CS230

Structural Topology Optimization using a Convolutional Neural Network

Anne L. Alter
Department of Mechanical Engineering
Stanford University
annealtr@stanford.edu

1 Introduction

Topology optimization is an under-utilized, yet powerful tool for design because it is able to form novel geometries for a given set of loads, frequencies, and constraints on a system. It works by taking input conditions (forces on the system and boundary conditions) and it fills in the optimal material density distribution as shown in Figure 1.

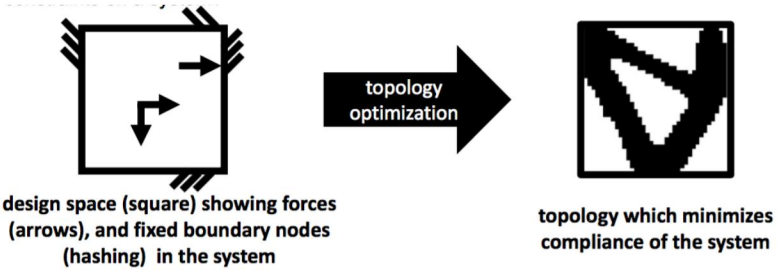

Figure 1: Topology optimization inputs and outputs.

Topology optimization starts with a rectangular design space discretized into n x n elements. Every optimization has a set of fixed elements, and loads acting on specific elements (ex. a bridge has two fixed ends with the force of gravity acting down on it's center in a simplified model). As the designs are optimized, each element gets filled in with a probability between 0 and 1 of being a material or void. There is an upper bound on the fraction of material in the design space that can be filled in, preventing the optimization from simply filling all of the elements in. At the end of the optimization each element is classified as either material (1) or void (0) with the overall compliance of the structure minimized for the given loads on the system.

While topology optimization can help form non-intuitive designs, it is computationally expensive and slow, which opens the door for other computational techniques like deep learning to help out the process. In this project, the inputs to the model are 2 channels: (1) the topology optimization that was stopped after just 5 iterations (it usually takes 100 to converge) and (2) the difference between the topology from iterations 4 and 5; both are shown at left in Figure 2. The output is the optimal topology of the system; in this study we use iteration 100 of the topology optimization from the dataset as the ground truth labels (discussed in more detail in Dataset section).

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

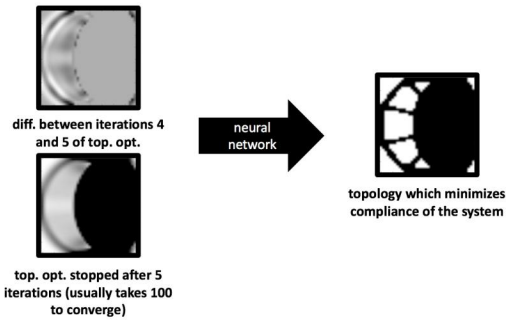

Figure 2: Neural network inputs and outputs.

To form the outputs from the given input channels this project tried out 2 different neural network architectures: (1) a shallow neural network and (2) a deep neural network that has the U-net architecture [1]. The deep neural network follows the implementation in [2] closely.

2 Related work

Overall, structural optimization and neural networks have not been studied together very much. The paper in [2], which this work follows closely, stops the structural topology optimization early and implements the U-net architecture [1] to predict the output topology. The strength to their model is that it very accurately predicts the topology of the system, however they don't discuss how to bridge the gap to remove the topology optimization process entirely and go straight from the forces and constraints on the system to forming the optimal topology. This paper is currently the most state-of-the-art and is unique in its application. In [3], they implement a unique model which smooths out optimized topologies to make them manufacturable; their CNN groups certain similar shapes and smooths the surfaces to make them all conform to a more manufacturable geometry. There are a few papers like [4,5] which are focused on predicting structural failure using a neural network, but these approaches rely heavily on additional simulation to predict the failures.

3 Dataset and Features

This project uses a dataset from [2], which contains 10,000 optimized topologies all at a resolution of 40x40 pixels, which were generated directly from [6]. Figure 3 below shows the input images after 5 iterations (at left) of the costly topology optimization and the ground truth labels (at right) after 100 iterations of topology optimization. The goal is to build a neural network to speed up the process of getting from iteration 1 to 100.

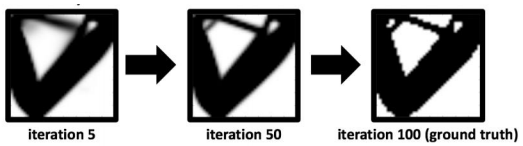

Figure 3: Each of the 10,000 datapoints contains iterations 1 through 100 from the topology optimization; iterations 5, 50, and 100 are shown here.

The dataset was split into 8,000 training set images and 1,000 each for dev and test sets. The 8,000 train set images were additionally augmented with 32,000 more optimizations by doing horizontal and vertical flips and 90 degree CW and CCW rotations, shown in Figure 4.

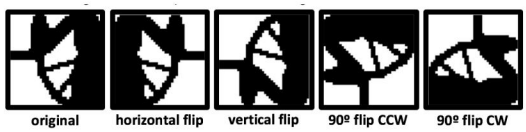

Figure 4: The data was flipped horizontally, vertically, and rotated 90 degrees CW and CCW shown here for one datapoint.

4 Methods

This project implemented two neural network architectures. A shallow neural network to determine baseline performance and a deep neural network that used the U-net[1] architecture.

The shallow neural network, shown in Figure 5, has two convolutional/relu/maxpool layers separated by a dropout layer. The final layer simply flattens the data and has sigmoid activation function to predict the outputs.

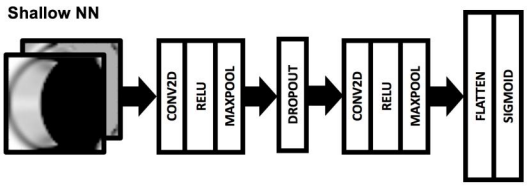

Figure 5: Shallow neural network architecture.

The deep neural network, shown in Figure 6, follows the U-net structure [1]. There are 2 conv/relu layers at each step of the inverted pyramid structure. To reduce or encode the image size it down-samples with max pool layers. Following downsampling, upsampling is performed and layers are concatenated across the architecture. There are two dropout layers in both the encoding and decoding phases for regularization. The output layer flattens and uses a sigmoid activation function at the output. The U-net architecture allows for important features of the image to be passed through the network, while reducing the overall number of parameters the model needs to process.

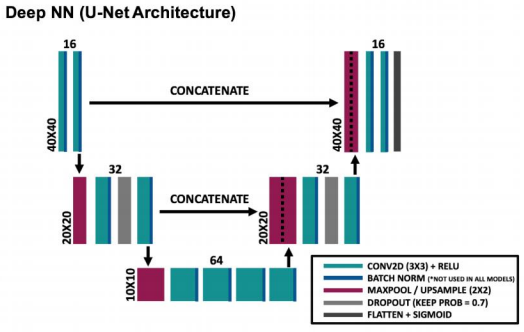

Figure 6: Deep neural network architecture.

The loss function used in both models is the pixel-wise cross-entropy loss to classify each pixel into a material or void category.

5 Experiments/Results/Discussion

Hyperparameters were tuned throughout the project. The learning rate seemed to be particularly important and a rate of 0.0001 seemed to give the best results. Any rate larger than this tended to

diverge after a number of iterations and rates smaller than this didn't show significant performance improvements. A mini-batch size of 64 was chosen because it gave a steady, although expectedly noisy, decline in cost. Different keep-probabilities were tried out to regularize the model, with 0.7 proving to be too small and 0.5 proving to provide decent performance.

The primary metrics used in this study were binary pixel-wise accuracy and intersection-over-union, whose equations are shown in Figure 7.

$$IOU = \frac{1}{m}\sum_{i=1}^{m}\frac{Area\ of\ Intersection}{Area\ of\ Union} \qquad Acc = \frac{1}{m}\sum_{i}^{m}\frac{\#\ correctly\ classified\ pixels}{total\ \#\ pixels}$$

Figure 7: IOU and binary pixel-wise accuracy metrics used to evaluate the models.

The table in Figure 8 shows the performance of the two models, with and without data augmentation. All 4 of these models trained with the same hyperparameters to make the analysis as congruent as possible. As you can see there was a consistent improvement in performance, for both test accuracy and IOU, as you move down the columns to models that are increasingly more complex and were trained on increasingly more data.

Model	Test Acc.	Test IOU
Shallow NN	84.9	91.8
Shallow NN w/ data augmentation	87.7	93.5
Deep NN	90.1	94.8
Deep NN w/ data augmentation	91.8	95.7

Figure 8: Deep neural network architecture.

The ultimate test accuracy and IOU of 91.8% and 95.7% show that the model was able to closely predict the ground truth labels. In this case, this would be an acceptable accuracy and IOU because the topology would need to be smoothed and slightly modified for manufacturability after optimization, so getting the topology 'close' would be fine. Then the designer could run a final simulation (typically a finite element stress analysis) to double check the model (this would have to happen even if the topology optimization matched perfectly to the ground truth labels).

The regularized model had significant performance improvements as shown in Figure 9, which is especially noticeable in later training epochs. This suggests that the model was definitely overfitting slightly to the training data. The keep probability in the dropout layers had to be at least 0.5 to give a noticeable improvement to the cost.

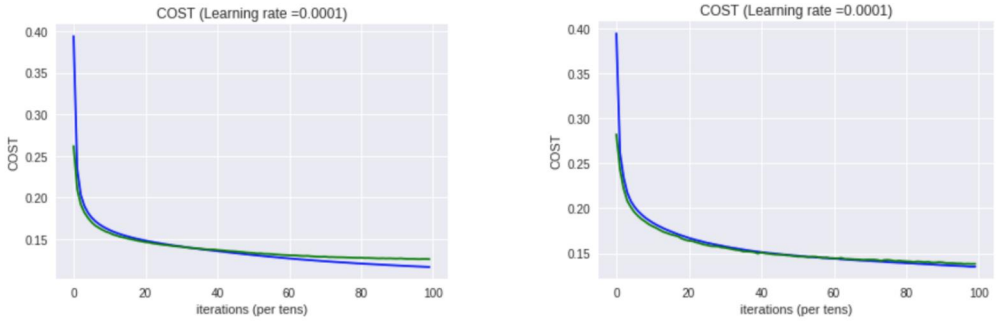

Figure 9: Unregularized and regularized cost over number of epochs for the deep neural network. Green line is dev set loss and blue line is train set loss.

Batch normalization was tried in both models but did not give performance improvements and greatly increased the dev set cost in all models. As you can see in the pictures below the sigmoid output does not seem to closely match the ground truth labels, especially for the structure on the right. It is presumed the some of the structures with 'fine grained features' (i.e. features like thin connections or small holes) got blurred out of the picture so the model had a harder time predicting their exact structures.

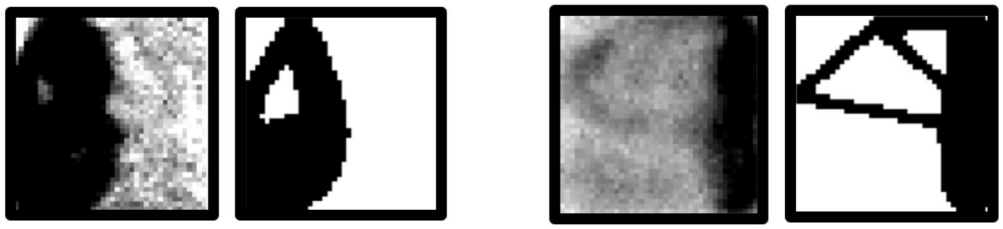

Figure 10: Batch normalization showing sigmoid output predictions and ground truth labels for 2 structures.

Overall, it was impressive to watch the model predict the geometry over time. It seemed like the model made course refinements in the earlier epochs and was just working on fine tuning in the later epochs as shown below in Figure 11

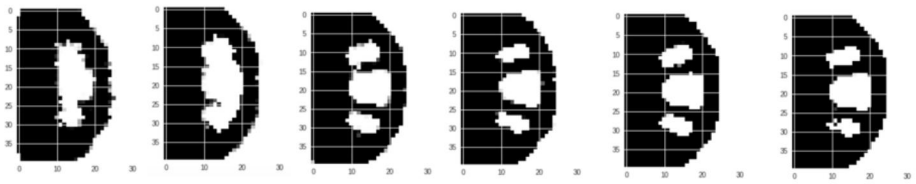

Figure 11: Deep neural network predicted structures for epochs 5, 10, 20, 30, 40, and 50.

6 Conclusion/Future Work

In conclusion, a neural network was able to give good approximations of structural topologies. The models had the most difficulty in classifying the fine-grained features of the structures, and were able to learn the course approximation much more quickly. Neural network applications to topology optimization have potential to be further refined in the future.

Ultimately, beyond the scope of this class, I would like to investigate if we're able to implement this model to remove the topology optimization entirely and just input the forces and boundary constraints to output the topology. I think it would be interesting to see if the model would still be able to predict the topology or how close it could get. Unfortunately, that was one too many things to try for this class.

In general I think there is more potential to use machine learning in design, particularly in designing optimally engineered structures, like vehicles. Topology optimization is just a starting point and has the potential to form non-intuitive designs in many different fields.

7 Contributions

Anne Alter - did all the work on the project (with lots of help from references!)

References

[1] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.

[2] Sosnovik, Ivan, and Ivan Oseledets. "Neural networks for topology optimization." arXiv preprint arXiv:1709.09578 (2017).

[3] Yildiz, A. R., et al. "Integrated optimal topology design and shape optimization using neural networks." Structural and Multidisciplinary Optimization 25.4 (2003): 251-260.

[4] Papadrakakis, Manolis, and Nikos D. Lagaros. "Reliability-based structural optimization using neural networks and Monte Carlo simulation." Computer methods in applied mechanics and engineering 191.32 (2002): 3491-3507.

[5] Papadrakakis, Manolis, Nikos D. Lagaros, and Yiannis Tsompanakis. "Structural optimization using evolution strategies and neural networks." Computer methods in applied mechanics and engineering 156.1-4 (1998): 309-333.

[6] Hunter, et al., Topy - topology optimization with python, https://github.com/williamhunter/ topy (2017).