Taming Connectedness in Machine-Learning-based Topology Optimization with Connectivity Graphs

Mohammad Mahdi Behzadi[a], Jiangce Chen[b], Horea T. Ilies[a]

[a]Department of Mechanical Engineering, University of Connecticut,
[b]Department of Mechanical Engineering, Carnegie Mellon University,

Abstract

Despite the remarkable advancements in machine learning (ML) techniques for topology optimization, the predicted solutions often overlook the necessary structural connectivity required to meet the load-carrying demands of the resulting designs. Consequently, these predicted solutions exhibit subpar structural performance because disconnected components are unable to bear loads effectively and significantly compromise the manufacturability of the designs.

In this paper, we propose an approach to enhance the topological accuracy of ML-based topology optimization methods by employing a predicted dual connectivity graph. We show that the tangency graph of the Maximal Disjoint Ball Decomposition (MDBD), which accurately captures the topology of the optimal design, can be used in conjunction with a point transformer network to improve the connectivity of the design predicted by Generative Adversarial Networks and Convolutional Neural Networks. Our experiments show that the proposed method can significantly improve the connectivity of the final predicted structures. Specifically, in our experiments the error in the number of disconnected components was reduced by a factor of 4 or more without any loss of accuracy. We demonstrate the flexibility of our approach by presenting examples including various boundary conditions (both seen and unseen), domain resolutions, and initial design domains. Importantly, our method can seamlessly integrate with other existing deep learning-based optimization algorithms, utilize training datasets with models using any valid geometric representations, and naturally extend to three-dimensional applications.

Keywords: Topological Connectivity, Topology Optimization, Maximal Disjoint Ball Decomposition, Point Transformer, Connectivity Graphs.

1. Introduction

Machine learning techniques have gained momentum in practically all fields of engineering and science, and topology optimization (TO) is no exception. In fact, a large number of papers have been published recently and are aimed at reducing the computational expense associated with gradient-based topology optimization. These methods effectively generate predictions that closely resemble the output of optimal structures obtained through gradient-based optimization, and even exhibit some level of generalizability in exploring the design space. However, these predictive methods still face practical challenges. A significant and common limitation of these methods is their tendency to produce designs with structural disconnections and other undesirable features. Consequently, these suboptimal designs exhibit weaker structural performance and reduced manufacturability compared to the optimal solutions predicted by gradient-based optimization methods [1]. To enhance the quality of predicted structures, researchers have explored various techniques. Some approaches involve using extensive training datasets [2] or introducing physical constraints during the training process [3]. However, these methods are inefficient and computationally demanding, as generating large amounts of data and performing finite element analysis are both computationally expensive. We have previously demonstrated in [4] that by incorporating a topological loss function based on persistent homology into the training process, the number of structural disconnections is reduced by almost a factor of 2. However, this approach requires the explicit representation of topological features, and existing implementations of persistent homology metrics [5, 6] are computationally expensive - see also section 2. Consequently, the training process becomes time-consuming. Moreover, integrating additional loss function terms into the model requires frequent tuning of weight parameters for each new network or dataset, adding an additional layer of complexity and time investment.

One of the key similarities between almost all of the proposed learning-based topology optimization algorithms is that, in the end, they all produce structures represented as binary images in 2D or 3D. While this representation has clear advantages, including ease of understanding, straightforward visualization, and compatibility with powerful image-based ML models and algorithms, it also has several important limitations. For example, such a representation induces a very large number of variables, which becomes even more challenging in 3D, requiring large or very large-scale deep learning networks. Additionally, even a small number of prediction errors in individual cells (pixels in 2D and voxels in 3D) can result in structural discontinuity (i.e., partially connected or disconnected members) within the final structure.

Intuitively, reducing the number of variables should improve the quality of the predictions, but there are limited options to do so for standard image-based representations. While reducing the resolution intuitively leads to fewer variables, it also results in a loss of crucial geometric details necessary for predicting various real-world structures. In this work, we exploit the properties of the Maximal Disjoint Ball Decomposition (MDBD) proposed in [7] to convert a binary image representing a structure into a

*Corresponding author
Email address: horea.ilies@uconn.edu (Horea T. Ilies)

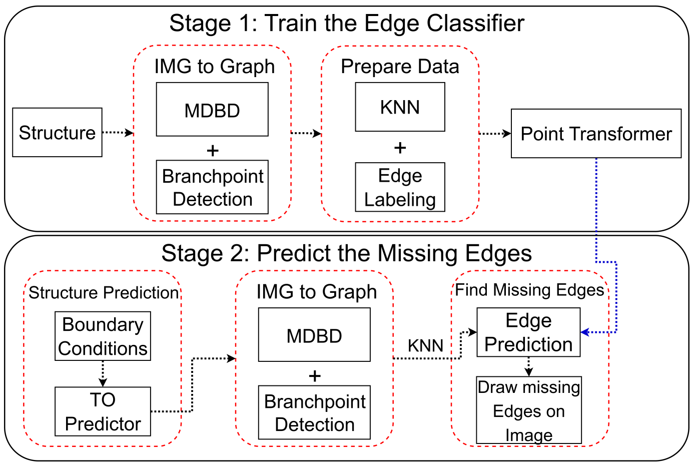


Figure 1: The proposed framework. In the initial phase, the structure is transformed into a graph using a combination of ball centers from MDBD and a branch point detection method to identify additional nodes of the graph. The edges are determined by connecting the adjacent nodes in the thinned structure. To further refine the graph, we use the K-Nearest Neighbors (KNN) algorithm to add additional potential edges. A Point Transformer Network is then trained on the labeled graphs. In the second stage, the optimal structure is predicted using boundary conditions and a TO predictor. This structure is then converted into a graph, where each node is connected to its K nearest neighbors using KNN. The trained model is then utilized to determine which edges should be retained and which should be discarded. The final step is to project the new edges onto the predicted structure and integrate them with the rest of the structure.

graph while maintaining its original topology. Our approach involves connecting each node in the graph to its closest neighbors and assigning binary labels to the edges. By doing so, we effectively reframe the task of enhancing topological connectivity as an edge-label classification problem.

2. Background

The high computational cost of gradient-based TO algorithms prompted the development of a number of machine learning methods that, once trained, can make predictions almost instantaneously. Recently proposed CNN algorithms can predict the optimal topology for 2D [8, 9, 10, 11] and 3D [12] domains. However, these algorithms have difficulties in predicting a high-quality and well-connected optimal structure.

Several recent papers have improved the quality of the generated structures by adding a large amount of data to their training process. For instance, the framework developed in [11] uses 80,000 low-resolution training cases and their corresponding high-resolution data to train and validate their deep learning model. By using a large amount of data, they have shown good prediction accuracy in terms of pixel-wise density errors. Their results were used as a benchmark in [2], where the authors used 296,000 training cases to train their CNN model. By using a training dataset three times as large as the one in [11], the authors in [2] reported improved prediction accuracy in terms of pixel-wise density errors, as expected. However, these approaches require very large amounts of training data, making it computationally prohibitive to apply them to higher resolution domains due to the computational expense of generating such large training sets. Moreover, these methods are unable to control the topological connectivity of their predictions, resulting in a potentially large prediction error in terms of design performance.

More complex networks have been proposed to improve the quality of the predicted structure. For example, generative adversarial networks (GANs) have been used in [13, 14], and the model developed in [13] has shown some generalizability. However, these models are often difficult to train, require a large amount of data, and frequently produce structures with disconnected or partially connected members, particularly for boundary conditions outside the training distribution. In an effort to generate better structures than those produced by GANs in terms of manufacturability, the model described in [15] uses a conditional diffusion model called TopoDiff. It utilizes external guidance strategies to minimize mechanical compliance and improve manufacturability by employing two neural networks responsible for predicting the compliance of the structure at different stages of training, as well as the probability that the topology does not contain ”floating material” or dangling structural members. TopoDiff employs a dataset of 30,000 cases to train the diffusion model, 60,000 cases to train the compliance regressor, and 58,000 cases to train the classifier that predicts the presence of floating material. This method could achieve an eight-fold reduction in average physical performance errors and an eleven-fold reduction in infeasibility compared to TopologyGAN [13]. Despite demonstrating better performance than GANs, the proposed model in [15] uses low-resolution 2D domains (64×64) for its training process, which contains significantly fewer details and thin structural members compared to the high-resolution domains obtained from gradient-based optimal solutions or higher resolution ML predictors. Moreover, using the higher-resolution domains requires more complex neural networks (i.e., more training data) for the external guidance methods to maintain the same level of error in predicting compliance and classifying floating material. On the other hand, the diffusion models are much slower than GANs in predicting the optimal design, taking 21.59s for TopoDiff compared to 0.06s for TopologyGAN. This difference in generating the design is significant for this domain resolution. For instance, by using the output of TopologyGAN as an input to a SIMP solver through several iterations, one can produce a solution of comparable quality in less than 20s for the same domain resolution of (64×64).

Several papers propose alternative approaches to enhance their main loss function and achieve improved results, not only in the field of topology optimization (TO) but also in other domains. For instance, the method proposed in [3] introduced a feature pyramid network (FPN) for TO, which trains using loss functions that combine pixel-wise errors and physical constraints. In their work, the authors proposed a physical loss function that measures the compliance error between the predicted and ground truth structures at each step. While their methods exhibited improvements over their baseline model, the necessity to perform finite element analysis during every loss function calculation made the process time-consuming. This can be one of the reasons why they conducted experiments using only low-resolution structures (64×32) in their experiment. In our own research [4], we adopted a topology-aware loss function in conjunction with binary cross-entropy to explicitly incorporate the topological information of the predicted and ground truth structures into the training process. By introducing the topological loss function, we achieved nearly a twofold improvement in the connectivity of the generated structures. However, our loss function included terms based on persistent homology, which proved to be computationally intensive to compute. Similar modifications to loss functions, incorporating topology-aware terms, have been explored in other domains, as demonstrated in [16]. It is important to note that adding supplementary term(s) to the main loss function necessitates tuning the weight of the new loss, thereby increasing the computational cost of training. Moreover, these additional terms cannot be directly applied to the output of other deep learning models unless those models are trained using the same loss functions.

3. Contributions

In this work, we leverage the properties of Maximal Disjoint Ball Decomposition (MDBD) [7] to convert the structure represented as a binary image into a graph that preserves the topology of the domain. We then demonstrate that by connecting each node in the graph to its nearest neighbors and by assigning a binary label to each edge, we can effectively transform the problem of improving topological connectivity into an edge label classification problem. This approach allows us to use a point transformer [17] network, which is a type of neural network that has shown strong performance in relevant geometric learning tasks, such as learning the topological features of the graph and graph edge labeling. Specifically, in this paper, we utilize this trained model to predict the missing edges in the graph obtained from the structure predicted by *any* deep learning model. By projecting the added edges onto the predicted structure, we can effectively connect disconnected members of the prediction made by the deep-learning model. Figure 1 illustrates the key steps of the proposed framework. Our results indicate that our method can significantly reduce the error in the 0^{th} Betti number [1] without compromising accuracy, and in our experiments the error in the 0^{th} Betti number was reduced by a factor larger than 4.

To the best of our knowledge, this is the first attempt to use graphs preserving the homotopy type of the domain to enhance the connectivity of the structures. Moreover, our method represents a promising step forward in improving the connectivity of the TO predictions made by machine learning algorithms using the image-based representation.

4. Method

The proposed framework is depicted in Figure 1, which shows how to transform the problem of improving the connectivity of the predicted optimal topology into a graph-based edge classification problem. Importantly, the method can augment any other deep learning prediction model for TO. Moreover, it can be used, in principle, with any of the well-known curve skeletons - a concept extensively studied [19, 20, 21, 22]. However, in this work, we employed the graph obtained from the MDBD spherical disjoint decomposition due to its theoretical topological correctness and its representation-agnostic definition in terms of distances. The latter attribute allows us to compute the MDBD decomposition for any valid geometric representation, including cellular decompositions. In the following section, we provide a detailed explanation of each stage of the proposed framework.

4.1. Binary Image to Graph

MDBD is defined as a recursive placement of maximal balls that fit inside a given domain [7]. Observe that connecting the centers of the MDBD balls that have at least 2 tangency points with the domain boundary can yield a graph that is homotopy equivalent with the domain[2], as shown in figure 2. The MDBD

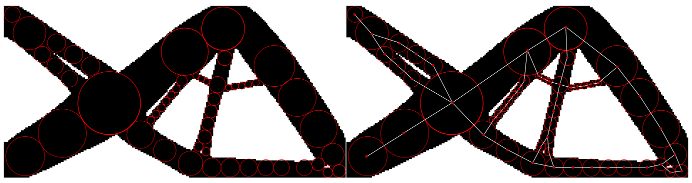


Figure 2: Connecting the centers of the disks that have more than one tangency with the domain boundary results in a graph that has the same homotopy type as the domain.

implementation used in this work prioritized efficiency over accuracy, and, as a result, our MDBD implementation does not compute tangency information for the balls, which is outside of the scope of this manuscript. Instead, to compute the graph nodes, we utilize here a truncated version of MDBD combined with a morphological thinning algorithm to determine the additional nodes required for the graph. Furthermore, edges are established by connecting the nearest neighbor nodes in the augmented thinned structure. This methodology is explained in greater detail in the following section.

4.1.1. Maximal Disjoint Ball Decomposition (MDBD)

The Maximal Disjoint Ball Decomposition (MDBD) [7] is a recently proposed geometric representation scheme that can be thought of as a recursive placement of the largest closed balls within a domain. This decomposition comprises a collection of non-overlapping d-dimensional balls, where d is the dimension of the space for the given domain. MDBD has a multitude of applications in shape similarity, processing, and analysis. This work relies on the facts that MDBD: (1) is representation-agnostic for the given domain; (2) places no restrictions on the complexity of the solid 2D or 3D shape being analyzed; (3) produces a connectivity graph that captures the topology of the domain; and (4) provides an intrinsic and compact parameterization of the optimal solution output by TO algorithms in terms of a collection of ball centers and their radii.

Hence, we used a truncated MDBD to identify potential nodes for the graph representation of the ground truth structures, as illustrated in Figures 3(a) and (b). Given that our MDBD implementation does not calculate the tangency points between the maximal balls and the boundary, as explained earlier, we select an MDBD truncation level that matches the size of the smallest feature of interest in the domain. For the examples investigated in this work, retaining approximately 20% of the total number of balls effectively captures the smallest features present in the structures, as demonstrated in Figure 3(c). We then apply morphological thinning [23, 24] to the structure and use it to generate candidate edges in the graph. The black pixels in Figure 3(d) depict the thinned structure. Next, we compute the minimum distance from the center of the balls to the thinned structure and project the centers having a minimum distance of less than 5 pixels onto the thinned structure, as depicted in Figure 3(e). Note that, in principle, steps 2-4 can be eliminated by computing the ball-boundary tangency information for all the balls and retaining only the balls that have at least 2 points of tangency with the domain boundary, as described above. In this case, a correct graph could be constructed by simply connecting the centers of these balls that are mutually tangent.

This process generates candidate nodes for the graph. However, this set of nodes would result in an incorrect graph if we assume that every node is connected to its immediate neighbors on the thinned structure, as shown in Figure 3(g). Thus,

[1]The k^{th} Betti number captures the number of k-dimensional holes on a topological surface [18].

[2]This leverages the fact that the balls with 2 or more tangency points with the domain boundary are centered, by definition, on the medial axis [7].

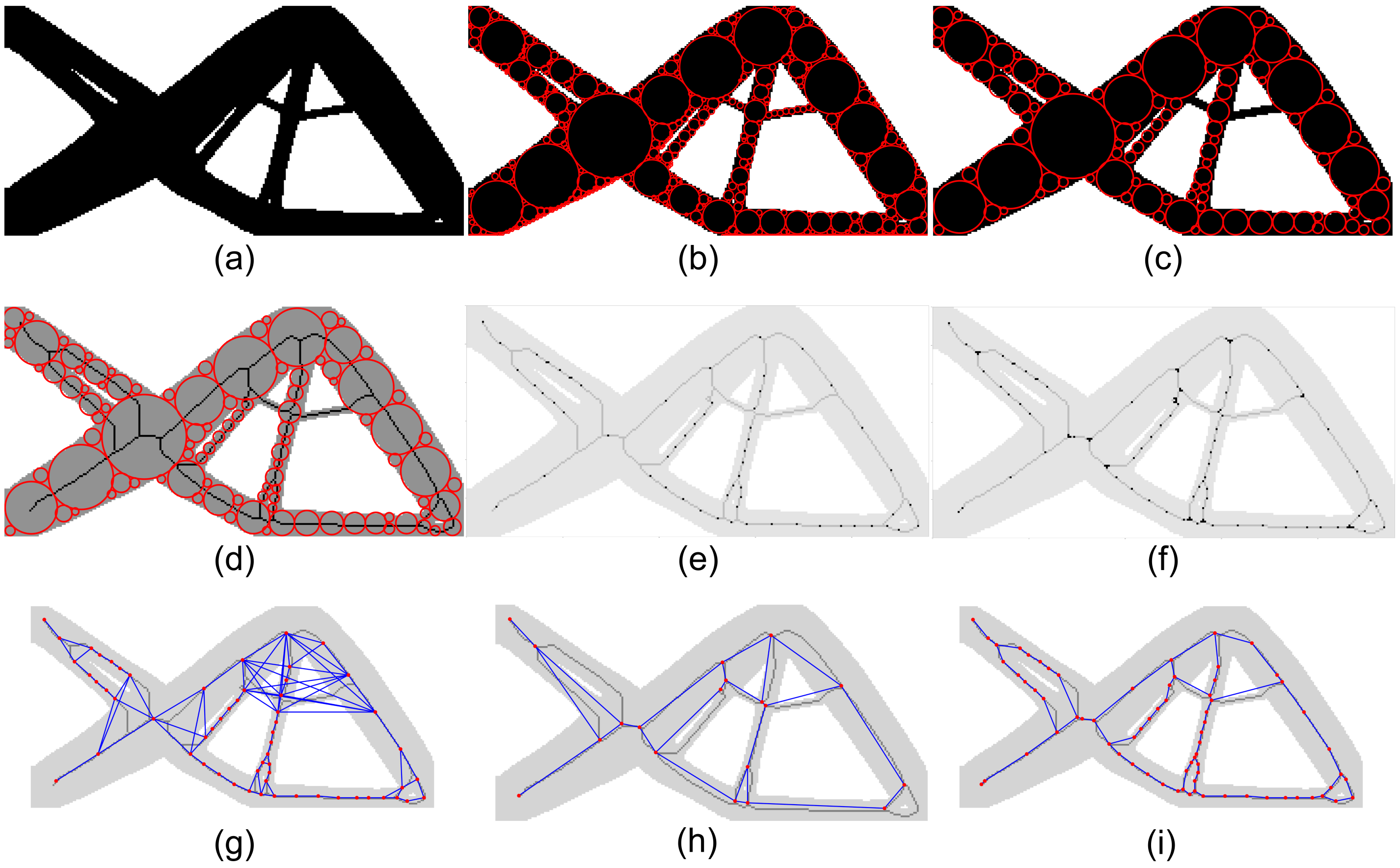


Figure 3: The step-by-step procedure to convert the structure to the graph. (a) shows the optimum structure, (b) is the output of the MDBD for the structure, (c) shows the remaining disks after removing the small ones, (d) shows the thinned structure with the disks, (e) shows the projected disk centers on the thinned structure, (f) shows the projected centers with corners and the endpoints, (g) the graph created by the projected centers alone, (h) the graph created by the corners and endpoints alone, (i) final graph created by combining the projected centers with corners and endpoints.

we have to add additional nodes to the graph such that the connected graph would have the same homotopy characteristics as that of the structure. The computation of these branch points is discussed next.

4.1.2. Branch and End Point Detection

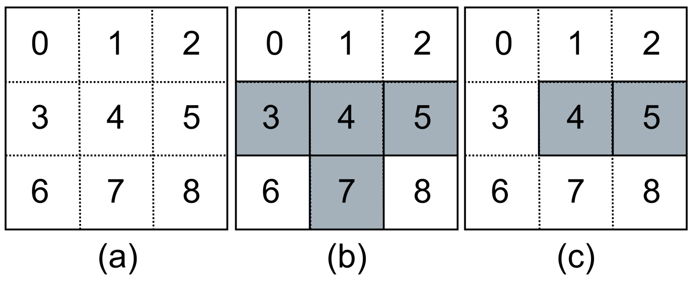


Figure 4: An example of the kernel used to detect branch points and end points of the thinned structure. (a) is the 3×3 kernel, while (b) and (c) are examples of branch and end points, respectively.

The additional graph nodes consist of branch points, i.e., pixels of the thinned structure that are connected to three or four other pixels of the structure, and end points, which are defined as pixels of the thinned structure that are connected to only one other pixel of the structure. These branch and end points were obtained by moving a 3×3 kernel over a thinned structure to identify points where the middle element of the kernel is 1, and the sum of the elements in the kernel is greater than 3 for branch points or equal to 2 for endpoints. Figure 4 illustrates the kernel corresponding to a branch point and an end point. Given the kernel in Figure 4(a), if the middle element is 1, along with three other elements of the kernel as shown in Figure 4(b), then the point on the thinned structure is considered a branch point. Additionally, if the kernel contains only one additional non-zero element besides the middle element, as illustrated in Figure 4(c), then the corresponding point on the thinned structure is an end point. By adding the branch points and end points to the list of graph nodes described above (Figure 3(f)), we can create the graph. It is important to note that using only the branch points and end points, without the graph nodes produced by MDBD, would not yield a graph that is homotopy equivalent with the domain. As shown in Figure 3(h), using branch points and end points alone creates a graph that misplaces the topological features and fails to identify the gaps between two adjacent neighbors. However, by combining these two methods, we can convert our structures into graphs that are homotopy equivalent with the structures, as shown in Figure 3(i). Algorithm 1 presents the pseudocode for our procedure to generate a graph from an image containing the optimal topology.

Observe that MDBD itself captures the topological information of the structure. Moreover, by connecting the centers of the circles that have two or more tangency points with the boundary of the domain, we generate a graph that is homeomorphic with the medial axis, which in turn is homotopy equivalent with the

Algorithm 1 Image to Graph algorithm

```
1:  procedure IMGTOGRAPH(img)
2:      disks ←MDBD(img)
3:      disks ←Remove_Smalldisks(disks)
4:      thinned ←Morphological_Thinning(img)
5:      min_dists ←Distance_to_Thinned(thinned,disks)
6:      nodes ←Project(thinned,disks,min_dists)
7:      nodes ←Corner_Endpoints(thinned)+nodes
8:      edges ←Find_Edges(nodes,thinned)
9:      graph ←Create_Graph(nodes,edges)
10:     return graph
11: end procedure
```

structure, as depicted in Figure 2. This is so because these centers will lie, by definition, on the medial axis of the domain, and the properties of the medial axis are well documented – see, for example, [25, 26], including its homotopy equivalence[3] with the domain. However, in the current implementation of the MDBD algorithm, we do not compute the tangency information between the circles and the boundary. Thus, we chose to employ image processing algorithms using morphological operations to generate the end points and branch points of the skeletons. Our algorithm represents a practical approach for converting binary images into graphs with correct topology. We verified our algorithm on complex 2D domains, including the one illustrated in Figure 5, which is homeomorphic to a 2D disk.

It is worth mentioning that the connectivity graph of an untruncated MDBD for a 2D domain is a 2-complex, as detailed in [7]. However, the graph obtained by connecting the centers of the 2D balls that have 2 or more tangencies with the boundary, whose centers are by definition on the medial axis, is not formed by collections of 2-simplices whose underlying space is part of the given domain. Instead, for such a 2D domain the resulting graph is a curve skeleton that is a simplicial 1-complex, i.e., collection of 1-simplices, as visually depicted in Figures 2 and 5. A good summary of the concepts from algebraic topology related to simplicial complexes can be found in [27].

4.2. Edge Classification with the Point Transformer

The main idea behind this work is to transform the connectivity problem into an edge-label classification problem. In doing so, we identify the missing edges in the graph corresponding to the predicted structure. Subsequently, by projecting the new edges onto the predicted structure, we connect the disconnected members to the main body of the structure. After converting the predicted structure into a graph, we connect each node to its 5 nearest neighbors and request the network to classify each edge. The edges labeled with 0 are removed from the main graph.

To create a dataset suitable for training the model, we connect each node in the ground truth graph to its 5 nearest neighbors and assign a label to each edge. The edges that exist in the ground truth graph are labeled with 1, while the edges that do not exist in the ground truth graph are labeled with 0. Figure 6 illustrates an example of this process. The nodes in the ground truth graph, as shown in figure 6(a), are connected to their 5 nearest neighbors, resulting in the graph depicted in figure 6(b). Finally, by assigning labels to each edge, we obtain the graph shown in 6(c), where the red edges have a label of 1 and the blue edges have a label of 0.

We developed a model based on the point transformer [17] to classify the edges. The network architecture is shown in figure 7, and the model consists of 6 point transformer convolutional layers, where each layer updates the node features according to:

$$x_i^{'} = \sum_{j \in N(i) \cup \{i\}} \alpha_{i,j} (W_3 x_j + \delta_{ij}) \tag{1}$$

where the attention coefficient $\alpha_{i,j}$ and the positional embedding δ_{ij} are computed as:

$$\alpha_{i,j} = \mathrm{softmax}(\gamma_\theta (W_1 x_i - W_2 x_j + \delta_{ij})) \tag{2}$$

and

$$\delta_{ij} = h_\theta (P_i - P_j). \tag{3}$$

Here, $x_i^{'}$ is the updated feature for node i, x_i and x_j are the feature vectors for node i and j, Ws are learnable weights, γ_θ and h_θ are the neural networks, and P_i and P_j are the position of node i and j, respectively.

The inputs to the model consist of the node features, which are the normalized coordinates of each node, the adjacency matrix, and the location of the nodes in the domain required for the message passing. The linear layer processes the node features before they are input to the point transformer layer. After each point transformer layer, there are ReLU activation and dropout layers to mitigate overfitting, except for the final point transformer layer. The output of the final point transformer layer is then passed through a normalization layer, from which the edge features are derived by calculating the average of the corresponding node features. These edge features are then input into a classifier to determine the label of each edge. The dataset we used for our experiment contains 2,250 binary images of size 120×240, which were previously used as training data in [4, 29]. The seen and unseen boundary conditions used for the train and test sets are defined in [4]. Before converting the structures to graphs, the data was augmented by mirroring and rotating by 180 degrees to artificially increase the size of the dataset. We trained our model for 150 epochs on a PC equipped with an Intel Core i7-10750H CPU, 32 GB of RAM, and an NVIDIA GeForce RTX 2070 Super GPU, and implemented early stopping to prevent overfitting and select the model with the best performance. The changes in the loss function at each epoch during training are depicted in Figure 8. It is worth mentioning that the training process lasted approximately 22 hours on a PC equipped with the aforementioned specifications.

The trained point transformer was then used to predict the missing edges in the graph corresponding to the predicted structures. Then the missing edges were projected onto the structure as a line segment with a thickness of 2 px.

4.3. Evaluation Metrics

We use several different evaluation metrics, including the pixel-wise accuracy, defined as the percentage of correctly classified pixels, and the Betti number error, which directly compares the topology (e.g., number of handles) between the prediction and the ground truth. We calculate the 0th Betti number by counting the number of components in the image and 1st Betti number by counting the number of holes in the structure. The Betti number error was obtained according to:

$$i^{th} Betti\ error = \frac{B^i(p) - B^i(g)}{B^i(g)} \tag{4}$$

[3]Note that in [26] it is proven that any bounded open subset in $\mathbb{R}^n$ has the same homotopy type as its medial axis. Consequently, the open set defined by the optimal topology and its medial axis have the same homotopy type.

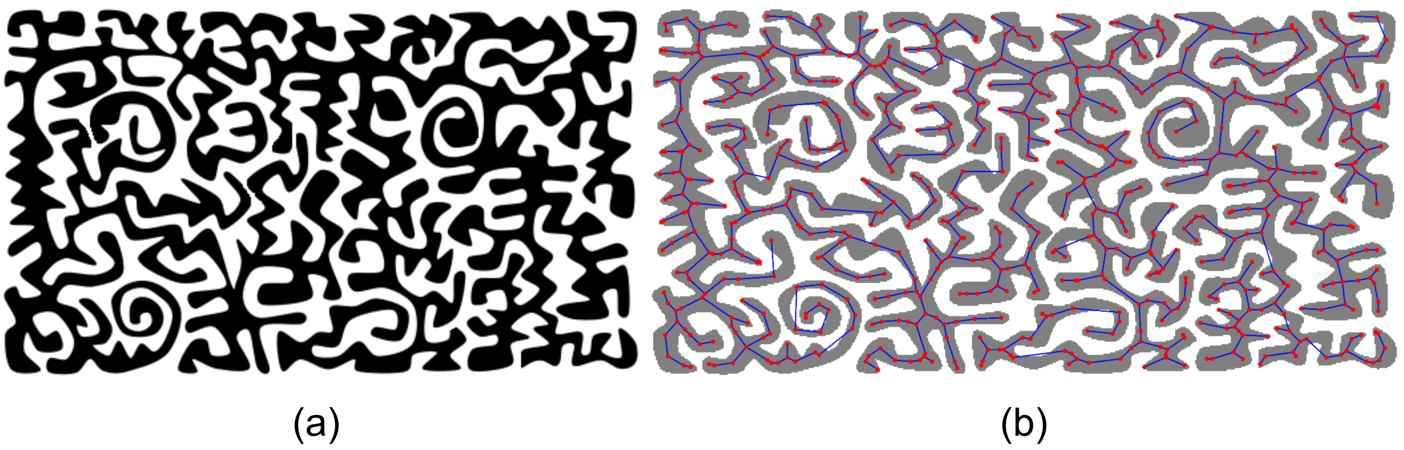


Figure 5: A complex and connected domain without holes, i.e., homeomorphic to a disc, from [28]. (a) shows the original image and (b) displays the graph produced by our algorithm.

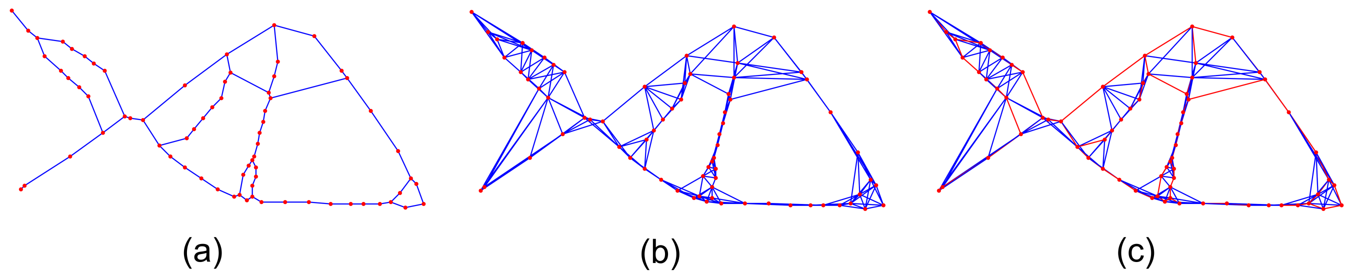


Figure 6: Creating the training dataset. (a) is the ground truth graph, (b) the ground truth graph after connecting each node to its 5 nearest neighbors, (c) shows the labeled graph where the edges with label 1 are shown in red and the label 0 are shown in blue.

where B^i is the i^{th} Betti number, p is short for predicted structure and g stands for ground truth.

Two other metrics that we used to measure the performance of the proposed method are the Compliance Error (CE) and the Volume Fraction Error (VFE). The compliance error can be calculated from:

$$CE = \frac{|C(p) - C(g)|}{C(g)} \tag{5}$$

where $C(p)$ and $C(g)$ are the compliance of the predicted and the ground truth structure, respectively. Similarly, VFE is obtained from:

$$VF\ error = \frac{|VF(p) - VF(g)|}{VF(g)} \tag{6}$$

where $VF(p)$ and $VF(g)$ are the volume fraction of the prediction and the ground truth, respectively.

5. Results and Discussion

5.1. Overall Results

We defined four testing scenarios to evaluate the performance of the proposed approach in improving the connectivity of the predictions. We used the predicted structure by GANTL [4] and the CNN proposed in [29] at two different resolutions: 120×240 and 200×400. We considered both seen and unseen boundary conditions (BC) and different initial domains, namely the L-shape, curved beam, L-shape with a hole, and frame. Figures 9 and 10 show a side-by-side comparison of the original prediction by the TO predictor and the processed structure using our model for all scenarios. The corresponding evaluation data, captured by the performance metrics described in section 4.3, are collected in Table 1.

The performance data shows that our method can improve the connectivity of the predicted structures using two different TO predictors with different resolutions, boundary conditions, and initial domains. Specifically, for the 1st scenario (120×240 domains with seen BCs shown in Figure 9a), our method could reduce the 0^{th} Betti error by a factor of 4. However, the error in the number of holes increased from 15.3% to 18.4%, which was predictable. The reasons may be that closing open loops by connecting them to the main body may add additional holes to the structure, coupled with the fact that our method, as formulated, cannot create new or destroy existing holes in the main body. Our method has also reduced the compliance error from 4.90% to 3.8%, thus improving the performance of the structure. This improvement indicates that connecting the disconnected members to the main body can enhance the structure's performance. Finally, as expected, connecting the disconnected or partially connected components to the main body adds additional material to the domain, leading to an increase in the volume fraction error. However, in our experiments, adding the additional material covered areas that were eliminated in the predicted structure, which we suspect may be the reason why the accuracy remained practically unchanged.

For the 2nd scenario (L-shape, curved beam, L-shape with hole, and frame) with a resolution of 120×240 shown in Figures 9b, 9c, and 9d), the proposed method reduced the 0^{th} Betti error of the predicted structure by [29] from 73.93% to 28.4%. The structures predicted by the CNN from [29] could not accurately depict the holes, so our method reduced the error in the number of holes by almost 11%. Correcting the topology of these predicted

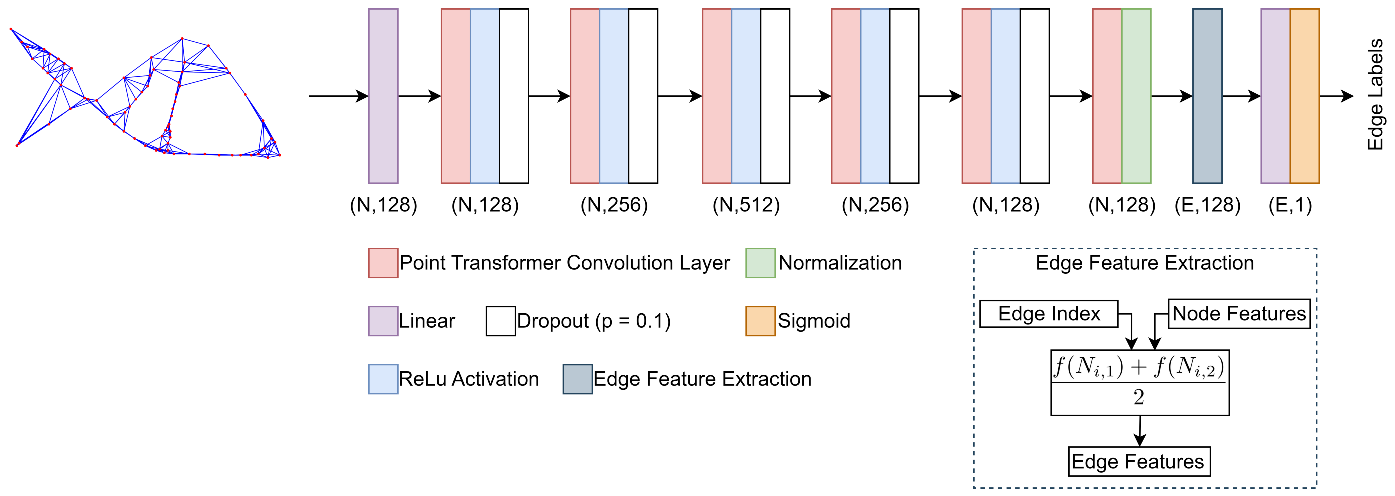
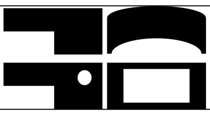


Figure 7: Architecture of the edge labeling network.

Table 1: The qualitative comparison between the predicted structure by the TO predictor and the processed structures using the proposed algorithm for the test datasets in fig. 9. The last row in the table shows the initial domain of the structure in each scenario.

Metrics	1st scenario (Figure 9a)		2nd scenario (Figures 9b-9d)		3rd scenario (Figure 10a)		4th scenario (Figure 10b)	
	Prediction	Processed	Prediction	Processed	Prediction	Processed	Prediction	Processed
0th Betti error	42.0 %	8.5 %	73.93 %	28.4 %	190.1 %	45.7 %	530.0 %	141.3 %
1st Betti error	15.3 %	18.4 %	35.5 %	24.6 %	20.6 %	27.4 %	116.2 %	152.9 %
Compliance Error	4.90 %	3.8 %	9.8 %	6.2 %	4.40 %	4.0 %	7.92 %	7.20 %
Volume Fraction Error	0.03 %	0.30 %	0.53 %	0.71 %	0.16 %	0.43 %	0.53 %	1.08 %
Accuracy	96.40 %	96.31 %	95.06 %	94.9 %	96.10 %	96.50 %	88.44 %	88.31 %
Initial Design Domain								

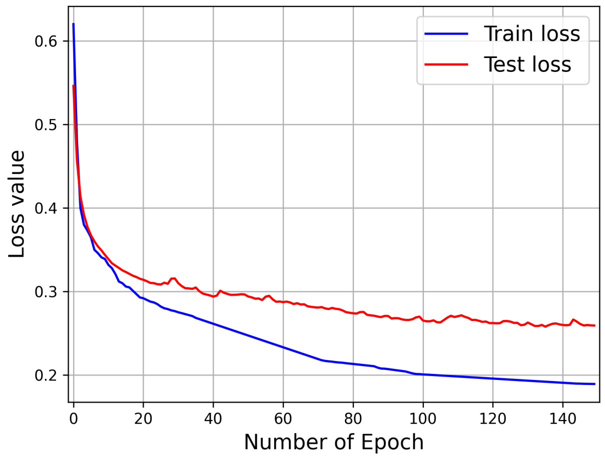


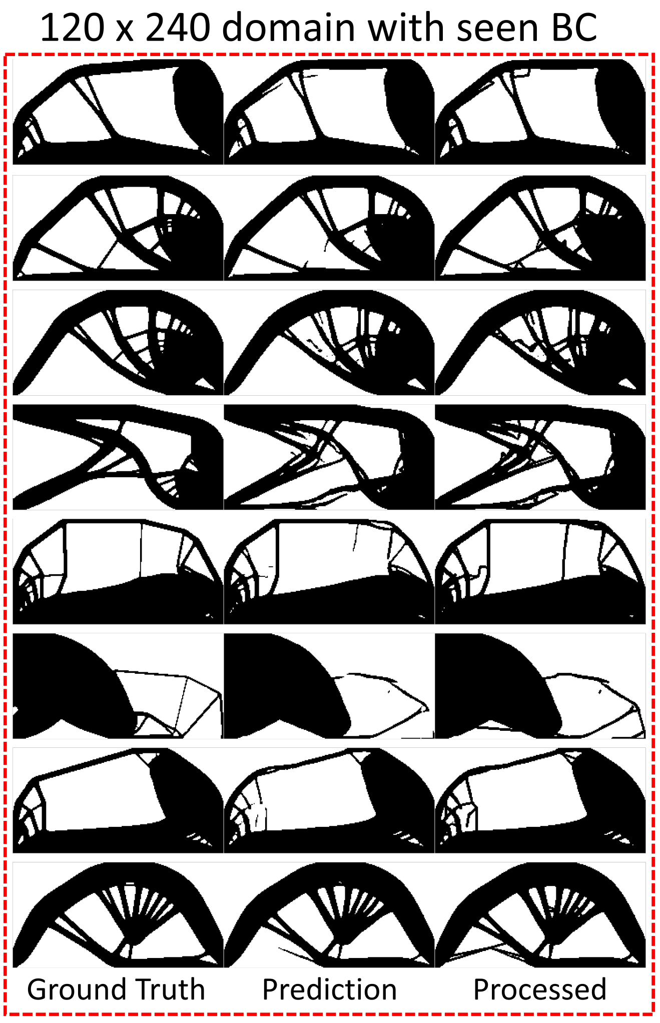
Figure 8: The training and the test loss values in each epoch.

structures reduced the compliance error from 9.8% to 6.2%. The volume fraction error and accuracy followed the same pattern as in the 1st scenario. It is worth noting that the point transformer is trained on the graphs obtained from 120×240 structures optimized from a rectangular domain. The results of this scenario indicate that our method can improve the connectivity of the predicted structures with different initial domains.

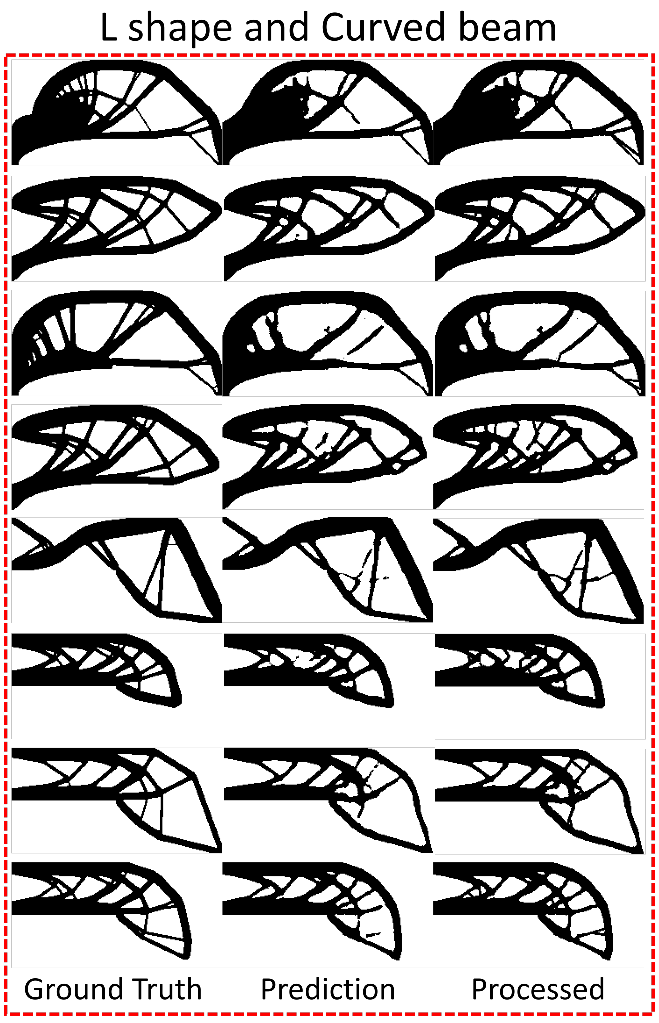
The next two scenarios are designed to demonstrate that our model can perform in domains with different resolutions and unseen boundary conditions, which were not present in the training data used for the point transformer. It is important to note that all training data used to create graphs for the point transformer are sourced from domains with a resolution of 120×240. Figure 10 showcases examples for these two scenarios, and their performance metrics are summarized in Table 1. In the 3rd scenario, the predictions processed by our method exhibit fewer disconnected members (by a factor higher than 4), improved structural performance, and slightly better accuracy. However, these processed structures also display a higher volume fraction error and 1st Betti error, as explained earlier. In the last scenario, the processed structures exhibit significantly fewer disconnected members compared to the original predictions, specifically a 0th Betti error of 141.3% versus 530.0%. Following the same pattern as that displayed in the first three scenarios, the processed structure in the last scenario demonstrates a higher 1st Betti error and volume fraction error, while slightly lower accuracy. The results obtained from these two scenarios demonstrate that the proposed method produces promising outcomes for domains with different resolutions and unseen boundary conditions, even without further training. This demonstrates the generalizability potential of the framework.

5.2. Time Complexity

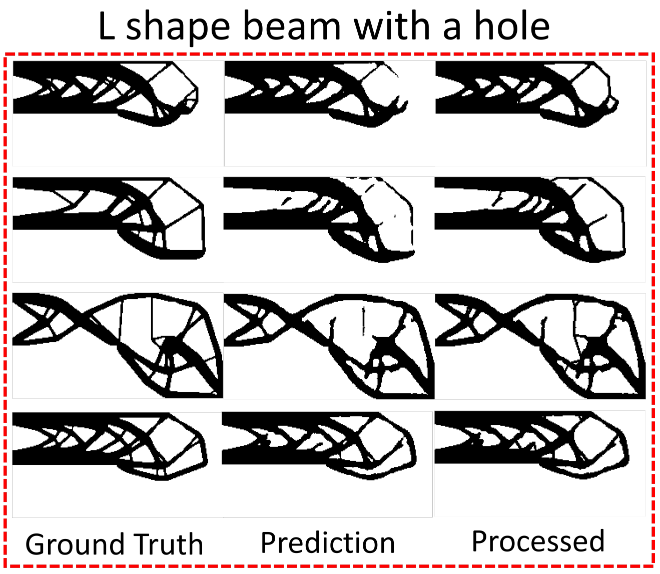
Table 2 depicts the per-case time consumption of each operation when executed on a PC equipped with an Intel Core i7-10750H CPU, 32 GB of RAM, and an NVIDIA GeForce RTX 2070 Super GPU. It is evident from the table that the duration



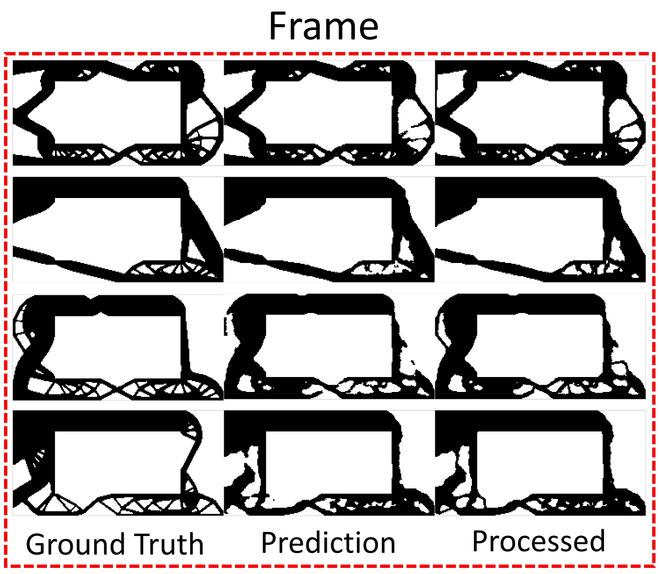
(a)



(b)

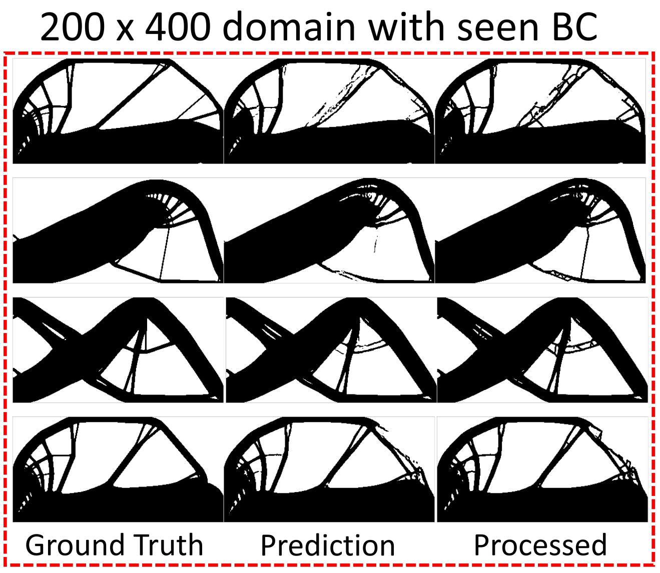


(c)

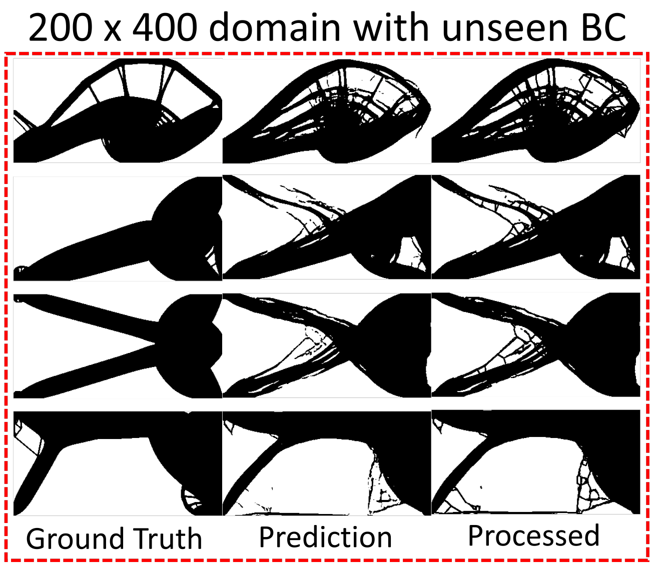


(d)

Figure 9: Comparison between the ground truth (SIMP optimized) 2D structures, predicted structures by a TO predictor, and the improved connectivity produced by our method for the 1^{st} and 2^{nd} scenarios. Fig. (a) shows the results of our method applied on the output of the GANTL [4] for domains with the resolution of 120×240 and seen BCs with respect to the training data used to train the point transformer. Figs. (b-d) shows the results of our approach applied to the output of the CNN proposed in [29] for the L-shape, the curved beam, the L-shape with hole, and the frame. The individual quality metrics for our predictions are presented in Table 1.



(a)



(b)

Figure 10: Comparison between the ground truth (SIMP optimized) 2D structures, predicted structures by GANTL [4], and the improved connectivity produced by our method for the 3^{rd} and 4^{th} scenarios. Figs (a-b) show the output of our method applied on the domains with the resolution of 200×400 with seen and unseen BC presented in [4]. The individual quality metrics for our predictions are presented in Table 1.

Table 2: The time required for each operation per case (second/case).

Domain resolution	MDBD	Img2Graph	Edge Prediction	Edge projection	Total time
120 x 240	0.8	2.3	0.043	0.0024	2.345
200 x 400	1.9	5.1	0.05	0.003	5.153

of the implemented MDBD process and, consequently, the conversion of the structure to a graph, varies significantly with the domain resolution. Once the transformation is complete, the remaining operations only require about 0.05 seconds to generate the processed structure. Also, note that an MDBD implementation that computes the ball tangency with the boundary of the optimal structure could reduce the total time by eliminating the need to compute the thinned structure and the branch and end points.

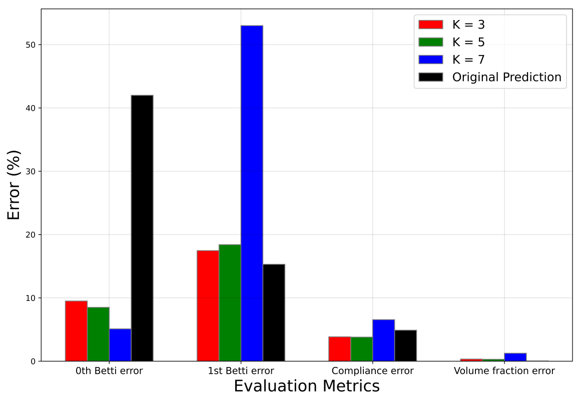
5.3. Ablation Study

We evaluated the impact of varying the number of nearest neighbors (K) and of the edge thickness on the predicted structure. Experiments were conducted using $K = 3$, $K = 5$, and $K = 7$, and edge thicknesses of 1 px, 2 px, and 4 px, respectively, representing the edges missing from the graph (labeled as 1). The results in section 5.1 are based on $K = 5$ and an edge thickness of 2 px. As shown in figure 11a, increasing K from 3 to 7 reduces the 0^{th} Betti error but increases the 1^{st} Betti error and the volume fraction error. This is because the model can predict more connections and create more holes in the domain, leading to a higher 1^{st} Betti error and volume fraction error. The overall performance of structures generated with $K = 3$ and $K = 5$ is similar, with $K = 3$ having a slightly higher 0^{th} Betti error but a lower 1^{st} Betti error. However, $K = 7$ results in poorer structural performance compared to the original prediction, demonstrating the importance of the careful selection of K. The model achieves the best volume fraction results with $K = 5$, indicating that it finds better connections (and therefore adds less material) with this setting.

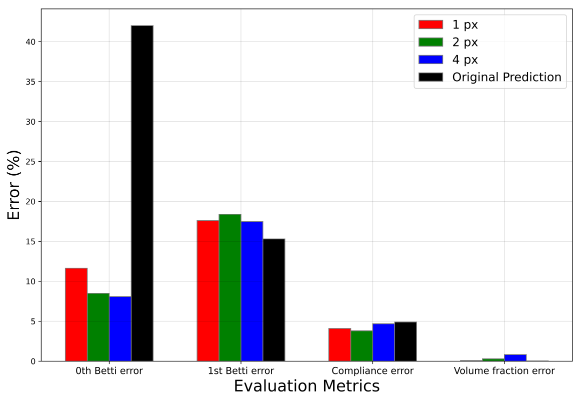
The evaluation metrics for structures processed with varying edge thicknesses are presented in Figure 11b. In this experiment, the thickness of the projected edges was assumed to be constant, although one can use different edge thicknesses, for example, by correlating the edge thickness to its length. As indicated in the figure, using thicker line segments to project missing edges results in a smaller 0^{th} Betti error, but an increase in the volume fraction error due to the added material in the domain. However, no clear correlation was observed between edge thickness and the 1^{st} Betti error or compliance error. Specifically, the 1^{st} Betti error is lower when the edge thickness is 4 px rather than 2 px, but the 1^{st} Betti error of the 2 px edge is higher than that of the 1 px edge. This suggests that thicker edges may fill some of the holes in the structure and result in a reduced 1^{st} Betti error. Additionally, comparing the compliance error for different thicknesses reveals that a higher edge thickness does not necessarily improve structural performance. In our experiments, an edge thickness of 2 px led to a better structure, and we expect that varying the edge thickness would further improve the results.

6. Conclusions

We have proposed a versatile technique to improve the connectivity of machine learning-based topology optimization algorithms. This technique leverages the graph skeleton derived from the Maximal Disjoint Ball Decomposition (MDBD) of the structure and a point transformer to enhance the structural connectivity and, hence, the manufacturability of the structure predicted by any deep learning model. The training data used in the point transformer consists of 2,250 graphs obtained from structures with a resolution of 120×240. We demonstrate that the trained model can connect disconnected or partially connected members in domains with different resolutions, for both seen and unseen boundary conditions, and different initial domains and topologies (e.g., curved beam, L-shaped beam, L-shaped beam with a hole, and frame) predicted by any deep learning model. Specifically,



(a)



(b)

Figure 11: Ablation study. Figure (a) shows the effect of choosing the number of nearest neighbors in the graph on the processed structure. Figure (b) shows the results of projecting the missing edges in the structure with different thicknesses.

we validate the effectiveness of our technique on structures with resolutions of 120×240 and 200×400, subjected to both seen and unseen boundary conditions, and with solutions predicted by a GAN network [4] or a CNN model [29]. Our results demonstrate that the proposed method can improve the connectivity of the structures by a factor of 4 and create design solutions with better structural performance.

One of the key advantages of our proposed method, compared to methods based on modified topological loss functions, is that our method can be applied to structures with different resolutions and boundary conditions without requiring additional training. In contrast, methods based on topological loss functions necessitate separate model training for different resolutions. Moreover, the weight of the topological loss function in such a training process needs to be tuned for every model, resulting in increased computational costs that highly depend on the domain resolution. Additionally, the proposed method can be applied to solutions output by any other deep learning model to improve the connectivity of the predicted solution.

Importantly, graphs need fewer variables than the corresponding images by a factor of at least 10^2 for a $120{\times}240$ image resolution, and this factor increases exponentially with the domain resolution as well as the dimension of the underlying space. Hence, the proposed edge labeling network is not only much easier to train than the accompanying image prediction network, but in principle it also relaxes the performance requirements demanded from the image prediction network, thus allowing a potentially significant reduction in the size of the corresponding training dataset.

One of the limitations evidenced by our experiments is that the method may create holes that do not exist in the ground truth and may create members that would be unable to carry loads. This limitation arises from the current formulation of the method, which fails to eliminate undesired members and instead connects them to the main structure. To overcome this constraint, one potential solution could involve integrating the proposed point transformer into a loss function of the TO predictor during training. This integration would allow us to consider not only the pixels that need to be added but also those that need to be removed. However, it is important to note that incorporating the model into a loss function is likely to increase the computational cost of the training process, as discussed earlier.

Moreover, it is conceivable that one could construct a graph generative model [30] to generate the graph describing the optimal structure while also predicting the thickness of the structure at various locations as node labels. Simultaneously, augmenting the proposed method with the ability to not only add but also remove material could lead to improvements in the manufacturability of the resulting structures. This is because thin, dangling, or disconnected members, as well as small voids, could be eliminated or reconnected to the main structure as needed. Such an extension could leverage the compact parametrization induced by MDBD and its tangency graph.

To the best of our knowledge, this is the first attempt to devise models that enhance the connectivity of the optimal structures generated by machine learning-based TO predictors by using connectivity graphs that have the same homotopy type as that of the domain. Our experiments suggest that utilizing this type of graph produces superior predictions of the optimal structural designs, resulting in structures that have better performance due to a higher load-carrying capacity induced by the improved connectivity. In turn, this work opens up new avenues for future research and applications in the field of TO, as well as in other areas where the topological connectivity of network-like structures is important, such as the segmentation of roads, blood vessels, and neurons from images. Importantly, the fact that the construction of MDBD requires only distance computations in both 2D and 3D implies that our method can use training datasets with models using any valid geometric representation, including cellular decompositions, and naturally generalizes to 3D.

Acknowledgements

The authors acknowledge the support from the National Science Foundation grants CMMI-2232612, CMMI-2312175, and the Office of Naval Research/NIUVT. The authors would like to thank the anonymous reviewers for their constructive feedback. The responsibility for any errors and omissions lies solely with the authors.

References

References

[1] R. V. Woldseth, N. Aage, J. A. Bærentzen, O. Sigmund, On the use of Artificial Neural Networks in Topology Optimisation, Structural and Multidisciplinary Optimization 65 (10) (2022) 294.

[2] K. Nakamura, Y. Suzuki, Deep Learning-based Topological Optimization for Representing a User-Specified Design Area, arXiv preprint arXiv:2004.05461.

[3] J. Luo, Y. Li, W. Zhou, Z. Gong, Z. Zhang, W. Yao, An Improved Data-Driven Topology Optimization Method Using Feature Pyramid Networks with Physical Constraints, CMES-COMPUTER MODELING IN ENGINEERING & SCIENCES 128 (3) (2021) 823–848.

[4] M. M. Behzadi, H. T. Ilieş, GANTL: Toward Practical and Real-Time Topology Optimization With Conditional Generative Adversarial Networks and Transfer Learning, Journal of Mechanical Design 144 (2).

[5] H. Edelsbrunner, J. Harer, et al., Persistent Homology-A survey, Contemporary Mathematics 453 (2008) 257–282.

[6] D. Cohen-Steiner, H. Edelsbrunner, J. Harer, Stability of Persistence Diagrams, in: Proceedings of the twenty-first Annual Symposium on Computational Geometry, 2005, pp. 263–271.

[7] J. Chen, H. T. Ilieş, Maximal Disjoint Ball Decompositions for Shape Modeling and Analysis, Computer-Aided Design 126 (2020) 102850.

[8] D. W. Abueidda, S. Koric, N. A. Sobh, Topology Optimization of 2D Structures with Nonlinearities using Deep Learning, Computers & Structures 237 (2020) 106283.

[9] I. Sosnovik, I. Oseledets, Neural Networks for Topology Optimization, Russian Journal of Numerical Analysis and Mathematical Modelling 34 (4) (2019) 215–223.

[10] D. Wang, C. Xiang, Y. Pan, A. Chen, X. Zhou, Y. Zhang, A Deep Convolutional Neural Network for Topology Optimization with Perceptible Generalization Ability, Engineering Optimization 54 (6) (2022) 973–988.

[11] Y. Yu, T. Hur, J. Jung, I. G. Jang, Deep Learning for Determining a Near-Optimal Topological Design without any Iteration, Structural and Multidisciplinary Optimization 59 (3) (2019) 787–799.

[12] S. Banga, H. Gehani, S. Bhilare, S. Patel, L. Kara, 3D Topology Optimization using Convolutional Neural Networks, arXiv preprint arXiv:1808.07440.

[13] Z. Nie, T. Lin, H. Jiang, L. B. Kara, TopologyGAN: Topology Optimization using Generative Adversarial Networks based on Physical Fields over the Initial Domain, Journal of Mechanical Design 143 (3).

[14] B. Li, C. Huang, X. Li, S. Zheng, J. Hong, Non-iterative Structural Topology Optimization using Deep Learning, Computer-Aided Design 115 (2019) 172–180.

[15] F. Mazé, F. Ahmed, Diffusion Models Beat GANs on Topology Optimization (2022). `doi:10.48550/ARXIV.2208.09591`.
URL `https://arxiv.org/abs/2208.09591`

[16] S. Shit, J. C. Paetzold, A. Sekuboyina, I. Ezhov, A. Unger, A. Zhylka, J. P. Pluim, U. Bauer, B. H. Menze, clDice - A Novel Topology-preserving Loss Function for Tubular Structure Segmentation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 16560–16569.

[17] H. Zhao, L. Jiang, J. Jia, P. H. Torr, V. Koltun, Point Transformer, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2021, pp. 16259–16268.

[18] M. Gardner, Sixth book of Mathematical Games from Scientific American, WH Freeman, 1971.

[19] D. Attali, A. Montanvert, Computing and Simplifying 2D and 3D Continuous Skeletons, Computer Vision and Image Understanding 67 (3) (1997) 261–273.

[20] N. D. Cornea, D. Silver, P. Min, Curve-Skeleton Applications, in: VIS 05. IEEE Visualization, 2005., IEEE, 2005, pp. 95–102.

[21] A. Sobiecki, A. Jalba, A. Telea, Comparison of Curve and Surface Skeletonization Methods for Voxel Shapes, Pattern Recognition Letters 47 (2014) 147–156.

[22] Z. Wu, X. Chen, L. Yu, A. Telea, J. Kosinka, Co-skeletons: Consistent Curve Skeletons for Shape Families, Computers & Graphics 90 (2020) 62–72.

[23] Z. Guo, R. W. Hall, Parallel Thinning with Two-Subiteration Algorithms, Communications of the ACM 32 (3) (1989) 359–373.

[24] L. Lam, S.-W. Lee, C. Y. Suen, Thinning Methodologies-A Comprehensive Survey, IEEE Transactions on Pattern Analysis & Machine Intelligence 14 (09) (1992) 869–885.

[25] D. Attali, J.-D. Boissonnat, H. Edelsbrunner, Stability and computation of medial axes-a state-of-the-art report, Mathematical foundations of scientific visualization, computer graphics, and massive data exploration (2009) 109–125.

[26] A. Lieutier, Any open bounded subset of rn has the same homotopy type as its medial axis, Computer-Aided Design 36 (11) (2004) 1029–1046.

[27] A. Requicha, Mathematical models of rigid solid objects.

[28] F. de Moura Pinto, C. M. D. S. Freitas, L. H. de Figueiredo, Fast medial axis transform for planar domains with curved boundaries, 2010.

[29] M. M. Behzadi, H. T. Ilieş, Real-time Topology Optimization in 3D via Deep Transfer Learning, Computer-Aided Design 135 (2021) 103014.

[30] J. You, R. Ying, X. Ren, W. Hamilton, J. Leskovec, GraphRNN: Generating Realistic Graphs with Deep Auto-Regressive Models, in: International Conference on Machine Learning, PMLR, 2018, pp. 5708–5717.

[31] J. Serra, Introduction to mathematical morphology, Computer vision, graphics, and image processing 35 (3) (1986) 283–305.

Appendix

Morphological Thinning

Thinning is a process through which a digital image is simplified while preserving its topological equivalence, and is commonly used to eliminate selected foreground pixels (pixels with a value of 1) from binary images. The mathematical definition of thinning can be expressed as follows [31]:

$$X \bigcirc T = X/(X \odot T) \tag{7}$$

where T represents the structuring element (kernel), X is the binary image, $X \bigcirc T$ is the resulting thinned image, / denotes logical subtraction, and $\odot$ refers to the hit-or-miss transform. The hit-and-miss operation involves moving the origin of the structuring element to all points in the image and comparing it with the underlying image pixels. If the foreground and background pixels in the structuring element exactly match those in the image, then the pixel underneath the origin of the structuring element is set to the foreground color. Otherwise, it is set to the background color. The thinning process is typically applied repeatedly until the image no longer changes (i.e., convergence is reached), resulting in another binary image [31].

We use here a thinning algorithm that draws inspiration from [23]. The algorithm makes multiple passes over the binary image, removing pixels that match a set of criteria designed to thin connected regions while maintaining connectivity. In the binary image, let p be a pixel with its corresponding neighbors p_i, where i ranges from 1 to 8. The neighbors p_2, p_4, p_6, and p_8 are p's side neighbors, and p_l, p_3, p_5, and p_7 are p's diagonal neighbors. The number of distinct eight-connected components of ones in p's eight-neighborhood is denoted as $C(p)$. If $C(p) = 1$, then p is eight simple when p is a boundary pixel.

For each pixel p in the image, p is deleted if all of the following conditions are met:

1. $C(p) = 1$
2. $2 \leq N(p) \leq 3$
3. Apply one of the following:
   - In the odd iterations: $(p_2 \vee p_3 \vee \bar{p}_5) \vee p_4 = 0$
   - In the even iterations: $(p_6 \vee p_7 \vee \bar{p}_1) \wedge p_8 = 0$

Where $^-$, $\wedge$, and $\vee$ are the logical complement, AND and OR, respectively. $N(p)$ also is defined as:

$$N(p) = \min[N_1(p), N_2(p)] \tag{8}$$

Where

$$N_1(p) = (p_1 \vee p_2) + (p_3 \vee p_4) + (p_5 \vee p_6) + (p_7 \vee p_8) \tag{9}$$

$$N_2(p) = (p_2 \vee p_3) + (p_4 \vee p_5) + (p_6 \vee p_7) + (p_8 \vee p_1) \quad (10)$$

Thinning stops when no further deletions occur. The pseudocode of the algorithm is shown in Algorithm 2. Condition 1 is essential for maintaining local connectivity in the event of pixel deletion and prevents the removal of pixels located along the central medial curves. For a comprehensive proof of this connectivity preservation, we recommend referring to [23].

Algorithm 2 The thinning algorithm

```
function THIN($img$)
    $iter$ ←0
    $n\_pts\_old$ ←inf
    $n\_pts\_new$ ←Sum(img)
    while $n\_pts\_old \neq n\_pts\_new$ do
        $n\_pts\_old$ ←n_pts_new
        for P in img do
            if $C(P) = 1$ and $2 \leq N(P) \leq 3$ then
                if $iter$ is odd and $(p_2 \vee p_3 \vee \overline{p_5}) \vee p_4 = 0$ then
                    $img[P] \leftarrow 0$
                else if $iter$ is even and $(p_6 \vee p_7 \vee \overline{p_1}) \wedge p_8 = 0$ then
                    $img[P] \leftarrow 0$
                end if
            end if
        end for
        $iter$ ←iter+1
        $n\_pts\_new$ ←Sum(img)
    end while
    return $img$
end function
```