Development of 3D Generative Adversarial Networks for Topology Optimization

A project report submitted in partial fulfilment of the requirements for the degree of B.Tech. in Smart Manufacturing

by

Rohan Madhav Shingre
MSM19B012

DEPARTMENT OF MECHANICAL ENGINEERING
INDIAN INSTITUTE OF INFORMATION TECHNOLOGY,
DESIGN AND MANUFACTURING, KANCHEEPURAM

May 2023

Certificate

I, **Rohan Madhav Shingre**, with Roll No: **MSM19B012** hereby declare that the material presented in the Project review report titled **Development of 3D Generative Adversarial Networks for Topology Optimization** represents original work carried out by me in the **Department of Mechanical Engineering** at the **Indian Institute of Information Technology, Design and Manufacturing, Kancheepuram** during the year **2023**. With my signature, I certify that:

- I have not manipulated any of the data or results.
- I have not committed any plagiarism of intellectual property. I have clearly indicated and referenced the contributions of others.
- I have explicitly acknowledged all collaborative research and discussions.
- I have understood that any false claim will result in severe disciplinary action.
- I have understood that the work may be screened for any form of academic misconduct.

Date: 07-05-2023 Student's Signature

In my capacity as supervisor of the above-mentioned work, I certify that the work presented in this Project review report is carried out under my supervision, and is worthy of consideration for the requirements of project work during the period January 2023 to May 2023.

Advisor's Name: Dr. Senthilkumaran K. Advisor's Signature

Abstract

Topology optimization (TO) is a computational method used in engineering to design and optimize the shape or topology of a structure or system in order to meet certain performance criteria. The goal of topology optimization is to find the best configuration of material for a given design space, loads, and constraints while minimizing the overall weight or volume of the structure.

However, the traditional topology optimization process often produces designs that are limited by the user's expertise and creativity. In addition, the optimization process can be computationally expensive, especially for large and complex structures.

Generative Adversarial Networks (GANs) have emerged as a promising approach to automate and accelerate the topology optimization process. GANs can generate new designs by learning from existing designs, which can potentially overcome the limitations of traditional optimization approaches. However, the integration of GANs with topology optimization poses several challenges.

Firstly, the GAN needs to be trained on a diverse set of examples to ensure that it produces smooth and innovative designs. Secondly, GANs are notoriously difficult to train and can be unstable, particularly when applied to complex engineering problems like topology optimization. Ensuring the training process is stable is crucial to achieving good results. Lastly, TO often involves incorporating physical constraints, such as design rules. It can be challenging to integrate these constraints into the GAN training process effectively.

Thus, we aim to develop a TO-based GAN framework to add constraints that can generate designs that are both optimal and feasible while overcoming the challenges associated with the integration of GANs and topology optimization. We propose a new approach to implement TO using GAN.

Acknowledgements

I am glad to express my deep gratitude to my institute guide, Dr. Senthilkumaran K., for providing me with an opportunity to work on a novel idea for a project. He is an excellent mentor who has given me ample time and sufficient space to work on my project. He encourages me to complete my work more thoroughly and efficiently. A special thanks to Mr. Jayakrishnan, Ph.D. Scholar, for guiding me through the entire process.

I would like to extend my gratitude to IIITDM Kancheepuram for providing a project experience for all the students. This has aided my personal as well as academic development. A special thanks to Prof. M.V. Karthikeyan, Director for giving us an opportunity to work under respected Professors. I would also like to thank Dr. K. Jayabal, Head of the Department of Mechanical Engineering, for providing motivation to carry out design project work.

I would like to thank all the review members of the panel Dr. K. Senthilkumaran, Dr. S. Jayavel, Dr. S. Karthick, and Dr. N. Rino Nelson, for their feedback regarding my project work. I would also like to thank the Open-Source Communities for providing the required resources. A special thanks to Mr. Divyam Yadav for supporting me throughout my project work. Lastly, I would like to thank my friends and family for supporting me mentally as well as financially during my project work.

Contents

List of Figures

List of Tables

Abbreviations

3D	**Three D**imensional
AM	**A**dditive **M**anufacturing
cGAN	**C**onditional **G**enerative **A**dversarial **N**etwork
ESO	**E**volutionary **S**tructural **O**ptimization
FEA	**F**inite **E**lement **A**nalysis
GANs	**G**enerative **A**dversarial **N**etworks
IoU	**I**ntersection **o**ver **U**nion
LeakyRELU	**Leaky Re**ctified **L**inear **U**nit
NN	**N**eural **N**etworks
RELU	**Re**ctified **L**inear **U**nit
SELTO	**S**ample-**E**fficient **L**earned **T**opology **O**ptimization
SELU	**S**caled **E**xponential **L**inear **U**nit
SIMP	**S**olid **I**sotropic **M**aterial with **P**enalization
TO	**T**opology **O**ptimization
TOuNN	**T**opology **O**ptimization **u**sing **N**eural **N**etworks
VAE	**V**ariational **A**uto**E**ncoder

Symbols

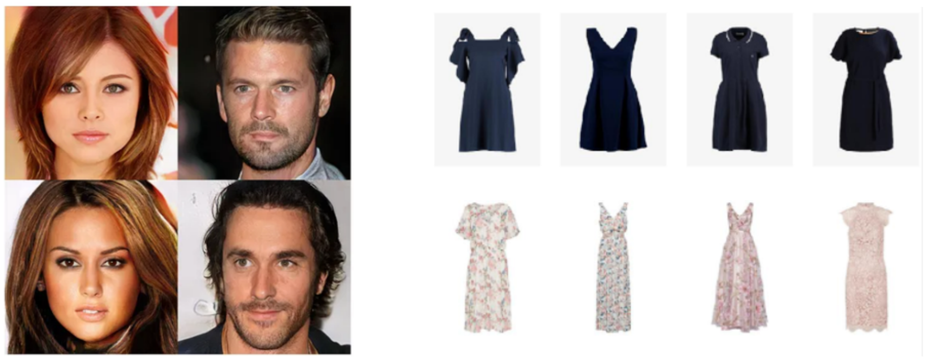
$D()$	Discriminator function
$G()$	Generator function
u	Displacement field
K	Stiffness matrix
f	Applied force
ρ_e	Density associated with the finite element e
v_e	Volume of the element
V^*	Prescribed volume

Dedicated to IIITDM Kancheepuram

Chapter 1

Introduction

Topology Optimization (TO) using Generative Adversarial Networks (GANs) is a relatively new research area that combines the principles of topology optimization and generative adversarial networks to optimize the design of structures or materials. Topology optimization aims to find the optimal distribution of material within a given design space to achieve specific performance criteria, such as minimizing weight or maximizing stiffness. Using GANs in topology optimization involves training a generator network to produce optimized designs that meet certain performance criteria, while a discriminator network evaluates the quality of the generated designs. The generator and discriminator networks are trained together in an adversarial manner, with the generator trying to produce designs that the discriminator cannot identify as generated, and the discriminator trying to distinguish between real and generated designs. Figure 1.1 shows some applications of GANs in real life.



a) Faces generated by GAN [1]   b) GAN in fashion [2]

FIGURE 1.1: Applications of GANs

1.1 Problem Statement

As discussed earlier, designs produced by the traditional topology optimization process may be constrained by the designer's skills and imagination. Moreover, this process can be computationally intensive, particularly for larger and more intricate structures. Our solution is to use a GAN-based topology optimization framework to add constraints that can generate multiple optimized outputs with less number of iterations. Another advantage of using GAN is that it produces designs that require less post-processing.

1.2 Objectives of the project

The aim of our project is:

- To develop a GAN-based TO algorithm for 3D models.
- To reduce the computational power required for optimization.
- To generate multiple outputs.

Chapter 2

Literature Survey

2.1 Topology Optimization

Topology optimization is an optimization approach that improves the arrangement of the material inside a specified space using a few boundary conditions, loads, or restrictions [1, 2, 3]. TO helps reduce the compliance of a part/model. Figure 2.1 is an example of topology optimization.

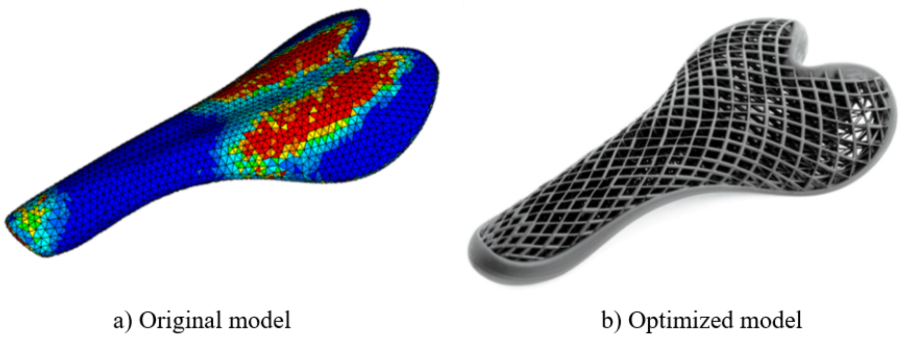


FIGURE 2.1: An example of Topology Optimization [4]

The Solid Isotropic Material with Penalization (SIMP) approach and the Evolutionary Structural Optimization (ESO) method are two widely utilized TO techniques. The most effective structure is achieved using the SIMP approach, which varies the material density. The SIMP approach uses Finite Element Analysis (FEA) for optimization. Contrarily, the ESO approach repeatedly discards material that appears ineffective. A theoretically calculable rejection criterion is used to identify this inefficiency [5].

2.2 Generative Adversarial Networks

Generative Adversarial Networks (GANs) are a type of deep learning model that consists of two neural networks: a generator and a discriminator. The generator network is trained to generate new data samples that are similar to a given training set, while the discriminator network is trained to distinguish between the generated data and real data from the training set. The framework mentioned here can be seen in Figure 2.2.

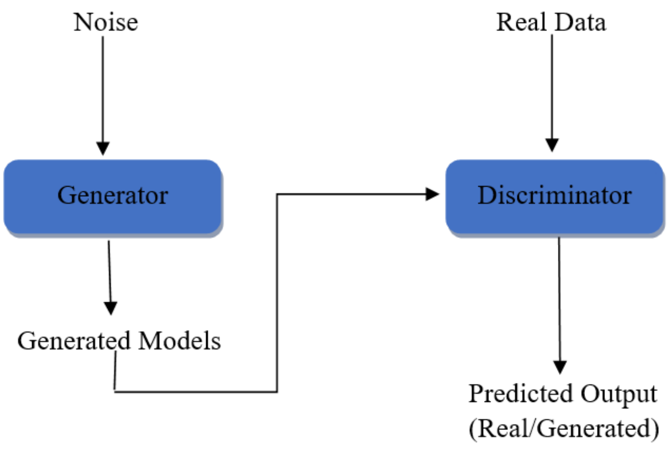


FIGURE 2.2: Structure of GAN

During training, the generator generates new data samples by mapping random noise vectors into the data space, while the discriminator tries to classify whether a given data sample is real or generated. The generator and discriminator are trained together in an adversarial manner, with the generator trying to produce samples that the discriminator cannot identify as generated, while the discriminator tries to accurately distinguish between real and generated samples [6]. Thus, GAN is a deep-learning framework that is based on unsupervised learning.

Despite being a relatively new technology, a lot of study has been done on GANs. The notion of GAN and some of its designs, for instance, are explained by [7]. We are utilizing the Python programming language to implement GAN for this project. [8] has offered a simple GAN implementation in Python.

2.2.1 Use of GAN for 2D images

The application of GANs to 2D images has become widely adopted in many fields. In the aerospace industry, [9] demonstrated the use of a conditional GAN (cGAN) to optimize airfoil designs. In another article published by [10] in 2020, GANs were applied to deblurring images with impressive accuracy. These examples highlight the versatility of GANs and their potential to bring about significant changes in various domains.

2.2.2 Use of GAN for 3D models

In comparison to the abundance of research on GANs for 2D image generation, there is relatively less published research on using GANs to generate 3D models[11]. [12] has proposed a framework called 3D-GAN that uses GANs to generate 3D models by creating a latent representation of the model in a random probabilistic space. Figure 2.3 shows the structure of the generator in 3D-GAN. This work has been further developed into 3D-VAE-GAN, which incorporates photogrammetry to convert 2D images into a latent space representation. These concepts have become fundamental to 3D model generation and have been applied to various projects. The implementation of 3D-GAN and 3D-VAE-GAN can be found in [13]'s GitHub repository.

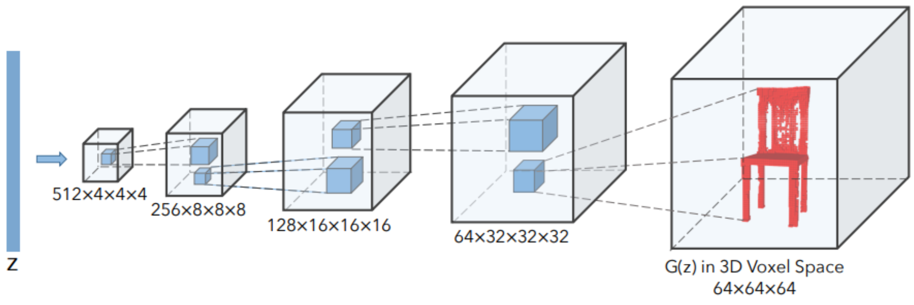


FIGURE 2.3: Structure of the Generator in 3D-GAN [12, 14]

[15] has created an interactive 3D model design application using the 3D-GAN framework, aimed at helping novice designers. Users are provided with a voxel editor to create simple models using blocks, and the GAN algorithm generates a new model based on the user's input. The user can make changes to the generated model and run the algorithm again to get a new output with updated requirements, enabling an iterative process that produces innovative designs according to the user's specifications.

In a separate 2019 article, [16] demonstrated the use of GANs for 3D model generation by replicating historical monuments from a single 2D image. Their method involves converting the 2D image into voxels and then creating a model from these voxels.

2.3 Machine Learning with Topology Optimization

2.3.1 Conditional GAN for Additive Manufacturing

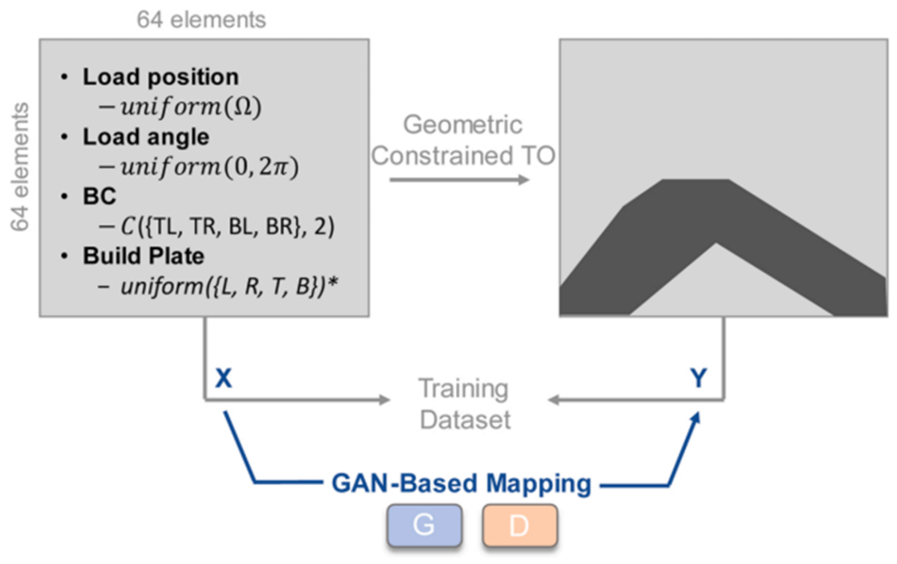


FIGURE 2.4: Structure of cGAN for TO [17, 18]

There is limited research on the use of GAN for Additive Manufacturing (AM). One such article was published in 2021 [17] on applying conditional GAN (cGAN) to topology optimization for AM models. cGAN allows the network to train with labeled data by enabling input of custom parameters for training the model.

The structure of the cGAN used in the article is depicted in Figure 2.4, and the labeled input data consists of four parameters: load position, load angle, boundary conditions, and build plate orientation. The article stores random values of these parameters in a set X and one optimized solution for these variables in set Y, which is used to train the GAN.

2.3.2 Topology Optimization using Neural Networks

Topology Optimization using Neural Networks (TOuNN) is a paper that proposes a new method for topology optimization, which is the process of finding the optimal material layout for a given design space. [19] The authors propose a new neural network architecture that is designed to learn and generate optimal material layouts for a given design problem. The network is trained using a combination of supervised and unsupervised learning techniques, and the resulting optimized designs are shown to outperform traditional optimization methods. Figure 2.5 shows the framework of TOuNN. Here, the neural network gives the density as output which is then used as an input for Finite Element Analysis (FEA). Further optimization is performed and finally, the required output is generated.

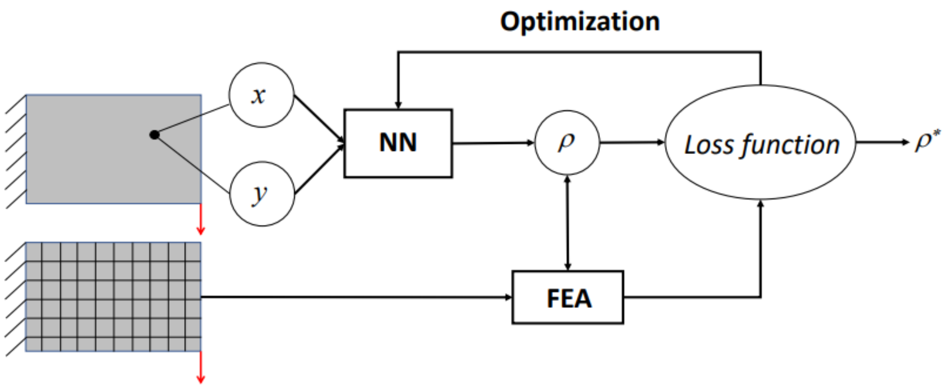


FIGURE 2.5: Overview of the TOuNN framework in 2D [19]

The main idea behind the TOuNN method is to use neural networks to learn the underlying physics of the design problem and use this knowledge to generate optimal material layouts. Their algorithm takes a density field as input that represents the material distribution in the design domain. The output of the network is a binary mask that indicates which parts of the domain should be filled with material and which should be empty.

2.3.3 Topology GAN

TopoGAN, another published paper, proposes a new GAN architecture for generating 3D shapes with certain topological features. [20] Specifically, TopoGAN is designed to generate shapes with a given number of holes or handles. The authors achieve this by introducing a novel loss function that penalizes the GAN for generating shapes that do not meet the desired topological constraints. They also use a technique called mesh-based convolution to enable the GAN to operate directly on the 3D mesh representation of the shapes, rather than using a voxel grid.

2.3.4 SELTO

Sample-Efficient Learned Topology Optimization (SELTO) focuses on the impact of including physics in deep learning techniques. [21] They use U-Net (a U-shaped deep learning framework) to implement topology optimization. A sample architecture of

U-Net is shown in figure 2.6. Then they include physics into the U-Net and compare the results of both the outputs with the ground truth. Their results indicate that including physics in deep learning techniques significantly improves the results. Their results can be seen in figure 2.7 where IoU (Intersection over Union) is a metric for accuracy. It can be seen that by including physics the outputs are very accurate. Finally, they publish four datasets as there is a lack of training data for topology optimization using deep learning.

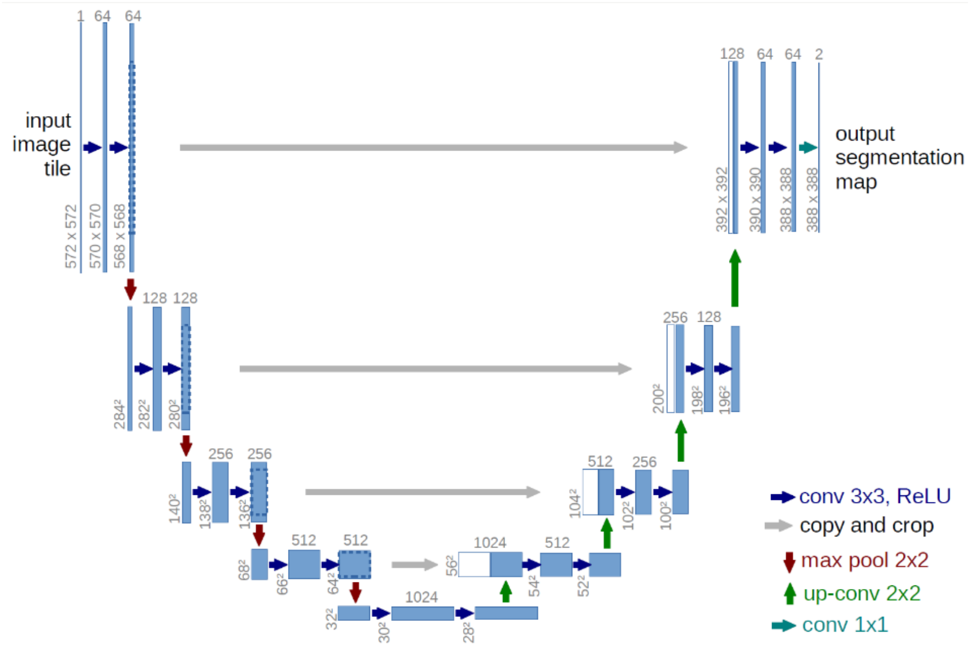


FIGURE 2.6: Structure of U-Net [22]

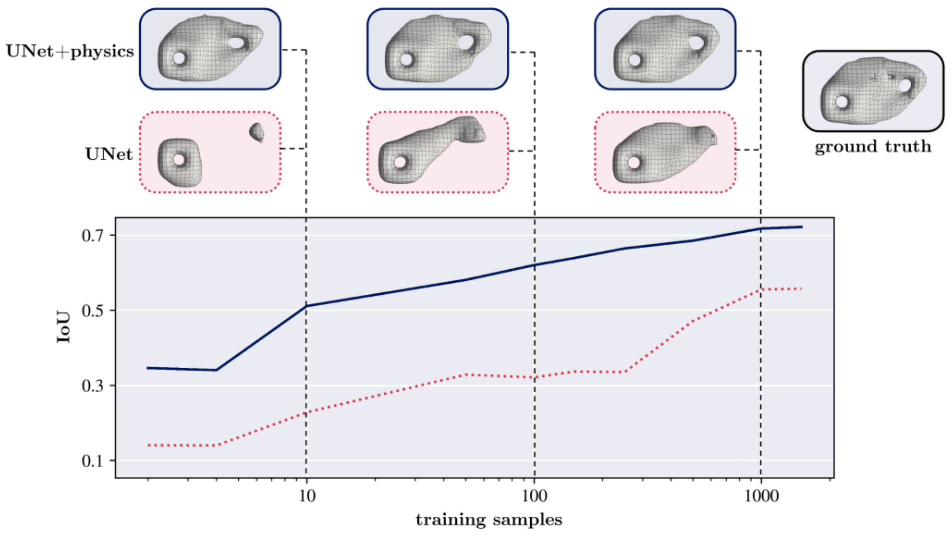


FIGURE 2.7: Results of SELTO [21]

2.3.5 Summary

Table 2.1 shows a summary that explains the key points of each paper and the need for GANs in TO.

Table 2.1: Summary of literature highlighting why we need to implement GAN for TO

S.No.	Title	Key Points	Need for GANs in TO
1	Generative Adversarial Network for Early-Stage Design Flexibility in Topology Optimization for Additive Manufacturing[17]	Uses GANs to generate design candidates that are feasible and flexible. Reduces design time and cost in additive manufacturing.	Traditional topology optimization methods can be computationally expensive and may not produce practical designs. GANs offer a way to generate feasible and flexible designs more efficiently.
2	Tounn: Topology Optimization Using Neural Networks[19]	Uses a convolutional neural network to learn the optimal design directly from the input boundary conditions. Produces efficient and practical designs.	Traditional topology optimization methods may not produce practical designs and can be computationally expensive. Neural networks offer a way to learn the optimal design directly from the input boundary conditions, resulting in more efficient and practical designs.
3	TopologyGAN: Topology Optimization Using Generative Adversarial Networks Based on Physical Fields over the Initial Domain[20]	Uses GANs to generate design candidates that satisfy physical field constraints over the initial domain. Produces efficient and practical designs for complex physical systems.	Traditional topology optimization methods may not be able to handle complex physical systems and boundary conditions and can be computationally expensive. GANs offer a way to generate efficient and practical designs that satisfy physical field constraints for complex physical systems.
4	SELTO: Sample-Efficient Learned Topology Optimization[21]	Uses a combination of deep reinforcement learning and Bayesian optimization to learn the optimal topology efficiently. Achieves state-of-the-art results with fewer computational resources.	Traditional topology optimization methods may require a large number of simulations to find the optimal topology, which can be computationally expensive. SELTO offers a way to learn the optimal topology with fewer computational resources.

Chapter 3

Methodology

Every GAN algorithm has 2 neural networks- a generator and a discriminator. We will use the TOuNN approach to design the Generator so that it will generate optimized models with less number of iterations.

3.1 Implemention of a GAN algorithm

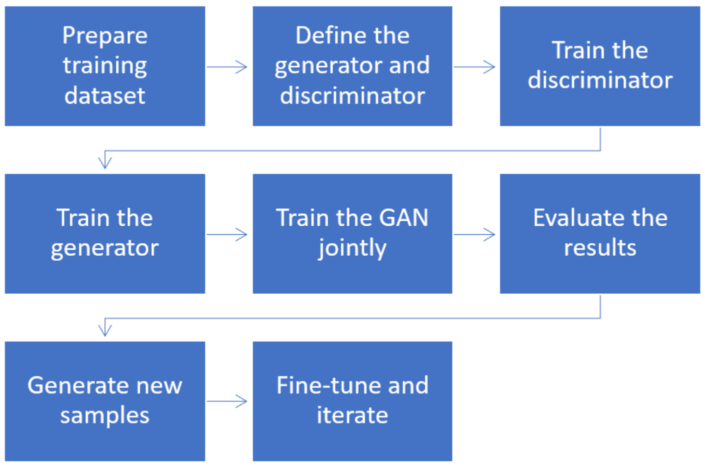


FIGURE 3.1: Steps to implement GAN

Implementing a GAN is difficult and it poses many challenges. This is how we will prepare our GAN algorithm (refer Figure 3.1):

1. Prepare training dataset: The first step is to collect or prepare the data that will be used to train the GAN. We are using the open-source tool, TopOpt [23], to generate

2D Topologically Optimized examples for the training and testing of our algorithm. For the 3D training dataset, we use the SELTO [21] dataset.

2. Define the generator and discriminator architectures: The next step is to define the architectures of the generator and discriminator neural networks. These networks can be designed and should be able to learn from the training data. For the 2D generator, we are using the TOuNN [19] approach. For the 3D generator and discriminator, we use the [17] approach with a few modifications as it is similar to our work.
3. Train the discriminator: The discriminator is trained using the training data to distinguish between real and generated samples generated by the generator. This process involves optimizing the discriminator's weights to minimize the loss function.
4. Train the generator: The generator is then trained to generate realistic samples that can fool the discriminator. This process involves optimizing the generator's weights to minimize the loss function.
5. Train the GAN jointly: The generator and discriminator are trained jointly, with the generator trying to create samples that can fool the discriminator, while the discriminator tries to correctly classify the samples as real or fake.
6. Evaluate the results: After training, the GAN's performance can be evaluated using metrics such as the discriminator's accuracy and the generator's ability to create realistic samples.
7. Generate new samples: The trained generator can be used to generate new samples that are similar to the training data but not identical to any specific sample.
8. Fine-tune and iterate: If the GAN does not perform well, fine-tuning and iterating the model architecture, training data, and training process can be done until the desired results are achieved.

3.2 Structure of GAN

3.2.1 Structure of 2D GAN

We have 4 layers in the generator:

- A dense layer with 100 nodes and the Scaled Exponential Linear Unit (SELU) activation function.

- Another dense layer with 150 nodes and SELU activation function to further scale the input images.
- Another dense layer with 28x28x4 nodes which are the dimensions of the input and the Sigmoid activation function to convert the output between 0 to 1.
- Finally, a reshape layer to convert the input array back to the original dimensions.

Similar to the generator, we have 4 layers in the discriminator:

- A flatten layer that takes the image as input.
- Another dense layer with 150 nodes and SELU activation function.
- A dense layer with 100 nodes and the SELU activation function.
- Finally, a dense layer with 1 node and the sigmoid activation function to scale the input between 0 to 1. If it is less than 0.5 then the discriminator returns 0 (generated model), else it returns 1 (real model).

Additionally, we use a batch size of 5 and the number of epochs as 50 while training the GAN. We use the gradient descent approach to reduce the losses generated by GAN.

3.2.2 Architecture of 3D GAN

We are following the GAN architecture of [17] with modifications as our approach is used for 3D models. Figure 3.2 shows the architecture of the 3D generator and discriminator.

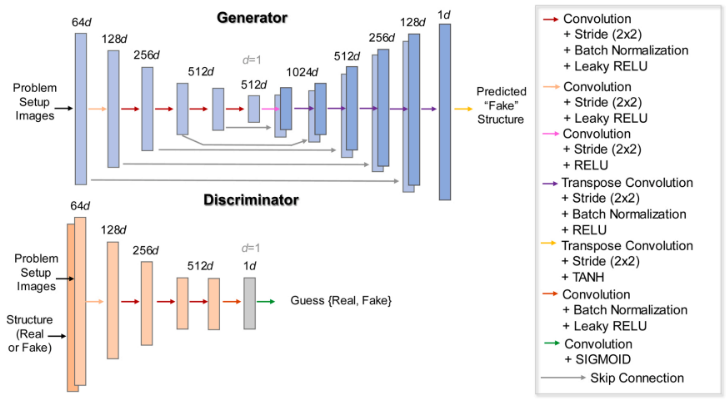

Generator
64d
128d
256d
512d
d=1
512d
1024d
512d
256d
128d
1d
Problem Setup Images
Predicted "Fake" Structure
Discriminator
64d
128d
256d
512d
d=1
1d
Problem Setup Images
Structure (Real or Fake)
Guess {Real, Fake}
Convolution + Stride (2x2) + Batch Normalization + Leaky RELU
Convolution + Stride (2x2) + Leaky RELU
Convolution + Stride (2x2) + RELU
Transpose Convolution + Stride (2x2) + Batch Normalization + RELU
Transpose Convolution + Stride (2x2) + TANH
Convolution + Batch Normalization + Leaky RELU
Convolution + SIGMOID
Skip Connection

FIGURE 3.2: Architecture of 3D Generator and Discriminator [17]

For our GAN, we have modified the architecture. We have 12 layers in the generator:

- A convolutional layer with 64 nodes and (3,3) strides followed by a LeakyRELU (Leaky Rectified Linear Unit) activation function.
- A convolutional layer with 128 nodes and (1,1) strides followed by batch normalization and LeakyRELU activation function.
- A convolutional layer with 256 nodes and (1,1) strides followed by batch normalization and LeakyRELU activation function.
- Two convolutional layers with 512 nodes and (1,1) strides followed by batch normalization and LeakyRELU activation function.
- A convolutional layer with 512 nodes and (1,1) strides followed by a RELU (Rectified Linear Unit) activation function.
- Two transpose convolutional layers with 1024 nodes and (1,1) strides followed by batch normalization and RELU activation function.
- A transpose convolutional layer with 512 nodes and (1,1) strides followed by batch normalization and RELU activation function.
- A transpose convolutional layer with 256 nodes and (1,1) strides followed by batch normalization and RELU activation function.
- A transpose convolutional layer with 128 nodes and (1,1) strides followed by batch normalization and RELU activation function.
- Finally, a transpose convolutional layer with 4 nodes, (3,3) strides, and tanh (hyperbolic tangent function) activation function.

We have 6 layers in the discriminator:

- A convolutional layer with 64 nodes and (1,1) strides followed by a LeakyRELU (Leaky Rectified Linear Unit) activation function.
- A convolutional layer with 128 nodes and (1,1) strides followed by batch normalization and LeakyRELU activation function.
- A convolutional layer with 256 nodes and (1,1) strides followed by batch normalization and LeakyRELU activation function.
- Two convolutional layers with 512 nodes and (1,1) strides followed by batch normalization and LeakyRELU activation function.

- Finally, a convolutional layer with 1 node and the sigmoid activation function to scale the input between 0 to 1. If it is less than 0.5 then the discriminator returns 0 (generated model), else it returns 1 (real model).

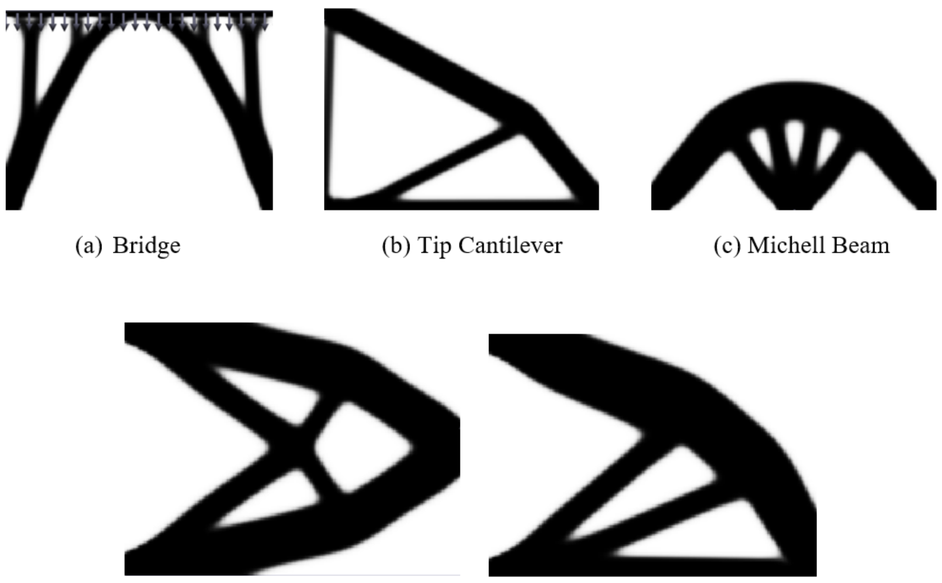
Chapter 4

Development of GAN

4.1 2D GAN

4.1.1 Preparation of Dataset

We have prepared a dataset that includes five models for each of the given model types. We have selected Mid Cantilever, Michell beam, MBB Beam, Bridge, and Tip Cantilever for our training dataset. This dataset was prepared using the TopOpt tool. Figure 4.1 shows one model prepared for each type.

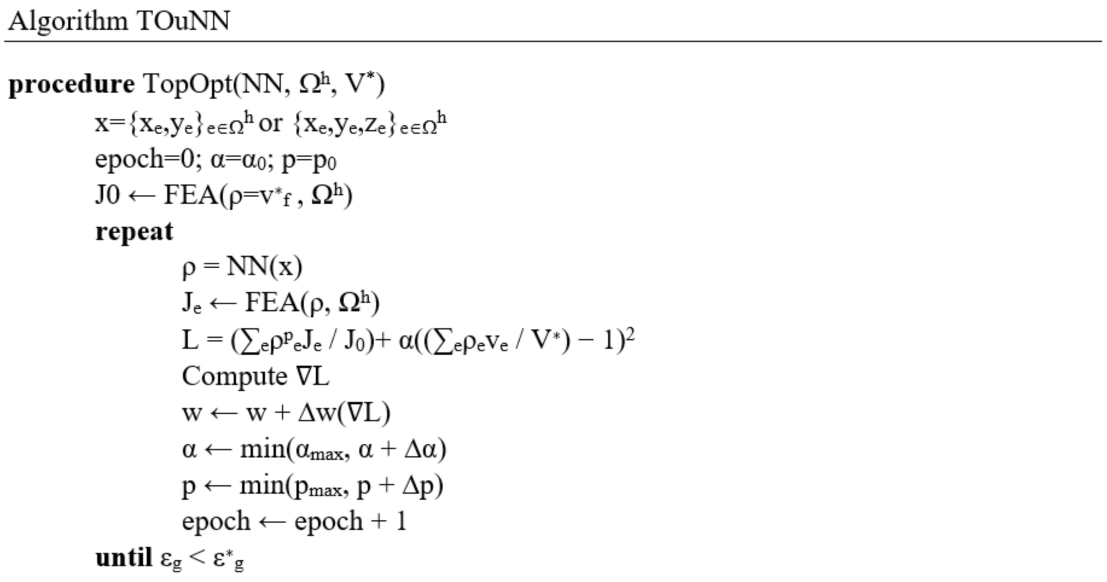


(a) Bridge (b) Tip Cantilever (c) Michell Beam

(d) Mid Cantilever (e) MBB Beam

FIGURE 4.1: Prepared training dataset

4.1.2 Implementation of TOuNN



```
Algorithm TOuNN

procedure TopOpt(NN, $\Omega^h$, $V^*$)
    $x=\{x_e,y_e\}_{e\in\Omega^h}$ or $\{x_e,y_e,z_e\}_{e\in\Omega^h}$
    epoch=0; $\alpha=\alpha_0$; $p=p_0$
    J0 $\leftarrow$ FEA($\rho=v^*_f$ , $\Omega^h$)
    repeat
        $\rho = NN(x)$
        $J_e \leftarrow FEA(\rho, \Omega^h)$
        $L = (\sum_e \rho^p_e J_e / J_0) + \alpha((\sum_e \rho_e v_e / V^*) - 1)^2$
        Compute $\nabla L$
        $w \leftarrow w + \Delta w(\nabla L)$
        $\alpha \leftarrow \min(\alpha_{max}, \alpha + \Delta\alpha)$
        $p \leftarrow \min(p_{max}, p + \Delta p)$
        epoch $\leftarrow$ epoch + 1
    until $\varepsilon_g < \varepsilon^*_g$
```

FIGURE 4.2: TOuNN algorithm [19]

In Figure 4.2, this algorithm is for the topology optimization of a finite element (FE) mesh using a neural network (NN) and a finite element analysis (FEA) solver. The input to the optimization is the coordinates of the center of each element in the FE mesh. The desired output is a density field (ρ) that indicates the presence or absence of material in each element of the mesh, which results in an optimized topology. The algorithm starts by initializing the penalty factor (α) and a variable p, which is used for continuation and calculates the initial objective function (J0) using the FEA solver with a uniform density field. Then, the optimization loop begins. In each iteration, the NN is used to compute the density field (ρ) for the current input. This is followed by solving the FEA problem for the given density field, which results in the objective function value Je. The loss function L is then calculated using Je, J0, and the penalty factor . The gradient of the loss function with respect to the input density field (ρ) is computed using backpropagation. The weights of the NN are then updated using an Adam optimizer with the computed gradient. The continuation variable p and the penalty factor are also updated in each iteration. The optimization continues until a convergence criterion is met. Input factors that are considered are volume fraction, load, boundary conditions, and domain constraint. Equations 4.1, 4.2 and 4.3 explain a classic topology optimization problem.

The topology optimization can be summarized as minimizing ρ subject to:

$$u^T * K(\rho) * u \tag{4.1}$$

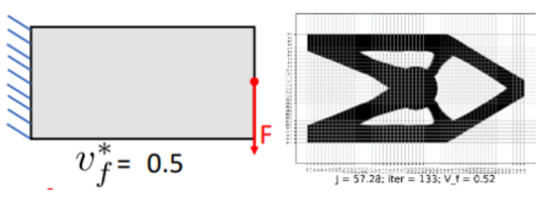
$$K(\rho) * u = f \tag{4.2}$$

$$\Sigma_e \rho_e v_e = V^* \tag{4.3}$$

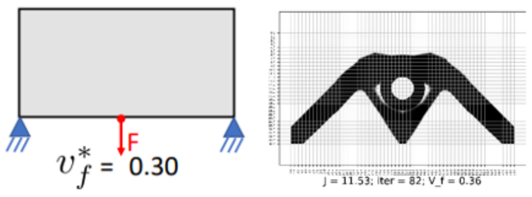
where u is the displacement field, K is the stiffness matrix, f is the applied force, ρ_e is the density associated with the finite element e, v_e is the volume of the element, and V^* is the prescribed volume.

4.1.2.1 Output of TOuNN

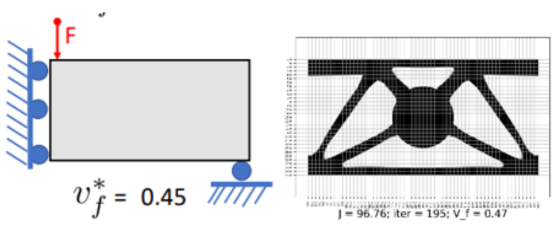
Figure 4.3 shows the outputs generated by TOuNN. An output for each of the give inputs can be seen. The models generated are: Mid Cantilever, Michell beam, MBB Beam, Bridge, and Tip Cantilever. It can be easily seen that the generated results have higher resolution. Table 4.1 shows the values of compliance (J), the number of iterations taken (iter), and the volume fraction (V_f) as the images are not clear.



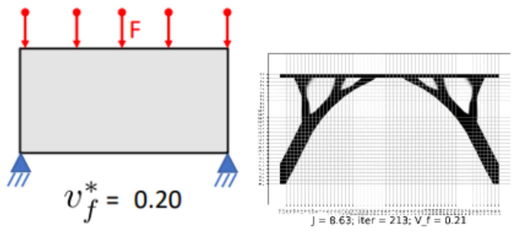
(a) Mid Cantilever



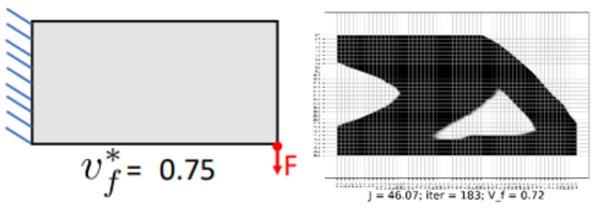
(b) Michell Beam



(c) MBB Beam



(d) Bridge



(e) Tip Cantilever

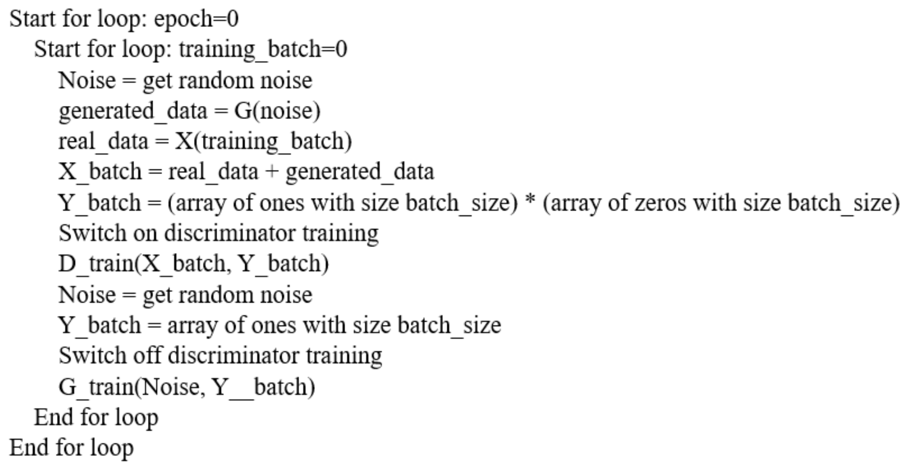
FIGURE 4.3: Inputs [19] and Outputs generated by TOuNN

TABLE 4.1: Comparison of ML approaches for TO

S.No.	Model	Compliance (J)	Iterationss	Volume fraction (V_f)
1	Mid Cantilever	57.28	133	0.52
2	Michell beam	11.53	82	0.36
3	MBB Beam	96.76	195	0.47
4	Bridge	8.63	213	0.21
5	Tip Cantilever	46.07	183	0.72

4.1.3 Implementation of GAN

Figure 4.4 shows a pseudo-code for GAN that works in the following way:



```
Start for loop: epoch=0
  Start for loop: training_batch=0
    Noise = get random noise
    generated_data = G(noise)
    real_data = X(training_batch)
    X_batch = real_data + generated_data
    Y_batch = (array of ones with size batch_size) * (array of zeros with size batch_size)
    Switch on discriminator training
    D_train(X_batch, Y_batch)
    Noise = get random noise
    Y_batch = array of ones with size batch_size
    Switch off discriminator training
    G_train(Noise, Y__batch)
  End for loop
End for loop
```

FIGURE 4.4: Implementation of GAN algorithm

For each epoch in the number of epochs: For each batch in the number of batches in the dataset:

- Generate a set of fake samples by providing random noise to the generator.
- Train the discriminator on both real and generated data by combining the real and fake samples.
- Create a batch of input data (X_batch) by concatenating the real and generated data.
- Create a batch of labels (y_batch) for the input data where the real data has a label of 1 and the generated data has a label of 0.
- Set the discriminator to trainable and train it on the input data (X_batch) and labels (y_batch).

- Generate a set of new noise to provide as an input to the generator to generate fake data.
- Set the labels for the generated data as 1 to trick the discriminator into thinking the generated data is real.
- Set the discriminator to not trainable and train the GAN model on the new noise and labels.

4.2 3D GAN

4.2.1 Preparation of Training Dataset

We use the SELTO dataset [21] for the training of the proposed GAN framework. The dataset contains 4 types of models: disc simple, disc complex, sphere simple, and sphere complex. Figure 4.5 shows the four types of models in the SELTO dataset.

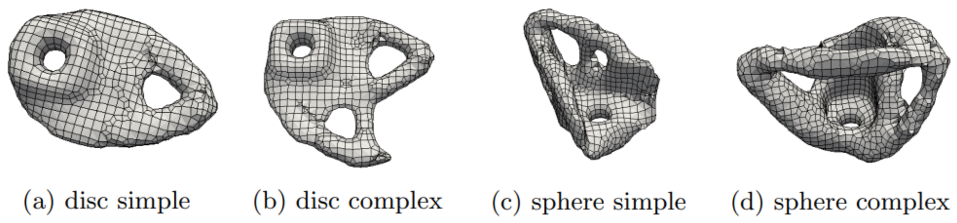


FIGURE 4.5: SELTO dataset [21]

The disc simple dataset contains 1509 training samples and 200 test samples. These samples are optimized discs for a given set of load and boundary conditions. Each sample contains 39x39x4 voxels.

The disc complex dataset contains 7419 training samples and 200 test samples. These samples are optimized discs for a given set of boundary conditions and two points of attack for two load conditions. Each sample contains 39x39x4 voxels.

The sphere simple dataset contains 150 training samples and 38 test samples. These samples are optimized spheres for a given set of load and boundary conditions. Each sample contains 39x39x21 voxels.

The sphere complex dataset contains 378 training samples and 38 test samples. These samples are optimized spheres for a given set of boundary conditions and two points of attack for two load conditions. Each sample contains 39x39x21 voxels.

Each type of dataset contains the following information: the x, y, and z coordinates of the start points of the voxels, the design space indicating the voxels that can be optimized, the Dirichlet boundary conditions indicating the voxels that cannot be modified, the x, y, and z coordinates of the voxels where the force is applied, and finally, the density of each voxel (1 if voxel is present and 0 if absent).

Figure 4.6 helps us visualize the boundary and load conditions. The green part represents the Dirichlet condition that cannot be modified. The blue portion represents the design space that is free to be modified or optimized. The red voxels indicate the point of application of the load. The grey model is the optimized model according to the given boundary and load conditions. Note that the blue, green, and grey parts have the same z coordinates but have been represented in different planes to help visualize the conditions.

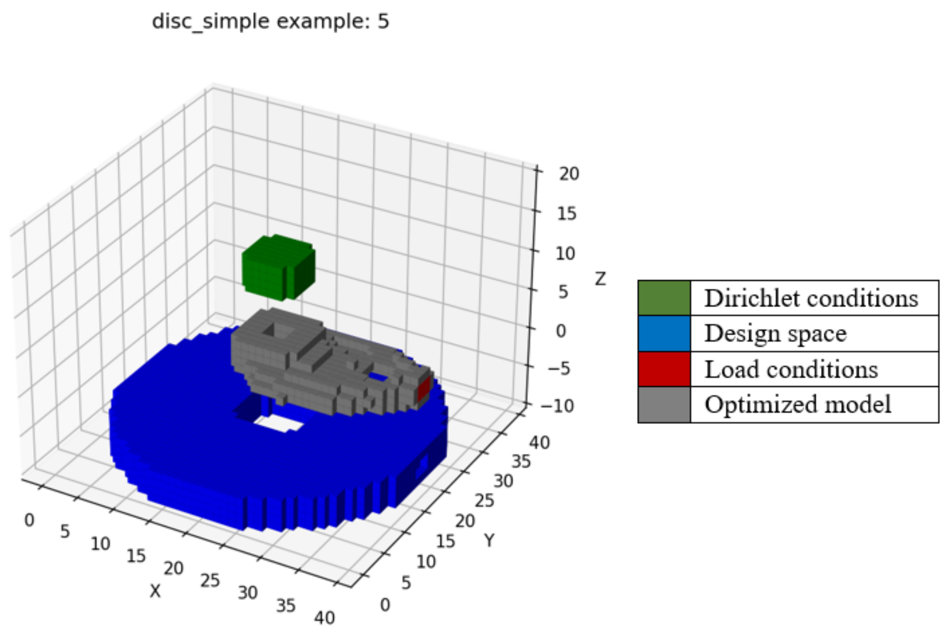


Figure 4.6: Visualizing the boundary conditions

4.2.2 Structure of 3D GAN

Since we are using the SELTO dataset, our labeled data will include the forces and Dirichlet conditions. Figure 4.7 shows the structure of the proposed 3D GAN framework. The generator will be trained with the forces and Dirichlet conditions. It will then produce new outputs with random noise as input. These generated models along with the real models will be passed on to the discriminator for training. The discriminator will then identify whether the model is real or generated.

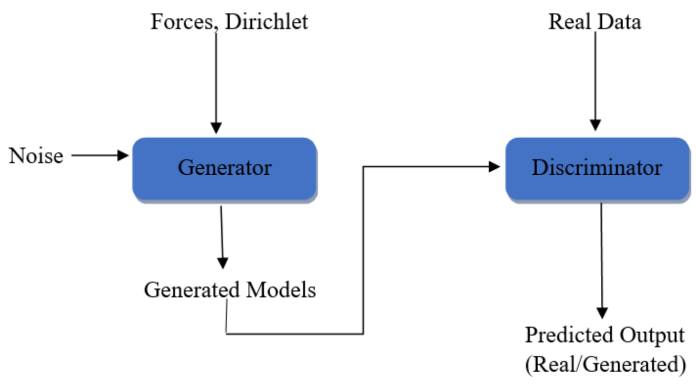


FIGURE 4.7: Structure of the proposed 3D GAN

4.2.3 Training the 3D GAN

For training the 3D GAN, we took the number of epochs as 100 and the batch size as 256. The 3D GAN has been implemented with TensorFlow-a Python library for deep learning [24]. The GAN model has been implemented on Nvidia GeForce GTX 1650 GPU with 8GB RAM. For the loss function and optimizer, we use the binary cross-entropy loss and Adam optimizer [25] with a learning rate of 10^{-4}.

Chapter 5

Results and Discussion

5.1 2D GAN Results

While training the GAN network for our training dataset, we use a loss function. The loss function calculates a value that evaluates the algorithm. If the value is high, then our predictions are not precise and if the value is low, then our algorithm works properly. We calculate the loss for the discriminator as well as the generator. Our aim is to reduce these losses.

Figure 5.1 shows the graph plotted between the discriminator loss and the generator loss. Since we calculate the loss after each training iteration, the total number of iterations (number of batches x number of epochs) is plotted on the X-axis. The loss has been plotted on the Y-axis. Notice how the losses slowly converge. This convergence depicts that the algorithm has been sufficiently trained. Hence, we have successfully trained our algorithm.

5.1.1 Limitations of 2D GAN

Due to a lack of training data, our 2D GAN was unable to produce a viable output. Furthermore, as seen in figure 5.1, the discriminator loss is negligible and almost equal to zero. Whereas, the loss of the generator is comparatively large. This is due to the lack of a training dataset as well because the generator fails to produce a proper output. The discriminator rejects all the models by simply predicting that the models are ”generated”,

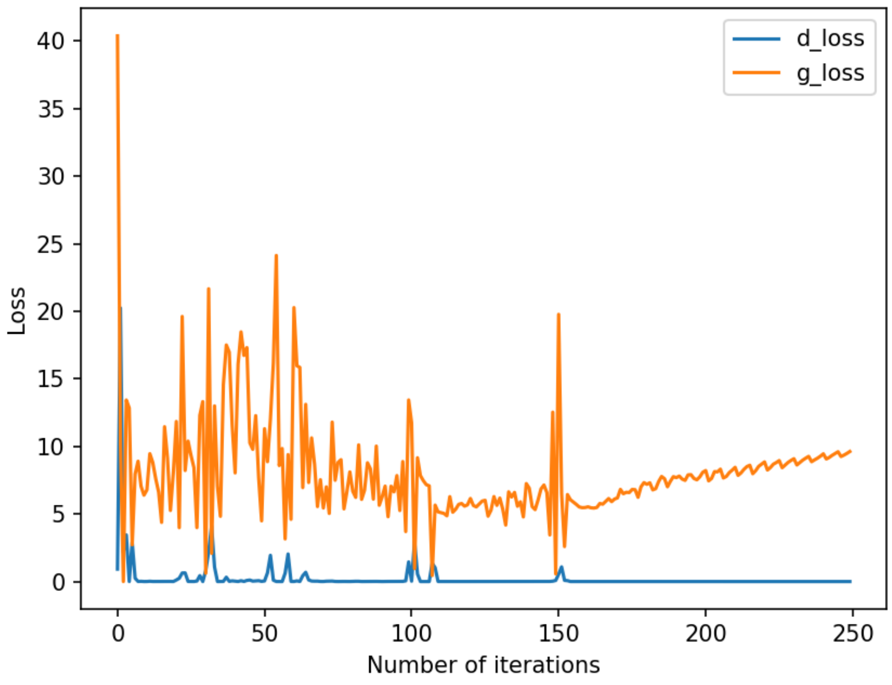


FIGURE 5.1: Plot of losses of 2D generator and discriminator

thereby, reducing its loss. This problem can be fixed by gathering a dataset that will have enough training samples.

5.2 3D GAN Results

Figure 5.2 displays the outputs generated by the 3D GAN for the disc simple dataset. The 2 models on the top left are generated and the model on the right is the ground truth corresponding to the given input conditions. It can be seen that both the generated designs are different and follow the given load and Dirichlet conditions. Similarly, the 2 models at the bottom left are generated and the model on the right is the ground truth.

Figure 5.3 shows the plot of the losses of the generator and the discriminator during the training process. For 3D GAN, we have plotted 3 types of losses: the generator loss (G_loss), the discriminator loss for real models (D_loss_real), and the discriminator loss for generated models (D_loss_fake). Again, the convergence of the losses depicts that our algorithm has been sufficiently trained. Hence, we have successfully trained our algorithm.

Table 5.1 shows a comparison of our 3D GAN approach with other ML approaches. The table also highlights the advantages and disadvantages of each approach.

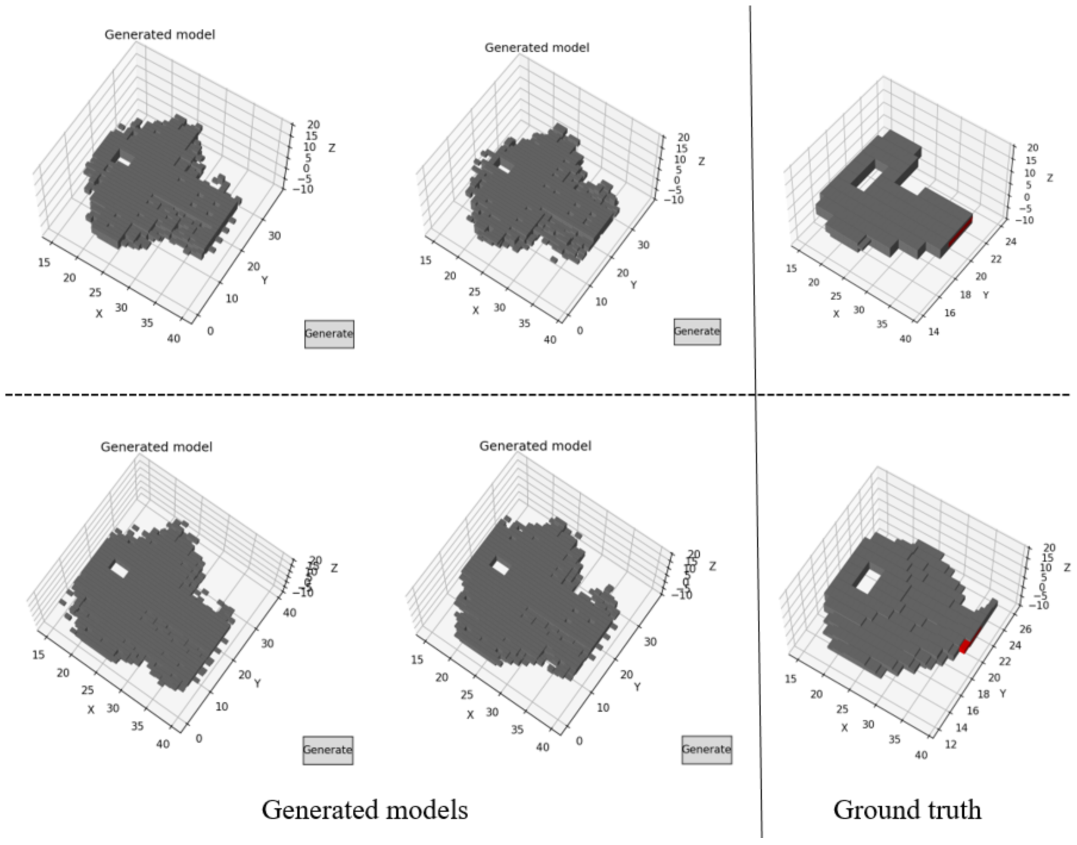


FIGURE 5.2: 3D GAN outputs for disc simple

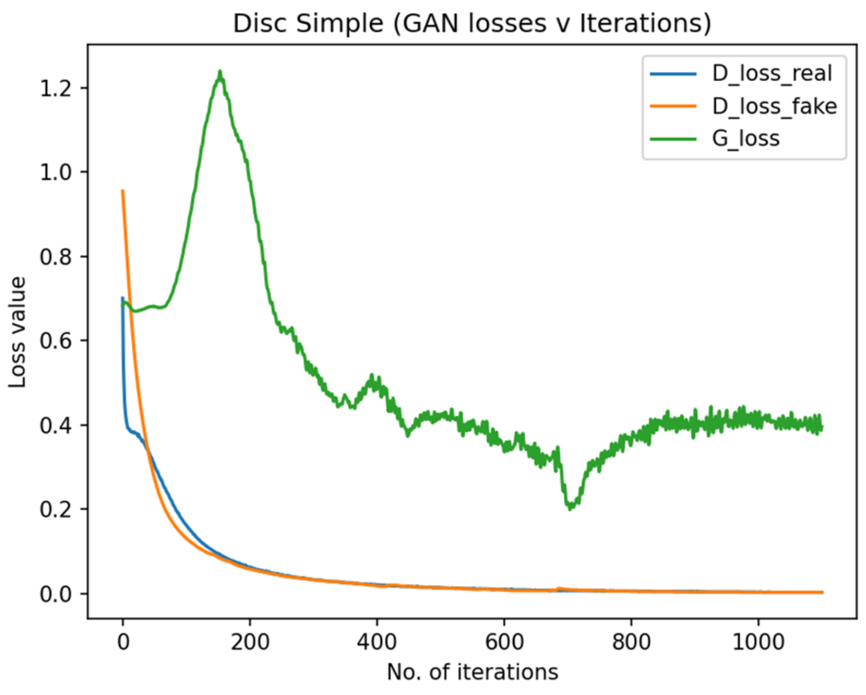


FIGURE 5.3: 3D GAN loss

TABLE 5.1: Comparison of ML approaches for TO

S.No.	Title	Key Points	Advantages	Disadvantages
1	Generative Adversarial Network for Early-Stage Design Flexibility in Topology Optimization for Additive Manufacturing[17]	Uses GANs to generate design candidates that are feasible and flexible.	Reduces design time and cost in additive manufacturing.	Works with a 2D dataset without much variation.
2	Tounn: Topology Optimization Using Neural Networks[19]	Uses a convolutional neural network to learn the optimal design directly from the input boundary conditions.	Produces efficient and practical 2D designs with less computational power.	For a given set of inputs, only one output will be produced. Produces relatively less efficient 3D outputs.
3	TopologyGAN: Topology Optimization Using Generative Adversarial Networks Based on Physical Fields over the Initial Domain[20]	Uses GANs to generate design candidates that satisfy physical field constraints over the initial domain.	Produces efficient and practical designs for complex physical systems.	Works with a 2D dataset. Requires high computation power.
4	SELTO (Sample-Efficient Learned Topology Optimization)[21]	Deep Learning strategy for topology optimization with physics-based pre-processing.	Including physical concepts not only drastically improves the sample efficiency but also the predictions' physical correctness.	For a given set of inputs, only one output will be produced.
5	**Proposed GAN**	**Uses GANs to produce 3D outputs with topology optimizations.**	**Multiple 3D outputs can be produced with less computation power.**	**Produces relatively less optimized outputs (local optima).**

5.2.1 Benefits of 3D GAN

The major benefit of the 3D GAN over other topology optimization methods is that it can generate a huge number of different outputs instead of just the single most optimized output. This can be helpful for the designing of new parts or products.

3D GAN with labeled conditions is a new topic with very few research resources. Our proposed 3D GAN can be helpful for future developments in GAN as we have successfully included conditions in our 3D GAN approach.

5.2.2 Limitations of 3D GAN

Since we are working with voxels, our models have been plotted without smoothness. Hence, the models look rough which will require some post-processing. Another complication is that for every given input condition, there is only a single most optimized output (Global Optima). As GAN takes random noise as input, it produces comparatively less optimized outputs (Local Optima). So the global optima may be reached after running the algorithm multiple times but it may require many iterations.

Chapter 6

Conclusion and Future Scope

To summarize the 2D GAN, we have prepared a small training dataset for GAN and we have successfully implemented the algorithm. Due to insufficient training data, we have not received a notable output. However, we have reduced the losses calculated for the discriminator and the generator.

The 3D GAN has also been successfully implemented for the SELTO dataset [21]. Every new output that is generated has some variation while following the load and boundary conditions. The losses for the discriminator and generator are very low which is a sign that we have trained our algorithm properly and with sufficient data.

The existing GAN frameworks have not been able to use GAN up to its full potential and our work is a small step towards achieving this. Our work can be continued by adding smoothness to the generated models. [26] provides a method for voxel smoothening. With enough training samples and a few modifications in the code can generate new models for any kind of part such as wheel rims, chairs, cars, planes, etc.

In summary, GANs have shown great potential in topology optimization by generating feasible, flexible, and efficient design candidates for various applications. They offer a promising approach to reduce design time and cost while improving the quality and practicality of resulting designs. GAN-based topology optimization techniques could address the limitations of traditional methods, which can be computationally expensive and may not produce practical designs. By enabling the generation of innovative and practical designs with minimal computational resources, GANs have the potential to revolutionize the field of engineering design. However, more research is needed to optimize the performance of GANs and address challenges such as the need for large amounts of training data and potential bias in generated designs. Overall, GANs hold great promise for the future of topology optimization and can drive significant innovation in engineering design.

Bibliography

[1] M. F. F. M. J. S. O. M. M. P. Nima Bakhtiary, Peter Allinger, “A new approach for sizing, shape and topology optimization,” 1996.

[2] “Topology optimization,” https://en.wikipedia.org/wiki/Topology_optimization.

[3] “Comparison between size, shape, and topology optimization,” https://www.researchgate.net/figure/Comparative-illustration-of-size-shape-and-topology-optimization-Structural_fig1_321771366.

[4] “An example of topology optimization,” https://www.fastradius.com/resources/topology-optimization-advantages/.

[5] J. M. W. Inês P. Rosinha, Krist V. Gernaey and U. Krühne, “Topology optimization for biocatalytic microreactor configurations,” 2015.

[6] “Generative adversarial networks,” https://en.wikipedia.org/wiki/Generative_adversarial_network.

[7] V. D. K. A. B. S. Antonia Creswell, Tom White and A. A. Bharath, *Deep learning for visual understanding*, 2018.

[8] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras TensorFlow*. O’Reilly Media.

[9] O. J. P. Gabriel Achour, Woong Je Sung and D. N. Mavris, “Development of a conditional generative adversarial network for airfoil shape optimization,” School of Aerospace Engineering Georgia Institute of Technology, Report, March, 2020.

[10] P. Burdziakowski, “A novel method for the deblurring of photogrammetric images using conditional generative adversarial networks,” *Remote Sensing*, 2020.

[11] "3d shapenets: A deep representation for volumetric shapes," https://3dshapenets.cs.princeton.edu/.

[12] T. X. W. T. F. Jiajun Wu, Chengkai Zhang and J. B. Tenenbaum, "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling," 2016.

[13] "3d-gan, 3d-vae-gan," https://github.com/meetps/tf-3dgan.

[14] "Architecture of the generator," http://3dgan.csail.mit.edu/.

[15] T. F. Jerry Liu, Fisher Yu, "Interactive 3d modeling with a generative adversarial network," 2017.

[16] V. K. V. V. Kniaz, F. Remondino, "Generative adversarial networks for single photo 3d reconstruction," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 2019.

[17] A. G. K. V. S. A. Nathan Hertlein, Philip R. Buskohl, "Generative adversarial network for early-stage design flexibility in topology optimization for additive manufacturing," *Journal of Manufacturing Systems*, 2021.

[18] "Structure of cgan for to," https://www.sciencedirect.com/science/article/pii/S027861252100087X.

[19] K. S. Aaditya Chandrasekhar, "Tounn: Topology optimization using neural networks," 2021.

[20] H. J. L. B. K. Zhenguo Nie, Tong Lin, "Topologygan: Topology optimization using generative adversarial networks based on physical fields over the initial domain," 2020.

[21] H. H. SOREN DITTMER, DAVID ERZMANN and P. MAASS, "Selto: Sample-efficient learned topology optimization," 2022.

[22] "Structure of u-net," https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/.

[23] "Topopt," https://www.topopt.mek.dtu.dk/.

[24] "Tensorflow," https://www.tensorflow.org/.

[25] J. L. B. Diederik P. Kingma, "Adam: A method for stochastic optimization," 2017.

[26] "Voxel smoothening," https://shorturl.at/apwBD.

[27] "Faces generated by gan," https://jonathan-hui.medium.com/gan-a-comprehensive-review-into-the-gangsters-of-gans-part-1-95ff52455672.

[28] "Gans in fashion," https://github.com/sonynka/fashion_gan.