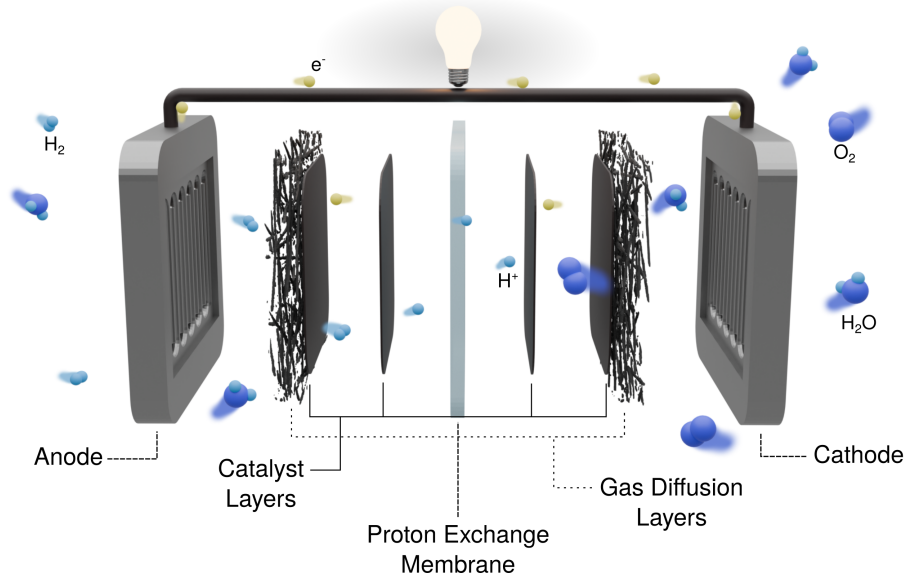


Figure 3-2: Architecture of the system

It helps generator to learn the features more. Those pixels will be used to create the 128x128 image from the very beginning by handling the interpolation and accurately present the class structures. Next is the one-hot encoding part that will resize the pic according to the neighboring pixels to find more patterns. The linear patterns are the most complex ones, therefore, there need to have been a continuity among the final stack of slices. Generator works with several convolutional layers that are going

to work consecutively of each other which are responsible for pattern recognition. After that, the resulting 128x128 images is being investigated by Discriminator. It works with the same convolutional layers as the generator, however, it is only used for comparing the generated images with the raw data.

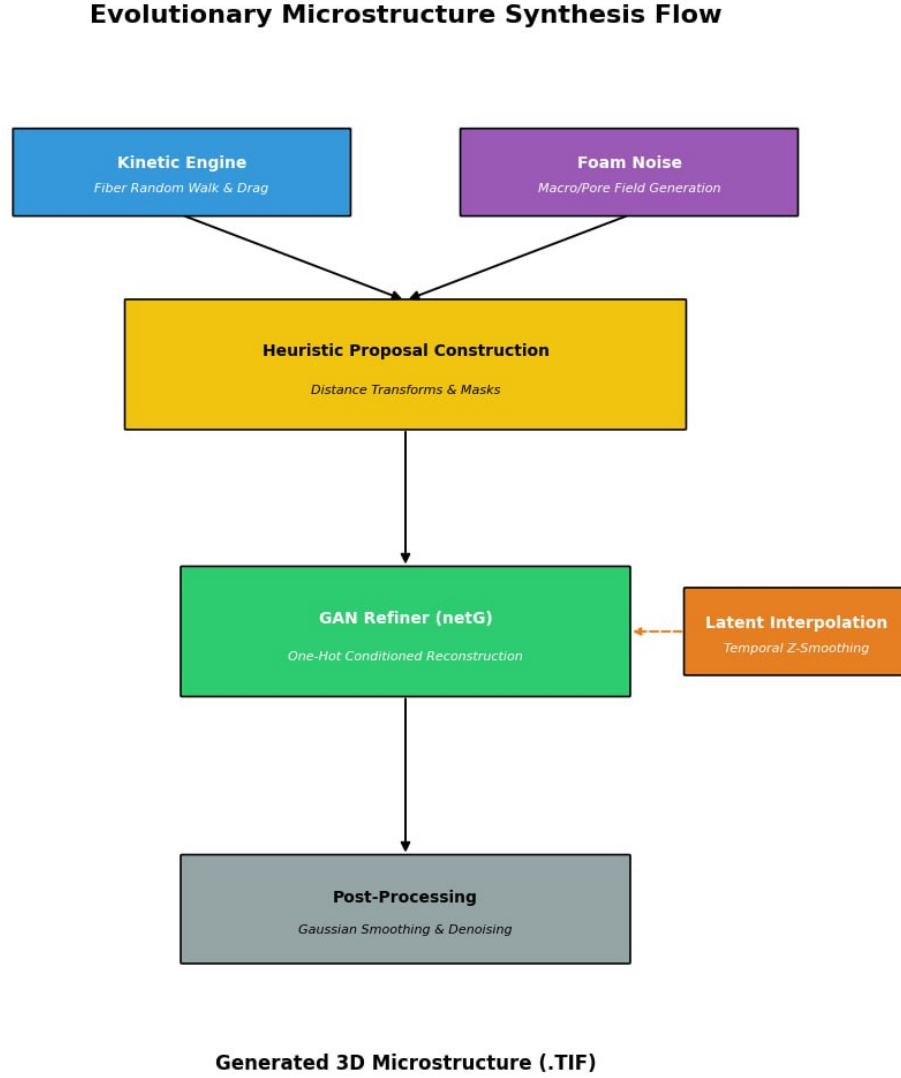


Figure 3-3: Overview of the Architecture

After this, loss function will be applied and calculate according to accuracy. The proposed model mainly works as a hybrid pipeline of scanning the 3D volumetric data to then produce an output simulating a 3D microstructure. Visually, it tries to mimick the physical properties of the dataset. The dataset for this research is

meant to be taken as a ground truth for the training. Therefore, much of the training is spent for analyzing the patterns of the fuel cell data. The module dedicated for learning investigates all the slices, to form continuity across the Z plane. After the training, refiner generator is going to try hallucinating the new slices based upon the learned properties so that there is no sudden discontinuity and it works as a 3D material.

To ensure a more proper continuity, it uses the visuals of carbon material that look like lines that are a little distorted. Those lines are going with their own trajectory. Therefore, while generating new slices, it also takes into account which line of carbon fiber is coming from and where it goes in terms of its path. It is calculated with simple x and y coordinate equations to ensure smooth travel across the line.

But the complexity for this project is to introduce the material that goes along those lines called MPL (Microporous Layer). The Complexity essentially is its structure that is highly reminiscent of foam and while being porous they not as easy to program to move along a certain line. They have to replicate a property of being foamy and porous. And they must change dynamically, but without flickering and with smooth movement. All those properties are seen visually on the dataset that is considered as ground truth. The estimation is a training for about 300-400 epochs that turn out to be optimal for training.

It is also important for the model to not lean too heavily into overfitting. That was the problem with the initial test of Unet convolutional model. To come up with a better model, that is GAN, Unet created accurate copies of the dataset which is useless, it was important to create structures for study that would be completely random to replicate the inside work of actual battery material. Generation part is the most complex one since it ensures that the model operates with complete randomness from scratch rather than simply giving the dataset's reminiscent copy.

This is a primary advantage of using GAN architecture so that the generator is learning from the discriminator, to make such a perfect replica of the dataset that the discriminator is not able to distinguish the fake one and real one. The architecture starts with the noise for initial training since generator and discriminator

are constantly rivaling each other. The complete noise at the beginning must converge into a meaningful image towards the end of the training. This is all controlled by the loss functions to see if the result is not getting too inaccurate and too much hallucinations. Generator initially works with the encoder.

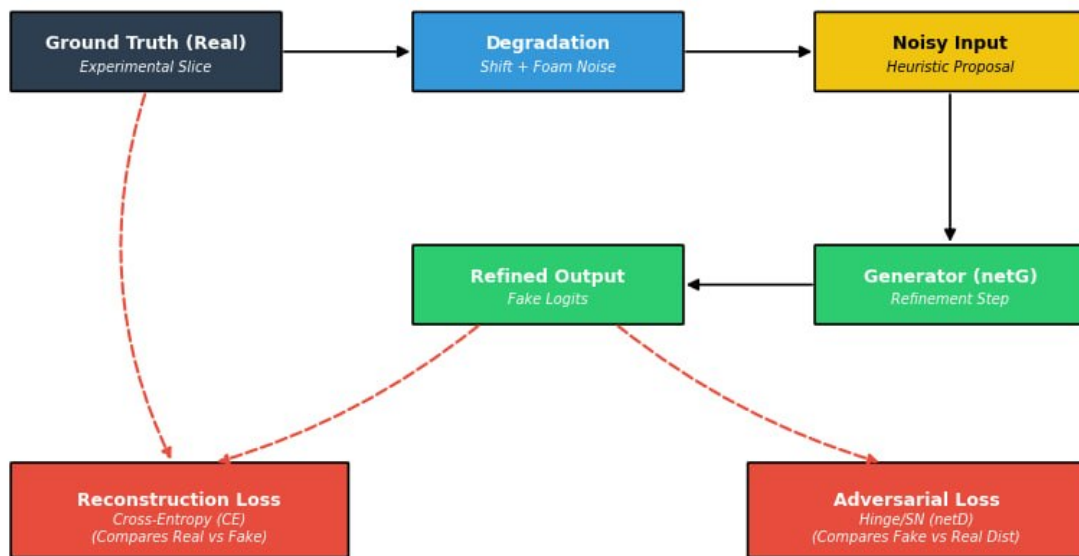
Firstly, the image is severely downsampled to determine the more visible and most significant features. This is further convoluted with more noise. After this, decoder will try to re-sample into a proper resolution. During the phase of Discriminator assessment, it will make normalization layer to be applied more rigorously, to converge the result faster and more accurate. However, it would be too penalizing for the discriminator to simply work as black and white true or false, instead, it operates with NxN patches to determine if parts of the images are within acceptable range.

This ensures steeper learning curve for better optimization. To make loss function more precise, it is divided into different sectors to ensure more profound picture of the generation. Firstly, it is adversarial loss that calculates the loss of GAN itself. Secondly, Cross Entropy Loss ensures pixel by pixel evaluation of accuracy. This is helpful to eliminate overfitting. A smaller part of generation is post processing to eliminate erroneous generated outputs but they deal more with refining the final image rather than profoundly changing it.

Table 3.5: Multi-Objective Loss Functions for Refiner GAN Training.

Loss Term	Equation / Logic	Scientific Purpose
Reconstruction (L_{CE})	$\text{CrossEntropy}(\hat{y}, y_{real})$	Preserves global topology and phase ratios.
Adversarial (L_{ADV})	$-\mathbb{E}[D(G(x, z))]$	Synthesizes high-frequency binder textures.
D-Real ($L_{D_{real}}$)	$ReLU(1 - D(y_{real}))$	Trains D to identify experimental features.
D-Fake ($L_{D_{fake}}$)	$ReLU(1 + D(\hat{y}))$	Trains D to identify synthetic artifacts.
Total Gen Loss	$\lambda_{CE}L_{CE} + \lambda_{ADV}L_{ADV}$	Balances structural fidelity with visual realism.

Adversarial Refinement Training Cycle



Self-Supervised Loop: netG learns to undo degradation by minimizing CE (structure) and ADV (texture) losses.

Figure 3-4: Training Cycle

Chapter 4

Results

The overall work was focused on generating inside structures of battery fuel cell material that is ready to be used for scientific research. It was important to make the battery sample with variety and randomness for it also to replicate physical properties. This will be thoroughly assessed from the model's both visual and generative network point of view. Below is the visual representation of the generated dataset.

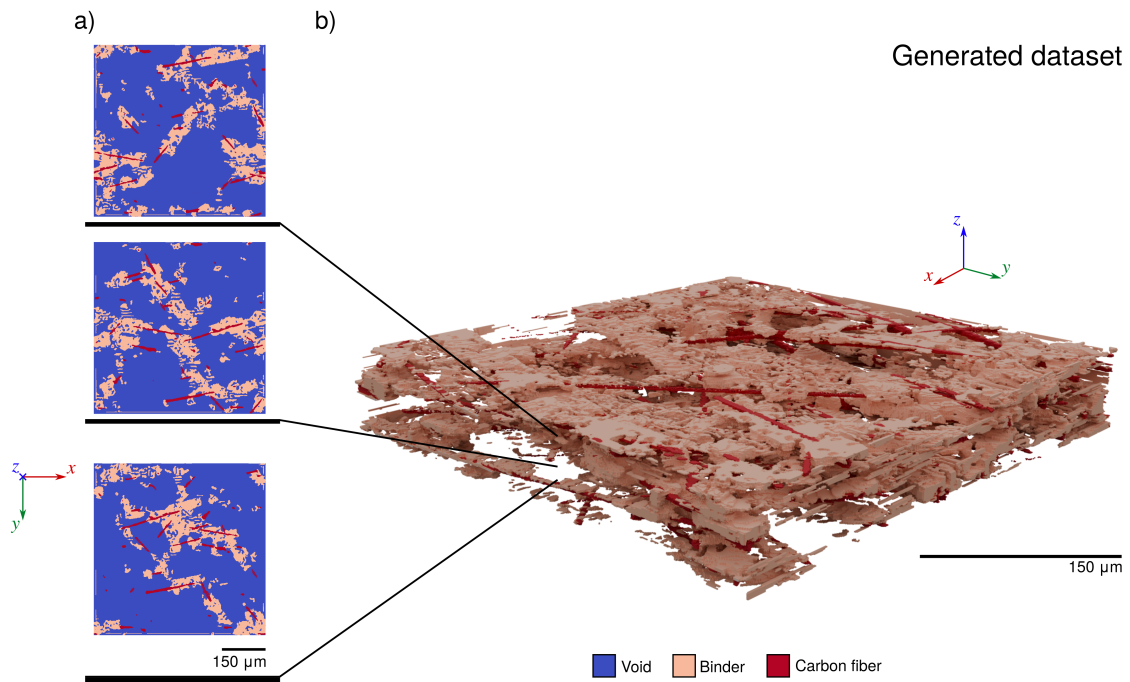


Figure 4-1: Generated Dataset Morphology

4.1 Visual Assessment

One of the primary aims for this thesis was to create a simulation of 3D image by a stack of 2D images in order to recreate the CT slices of fuel cells. The complex part for this work was to simulate the behavior of carbon fiber. Unlike the binder part, it has a more linear defined path behavior. The training on RTX 5080 has lasted for about 8 hours on average.

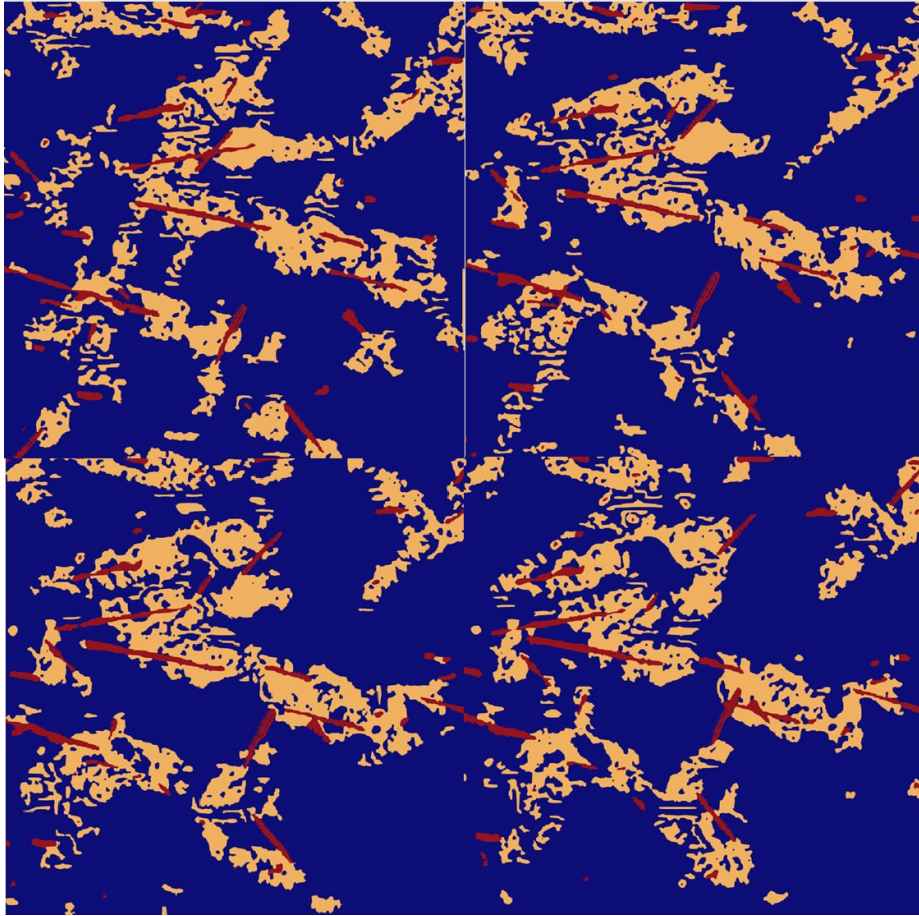


Figure 4-2: The elements change between slices

The above image shows that the change for carbon fibers and binder occur smoothly. 4 slices were taken as an example and this pattern holds for the whole generated 200 slices. To recreate this behavior, after identifying the linear path for carbon to travel. The model has learned to generate random paths that make the fiber travel in linear fashion. However, it is not bound to vertical and horizontal paths. It must be in

diagonal paths as well, and independently.

The 3D model shows that binder became somewhat like covering framing with uniform structure with smooth deepening mimicking the anisotropic features. The model itself looks accurate to the initial dataset that was used for the training, however, it does not copy it. That was the aim of avoiding overfitting, since the model must be enforced to only learn the specific physical behavior that is interpretable from physics perspective.

It was necessary to study the behavior of the model correctly. Therefore, the slices had a distinct feature of binder having the property of following the carbon fiber path. So that it is going to grow along with it with visually soap-foam like behavior. Both parameters are responsible for tortuosity and porosity physical parameters.

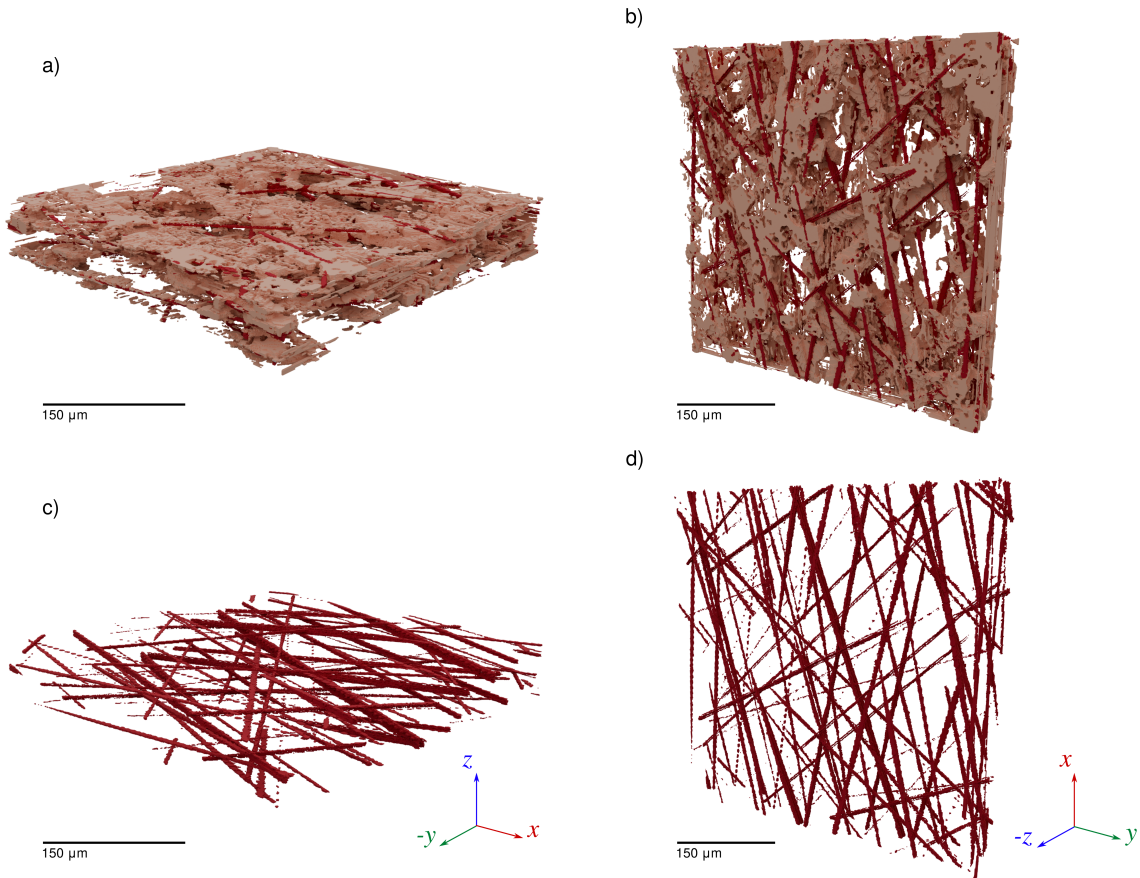


Figure 4-3: Carbon fibers extracted and vertical and horizontal view of the generated sample

The above generated slices visually resemble the original dataset at high rate.

Taking into account that it has smoothness of binder and determined travel of carbon fibers, it means that it meets the basic physical values that are represented visually. Nevertheless, it never acts as a copy of the dataset, rather understanding the inner algorithms than just simply copying it visually.

4.2 Physical Properties

One of the most important aspects and one of the objectives is preserving physical features of the dataset, and to have completely random samples so that the properties will remain. Those physical properties will be reviewed with tortuosity, carbon fiber, and binder calculations via TauFactor, and compared to the values found in the dataset. They are not expected to be identical, however, they should be in a plausible range for those values.

For assessing the physical properties, few parameters were chosen. It's tortuosity, porosity, total binder content and total fiber content. A limitation is that those are more of generalized values to hold onto, however, they still give a plausible insight how much proportion should be for the material to create natural boundaries. TauFactor has calculated that Porosity is about 73%, tortuosity is 1.94 as a mean value for the whole dataset, Binder is 19%, and Fiber content is 3.67%. Overall, fibers are designed to have least space since they are essentially lines, while void is taking most of the space, while binder is in-between as it is going around the lines of fiber.

Table 4.1: 3D Volume Phase Composition (Slices 80–150)

Metric	Value
Total Tortuosity (τ)	1.9423
Total Void (Porosity)	73.17%
Total Binder Content	19.71%
Total Fiber Content	3.67%

Below in Figure 4-5 is the graph for 25 samples to get distribution of different values. It shows that the samples are very close to the values of the dataset, with tortuosity being around 1.75, Porosity is 75%, Binder is just below 25%, and Carbon

Fibers are a little more than 5-6%. This shows that the physical and visual properties were fully retained and they produce completely different and random output at every generation of a sample.

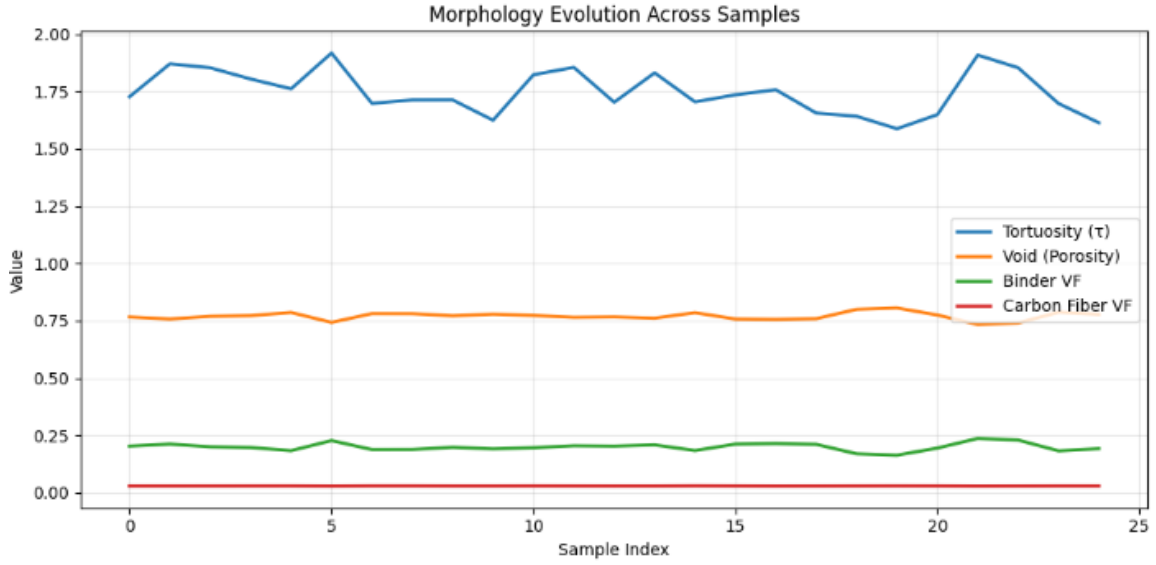


Figure 4-4: Physical properties of the samples

4.3 Model Assessment

Table 4.2: Structural Analysis of the Generator: Layer Magnitude and Sparsity.

Layer	Shape	L2 Norm	Sparsity (%)	Parameters
enc1.weight	[64, 6, 4, 4]	5.49	0.00	6,144
enc2.1.weight	[128, 64, 4, 4]	8.80	0.31	131,072
enc3.1.weight	[256, 128, 4, 4]	13.85	0.40	524,288
enc4.1.weight	[512, 256, 4, 4]	23.35	0.50	2,097,152
bottleneck.0.weight	[512, 576, 3, 3]	24.44	0.55	2,654,208
dec1.1.weight	[256, 512, 3, 3]	15.94	0.54	1,179,648
dec2.1.weight	[128, 512, 3, 3]	12.06	0.54	589,824
dec3.1.weight	[64, 256, 3, 3]	7.20	0.41	147,456
dec4.1.weight	[32, 128, 3, 3]	5.18	0.26	36,864
final.weight	[6, 32, 3, 3]	3.67	0.17	1,728

The table above shows that the model has very low sparsity which ranges from 0% to 0.55%. It keeps all convolutional layers active and contribute to learning process and enhancing accuracy. This means that special parts of the physical properties are not ignored and will be recreated accurately. This also helps for distinguishing binder from carbon fibers. This is represented by a graph in Figure 4-7. It shows that the sparsity starts very low on encoding beginning, gets to a peak with bottleneck, and stabilizes further as it reaches final stage.

Table 4.3: Evolution of Feature Complexity via Batch Normalization Variance.

Block	Mean Variance	Max Variance	Stability (σ)
enc2.2	0.011	0.048	0.009
enc3.2	1.888	6.622	0.995
enc4.2	12.019	103.760	19.264
dec1.2	494.994	2672.000	378.448
dec2.2	36.587	166.843	35.214
dec3.2	4.639	13.375	3.259
dec4.2	0.510	1.965	0.549

Table 4.3 shows evolution of feature complexity. There is an outlier of stability being around 378, but it reflects that the model is very sensitive about Z randomness. It means that it has distinct eye for visual patterns that are necessary to then recreate the physical properties. Nevertheless, there is a gradual convergence towards the end

with values being around 0.5 to 3, which indicates stabilization. This is represented in Figure 4-8.

Table 4.4: Distribution of Trainable Parameters across Functional Model Groups.

Block Group	Parameters	Total Budget (%)
Encoder (enc1–enc4)	2,758,656	37.44%
Bottleneck	2,654,208	36.02%
Decoder (dec1–dec4)	1,953,792	26.51%
Final Layer	1,728	0.02%
Total Sum	7,368,384	100.00%

Table 4.4 shows that the perception of the model is considerably high, with Encoder and Bottleneck taking the majority of the budget. It might be due to the complexity of the dataset that requires to follow continuity across slices and specific features like carbon fiber movement and naturally smooth features of binder. It shows that vectorization was performed successfully and it helped to recreate the images from scratch. It has used the power of GPU so that it will not get any overflow or memory leak. Figure 4-5 shows the difference between layer influence and sparsity, they follow a similar pattern, however, layer influence is decreasing towards the end, while sparsity decrease in a less rapid manner. This is reflected in the Figure 4-6 to see the change visually of how L2 norm changes across layers.

Table 4.5 shows Generator’s statistics summary based on the last checkpoint of the developed model. This indicates that training never had overflowing states and was stable overall. There is no gradient explosion and mode collapse. This is crucial to replicate the smoothness quality of the model. There is a slight increase in activation variance that indicates there was a steady learning curve. It helps with the anisotropic qualities of the binder while also maintaining a more strict movement feature of carbon fibers.

Table 4.6 refers to MSE, SSIM and mIoU scores of the dataset in isolation to test the model of GAN itself. This was also necessary to test the kinematic loop and properties of Refiner Generator. The global consensus for MSE (Mean Square Error) is around 2.022, SSIM is 0.62, and mIoU is 0.6. This indicated that the model handles

Table 4.5: Weight Statistics Summary for Generator Checkpoint

Layer Name	Shape	Mean	Std Dev
enc1.weight	[64, 6, 4, 4]	−0.00470	0.06994
enc1.bias	[64]	−0.00121	0.05886
enc2.1.weight	[128, 64, 4, 4]	−0.00097	0.02428
enc2.1.bias	[128]	0.26988	0.02547
enc2.2.weight	[128]	1.02610	0.04183
enc2.2.bias	[128]	0.00006	0.02839
enc2.2.running_mean	[128]	0.30340	0.18201
enc2.2.running_var	[128]	0.01163	0.00923
enc3.1.weight	[256, 128, 4, 4]	−0.00044	0.01912
enc3.1.bias	[256]	0.39856	0.02247
enc3.2.weight	[256]	1.01117	0.02318
enc3.2.bias	[256]	−0.02847	0.03086
enc3.2.running_mean	[256]	0.25870	1.49164
enc3.2.running_var	[256]	1.88761	0.99473
enc4.1.weight	[512, 256, 4, 4]	0.00002	0.01612
enc4.1.bias	[512]	0.00005	0.01297
enc4.2.weight	[512]	1.02186	0.05194
enc4.2.bias	[512]	−0.00311	0.09776
enc4.2.running_mean	[512]	0.04726	2.44778
enc4.2.running_var	[512]	12.01860	19.26410
bottleneck.0.weight	[512, 576, 3, 3]	0.00013	0.01500
bottleneck.0.bias	[512]	−0.05149	0.02292

the porosity very well and is able to reconstruct the visual attributes.

Geometric placement (IoU) analyses how the model studies the placement of certain pixel elements as shown in Figure 4-10. There might be borders and it is due to the model’s deliberate randomness about binder to mimic the natural foamy ability. While figure 4-9 shows that GAN is fully able to recreate the image from ground truth indicating that it correctly places all the elements without noise. There is a high diagonal accuracy in Phase Attribution of figure 4-10. There are missing rows, but that’s because certain elements are absent. Class 0 is very high accuracy about 0.89, and Binder which is 0.79 also indicate that it is true to the actual physical property and visuals.

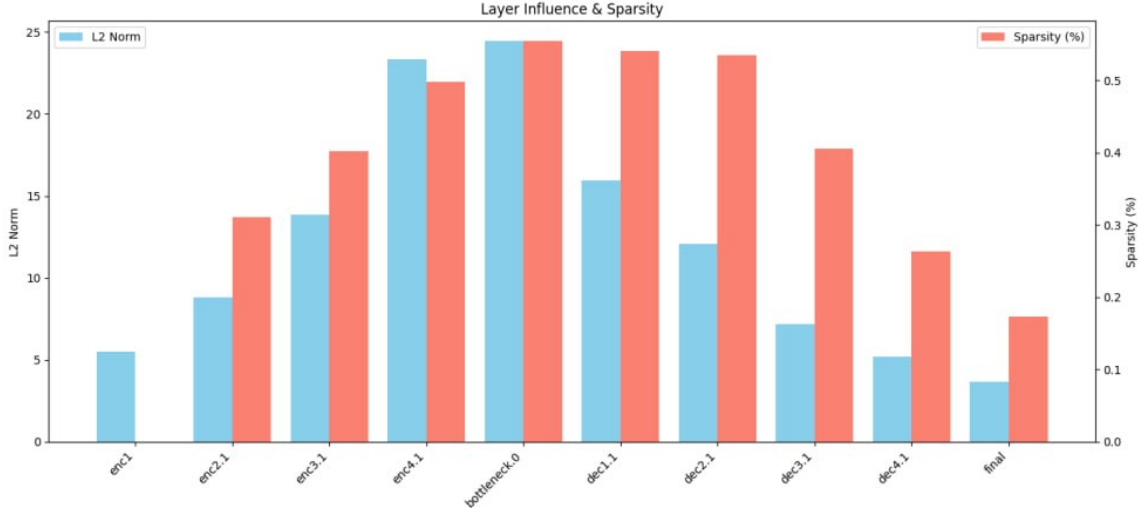


Figure 4-5: Comparison of layers

Table 4.6: Slice-wise Evaluation Metrics with Global Summary

Slice	MSE	SSIM	mIoU
80	2.2571	0.5942	0.5721
100	2.7339	0.5290	0.5058
120	1.7235	0.6690	0.6209
140	1.3750	0.7132	0.7120
Global Avg	2.0224	0.6263	0.6027

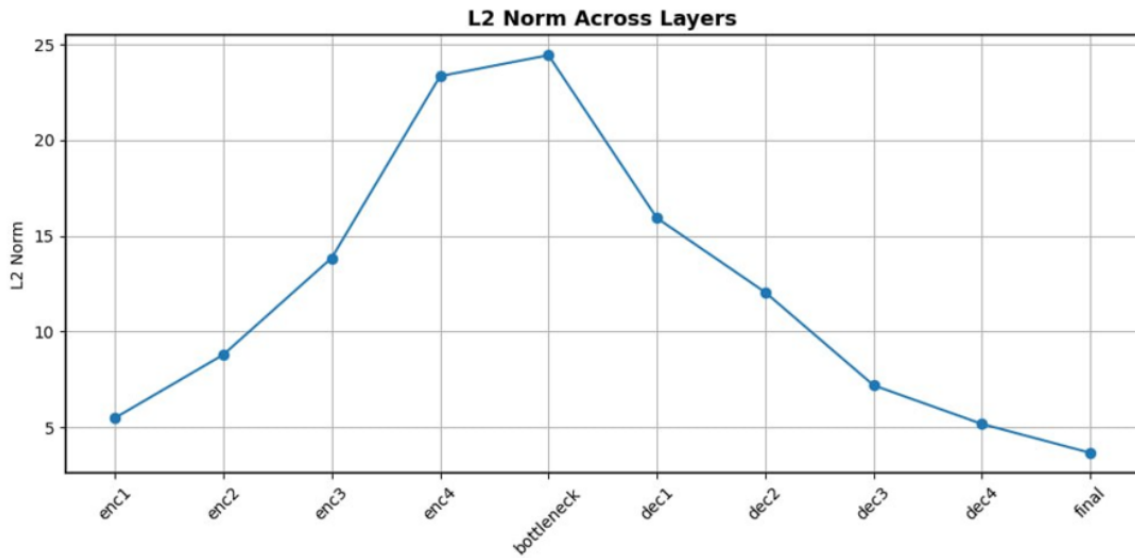


Figure 4-6: L2 Norm Across Layers

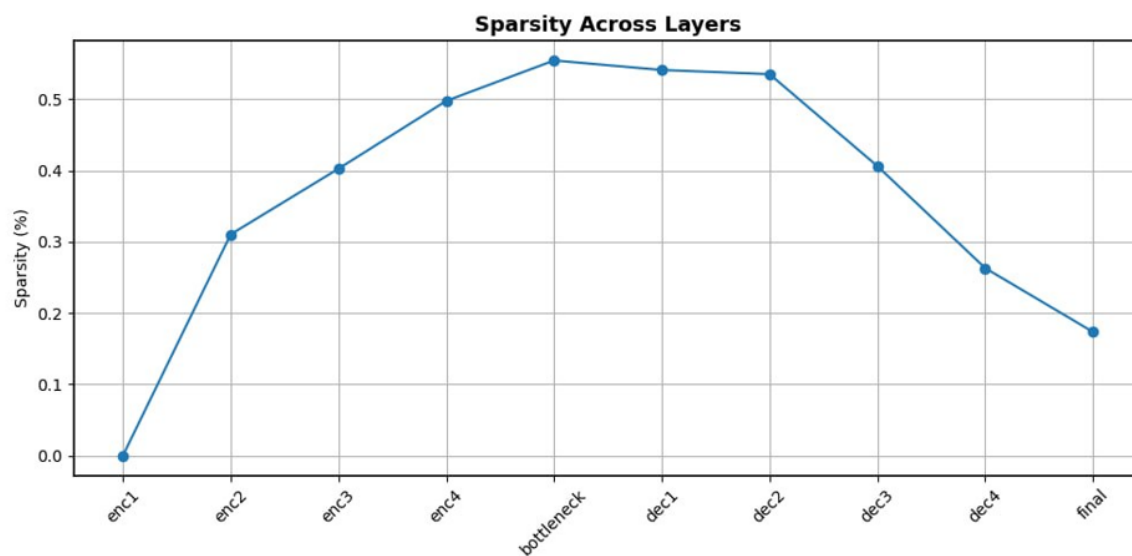


Figure 4-7: Sparsity of the Data

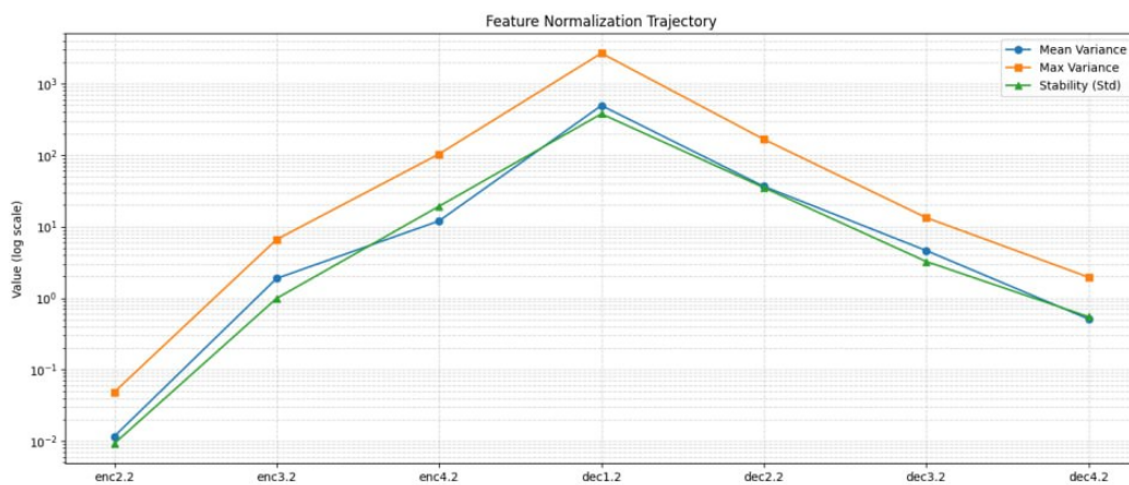


Figure 4-8: Feature Normalization Trajectory

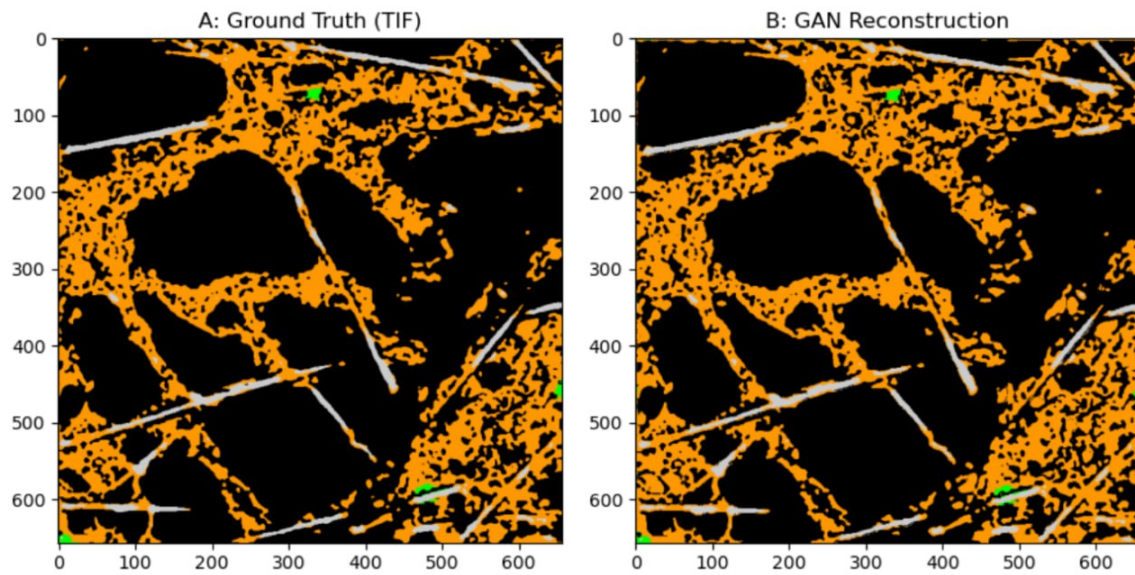


Figure 4-9: Recreation from Ground Truth

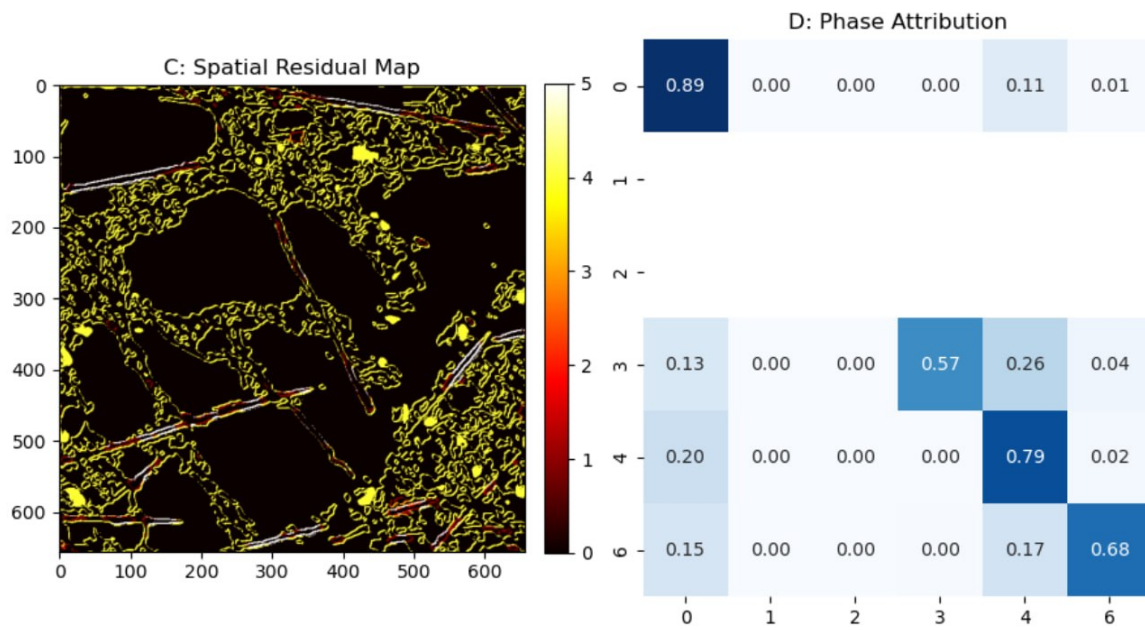


Figure 4-10: Residual Map and Phase Attribution

Chapter 5

Conclusion

5.1 Addressing Key Objectives

The model has demonstrated substantial results of being able to train on a large dataset, and to generate multiple datasets that are varying and have no overfitting problem. However, once it hits the accuracy level, the generation part is considerably swift. From both visual and physical perspective, GAN model has managed to take into account the complex structure of carbon fibers to mimic their train-like movement, and it is completely random at the generation part.

This gives significant advantage to researchers that will have no need to find and scan numerous fuel cell CTs. The main aim of this project was to save time and give advantage to the scientists as the energy demand is being reconsidered in modern world.

Firstly, it is usage of GAN framework for development of physically realistic 3D samples. While it is a stack of 3D images, it is still functional as actual 3D. Physical parameters were also achieved by programming the behavior during the training process.

Secondly, the physical features of carbon fiber and binder were fully represented visually. Carbon fiber is moving in a specified angle with no breaks and discontinuity. While binder is keeping being smooth and foamy around fiber lines without chaotic distribution or collapse.

Thirdly, behavior of binder and carbon should also be linked to each other not just physical parameters in a vacuum. Therefore, binder is forming around carbon fiber distribution and high concentration areas rather than forming idly.

Fourthly, by keeping it as a stack of 2D images, and every subsequent image looking at the previous one as a reference point, it has helped to address a common problem in GAN that it is difficult to maintain a Z axis continuity, but this model consistently shows that the elements are smooth in 3D across all axis.

Overall, the main objectives of converging physical parameters and highly efficient model training were achieved.

5.2 Future Work

While the objectives of recreating and simulating 3D slices with utilizing a stack of 2D slices was achieved. There is still a room for further development since the increase in computational power will give opportunity to train on full on 3D convolutional layers that could make it easier to get a more accurate overall picture rather than enforcing Z axis continuity. It should also be noted that 2D stacking also created a limitations that there is a subtle roughness and sharpness of edges compared to original dataset.

It does not affect the physical parameters by with a big margin, however, future work and implementation of 3D GAN would help to smooth across the Z axis. Additionally, a limitation was to only consider carbon fiber and binder due to the complexity of the visuals. However, the future work should also include catalyst layer and membrane to make a full picture of PEMFCs. This means that the physical parameters should also be readjusted and integrate numerical data possibly from csv files during the training process so that the physical parameters would be the integral part for maximum physical realism.

Bibliography

- [1] C. Chuaypat, S. Bardeeniz, M. A. Hussain, and C. Panjapornpon, “Energy efficiency modeling considered chemical process anomalies using contrastive learning-guided generative adversarial imputation network for operation-aligned data reconstruction,” *Digital Chemical Engineering*, vol. 19, p. 100296, 2026.
- [2] U. M. Nabila, L. Lin, X. Zhao, W. L. Gurecky, P. Ramuhalli, and M. I. Radaideh, “Data efficiency assessment of generative adversarial networks in energy applications,” *Energy and AI*, vol. 20, p. 100501, 2025.
- [3] K. Tang, Q. Meyer, R. White, R. T. Armstrong, P. Mostaghimi, Y. Da Wang, S. Liu, C. Zhao, K. Regenauer-Lieb, and P. K. M. Tung, “Deep learning for full-feature x-ray microcomputed tomography segmentation of proton electron membrane fuel cells,” *Computers and Chemical Engineering*, vol. 161, p. 107768, 2022.
- [4] N. Morizet *et al.*, “Time to market reduction for hydrogen fuel cell stacks using generative adversarial networks,” *Journal of Power Sources*, 2023.
- [5] L. Zhu, B. Bijeljic, and M. J. Blunt, “Generation of pore-space images using improved pyramid wasserstein generative adversarial networks,” *Advances in Water Resources*, vol. 190, p. 104748, 2024.
- [6] J. Schmidhuber, “Learning factorial codes by predictability minimization,” *Neural Computation*, vol. 4, no. 6, pp. 863–879, 1992.
- [7] Y. Bian, J. Wang, J. J. Jun, and X.-Q. Xie, “Deep convolutional generative adversarial network (dcgan) models for screening and design of small molecules targeting cannabinoid receptors,” *Molecular Pharmaceutics*, vol. 16, no. 11, pp. 4451–4460, 2019. PMID: 31589460.
- [8] X. Qiu, S. Wang, and K. Chen, “A conditional generative adversarial network-based synthetic data augmentation technique for battery state-of-charge estimation,” *Applied Soft Computing*, 2023.
- [9] R. S. Kharat *et al.*, “Thermal analysis of fuel cells using gans and reinforcement learning,” *Thermal Science and Engineering Progress*, 2024.
- [10] X. Liu *et al.*, “3d generation and reconstruction of fuel cell microstructures using 2d images,” *Journal of Power Sources Advances*, 2022.

- [11] S. Liu *et al.*, “Fault detection for power batteries using gan-cnn-lstm,” *Applied Sciences*, 2025.
- [12] Y. Ma *et al.*, “Generative-enhanced physics-informed neural network for energy systems,” *IEEE Sensors Journal*, 2025.
- [13] Z. Niu *et al.*, “Generative ai for designing hydrogen fuel cell nanostructures,” *ACS Nano*, 2024.
- [14] Y. D. Wang *et al.*, “Large-scale physically accurate pem fuel cell modeling,” *Nature Communications*, 2023.
- [15] S. Kench *et al.*, “Li-ion battery design through microstructural optimization,” *Matter*, 2024.
- [16] F. Bai *et al.*, “3d digital twin and upscaling strategy for energy system modeling,” *Applied Energy*, 2024.
- [17] T. Testasecca *et al.*, “Toward a digital twin for solid oxide fuel cells,” *Energies*, 2024.
- [18] Y. Niu *et al.*, “Deep learning methods for super-resolution carbonate rock images,” *Transport in Porous Media*, 2022.
- [19] Y. D. Wang *et al.*, “Characterization of heterogeneous surface wetting,” *Advances in Colloid and Interface Science*, 2024.
- [20] B. U. Mahmud *et al.*, “Deep learning-based segmentation of 3d volumetric images,” *Sensors*, 2023.
- [21] Y. Yang *et al.*, “Microstructure evolution of lithium-ion battery electrodes,” *Electrochemistry Communications*, 2022.
- [22] S. Tamilarasan, C.-K. Wang, Y.-C. Shih, and Y.-D. Kuan, “Generative adversarial networks for stack voltage degradation and rul estimation in pemfcs under static and dynamic loads,” *International Journal of Hydrogen Energy*, vol. 89, pp. 66–83, 2024.
- [23] M. Bode, M. Gauding, D. Goeb, T. Falkenstein, and H. Pitsch, “Applying physics-informed enhanced super-resolution generative adversarial networks to turbulent premixed combustion and engine-like flame kernel direct numerical simulation data,” *Proceedings of the Combustion Institute*, vol. 39, no. 4, pp. 5289–5298, 2023.
- [24] F. Liu, Y. Yang, H. Yang, L. Tao, Y. Tao, K. Zhang, H. Sun, L. Zhang, J. Zhong, and J. Yao, “Pore structure properties characterization of shale using generative adversarial network: Image augmentation, super-resolution reconstruction, and multi-mineral auto-segmentation,” *Petroleum Exploration and Development*, vol. 52, no. 5, pp. 1262–1274, 2025.

- [25] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb, “Learning from simulated and unsupervised images through adversarial training,” 2017.