An Artificial Neural Network Approach to Generating High-Resolution Designs from Low-Resolution Input in Topology Optimization

Nicholas Napier
Research Assistant,
Department of Aerospace Engineering,
University of Illinois at Urbana-Champaign,
Urbana, IL 61801
e-mail: napier2@illinois.edu

Sai-Aksharah Sriraman
Research Assistant,
Department of Aerospace Engineering,
University of Illinois at Urbana-Champaign,
Urbana, IL 61801
e-mail: sssrira2@illinois.edu

Huy T. Tran
Research Assistant Professor,
Department of Aerospace Engineering,
University of Illinois at Urbana-Champaign,
Urbana, IL 61801
e-mail: huytran1@illinois.edu

Kai A. James
Assistant Professor,
Department of Aerospace Engineering,
University of Illinois at Urbana-Champaign,
Urbana, IL 61801
e-mail: kaijames@illinois.edu

NOMENCLATURE

z_j^l	Weighted sum of connections to a single node in the neural network
w_{ji}^l	Current weight associated with a given connection in the neural network
x_i	Input node value in the neural network
a_j^l	Neural network node activation value
C_f	Cost function
$\hat{y}_i$	Predicted output
y_i	Target output
$w_{ji}^{l\prime}$	Updated weight associated with a given connection in the neural network
η	Learning rate parameter used in network weight optimization
ρ	Current output density from the neural network
ρ_{new}	Filtered density value
$\boldsymbol{J}$	Jacobian of network output with respect to the network weights
μ	Damping coefficient between Gauss-Newton and Gradient Descent methods
$\boldsymbol{I}$	Identity matrix
$\boldsymbol{e}$	Network error at the output nodes
MSE	Mean-squared-error
B	Batch size of observations during neural network training
N	Number of output nodes in the neural network

C	Structural compliance
U	Global displacement used during finite element analysis
K	Global stiffness matrix used during finite element analysis
F	Global consistent force vector used during finite element analysis
u_e	Element displacement matrix of topology optimization
k_e	Element stiffness matrix for topology optimization
x_{min}	Minimum relative density for topology optimization
D	Number of elements in discretized domain for topology optimization
V	Material domain volume
V_0	Topology optimization design domain volume
p	Penalization power for topology optimization
f	Prescribed volume fraction for topology optimization
A	Area of elements in the topology optimization domain
$\bar{T}$	Dirichlet boundary temperature condition
q	Applied heat flux

ABSTRACT

We address a central issue that arises within element-based topology optimization: To achieve a sufficiently well-defined material interface, one requires a highly refined finite element mesh, however, this leads to an increased computational cost due to the solution of the finite element analysis problem. By generating an optimal structure on a coarse mesh and using an artificial neural network to map this coarse solution to a refined mesh, we can greatly reduce computation time. This approach resulted in time savings of up to 85% for test cases considered. This significant advantage in computation time also preserves the structural integrity when compared to a fine-mesh optimization with limited error. Along with the savings in computation time, the boundary edges become more refined during the process, allowing for a sharp transition from solid to void. This improved boundary edge can be leveraged to improve the manufacturability of the optimized designs.

1. INTRODUCTION

Topology optimization is a widely used computational method for generating optimal structural designs by systematically distributing material throughout a two- or three-dimensional working domain. The method was first proposed by Bendsøe and Kikuchi [1] in 1988 and has been used to solve a wide variety of problems, from the design of piezoelectric nanostructures [2] to synthesis of bistable viscoelastic systems [3]. The method relies upon finite element analysis to discretize the working domain and model the mechanical response of the structure. The material distribution within the optimized structure is generally represented using a pixelated grayscale image, where each pixel corresponds to the material properties of a discrete element within the finite element mesh. Consequently, one of the major challenges encountered when using topology optimization is achieving a smooth material interface, as the resolution of the boundary between disparate material regions is limited by the coarseness of the finite element mesh. A jagged boundary is impractical for visualization purposes, as well as for manufacturing.

The simplest solution to this problem is to increase the number of elements in the finite element mesh. While this approach can lead to mesh-dependence issues [4, 5], these issues can be resolved using filtering techniques [6] or Heaviside projection [7]. Instead, the main drawback of increasing the number of elements is the increased computational cost of the finite element analysis. This cost is particularly high when solving problems involving material or geometric nonlinearities [8, 9]. Other authors have sought to mitigate the increased computational cost by performing adaptive mesh refinement [10, 11]. In this way, they increase the number of elements only in the regions of the domain that contain transitions. While this approach leads to fewer elements than uniform mesh-refinement, it still significantly increases the cost of the finite element analysis, and it introduces conceptual challenges as the code developer must implement a complex procedure for detecting and refining the material interface. A third option is to take a *level set* approach, in which users directly optimize the location of the material boundary, by implicitly representing the path of the boundary as a higher-order function [12]. This method enables smooth boundary representation even with a relatively coarse finite element mesh. However, like adaptive mesh-refinement, this method introduces additional complexity to the algorithm, as the developer must implement functions for interpolating, evolving, and re-initializing the *level set* function [13].

Another class of solutions to the issues of boundary coarseness involves post-processing of low-resolution designs. One such approach is to smooth the boundary using spline interpolation [14]. Yet there is no closed form interpolation solution for increasing domain resolution while still maintaining the precise geometry of the underlying optimized structure. An alternative approach is to optimize a nonlinear function approximator to predict the domain of a high-resolution design given a low-resolution input. Framed in this manner, the problem becomes

one of single image super-resolution; i.e., enhancing the resolution of a low-resolution image. Machine learning algorithms are one approach to solving this problem. For example, deep convolutional neural networks (CNNs) have been shown to successfully produce an end-to-end mapping between low- and high-resolution images from color image data sets [15, 16]. In this context, end-to-end refers to the ability of these models to output high-resolution images without the need for detailed feature engineering or independent optimization of various model aspects. However, one drawback of a deep CNN implementation is that it typically requires optimization of a large number of network parameters, thus requiring a large training data set to achieve high model accuracy. For example, Dong et al. train a network with 8,032 parameters using data sets ranging from 24,800 to over 5 million observations [16]. Large, publicly available image data sets such as ImageNet support the use of such deep networks. However, images in these data sets are complex color images which may not be representative of grayscale images produced by topology optimization algorithms. We therefore focus on shallow multilayer perceptron artificial neural networks (referred to as ANNs for simplicity) trained on topology optimization-specific images for this paper. Our proposed method uses 20 images partitioned down to a range of 1,198 to 3,486 observations depending on the model selected, thus requiring significantly less data than what is common in deep CNN applications. ANNs similar to our proposed models have successfully been used for pattern recognition of grayscale handwritten numbers [17] and interpolation of grayscale source images for conversion to color [18].

We therefore present the following as the guiding hypothesis of the current study: ANNs can be used to process low-resolution grayscale images generated by topology optimization algorithms and produce high-resolution presentations of the optimized designs. Furthermore, this task can be accomplished without compromising structural integrity, without requiring prohibitively large training data sets, and with a fraction of the computational effort required to implement mesh-refinement strategies.

2. ARTIFICIAL NEURAL NETWORK IMPLEMENTATION

2.1 Background

ANNs have been successfully applied to a variety of problems, including pattern recognition and classification [19], surrogate-based design [20, 21], reliability analysis [22], and other prediction tasks in domains ranging from energy systems [23, 24] to bioengineering [25] and many others. ANNs are a type of machine learning method commonly applied to supervised learning problems. In supervised learning, a model is trained from a set of labeled data (i.e., each input observation has a corresponding labeled target) to produce an inferred function mapping input observation to corresponding outputs [26]. ANNs represent this inferred mapping through their network architecture, in which each connection in the network is assigned a weight whose value is optimized to minimize resulting errors in prediction of training data. Prediction tasks are often separated into classification and regression. Classification concerns prediction of a category (e.g., a hand-written numerical digit) or qualitative variable (e.g., gender) for an input observation; in comparison, regression concerns prediction of a quantitative numerical value associated with an input observation. Here, we apply ANNs as a nonlinear function approximator for a supervised learning regression problem, namely single image super-resolution of optimized topology domains.

An ANN is represented by a network of artificial neurons, inspired by connectivity among millions of neurons within the body. Information is provided to an ANN at its layer of input nodes, in the form of an observation (i.e., a record within a data set) represented by a vector or multi-dimensional array of matching dimension to the layer of input nodes. This information then passes

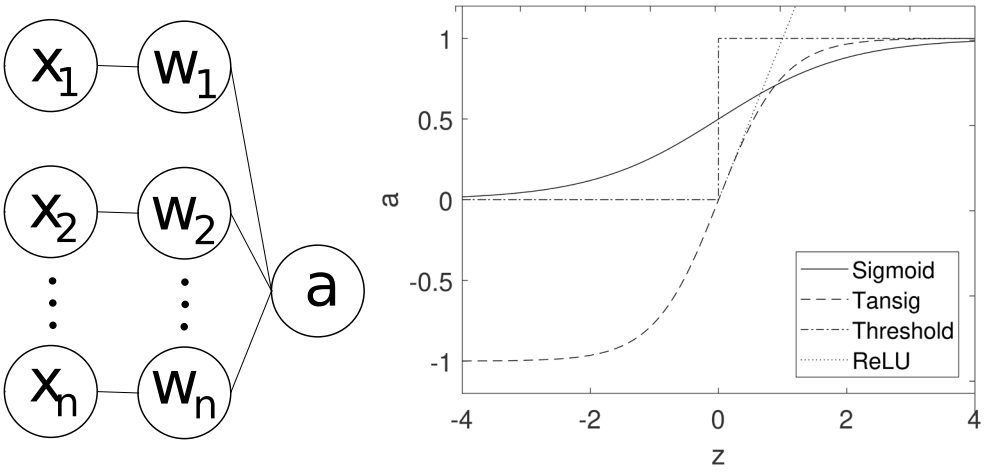
through one or more connected layers of nodes to produce predicted values at output nodes. Information passing through the network goes through activation functions defined at each node. A simple artificial neuron is the perceptron, which uses a threshold-based activation function to produce a binary output given a linear combination of input values [27]. Other common activation functions for ANNs are linear, sigmoid, tansig, and rectified linear units.

Figure 1 shows a notional artificial neuron and common activation functions employed within ANNs. Edge weights represent the influence of each input to a node. The weighted sum of inputs is then supplied to a node's activation function to determine the output of that node as follows (using a sigmoid activation as an example),

$$z_j^l = \sum_i w_{ji}^l x_i \tag{1}$$

$$a_j^l = \frac{1}{1+e^{-z_j^l}} \tag{2}$$

where x_i is the value for the i^{th} input variable, w_{ji}^l is the weight for the edge from the i^{th} node in the (l^{th} – 1) layer to the j^{th} node in the l^{th} layer, z_j^l is the weighted sum of inputs to the j^{th} node in the l^{th} layer, and a_j^l is the activation value of the j^{th} node in the l^{th} layer. Nonlinear activation functions such as the sigmoid function, combined with highly connected network architectures, allow ANNs to model complex, highly nonlinear relationships in data.



(a) Generic artificial neuron (b) Common activation functions

Figure 1: Notional artificial neuron with common activation functions

Figure 2 shows a fully connected ANN with one hidden layer for predicting *N* output variables given an observation consisting of *d* features (i.e., input variables). Note that bias nodes are not shown in Figures 1 or 2. Weight values for the entire network are adjusted during the training process to minimize prediction errors using a defined cost function. We choose mean squared error (MSE) due to its smoothness and analytical differentiability, calculated as follows,

$$C_f = \frac{1}{N}\sum_i^N \|\hat{y}_i - y_i\|^2 \tag{3}$$

where $\hat{y}_i$ is the predicted output and y_i the actual target value for data point *i*. A differentiable cost function enables use of gradient-based optimization methods such as gradient descent for optimizing network weights. Individual weights can then be updated as follows,

$$w_{ji}^{l\prime} = w_{ji}^{l} - \eta \nabla C \tag{4}$$

where η is the learning rate hyper-parameter, ∇C is the cost function gradient, w_{ji}^{l} is a network weight for the current iteration, and $w_{ji}^{l\prime}$ is the updated weight to be used in the next iteration. Backpropagation can be used to determine cost function gradients. Choosing the architecture of an ANN (i.e., its number of nodes, hidden layers, and connections) and associated hyper-parameters is often driven by heuristic and practical guidelines, though grid searches and response surface mapping have also been used to guide the process [29].

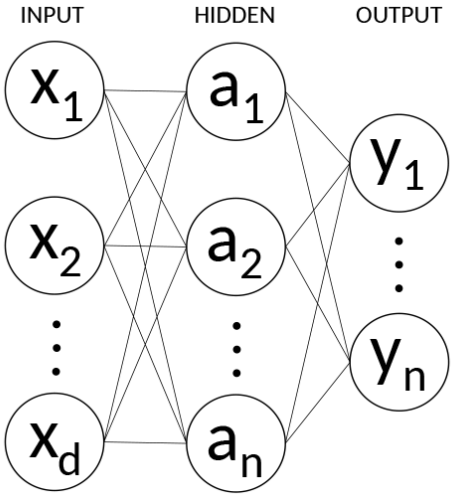


Figure 2: Example of a fully connected ANN architecture

2.2 Neural Network Architecture

We use a fully connected, feedforward ANN architecture with a single hidden layer of 15 nodes for all results from this study. This architecture provides a computationally efficient method able to capture desired nonlinearities within domain data, without unnecessarily complicating the training and hyper-parameter selection process. Figure 3 shows an example architecture explored in this paper, with 3x3 input and 6x6 output patches. Inputs to and outputs from our ANNs are patches of a grayscale image representing an optimized topology domain (discussed further in Section 2.3). Input and output vectors therefore represent density values within a given portion of the domain and take values between zero and one. As such, we use sigmoid activation functions at output nodes to restrict outputs to be in the range of zero and one. However, the sigmoid activation at the output produces 0.5 with an input of zero, making it difficult for the network to learn to produce an output of exactly zero for void inputs. We therefore employ a hyperbolic tangent filter to map output values of the ANN so that void input elements produce a strictly zero output, as follows,

$$\rho_{new} = \frac{e^{\rho} - e^{-\rho}}{e^{\rho} + e^{-\rho}} \tag{5}$$

where ρ is the ANN output and ρ_{new} represents final predicted density value. We vary input-output patch sizes and hidden layer activation functions during this study. We consider input patches ranging from 2x2 to 5x5 (with corresponding output patches ranging from 4x4 to 10x10). We also consider sigmoid and tansig activation functions for hidden nodes, where all nodes within a layer have the same activation function.

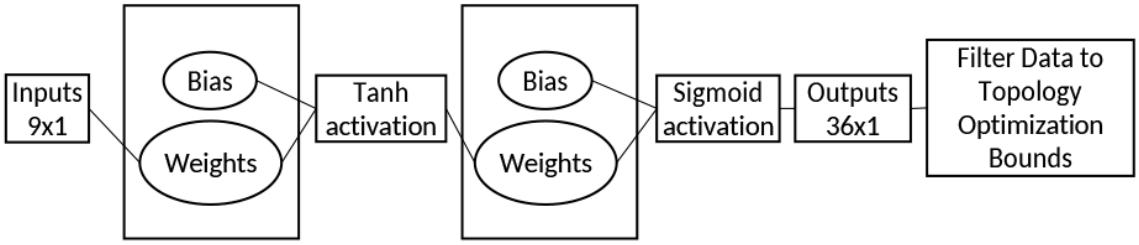


Figure 3: Example network architecture explored, with tansig hidden layer activations and 3x3 input and 6x6 output patches

2.3 Network Training

To support supervised training of our networks, we generate a data set of domain image pairs, such that each low-resolution input image has a corresponding high-resolution target image. We use Sigmund's minimum compliance Matlab code [30] to generate 20 different minimum compliance domains used within the training data set after appropriate post-processing [31]. Figure 4 shows the geometry and boundary conditions for the topology optimization design domain used for generating all structures used in the training process.

Figure 4: Topology optimization design domain

The topology optimization for minimization of compliance can be described as follows,

$$\begin{aligned} \min_{x}:\ & \mathrm{C(x)} = \mathrm{U^T KU} = \textstyle\sum_{e=1}^{D} (x_e)^p u_e^T k_0 u_e \\ & subject\ to: \frac{V(x)}{V_0} = f \\ & : KU = F \\ & : 0 < x_{min} \leq x \leq 1 \end{aligned} \tag{6}$$

where U and F are the global displacement and force vectors, respectively, K is the global stiffness matrix, u_e and k_e are the element displacement and stiffness matrices, respectively, x is the vector of design variables, x_{min} is the vector of minimum relative densities, D is the number of elements used to discretize the design domain, p is the penalization power, $V(x)$ and V_0 are the material volume and design domain volume, respectively, and f is the prescribed volume fraction.

We use an original optimized domain output as the high-resolution target within a training pair. We then produce a corresponding low-resolution domain by pairwise averaging adjacent rows and columns. Figure 5 shows a target and input domain after the rows and columns have been averaged.

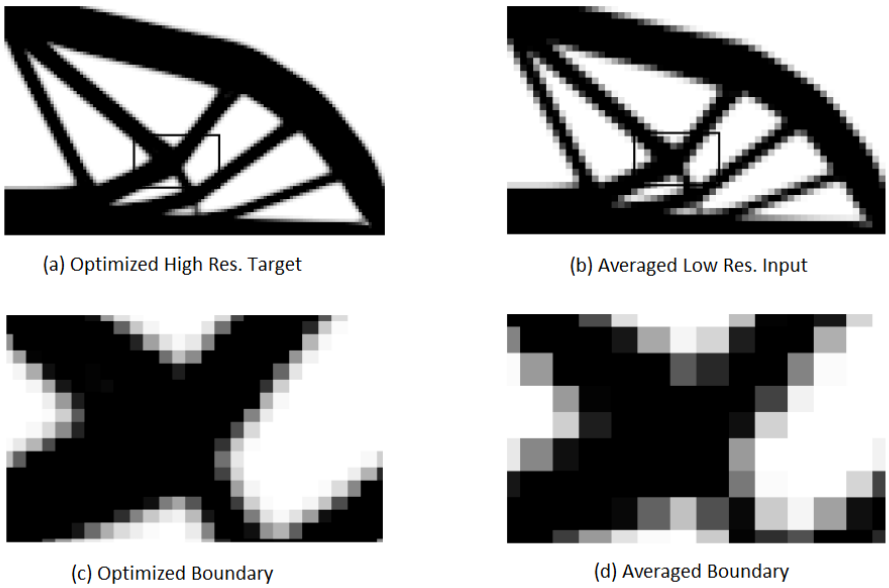
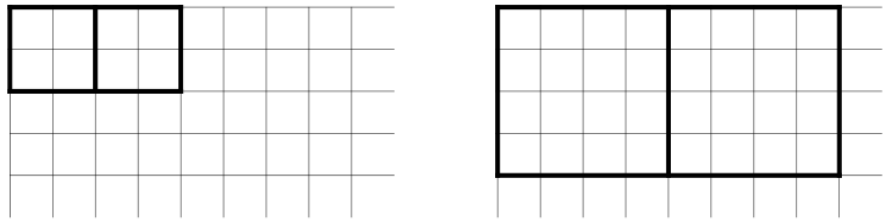


Figure 5: Optimized structure before and after averaging

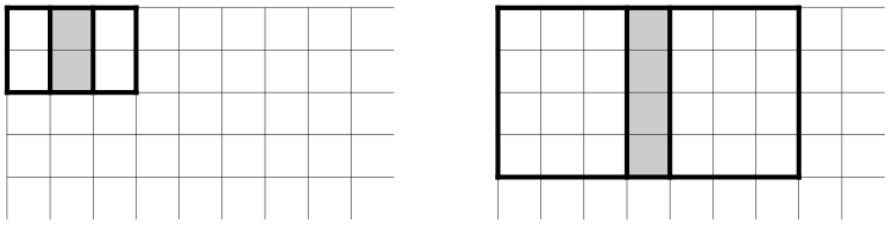
All domains are then partitioned into smaller patches to generate a data set of input-target pairs large enough for training the network. Two approaches are considered for generating domain patches: an independent approach and an overlapping one. Each approach is similar to how a CNN utilizes kernel filters to generate feature maps. An initial kernel size is selected to determine an observation area of the structure (e.g., 2x2 pixels). A stride is then selected to determine how far the kernel moves to select the next observation. Both approaches begin by partitioning the low-resolution domain within a training pair. The independent approach starts with a defined kernel or patch size (we use kernel sizes ranging from 2x2 to 5x5 pixels for different models) and defines the initial starting location to be the upper left of the domain. Additional patches are then defined by shifting the kernel by the selected stride such that each element belongs to at most one patch. The corresponding high-resolution target domain is then partitioned into patches of twice the resolution of the inputs (i.e., if the input kernel is 2x2 pixels, then the target kernel is 4x4) in a similar manner. Figure 6a shows the first two observations generated from a notional domain using the independent partition approach with a 2x2 kernel and stride of two.

The overlapping approach follows a similar pattern to that of the independent approach, except we decrease the stride such that generated patches contain overlapping portions of the original domain. This approach allows for more neighboring element information to be passed

into the network between input observations, enabling smoother transitions between patches. Although this method allows for more information to be passed between observations, reassembly of the output requires additional processing. When patches overlap during reassembly, values of elements contained by multiple patches are calculated as the average over their values within those patches. All overlapping results presented in this study only allow one column of overlapped elements between patches; i.e., all considered results with the overlapping approach use a stride length of one less than the kernel size along a given axis. Figure 6b shows the first two observations generated from a notional domain using the overlapping partition approach with a 2x2 kernel and stride of one.



(a) Independent input (left) and target (right) domain patch generation



(b) Overlapping input (left) and target (right) domain patch generation

Figure 6: Data set generation for input and target observations of both independent and overlapping approaches, with overlapped elements highlighted in grey

These low-resolution/high-resolution patch pairs are then used to train the network, where low-resolution patches are used as inputs and high-resolution patches as targets. An "observation" then refers to a single input-target patch pair. We partition the data such that 80% of observations are used for training, 10% for validation, and 10% for testing with unseen data. The training set is used to optimize network weights, while the validation set is used to prevent model overfitting by providing a separate data set to calculate errors used for stopping criteria in network training. The test set is composed of data unseen by the network during training, and therefore used to assess overall model performance. Observations are randomly assigned to one of the three data sets. We also looked at a data split of 60%-20%-20% but the results showed minor reduction in performance (see Section 3.1) and therefore all results presented will follow an 80%-10%-10% data split.

We use gradient descent to optimize weights in the network, specifically using the Levenberg-Marquardt (LM) algorithm. We choose the LM algorithm because it is a hybrid of the Gauss-Newton and gradient descent algorithms, utilizing steepest descent when further away from a minimum and switching to a second order Gauss-Newton approximation when nearing a minimum. The LM algorithm uses the following approximation for updating a minimum where $\boldsymbol{J}$ is the Jacobian, μ is the damping coefficient between the Gauss-Newton and gradient descent methods, and $\boldsymbol{e}$ is the network error,

$$w_{ji}^{l\prime} = w_{ji}^{l} - [\boldsymbol{J}^T\boldsymbol{J} + \mu\boldsymbol{I}]^{-1}\boldsymbol{J}^T\boldsymbol{e} \tag{7}$$

Figure 7 describes the overall training process, including an iterative loop involving the training data set. The training data set was generated from a set of 20 structures, seen in Figure 8. These 20 structures were created by varying the mesh resolution and aspect ratio used in the topology optimization algorithm. This enabled us to generate a variety of structures that differed in both their topology and mesh resolution. In this way, the network was trained to handle various input domain resolutions, rather than being trained specifically for a single resolution.

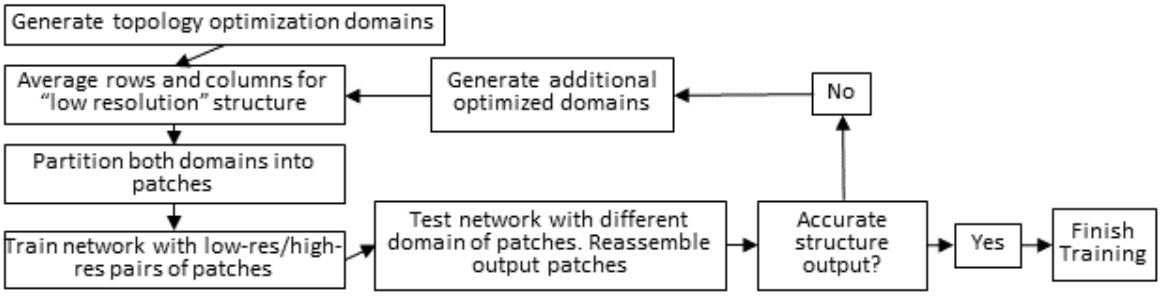


Figure 7: Overview of the network training process

Figure 8: Optimized structures used to generate all training data, with each structure's resolution shown below it

3. RESULTS

3.1 Network Error

Once a network has been trained, the question of model adequacy is of prime concern. For simplicity, network architectures were held constant between cases with only a change in activation functions. Based on the cost function shown in equation 3, the average observation MSE of a network can be calculated as

$$MSE = \frac{1}{B}\sum_j^B\left(\frac{1}{N}\sum_i^N\left\|\hat{y}_{ij} - y_{ij}\right\|^2\right) \tag{7}$$

where *N* is the number of output nodes, and *B* is the number of total observations. Note that this MSE only characterizes the average error in predicting individual observations or patches, not the total error in a high-resolution topology generated by reassembling output patches.

Figure 9 displays average observation MSE for training and test data, using architectures with various input patch sizes, data generated using independent and overlapping approaches, and hyperbolic tangent and sigmoid activation functions for hidden layer nodes. As discussed in Section 2.2, a sigmoid function was used for output layer nodes in all cases. The average observation MSE for training and test cases are relatively low, suggesting that valid models are produced and the data is not being overfitted. Comparing models trained on data generated using independent versus overlapping approaches, overlapping cases generally produce lower average observation MSE than corresponding independent cases. The benefits of the overlapping approach over the independent one become particularly pronounced as patch size is increased. Training errors increase with patch size for models trained using the independent approach; test errors generally increase with patch size as well. The opposite trend is seen for models trained using the overlapping approach; here, training and test errors generally decrease as patch size increases. These trends are likely due to the increase in information of neighbor cells provided by averaging of overlapping patches in the overlapping approach. Overall, the activation function of the hidden layer does not notably influence trends in model errors, suggesting that network weights are able to handle major discrepancies between the considered activation functions. Therefore, the remaining results will follow a 4x4 input patch size using tanh hidden layer activation due to its lowest test MSE per observation, unless noted otherwise.

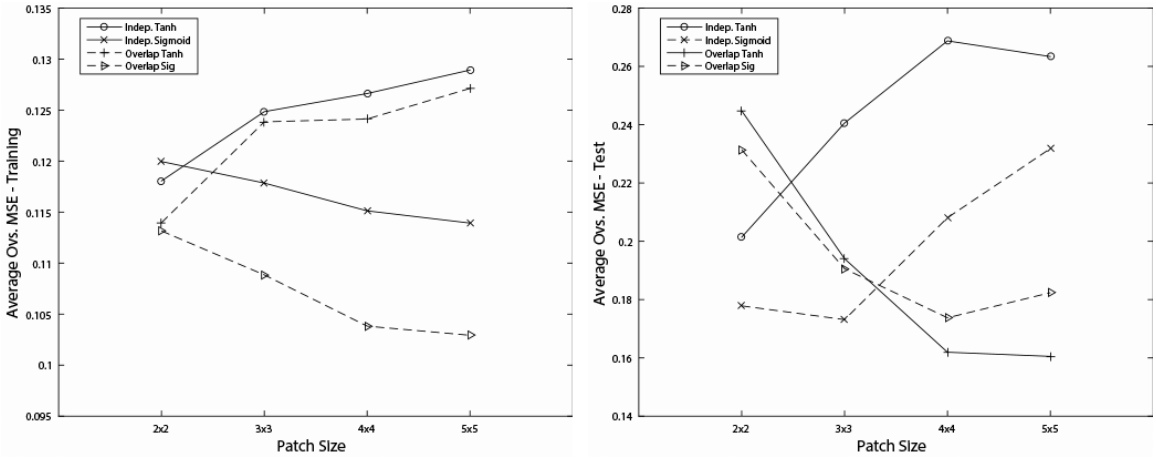


Figure 9: Average observed MSE versus patch size using various network models for training (left) and test (right) data

Table 1 shows percent errors for models trained using a 60%-20%-20% data split, rather than a 80%-10%-10% split. The errors are calculated similarly to Figure 9, using the MSE per input patch compared to the desired target over the whole training, validation, and test sets. Network activations were tanh for hidden layers, with training data generated by the overlapping approach. These results show higher test errors for 60%-20%-20% cases with larger patch sizes than corresponding 80%-10%-10% cases. However, test errors are lower for 60%-20%-20% cases with smaller patch sizes than corresponding 80%-10%-10%. Since networks trained with larger patch sizes generally provide lower errors than those trained with smaller patches sizes, we use the 80%-10%-10% split for remaining results.

Table 1: Varying input domain error for 60%-20%-20% data splits for overlapping cases

Input Patch Size	Training Error	Validation Error	Testing Error
2x2	0.1186	0.1257	0.1696
3x3	0.1233	0.1413	0.1952
4x4	0.1278	0.1592	0.2611
5x5	0.1301	0.1790	0.2936

3.2 Compliance Analysis of Network Output

Although our ANN models produce adequate MSE results for training and test observations, structural integrity must also be verified. We tested the structural integrity of network outputs by first reassembling output patches to form a complete structure (e.g., reassembling patches to form one of the training structures shown in Figure 8). We then performed finite element analysis on the reassembled high-resolution output structure and compared that analysis to one performed on the original high-resolution optimized structure (i.e., the target structure). We specifically analyzed structural compliance in these comparisons, as structural compliance is the actual objective of the topology optimization problem. The structural compliance (*C*) is proportional to the internal strain energy of the structure, and by minimizing this function, we effectively maximize the structure's stiffness with respect to a given load, *F*. The compliance is evaluated using the following equation,

$$C = U^T K U \tag{8}$$

where

$$U = K^{-1} F \tag{9}$$

and *U* is the global displacement vector, *F* is global consistent force vector, and *K* is the global stiffness matrix used in the finite element analysis.

As the resolution is refined, the compliance of the structure will change due to an increase or decrease in volume. The following formula is used to calculate the total volume of a structure before and after being passed into the ANN,

$$V = \sum_i \rho_i A_i \tag{10}$$

where *V* is the structure's overall mass (per unit thickness), ρ_i is the density corresponding to element *i, and* A_i is the area corresponding to element *i*. Note that our mesh domains are uniform and they exclusively contain unit square elements, so the area of each element is simplified to one.

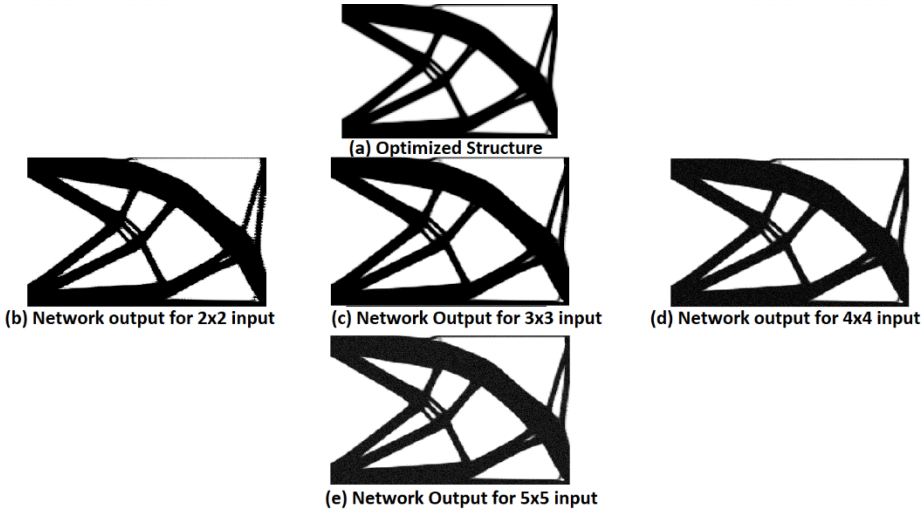


Figure 10: Optimized structure compared to network outputs for various patch sizes

Figure 10 shows a comparison of a target high-resolution optimized test structure (from Figure 11a) to high-resolution reassembled outputs from networks trained for various patch sizes. Table 2 shows corresponding compliance and volume errors for these reassembled structures. Compliance and volume errors are calculated relative to the target optimized high-resolution structure. These results show that the networks slightly change output structures relative to their target; this change can be observed in the elastic response of the output structures. However, this error is relatively modest and is similar in magnitude to the difference in compliance that we would have observed if we were to, for instance, double the filter radius. These results also indicate that the size of the patches used in the training data impacts the fidelity of output structures, with larger patches leading to network output topologies that have slightly less error. This outcome follows the trend of network model error decreasing as patch size increases, seen in Section 3.1.

Table 2: Compliance and Volume Comparison of Network Outputs versus Optimized High-resolution Structure

Patch Size	**2x2**	**3x3**	**4x4**	**5x5**
Compliance Error	11.97%	11.84%	11.87%	11.49%
Volume Error	8.44%	7.91%	7.72%	7.05%

One possible source of error in our network output topologies is that low-resolution inputs in the training data set are generated by averaging target high-resolution optimized structures. While this averaging process captures the overall characteristics of the target structure, it may be introducing errors that are then propagated through network models and into high-resolution reassembled output structures. Table 3 shows compliance and volume errors for averaged input structures compared to their high-resolution optimized target structures, for four of the structures used within the training data. Specifically, these structures correspond to the four structures in the left most column of Figure 8 going from top to bottom. Table 3 also shows minimum, maximum, and average errors calculated for all 20 structures used in the training data set. These results show that while some of the errors observed in network output structures

may come from the models themselves, their low-resolution inputs are likely contributing to observed errors as well.

Table 3: Compliance and Volume Errors for Averaged Input Structures versus Optimized Target Structures.

	Compliance Error	Volume Error
Training 1	4.63%	3.42%
Training 2	6.81%	4.52%
Training 3	6.13%	5.73%
Training 4	5.34%	4.60%
Minimum	3.53%	2.96%
Maximum	11.94%	9.89%
Average	7.05%	6.50%

3.3 Testing Networks Outside of Training Data

We also test network models with six additional topology domains generated to be significantly different than the 20 training cases. These additional domains were kept outside of the original data set, such that they are unseen during model training. High-resolution target structures for the six additional test cases are presented in Figure 11.

Figure 12 displays the target high-resolution topology from Figure 11b, along with its corresponding averaged network input topology and network output topology. Based on visual inspection, the main features of this structure are preserved between the optimized topology and network output; however, there are some differences between the optimized and network output topologies, as the network topology becomes thinner than the original in some regions. Figure 12d-f show close-up views of the interface region where the domain transitions from solid to void, demonstrating the ability of the network to improve the resolution of most domain boundaries. Thus, while there are some errors associated with predicting thin members of the structure, the ability of the network to accurately improve resolution in other areas of the structure suggests this approach has potential value for producing accurate high-resolution topologies. We also note that this structure was generated to be notably different than those used within the training process, and thus represents a worst-case scenario for this approach.

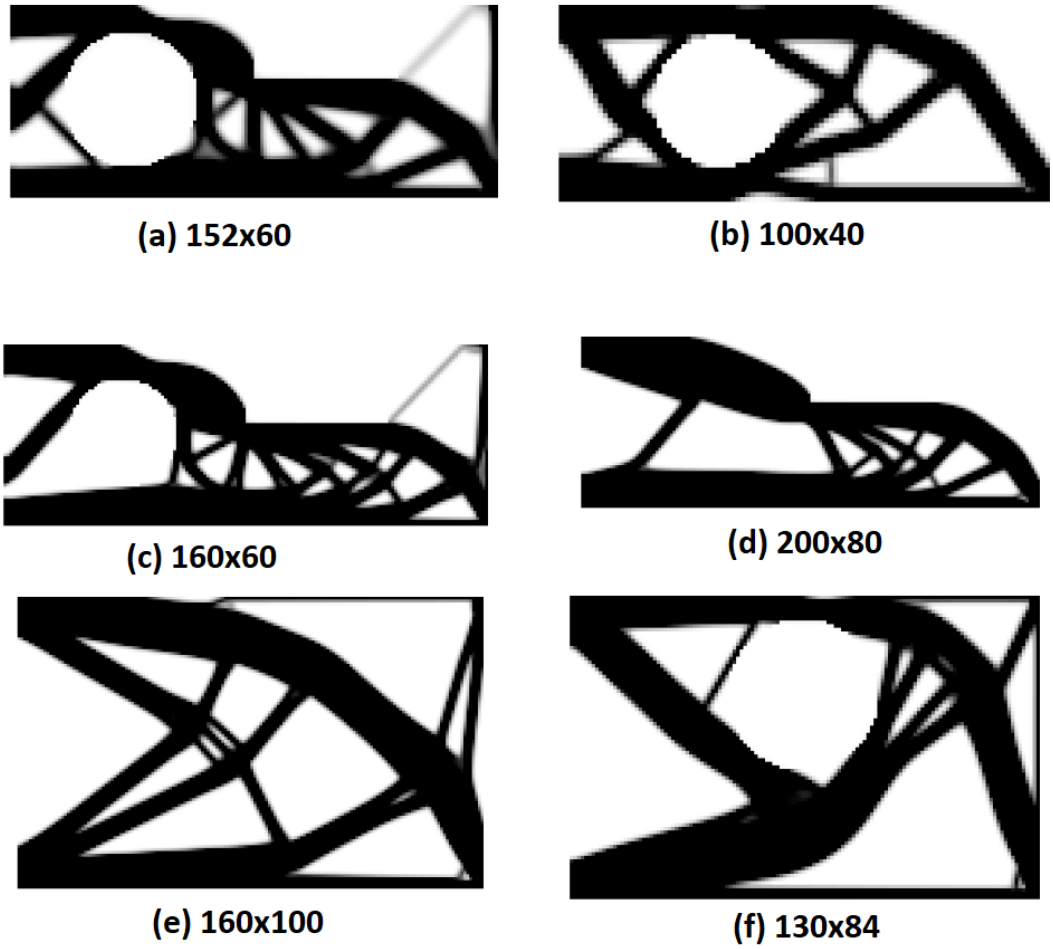


Figure 11: Six additional test structures, with each structure's resolution shown below it

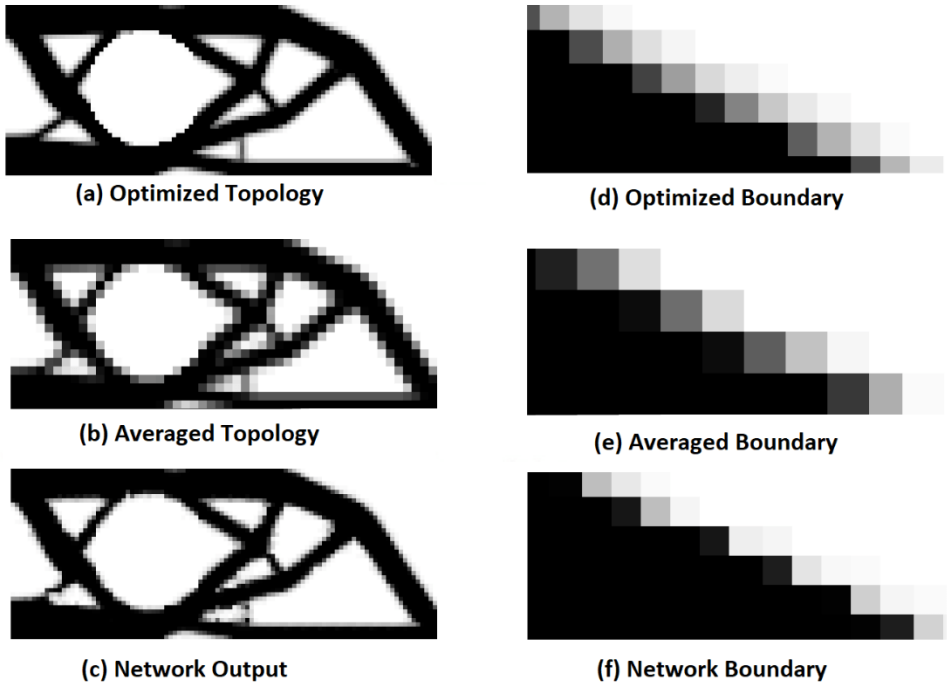


Figure 12: Example results for an additional test structure, generated to be significantly different from those included in the training data

We also consider using the high-resolution optimized topology as an input to be passed through a trained network, with the goal of producing an even higher-resolution output. This approach bypasses the input of an averaged domain into the network; note that this network has already been trained using averaged domains as discussed in Section 2.3. Figure 13 shows outputs from passing two of the additional test structures through a trained network to produce an even higher-resolution output. As expected, these results show that passing the original optimized topology through the network produces an even higher-resolution topology with more clearly defined domain boundaries. Due to the presence of the sensitivity filter [30], all optimized designs contain a narrow band of intermediate density elements, as the transition from solid to void must occur gradually across several elements. This feature is undesirable, but is necessary to avoid checkerboarding and mesh-dependence [31]. While there exist several methods for avoiding the fuzzy boundaries associated with density filtering (i.e. level set methods [12] and Heaviside filtering [7]), these methods increase the complexity and computational cost of the topology optimization algorithm. By applying the network model, we are able to substantially reduce the prevalence of the transition layer, producing a more well-defined material boundary.

3.4 Recursive Application

Users of this approach may also wish to further increase the resolution of output topologies by applying a network recursively. Here, we explore the impact of applying a network multiple times on a single input topology. Two cases are considered, both involving trusses similar to those used in the training set. These include a low-resolution input (30x15) and a high-resolution input (144x80), each of which is passed through a network twice. Figure 14 shows the 30x15 pixel structure being increased to 60x30 pixels then to 120x60, along with close-up images of the corresponding boundary edge transition for all outputs. Figure 15 follows the same process starting with the 144x80 pixel structure.

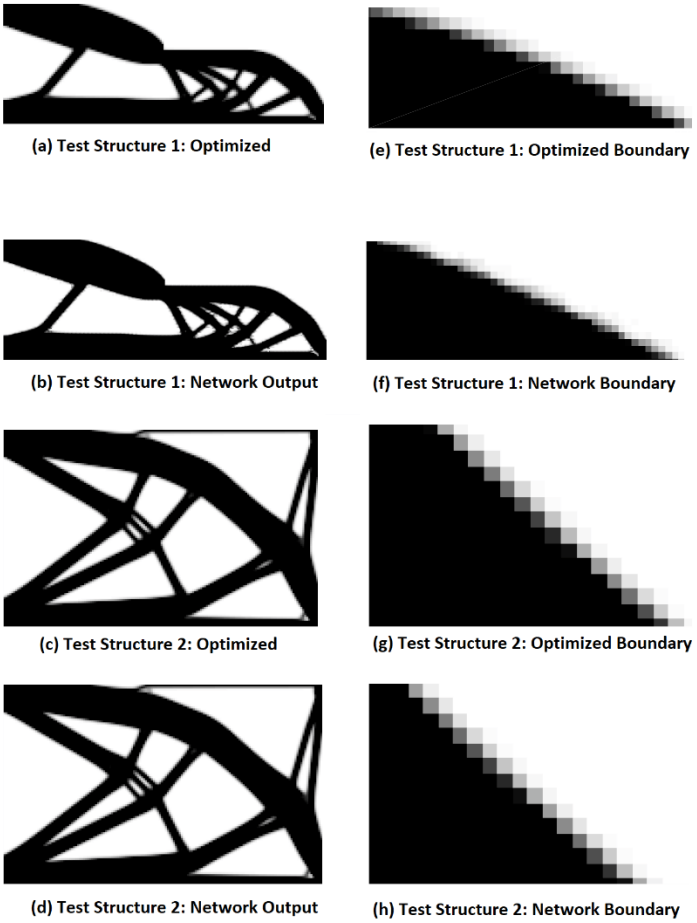


Figure 13: Optimized test structures versus network output

From Figure 14, the low-resolution structure starts producing structural errors within the first pass through the network. These errors are then propagated forward to the second output. One possible explanation for this result is that the low-resolution input does not have very distinguished features, thus leading the network model to struggle with interpolating a mid-density value into a whole feature. In comparison, Figure 15 shows that repeating this recursive process with the high-resolution structure produces notably better results. The overall characteristic features of the output structure are nearly identical to the input due to its well-defined features, thus enabling resolution increases and improved boundary definition with minimal structural errors. Note that when recursively applying the network, the output must be re-adjusted following each recursion using the hyperbolic tangent filter described in Section 2.2.

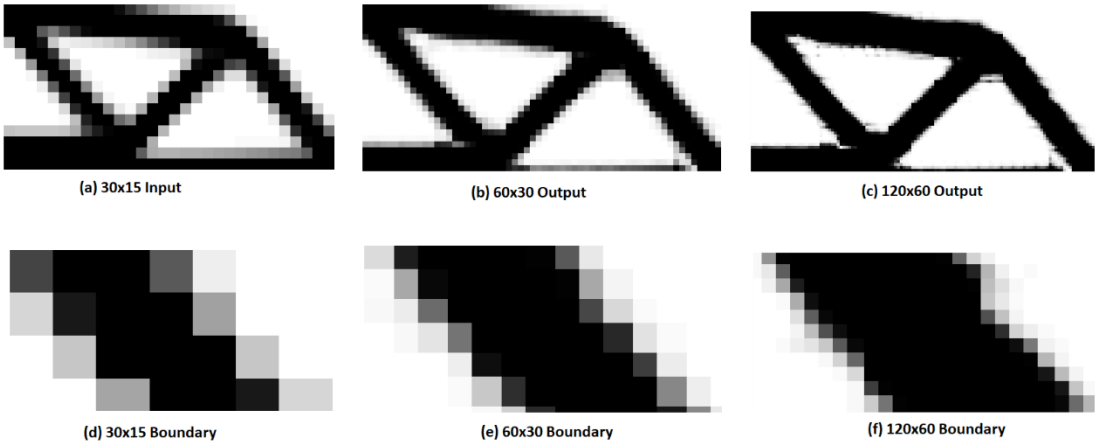


Figure 14: Recursive application for low-resolution optimized structure

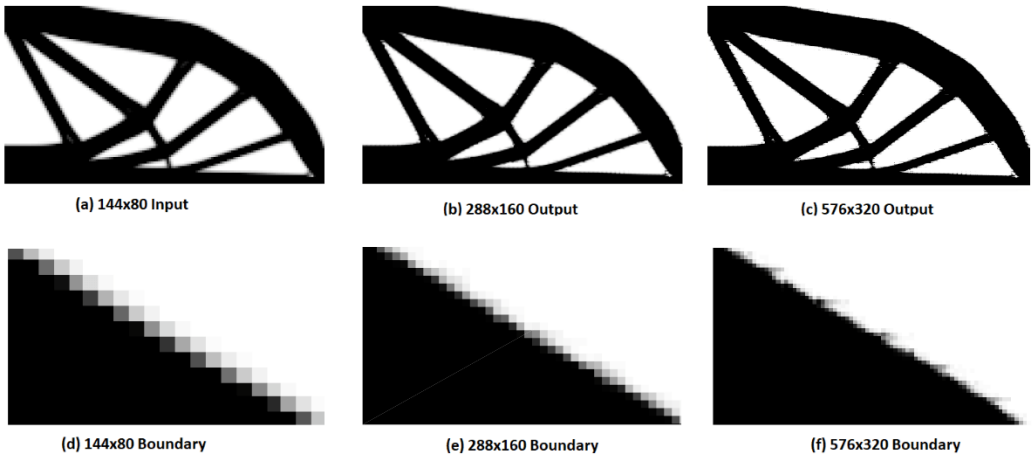


Figure 15: Recursive application for high-resolution optimized structure

3.5 Coupled, Optimized Training Pairs: An Alternative Approach

In the prior results, the low-resolution input topologies for training data were generated by *averaging* optimized high-resolution topologies. This approach was chosen based on the hypothesis that averaging was an efficient way to obtain low-resolution/high-resolution observation pairs that are very similar in their overall topology. As an alternative approach, we can generate our low-resolution/high-resolution pairs by performing two separate topology optimizations; one with a high-resolution mesh and the other with a low-resolution mesh that has half as many elements in each direction as the high-resolution mesh. We can ensure the similarity of the two designs by applying consistent boundary conditions and maintaining a constant filter radius. We choose a filter radius that spans 1.25 elements for the low-resolution mesh, and a filter radius that spans 2.5 elements for the high-resolution mesh. In this way, the filter radius, when measured in meters, is constant across both optimizations, and the resulting topologies are mesh-independent. We refer to this approach for generating observation pairs as the *coupled* approach.

Figure 16 shows outputs of networks trained using coupled and averaged approaches for generating observation pairs, for one of the additional test structures. These results show that output of the coupled network produces more checkered patterning around the boundary in void regions, as well as more feature loss in areas containing thin low-density connections relative to

the averaged network. These characteristics indicate that significant differences exist between the optimized low-resolution and high-resolution topologies used as inputs and targets in the coupled training set. These results suggest that the averaging approach is a more effective, as well as computationally efficient, approach for generating low-resolution/high-resolution pairs for training neural networks.

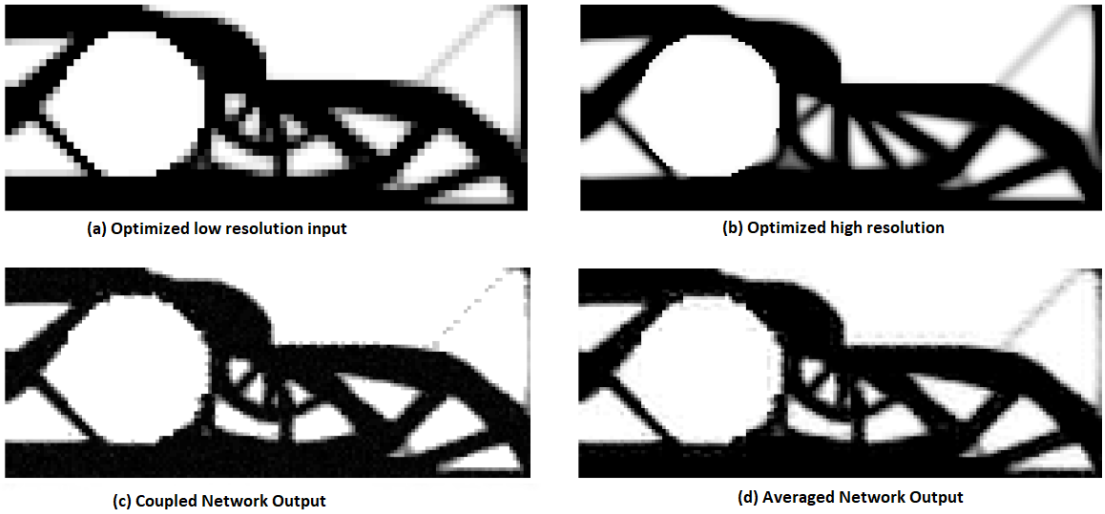


Figure 16: Comparison of network outputs using coupled versus averaged low-resolution inputs

3.6 Computational Savings

One feature motivating the use of our proposed neural network approach is expected computational time savings for generating a high-resolution structure. To quantify time savings, we add the overall time taken to generate an optimized low-resolution structure to the time taken to pass this design through a pre-trained network. The resulting computational time is compared to the convergence time of a high-resolution topology optimization. Table 4 compares total time in seconds to execute each method for the 160x100 (used as a target) and 160x60 (used as an input) resolution structures shown in Figure 11. These computations were made with a machine using an Intel i7-5930k CPU at 3.50GHz and 16GB of ram.

Table 4: Computation Time for ANN Process versus High-Resolution Optimization

	Optimization Time (s)	**Network Time (s)**	**Total Time (s)**
ANN Process (80x50 to 160x100)	17.28 (80x50)	5.31	22.59
Optimization (160x100)	591.72 (160x100)	-	591.72
ANN Process (160x60 to 320x120)	536.29 (160x60)	15.62	554.91
Optimization (320x120)	3912.62 (320x120)	-	3912.62

As the domain sizes increase, the computation time of the high-resolution topology optimization becomes significantly longer than that of the low-resolution, with the neural network operation contributing a small fraction to the total computation time in each case. For example, generating the high-resolution structure with a domain size of 320x120 elements took approximately 7.3 times longer than generating a 320x120 element structure using the ANN process. Note that these computational advantages are only realized after a network has been

trained. The required time to generate the data for network training, along with the training process itself, was not taken into consideration since training can be carried out once. The resulting network can then be used many times to refine any number of low-resolution designs. Overall, the estimated time to generate and average the 20 optimized domains used for training data is about 72 hours for domains ranging from 60x20 to 200x140 elements. Also note that little effort was given to improving code efficiency for generating training data; further research into reducing training time would likely reduce this overhead cost of the proposed approach.

We again note that our proposed approach treats the problem of achieving high-resolution designs as one of image super-resolution. That is, we propose integration of an image processing technique within the overall topology optimization scheme. We therefore compare our ANN approach with bilinear, bicubic, and nearest neighbor interpolation schemes (implemented using Matlab's imresize function), which could alternatively be used to increase the resolution of optimized domains. Figure 17 shows general scaling trends for algorithm run time as input image resolution increases, with corresponding regression model fits. The equations for the regression fits are given in Table 5. The resolution of the input topology was increased by repeatedly doubling the resolution of the original, using the algorithm being analyzed to perform the resolution increase at each refinement level. The ANN used a 3x3 input patch size with independently defined patches, with ANN run time calculated as the total time required to pass through a pre-trained model and complete necessary post-processing. Results for the interpolation schemes are not shown for highly refined resolutions because imresize failed to compute due to internal Matlab memory limitations; our ANN model was also implemented in Matlab for this study.

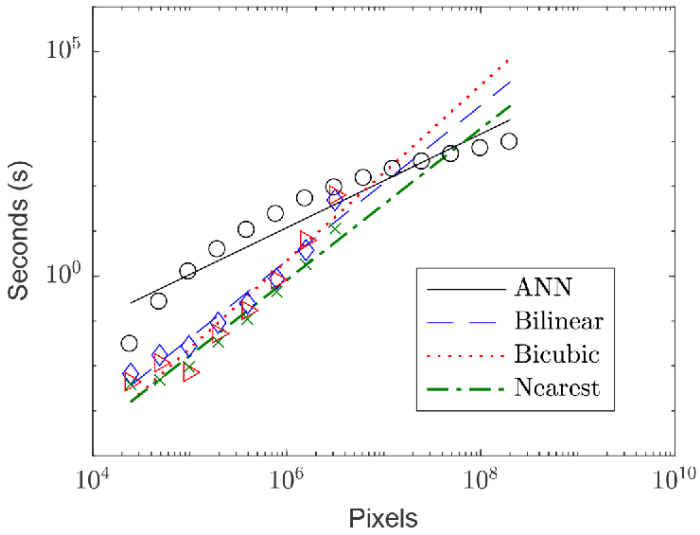


Figure 17: Scaling comparison of algorithm run time vs. image resolution for considered algorithms using "Thermal test structure 1." The markers represent observed data, with the lines showing a least-squares linear regression fit to log-transformed data.

Figure 17 suggests that while our ANN run times may be worse for smaller images, run times scale more desirably for higher resolution images. In fact, we see that the ANN algorithm scales nearly linearly ($n \approx 1$) with the number of pixels in the input image. By contrast, for all three interpolation schemes, the approximate order of complexity ($n > 1.68$) is significantly higher than that of the ANN algorithm. Furthermore, these results show that the trend for ANN run time is concave down whereas the trend for the interpolation run times is concave up, suggesting that the ANN algorithm may offer better than linear scaling for large images, whereas interpolation scaling would be quadratic or worse. However, we note that the coefficient of the regression fit,

C, is many orders of magnitude higher for the ANN algorithm, suggesting that the imresize function will be faster for images where the number of pixels, x, is less than 10^7. Table 6 compares errors in output structures for the ANN and imresize interpolation schemes; here, we see that the imresize function outperforms our ANN approach for preserving the image volume. These results suggest that, for smaller images, existing interpolation schemes have advantages over our ANN algorithm with respect to execution time and image accuracy. However, our algorithm run time scales nearly linearly with the size of the image being refined and does not reach memory limitations for the resolutions considered in this study. Furthermore, we have not attempted to optimize our algorithm implementation for run time and have conducted only limited exploration of alternative network architectures within this study. Further research into our proposed approach may address many of these issues.

Table 5: Linear regression fits to observed run-times, where x is the number of pixels and T is the run time (in seconds).

	$T = Cx^n$	
	n	*C*
ANN	1.044	6.58e-6
Bilinear	1.957	3.98e-12
Bicubic	1.730	9.12e-11
Nearest Neighbor	1.687	6.15e-11

Table 6: Percent errors in volume of the structure normalized over the total elements in the finite element mesh.

	ANN 2x2	ANN 3x3	Bilinear	Bicubic	Nearest Neighbor
Test Structure 1	1.13%	4.49%	0.09%	0.13%	0.05%
Test Structure 2	2.86%	6.94%	0.11%	0.06%	0.05%

3.7 Network Generalization

The results presented in previous sections all involve elastic structures optimized for minimum compliance. Here we investigate how well a trained network will perform when used to refine structures designed for other physics phenomena. In the test cases that follow, we seek to refine structures that have been optimized for maximum heat dissipation. Figure 18 shows the geometry and boundary conditions used in the optimization problem. The structure is subject to a constant thermal load (i.e. fixed-temperature, $\bar{T}$=100°C, Dirichlet boundary condition) at the center of the bottom surface. There is also an outward heat flux, q = 50W/m^2, applied to the remaining portion of the structure's perimeter. The domain has dimensions of 12m x 2m, with a thickness of 1m in the z dimension. The domain is discretized into a uniform grid containing 6144 square elements. The thermal structures are optimized for maximum average temperature throughout the material domain, based on a linear, steady-state, thermal conductivity analysis [33], and subject to a constraint on the material volume. We test the neural network using two different structures, both optimized using the same boundary conditions, but containing different materials with different thermal conductivity values, and constrained to different volume fractions. Table 7 shows the optimization problem specifications for each test structure.

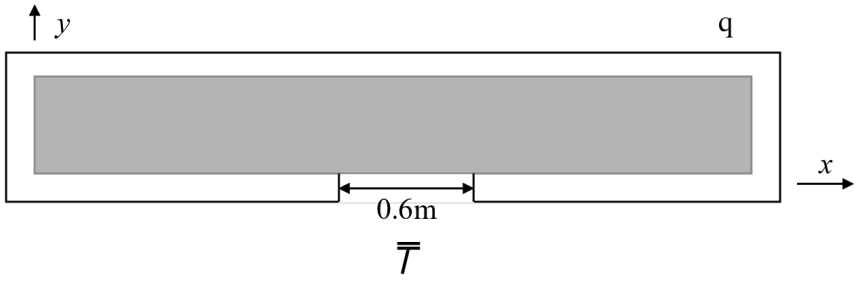


Figure 18: Geometry and boundary conditions for the thermal optimization problem

Table 7: Optimization problem specifications for the thermally optimized structures

	Material	Thermal Conductivity	Volume Fraction
Thermal Structure 1	BC	105 K/(W°C)	0.3
Thermal Structure 2	SAC305	59 K/(W°C)	0.55

To test the generalizability of the ANN-based refinement approach, both of the thermally optimized structures were refined using a single forward pass through a neural network already trained using structures designed for elastic stiffness. Figures 19 and 20 show the original optimized structures and the corresponding output structure after ANN-based refinement. We see that the algorithm performs similarly to the prior test results with minor errors. This comparison shows the generalizability of the method and indicates that, once the network is trained, it can be used to perform refinement on a variety of structures. To quantify the error introduced by the refinement process, we compute the total volume fraction (described by Eq. 10 but normalized by the number of elements) of each refined structure, and compare this number with the volume fraction of the original (input) structure. The results are shown in Table 8. The values shown in Table 8 are similar to the volume changes observed in Table 3, thus further supporting the hypothesis that the ANN approach is largely agnostic to the physics used in the optimization problem.

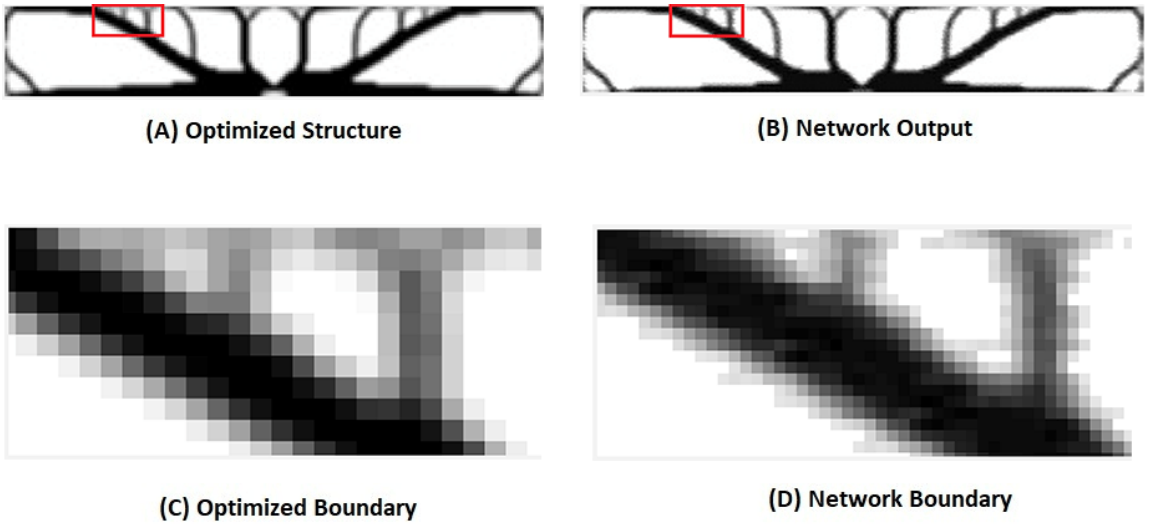


Figure 19: Thermal test structure 1

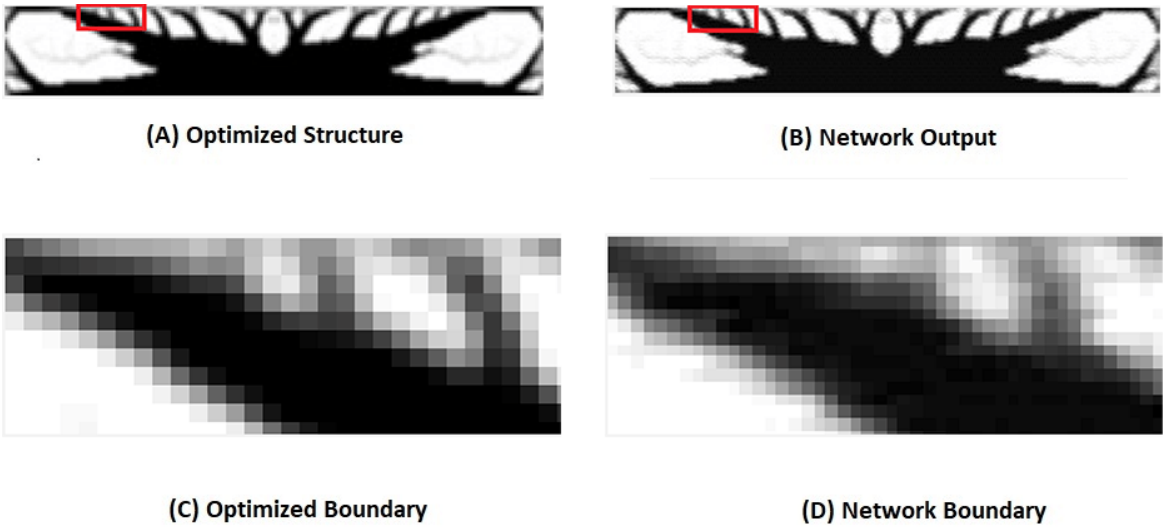


Figure 20: Thermal test structure 2

Table 8: Volume of thermally loaded optimized structures

	Thermal Structure 1	Thermal Structure 2
Original Volume Fraction	0.2998	0.5486
ANN Output	0.3115	0.5902
Percent Change	3.90%	7.58%

4 CONCLUSION

This study presents a new approach for achieving high-resolution topology optimization designs by training an ANN on a small set of optimized structures that were further divided into small patches to represent an observation. Overall, ANNs were able to significantly reduce the

computation time of a typical high mesh optimization procedure, while also improving upon the boundary transition region of the structure. This process can allow for a quicker and easier manufacturing and prototype testing process for topology optimization-derived structures with only minor losses in design precision.

Further application of this study showed that it is possible to iteratively apply the ANN process to an optimized structure, though this process has the drawback of needing a sufficiently optimized starting structure. As the ANN is applied iteratively, further error will be introduced into the model and resolution issues arise if the desired resolution output is much larger than the resolutions seen in training. Considering the alternative approach of coupling optimized low-resolution and high-resolution structure pairs, there was no real performance advantage. Since these results are focused on reducing computation time, the averaging approach will perform adequately on the resolutions presented in this study. Building upon this study, future investigation will explore the application of this method to multi-material topology optimization and three-dimensional structures.

REFERENCES

[1] Bendsøe, M. P., and Kikuchi, N., "Generating optimal topologies in structural design using a homogenization method," *Computer Methods in Applied Mechanics and Engineerin*g, vol. 71, 1988, pp. 197–224.

[2] Nanthakumar, S., Lahmer, T., Zhuang, X., Park, H. S., and Rabczuk, T., "Topology optimization of piezoelectric nanostructures," *Journal of the Mechanics and Physics of Solids*, vol. 94, 2016, pp. 316–335.

[3] Jensen, K. E., Szabo, P., and Okkels, F., "Optimization of bistable viscoelastic systems," *Structural and Multidisciplinary Optimization*, vol. 49, 2013, pp. 733–742.

[4] Kutylowski, R., and Rasiak, B., "Influence of various design parameters on the quality of optimal shape design in topology optimization analysis," *Pamm*, vol. 8, 2008, pp. 10797–10798.

[5] Sigmund, O., and Peterson, J., "Numerical instabilities in topology optimization: A survey on procedures dealing with checkerboards, mesh-dependencies and local minima," *Structural Optimization*, vol. 16, 1998, pp. 68–75.

[6] Le, C., Norato, J., Bruns, T., Ha, C., and Tortorelli, D., "Stress-based topology optimization for continua," *Structural and Multidisciplinary Optimization*, vol. 41, Jun. 2009, pp. 605–620.

[7] Guest, J. K., Prévost, J. H., and Belytschko, T., "Achieving minimum length scale in topology optimization using nodal design variables and projection functions," *International Journal for Numerical Methods in Engineering*, vol. 61, 2004, pp. 238–254.

[8] James, K. A., and Waisman, H., "Failure mitigation in optimal topology design using a coupled nonlinear continuum damage model," *Computer Methods in Applied Mechanics and Engineering,* vol. 268, 2014, pp. 614–631.

[9] James, K. A., and Waisman, H., "Topology optimization of viscoelastic structures using a time-dependent adjoint method," *Computer Methods in Applied Mechanics and Engineering*, vol. 285, 2015, pp. 166–187.

[10] Wang, S., de Sturler, E., and Paulino, G. H., "Dynamic Adaptive Mesh Refinement for Topology Optimization," *ResearchGate*, Sep. 2010.

[11] "An adaptive refinement approach for topology optimization based on separated density field description," *Computers and Structures* Available: http://dl.acm.org/citation.cfm?id=2435891.

[12] Allaire, G. C. A., Jouve, F., and Toader, A.-M., "Structural optimization using sensitivity analysis and a level-set method," *Journal of Computational Physics*, vol. 194, 2004, pp. 363–393.

[13] James, K. A., and Martins, J. R., "An isoparametric approach to level set topology optimization using a body-fitted finite-element mesh," *Computers & Structures*, vol. 90-91, 2012, pp. 97–106.

[14] Zhou, H., and Mohammed, S. A., "The Boundary Smoothing in Discrete Topology Optimization of Structures," *Volume 3A: 39th Design Automation Conference*, Apr. 2013.

[15] Kim, J., Lee, J. K., and Lee, K. M., "Accurate Image Super-Resolution Using Very Deep Convolutional Networks," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2016.

[16] Dong, C., Loy, C. C., He, K., and Tang, X., "Image Super-Resolution Using Deep Convolutional Networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, 2016, pp. 295-307.

[17] Burel, G., Pottier, I., and Catros, J.-Y., "Recognition of handwritten digits by image processing and ANN," *[Proceedings 1992] IJCNN International Joint Conference on ANNs*.

[18] Karlik, B., and Sarioz, M., "Coloring gray-scale image using artificial ANNs," *2009 2nd International Conference on Adaptive Science & Technology (ICAST)*, 2009.

[19] Bishop, C. M., *Pattern recognition and machine learning*, New Delhi: Springer, 2013.

[20] Wang, G. G., and Shan, S., "Review of Metamodeling Techniques in Support of Engineering Design Optimization," Volume 1: 32nd Design Automation Conference, Parts A and B, 2006.

[21] Gorrisen, D., Couckuyt, I., Demeester, P., Dhaene, T., and Crombecq, K., "A Surrogate Modeling and Adaptive Sampling Toolbox for Computer Based Design," *Journal of Machine Learning Research*, vol. 11, 2010, pp. 2051-2055.

[22] Chojaczyk, A. A., Teixeira, A. P., Neves, L. C., Cardoso, J. B., and Soares, C. G., "Review and application of Artificial Neural Networks models in reliability analysis of steel structures," *Structural Safety*, vol. 52, 2015, pp. 78–89.

[23] Quan, W., and Pimentel, A. D., "Towards Exploring Vast MPSoC Mapping Design Spaces Using a Bias-Elitist Evolutionary Approach," *2014 17th Euromicro Conference on Digital System Design*, 2014.

[24] Yadav, V., Malik, P., Sahoo, G., and Chauhan, A. S., "Energy Efficient Virtual Machine Optimization," *International Journal of Computer Applications*, vol. 106, 2014, pp. 23-28.

[25] Zadpoor, A. A., "Open forward and inverse problems in theoretical modeling of bone tissue adaptation," *Journal of the Mechanical Behavior of Biomedical Materials*, vol. 27, 2013, pp. 249–261.

[26] Mohri, M., Rostamizadeh, A., and Talwalkar, A., *Foundations of machine learning*, Cambridge, MA: The MIT Press, 2012.

[27] Malsburg, C. V. D., "Frank Rosenblatt: Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms," *Brain Theory*, 1986, pp. 245–248.

[28] Nair, V. and Hinton, G. E., "Rectified Linear Units Improve Restricted Boltzmann Machines," *International Conference on Machine Learning*, 2010.

[29] Bengio, Y., Courville, A., and Vincent, P., "Representation Learning: A Review and New Perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, 2013, pp. 1798–1828.

[30] Sigmund, O., "A 99 line topology optimization code written in Matlab," *Structural and Multidisciplinary Optimization*, vol. 21, 2001, pp. 120–127.

[31] Pedersen, C., Lund, J. Damkilde, L., and Kristensen, Anders, "Topology Optimization – Improved checkerboard filtering with sharp contours", *Aalborg University*, 2006

[32] Krizhevsky, A., Sutskever, I., and Hinton, G. E., 2017, "ImageNet classification with deep convolutional neural networks," Communications of the ACM, **60**(6), pp. 84–90.

[33] Kang, Z., and James, K.A., "Multimaterial Topology Design for Optimal Elastic and Thermal Response with Materials-Specific Temperature Constraints", *International Journal for Numerical Methods in Engineering*, vol. 117(10), pp. 1019-1037, 2019.