Real-Time Topology Optimization in 3D via
Deep Transfer Learning

Mohammad Mahdi Behzadi[a], Horea T. Ilieş[a,]

[a]*Computational Design Laboratory (CDL), University of Connecticut, Storrs, CT 06269, USA*

Abstract

The published literature on topology optimization has exploded over the last two decades to include methods that use shape and topological derivatives or evolutionary algorithms formulated on various geometric representations and parametrizations. One of the key challenges of all these methods is the massive computational cost associated with 3D topology optimization problems.

We introduce a transfer learning method based on a convolutional neural network that (1) can handle high-resolution 3D design domains of various shapes and topologies; (2) supports real-time design space explorations as the domain and boundary conditions change; (3) requires a much smaller set of high-resolution examples for the improvement of learning in a new task compared to traditional deep learning networks; (4) is multiple orders of magnitude more efficient than the established gradient-based methods, such as SIMP. We provide numerous 2D and 3D examples to showcase the effectiveness and accuracy of our proposed approach, including for design domains that are *unseen* to our source network, as well as the generalization capabilities of the transfer learning-based approach. Our experiments achieved an average binary accuracy around 95% at real-time prediction rates. These properties, in turn, suggest that the proposed transfer-learning method may serve as the first practical underlying framework for real-time 3D design exploration based on topology optimization.

Keywords: Topology optimization, Transfer learning, Real-time predictions, Design space explorations, Deep learning.

1. Introduction

Topology optimization (TO) seeks the optimum material distribution given an objective function, a design domain, and a set of boundary conditions. Each optimization is an iterative process, often requiring hundreds of iterations. From a computational standpoint, running a 3D TO requires the investment of most of the available computational effort to solve the analysis equations for every iteration. The associated computational cost, which is significant for any problem that approaches real conditions, typically depends on the dimensionality and resolution of the domain, on the number of design variables being used, as well as on the numerical solution procedure itself. A good and relatively recent review of the popular numerical solution procedure approaches can be found in [1] and a detailed discussion of the computational cost involved can be found in [2] and [3]. Very recently, various machine learning algorithms have been proposed to tackle the massive computational cost of established gradient-based TO methods, including convolutional neural network (CNN) [4], generative adversarial network [5], and conditional generative adversarial network [6]. While these approaches are promising, the training needed by these methods requires large datasets that are computationally expensive to generate, as discussed below. Furthermore, these methods have not been shown to be capable of handling multiple boundary conditions, domains with different topologies and geometries, or high-resolution 3D domains.

In this paper we introduce a transfer learning method based on a convolutional neural network that (1) can handle high-resolution 3D design domains of various shapes and topologies; (2) supports real-time design space explorations as the domain and boundary conditions change; (3) requires a much smaller set of high-resolution examples for the improvement of learning in a new task compared to traditional deep learning networks; and (4) is multiple orders of magnitude more efficient than the established gradient-based methods.

2. Background

The published literature on topology optimization has exploded over the last two decades to include methods that use shape and topological derivatives or evolutionary algorithms formulated on various geometric representations and parametrizations. Different approaches have been developed to search for the optimal shape, including approaches based on material density functions [7]; level sets [8, 9]; topological derivatives [10, 11]; phase fields [12]; and several other variations [1]. One popular and established approach based on material density, known as Solid Isotropic Material with Penalization (SIMP) [7], finds the optimal topology by changing the material density of elements in the unit interval [0, 1]. The methods based on topological derivatives aim to predict the sensitivity of the problem to the addition of an infinitesimal hole at prescribed locations inside the design domain, and this information is used to generate new holes [13].

One of the key challenges of all these methods is the immense computational cost associated with 3D topology optimization problems, so it is not surprising that there is an extensive body of work dedicated to improving the computational cost of TO. Parallel computing has been employed in [14] for large scale topology optimization by dividing the domain into sub-domains that are independently solved on separate processors. The results presented in [14] present an optimization of a domain with 24x80x128 elements in 234 minutes using 16 processors. The work described in [15] relies on a high-performance multigrid GPU solver to find the optimum solution of models with millions

of elements. Their method solved on the GPU a 200x100x100 cantilever beam with 50 iterations, and volume fraction of 0.8 in 2.4 minutes. Observe that standard SIMP implementations [16, 17] require more than 120 iterations for convergence, even for simple cantilever beams. Design space adjustment and refinement is employed in [18] to speed up the computations for large-scale domains. This work performs TO for an MMB beam with 10800 elements in 5.5 minutes, which is roughly a third of the time required by a SIMP implementation to produce a solution to the same problem. Other papers, such as [19], have employed the reduction of the the number of design variables to decrease the computational cost, and showed the capability to produce an optimal solution of a cantilever beam in 10.12 minutes compared to 20.2 minutes required by a SIMP-based solver.

As mentioned above, machine learning methods have been recently used to tackle the computational efficiency of the topology optimization problem. For example, the work discussed in [20] uses machine learning to tune the numerical parameters that control the convergence of established topology optimization algorithms to avoid the manual tuning, which is computationally costly. CNNs have been applied to estimate the optimal topologies for 3D low resolution beams [4], or 2D domains [21, 22, 23, 24]. Moreover, CNNs have been very recently used to predict the optimum 2D structure for simple low resolution 2D domains [25]. In this paper, the authors claim a generalization ability of their network primarily because they explicitly input into the network the initial displacement and strain fields as well as the volume fraction, rather then the explicit boundary conditions. However, they use 80,000 training samples and 10,000 test samples, which is simply prohibitive for any problem that is of reasonable complexity. At the same time, different versions of GANs have been used to optimize the topology of 3D cantilever beams [26], 2D low resolution cantilever beams [27, 28], and 2D high resolution beams [5, 6]. Variational auto-encoders (VAE) and supported vector regression (SVR) have been applied to 2D low resolution domains with different boundary conditions [29], and cantilever beam [30], respectively. All these methods can reduce the TO computational time for given domain resolution, set of boundary conditions, and initial domain as long as large amounts of data are available. However, applying these methods to tasks that the algorithms have not been trained on requires new large sets of training data. This is not only impractical for any topology optimization problem of reasonable complexity, but also makes these methods unsuitable for design space explorations.

A key assumption of most machine learning algorithms is that the training data and task data are in the same feature space and have the same distribution. This is a reasonable assumption as long as the training data is relatively painless to generate and abundant. However, this is definitely not the case in many engineering application, including topology optimization. Transfer learning has emerged as one promising learning algorithm that has the potential to greatly improve the learning performance by limiting the amount of training that needs to be performed to adapt the algorithms to new scenarios. It aims to imitate one of the distinctive features of human intelligence, that is, to effectively transfer previously learned knowledge to new domains [31, 32]. Transfer learning has been successfully used in medical image processing [33], and brain-computer interface calibration [34].

2.1. Contributions and Outline

In this paper we propose a transfer learning approach that we developed specifically for topology optimization. We show that
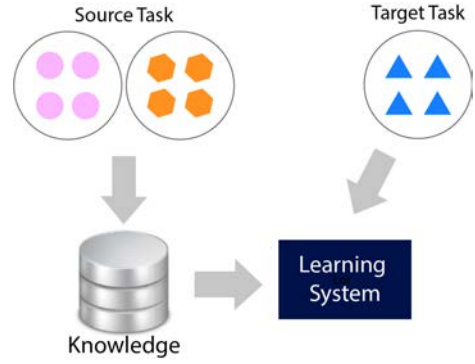

Figure 1: A schematic diagram of transfer learning after [35]

the method produces highly accurate predictions of the optimal 3D topologies at real-time rates for non-trivial 2D and 3D high resolution TO problems. Furthermore, we show that the proposed method serves as the first practical underlying framework for real-time 3D design explorations based on topology optimization, and that fine tuning/retraining the proposed learning algorithm for new tasks can be done with a much smaller high-resolution dataset than the traditional deep learning networks. To the best of our knowledge, this paper documents the first promising attempt to use transfer learning for topology optimization.

The rest of the paper is organized as follows. Section 3 presents the formulation of the proposed method, as well as a detailed description of the implementation and data generation. This section also includes a discussion of the source and target networks as well as of the two metrics that we used to evaluate our network. Section 4 illustrates the generality and flexibility of our approach by providing a variety of 2D and 3D examples with different resolutions, boundary conditions and design spaces, including design domains that are *unseeen* to the source network. These examples show that the proposed method supports real-time design space explorations as the domain and boundary conditions change and is multiple orders of magnitude more efficient than the established methods for both 2D and 3D design scenarios. Finally, Section 5 summarizes the key advantages and limitations of the proposed method and of its potential applications.

3. Problem Formulation

We describe the proposed method in the context of the well known minimum compliance topology optimization problem, which is considered to be a "global response" of a structure. However, our method can be equally well applied to TO problems that consider more local responses in their objective functions, such as stress-based TO [36].

The general topology optimization method aims to find the spatial distribution of material $\rho(\mathbf{x})$ that minimizes an objective function $f(\Omega, \rho)$, subject to various constraints $g_i \leq 0$, where the state field $\mathbf{u}$ satisfies given state equation. It is common to assume that the objective function is expressed as an integral over a local function such as the strain energy density [1], and to solve the problem using the finite element method. In the SIMP method, each element is associated with a density variable $\rho_e \in [0, 1]$, where 0 corresponds to an empty element, and 1 to an element completely filled with material. This optimization problem can be formulated as
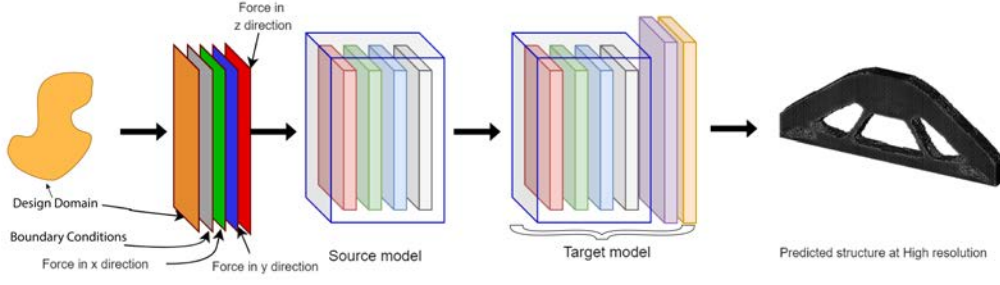
Figure 2: A diagram of the proposed method for 3D topology optimization.

$$min \quad : \quad f(\Omega, \rho) = \mathbf{U^t K U} = \sum E_e(\rho_e)\mathbf{u}_e^t \mathbf{k}_e^0 \mathbf{u}_e \tag{1}$$

$$s.t \quad : \quad \mathbf{KU} = \mathbf{F} \tag{2}$$

$$\sum \rho_e v_e \leq V_{max} \tag{3}$$

$$0 \leq \rho \leq 1 \tag{4}$$

where Ω is the domain; ρ is the design variable vector of densities; $\mathbf{U}$ and $\mathbf{F}$ are the global displacement and force vectors, respectively; $\mathbf{K}$ is the global stiffness matrix; $\mathbf{u}_e$ is the element displacement vector; $\mathbf{k}_e^0$ is the element stiffness matrix for an element with unit Young's modulus; v_e and ρ_e are the element volume and density of element e, respectively; and V_{max} is the volume upper bound. $E_e(\rho_e)$ is the element's Young's modulus determined by the element density ρ_e:

$$E_e(\rho_e) = E_{min} + \rho_e^p(E_0 - E_{min}), \qquad \rho_e \in [0, 1] \tag{5}$$

In equation (5), E_0 represents the stiffness of the material, E_{min} is a very small number, and p is a penalization factor [37, 1].

3.1. Topology Optimization with Transfer Learning

One of the distinctive features of human intelligence is the ability to effectively transfer previously learned knowledge to new application domains. We all use this capability every single day even without realizing it [38]. In contrast, most machine learning algorithms are trained on and function only on well defined tasks. Transfer learning aims to improve this limitation by transferring the knowledge from a source task to improve the performance in target task that is different from but related to the source task. Every transfer learning algorithm uses specific learning algorithms to learn the tasks at hand, which is why transfer learning is often described as extensions of those learning algorithms.

In a broad sense, transfer learning deals with 3 different questions: (1) *what* information should be transferred; (2) *how* to transfer the information; and (3) *when* to transfer it. As an example, for the topology optimization task discussed in this paper, we transfer the weights and biases of all layers of the source network except for the last layer. To address the second question, we implemented a mechanism to transfer this knowledge from the source network to the target network with minimal retraining. The third question deals with establishing the cases when the knowledge transfer should be performed, i.e., the validity of the transfer learning method [35], and this is often addressed by measuring the performance to new tasks. This is also the approach that we take in this paper.

Because optimizing a high resolution domain using state of the art gradient based TO methods, including the popular SIMP

method, is always computationally very expensive, generating sufficient training data for practical design scenarios becomes a crucial bottleneck. This is why we developed a deep transfer learning method that uses a fully convolutional neural network, which allows us to transfer the knowledge obtained from training the algorithm on low resolution models to be usable on high resolution cases with different design domains and boundary conditions that the source network has not been trained on.

Figure 1 illustrates how transfer learning works. First, a source model is built and trained with a large amount of low resolution data, which is relatively inexpensive from a computational point of view. This is followed by transferring what the source model learned to a target model operating on different but related tasks. This process allows the use of a much smaller amount of data to retrain/fine tune the target model to improve learning in the target task. In our implementation, the source and target models are CNN based decoder-encoders with the source model trained on large amount of low resolution (and thus relatively inexpensive) data, and the target model retrained for the new task with relatively small high-resolution datasets.

The overall architecture is illustrated in Fig 2. The input is the design space and the boundary conditions. For the examples presented in this paper we only considered externally applied forces, but adding externally applied torques is straightforward. During the final step, we build the target network described in section 3.2 by augmenting the source network with additional layers, and train it with the high resolution data.

3.2. Network Architecture and Network Training

Convolutional Neural Networks (CNNs) have been successfully used in object classification and segmentation tasks. More recently, a number of papers have shown CNNs to be trainable and effective on large datasets for solving inverse problems in imaging, such as denoising, deconvolution, super-resolution, and medical image reconstruction. These inverse problems practically involve the determination of an image from noisy measurements. Since the datasets output by SIMP can loosely be considered as a special case of these inverse problems, the architectures of our source and target networks are inspired by some of the work described in reviewed in [39] and elsewhere.

Our source network is a two-dimensional encoder-decoder based convolutional neural network (CNN) illustrated in Fig 3. The encoder part includes eight convolutional layers with the rectified linear unit (ReLU) activation function, and three max-pooling layers. The decoder part of our source network includes three transposed convolutional layers and seven convolutional layers using the ReLU activation function. We trained the network using the ADAM optimizer [40], which is a standard gradient-based optimization algorithm. The ADAM optimizer
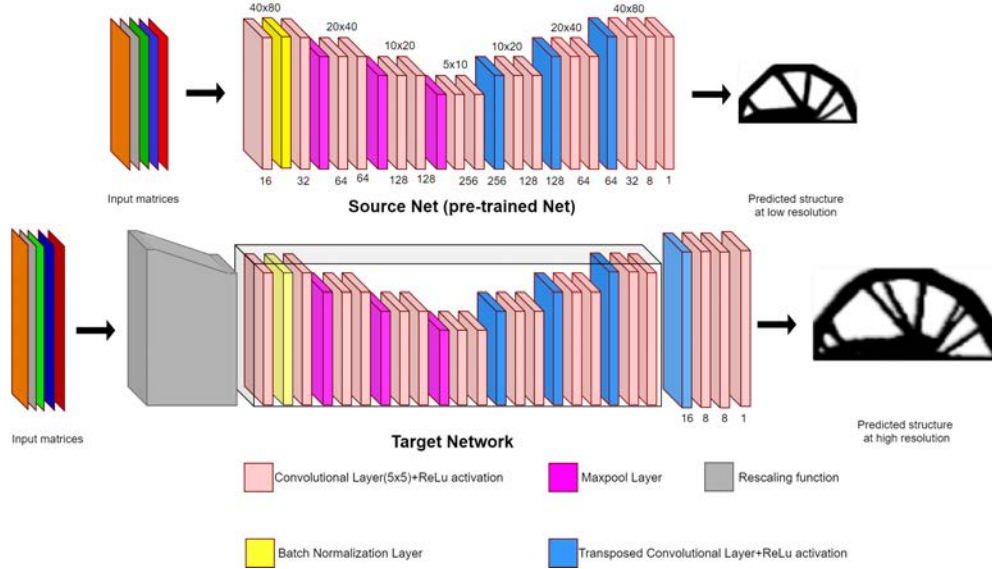
Figure 3: The architecture of the source and target networks. Numbers below the boxes denote the number of filters used.

finds the optimal weights and biases of the network that minimize the loss between the predicted structures and the simulated structures according to the mean squared error (MSE).

We built our target network on top of the source network as illustrated in Figure 3. To the end of the source network we added one transposed convolutional layer to increase the output dimension of the pre-trained network to the higher resolution, as well as three trainable convolutional layers. To the front of the network, we added a rescaling function to downsample the higher resolution input to the target network to the lower resolution required by the source network.

Prior to training the target network, we remove the last layer of the pre-trained network. This modification is based on measuring the accuracy of the predictions as described in the next section. This modified target network is trained as described in section 4 with ADAM as the optimizer and MSE as the loss function.

Importantly, we trained the source network only once for a given dimension of the space (2D or 3D). This trained source network is integrated within the target network, which is re-trained/fine tuned with smaller datasets for new design domains and/or boundary conditions.

3.3. Data Generation

We used freely available SIMP-based topology optimization finite element codes [16, 17] to generate our training and test cases, and we modified the codes to automate the training/test case generation for different domains and boundary conditions. The input to the codes are the following design variables and physical quantities: voxelized domain geometry, volume fraction, filter radius (see [37, 41] for details), loading boundary conditions (number, magnitudes, directions, spatial locations), and displacement boundary conditions. We prescribed to the SIMP codes a volume fraction equal to 0.5 and a 1.5 filter radius.

We sampled the magnitudes of the force components using uniform random sampling within the range $[-100, 100]$N. The spatial location on the domain boundary where the external load was applied was chosen based on a uniform random sampling within prescribed ranges along the coordinate axes. For example, the force components P_x and P_y for a 2D beam domain are

applied within the range $[\frac{b_x}{2}, b_x]$ and $[0, b_y]$, respectively, where b_x and b_y are the beam dimensions in the x and y directions. We also used a discrete random sampling to select one of the defined displacement boundary constraints illustrated in Fig 4.

With this random sampling of the boundary conditions we generated two datasets: a low-resolution dataset for our source network, and a smaller, high resolution dataset for the target model. Those two datasets with distinct elements, were then split into training and testing datasets, where the latter was at least 20% of the former. Details about the sizes of these datasets are provided in Sections 4.1 and 4.2. The individual sizes of the testing datasets for our examples are shown in Tables 1 and 4.

We use matrices to store the data fed into the five channels to our network architecture. For example, we used five channels for the 2D cases, as follows:

1. First channel: Initial density value for each voxel.
2. Second channel: Constraints in the x direction initialized to zero, then the elements corresponding to the constrained elements are set to 1.
3. Third channel: Constraints in the y direction initialized to zero, then the elements corresponding to the constrained elements are set to 1.
4. Fourth channel: Force value in x direction at each voxel,
5. Fifth channel: Force value in y direction at each voxel.

3.4. Evaluating the Network

We employed the following two criteria for the evaluation of our method against state of the art methods:

1. *Mean Squared Error* (MSE), which is defined as:

$$MSE = \frac{\sum_{j=1}^{M} \sum_{i=1}^{N} (y_{pred}^{ij} - y_{true}^{ij})^2}{N \cdot M}$$

where N and M are the number of rows and columns, y_{pred}^{ij} and y_{true}^{ij} are the predicted and reference values of element located in the ith row and j^{th} column, respectively.
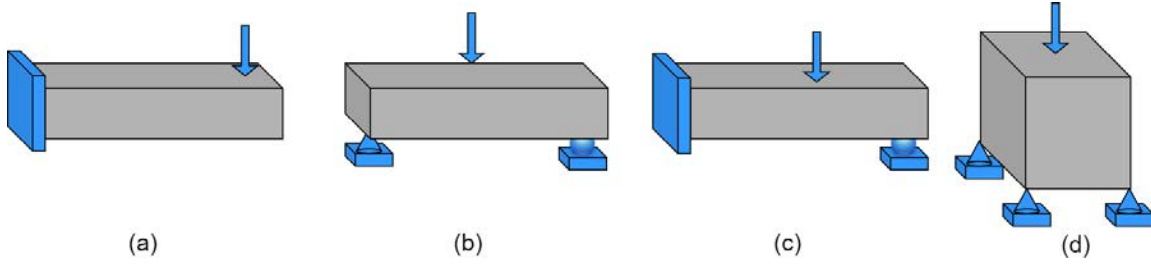
Figure 4: The randomly selected displacement boundary constraint cases: (a) Cantilever Beam, (b) Simply Supported Beam, (c) Constrained Cantilever, and (d) The boundary conditions for the cases shown in Figures 10 and 11.

2. *Binary Accuracy* (BA), a widely used measure that compares the binarized value of predicted elements with the actual value [42]:

$$BA = \frac{TP + TN}{N}$$

where TP (True Positive) is the number of elements correctly predicted as 1, TN (True Negative) is the number of elements correctly predicted as 0, and N is again the total number of elements. Prior to calculating the binary accuracy, we rounded the element values to the nearest integer, that is, either to 0 or 1.

As the average of the squared deviations, the Mean Squared Error is a second sample moment about the origin of the error, and is the minimum variance unbiased estimator [43]. For a given predictor, the closer MSE is to zero, the better the predictor performance. Furthermore, accuracy reflects the overall ratio of correct predictions. A common version of this measure, namely the binary accuracy, uses binarized density values in the ground-truth and predicted domains, and shows how accurately the network predicts the existence of material in each voxel. The values of the binary accuracy lie within the unit interval, and the closer the binary accuracy is to 1, the better the prediction. In this work, we use the binary accuracy to measure how accurate our network is in predicting the existence of material, and we use MSE to estimate[1] how close are the density predictions to the ground-truth values. Our experiments achieved an average binary accuracy and MSE around 95% and 3%, respectively. All predictions were performed on a Dell Intel Xeon Processor E5-2650 v3 CPU with 64 GB RAM and Nvidia Quadro K2200 4GB GPU. All training and test cases were generated on the UConn HPC facility running Red Hat RHEL7 operating system.

For the optimal topologies predicted by our algorithms, we are also providing the corresponding compliance error relative to the compliance of the ground truth optimal structures. Since our predicted structures are directly output by our algorithms without any post-processing/beautification steps, the resulting compliance errors shown for the examples are particularly promising.

---

[1]These two evaluation metrics do not always agree. For example, assume that the predicted and actual values are 0.49 and 0.51 respectively. MSE tells us that the error is 0.04% which means the prediction is very accurate, But binary accuracy treats the predicted value as 0 and the actual value as 1, and implies a 0 binary accuracy. A more detailed discussion can be found in standard texts on probability.

4. Results

4.1. 2D Structures

Comparison with Ground Truth (SIMP)

We used a freely available Matlab Code [16] to generate the ground truth results. For the 2D examples, we trained the source network with 8,000 low resolution cases (40 x 80), and fine-tuned the target model (various resolutions, as shown in Table 1) with 1500 high resolution cases.

Importantly, the source network is trained only once and the learned information is being reused. The high resolution cases used for fine tuning the target network have the same resolution as the test cases used in our examples and shown in Table 1. To generate the training data for all 2D examples, boundary conditions have been randomly selected from one of the 3 cases shown in Figure 4(a-c), and the location, orientation and magnitude of the external force have also been randomized.

We first compared our approach with the performance of the SIMP solver [16] for 2D structures in terms of the output accuracy as described in section 3.4, and the average time required to obtain the optimal topology. For this evaluation, we used 2,000 low resolution test cases for the source network and a smaller set of high resolution cases with randomly generated boundary conditions, as summarized in Table 1.

Fig. 5 shows a side-by-side comparison between the optimal topology output by our method relative to the corresponding ground-truth result (i.e., SIMP-based optimal structures) for different domains and boundary conditions. Specifically, Figure 5a shows the optimal 2D structure output by our source network alone on a low resolution design domain as well as the corresponding MSE, binary accuracy and compliance error. Moreover, Figures 5(b-f) show the optimal 2D structures output by our target network for different domains and boundary conditions, and for design domains with higher resolutions. The predicted and ground truth solutions are not only visually similar, but the corresponding average MSE and average binary accuracy are around 3.6% and 95%, respectively, and the resulting compliance error is around 8.7%.

Figures 6 and 7 show a similar comparison for domains that have different geometry, topology and boundary conditions that the source network has *not* been trained on. The examples shown in Figures 6 and 7 also use different volume fractions (i.e.,0.3 and 0.4, respectively). Importantly, these Figures illustrate not only the performance of our predictions, but also the fact that the same source network can be used to build different target models for substantially different design spaces and boundary conditions.

The quality metrics for the individual examples have been compiled in Table 1. Furthermore, we show in Table 2 the time required by our transfer learning-based method to make predictions for several 2D optimal topologies: the average prediction
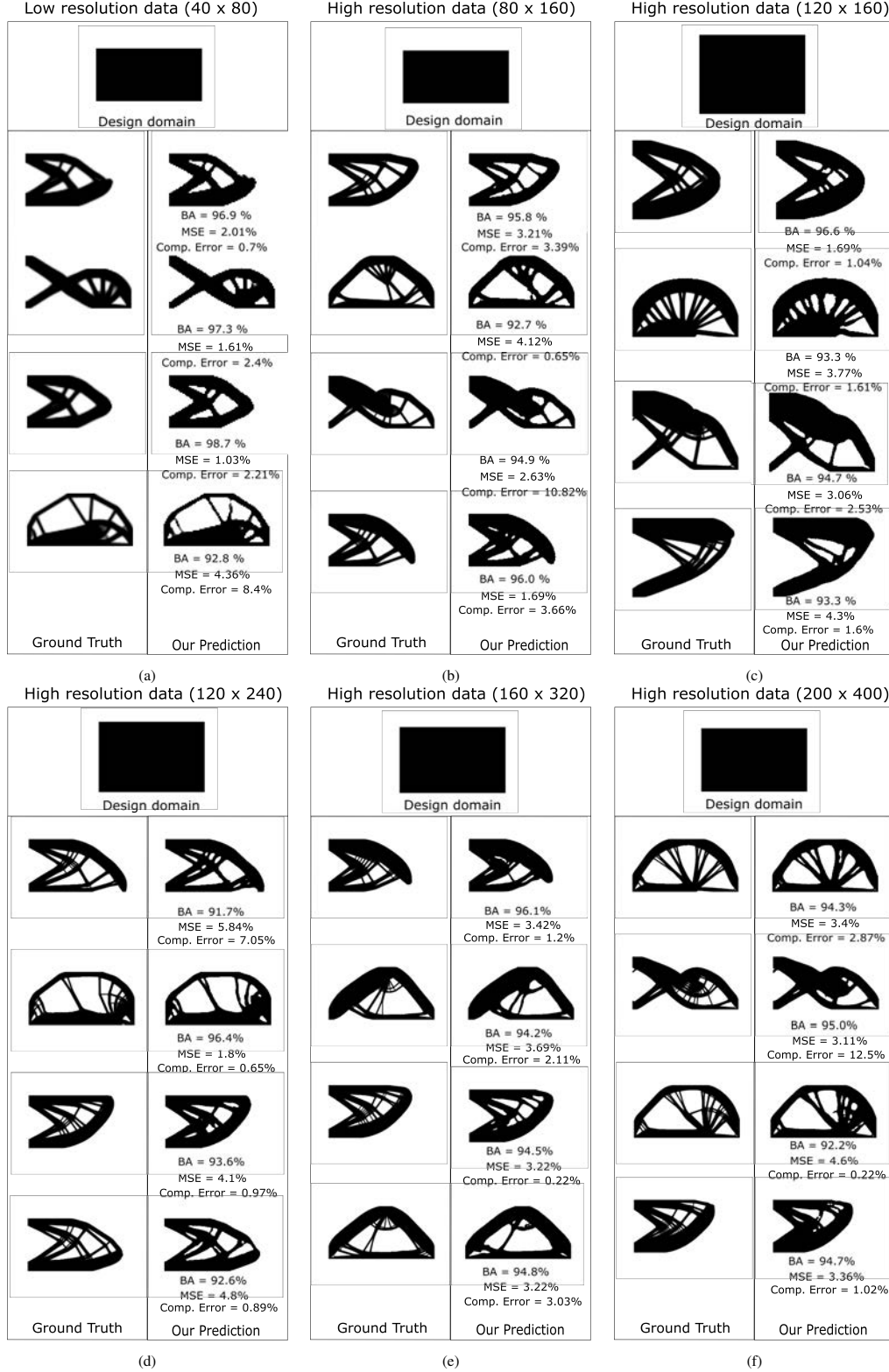
Figure 5: Comparison between the ground truth (SIMP optimized) 2D structures and our predictions of the optimal structures. Fig. (a) shows the prediction of our source network alone. Figs (b-f) show the prediction of the optimal structures output by the fine tuned target model. The individual quality metrics for our predictions are presented in Table 1, and the prediction time is shown in Table 2
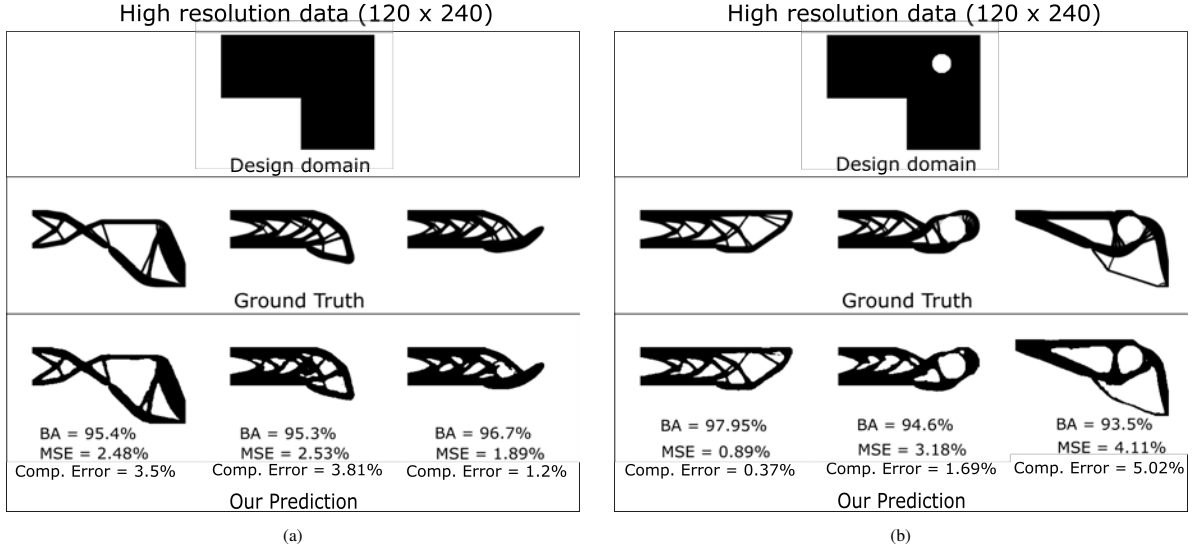
Figure 6: Predicted optimal structures versus ground truth (SIMP optimized) for high resolution domains: (a) an L-shaped domain of genus 0, and (b) L-shaped domain with a hole (genus 1). The individual quality metrics for our predictions are presented in Table 1, and the prediction time is shown in Table 2. Observe that the source network for the 2D examples has only been trained on the domains shown in Figure 5, so the domains shown in this Figure are *unseen* to the source network.
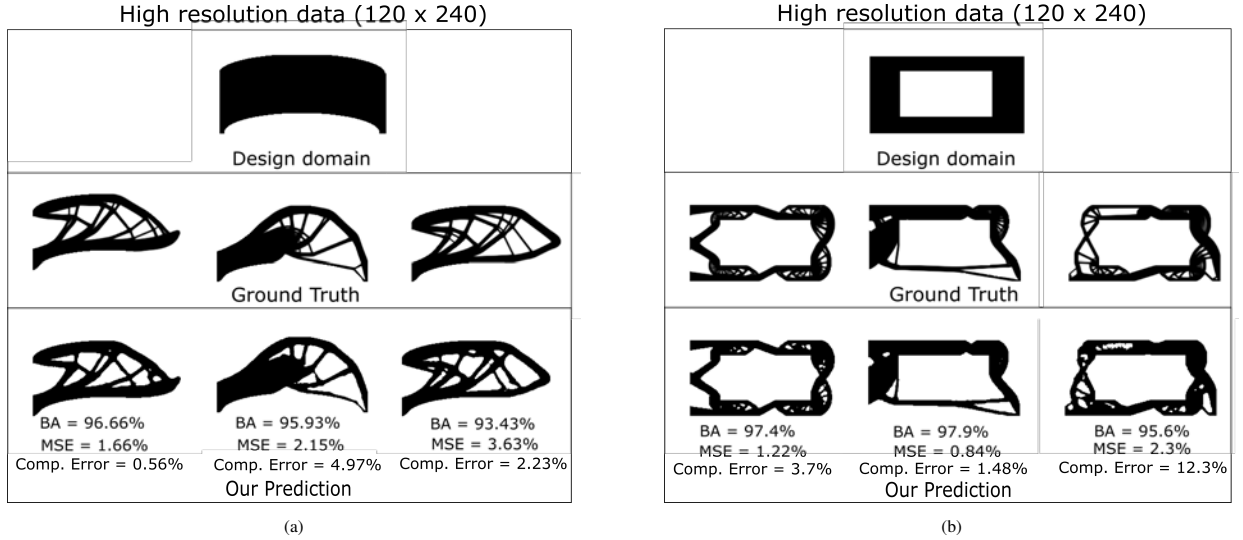

Figure 7: Predicted optimal structures versus ground truth (SIMP optimized) for high resolution domains that have different geometry and topology. The individual quality metrics for our predictions are presented in Table 1, and the prediction time is shown in Table 2. Observe that the source network for the 2D examples has only been trained on the domains shown in Figure 5, so the domains shown in this Figure are *unseen* to the source network.

time is 0.017 seconds per design case versus 138.0 seconds for the SIMP solver.

These resulting quality metrics are very promising, and are particularly so for the resolutions considered in the our experiments. Moreover, as the resolution increases the level of detail that is picked up by our transfer learning-based predictor increases as well. Notably, the normalized prediction time increases at a much slower rate compared to that of the SIMP algorithm.

Comparison With Traditional Deep Learning Methods

In order to compare our transfer learning-based method with other published deep learning methods for topology optimization, we examined two criteria. Specifically, we compare our method in terms of:

1. time required to generate equivalent training data producing similar prediction performance; and

2. prediction accuracy with the training data generated in the same amount of time.

One key advantage of our method compared to the traditional deep learning methods is that the method proposed in this paper requires a much smaller number of high-resolution cases to train the target network than to train an equivalent deep learning network. For example, we used 8,000 low resolution 2D cases and 1500 high resolution cases to obtain the high quality predictions shown in Figures 5-7. On top of that, the source network has to be trained only once with the 8,000 low resolution 2D cases.

On the other hand, training a deep learning network to produce a prediction of similar quality requires at least 8,000 high

Table 1: 2D structures: MSE, binary accuracies, and compliance error relative to SIMP.

Design Domain	Resolution	Number of test cases	MSE	Binary Accuracy	Compliance Error	Compliance Error Std.
from Fig. 5a (predicted by the source network)	40 x 80	2000	2.14%	96.61%	2.1%	0.054
from Fig. 5b	80 x 160	750	3.70%	94.50%	3.65%	0.055
from Figure 5c	120 x 160	500	3.45%	94.61%	4.81%	0.056
from Figure 5d	120 x 240	625	4.83%	94.59%	4.93%	0.062
from Figure 5e	160 x 320	375	4.29%	94.46%	6.54%	0.080
from Figure 5f	200 x 400	375	4.35%	94.55%	9.57%	0.106
Curved beam (Fig. 7a)	80 x 160	500	4.01%	94.44%	9.2%	0.088
Curved beam (Fig. 7a)	120 x 240	750	4.13%	93.94%	11.3 %	0.091
Frame (Fig. 7b)	80 x 160	500	2.61%	95.53%	5.39%	0.070
Frame (vol. fr. = 0.4) (Fig. 7b)	120 x 240	500	3.00%	94.80%	15.50%	0.182
L shape (vol. fr. = 0.3) (Fig. 6a)	120 x 240	500	2.50%	95.75%	13.20%	0.130
L shape w/hole (vol. fr. = 0.3) (Fig. 6b)	120 x 240	500	2.60%	95.72%	20.00%	0.210
Average			3.46%	94.95%	8.85%	0.098

Table 2: 2D Structures: Comparison of computational time of our predictions vs. the SIMP algorithm.

Resolution	SIMP (sec. per case)	Our method (sec. per case)
80 x 160 (Fig. 5b)	24	0.0093
120 x 160 (Fig. 5c)	36	0.010
120 x 240 (Fig. 5d)	80	0.015
160 x 320 (Fig. 5e)	200	0.022
200 x 400 (Fig. 5f)	350	0.030
Average	138	0.017

resolution cases for every design space and every type of boundary conditions. This is a significant time difference that becomes more severe as the resolution of the design space increases and as we move to 3D. For example, the SIMP algorithm [16] requires 5 seconds to produce the optimal topology for every low resolution (40 x 80) case and 350 seconds for every high resolution case (200 x 400). Thus, training our source and target networks requires *5.3 times* fewer high resolution cases than the equivalent CNN, and about 5 times less computation time. To put this in perspective, by using this particular SIMP algorithm, generating equivalent training datasets for our method is 620 hours faster than for an equivalent CNN. This difference rapidly increases with the increase in the fidelity of the desired results.

To be able to compare the prediction accuracies of transfer learning based vs deep learning based methods for the same amount of time, we replicated our target network in terms of layers, loss function, optimizer and so on, and trained it as a normal deep learning network. We determined the average total time needed to generate the training sets for our method for the example shown in Figure 5f, and then generated as many high-resolution cases for the deep CNN as possible in the same amount of time. Finally, we trained the deep CNN network with this dataset and compared the quality of the predictions between our method and of the corresponding deep CNN, as illustrated in Figure 8. This experiment clearly, although unsurprisingly, illustrates the significant superiority of our predictions compared to other deep learning methods based on data generated in the *same* amount of time.

Furthermore, the much smaller size of the training dataset re-
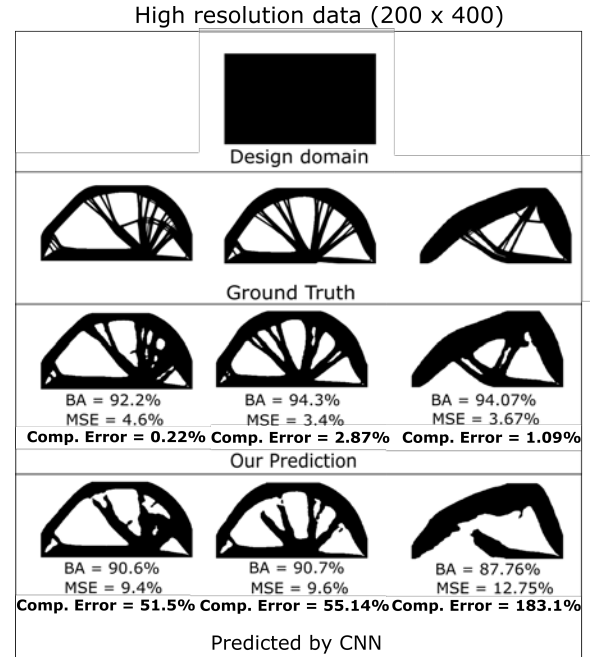
High resolution data (200 x 400)

Figure 8: Comparison between our predictions and those of a traditional CNN-based algorithm trained on 1500 and 1650 high resolution data, respectively. Our source network has been trained with low resolution cases as well, and we took into account the time required to generate these low resolution cases when selecting the number of high resolution cases for the traditional CNN-based algorithm.

quired by our transfer learning network implies that the corresponding training time is also much smaller than that required by traditional deep learning networks, as illustrated in Table 3.

4.2. 3D structures

We also applied our transfer-learning based method to 3D domains and we used a freely available Matlab Code [17] to generate the ground truth results. For the 3D examples, we used 12,000 low resolution data ($20 \times 40 \times 10$) to train the source model, and 1500 high(er) resolution data for fine-tuning the target model. Solving the TO problem in 3D is notoriously time

Table 3: 2D Structures: Comparison of the corresponding training time. The last two rows show the training time required by an equivalent Deep CNN.

Resolution	Number of training cases	Training time (seconds, per epoch)	Number of epochs	Training time (minutes)
40 x 80 (Fig. 5a)	8,000	62.87	29	30.3
80 x 160 (Fig. 5b)	1,500	26	4	1.73
120 x 160 (Fig. 5c)	1,500	27.19	5	2.26
120 x 240 (Fig. 5d)	1,500	34.88	10	5.81
160 x 320 (Fig. 5e)	1,500	41.25	8	5.5
200 x 400 (Fig. 5f)	1,500	58.68	9	8.80
200 x 400 (CNN) (Fig. 8)	1,650	67.43	28	31.46
200 x 400 (CNN)	8,000	307.54	29	148.64

consuming. Thus, to generate the ground truth cases for the examples used in this section, we limited the number of iterations of the SIMP solver to 150. To generate the training datasets for all 3D examples with parallelepipedic domains (Figures 9), the boundary conditions were randomly chosen from one of the 3 cases shown in Figures 4(a-c), and the location, orientation and magnitude have also been randomized. The cases shown in Figures 10 and 11 used randomized boundary conditions according to Figure 4(d).

We first show in Figures 9(a) & (b) the comparison between our predicted and the ground truth (SIMP) 3D optimal structures for two beams obtained for two different resolutions of the design space, namely $40 \times 80 \times 10$ and $80 \times 160 \times 10$. Moreover, Figures 9(c) & (d) show the same comparison for beams that have different topologies for a $40 \times 80 \times 10$ resolution. In order to help illustrate the difference between the two solutions more clearly, we also provide the symmetric difference between the ground truth and predicted result, i.e., the voxels that are in either one but not the other structure.

Importantly, our method can provide impressive performance even for cases for which the source network has not been specifically trained for. As an example, consider the design space illustrated in Figure 11, which was not part of the training set for our source model, but it was included in the much smaller dataset used to fine-tune the target model. The predicted optimal structures for this new design problem, which are summarized in Figures 10 & 11 are still highly accurate. Table 4 shows the average MSE, binary accuracy and compliance error for the 3D cases described above. Note that average MSE and binary accuracy are also around 3% and 95% for these 3D cases. Furthermore, we summarize in Table 5 the time required by our method to predict the 3D optimal structures, and compare these times with those required by the SIMP method to reach 150 iterations for the same 3D problems. Our method is consistently multiple orders of magnitude faster than the SIMP method, and achieves real-time rates even for our preliminary and non-optimized implementation.

Similarly with the 2D case described above, the time required to generate a comparable training dataset requires much less time for our method relative to the time required for an equivalent deep CNN. By extrapolating from our 3D experiments, our method requires a time that is at least 5 times less than the time required to generate the comparable dataset for the equivalent deep CNN, and the difference increases with the resolution. For example, for the case shown in Figure 11, by using the algorithm described in [17] run in parallel on 100 processors, for a $40 \times 40 \times 40$ resolution, and with the optimization stopped early at the 150th iteration for each case, the dataset generation for our method requires approximately 152 hours (6.3 days) less than the time required to generate an equivalent dataset for the deep CNN, even with such an aggressive parallelization. At higher resolutions, and assuming that one can use a highly parallelized algorithm, such as the one implemented on the GPU described in [15] that computes an optimal topology for a cantilever beam for a $200 \times 100 \times 100$ resolution in 144 seconds, our transfer learning-based method would need at least 388 hours (16.2 days!) less time than the time required to generate the equivalent training set for the deep CNN.

4.3. Generalizability of Network Predictions

The solution manifolds between our low-resolution and high-resolution domains for some of the examples shown in this paper can be considered to already be dissimilar. Specifically, our source network is trained on rectangular/parallelepipedic low-resolution domains of genus 0, but the target network is fine-tuned on domains whose geometry, topology and boundary conditions are unseen to the source network. Nevertheless, to further confirm the generalization performance of our method, we also considered the types of problems from [44], which use local density constraints for the topology optimization. It is important to note that we used here the same source network trained on the dataset as explained above, but only fine-tuned the target network on the data set used in [44].

The results are shown in Figure 12(a). Note that the predictions of our network are visually very similar to the ground truth. Moreover, the work presented in [44] used 7000 training cases, while our target network only required 1500 cases for fine-tuning. The fact that we used the same source network for this new, more complex TO problem strongly affirms that knowledge is being transferred between our source and target networks.

Moreover, we have also explored the heat sink design problem, which results in a tree-like optimal structure with very thin branches. The ground truth solution in this case has genus 0, so this can be considered more of a size rather than a topology optimization problem.

We adapted our network to the heat sink design problem to further illustrate the capabilities of our method and of the knowledge transfer abilities between our source and target network. Figure 12(b) shows the network performance for some of these test cases, including the predicted optimal solutions before and after the application of the density threshold, i.e., integer rounding for the density values. While capturing the very thin members of the optimal structure remains a limitation, which is discussed in the next section, our results indicate that knowledge is transferred between the source and the target network, as intended, even for the case of heat sink design.

(a)   (b)

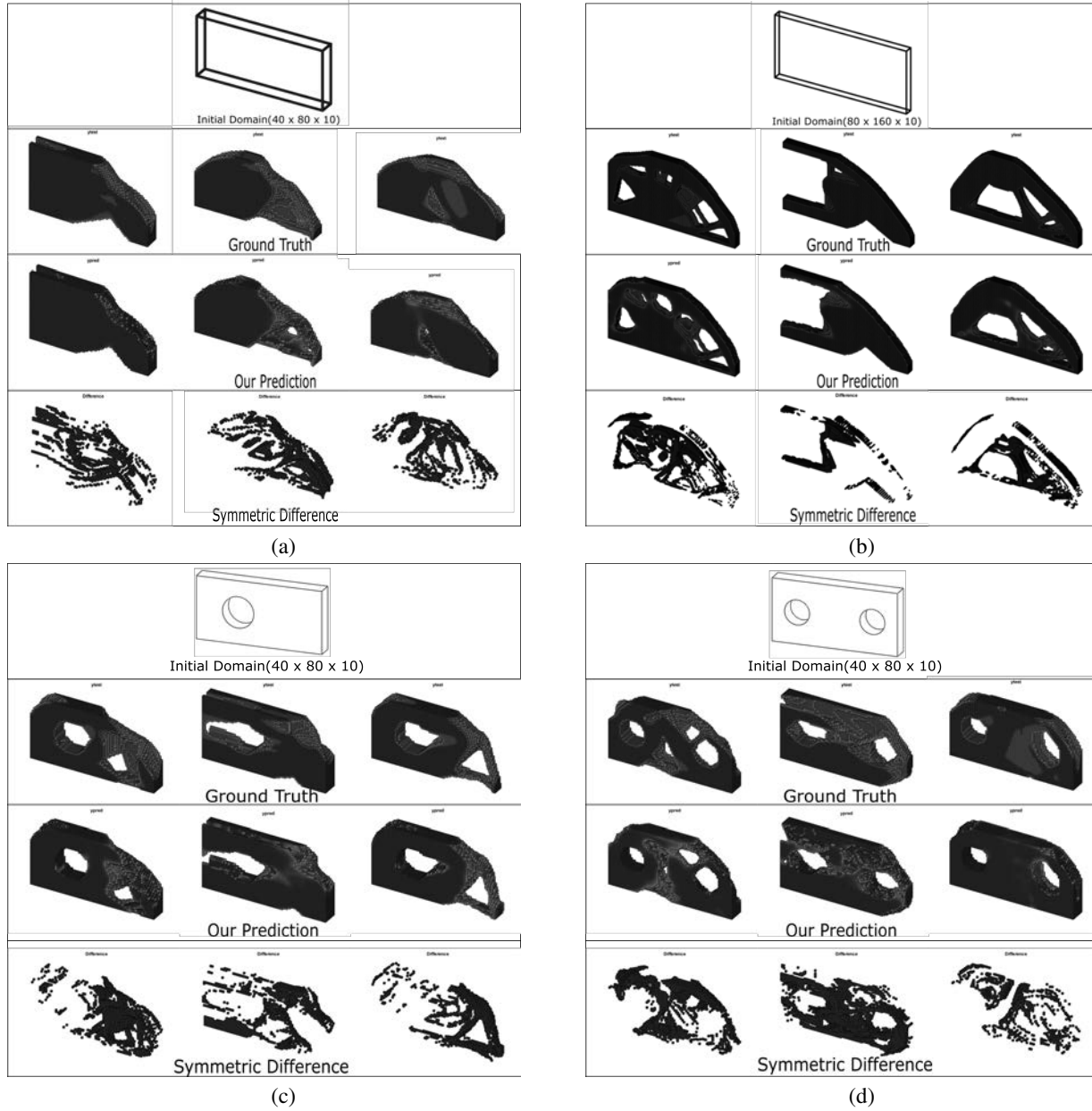(c)   (d)

Figure 9: The predicted and ground truth optimal structures for 3D design spaces, including their symmetric difference. Figures (a) and (b) show a simple parallelepi-pedic domain with two resolutions and figures (c) and (d) show a beam with one or two holes, respectively. The individual quality metrics for our predictions are presented in Table 4, and the prediction time is shown in Table 5.

It is also important to note that in all the examples presented in the paper we used the same source network trained only once, as described above.

4.4. *Current Limitations in Capturing Thin Members*

Our transfer learning network has not been designed to achieve high prediction performance of very thin members. We hypothesize that there are several reasons for this behavior. First, our current network does not have a complexity that is sufficiently high, mainly due to GPU limitations imposed by our hardware on the size of the layer output during training. A more complex network would definitely improve detection performance

for the thin members. On the other hand, a more complex network would need more training data, which is expensive to generate with the SIMP algorithms that we used. Second, the thin members that we predict with a somewhat lower accuracy have a width of 1-2 pixels/voxels, which is at or below 1% of the largest dimension of the domain. Increasing the resolution relative to the width of the members would also improve prediction performance for the thin members. This, in turn, would require higher resolution training data, whose computational cost increases exponentially with the resolution, as well as a more complex network architecture. Finally, we applied in all our examples a simple integer rounding as a threshold for the densities of the in-
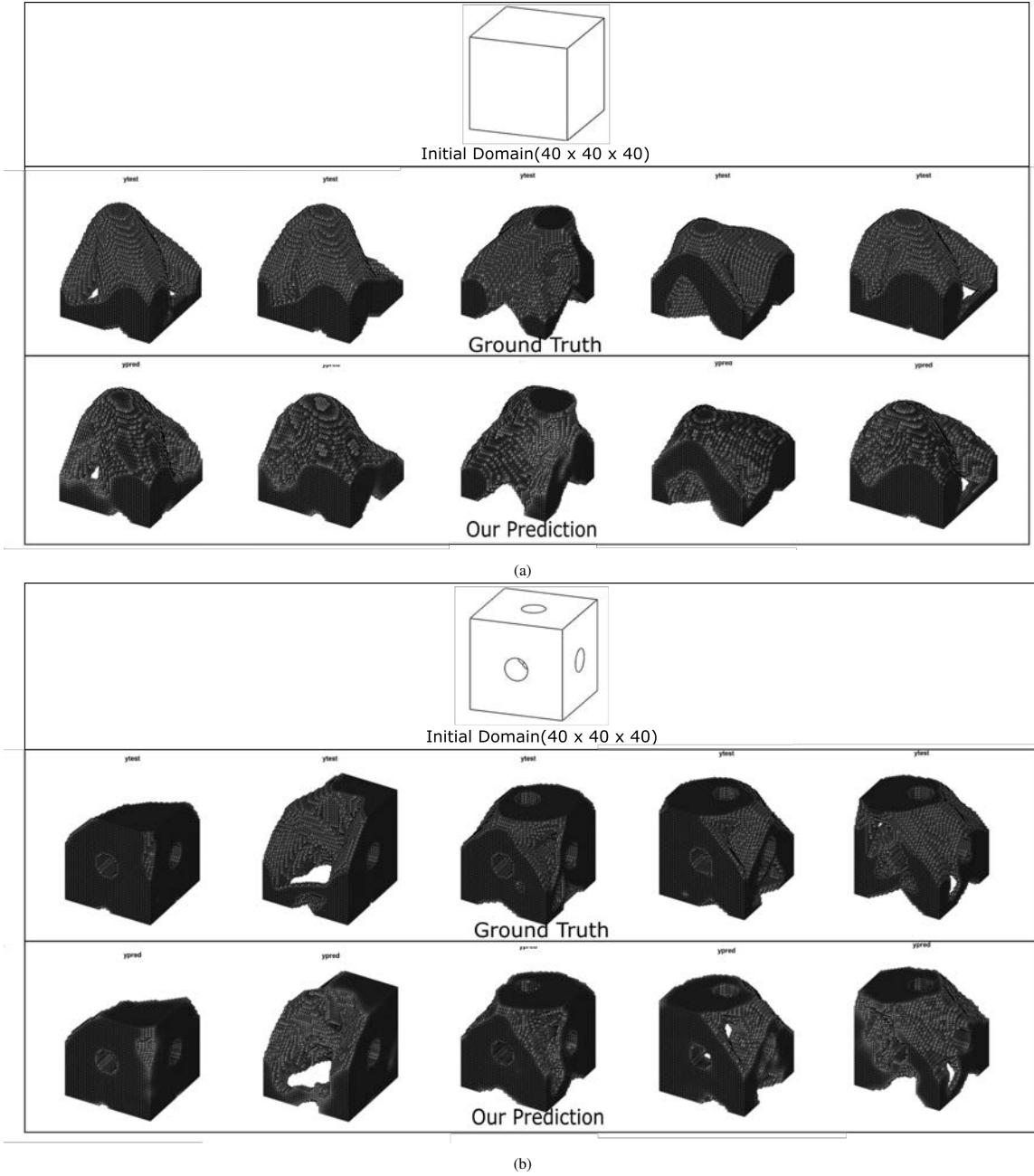
(a)

(b)

Figure 10: Predicted versus ground truth structures for two different domains as well as the boundary conditions shown in Figure 4(d). Both domains and boundary conditions are unseen to the source network. The individual quality metrics for our predictions are presented in Table 4, and the prediction time is shown in Table 5.

dividual elements. This, in turn, eliminated from the solutions that we report all pixels/voxels whose density was below 0.5, including those that belonged to the thin members. However, our network predicts a much more nuanced density field, as shown in the last row of Figure 12.

4.5. Transfer Learning in Conjunction with SIMP

One way to take full advantage of the information output by our network without increasing the complexity of the network architecture and the data resolution is to couple our transfer learning-based predictions to SIMP to obtain a more accurate definition of the thin members. Table 7 shows the time required

for the SIMP algorithm to generate the optimal structure by using as input the prediction output by our network. Specifically, for a 200 x 400 resolution domain, the SIMP algorithm that we used calculates such an optimum structure in 8 seconds (with an average of 4.5 seconds) compared to 350 seconds needed by a normal SIMP optimization starting with the full design domain.

5. Conclusions

We proposed in this paper a highly efficient and accurate non-iterative topology optimization method that uses transfer learning on a convolutional neural network architecture. Our method
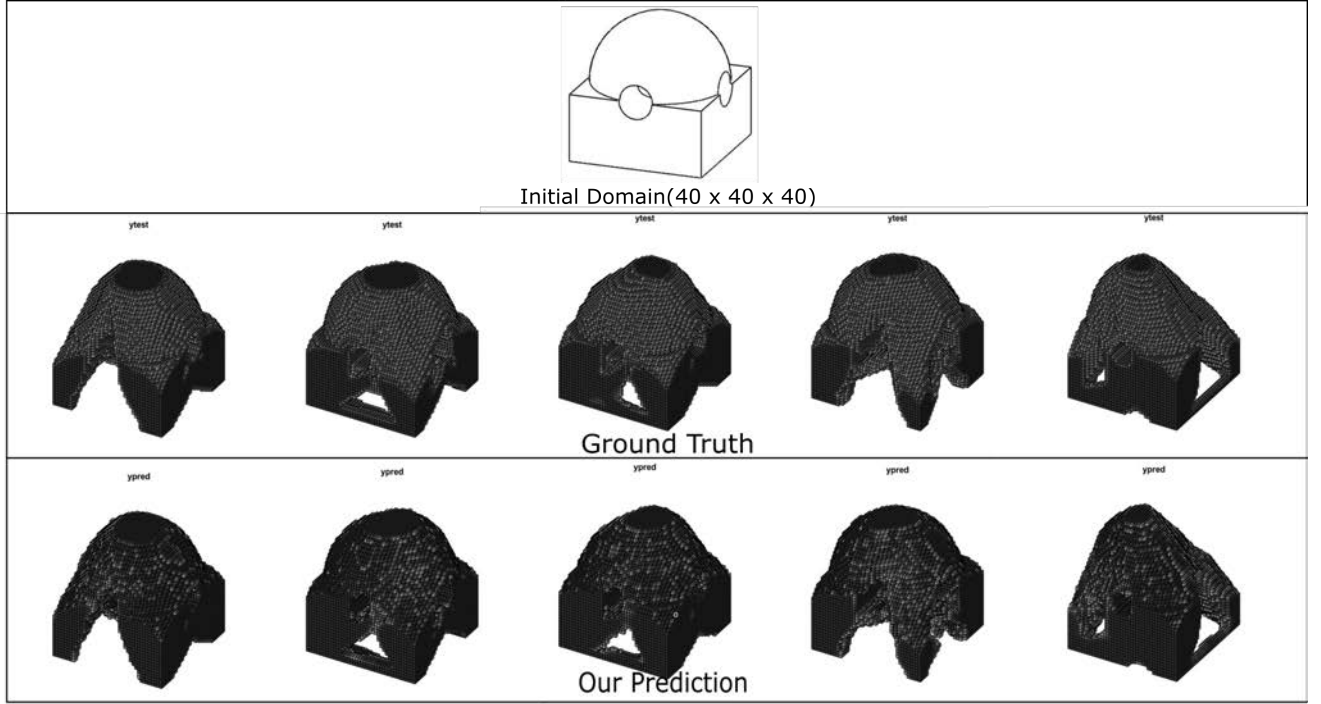
Figure 11: A domain and boundary conditions that were not in the training set for our source model, but it was included in the much smaller dataset used for fine-tuning the target model. The individual quality metrics for our predictions are presented in Table 4, and the training time is shown in Table 6.

Table 4: 3D structures: MSE, Binary Accuracy and Compliance Error relative to SIMP.

Design Domain	Resolution	Number of test cases	MSE	Binary Accuracy	Compliance Error
Domain	20 x 40 x 10	1000	2.04%	95.62%	1.56%
Domain (Fig. 9a)	40 x 80 x 10	300	3.14%	94.31%	2.43%
Domain (Fig. 9b)	80 x 160 x 10	100	3.1%	93.9 %	10.1 %
With hole (Fig. 9c)	40 x 80 x 10	150	3.45%	94.00%	2.05%
With 2 hole (Fig. 9d)	40 x 80 x 10	150	3.52%	94.31%	2.85%
Cube (Fig. 10a)	40 x 40 x 40	175	3.28%	93.29%	9.9%
Cube with holes (Fig. 10b)	40 x 40 x 40	180	3.51%	93.11%	7.5%
Dome with holes (Fig. 11)	40 x 40 x 40	200	2.41%	95.71%	0.38%
Average			3.05%	94.28%	4.60%

Table 5: 3D structures: comparison of prediction time vs. SIMP algorithm.

Resolution	SIMP (sec. per case)	Our method (sec. per case)
20 x 40 x 10	300	0.015
40 x 80 x 10 (Fig. 9a)	4,500	0.031
80 x 160 x 10 (Fig. 9b)	7,500	0.04
40 x 40 x 40 (Fig. 10)	5,550	0.033
Average	4,462.5	0.029

uses low resolution datasets to train a source network and a much smaller high resolution dataset to fine-tune a target network. The learned knowledge captured by the source network once is transferred to the target network, so that the latter requires a much smaller number of training cases than an equivalent deep CNN to

make predictions with the same level of accuracy. We provided numerous examples to show that the proposed method produces predictions of the optimal 3D topologies at real-time rates for non-trivial 3D high resolution TO problems. Furthermore, we showed that the proposed method can produce accurate predictions efficiently for various design spaces, boundary conditions, and volume fractions, including for cases that have not been part of the source network's training set. Our experiments achieved an average binary accuracy and MSE around 95% and 3%, respectively, at real-time rates in both 2D and 3D.

Like any other data-driven method, our approach inherits any existing data biases in the datasets. In our experiments, we reduced the biases by randomizing the input used to generate the ground truth structures. Moreover, the capability of this method to explore regions of the design space for which the algorithm has not been trained for has the same limitations as most other transfer learning algorithms. In addition, when the source task and the

12

Table 6: 3D structures: Training time

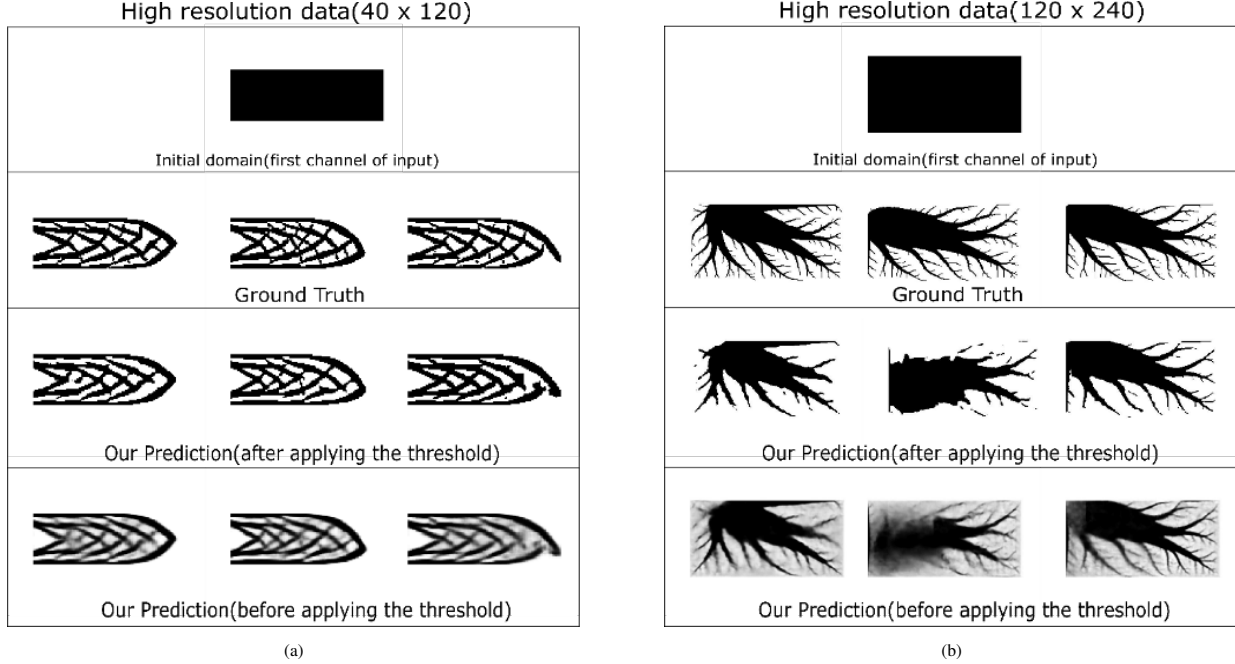| Resolution	Number of training cases	Training time (seconds, per epoch)	Number of epochs	Training time (minutes)
20 x 40 x 10	12,000	1,101.87	20	367.25
40 x 80 x 10 (Fig. 9a)	1,500	174.56	5	14.54
80 x 160 x 10 (Fig. 9b)	1,500	290	15	72.5
40 x 40 x 40 (Cube) (Fig. 10a)	1,900	252	20	84
40 x 40 x 40 (Cube with holes) (Fig. 10b)	1,700	230	30	115
40 x 40 x 40 (Dome with 2 holes) (Fig. 11)	1,500	202	35	117.8

(a)

(b)

Figure 12: Predicted optimal structures versus ground truth (SIMP optimized) for problems with local density constraints [44] and heat sink design. The fact that we used the same source network for these new, more complex TO problems strongly affirms that knowledge is being transferred between our source and target networks.

Table 7: Average time required by SIMP to generate the optimal solution starting from our predicted structure.

Resolution	Time (sec. per case)
80 x 160 (Fig. 5b)	1
120 x 160 (Fig. 5c)	3.5
120 x 240 (Fig. 5d)	4
160 x 320 (Fig. 5e)	6
200 x 400 (Fig. 5f)	8
Average	4.5

target task are not similar enough, negative transfer may occur and the algorithm performance may fail to improve [32] without additional information. This, however, is not unlike what happens in real life. Consider one of the traditional examples used to explain transfer learning, namely that of learning how to ride a bicycle. Clearly these skills can be transferred by bicycle riders and used to learn how to ride other two wheeled devices, such as motorcycles or scooters. However, the same bicycle riding skills cannot be easily employed to ride, for example, unicycles - as anyone that has tried to ride a unicycle can attest to.

Perhaps the key bottleneck of any data driven TO method is the computational cost to generate suitable training data, which is computationally demanding for gradient-based optimization algorithms, and is particularly so in 3D. On one hand, our experiments show that the proposed transfer learning-based method requires much less time than equivalent deep CNN to generate the training dataset to reach the same accuracy. On the other hand, employing more efficient gradient-based approaches to generate ground truth optimal structures are needed to be able to perform careful studies of how to best train the proposed transfer learning-based method and to better understand its generalization capabilities, scalability and limitations. Fortunately, recent advances in software and hardware architectures, such as the recently announced optimized physics libraries from AMD and NVIDIA, which promise to include FEA capabilities, come at the right time and with the potential to dramatically speed up the data generation for our purposes.

Generalizability is a critical aspect of any machine learning-based method. On one hand, we discussed In section 4 the capability of the proposed transfer learning method to transfer knowledge between the source and the target network, and we illustrated the generalizability of its predictions. However, a more complete treatment of this difficult question, which has not been addressed yet in the literature, would likely involve constructions

of local subspaces that approximate the solution manifolds corresponding to the low and high resolution problems, and definitions of new metrics that would measure the "distance" between the corresponding approximations of the solution manifolds. Very likely, an effective distance would probably require the projection of these approximations of the solution manifolds onto some common subspace, but there are a number of important open problems that need to be solved first. Perhaps the current efforts in the mathematical optimization community focused on Reduced Bases Methods could provide some insight into these issues.

It is also possible that modifying the type of input to replace or augment the explicit boundary conditions by one or more physical fields, such as displacement and stress/strain, would further improve the generalization capabilities of the proposed method. However, such a study is outside the scope of this paper.

Nevertheless, the proposed approach shows that transfer learning can serve as a practical underlying framework for performing real-time 3D design space explorations with topology optimization. To the best of our knowledge, this paper documents the first attempt to use transfer learning for topology optimization and provides exciting and important directions for future research.

[1] O. Sigmund, K. Maute, Topology optimization approaches, Structural and Multidisciplinary Optimization: A Comparative Review 48 (6) (2013) 1031–1055.

[2] O. Amir, O. Sigmund, On reducing computational effort in topology optimization: how far can we go?, Structural and Multidisciplinary Optimization 44 (1) (2011) 25–29.

[3] A. Limkilde, A. Evgrafov, J. Gravesen, On reducing computational effort in topology optimization: we can go at least this far!, Structural and Multidisciplinary Optimization 58 (6) (2018) 2481–2492.

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

[5] B. Li, C. Huang, X. Li, S. Zheng, J. Hong, Non-iterative structural topology optimization using deep learning, Computer-Aided Design.

[6] 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.

[7] M. P. Bendsøe, Optimal shape design as a material distribution problem, Structural optimization 1 (4) (1989) 193–202.

[8] G. Allaire, F. Jouve, A.-M. Toader, A level-set method for shape optimization, Comptes Rendus Mathematique 334 (12) (2002) 1125–1130.

[9] J. Chen, V. Shapiro, K. Suresh, I. Tsukanov, Shape optimization with topological changes and parametric control, International Journal for Numerical Methods in Engineering 71 (3) (2007) 313–346.

[10] H. A. Eschenauer, V. V. Kobelev, A. Schumacher, Bubble method for topology and shape optimization of structures, Structural optimization 8 (1) (1994) 42–51.

[11] A. A. Novotny, R. A. Feijóo, E. Taroco, C. Padra, Topological sensitivity analysis, Computer methods in applied mechanics and engineering 192 (7-8) (2003) 803–829.

[12] B. Bourdin, A. Chambolle, Design-dependent loads in topology optimization, ESAIM: Control, Optimisation and Calculus of Variations 9 (2003) 19–48.

[13] J. Sokolowski, A. Zochowski, Topological derivative in shape optimization, Encyclopedia of Optimization (2009) 3908–3918.

[14] T. Borrvall, J. Petersson, Large-scale topology optimization in 3D using parallel computing, Computer methods in applied mechanics and engineering 190 (46-47) (2001) 6201–6229.

[15] J. Wu, C. Dick, R. Westermann, A system for high-resolution topology optimization, IEEE transactions on visualization and computer graphics 22 (3) (2015) 1195–1208.

[16] E. Andreassen, A. Clausen, M. Schevenels, B. S. Lazarov, O. Sigmund, Efficient topology optimization in MATLAB using 88 lines of code, Structural and Multidisciplinary Optimization 43 (1) (2011) 1–16.

[17] K. Liu, A. Tovar, An efficient 3D topology optimization code written in Matlab, Structural and Multidisciplinary Optimization 50 (6) (2014) 1175–1196.

[18] I. G. Jang, B. M. Kwak, Design space optimization using design space adjustment and refinement, Structural and Multidisciplinary Optimization 35 (1) (2008) 41–54.

[19] S. Y. Kim, I. Y. Kim, C. K. Mechefske, A new efficient convergence criterion for reducing computational expense in topology optimization: reducible design variable method, International Journal For Numerical Methods In Engineering 90 (6) (2012) 752–783.

[20] M. E. Lynch, S. Sarkar, K. Maute, Machine learning to aid tuning of numerical parameters in topology optimization, Journal of Mechanical Design 141 (11).

[21] I. Sosnovika, I. Oseledetsb, Neural networks for topology optimization, arXiv:1709.09578.

[22] Q. Lin, J. Hong, Z. Liu, B. Li, J. Wang, Investigation into the topology optimization for conductive heat transfer based on deep learning approach, International Communications in Heat and Mass Transfer 97 (2018) 103–109.

[23] A. Gaymann, F. Montomoli, Deep neural network and monte carlo tree search applied to fluid-structure topology optimization, Scientific reports 9 (1) (2019) 1–16.

[24] N. J. O'Neill, Standard and inception-based encoder-decoder neural networks for predicting the solution convergence of design optimization algorithms, Master's thesis, University of Colorado Boulder (2019).

[25] Y. Zhang, A. Chen, B. Peng, X. Zhou, D. Wang, A deep Convolutional Neural Network for topology optimization with strong generalization ability, arXiv preprint arXiv:1901.07761.

[26] S. Rawat, M. H. Shen, Application of Adversarial Networks for 3D structural topology optimization, Tech. rep., SAE Technical Paper (2019).

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

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

[29] T. Guo, D. J. Lohan, R. Cang, M. Y. Ren, J. T. Allison, 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, 2018, p. 0804.

[30] X. Lei, C. Liu, Z. Du, W. Zhang, X. Guo, Machine learning-driven real-time topology optimization under moving morphable component-based framework, Journal of Applied Mechanics 86 (1) (2019) 011004.

[31] K. Weiss, T. M. Khoshgoftaar, D. Wang, A survey of transfer learning, Journal of Big data 3 (1) (2016) 9.

[32] L. Torrey, J. Shavlik, Transfer learning, in: Handbook of research on machine learning applications and trends: algorithms, methods, and techniques, IGI Global, 2010, pp. 242–264.

[33] A. Khatami, M. Babaie, H. R. Tizhoosh, A. Khosravi, T. Nguyen, S. Nahavandi, A sequential search-space shrinking using cnn transfer learning and a radon projection pool for medical image retrieval, Expert Systems with Applications 100 (2018) 224–233.

[34] I. Hossain, A. Khosravi, I. Hettiarachchi, S. Nahavandi, Multiclass informative instance transfer learning framework for motor imagery-based brain-computer interface, Computational Intelligence and Neuroscience 2018.

[35] S. J. Pan, Q. Yang, W. Fan, S. J. P. (ph. D), A survey on transfer learning, IEEE Transactions on Knowledge and Data Engineering.

[36] C. Le, J. Norato, T. Bruns, C. Ha, D. Tortorelli, Stress-based topology optimization for continua, Structural and Multidisciplinary Optimization 41 (4) (2010) 605–620.

[37] O. Sigmund, Morphology-based black and white filters for topology optimization, Structural and Multidisciplinary Optimization 33 (4-5) (2007) 401–424.

[38] G. Steiner, Transfer of learning, cognitive psychology of, in: N. J. Smelser, P. B. Baltes (Eds.), International Encyclopedia of the Social & Behavioral Sciences, Pergamon, 2001, pp. 15845 – 15851.

[39] M. T. McCann, K. H. Jin, M. Unser, Convolutional neural networks for inverse problems in imaging: A review, IEEE Signal Processing Magazine 34 (6) (2017) 85–95.

[40] J. Lehman, K. O. Stanley, Revising the evolutionary computation abstraction: minimal criteria novelty search, in: Proceedings of the 12th annual conference on Genetic and evolutionary computation, ACM, 2010, pp. 103–110.

[41] B. Bourdin, Filters in topology optimization, International Journal For Numerical Methods In Engineering 50 (9) (2001) 2143–2158.

[42] P. Christen, K. Goiser, Quality and complexity measures for data linkage and deduplication, in: Quality measures in data mining, Springer, 2007, pp. 127–151.

[43] J. Birch, T. Robertson, A classroom note on the sample variance and the second moment, The American Mathematical Monthly 90 (10) (1983) 703–705.

[44] R. Cang, H. Yao, Y. Ren, One-shot generation of near-optimal topology through theory-driven machine learning, Computer-Aided Design 109 (2019) 12–21.