Deep learning frameworks for structural topology optimization

by

Jaydeep Ravindra Rade

A thesis submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

Major: Electrical Engineering (Communications and Signal Processing)

Program of Study Committee:
Adarsh Krishnamurthy, Co-major Professor
Soumik Sarkar, Co-major Professor
Baskar Ganapathysubramanian

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this thesis. The Graduate College will ensure this thesis is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2021

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

ACKNOWLEDGMENTS

I would like to take this opportunity to express my thanks to those who helped me with various aspects of conducting research and the writing of this thesis. First and foremost, Dr. Adarsh Krishnamurthy and Dr. Soumik Sarkar for their constant guidance, support, and patience throughout this research and the writing of this thesis. Their insights and words of encouragement have often inspired me and renewed my hopes for completing my graduate education. I would also like to thank my committee member, Dr. Baskar Ganapathysubramanian, for their efforts and contributions to this work.

I would additionally like to thank all lab members from Integrated Design and Engineering Analysis Laboratory (IDEA Lab) as well as from Self-aware Complex Systems Lab (SCS LAb), they have been always helpful and motivational. I would also like to thank Dr. Aditya Balu for his help throughout this research and making me a better programmer.

I would also like to thank Jay Pathak, my manager at ANSYS, and the whole team for giving me an internship opportunity and helped me for my research.

Finally, I would like to thank my parents and grandparents, without them I wouldn't have achieve this.

This work was partly supported by the National Science Foundation under grant CMMI-1644441 and ANSYS Inc. We would also like to thank NVIDIA® for providing GPUs used for testing the algorithms developed during this research. This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by NSF grant ACI-1548562 and the Bridges system supported by NSF grant ACI-1445606, at the Pittsburgh Supercomputing Center (PSC).

ABSTRACT

Topology optimization has emerged as a popular approach to refine a component's design and increasing its performance. However, current state-of-the-art topology optimization frameworks are compute-intensive, mainly due to multiple finite element analysis iterations required to evaluate the component's performance during the optimization process. Recently, machine learning-based topology optimization methods have been explored by researchers to alleviate this issue. However, previous approaches have mainly been demonstrated on simple two-dimensional applications with low-resolution geometry. Further, current methods are based on a single machine learning model for *end-to-end* prediction, which requires a large dataset for training. These challenges make it non-trivial to extend the current approaches to higher resolutions.

In this thesis, we explore deep learning-based frameworks that are consistent with traditional topology optimization algorithms for three-dimensional topology optimization with a reasonably fine (high) resolution. We achieve this by training multiple networks, each trying to learn a different step of the overall topology optimization methodology, making the framework more consistent with the topology optimization algorithm. We demonstrate the application of our framework on both 2D and 3D geometries. The results show that our approach predicts the final optimized design better than current ML-based topology optimization methods.

CHAPTER 1. INTRODUCTION

Over the past few decades, there has been an increased emphasis on designing components with optimal performance, especially using topology optimization [36, 29]. Topology optimization (a subset of design optimization methods), initially developed by Bendsøe and Kikuchi [4], refers to a set of numerical design optimization methods developed to find appropriate material distribution in a prescribed design domain to obtain geometric shapes with optimal performances. Here, the performance could be any physical phenomenon such as structural strength (or mechanical design), heat transfer, fluid flow, acoustic properties, electromagnetic properties, optical properties, etc. [46]. The domain refers either to a 2D or 3D mesh representation of the CAD geometry, typically used for finite element analysis. Among the different topology optimization methods, some of the most prominent approaches are solid iso-tropic material with penalization (SIMP) [3], level-sets [53], and evolutionary optimization [9, 54]. These approaches are used for several topological design problems where structural, acoustic, or optical performance needs to be optimal [12, 46] while removing the material to satisfy a total material(or volume) constraint.

1.1 Motivation

One of the main challenges in performing topology optimization is the high computational cost associated with it. The performance measure that is being optimized needs to be computed after each iteration of the optimization process. These performance measures are usually obtained from physics simulations (often using numerical solution approaches, such as finite element analysis) that are usually compute-intensive. Due to this computational challenge, performing topology optimization for a fine (high resolution) topological mesh could take a few hours to even days. This computational challenge has inspired several researchers to develop deep learning-based topology optimization to reduce or eliminate the need for numerical simulations.

Although deep learning (DL) has many diverse applications and has demonstrated exceptional results in several real-world scenarios, our focus in this thesis is the recent application of deep learning to learn a system's underlying physics. There has been an increased interest in learning physical phenomena with neural networks in order to reduce the computational requirements and achieve better performance with very little or no data [37, 59, 44, 51, 30, 18, 58, 38, 40, 5]. A popular approach is by modifying the loss function to ensure that a set of physical constraints (boundary conditions) are satisfied. This approach has been especially successful in using deep learning to solve partial differential equations such as Burger's equation, Navier-Stokes equation, and Cahn-Hilliard's equation [44, 47, 58, 30, 18, 59, 38]. These approaches help the framework learn about the physical phenomena and make the learning consistent with the underlying physics. At the same time, algorithmic alignment between the design of the neural networks and the phenomena being learned has been shown to be successful [55]. With this motivation, we propose an algorithmically-consistent deep learning framework for structural topology optimization.

1.2 Overview

An algorithmically-consistent deep learning framework for the structural topology optimization (STO) need to (i) learn the underlying physics for computing the compliance, (ii) learn the topological changes that occur during the optimization process, and (iii) produce results that respect the different geometric constraints and boundary conditions imposed on the domain. To simplify the problem, we first discuss three essential elements that form the backbone of any data-driven approach: (i) the data representation, (ii) training algorithms, and (iii) model representation. For algorithmically-consistent learning, each of these three elements must be consistent with the classical structural topology optimization algorithm. In this particular instance, we focus on topology optimization using the solid isotropic material condition with penalization (called SIMP [3]) algorithm for our framework.

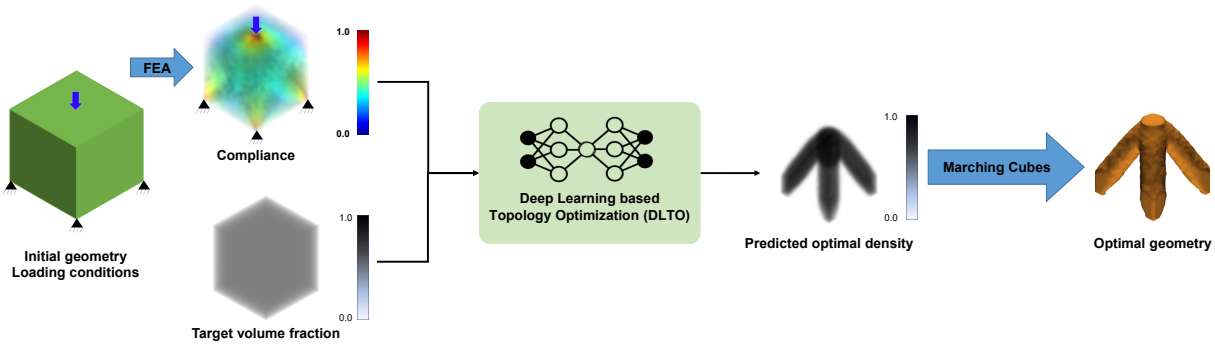

Figure 1.1 We propose a deep learning based topology optimization framework. The input to this framework is the compliance of the initial geometry along with the target volume fraction. Using the DLTO framework, we predict the optimal density of the geometry without any requirement of iterative finite element evaluations. We then convert the predicted optimal density of the geometry and convert it into triangular surface mesh representation using the marching cubes algorithm to give the final optimal design geometry.

The first element is the identification of an algorithmically-consistent representation for structural topology optimization. Structural topology optimization is an iterative process where the design is modified through several iterations until the objective function (total compliance) converges to an optimal value. Further, each element's compliance is used in the sensitivity analysis for updating the element densities at each iteration. Thus, the element compliance is a valid and consistent representation of the geometry compared to other representations (such as voxel densities, strains, etc.) used in current deep learning approaches. Therefore, in the proposed framework, we use the element compliance as the CAD model representation of the geometry, loading, and boundary conditions (as shown in Figure 1.1). Note that, unlike the use of strain tensor and displacement tensor as proposed by Zhang et al. [60], this representation is compact, leading to better scaling at higher resolutions.

The next elements in an algorithmically-consistent approach are the training and inference pipelines, which need to be consistent with the classical structural topology optimization pipeline. In our experiments, we observe a non-trivial transformation of the densities from the first iteration

to the final converged one. Due to this non-trivial transformation, learning the mapping between the initial topology and the final optimized topology is not a trivial *one-step* learning task. Therefore, we use the intermediate densities obtained during the process of data generation to enhance the performance of our proposed framework along with the initial compliance and target volume fraction as input and the final optimal density as the target.

Finally, an algorithmically-consistent framework should simultaneously satisfy two constraints for structural topology optimization: the topological constraint of matching the target volume (often prescribed as a volume fraction or percentage of volume removed) and the physical constraint of minimizing the compliance. While computing the volume fraction is trivial, computing the compliance involves performing a finite element analysis. To avoid this computation, we propose developing a surrogate model for learning the mapping of a given intermediate density to its corresponding intermediate compliance.

In summary, we have developed two algorithmically consistent frameworks for structural topology optimization, namely, the Density Sequence (DS) prediction and the Coupled Density and Compliance Sequence (CDCS) prediction. The first approach uses a sequential prediction model approach to transform the densities without any additional compliance information. In the second approach, we add intermediate compliance to train a compliance-predicting surrogate model for improving the results. We compare the proposed approaches with the baseline method of Direct Optimal Density (DOD) prediction. Direct optimal density prediction is an end-to-end learning approach where the final optimal density is directly predicted using just the initial compliance and target volume fraction. The DS framework involves two convolutional neural networks for obtaining the final prediction. In CDCS, we use three convolutional neural networks iteratively during inference to predict the final optimal density.

1.3 Thesis Organization

This thesis is arranged as follows. We start with discussing formulation and related work in Chapter 3, where we talk about the mathematical representation and the flowchart for the SIMP algorithm. We will also talk about the relative work published in this domain of research area.

Chapter 2 deals with the primary explanation of deep learning architectures, specifically on convolutional neural networks (CNN) and long short-term memory cells (LSTMs). We will explain each layer involved in CNN and LSTM architectures.

Next, we introduce the deep learning methods proposed in this thesis in Chapter 4. We discuss the different frameworks used for each method in detail. Additionally, we will talk about the data representation and training algorithm for each method.

We cover the details of the data generation process in Chapter 5, which is used as training data for our proposed approaches. Both 2D and 3D data generation pipelines will be explained in this chapter.

In Chapter 6, we show the statistical results from our experiments and demonstrate the performance of our proposed methods on both 2D and 3D structural topology optimization. We also visualize our results and show them in this chapter.

Finally, we conclude this work with some future directions of research in Chapter 7.

CHAPTER 2. BACKGROUND ON DEEP LEARNING

2.1 Deep Learning

Deep learning (DL) is the subset of machine learning (ML), which is a subset of artificial intelligence (AI). AI is a technique that enables machines to mimic human behavior, where ML is the technique to use statistical methods to enable machines to improve with experience. Deep learning which is, on the other hand, is inspired by the structure of the human brain. Deep learning methods aim to draw similar conclusions as humans would by continually analyzing data with a given logical structure. For this, deep learning uses the multi-layered inter-connected structure of neurons, also known as deep neural networks.

Basically, given the pair of inputs X and outputs Y, deep NN tries to approximate the function $F()$, which is parameterized by weights W. The parameter weights (W) here is the numerical value which is the weight of the linkage between two neurons. This approximation is possible using numerous layers of neurons that transform the input in latent space. Then NN extract features from this latent space and maps them to the output. Therefore, in multi-layer NN, at any layer l in the NN, the the output y_l is computed from x_l as: $y_l = \sigma(w_l.x_l + b_l)$, where w_l and b_l are the weights and biases used as a weighted linkage from layer $l-1$ to l, respectively. The function $\sigma(.)$ is the non-linear transformation known as activation function, which is only non-linear mapping in the NN. So combining such layers to form multilayered NN, we get W as the set of all weights from every layer, and the prediction at the output layer, $\widehat{Y}$, can be realized with $\widehat{Y} = F(W, X)$.

The goal of the deep NN is to find the best parameters W^* such that the prediction $\widehat{Y}$ is as much as close to the output Y. The process of finding appropriate weights is called training. At the start of the training, the weights W are randomly initialized. During the training process, the W is modified W^* such that the loss function is minimized computed between $\widehat{Y}$ and W during training. This optimization of parameters is performed using gradient-descent-based back-

propagation method [25]. There are several gradient descent methods like Adam optimizer [20], RMSProp and Adagrad [32], Adadelta [57] and many others. Also, various loss functions can be used based on the application; some examples for loss function are mean square error (MSE), binary cross-entropy (BCE), KL-divergence loss.

For better training performance, we can perform batch-normalization [17], regularization [23], and dropout [49]. In batch normalization, output from the previous layer is normalized for each batch to enforce the mean close to 0 and standard deviation close to 1. In regularization, an additional term is added in the loss function termed as the penalty function to avoid overfitting. On the other hand, in dropout, multiple connections between two layers are dropped randomly during the training process; the effect is that the network becomes less sensitive to the specific weights of neurons and generalizes better.

In an upcoming couple of sections, we will be discussing some popular classes of deep NN, the convolutional neural networks (CNNs) and long short-term memory cell(LSTM), variant of recurrent neural networks (RNNs) architecture.

2.2 Convolutional Neural Networks (CNNs)

As discussed earlier, NN tries to fit function $F()$, which maps input data to output and is done through stacking multiple layers of neurons. In typical NN, each layer neuron is connected with its previous and next layer neuron, known as fully-connected layers. This fully-connected deep NN, when used for the input of large size, the number of parameters get really high makes it computationally expensive, and over-fitting occurs. We understand from it that every connection of neuron to neuron may not be useful. Instead of fully-connection, CNNs connect the neuron to only a local region of the input volume. This layer is termed as convolutional layer in [25] as the output of this layer is obtained by convolving the input with weights (filters). These weights are shared for all such local regions in the input, reducing the number of parameters dramatically. The convolutional layer is depicted in Figure 2.1. The output at of any convolutional layer l with convolution connection with 3D input (width, height, channels) is given by,

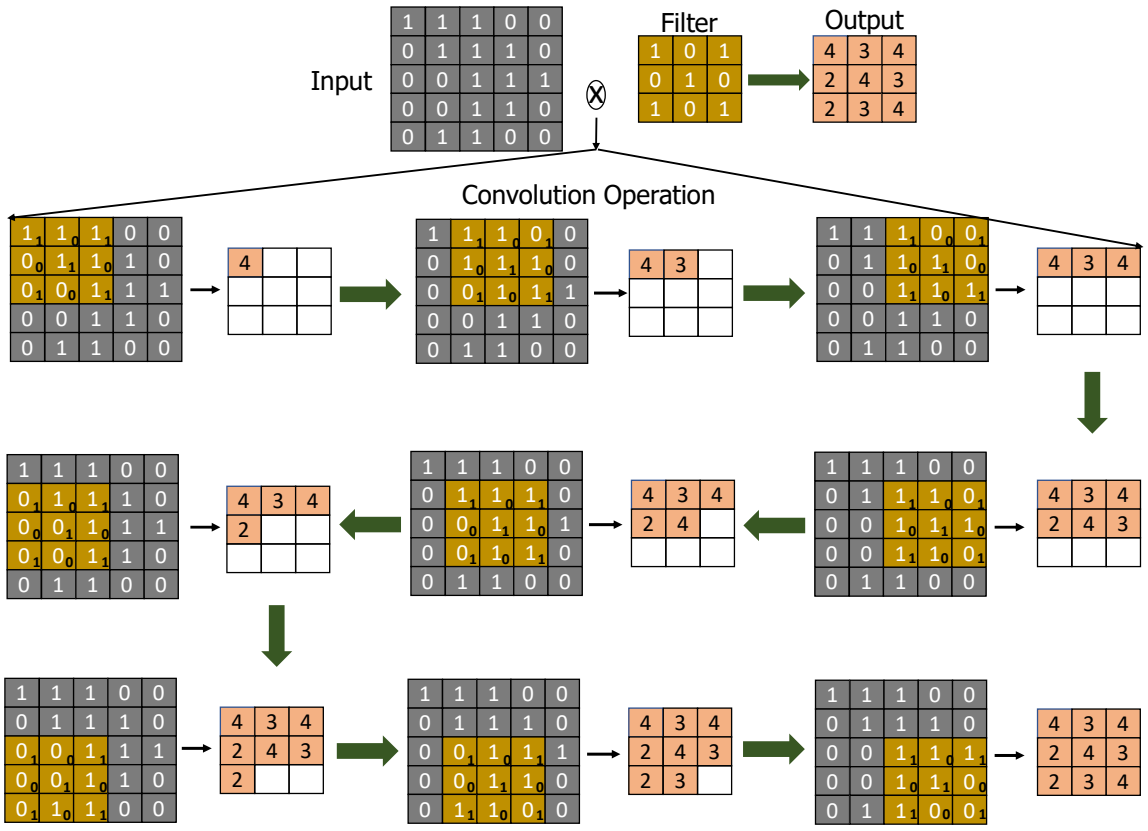

Figure 2.1 Illustration of convolution operation performed on an image with size 5×5 with filer of size 3×3.

$$
\begin{aligned}
Z_l[i,j,m] &= W_l \otimes X_l \\
W_l \otimes X_l &= \sum_{k_1} \sum_{k_2} \sum_{n} W_l[k_1,k_2,n,m] X_l[i+k_1, j+k_2, n] + b_l[m] \\
Y_l &= \sigma(Z_l)
\end{aligned}
\tag{2.1}
$$

Where X_l is the input to layer l, and Y_l gives the output of the layer by convolving the input with the weights and biases and passing it through non-linear transformation $\sigma(.)$. The weights here are also known as filter or kernel. The shape of the kernel is (k_1, k_2, n, n), where k_1, k_2 is the spatial resolution, and n denotes the depth of the kernel same as the number of channels in

input, and m represents the number of kernels same as the number of channels in output. Y_l, the intermediate output of the layer, is called a feature map extracted from the input.

In CNNs, the other operation performed is the pooling operation. As the neighboring pixels share similar features, these features can be discarded by using better representation in lower dimensions by performing the pooling operation. One of the popular pooling operations is max-pooling [22], where we select the feature which has maximum numerical value within the local region. This layer is known as the pooling layer or max-pool layer in case the max-pooling function is used. As it is just a downsampling layer, it does not have any trainable parameters.

Another layer we use in CNNs is the upsampling layer, where we increase the dimension of the data to get the output of the desired shape. For example, taking an image as an input and predicting the image of some shape at the output. This upsampling is necessary as we have used a pooling layer to decrease the dimensionality of the data. By arranging and stacking these convolutional layers, pooling layers, upsampling layers, and sometimes fully-connected layers, we get the convolutional neural network.

2.3 Long Short Term Memory Cells (LSTMs)

We talked in an earlier section about the use of CNNs for spatial data representation. However, when the data contains the temporal representation or has temporal correlation, recurrent neural networks (RNN) [31] is preferred. Simple RNN suffers from short-term memory; that is, if the input data sequence is long enough, RNN will struggle to carry the information from earlier steps to later ones. LSTMs are the type of RNN that can learn long-term dependencies in the data. LSTMs are introduced by Hochreiter and Schmidhuber [15] in 1997 and refined by many researchers since then. LSTMs have an internal mechanism called gates that can control the flow of information.

The main attributes of the LSTM are cell state and multiple gates. The cell state acts as the memory of the network, which carries the relevant information throughout the processing of the input sequence. So the information from the earlier step can also propagate in later steps avoiding short-term memory problems. The information in the cell state is modified according to the gates.

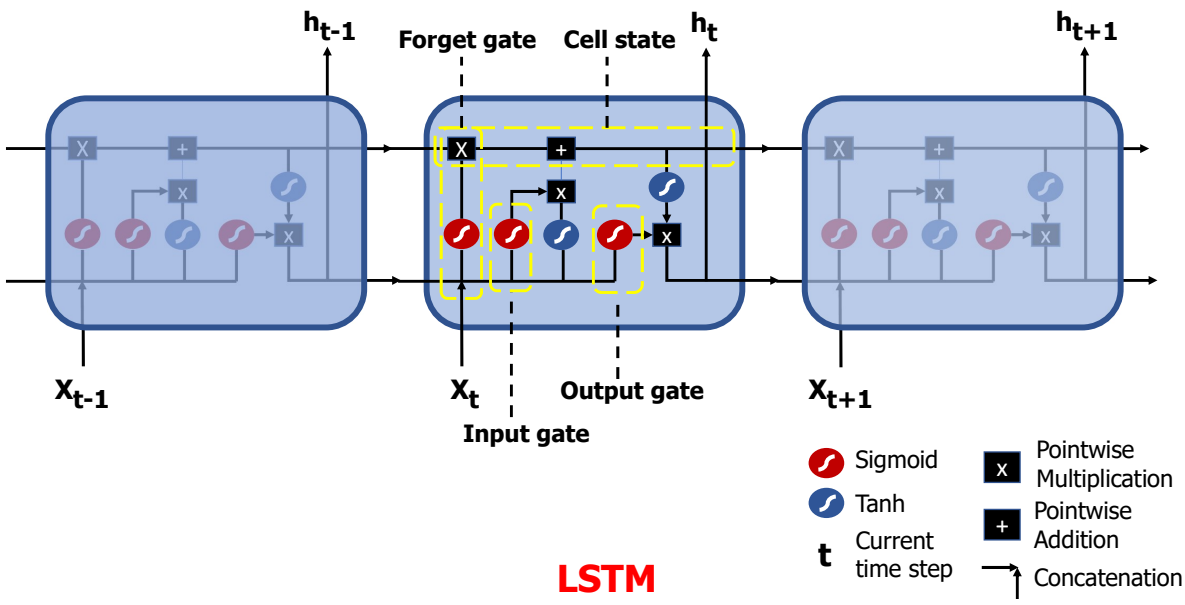

Figure 2.2 Example of LSTM network with X_t as the input sequence at current step t and h_t is the corresponding hidden state.

The gates decide which information needs to be added or removed from the cell state. During training, gates can learn what information should cell state keep or forget. The gates contain a sigmoid activation function which transforms the input into 0 to 1 range; this helps to update or forget the data. The architecture of LSTM is showed in Figure 2.2.

We will discuss each element of Figure 2.2 in details. One of the gates in LSTM is forget gate. This gate, as the name suggests, decides what information to be or remember. Information from previous hidden state and current input information is passed through the sigmoid layer. Sigmoid outputs the value between 0 and 1; the value closer to 0 means to forget, and the value closer to 1 means to keep the information.

Now, as the new input is processed, the cell state needs to be updated. For this, LSTM uses an input gate. Again, the current input and the previously hidden state information are passed through the sigmoid to obtain a value between 0 and 1. Additionally, both quantities are also passed through the tanh function to transform the value into a -1 to 1 range to help regulate the

network. Then multiplying the output of the tanh with the output of the sigmoid will decide which information relevant to keep from tanh output.

With the help of the forget get and input gate, the cell state can be updated. First, the cell state from the previous hidden step is multiplied by the output of forget get. This will discard the less important information. Then the point-wise addition with the output of the input gate is performed to update the current cell state.

Finally, LSTM uses an output gate that decides what the next hidden state should be. Hidden states contain the information from previous inputs, and they can also be used to make predictions. The first step is to pass the previous hidden state and current input through the sigmoid function. Next, pass the updated current cell state through the tanh function and perform point-wise multiplication. The multiplication with sigmoid output decide which information should hidden state should carry.

Summarizing these multiple gates, the forget gate decides what information should be kept from the previous hidden state. The input gate determines what relevant information needs to be added from the current input step, and the output gate decides what the next hidden step should be. It can be used for predictions as well.

CHAPTER 3. FORMULATION AND RELATED WORKS

3.1 Mathematical Representation

Formally, topology optimization is represented as:

$$\begin{aligned} &\text{minimize: } C(U) \\ &\text{subject to: } \mathbf{KU} = \mathbf{F} \\ &g_i(\mathbf{U}) \leq 0. \end{aligned} \tag{3.1}$$

Here, $C(U)$ refers to the objective function of topology optimization. In the case of structural topology optimization, this is the compliance of the system,

$$C = \int_{\Omega \in \mathcal{S}} bu \; d\Omega + \int_{\tau \in d\mathcal{S}} tu \; d\tau \tag{3.2}$$

where b represents the body forces, u displacements, t surface traction, and Ω and τ are volume and surface representations of solid. The constraint $g_i(U)$ includes a volume fraction constraint, $g_i = (v/v_0) - v_f$. Since, this optimization is performed for every element in the mesh, the combinatorial optimization is computationally intractable. Naturally, an alternative solution is to represent the same set of equations above as a function of density ρ for every element.

$$\begin{aligned} &\text{Minimize: } C(\rho, U) \\ &\text{subject to: } \mathbf{K}(\rho)\mathbf{U} = \mathbf{F} \\ &g_i(\rho, \mathbf{U}) \leq 0 \\ &0 < \rho \leq 1 \end{aligned} \tag{3.3}$$

3.1.1 SIMP Algorithm and Flowchart

This design problem is relaxed using SIMP, where the stiffness for each element may be described as, $E = E_{min} + \rho^p(E_{max} - E_{min})$. Here, p is the parameter used for penalizing the element

Algorithm 3.1: SIMP topology optimization [3]

Input : $\mathcal{S}$, L, BC, V_0
Output: D_{fin}(set of all densities for each element, ρ)
Load design; apply loads and boundary conditions
Initialize: $D_0 \rightarrow V_0 / \int_\Omega d\Omega$
Initialize : $ch = \text{inf}$
while $ch < threshold$ **do**
  Assemble global stiffness matrix $\mathbf{K}$ for element stiffness matrix $k_e(\rho_e)$
  Solve for $\mathbf{U}$, using $\mathbf{K}$, loads (L) and boundary conditions (BC)
  Compute objective function, $\mathbf{C} = \mathbf{U}^T\mathbf{K}\mathbf{U} = \sum_{e=1}^{N} \rho^p u^T k_e u$
  Perform sensitivity analysis, $\frac{\partial c}{\partial e} = -p\rho^{(p-1)} u^T k_e u$
  Update the densities (D_i) using a optimality criterion
  $ch = ||D_i - D_{i-1}||$
end

density to be closer to 1.0. A typical SIMP-based topology optimization pipeline is shown in Algorithm 3.1. We can understand more about the SIMP algorithm with the help of flowchart in Figure 3.1. We also visualize the evolution of topology shape on right hand side through each step of SIMP algorithm. Starting with initializing design domain and defining the loading and boundary condition, then computing the compliance using finite element analysis (FEA). Then using each element's compliance in the sensitivity analysis for updating the element densities at each iteration. This process is continued until the change in the shape of the design for consecutive iterations is within some small margin.

While this is a naive implementation, more sophisticated methods for structural topology optimization such as level-set methods [53] and evolutionary optimization methods [9, 54] are also popularly employed. Despite several advancements in structural topology optimization, a common challenge in all these approaches is that it requires several iterations of the finite element queries to converge on the final density distribution. Different optimization methods result in different, yet comparable, optimal solutions alluding to the fact that multiple optimal solutions exist for the same topology optimization problem. Deep learning-based methods are a natural fit for accelerating this task, which has been explored in several works about which we will talk about it in next section.

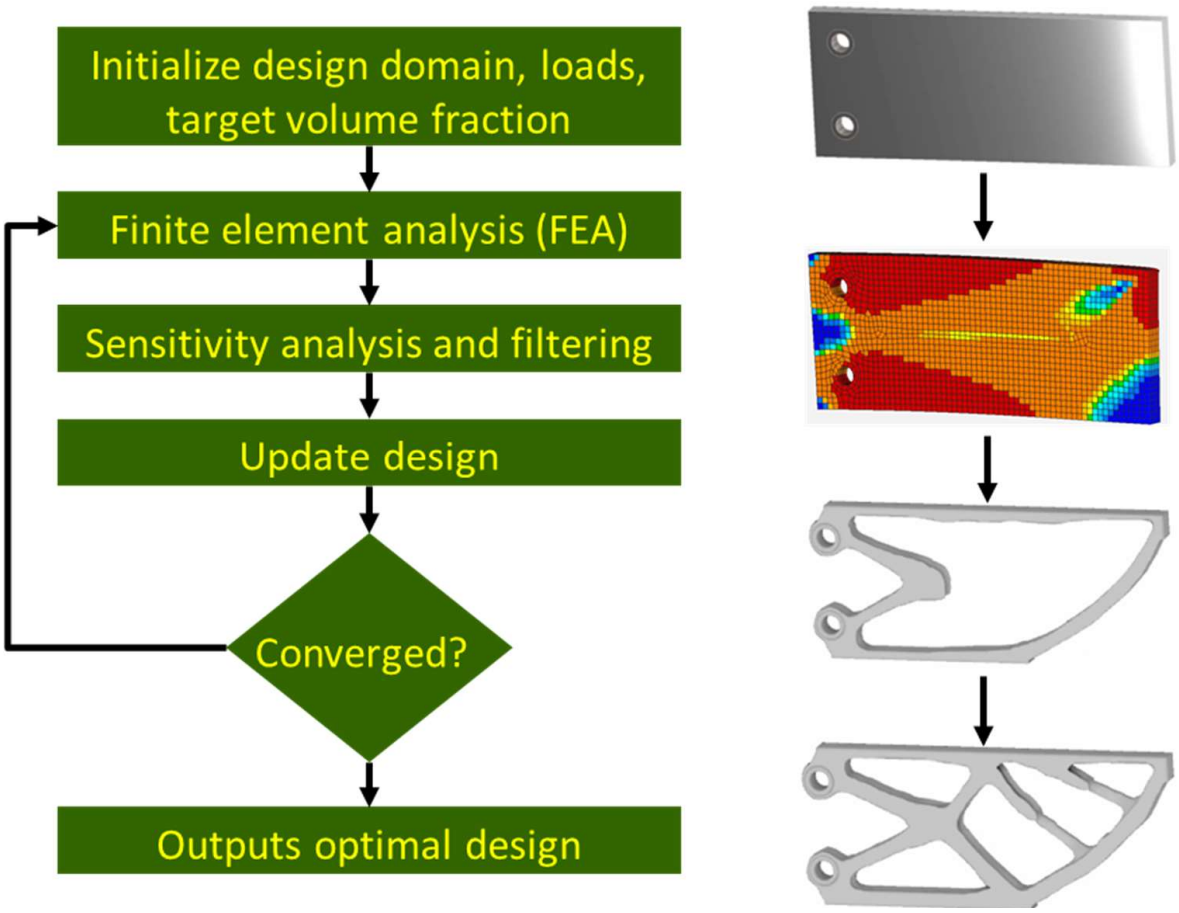

Figure 3.1 A flowchart for SIMP method with showing evolution of design shape on right hand side.

3.2 Related Works

3.2.1 Deep Learning for Topology Optimization

Several deep learning-based topology optimization frameworks have been proposed [48, 2, 56, 60, 33, 7, 28, 21, 41, 14, 11, 24, 34, 27, 43, 56, 35, 61, 13, 1, 26, 10, 6, 45, 50, 39, 19]. Among these several works, the most relevant and significant ones are discussed below. Initially, Banga et al. [2] and Sosnovik and Oseledets [48] proposed to perform the fine refinement of the design using deep convolutional autoencoders since the fine refinement stage usually requires several finite element iterations during the optimization process. Sosnovik and Oseledets [48] used the densities

obtained after five iterations of the SIMP-based structural topology optimization as input to a deep learning network that directly predicts the final density. Banga et al. [2] extend this idea to 3D design geometries, along with an additional input of the boundary conditions, but for a very coarse geometric resolution ($12 \times 12 \times 24$). Yu et al. [56] developed a framework that takes the input design, boundary conditions, and the prescribed volume fraction and predicts the final target shape. They also create a generative framework where they generate several optimal designs. However, their research was restricted to only one type of boundary condition. A more generic framework to accommodate all possible boundary conditions using this method would require an impractically large dataset. Therefore, Zhang et al. [60] developed an improved representation of the geometry, loading conditions, and boundary conditions using the strain tensor and displacement tensor as input. They demonstrate this framework using 2D geometries and represent each component of the strain tensor and displacement tensor as a different channel of the 2D image input. Using convolutional neural networks, they predict the final density. While their results are an improvement over earlier methods, this representation is not scalable to 3D. The strain tensor has three more components in addition to the increase in overall data size due to representing the geometry using 3D voxels, leading to several computational challenges. Recently, Chandrasekhar and Suresh [7] propose a topology optimization algorithm using neural networks where the neural network is used for identifying the density for each element at each iteration of the optimization process. This approach produces faster convergence and comparable results with SIMP-based structural topology optimization. However, this approach's main drawback is that finite element evaluations are still needed (although lesser than SIMP-based structural topology optimization). To the authors' best knowledge, very few researchers consider the idea of using compliance and the intermediate densities and compliances for improving the learning of structural topology optimization. Further, most of the efforts in the area have been in the 2D representation of geometries and very low-resolution 3D representation of geometries. Therefore, a scalable 3D framework for algorithmically-consistent deep learning framework for structural topology optimization is needed.

CHAPTER 4. ALGORITHMICALLY-CONSISTENT DEEP LEARNING

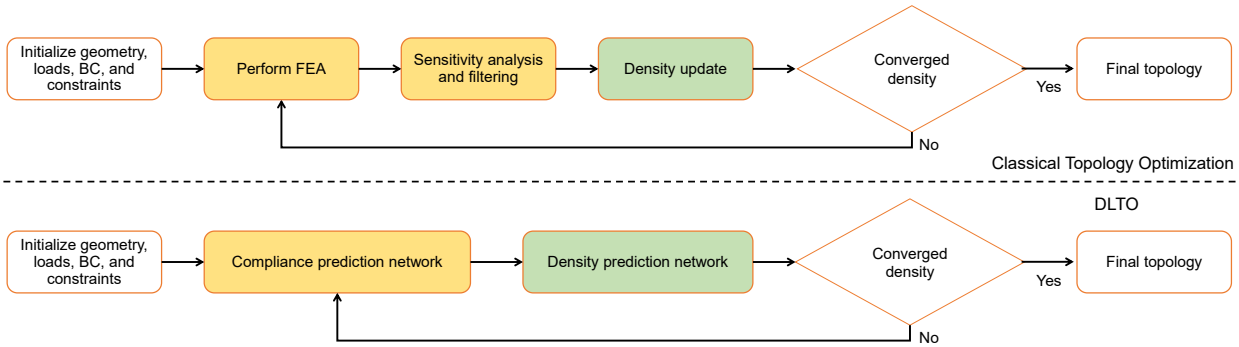

Figure 4.1 Topology optimization pipeline: The traditional topology optimization performs several iterations of finite element analysis, followed by sensitivity analysis and filtering. Using the filtered densities and compliance, we perform a density update. These iterations are performed several times till the density has converged. The DLTO approach replaces the repetitive performance of finite element analysis using a compliance prediction network and the density update with density prediction network.

In this section, we first explain the baseline deep learning approach, which we use to compare our results. We also compare the performance of our proposed frameworks and the baseline against the classical SIMP-based structural topology optimization method. After explaining the baseline, we explain the two proposed frameworks, the density sequence (DS) prediction and the coupled density and compliance sequence (CDCS) prediction. Figure 4.1 shows how our proposed frameworks are algorithmically consistent with the SIMP topology optimization.

4.1 Baseline Direct Optimal Density Prediction

Recently, U-Nets [42, 8] have been known to be effective for applications such as semantic segmentation and image reconstruction. Due to its success in several applications, we chose a U-Net for this task. The input to U-Net is a tuple of two tensors. The first is the initial compliance

(represented in the voxel or pixel space); the second is a constant tensor of the same shape as the compliance tensor. Each element of the constant tensor is initialized to the target volume fraction, which is a number between $[0, 1]$. First, a block of convolution, batch normalization, is applied. Then, the output is saved for later use for the skip-connection. This intermediate output is then downsampled to a lower resolution for a subsequent block of convolution, batch normalization layers, which is performed twice. The upsampling starts where the saved outputs of similar dimensions are concatenated with upsampling output for creating the skip-connections followed by a convolution layer. This process is repeated until the final image shape is reached. At this point, the network utilizes a final convolution layer before producing the final density. The network architecture is shown in Figure 4.2.

We preprocess the compliance to transform it to the $[0, 1]$ range. We first take the log_{10} of the compliance and then normalize it by subtracting the minimum value and then dividing by the difference of maximum and minimum values to scale the log values to $[0, 1]$ range, so all the inputs are in the same range.

To train the neural network model such that it is robust to the loads applied on the input geometry, we augment the inputs by rotating the input tensor by 90° clockwise and counter-clockwise around all three axes and by mirroring the tensor along the X-Y plane, X-Z plane and, Y-Z plane. We threshold the final target density to get a binary density with a value of 0 and 1. The density value 1 corresponds to the element where the material is present, while density value 0 corresponds to the element where the material is absent or removed. We do not use any intermediate compliance or intermediate densities to train this network as it is end-to-end learning; we only need initial compliance and final optimal density.

We use the Adam [20] optimizer during the training phase. We use an adaptive learning rate, which helps the optimization process. To guide the optimizer, we use the binary cross-entropy function to calculate loss between the predicted and the target density.

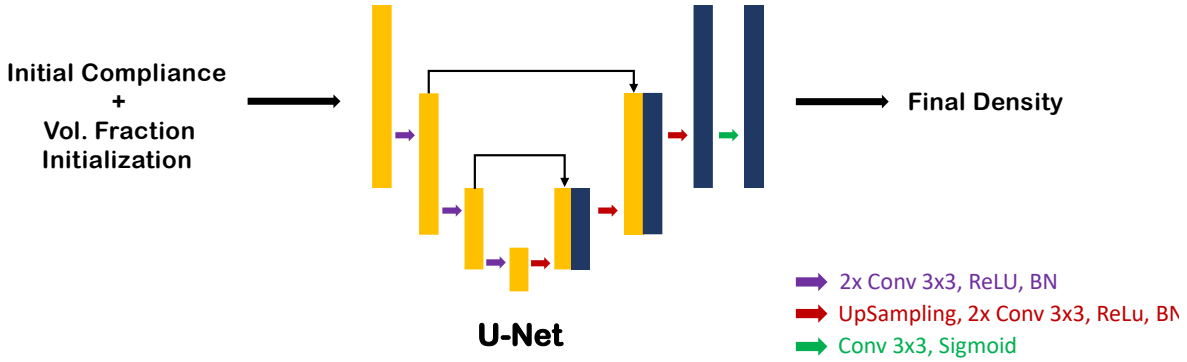

Figure 4.2 Direct optimal density (DOD) prediction: This baseline model is used for comparing our proposed frameworks. The input in this approach is the initial compliance for the geometry along with the target volume fraction initialized. Then we use a U-Net architecture for predicting the optimal density.

4.2 Density Sequence Prediction

For the data representation to be algorithmically-consistent we learn the structural topology optimization from compliance of the initial geometry. However, the compliance keeps evolving during the iterations since the densities also change during optimization. Therefore, the mapping between the original compliance and the final density is not trivial and may not directly correlate with the final density. To improve the performance, we develop the framework in two phases, as shown in Figure 4.3. The first phase is called an initial density prediction network (IDPN), which predicts the topology's initial density distribution based on the initial compliance per element obtained for the original geometry. With initial density, we use the iterative density transformation information available from the topology optimization process to transform the initially proposed density to the final optimized density. We perform this transformation using another network (density transformation network, DTN). The DTN does not use any information about the compliances. Therefore, using IDPN and DTN, we can predict the final densities for a given initial design and its corresponding original compliances. This process is shown in Figure 4.3.

The two phases of the Density Sequence Prediction method require two different network architectures, with each performing algorithmically-consistent transformations of the given input

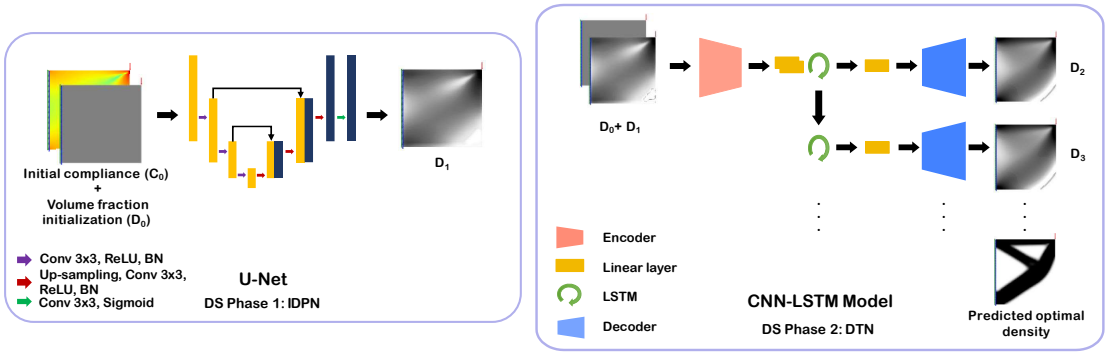

Figure 4.3 Density sequence (DS) prediction: In this framework, we perform the task in two phases. In the first phase, we take the initial compliance and volume fraction initialization to predict an initial density map. Using the initial density and the volume fraction initialization, we predict a series of densities similar to the prediction from a SIMP topology optimizer to finally predict the optimal density.The details of the training process is covered in the text.

information to obtain the final optimized shape. The first architecture corresponds to the first phase, where the task is to predict an initial density. The second architecture corresponds to the second phase, where the density obtained from phase 1 is transformed to a final density.

Phase 1: Initial Density Prediction:

As a first phase of the method, IDPN uses the initial elemental compliances and initialized volume fraction as input and predicts an initial density. We use U-Net [42, 8] network architecture for this phase. The architecture is similar to the architecture described in Section 4.1 and is shown in the left part of the Figure 4.3.

For 2D phase 1 (IDPN), the initial compliance and the volume fraction constraint are represented as a two-channel *"image"*, and the target is a one-channel *"image"* of the element densities obtained after the first iteration of structural topology optimization. For 3D structural topology optimization, the input is a four-dimensional tensor with two 3D inputs concatenated along the fourth axis, and the target is a 3D element density. An additional data processing step on the compliance is necessary for the efficient performance of IDPN. The operation performed is described in Section 4.1.

Phase 2: Density transformation:

The training of phase 2 is more involved than phase 1. We train a convolutional neural network with long short term memory (CNN-LSTM). Long short term memory cells (LSTMs) are helpful in learning from data with temporal history. In phase 2, there is a sequence of density transformations from the beginning to the end. Given these transformations are non-linear, a short-term history is not sufficient for robust prediction of the transformation. Capturing both long-term and short-term temporal dependencies is one of the salient features of LSTMs. Therefore, we use LSTMs along with CNNs (traditionally used for spatial data such as images) to transform the densities. The architecture of the CNN-LSTM used for DTN is shown in the right part of the Figure 4.3.

The CNN-LSTM architecture starts with a set of convolution, max pooling, and batch normalization layers (called the encoder), which is used to transform the image to a latent space flattened embedding used by the LSTM. A sequence of LSTM layers is used to obtain a transformed latent layer. A set of deconvolution and upsampling layers (called Decoder) is used to obtain an image (representing the element densities after one iteration of structural topology optimization). The LSTM is unrolled for predicting a sequence in order to provide back-propagation through time. So, the intermediate densities of the structural topology optimization process are loaded as a sequence and processed to obtain the transformed density during the training process.

For phase 2 (DTN), the intermediate densities (each represented as a one-channel image) are used for performing the training. However, all the iterations of topology optimization are not significant in the learning process. Therefore, we curate the intermediate densities to only have unique densities (defined by a metric of L_2 norm). This uniquely curated set of densities are used for performing the training of DTN. Since DTN only deals with densities, no processing is required. To make the neural network more robust, we implement on the fly data augmentation, as discussed in Section 4.1.

Training Algorithms

For training IDPN, we use two different loss functions: (i) the mean-squared error between the predicted and target densities and (ii) the mean-squared error between the mean of the predicted and target densities. The second loss function ensures that the volume fraction of the target and predicted densities are the same. While training DTN, an additional loss function is added. Since the final geometry cannot have densities between $(0.0, 1.0)$, the densities should belong to the set $\{0, 1\}$ because of solid isotropic material. To impose this condition, we use the binary cross-entropy loss function and the two loss functions used for IDPN. In addition to loss functions, stochastic gradient descent based optimizers such as Adam [20] were used for performing the optimization.

Once the training is performed, the *learned* parameters for both the networks are joined such that an *end-to-end* inference scheme may be implemented. This inference scheme only requires the initial compliance and the volume fraction constraint (input to IDPN). The output of IDPN is used as input to DTN to get the final density without any additional information required. This *end-to-end* scheme makes it applicable to any generic design.

4.3 Coupled Density and Compliance Sequence Prediction

Inspired by the iterative SIMP method, we use deep neural networks to develop a coupled density and compliance sequence prediction framework. In our dataset, we observed that the first five density iterations from the SIMP-based topology optimization method underwent more significant transformations compared to later iterations (also referred to as coarse and fine refinement by Sosnovik and Oseledets [48]). We design three network architectures that use the intermediate compliances and intermediate densities to predict the final optimal density. The first two networks, namely, compliance prediction network (CPN) and density prediction network (DPN), feed their output as an input to each other as coupled interaction, and the third network, the final density prediction network (FDPN), uses the last output of density prediction network to produce the final optimal density (similar to the approach taken by Sosnovik and Oseledets [48]).

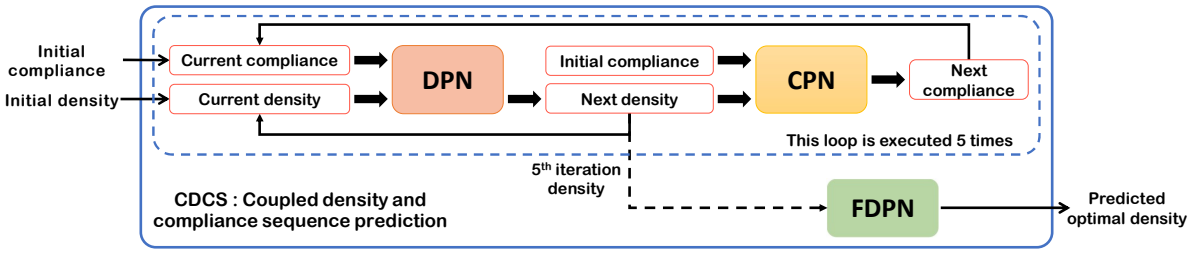

Figure 4.4 Coupled density and compliance sequence (CDCS) prediction: In this framework, the initial compliance (see text for more details) and volume fraction initialization is transformed by an iterative coupled prediction from a density prediction network (DPN) and compliance prediction network (CPN). Five iterations of this process is performed to finally get the density and predicting the optimal density using a final density prediction network (FDPN). The details of the training process is covered in the text.

As the name suggests, the compliance prediction network predicts the elemental compliance for a given iteration's density. It uses initial elemental compliance and the current iteration density obtained from the DPN. For CPN, we use Encoder-Decoder architecture. In the encoder, we use blocks of two convolutional layers followed by batch-normalization. Similarly, we use an upsampling layer, two convolutional layers, and batch-normalization blocks for the decoder. The encoder encodes the input to the lower resolution latent space, and the decoder then decodes the encoded input to the next elemental compliance.

We use the current iteration elemental compliance and the current iteration density to predict the next iteration density for the density prediction network. We use U-SE-ResNet [33] architecture for the DPN. Adding SE-ResNet [33] blocks in the bottleneck region of U-Net architecture, in addition to the skip connections of U-Net from the encoder to the decoder, builds the U-SE-ResNet. The SE-ResNet block consists of two convolutional layers followed by SE(Squeeze-and-Excitation) block [16] with residual skip-connection from the input of the block. The encoder and decoder of U-SE-ResNet are the same as used in CPN architecture. Refer to Section 4.3 for more details on the architectures of U-SE-ResNet.

The final model in this method is FDPN. As mentioned earlier, the elemental density has undergone a significant transformation during the first five iterations. So, taking advantage of the neural network, we avoid the iterative process to obtain the final density. We only use the fifth iteration density to predict the final optimal density directly. For FDPN we implement U-Net [42, 8] architecture. The encoder and decoder part of the U-Net used here is the same as discussed in CPN architecture.

Compliance is preprocessed before feeding it to the neural networks. We normalize the compliance values to be in the $[0, 1]$ range. The method for normalizing the compliance is explained in detail in Section 4.1. In addition to this, we perform data augmentation discussed in Section 4.1 for all three networks. More details on architectures mentioned in this section can be found in Section 4.3.

Training Algorithms

All three networks are trained independently. During the training phase, we use Adam [20] optimizer for all three networks. For more efficient training, we use an adaptive learning rate. The mean absolute error loss function is used for CPN. Moreover, for DPN and FDPN, the binary cross-entropy loss function is used since they are predicting the densities.

During inference, the first two networks are used in a loop (see Figure 4.4). We start with the initial compliance and initial density, which is initialized with a volume fraction value as a tensor with the same shape as the initial compliance tensor. Using the density prediction network, we predict the subsequent iteration's density and feed it as input to the compliance prediction network, producing the compliance corresponding to the new predicted density. This loop is executed five times, so we get the fifth iteration's density prediction at the end of the loop. We use this predicted fifth iteration density as input to the final density prediction network and directly predict the final optimal density.

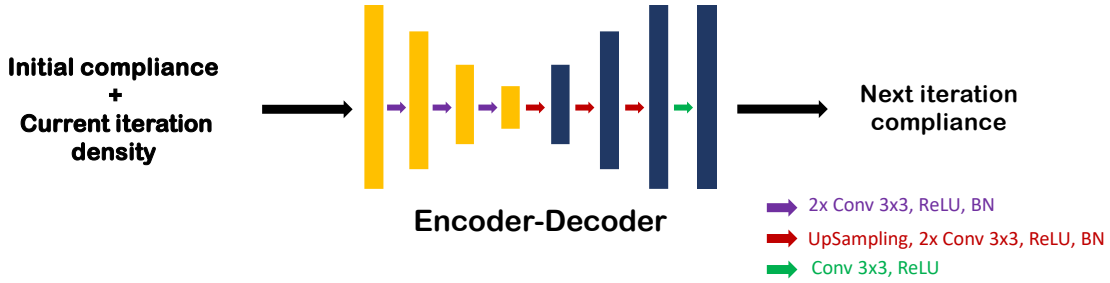

Figure 4.5 Compliance Prediction Network (CPN) prediction: This CNN Encoder-Decoder model is used to predict the next iteration compliance.

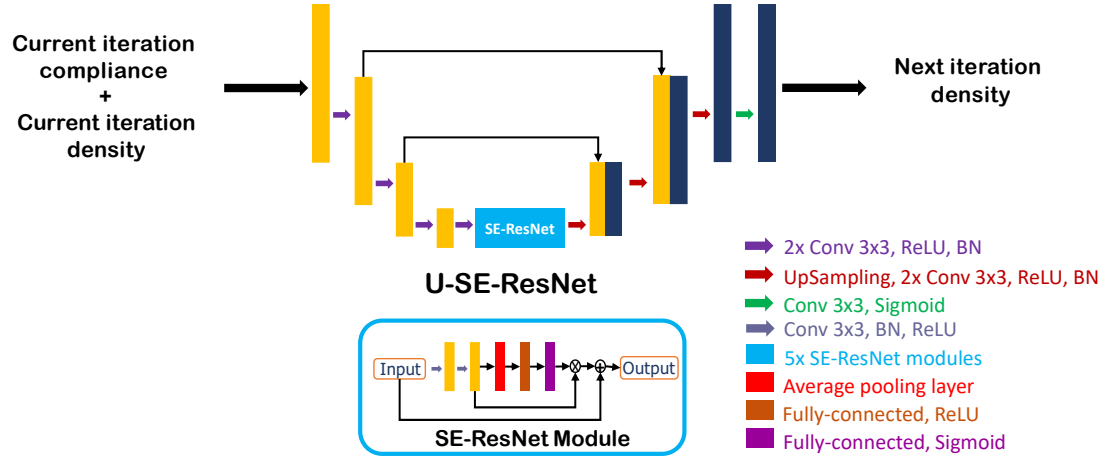

Figure 4.6 Density Prediction Network (DPN) prediction: This U-SE-ResNet[33] architecture is used to predict the next iteration density.

Neural Network Architectures in CDCS

In this section, we provide details about the different architectures used in the CDCS method. As mentioned earlier, we implemented Encoder-Decoder for CPN, U-SE-ResNet for DPN, and U-Net for the FDPN part.

The next architecture for FDPN is a U-Net [42, 8] as shown in Figure 4.6. This architecture is the modified version of encoder-decoder architecture. As we can see in Figure 4.7, the skip-connections are introduced from encoder part to decoder part at each resolution level. These connections help transfer the encoder's contextual information to the decoder for better local-

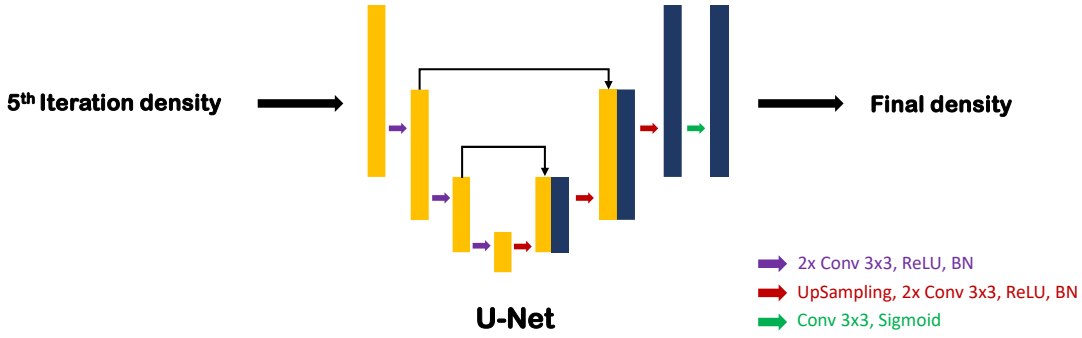

Figure 4.7 Final Density Prediction Network (FDPN) prediction: This U-Net architecture is used to predict the next iteration density.

ization [42]. The encoder and decoder of U-Net are exactly the same as in the encoder-decoder architecture discussed above.

The Encoder-Decoder architecture is a simple convolution neural network (CNN) consisting of two parts: the encoder and the decoder (Figure 4.5). The input is passed through the encoder and converted to a lower-dimensional latent space, further expanding to the higher dimension required by the decoder. The encoder is a collection of encoding blocks that consist of strided convolutional layers, followed by non-linearity (ReLU) transformation and batch normalization. Similarly, the decoder blocks of the decoder have an up-sampling layer, convolutional layers, non-linearity(ReLU), and batch normalization. Finally, we use the last convolution and non-linearity to get the output of the desired shape.

U-SE-ResNet [33] is constructed using U-Net with addition of SE-ResNet blocks as shown in Figure 4.6. Each SE-ResNet block is a combination of ResNet and Squeeze-and-Excitation(SE) blocks [16]. These blocks are introduced in the U-Net architecture at the bottle-neck region between the encoder and decoder. SE block enhances the network's performance by recalibrating the channel-wise features by explicitly weighing the inter-dependencies between channels. SE block consists of a pooling layer followed by fully-connected (FC) and ReLU transformation and again passing through the FC and sigmoid transformation, and at the end, the output of sigmoid layer is scaled by multiplying with the input of SE block with which we get the same shape as the input

of SE block. In SE-ResNet, ResNet is combined with SE block to improve the performance [33] by adding the residual connection between the input to the output of the SE block. Also, in this architecture, we use the same encoder and decoder as explained earlier.

CHAPTER 5. DATA GENERATION

5.1 2D Data Generation

The data required for training the networks is obtained by performing several simulations of topology optimization on different designs and volume fraction constraints. We represent each design using a 2D mesh made up of quadrilateral elements. The nodes of the mesh form a regular grid such that each element represents a square element. With this representation, we can directly convert the elements of the mesh to pixels of an image. Therefore, we represent the geometry as an image such that the pixel intensity values represent the element compliances and the element densities of the 2D mesh.

For training data, we need raw compliance values, the volume fraction constraint, the intermediate element densities obtained during the intermediate iterations of the structural topology optimization process, and the final element densities. We generated 30,141 simulations of the structural topology optimization with different randomly generated load values, loading directions, load locations, and a randomly generated set of nodes in the mesh, fixed with zero displacements. We performed each simulation for 150 iterations of SIMP-based structural topology optimization. All the relevant information from each structural topology optimization simulation is stored for use during the training process.

5.2 3D Data Generation

The 3D data used for DLTO is generated using ANSYS Mechanical APDL v19.2. We use a cube of length 1 meter in the form of 3D mesh as an initial design domain (see Figure 5.1). The mesh created has 31093 nodes and 154,677 elements, and each element consists of 8 nodes. To ensure we sample a diverse set of topologies from the complete distribution of topologies originating from the cube, we use several available sets of boundary and load conditions in ANSYS software such as

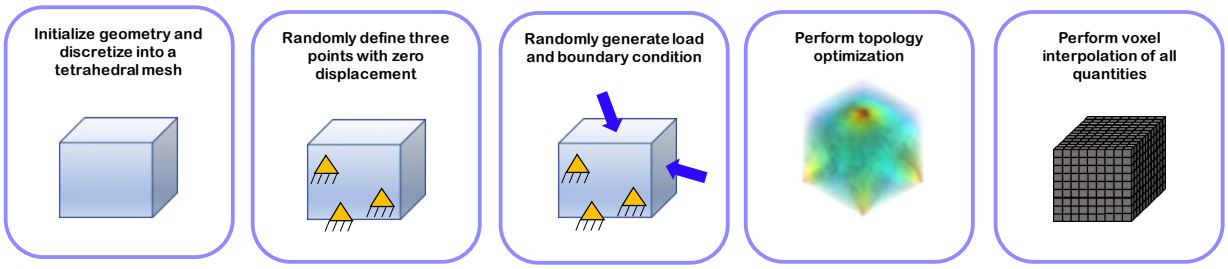

Figure 5.1 3D data generation pipeline: Each sample in the dataset if generated using this data generation pipeline. First we initialize the geometry (a cube with side length of 1 meter). This geometry is discretized into tetrahedrons to get the mesh. On this mesh, we define three non-collinear nodes to fix the mesh from any rigid body motion. Then we apply randomly generated boundary conditions and loading conditions with different magnitude and direction.

Nodal Force, Surface Force, Remote Force, Pressure, Moment, Displacement. First, we randomly sample three non-collinear nodes on one side of the cube, and we define zero displacements for these points; so they are fixed. This is necessary to avoid any rigid body motion of the geometry. The next step is to randomly select the load location, which is not close to the fixed support nodes. The nature of the load (nodal, surface, remote, pressure, or moment), the value, and direction is sampled randomly. We employ a rejection sampling strategy to ensure that each sampled topology is unique. We obtained a total of 1500 configurations of load and boundary conditions, and then by sampling the volume fraction, we generated a total of 13500 samples. In our dataset, the topology optimization took an average of 13 iterations; the minimum number of iterations is 6, and the maximum is 72; this number depends on several factors such as the mesh resolution, boundary conditions, and the target volume fraction.

In ANSYS, we store the topology optimization output, the original strain energy, and the intermediate results stored using the starting mesh representation. We now need to convert the mesh representation to a voxel representation for training 3D CNN models. This conversion process involves first discretizing the axis-aligned bounding box into a regular structured grid of voxels based on the grid's grid size/resolution. We compute the barycentric coordinates for each of the tetrahedra in the mesh for each of the voxel centers. Using the barycentric coordinates, we can

estimate if the grid point is inside the tetrahedron or not. If the grid point is inside that tetrahedron, we now interpolate the field values (such as density, strain energy, etc.) from the tetrahedron nodes to the voxel centers. Through this process, we obtain the voxel-based representation of the topology optimization data. Each sample's voxelization takes about 5-15 minutes, depending on the resolution and the number of tetrahedral elements. We parallelize this process using GNU parallel to complete this process in a few hours (depending on compute nodes' availability). To calculate the element compliance, we multiply strain energy obtained from ANSYS with the cube of the density to obtain the compliance ($C = \rho^p \mathbf{u} k_e \mathbf{u} = \rho^p * SE$, where p is the penalty of the SIMP approach, set to 3 in our data generation process, SE refers to the elemental strain energy).

Once we obtain the voxel-based representation, we also perform other preprocessing steps such as normalizing the compliance by the maximum value of the compliance, converting the compliances to log scale for better learning. We even perform on-the-fly data augmentation by rotating the model in any of the six possible orientations. Thus we finally get the data for training the neural network.

CHAPTER 6. RESULTS

We split both the datasets (i.e., for 2D and 3D geometries) into two parts for training the neural networks: training and testing dataset. Out of all data generated, we use 75% of the topology optimization data for training and the remaining 25% for testing. We use the testing dataset to evaluate the performance of all three methods. We will discuss the results for 2D and 3D topologies in the following subsections.

6.1 Results on 2D topology optimization

To compare the performance of our proposed methods with the baseline DOD method, we start with the volume fraction (VF) constraint. We compute the predicted volume fraction of the final predicted topology by averaging the density values over the whole design domain. We compute the mean-squared error (MSE) between the predicted VF and the actual VF on the test data. This metric is shown as MSE of volume fraction in Table 6.1. The values for the best performing method has been highlighted in bold in all the tables. We plot the correlation plot between the predicted and actual VF for all the three methods in Figure 6.1 and compute the Pearson's correlation coefficient between the predicted VF and actual VF for 2D test data in Table 6.2. We observe that the CDCS method performs better than the other two methods for satisfying the volume fraction constraint. Further, from the histogram plots in Figure 6.2(a), we see that CDCS consistently performs better than DOD and DS. We see comparable values for DOD ($R = 0.8986$) and CDCS ($R = 0.8945$) method, while the DS ($R = 0.6883$) method performs poorly in satisfying the VF constraint.

Next, we evaluate the performance of the methods using the physical constraint of topology optimization: the total compliance (TC). TC is the SIMP algorithm's objective function value, which it tries to minimize while simultaneously satisfying the volume fraction constraint. We compute and compare the MSE between the predicted and actual TC values. To determine the

Table 6.1 Comparison of test loss metrics of our three methods on 2D test data.

Method	MSE of VF	MSE of TC	Accuracy	BCE	MAE	MSE
DOD	0.003	1.51e+05	88.54%	**0.2354**	0.1368	**0.0737**
DS	0.006	2.35e+04	84.03%	0.4421	0.1826	0.1206
CDCS	**0.002**	**2.62e+04**	**89.40%**	0.3146	**0.1195**	0.0812

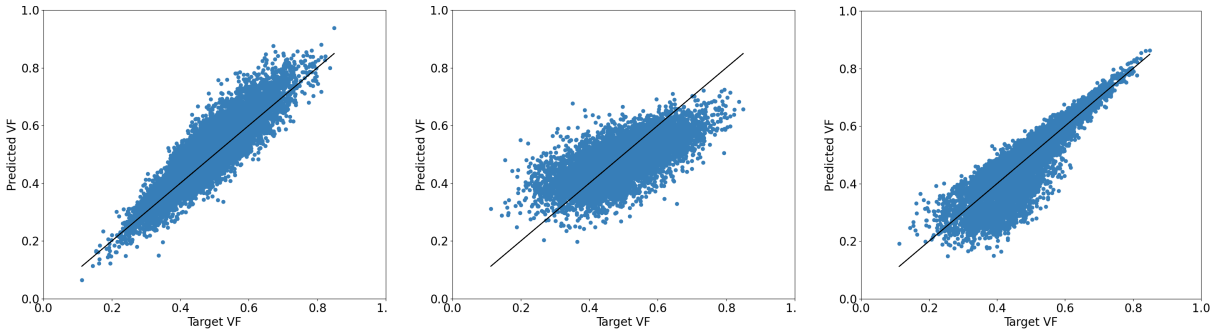

Figure 6.1 Correlation plots between predicted volume fraction and target volume fraction on 2D test data for: (a) DOD, (b) DS, (c) CDCS.

TC of the predicted final topology, we use the compliance prediction network (CPN), part of the CDCS framework, to predict the elemental compliance and take a sum of it over the whole design domain. We sum the elemental compliance of the target optimal topology to get the ground truth total compliance value; this is the optimal minimum value achieved at the end of the SIMP method. From Table 6.1, we observe almost 10× times lesser error value when we compare the MSE between the predicted and the actual total compliance for both DS and CDCS with the DOD. To support this statement, we plot the histogram of the MSE values on the 2D test data in Figure 6.2(b). We can see that both our proposed methods, DS and CDCS, predict the TC very close to the predicted optimal TC minimum value. We also compute the Pearson's correlation coefficient between predicted total compliance and actual total compliance values and the correlation plots between these two values for each of the three methods (Figure 6.4). From Table 6.2, total compliance of predicted topology by DS ($R = 0.9551$) and CDCS ($R = 0.8926$) is highly correlated with actual total compliance than the baseline DOD ($R = 0.8403$) approach.

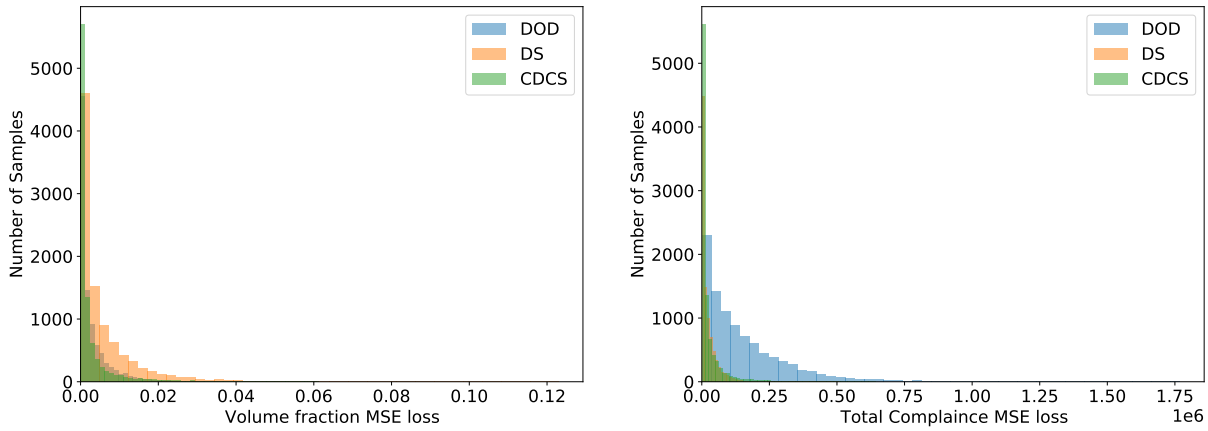

Figure 6.2 Histogram of (a) total volume fraction loss and (b) total compliance loss on the 2D test data.

Table 6.2 Comparison of correlation coefficient(R) for volume fraction and total compliance on on 2D test data.

Method	**R for volume fraction**	**R for total compliance**
DOD	**0.8986**	0.8403
DS	0.6883	**0.9551**
CDCS	0.8945	0.8926

In addition to the results above, we also perform statistical analysis on MSE loss between predicted and actual values of both topological constraints (VF) and physics constraints (TC). We summarize the minimum, median and maximum value of MSE for 2D test data in Table 6.3. We see that all the three metrics listed have comparable values for DOD and CDCS and slightly better than the DS for volume fraction constraint. For the MSE values of total compliance, for 2D data, we see that both DS and CDCS perform much better than the DOD in all three statistics, and DS and CDCS have comparable median and maximum values. However, DS has a minimum loss value in all three methods. We also compare the accuracy and different loss metrics like binary cross-entropy (BCE), mean absolute error (MAE), and mean squared error (MSE) between the density values of predicted topology and the ground truth optimal topology. The CDCS and DOD

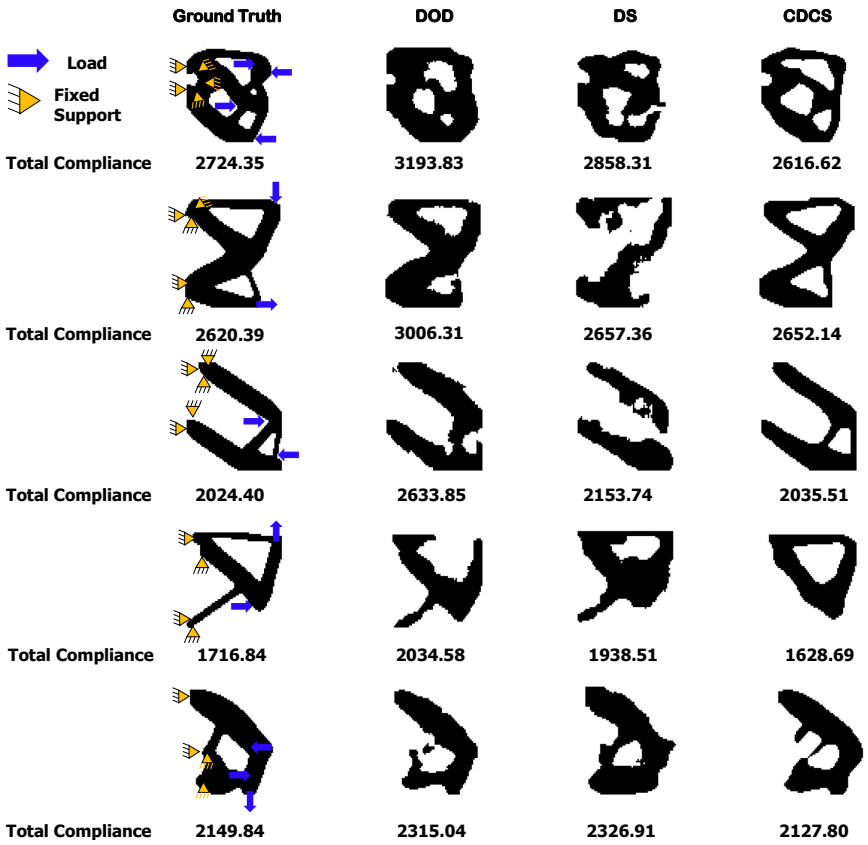

Figure 6.3 Visualization of test data in 2D: (i) Ground truth final topology shape with fixed supports and load locations (ii) Method 1: Baseline direct optimal density prediction, (iii) Method 2: Density sequence prediction, (iv) Method 3: Coupled density and compliance sequence prediction. The results show the target design and the predicted design.

method's performance is comparable in terms of these four metrics and is better than the DS method, as shown in Table 6.1.

Apart from the numerical analysis, to further qualify the performance of our method, we compare the visualizations of the predicted final topology, obtained by performing end-to-end prediction using all three methods, with ground truth final optimal topology in Figure 6.3. Additionally, we compute each sample's total compliance value in the visualization. In the ground truth column, we also show the boundary and load conditions applied for each sample. We observe that the CDCS predicts the final shape significantly closer to the ground truth, and also, the predicted total com-

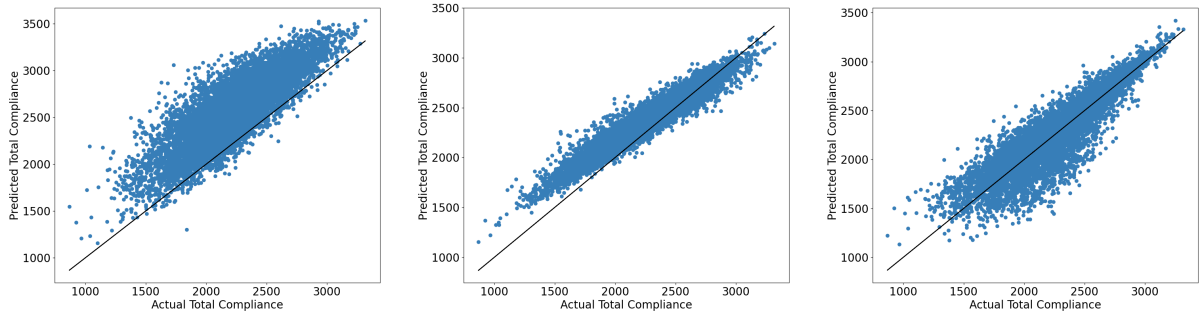

Figure 6.4 Correlation plots between predicted total compliance and actual total compliance on 2D test data for all three methods: (a) DOD, (b) DS, (c) CDCS.

Table 6.3 Statistics on the volume fraction and total compliance loss on 2D test data.

Statistics	**MSE of volume fraction**			**MSE of total compliance**		
Method	Min.	Median	Max	Min.	Median	Max
DOD	5.96e-08	1.30e-03	**6.08e-02**	5.33e-01	1.01e+05	1.76e+06
DS	6.47e-07	2.58e-03	1.23e-01	7.47e-06	1.13e+04	**5.06e+05**
CDCS	**1.85e-08**	**7.74e-04**	6.12e-02	**2.87e-04**	**8.07e+03**	6.41e+05

pliance value is much closer to the actual value. Although there are some cases where the shape predicted by DOD and DS is slightly better than the CDCS, the predicted total compliance value is much higher than the actual value.

We further evaluate the CDCS method by visualizing the evolution of intermediate iteration densities and elemental compliance predicted by the DPN and CPN, respectively. As we discussed in Section 4.3, we feed the actual initial and current iteration elemental compliance and the density values to DPN and CPN to predict the next iteration quantities. We visualize this iteration-wise evolution of the topology and its compliance in Figure 6.5 and Figure 6.6, respectively. Looking at the visualizations, it is evident that both DPN and CPN are efficient at predicting the next iteration density and compliance values. Also, it depicts the non-trivial transformation flow of the initial topology shape towards the final optimal shape.

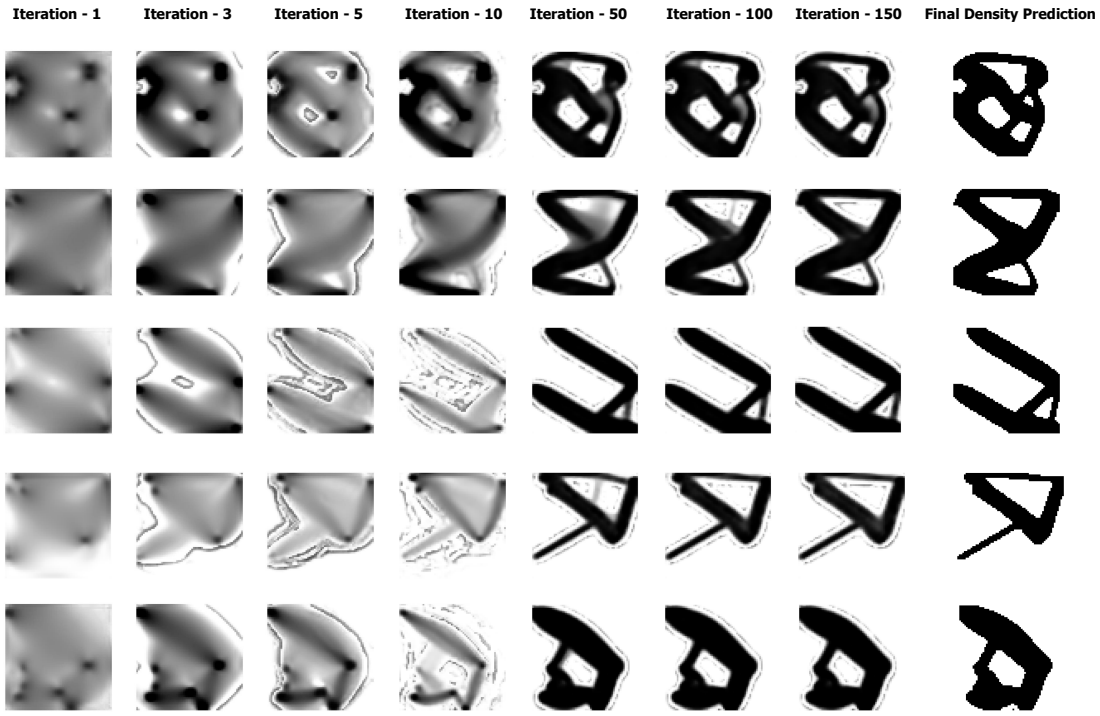

Figure 6.5 Visualizations of DPN predicting intermediate iterations density on 2D test data.

From the results discussed above, we observe the CDCS method consistently performs better than the baseline DOD and the DS method. Although DS satisfies the physics constraint (minimizing the TC) better, it does not satisfy the topological constraint (VF) to the same extent. On the other hand, CDCS accomplishes the best balance in satisfying the volume fraction constraint and achieving a total compliance value close to the actual optimal minimum value. Hence, we only extend the CDCS method to the 3D dataset and compare it with the baseline DOD method.

6.2 Results on 3D topology optimization

We perform a similar set of evaluations on the 3D data as discussed in Section 6.1 to assess the performance of the CDCS method, comparing it to the DOD method. First, comparing the MSE between the volume fraction (VF) of predicted with actual final topology, in the Table 6.4, we see that the MSE of VF using CDCS method is 2× lower than using the DOD method. From

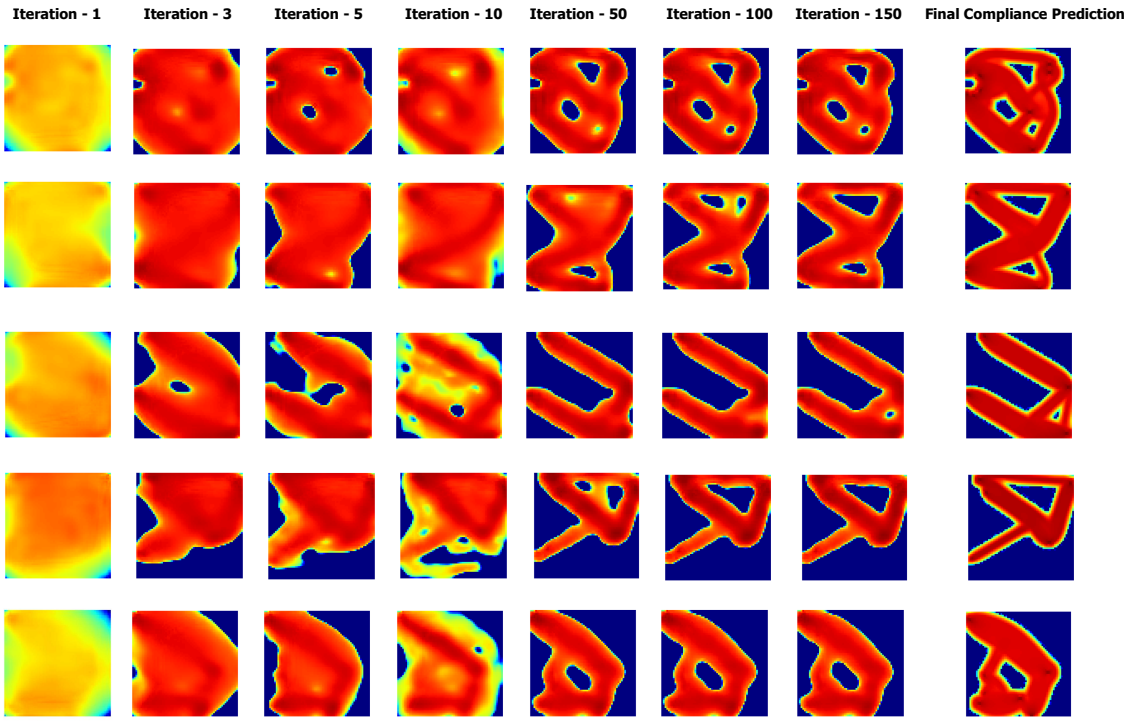

Figure 6.6 Visualizations of CPN predicting intermediate iterations compliance on 2D test data.

the histogram plots in Figure 6.8(a), we can infer that more samples have minimum MSE of VF using CDCS than the DOD. We also compute the Pearson's correlation coefficient between the VF values of predicted and actual final topology using both DOD and CDCS and notice that both values are highly correlated (Table 6.5). We confirm this visually by plotting the correlation plots in Figure 6.7.

Comparing the MSE of total compliance(TC), DOD has 2× more error value than the CDCS. Plotting the histogram for values MSE of TC in Figure 6.8(b), we see that most of the samples have the lower MSE of TC value using CDCS. In the Table 6.5, we calculate the Pearson's correlation coefficient between the TC values of the predicted and the actual optimal topology, and we observe that the TC value predicted by the CDCS($R = 0.9578$) are highly correlated to actual TC values than the DOD($R = 0.9139$). We also observe this high correlation when we plot the correlation plots in Figure 6.9.

Table 6.4 Comparison of test loss metrics using DOD and CDCS on 3D test data.

Method	MSE of VF	MSE of TC	Accuracy	BCE	MAE	MSE
DOD	0.0002	8.04e+05	**95.61%**	**0.1008**	**0.0648**	**0.0312**
CDCS	**0.0001**	**3.95e+05**	92.74%	0.1965	0.0875	0.0544

Table 6.5 Comparison of correlation coefficient(R) for volume fraction and total compliance on on 3D test data.

Method	R for volume fraction	R for total compliance
DOD	**0.9966**	0.9139
CDCS	0.9947	**0.9578**

In Table 6.6, we summarize the statistical analysis of the MSE value of both VF and TC on 3D test data. In the case of MSE of VF, all the three metrics values are comparable. We see better performance when we consider the MSE of TC. We notice that the median value using CDCS is 2× lower than the DOD method. Also, the maximum value of MSE of TC using CDCS is 15× smaller than the DOD value, which affirms the greater performance of CDCS over the baseline DOD approach in satisfying the physics constraint (TC). We have summarized the different loss metrics like BCE, MAE, MSE, and accuracy between the predicted and actual topology for both CDCS and DOD in Table 6.4. In terms of MAE and MSE, both CDCS and DOD are comparable, while DOD performs slightly better when comparing the BCE and the accuracy values. But overall, like in the case of 2D, CDCS achieves the balance of satisfying both topological (VF) and physical (TC) constraints on the 3D dataset.

As discussed earlier, the CDCS method has three different neural networks dedicated to learning the different aspects of structural topology optimization. We have compared the performance of the different architectures for each task of TO. We have experimented with three architectures, which are: (i) Encoder-Decoder architecture, (ii) U-Net [42, 8] architecture, and (iii) U-SE-ResNet [33]. For CPN, we compare MAE and MSE metrics, while for DPN and FDPN, we evaluate the performance based on the BCE, MAE, and MSE values on 3D test data. All these metric values are

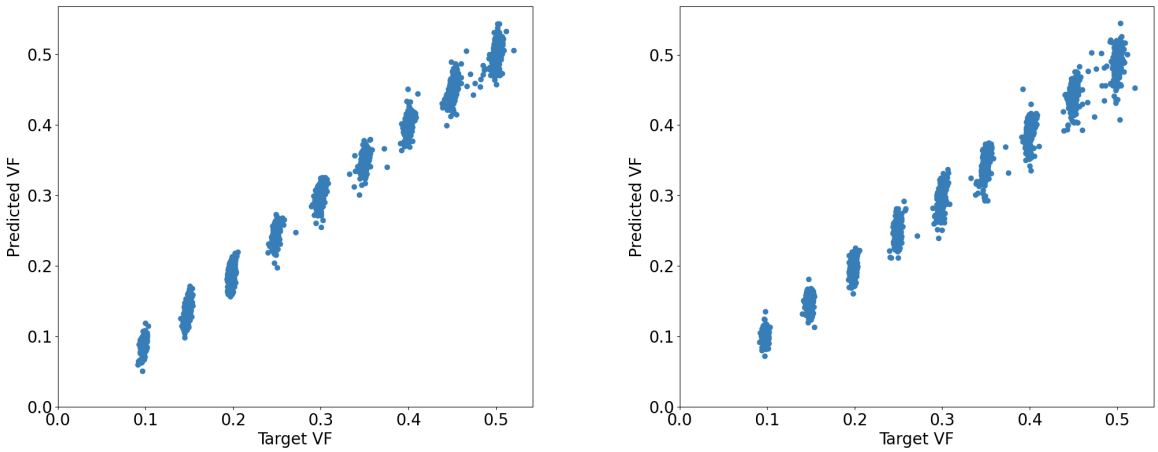

Figure 6.7 Correlation plots between predicted volume fraction and target volume fraction on 3D test data for DOD and CDCS.

Table 6.6 Statistics on the volume fraction and total compliance loss on 3D test data.

Statistics	**MSE of volume fraction**			**MSE of total compliance**		
Method	Min.	Median	Max	Min.	Median	Max
DOD	9.31e-10	**4.80e-05**	2.75e-02	4.64e-01	1.83e+05	3.26e+07
CDCS	**9.31e-10**	6.19e-05	**8.97e-03**	**1.12e-01**	**7.63e+04**	**2.35e+06**

summarized in Table 6.7. From the Table 6.7 we selected the best of three for each task, like for CPN, we implemented Encoder-Decoder, for DPN used U-SE-ResNet, and similarly, for FDPN, we used U-Net architectures.

For 3D visualizations, we use marching cube methods to visualize the predicted and actual optimal topology shapes. As mentioned earlier in Section 6.1, using the end-to-end prediction, we obtain the predicted final topology. We visualize some samples from the test data(in-distribution samples) as well as some out-of-distribution samples. As discussed in Section 5.2 about the 3D data generation, the in-distribution dataset has three nodes with fixed support and one loading condition. On the other hand, we generated few samples with more than three fixed support locations and multiple loads acting on the topology; we termed these samples as out-of-distribution samples. We visualize some samples from the in-distribution test data in Figure 6.10 and the out-

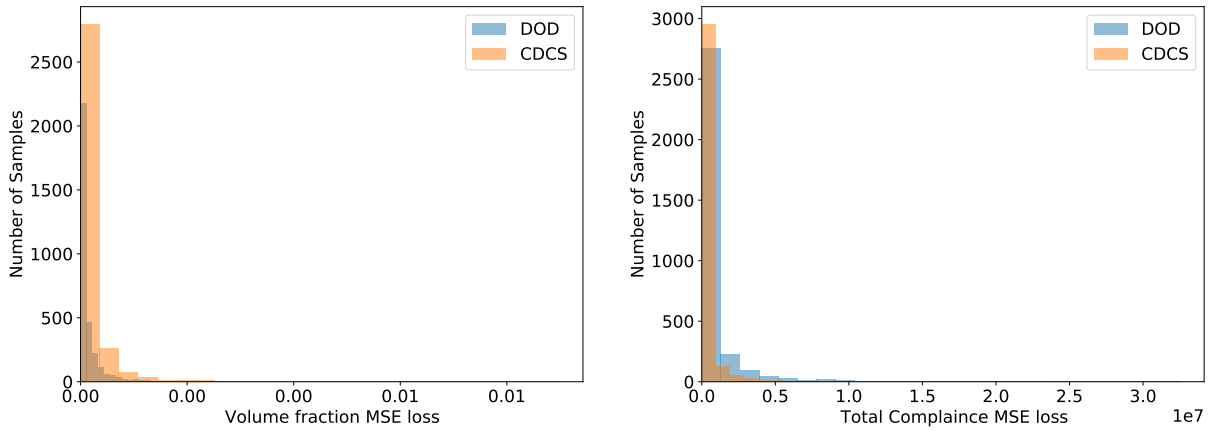

Figure 6.8 Histogram of (a) total volume fraction loss and (b) total compliance loss on the 3D test data.

Table 6.7 Comparison of different neural network architectures for each task of CDCS on 3D test data.

Method	**CPN**		**DPN**			**FDPN**		
Architecture	MAE	MSE	BCE	MAE	MSE	BCE	MAE	MSE
AE	**0.0221**	**0.0009**	0.2838	0.0211	0.0014	**0.1140**	0.0178	0.0026
U-Net	0.0286	0.0013	**0.2810**	0.0144	0.0005	0.1152	**0.0145**	**0.0019**
U-SE-ResNet	0.0294	0.0016	0.3157	**0.0131**	**0.0006**	0.1188	0.0155	0.0021

of-distribution samples in Figure 6.11. In both figures, the first column shows the initial geometry and the locations of the fixed support and the load. We notice that CDCS performs significantly better than the baseline DOD on in-distribution test data and even on out-of-distribution samples. We see much smoother shapes, even smoother than actual ground truth was obtained by CDCS. We also calculated the total compliance (TC) value for each sample. We see that the TC value of topology predicted using CDCS is very close to the ground truth TC value than using the DOD.

From the numerical analysis performed and supported by the visualizations on both 2D and 3D datasets, we claim that the performance of the CDCS is better than the baseline DOD and DS.

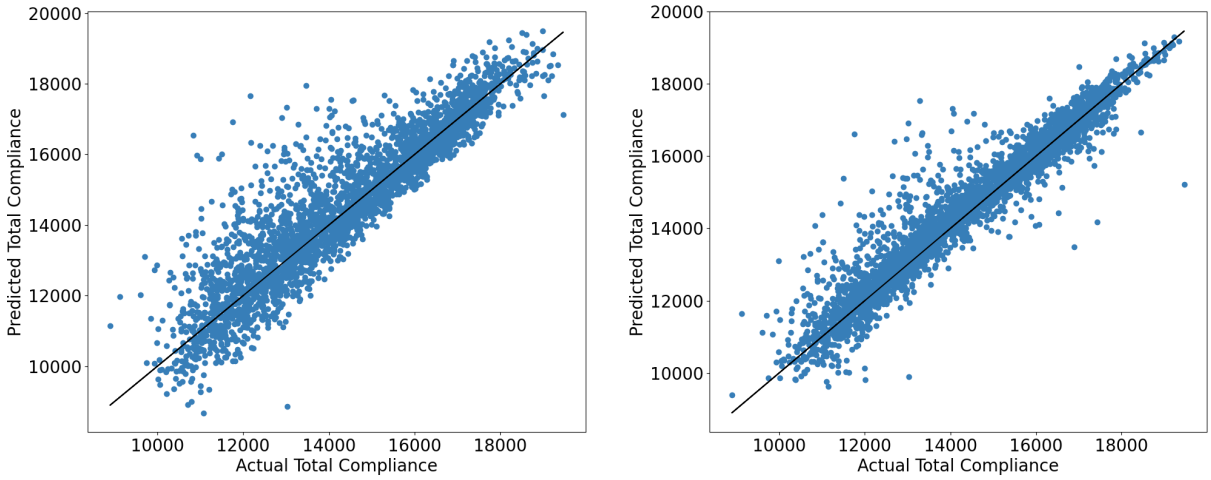

Figure 6.9 Correlation plots between predicted total compliance and actual total compliance on 3D test data for DOD and CDCS.

6.3 Performance Plots During Training of Both 2D and 3D Data

This section has summarized different performance plots of training the neural networks used in all three methods on both 2D and 3D data. Figure 6.12 shows the L2 and L1 loss plots when DOD is used to predict the first, second, fifth, tenth, and final densities of the 2D dataset. In Figure 6.13, the left plot is showing the training losses(L2 loss, L1 loss, and L2 loss of VF) of IDPN(Phase1 of DS), and similarly, the right plot is of training losses(L2 loss and L2 loss of VF) of DPN(Phase2 of DS). Figure 6.14 shows the training losses of all three networks of the CDCS method on 2D data. Similarly for 3D dataset, Figure 6.15 has the plot of training losses of DOD, Figure 6.16 has the loss plots for each of the three networks of CDCS. From these loss plots, we see the losses decrease as we progress in training.

We also plot the histograms of different loss metrics values, between predicted and actual topology, like BCE, MAE and MSE loss for all the methods on 2D and 3D test dataset in Figure 6.17 and Figure 6.18, respectively. Based on these histograms of metrics, in the case of the 2D dataset, CDCS performs better than DOD and DS, while in the case of the 3D dataset, CDCS and DOD have comparable performance.

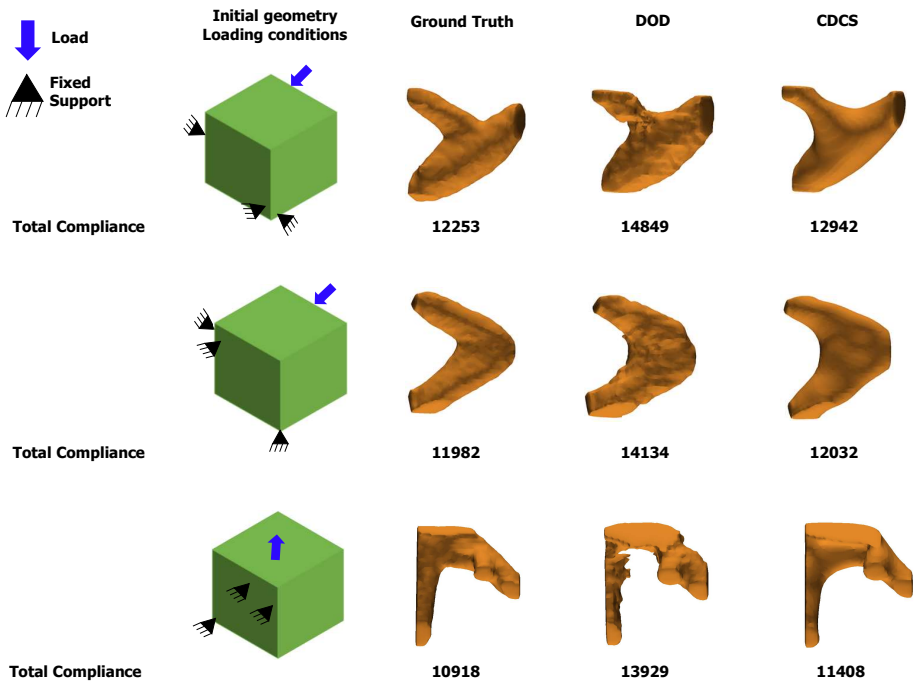

Figure 6.10 Visualization of In-distribution test data in 3D: (i) Initial geometry with the fixed supports and load locations, (ii) Ground Truth final topology (iii) Method 1: Baseline direct optimal density prediction, (iii) Method 3: Coupled density and compliance sequence prediction. The results show the target shape and the predicted shape.

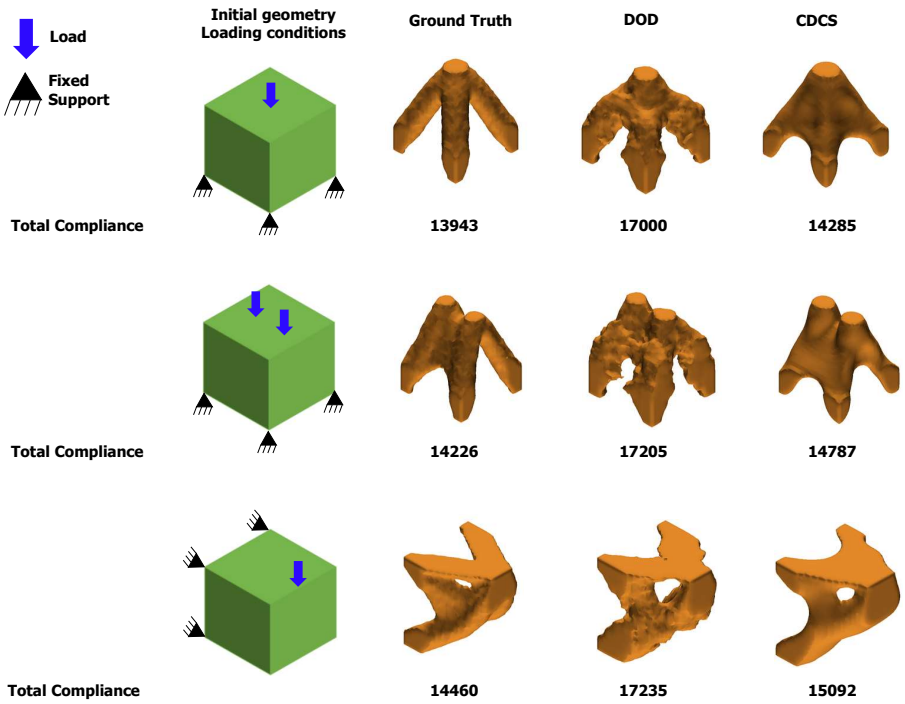

Figure 6.11 Visualization of Out-distribution test data in 3D: (i) Initial geometry with the fixed supports and load locations, (ii) Ground Truth final topology (iii) Method 1: Baseline direct optimal density prediction, (iii) Method 3: Coupled density and compliance sequence prediction. The results show the target shape and the predicted shape.

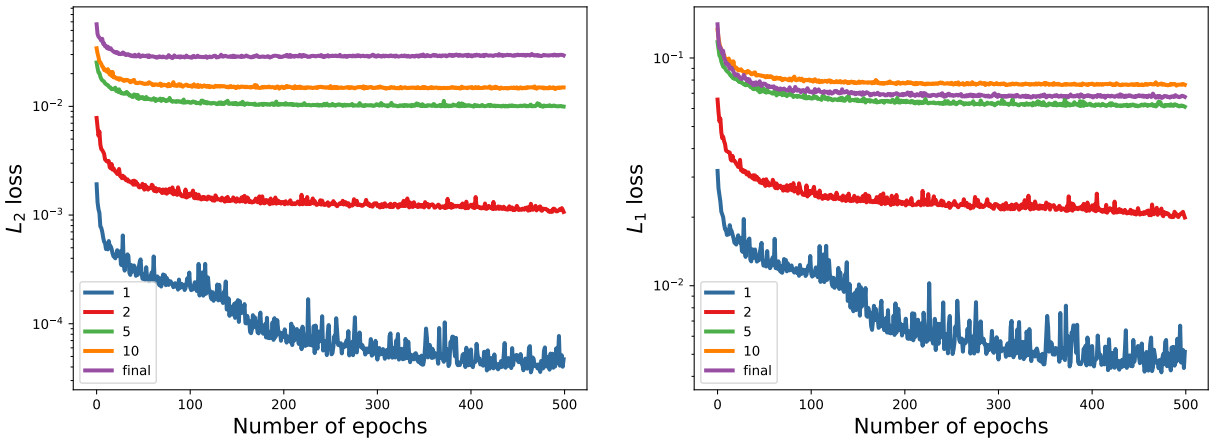

Figure 6.12 Performance plots of intermediate density prediction networks while predicting first, second, fifth, tenth intermediate densities and final density. (a) L_2 loss (b) L_1 loss.

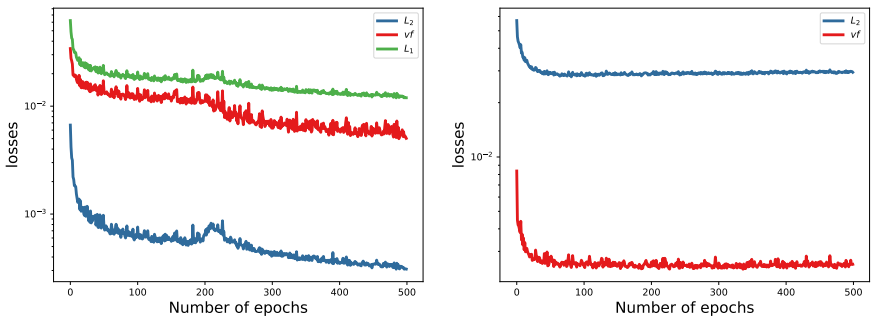

Figure 6.13 Performance plots of two phases of Density sequence prediction method on 2D data. (a) DS Phase1:IDPN loss (b) DS Phase2: DTN loss.

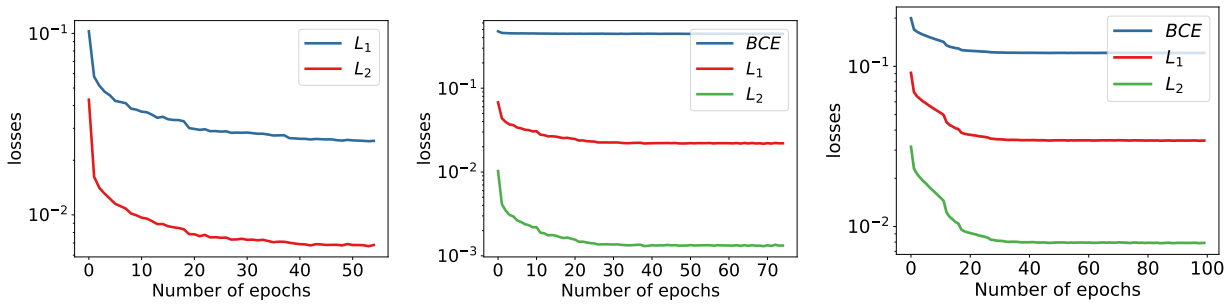

Figure 6.14 Performance plots of (a) Compliance Prediction Network (CPN) and (b) Density Prediction Network (DPN) and (c) Final Density Prediction Network (FDPN). Each plot shows different loss functions used in training with 2D data.

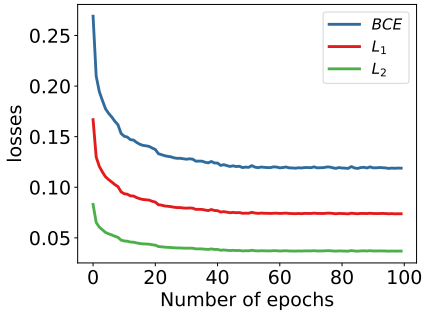

Figure 6.15 Performance plots of Direct Optimal Density Prediction. Plot shows different loss functions used in training with 3D data.

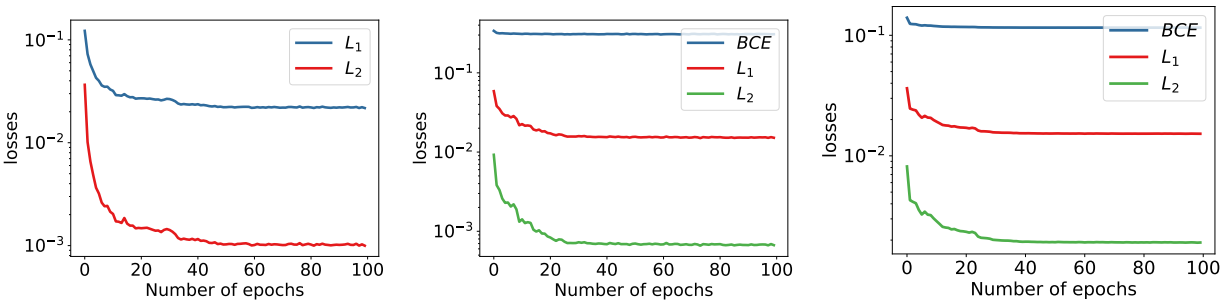

Figure 6.16 Performance plots of (a) Compliance Prediction Network (CPN) and (b) Density Prediction Network (DPN) and (c) Final Density Prediction Network (FDPN). Each plot shows different loss functions used in training with 3D data.

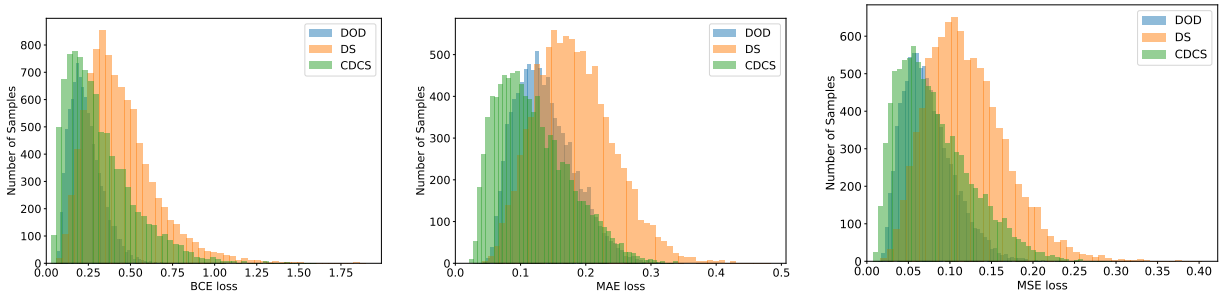

Figure 6.17 Distribution of BCE, MAE, MSE losses on 2D test data.

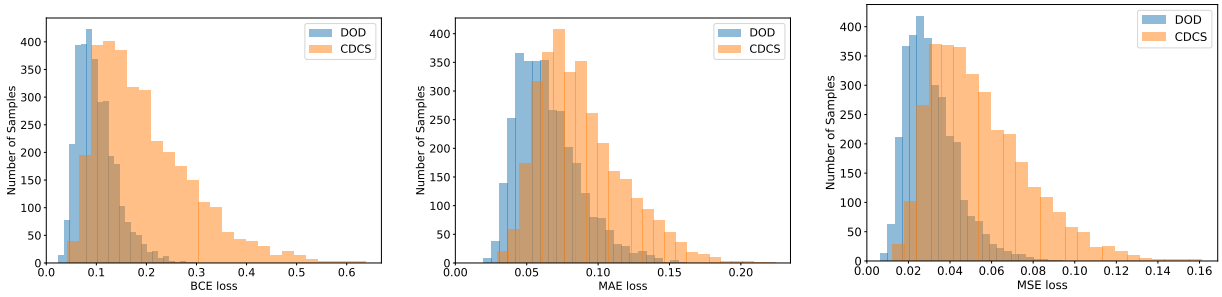

Figure 6.18 Distribution of BCE, MAE, MSE losses on 3D test data.

CHAPTER 7. CONCLUSION AND FUTURE WORK

7.1 Conclusion

In this thesis, we explore the application of algorithmically-consistent deep learning methods for structural topology optimization. We developed two approaches (density sequence and coupled density compliance sequence models), consistent with the physics constraints, topological constraints, and the SIMP topological optimization algorithm. We generated datasets for topology optimization in both 2D and 3D representations and then demonstrated the superior performance of our proposed approach over a direct density-based baseline approach. Finally, we visualize a few anecdotal topology optimization samples to visually compare the three methods with the SIMP-based topology optimization process.

7.2 Future Work

The research conducted in this thesis can be improved by several ways. Some of the future steps are listed below:

1. Current topology optimization is performed on the geometries with a simple shape as initial geometry. To make it more generalized to real-world designs, including 3D topology optimization performed on a generic 3D CAD model will help and learn the PDE underlying the structural analysis to reduce training data requirements.

2. Topology optimization provides optimal design ready for additive manufacturing. Adding manufacturing constraints in the deep learning-based structural topology optimization would create a design that can be directly manufactured. One example would be imposing the building direction 'b' constraint for additive manufacturing as shown in Figure 7.1.

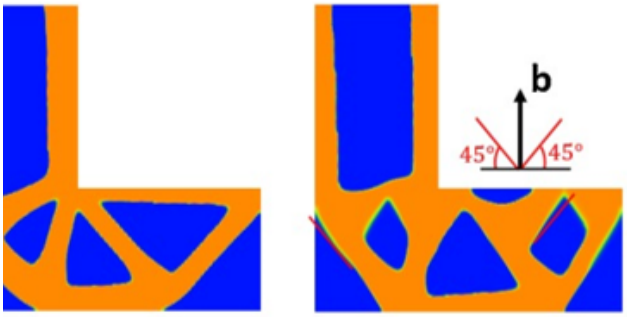

Figure 7.1 Imposing building direction 'b' constraint for additive manufacturing [52]

We believe that our proposed algorithmically consistent approach for topology optimization provides superior quality results and can considerably speed up the topology optimization process over existing finite-element-based approaches.

BIBLIOGRAPHY

[1] Abueidda, D.W., Koric, S., Sobh, N.A., 2020. Topology optimization of 2D structures with nonlinearities using deep learning. Computers & Structures 237, 106283.

[2] Banga, S., Gehani, H., Bhilare, S., Patel, S., Kara, L., 2018. 3D topology optimization using convolutional neural networks. arXiv preprint arXiv:1808.07440 .

[3] Bendsøe, M.P., 1989. Optimal shape design as a material distribution problem. Structural optimization 1, 193–202.

[4] Bendsøe, M.P., Kikuchi, N., 1988. Generating optimal topologies in structural design using a homogenization method. Computer Methods in Applied Mechanics and Engineering .

[5] Bhatnagar, S., Afshar, Y., Pan, S., Duraisamy, K., Kaushik, S., 2019. Prediction of aerodynamic flow fields using convolutional neural networks. Computational Mechanics , 1–21.

[6] Bujny, M., Aulig, N., Olhofer, M., Duddeck, F., 2018. Learning-based topology variation in evolutionary level set topology optimization, in: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 825–832.

[7] Chandrasekhar, A., Suresh, K., 2021. Tounn: Topology optimization using neural networks. Structural and Multidisciplinary Optimization 63, 1–15. doi:10.1007/s00158-020-02748-4.

[8] Çiçek, Ö., Abdulkadir, A., Lienkamp, S.S., Brox, T., Ronneberger, O., 2016. 3D U-Net: learning dense volumetric segmentation from sparse annotation, in: International conference on medical image computing and computer-assisted intervention, Springer. pp. 424–432.

[9] Das, R., Jones, R., Xie, Y.M., 2011. Optimal topology design of industrial structures using an evolutionary algorithm. Optimization and Engineering 12, 681–717.

[10] De, S., Hampton, J., Maute, K., Doostan, A., 2019. Topology optimization under uncertainty using a stochastic gradient-based approach. arXiv:1902.04562.

[11] Doi, S., Sasaki, H., Igarashi, H., 2019. Multi-Objective Topology Optimization of Rotating Machines Using Deep Learning. IEEE Transactions on Magnetics 55, 1–5. doi:10.1109/TMAG.2019.2899934.

[12] Eschenauer, H.A., Olhoff, N., 2001. Topology optimization of continuum structures: a review. Appl. Mech. Rev. 54, 331–390.

[13] Guo, T., Lohan, D.J., Cang, R., Ren, M.Y., Allison, J.T., 2018. An indirect design representation for topology optimization using variational autoencoder and style transfer, in: 2018 AIAA/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference, p. 0804.

[14] Hamdia, K.M., Ghasemi, H., Bazi, Y., AlHichri, H., Alajlan, N., Rabczuk, T., 2019. A novel deep learning based method for the computational material design of flexoelectric nanostructures with topology optimization. Finite Elements in Analysis and Design 165, 21–30. URL: https://www.sciencedirect.com/science/article/pii/S0168874X1930023X, doi:https://doi.org/10.1016/j.finel.2019.07.001.

[15] Hochreiter, S., Schmidhuber, J., 1997. Long Short-Term Memory. Neural Comput. 9, 1735–1780. URL: https://doi.org/10.1162/neco.1997.9.8.1735, doi:10.1162/neco.1997.9.8.1735.

[16] Hu, J., Shen, L., Sun, G., 2018. Squeeze-and-Excitation Networks, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7132–7141. doi:10.1109/CVPR.2018.00745.

[17] Ioffe, S., Szegedy, C., 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift, in: Bach, F., Blei, D. (Eds.), Proceedings of the 32nd International Conference on Machine Learning, PMLR, Lille, France. pp. 448–456. URL: http://proceedings.mlr.press/v37/ioffe15.html.

[18] Jagtap, A.D., Karniadakis, G.E., 2019. Adaptive activation functions accelerate convergence in deep and physics-informed neural networks. arXiv preprint arXiv:1906.01170 .

[19] Jang, S., Kang, N., 2020. Generative design by reinforcement learning: Maximizing diversity of topology optimized designs. arXiv:2008.07119.

[20] Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 .

[21] Kollmann, H.T., Abueidda, D.W., Koric, S., Guleryuz, E., Sobh, N.A., 2020. Deep learning for topology optimization of 2D metamaterials. Materials & Design 196, 109098. doi:https://doi.org/10.1016/j.matdes.2020.109098.

[22] Krizhevsky, A., Sutskever, I., Hinton, G.E., 2017. ImageNet Classification with Deep Convolutional Neural Networks. Commun. ACM 60, 84–90. URL: https://doi.org/10.1145/3065386, doi:10.1145/3065386.

[23] Kukačka, J., Golkov, V., Cremers, D., 2017. Regularization for Deep Learning: A Taxonomy. arXiv:1710.10686.

[24] Lagaros, N., Kallioras, N., Kazakis, G., 2020. Accelerated topology optimization by means of deep learning. Structural and Multidisciplinary Optimization 62. doi:10.1007/s00158-020-02545-z.

[25] Lecun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE 86, 2278–2324. doi:10.1109/5.726791.

[26] Lee, S., Kim, H., Lieu, Q.X., Lee, J., 2020. CNN-based image recognition for topology optimization. Knowledge-Based Systems 198, 105887. URL: https://www.sciencedirect.com/science/article/pii/S0950705120302379, doi:https://doi.org/10.1016/j.knosys.2020.105887.

[27] Li, B., Huang, C., Li, X., Zheng, S., Hong, J., 2019. Non-iterative structural topology optimization using deep learning. Computer-Aided Design 115, 172–180. URL: https://www.sciencedirect.com/science/article/pii/S001044851930185X, doi:https://doi.org/10.1016/j.cad.2019.05.038.

[28] Lin, Q., Hong, J., Liu, Z., Li, B., Wang, J., 2018. Investigation into the topology optimization for conductive heat transfer based on deep learning approach. International Communications in Heat and Mass Transfer 97. doi:https://doi.org/10.1016/j.icheatmasstransfer.2018.07.001.

[29] Liu, J., Ma, Y., 2016. A survey of manufacturing oriented topology optimization methods. Advances in Engineering Software 100, 161–175.

[30] Lu, L., Meng, X., Mao, Z., Karniadakis, G.E., 2019. DeepXDE: A deep learning library for solving differential equations. arXiv preprint arXiv:1907.04502 .

[31] Mikolov, T., Karafiát, M., Burget, L., Cernocký, J., Khudanpur, S., 2010. Recurrent neural network based language model, in: Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010, pp. 1045–1048.

[32] Mukkamala, M.C., Hein, M., 2017. Variants of RMSProp and Adagrad with Logarithmic Regret Bounds, in: Proceedings of the 34th International Conference on Machine Learning - Volume 70, JMLR.org. p. 2545–2553.

[33] Nie, Z., Lin, T., Jiang, H., Kara, L.B., 2020. TopologyGAN: Topology Optimization Using Generative Adversarial Networks Based on Physical Fields Over the Initial Domain. arXiv preprint arXiv:2003.04685 arXiv:2003.04685.

[34] Oh, S., Jung, Y., Kim, S., Lee, I., Kang, N., 2019. Deep Generative Design: Integration of Topology Optimization and Generative Models. Journal of Mechanical Design 141. URL: http://dx.doi.org/10.1115/1.4044229, doi:10.1115/1.4044229.

[35] Oh, S., Jung, Y., Lee, I., Kang, N., 2018. Design automation by integrating generative adversarial networks and topology optimization, in: International Design Engineering Technical Conferences and Computers and Information in Engineering Conference, American Society of Mechanical Engineers. p. V02AT03A008.

[36] Orme, M.E., Gschweitl, M., Ferrari, M., Madera, I., Mouriaux, F., 2017. Designing for additive manufacturing: lightweighting through topology optimization enables lunar spacecraft. Journal of Mechanical Design 139.

[37] Pakravan, S., Mistani, P.A., Aragon-Calvo, M.A., Gibou, F., 2020. Solving inverse-pde problems with physics-aware neural networks. arXiv preprint arXiv:2001.03608 .

[38] Pan, S., Duraisamy, K., 2019. Physics-Informed Probabilistic Learning of Linear Embeddings of Non-linear Dynamics With Guaranteed Stability. arXiv preprint arXiv:1906.03663 .

[39] Qian, C., Ye, W., 2020. Accelerating gradient-based topology optimization design with dual-model artificial neural networks. Structural and Multidisciplinary Optimization , 1–21doi:10.1007/s00158-020-02770-6.

[40] Raissi, M., Perdikaris, P., Karniadakis, G.E., 2017. Physics Informed Deep Learning (Part I): Data-driven Solutions of Nonlinear Partial Differential Equations. arXiv:1711.10561.

[41] Rawat, S., Shen, M.H.H., 2019. A Novel Topology Optimization Approach using Conditional Deep Learning. arXiv:1901.04859.

[42] Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical image computing and computer-assisted intervention, Springer. pp. 234–241.

[43] Sasaki, H., Igarashi, H., 2018. Topology optimization of IPM motor with aid of deep learning. International Journal of Applied Electromagnetics and Mechanics 59, 1–10. doi:10.3233/JAE-171164.

[44] Shah, V., Joshi, A., Ghosal, S., Pokuri, B., Sarkar, S., Ganapathysubramanian, B., Hegde, C., 2019. Encoding Invariances in Deep Generative Models. arXiv preprint arXiv:1906.01626 .

[45] Shen, M.H.H., Chen, L., 2019. A New CGAN Technique for Constrained Topology Design Optimization. arXiv:1901.07675.

[46] Sigmund, O., Maute, K., 2013. Topology optimization approaches. Structural and Multidisciplinary Optimization 48, 1031–1055.

[47] Singh, R., Shah, V., Pokuri, B., Sarkar, S., Ganapathysubramanian, B., Hegde, C., 2018. Physics-aware Deep Generative Models for Creating Synthetic Microstructures. arXiv preprint arXiv:1811.09669 .

[48] Sosnovik, I., Oseledets, I., 2019. Neural networks for topology optimization. Russian Journal of Numerical Analysis and Mathematical Modelling 34, 215–223.

[49] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15, 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[50] Takahashi, Y., Suzuki, Y., Todoroki, A., 2019. Convolutional Neural Network-based Topology Optimization (CNN-TO) By Estimating Sensitivity of Compliance from Material Distribution. arXiv:2001.00635.

[51] Teichert, G., Natarajan, A., Van der Ven, A., Garikipati, K., 2019. Machine learning materials physics: Integrable deep neural networks enable scale bridging by learning free energy functions. Computer Methods in Applied Mechanics and Engineering 353, 201–216.

[52] Wang, C., Qian, X., 2020. Simultaneous optimization of build orientation and topology for additive manufacturing. Additive Manufacturing 34, 101246. URL: https://www.sciencedirect.com/science/article/pii/S2214860420306187, doi:https://doi.org/10.1016/j.addma.2020.101246.

[53] Wang, M.Y., Wang, X., Guo, D., 2003. A level set method for structural topology optimization. Computer methods in applied mechanics and engineering 192, 227–246.

[54] Xie, Y.M., Steven, G.P., 1993. A simple evolutionary procedure for structural optimization. Computers & structures 49, 885–896.

[55] Xu, K., Li, J., Zhang, M., Du, S.S., Kawarabayashi, K.i., Jegelka, S., 2020. What Can Neural Networks Reason About? ICLR 2020 .

[56] Yu, Y., Hur, T., Jung, J., Jang, I.G., 2019. Deep learning for determining a near-optimal topological design without any iteration. Structural and Multidisciplinary Optimization 59, 787–799.

[57] Zeiler, M.D., 2012. ADADELTA: An Adaptive Learning Rate Method. arXiv:1212.5701.

[58] Zhang, D., Guo, L., Karniadakis, G.E., 2019a. Learning in Modal Space: Solving Time-Dependent Stochastic PDEs Using Physics-Informed Neural Networks. arXiv preprint arXiv:1905.01205 .

[59] Zhang, D., Lu, L., Guo, L., Karniadakis, G.E., 2019b. Quantifying total uncertainty in physics-informed neural networks for solving forward and inverse stochastic problems. Journal of Computational Physics .

[60] Zhang, Y., Peng, B., Zhou, X., Xiang, C., Wang, D., 2020. A deep Convolutional Neural Network for topology optimization with strong generalization ability. arXiv:1901.07761.

[61] Zhou, Y., Zhan, H., Zhang, W., Zhu, J., Bai, J., Wang, Q., Gu, Y., 2020. A new data-driven topology optimization framework for structural optimization. Computers & Structures 239, 106310.