Neural reparameterization improves structural optimization

Stephan Hoyer
Google Research
shoyer@google.com

Jascha Sohl-Dickstein
Google Research
jaschasd@google.com

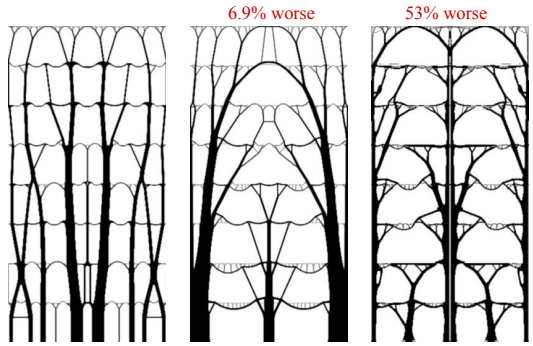
Sam Greydanus
Google Research
sgrey@google.com

Abstract

Structural optimization is a popular method for designing objects such as bridge trusses, airplane wings, and optical devices. Unfortunately, the quality of solutions depends heavily on how the problem is parameterized. In this paper, we propose using the implicit bias over functions induced by neural networks to improve the parameterization of structural optimization. Rather than directly optimizing densities on a grid, we instead optimize the parameters of a neural network which outputs those densities. This reparameterization leads to different and often better solutions. On a selection of 116 structural optimization tasks, our approach produced significantly better designs than baseline methods.

1 Introduction

One of the driving forces behind the success of deep computer vision models is the so-called "deep image prior" of convolutional neural networks (CNNs). This phrase loosely describes a set of inductive biases, present even in untrained models, that make them effective for image processing. Researchers have taken advantage of this effect to perform inpainting, noise removal, and super-resolution on images with an untrained model [29].



(a) L-BFGS (neural net) (b) MMA (pixels) (c) L-BFGS (pixels)

Figure 1: A multi-story building task. Figure (a) is a structure optimized in CNN weight space. Figures (b) and (c) are structures optimized in pixel space.

There is growing evidence that this implicit prior extends to domains beyond natural images. Some examples include style transfer in fonts [3], uncertainty estimation in fluid dynamics [32], and data upsampling in medical imaging [8]. Indeed, whenever data contains translation invariance, spatial correlation, or multi-scale features, the deep image prior may be a useful tool.

One field where these characteristics are important – and where the deep image prior is underexplored – is computational science and engineering. Here, parameterization is extremely important – substituting one parameterization for another has a dramatic effect. Consider, for example, the task of designing a multi-story building via structural optimization. The goal is to distribute a certain quantity of building material over a two-dimensional grid in order to maximize the resilience of the structure. As Figure 1 shows, different optimization methods (LBFGS [19] vs. MMA [28]) and parameterizations (pixels vs. neural net) have big consequences for the final design.

How can we harness the deep image prior to better solve problems in computational science? In this paper, we propose reparameterizing optimization problems from the basis of a grid to the basis of a

First Workshop on Solving Inverse Problems with Deep Networks (NeurIPS 2019), Vancouver, Canada.

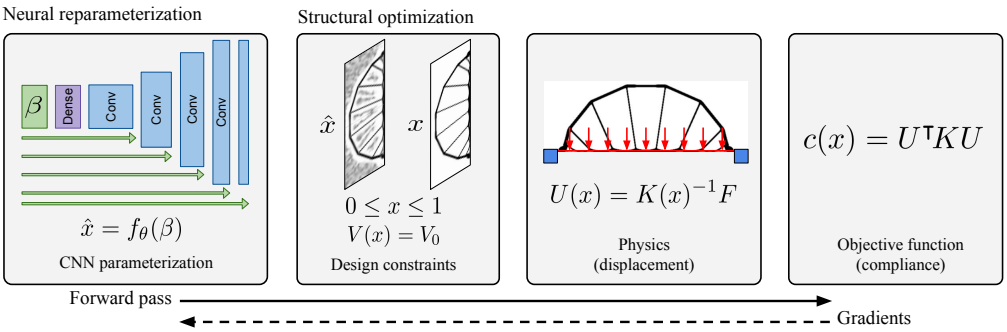


Figure 2: Schema of our approach to reparameterizing a structural optimization problem with a neural network. Each of these steps – the CNN parameterization, the constraint step, and the physics simulation – is differentiable. We implement the forward pass as a TensorFlow graph and compute gradients via automatic differentiation.

neural network. We use this approach to solve 116 structural optimization tasks and obtain solutions that are quantitatively and qualitatively better than the baselines.

2 Methods

While we apply our approach to structural optimization in this paper, we emphasize that it is generally applicable to a wide range of optimization problems in computational science. The core strategy is to write the physics model in an automatic differentiation package with support for neural networks, such as JAX, TensorFlow, or PyTorch. We emphasize that the differentiable physics model need not be written from scratch: adjoint models, as these are known in the physical sciences, are widely used [23, 9, 13], and software packages exist for computing them automatically [10].

The full computational graph begins with a neural network forward pass, proceeds to enforcing constraints and running the physics model, and ends with a scalar loss function ("compliance" in the context of structural optimization). Figure 2 gives an overview of this process. Once we have created this graph, we can recover the original optimization problem by performing gradient descent on the inputs to the constraint step ($\hat{x}$ in Figure 2). Then we can reparameterize the problem by optimizing the weights and inputs (θ and β) of a neural network which outputs $\hat{x}$.

Structural optimization. We demonstrate our reparameterization approach on the domain of structural optimization. The goal of structural optimization is to use a physics simulation to design load-bearing structures, given constraints such as conservation of volume. We focus on the general case of free-form design without configuration constraints, known as topology optimization [6].

Following the "modified SIMP" approach described by [2], we begin with a discretized domain of linear finite elements on a regular square grid. The physical density $\tilde{x}_{ij}$ at grid element (or pixel) (i, j) is computed by applying a cone-filter with radius 2 on the input densities x_{ij}. Then, letting $K(\tilde{x})$ be the global stiffness matrix, $U(K, F)$ the displacement vector, F the vector of applied forces, and $V(\tilde{x})$ the total volume, we can write the optimization objective as:

$$\min_x : c(x) = U^T KU, \quad \text{such that:} \quad KU = F, \quad V(x) = V_0, \quad \text{and } 0 \leq x_{ij} \leq 1 \ \forall (i, j). \tag{1}$$

We implemented this algorithm in NumPy, SciPy and Autograd [20]; see our source code[1] for full details. The computationally limiting step is the linear solve $U = K^{-1}F$, for which we use a sparse Cholesky factorization [7].

One key challenge was enforcing the volume and density constraints of Equation (1). Standard topology optimization methods satisfy these constraints directly, but only when directly optimizing the design variables x. Our solution was to enforce the constraints in the forward pass, by mapping unconstrained logits $\hat{x}$ into valid densities x with a constrained sigmoid transformation:

$$x_{ij} = 1/(1 + \exp[-\hat{x}_{ij} - b(\hat{x}, V_0)]), \quad \text{such that:} \quad V(x) = V_0. \tag{2}$$

[1] `https://github.com/google-research/neural-structural-optimization`

where $b(\hat{x}, V_0)$ is solved for via binary search on the volume constraint. In the backwards pass, we differentiate through the transformation at the optimal point using implicit differentiation [14].

A note on baselines. Structural optimization problems are sensitive not only to choice of parameterization but also to choice of optimization algorithm. Unfortunately, standard topology optimization algorithms like the Method of Moving Asymptotes (MMA) [28] and the Optimality Criteria (OC) [5] are ill-suited for training neural networks. How, then, can we separate the effect of parameterization from choice of optimizer? Our solution was to use a standard gradient-based optimizer, L-BFGS [22], to train both the neural network parameterization (CNN-LBFGS) and the pixel parameterization (Pixel-LBFGS). We found L-BFGS to be significantly more effective than stochastic gradient descent when optimizing a single design, similar to findings for style transfer [12].

Since constrained optimization is often much more effective at topology optimization (in pixel space, at least), we also report the MMA and OC results. In practice, we found that these provided stronger baselines than Pixel-LBFGS. Figure 3 is a good example: it shows structural optimization of an MBB beam using the three baselines. All methods except Pixel-LBFGS converge to similar, near-optimal solutions.

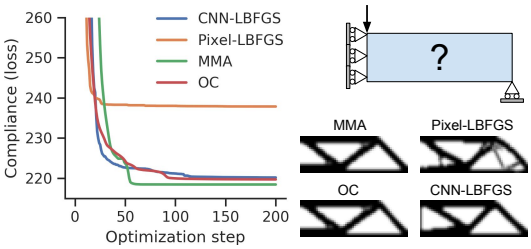


Figure 3: Comparing baselines on the MBB beam example, on a 60×20 grid. Whereas Pixel-LBFGS and CNN-LBFGS use the same optimizer, we found that MMA and OC are much stronger baselines, so we decided to report all three. We use the implementation of MMA from NLopt [17]. We re-implemented OC, but verified the results agree exactly on the tasks reported in [2].

Choosing the 116 tasks. In designing the 116 structural optimization tasks, our goal was to create a distribution of diverse, well-studied problems with real-world significance. We started with a selection of problems from [30] and [26]. Most of these classic problems are simple beams with only a few forces, so we hand-designed additional tasks reflecting real-world designs including bridges with various support restrictions, trees, ramps, walls and buildings. The final tasks fall into 28 categories, with $V_0 \in [0.05, 0.5]$ and between 2^{11} to 2^{16} elements.

Neural network methods. Our convolutional neural network architecture was inspired by the U-net architecture used in the Deep Image Prior paper [29]. We were only interested in the parameterization capabilities of the this model, so we used only the second, upsampling half of the model. We also made the first activation vector (β in Figure 2) into a trainable parameter. Our model consisted of a dense layer into 32 image channels, followed by five repetitions of tanh nonlinearity, 2x bilinear resize (for the middle three layers), global normalization by subtracting the mean and dividing by the standard deviation, a 2D convolution layer, and a learned bias over all elements/channels. The convolutional layers used 5×5 kernels and had 128, 64, 32, 16, and 1 channels respectively.

3 Analysis

We found that reparameterizing structural optimization problems with a neural network gave equal performance to MMA on small problems and compellingly better performance on large problems. On both small and large problems, it produced much better designs than OC and Pixel-LBFGS.

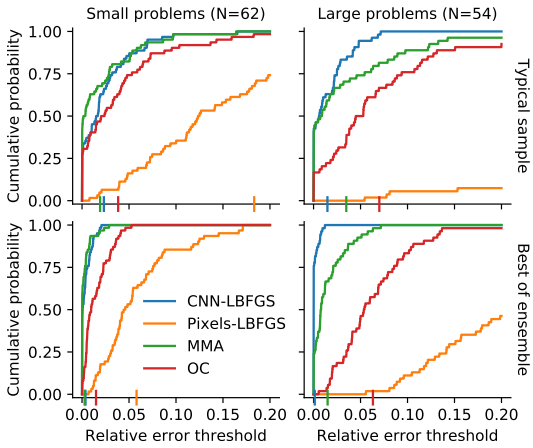


Figure 4: Empirical distribution of the relative error across design tasks. The x-axes measure design error relative to the best overall design. The y-axes measure the probability that the method's solution has an error below the x-axis threshold. Corresponding ticks on the x-axis indicate the average error for each model.

For each task, we report typical performance (median over 101 random seeds for the CNN, and constant initialization for the other models, which was always better than the median) and "best-of-ensemble" performance (same initialization for all models, taken from the untrained

CNN). Figure 4 summarizes our results. On large problems (defined by $\geq 2^{15}$ grid points) both typical and best-of-ensemble CNN-LBFGS solutions were much more likely to have low error.

Why do large problems benefit more? Returning to the literature, we found that finite grids can suffer from a "mesh-dependency problem", with varying solutions as grid resolution changes [25]. When grid resolution is high, small-scale "spiderweb" structures tend to form first and then interfere with the development of large-scale structures. We suspected that optimizing the weights of a CNN allowed us to instead optimize structures at several spatial scales at once, thus improving optimization dynamics. To investigate this idea, we plotted structures from all 116 design tasks (see Appendix). Then we chose five examples to highlight and showcase important qualitative trends (Figure 5).

Reparameterized designs are often simpler. The CNN-LBFGS designs have fewer "spiderweb" artifacts as shown in the cantilever beam, MBB beam, and suspended bridge examples. On the cantilever beam, CNN-LBFGS used a total of eight supports whereas MMA used eighteen. We see simpler structures as evidence that the CNN biased optimization towards large-scale structure. This effect was particularly pronounced for large problems, which may explain why they benefited more.

Convergence to different solutions. We also noted that the baseline structures resembled each other more closely than they did CNN-LBFGS. In the thin support bridge example, the baseline designs feature double support columns whereas CNN-LBFGS used a single vertical support. In the center support task, only CNN-LBFGS avoids using any horizontal beam.

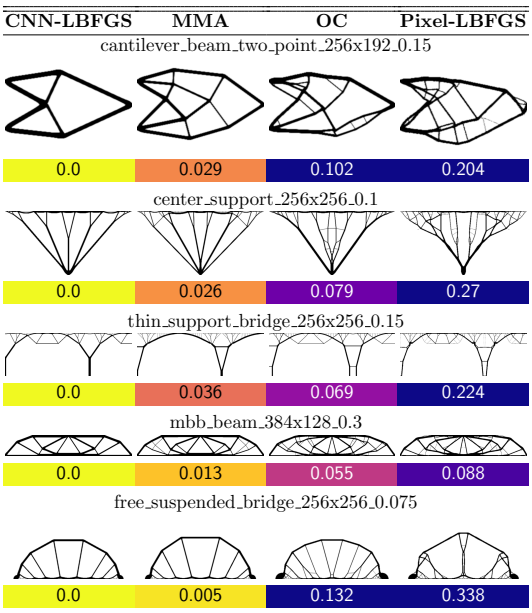


Figure 5: Qualitative examples of structural optimization via reparameterization. The scores below each structure measure relative difference between the design and the best overall design in that row. The CNN solutions are qualitatively different from the baselines and often involve simpler and more effective structures.

4 Related work

Differentiable image parameterizations have been shown to be a powerful tool for controlling image-based optimization tasks [21].

Parameterizing topology optimization. The most common parameterization for topology optimization is a grid mesh [2, 24, 31]. Sometimes, polyhedral meshes are used [11]. Some domain-specific structural optimizations feature locally-refined meshes and multiple load case adjustments [18]. Like locally refined meshes, our method permits structure optimization at multiple scales. Unlike them, our method permits optimization on both scales *at once*.

Neural networks and topology optimization. Several papers have proposed replacing topology optimization methods with CNNs [4, 27, 1, 16]. Most of them begin by creating a dataset of structures via regular topology optimization and then training a model on the dataset. While doing so can reduce computation, it comes at the expense of relaxing physics and design constraints. More problematically, these models can only reproduce their training data. In contrast, our approach produces *better* designs that also obey exact physics constraints. One recent work resembles ours in that they use adjoint gradients to train a CNN model [15]. Their goal was to learn a joint, conditional model over a range of related tasks, which is different from our goal of reparameterizing a single structure.

5 Conclusions

Choice of parameterization has a powerful effect on solution quality for tasks such as structural optimization, where solutions must be computed by numerical optimization. Motivated by the observation that untrained deep image models have good inductive biases for many tasks, we

reparameterized structural optimization tasks in terms of the output of a convolutional neural network (CNN). Optimization then involved training the parameters of this CNN for each task. The resulting framework produced qualitatively and quantitatively better designs on a set of 116 tasks.

References

[1] Alter, A. Structural topology optimization using a convolutional neural network. *preprint*, 2018. URL `http://cs230.stanford.edu/files_winter_2018/projects/6907833.pdf`.

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

[3] Azadi, S., Fisher, M., Kim, V. G., Wang, Z., Shechtman, E., and Darrell, T. Multi-content gan for few-shot font style transfer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

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

[5] Bendsøe, M. P. *Optimization of Structural Topology, Shape, and Material*. Springer, Berlin, Heidelberg, 1995.

[6] Bendsoe, M. P. and Sigmund, O. *Topology Optimization: Theory, Methods, and Applications*. Springer Science & Business Media, April 2013.

[7] Chen, Y., Davis, T. A., Hager, W. W., and Rajamanickam, S. Algorithm 887: Cholmod, supernodal sparse cholesky factorization and update/downdate. *ACM Trans. Math. Softw.*, 35(3):22:1–22:14, October 2008. ISSN 0098-3500. doi: 10.1145/1391989.1391995. URL `http://doi.acm.org/10.1145/1391989.1391995`.

[8] Dittmer, S., Kluth, T., Maass, P., and Baguer, D. O. Regularization by architecture: A deep prior approach for inverse problems. *arXiv preprint arXiv:1812.03889*, 2018.

[9] Errico, R. M. What is an adjoint model? *Bull. Am. Meteorol. Soc.*, 78:2539, 1997.

[10] Farrell, P. E., Ham, D. A., Funke, S. W., and Rognes, M. E. Automated derivation of the adjoint of High-Level transient finite element programs. *SIAM Journal on Scientific Computing*, 35(4): C369–C393, 2013.

[11] Gain, A. L., Paulino, G. H., Duarte, L. S., and Menezes, I. F. Topology optimization using polytopes. *Computer Methods in Applied Mechanics and Engineering*, 293:411–430, 2015.

[12] Gatys, L. A., Ecker, A. S., and Bethge, M. Image style transfer using convolutional neural networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[13] Giles, M. B. and Pierce, N. A. An introduction to the adjoint approach to design. *Flow, Turbulence and Combustion*, 65(3/4):393–415, 2000.

[14] Griewank, A. and Faure, C. Reduced functions, gradients and hessians from fixed-point iterations for state equations. *Numerical Algorithms*, 30(2):113–139, 2002.

[15] Jiang, J. and Fan, J. A. Global optimization of dielectric metasurfaces using a Physics-Driven neural network. *Nano Lett.*, 19(8):5366–5372, August 2019.

[16] Jiang, J., Sell, D., Hoyer, S., Hickey, J., Yang, J., and Fan, J. A. Free-form diffractive metagrating design based on generative adversarial networks. *ACS Nano*, 13(8):8872–8878, 2019. doi: 10.1021/acsnano.9b02371. URL `https://doi.org/10.1021/acsnano.9b02371`. PMID: 31314492.

[17] Johnson, S. G. The NLopt nonlinear-optimization package. URL `http://github.com/stevengj/nlopt`.

[18] Krog, L., Tucker, A., Kemp, M., and Boyd, R. Topology optimisation of aircraft wing box ribs. In *10th AIAA/ISSMO multidisciplinary analysis and optimization conference*, pp. 4481, 2004.

[19] Liu, D. C. and Nocedal, J. On the limited memory bfgs method for large scale optimization. *Mathematical programming*, 45(1-3):503–528, 1989.

[20] Maclaurin, D., Duvenaud, D., and Adams, R. P. Autograd: Effortless gradients in numpy. In *ICML 2015 AutoML Workshop*, 2015. URL `https://github.com/HIPS/autograd`.

[21] Mordvintsev, A., Pezzotti, N., Schubert, L., and Olah, C. Differentiable image parameterizations. *Distill*, 2018. doi: 10.23915/distill.00012. https://distill.pub/2018/differentiable-parameterizations.

[22] Nocedal, J. Updating quasi-newton matrices with limited storage. *Math. Comput.*, 35(151): 773–773, 1980.

[23] Plessix, R.-E. and E. Plessix, R. A review of the adjoint-state method for computing the gradient of a functional with geophysical applications. *Geophys. J. Int.*, 167(2):495–503, 2006.

[24] Sigmund, O. A 99 line topology optimization code written in matlab. *Structural and multidisciplinary optimization*, 21(2):120–127, 2001.

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

[26] Sokół, T. A 99 line code for discretized michell truss optimization written in mathematica. *Structural and Multidisciplinary Optimization*, 43(2):181–190, 2011.

[27] Sosnovik, I. and Oseledets, I. Neural networks for topology optimization. *Russian Journal of Numerical Analysis and Mathematical Modelling*, 34(4):215–223, 2019.

[28] Svanberg, K. The method of moving asymptotes—a new method for structural optimization. *International Journal for Numerical Methods in Engineering*, 24(2):359–373, 1987.

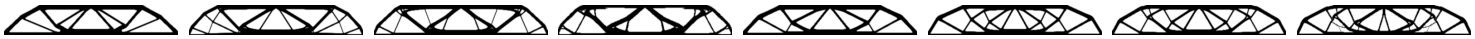
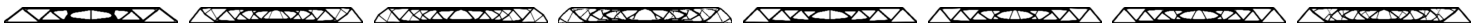
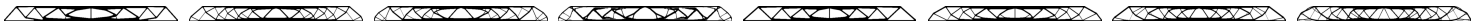
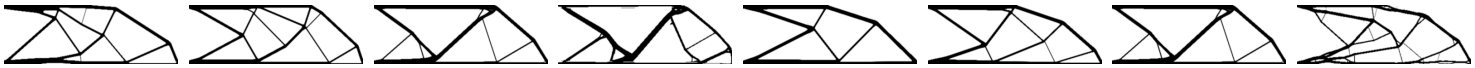
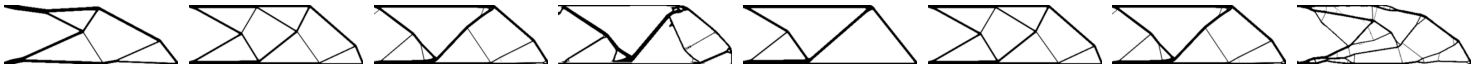
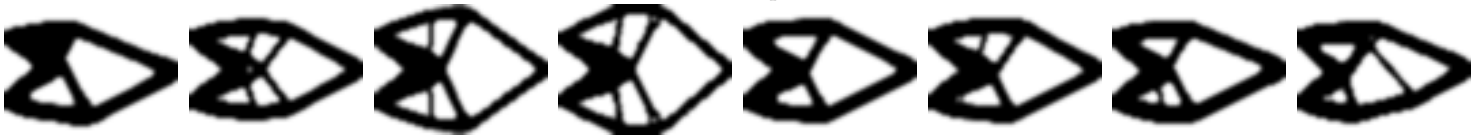
[29] Ulyanov, D., Vedaldi, A., and Lempitsky, V. Deep image prior. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 9446–9454, 2018.

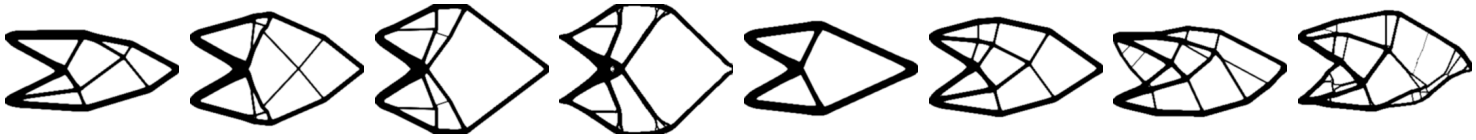
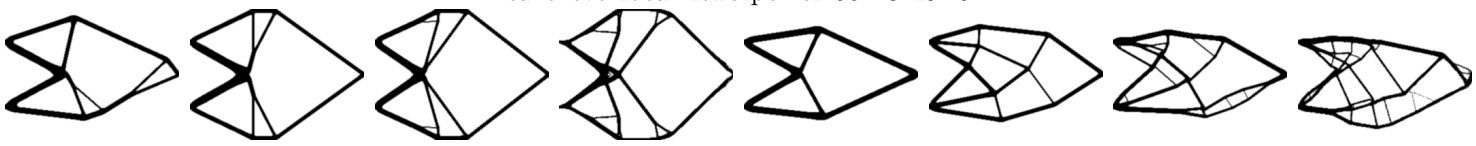
[30] Valdez, S. I., Botello, S., Ochoa, M. A., Marroquín, J. L., and Cardoso, V. Topology optimization benchmarks in 2d: Results for minimum compliance and minimum volume in planar stress problems. *Arch. Comput. Methods Eng.*, 24(4):803–839, November 2017.

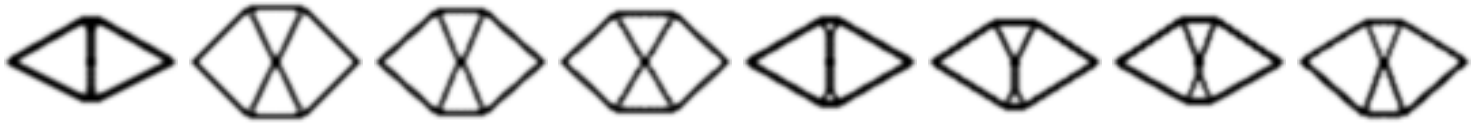
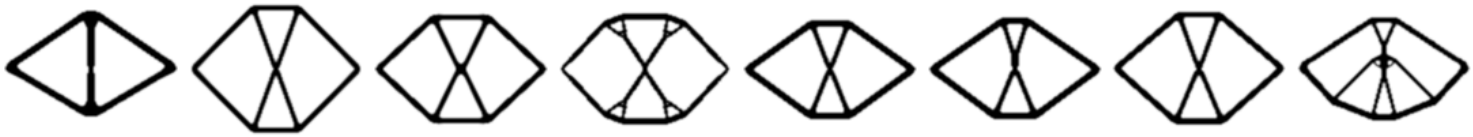
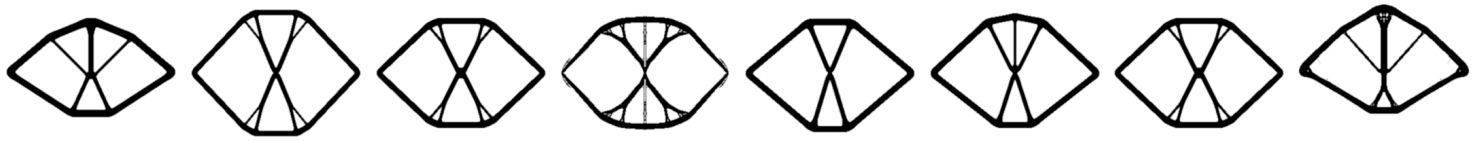
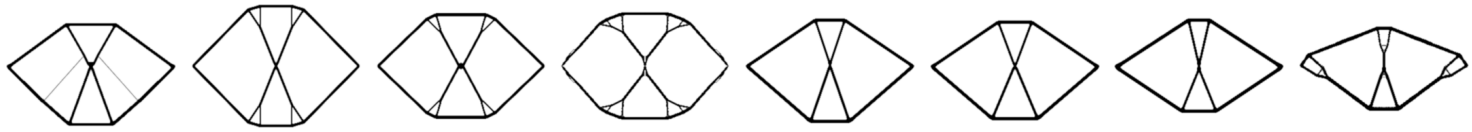
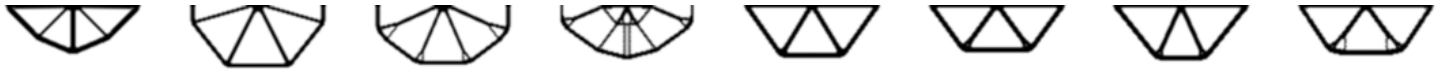
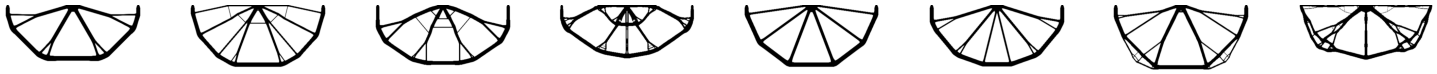
[31] Zhu, J.-H., Zhang, W.-H., and Xia, L. Topology optimization in aircraft and aerospace structures design. *Archives of Computational Methods in Engineering*, 23(4):595–622, 2016.

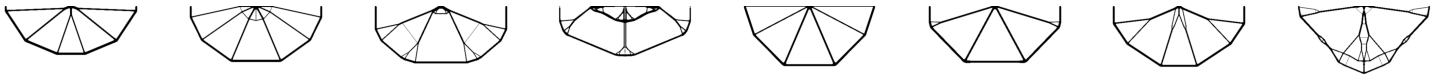
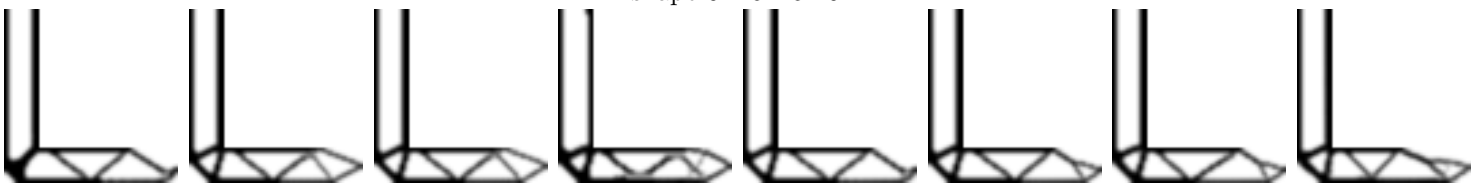
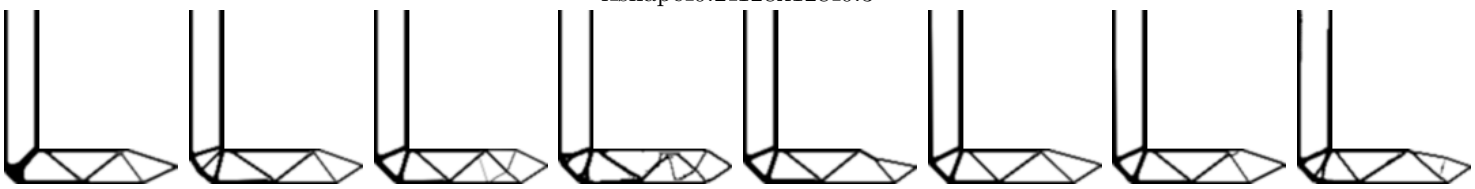
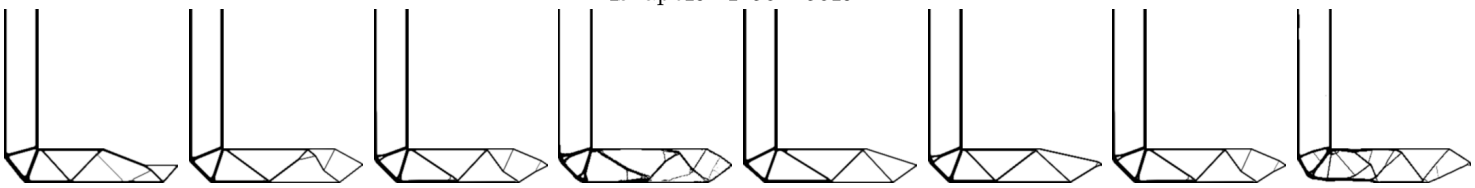
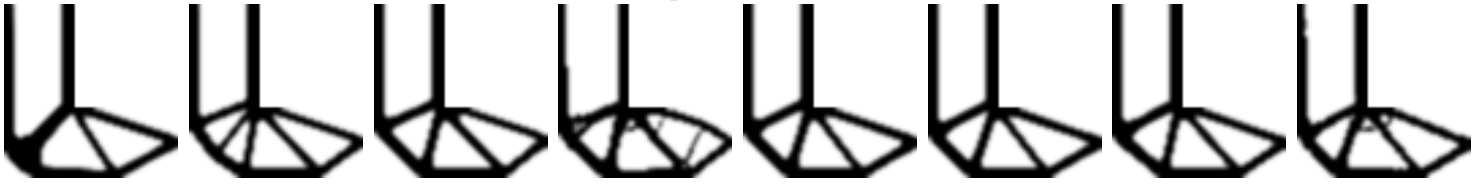
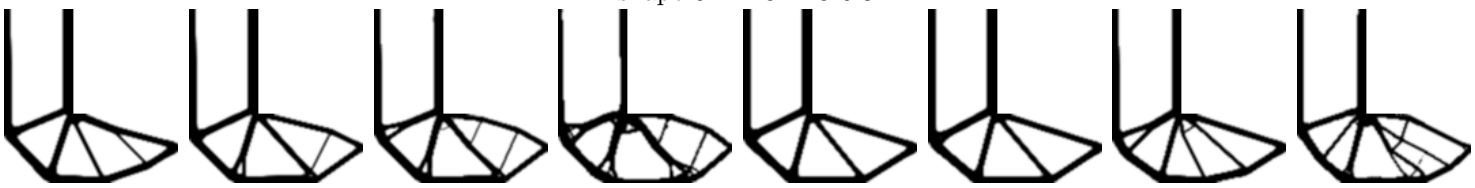
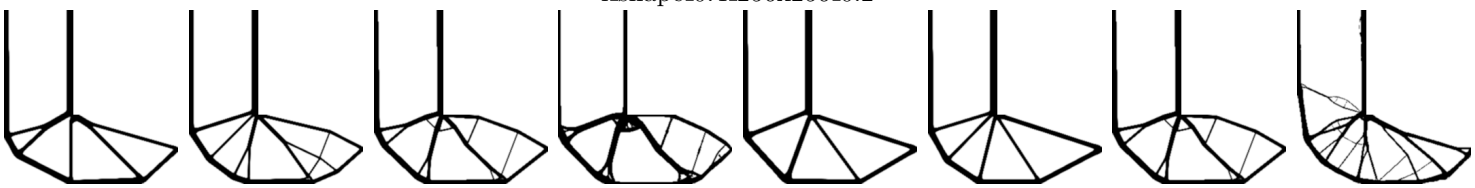
[32] Zhu, Y., Zabaras, N., Koutsourelakis, P.-S., and Perdikaris, P. Physics-constrained deep learning for high-dimensional surrogate modeling and uncertainty quantification without labeled data. *Journal of Computational Physics*, 394:56–81, 2019.

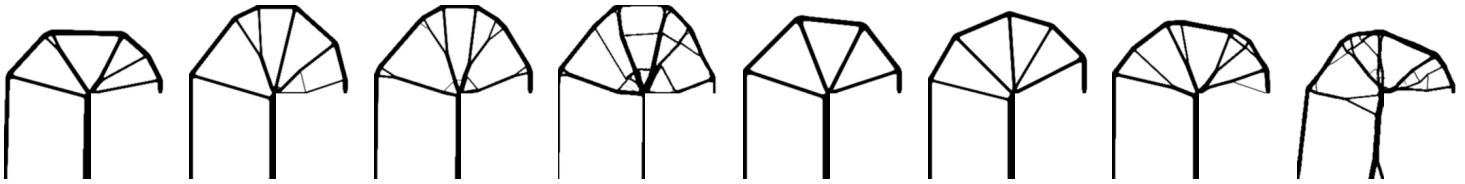
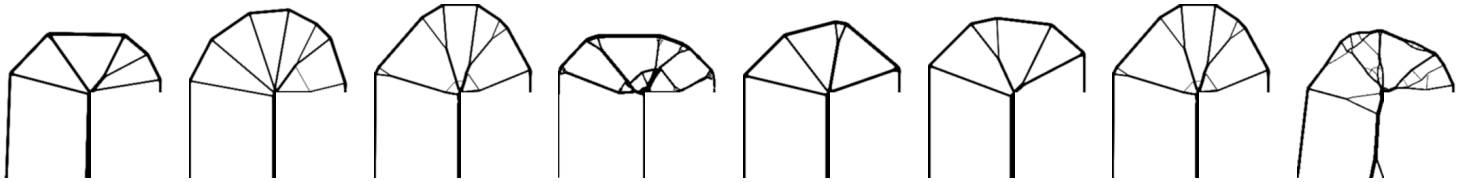
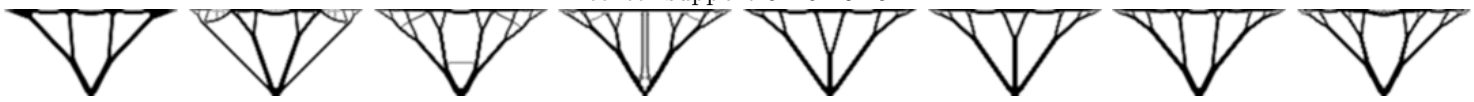
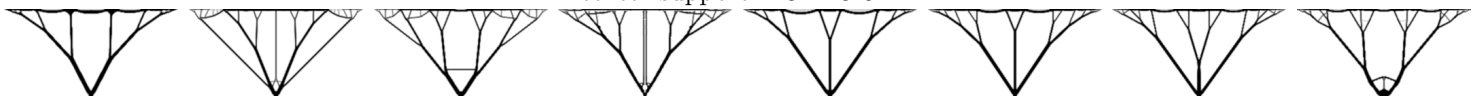
Table A1: Relative and absolute compliance for each of our 116 designs, for both the "typical sample" and the "best of ensemble."

Typical sample				Best of ensemble			
CNN-LBFGS	**MMA**	**OC**	**Pixel-LBFGS**	**CNN-LBFGS**	**MMA**	**OC**	**Pixel-LBFGS**
mbb_beam_96x32_0.5							
0.0 / 206.5	0.016 / 209.8	0.02 / 210.5	0.069 / 220.7	0.0 / 205.5	0.003 / 206.0	0.006 / 206.8	0.019 / 209.4
mbb_beam_192x64_0.4							
							
0.0 / 241.5	0.006 / 242.9	0.015 / 245.1	0.153 / 278.5	0.0 / 237.9	0.003 / 238.6	0.008 / 239.7	0.054 / 250.7
mbb_beam_384x128_0.3							
0.006 / 302.9	0.0 / 301.0	0.054 / 317.2	0.342 / 403.9	0.0 / 294.8	0.013 / 298.5	0.055 / 311.0	0.088 / 320.8
mbb_beam_192x32_0.5							
							
0.0 / 1235.8	0.0 / 1236.1	0.006 / 1243.6	0.084 / 1339.7	0.0 / 1221.3	0.004 / 1226.2	0.004 / 1225.9	0.028 / 1255.0
mbb_beam_384x64_0.4							
							
0.003 / 1397.8	0.0 / 1392.9	0.016 / 1415.0	0.168 / 1627.0	0.0 / 1375.3	0.004 / 1381.2	0.014 / 1394.3	0.076 / 1479.1
cantilever_beam_full_96x32_0.4							
0.0 / 250.1	0.021 / 255.5	0.026 / 256.7	0.126 / 281.8	0.0 / 247.3	0.004 / 248.3	0.01 / 249.7	0.031 / 254.9
cantilever_beam_full_192x64_0.3							
0.001 / 312.8	0.0 / 312.4	0.008 / 314.9	0.253 / 391.5	0.002 / 304.8	0.0 / 304.2	0.023 / 311.1	0.08 / 328.4
cantilever_beam_full_384x128_0.2							
							
0.02 / 446.8	0.0 / 437.9	0.035 / 453.1	0.699 / 744.1	0.0 / 431.6	0.002 / 432.5	0.05 / 453.1	0.189 / 513.2
cantilever_beam_full_384x128_0.15							
							
0.068 / 641.5	0.0 / 600.4	0.055 / 633.4	0.978 / 1187.5	0.0 / 597.7	0.005 / 600.4	0.06 / 633.4	0.285 / 767.7
cantilever_beam_two_point_64x48_0.4							
							
0.007 / 85.9	0.0 / 85.3	0.046 / 89.2	0.072 / 91.4	0.0 / 84.8	0.002 / 85.0	0.004 / 85.1	0.013 / 85.9
cantilever_beam_two_point_128x96_0.3							
0.0 / 111.2	0.001 / 111.2	0.043 / 115.9	0.123 / 124.9	0.0 / 108.9	0.004 / 109.4	0.023 / 111.4	0.035 / 112.8

Typical sample				Best of ensemble			
CNN-LBFGS	**MMA**	**OC**	**Pixel-LBFGS**	**CNN-LBFGS**	**MMA**	**OC**	**Pixel-LBFGS**
cantilever_beam_two_point_256x192_0.2							
							
0.0 / 158.8	0.032 / 163.8	0.077 / 171.0	0.288 / 204.5	0.0 / 155.1	0.008 / 156.4	0.06 / 164.4	0.134 / 175.9
cantilever_beam_two_point_256x192_0.15							
							
0.0 / 212.8	0.063 / 226.3	0.111 / 236.3	0.337 / 284.6	0.0 / 204.9	0.029 / 210.7	0.102 / 225.7	0.204 / 246.6
pure_bending_moment_32x64_0.15							
0.165 / 9.6	0.067 / 8.8	0.0 / 8.2	0.088 / 9.0	0.001 / 7.8	0.017 / 8.0	0.0 / 7.8	0.052 / 8.2
pure_bending_moment_64x128_0.125							
0.085 / 10.1	0.053 / 9.8	0.0 / 9.3	0.176 / 10.9	0.0 / 9.1	0.0 / 9.1	0.0 / 9.1	0.04 / 9.5
pure_bending_moment_128x256_0.1							
0.071 / 12.8	0.046 / 12.5	0.0 / 11.9	0.411 / 16.9	0.0 / 11.7	0.007 / 11.8	0.005 / 11.8	0.157 / 13.6
ground_structure_64x64_0.12							
0.0 / 47.2	0.097 / 51.8	0.266 / 59.8	0.054 / 49.7	0.0 / 44.8	0.0 / 44.8	0.016 / 45.5	0.059 / 47.5
ground_structure_128x128_0.1							
0.0 / 53.0	0.051 / 55.7	0.062 / 56.3	0.008 / 53.4	0.0 / 49.3	0.009 / 49.7	0.046 / 51.6	0.082 / 53.3
ground_structure_256x256_0.07							
0.005 / 72.2	0.0 / 71.9	0.022 / 73.5	0.078 / 77.5	0.0 / 67.0	0.012 / 67.8	0.074 / 71.9	0.157 / 77.5

Typical sample				Best of ensemble			
CNN-LBFGS	**MMA**	**OC**	**Pixel-LBFGS**	**CNN-LBFGS**	**MMA**	**OC**	**Pixel-LBFGS**
ground_structure_256x256_0.05							
0.048 / 109.3	0.0 / 104.3	0.089 / 113.5	0.371 / 142.9	0.0 / 97.7	0.019 / 99.5	0.134 / 110.8	0.44 / 140.6
michell_centered_both_32x64_0.12							
							
0.0 / 51.0	0.097 / 55.9	0.064 / 54.3	0.086 / 55.4	0.0 / 50.0	0.0 / 50.0	0.0 / 50.0	0.04 / 52.0
michell_centered_both_64x128_0.12							
							
0.018 / 38.1	0.024 / 38.3	0.0 / 37.4	0.141 / 42.7	0.0 / 36.3	0.007 / 36.5	0.011 / 36.7	0.116 / 40.5
michell_centered_both_128x256_0.12							
							
0.022 / 32.9	0.014 / 32.6	0.0 / 32.2	0.291 / 41.5	0.001 / 31.6	0.0 / 31.6	0.02 / 32.2	0.113 / 35.1
michell_centered_both_128x256_0.06							
							
0.022 / 80.9	0.007 / 79.7	0.0 / 79.1	0.384 / 109.5	0.001 / 74.5	0.0 / 74.4	0.017 / 75.7	0.186 / 88.3
michell_centered_below_64x64_0.12							
							
0.0 / 86.9	0.004 / 87.2	0.048 / 91.0	0.198 / 104.1	0.0 / 80.1	0.019 / 81.6	0.021 / 81.8	0.037 / 83.1
michell_centered_below_128x128_0.12							
0.018 / 69.4	0.0 / 68.2	0.015 / 69.2	0.239 / 84.4	0.0 / 65.4	0.008 / 65.9	0.01 / 66.1	0.042 / 68.2
michell_centered_below_256x256_0.12							
							
0.002 / 61.8	0.0 / 61.7	0.043 / 64.3	0.231 / 75.9	0.0 / 59.6	0.012 / 60.4	0.078 / 64.3	0.199 / 71.5

Typical sample				Best of ensemble			
CNN-LBFGS	**MMA**	**OC**	**Pixel-LBFGS**	**CNN-LBFGS**	**MMA**	**OC**	**Pixel-LBFGS**
michell_centered_below_256x256_0.06							
							
0.01 / 141.5	0.0 / 140.1	0.084 / 151.9	1.194 / 307.5	0.0 / 132.1	0.02 / 134.8	0.069 / 141.2	0.369 / 180.9
l_shape_0.2_64x64_0.4							
							
0.014 / 2732.5	0.0 / 2693.7	0.001 / 2695.3	0.048 / 2822.6	0.002 / 2690.0	0.0 / 2683.9	0.001 / 2687.8	0.012 / 2714.8
l_shape_0.2_128x128_0.3							
							
0.018 / 3079.6	0.0 / 3025.4	0.007 / 3045.3	0.094 / 3309.7	0.0 / 2993.9	0.001 / 2995.5	0.005 / 3008.1	0.035 / 3097.8
l_shape_0.2_256x256_0.2							
							
0.022 / 4418.9	0.0 / 4322.7	0.009 / 4360.2	0.202 / 5193.7	0.0 / 4277.1	0.006 / 4304.1	0.019 / 4360.2	0.141 / 4881.1
l_shape_0.4_64x64_0.4							
							
0.037 / 230.2	0.007 / 223.5	0.0 / 222.0	0.059 / 235.1	0.0 / 220.6	0.001 / 220.9	0.004 / 221.6	0.019 / 224.8
l_shape_0.4_128x128_0.3							
							
0.004 / 291.4	0.0 / 290.1	0.02 / 295.9	0.158 / 335.9	0.0 / 286.8	0.001 / 287.1	0.031 / 295.6	0.075 / 308.2
l_shape_0.4_256x256_0.2							
							
0.0 / 410.4	0.001 / 410.6	0.021 / 418.9	0.345 / 551.9	0.0 / 399.4	0.001 / 399.9	0.049 / 418.9	0.203 / 480.4

Typical sample				Best of ensemble			
CNN-LBFGS	**MMA**	**OC**	**Pixel-LBFGS**	**CNN-LBFGS**	**MMA**	**OC**	**Pixel-LBFGS**
crane_64x64_0.3							
0.044 / 102.8	0.001 / 98.5	0.0 / 98.4	0.111 / 109.4	0.001 / 97.5	0.0 / 97.4	0.0 / 97.4	0.028 / 100.2
crane_128x128_0.2							
0.0 / 141.1	0.032 / 145.6	0.031 / 145.4	0.242 / 175.3	0.0 / 136.1	0.001 / 136.1	0.033 / 140.6	0.085 / 147.6
crane_256x256_0.15							
							
0.0 / 174.4	0.013 / 176.8	0.053 / 183.7	0.563 / 272.6	0.0 / 170.1	0.003 / 170.7	0.053 / 179.2	0.224 / 208.1
crane_256x256_0.1							
							
0.0 / 292.3	0.022 / 298.7	0.035 / 302.5	0.46 / 426.7	0.0 / 277.6	0.021 / 283.4	0.089 / 302.5	0.348 / 374.4
center_support_64x64_0.15							
							
0.0 / 39.3	0.041 / 41.0	0.04 / 40.9	0.111 / 43.7	0.002 / 37.2	0.0 / 37.1	0.018 / 37.8	0.042 / 38.7
center_support_128x128_0.1							
							
0.0 / 54.6	0.052 / 57.4	0.059 / 57.8	0.212 / 66.2	0.003 / 51.5	0.0 / 51.4	0.01 / 51.9	0.171 / 60.1
center_support_256x256_0.1							
0.0 / 43.8	0.061 / 46.5	0.057 / 46.3	0.29 / 56.5	0.0 / 41.9	0.026 / 43.0	0.079 / 45.3	0.27 / 53.3
center_support_256x256_0.05							
0.0 / 120.3	0.122 / 134.9	0.121 / 134.9	0.518 / 182.5	0.0 / 109.7	0.024 / 112.3	0.137 / 124.7	0.478 / 162.1

Typical sample				Best of ensemble			
CNN-LBFGS	**MMA**	**OC**	**Pixel-LBFGS**	**CNN-LBFGS**	**MMA**	**OC**	**Pixel-LBFGS**

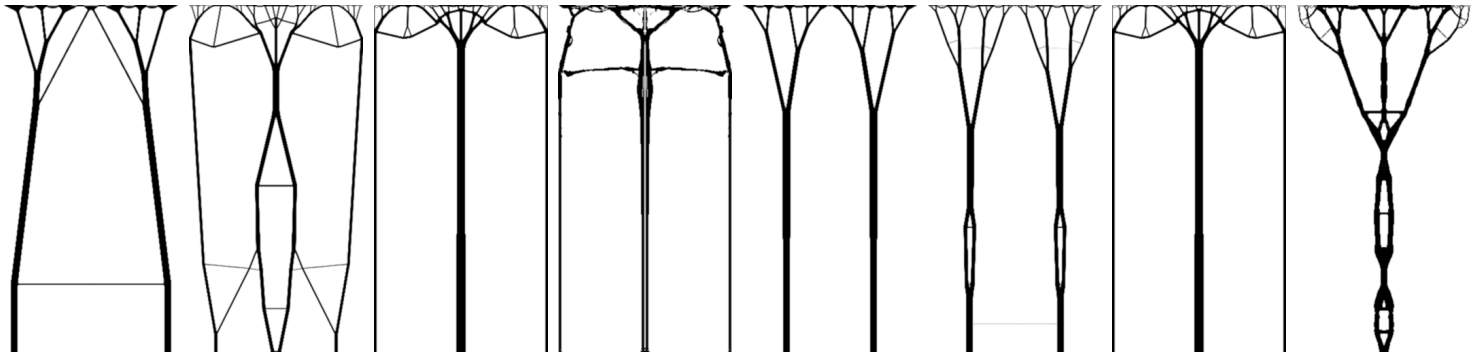
column_32x128_0.3

0.015 / 19.7	0.0 / 19.5	0.009 / 19.6	0.292 / 25.1	0.012 / 19.0	0.0 / 18.8	0.005 / 18.9	0.024 / 19.2

column_64x256_0.3

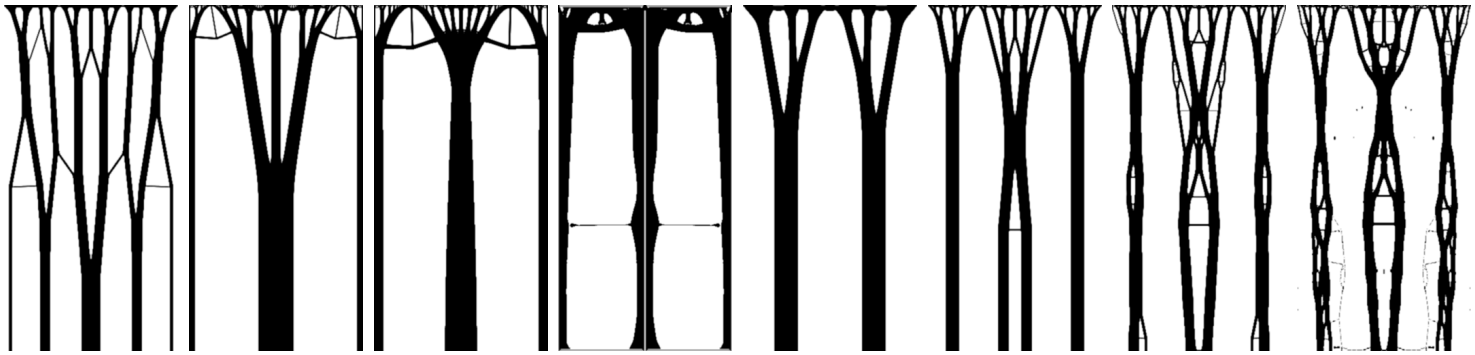
0.018 / 17.2	0.0 / 16.9	0.024 / 17.3	0.263 / 21.4	0.008 / 16.5	0.0 / 16.4	0.041 / 17.0	0.13 / 18.5

column_128x512_0.1



0.029 / 62.9	0.262 / 77.2	0.0 / 61.2	0.388 / 84.9	0.0 / 55.3	0.058 / 58.5	0.106 / 61.2	0.366 / 75.5

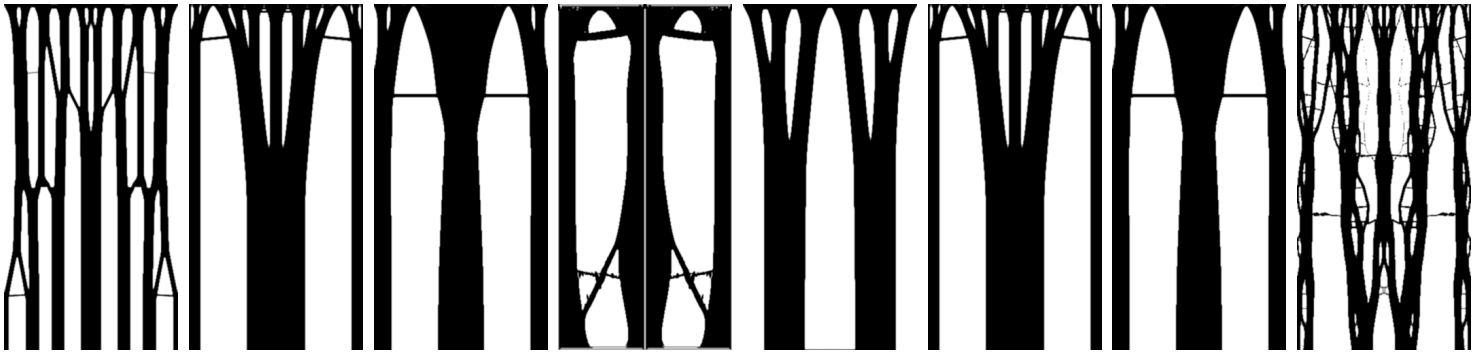
column_128x512_0.3



0.048 / 16.3	0.0 / 15.6	0.053 / 16.4	0.288 / 20.1	0.0 / 15.3	0.007 / 15.4	0.065 / 16.3	0.151 / 17.6

Typical sample				Best of ensemble			
CNN-LBFGS	MMA	OC	Pixel-LBFGS	CNN-LBFGS	MMA	OC	Pixel-LBFGS

column_128x512_0.5



0.042 / 9.1	0.0 / 8.7	0.026 / 9.0	0.238 / 10.8	0.0 / 8.7	0.01 / 8.7	0.037 / 9.0	0.122 / 9.7

roof_64x64_0.2

0.0 / 10.0	0.03 / 10.3	0.045 / 10.4	0.751 / 17.5	0.004 / 9.2	0.0 / 9.1	0.039 / 9.5	0.071 / 9.8

roof_128x128_0.15

0.05 / 11.9	0.164 / 13.1	0.0 / 11.3	1.036 / 23.0	0.0 / 11.0	0.023 / 11.3	0.025 / 11.3	0.167 / 12.8

roof_256x256_0.4

0.042 / 3.0	0.0 / 2.8	0.035 / 2.9	0.48 / 4.2	0.005 / 2.9	0.0 / 2.8	0.035 / 2.9	0.129 / 3.2

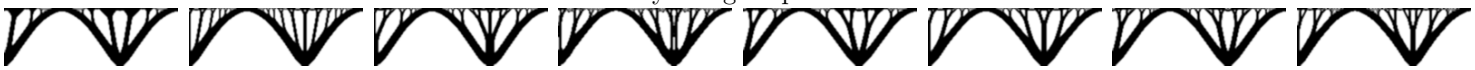
roof_256x256_0.2

0.035 / 6.9	0.086 / 7.3	0.0 / 6.7	1.071 / 13.9	0.0 / 6.5	0.008 / 6.5	0.031 / 6.7	0.219 / 7.9

roof_256x256_0.1

0.062 / 17.2	0.143 / 18.5	0.0 / 16.2	1.975 / 48.2	0.0 / 16.0	0.051 / 16.8	0.014 / 16.2	0.355 / 21.7

causeway_bridge_top_64x64_0.3

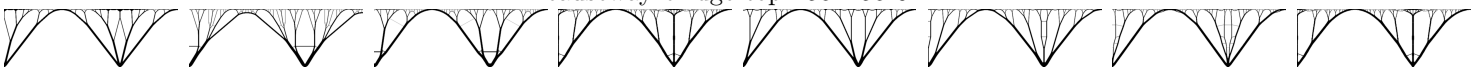


0.009 / 12.7	0.004 / 12.6	0.0 / 12.6	0.021 / 12.8	0.005 / 12.6	0.0 / 12.5	0.002 / 12.5	0.007 / 12.6

causeway_bridge_top_128x128_0.2

0.023 / 17.8	0.0 / 17.4	0.033 / 17.9	0.068 / 18.5	0.004 / 17.4	0.0 / 17.3	0.006 / 17.4	0.054 / 18.2

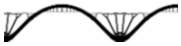
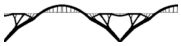
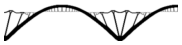
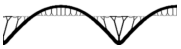
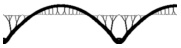
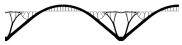
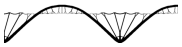
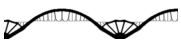
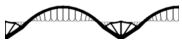
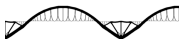
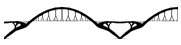
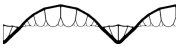
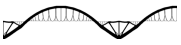
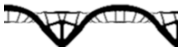
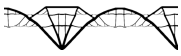
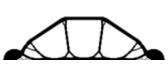
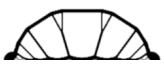
causeway_bridge_top_256x256_0.1

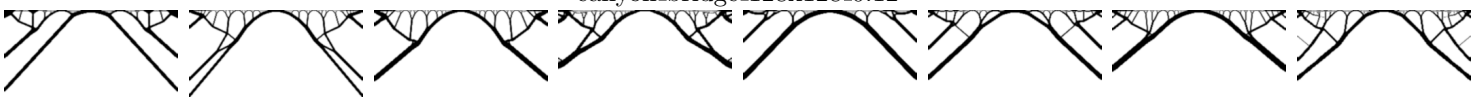
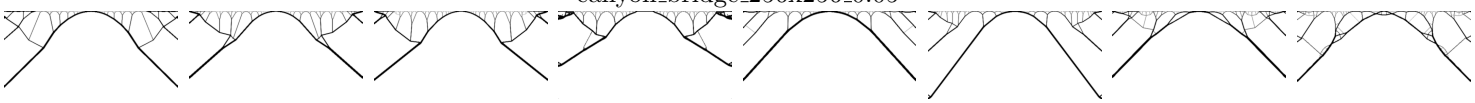


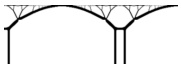
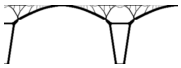
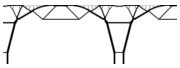
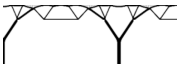
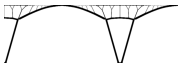
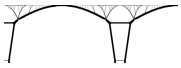
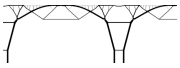
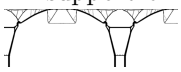
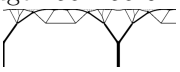
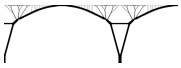
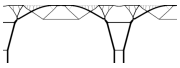
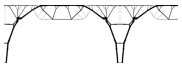
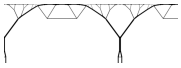
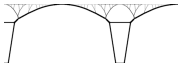
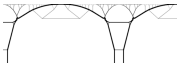
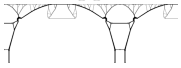
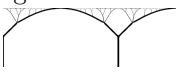
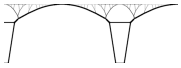
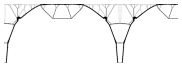
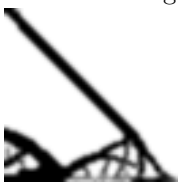
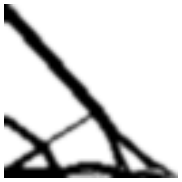
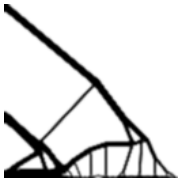
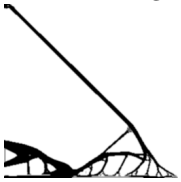
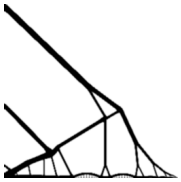
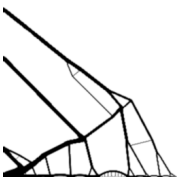
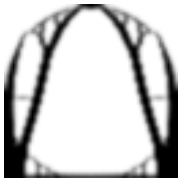
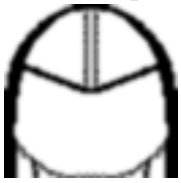
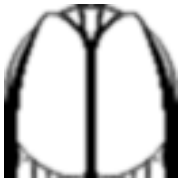
0.0 / 33.9	0.098 / 37.2	0.034 / 35.1	0.081 / 36.7	0.0 / 32.7	0.009 / 33.0	0.04 / 34.1	0.12 / 36.7

causeway_bridge_top_128x64_0.3

0.01 / 21.1	0.0 / 20.9	0.001 / 20.9	0.04 / 21.8	0.005 / 20.9	0.0 / 20.8	0.006 / 20.9	0.017 / 21.1

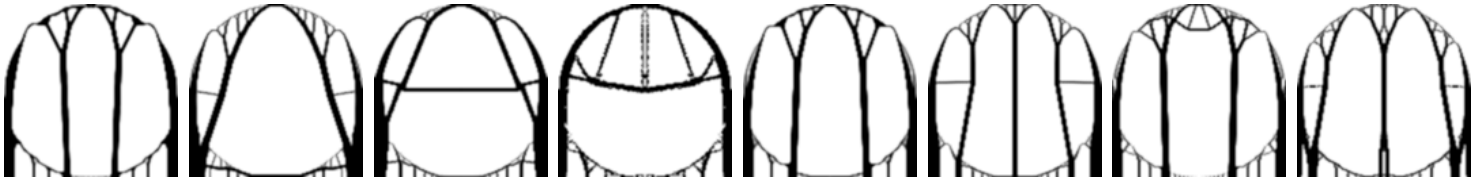
Typical sample				Best of ensemble			
CNN-LBFGS	**MMA**	**OC**	**Pixel-LBFGS**	**CNN-LBFGS**	**MMA**	**OC**	**Pixel-LBFGS**
causeway_bridge_top_256x128_0.2							
0.0 / 30.0	0.017 / 30.5	0.006 / 30.2	0.054 / 31.6	0.003 / 29.4	0.0 / 29.3	0.015 / 29.7	0.081 / 31.6
causeway_bridge_middle_64x64_0.12							
							
0.01 / 27.3	0.029 / 27.8	0.0 / 27.0	0.072 / 29.0	0.001 / 26.6	0.0 / 26.6	0.005 / 26.8	0.033 / 27.5
causeway_bridge_middle_128x128_0.1							
							
0.026 / 29.8	0.029 / 29.8	0.0 / 29.0	0.125 / 32.6	0.0 / 28.5	0.0 / 28.4	0.005 / 28.6	0.053 / 30.0
causeway_bridge_middle_256x256_0.08							
							
0.009 / 34.1	0.123 / 38.0	0.0 / 33.8	0.263 / 42.7	0.0 / 32.7	0.003 / 32.8	0.035 / 33.8	0.242 / 40.6
causeway_bridge_low_64x64_0.12							
0.032 / 29.5	0.015 / 29.0	0.0 / 28.6	0.11 / 31.8	0.01 / 28.8	0.001 / 28.5	0.0 / 28.5	0.044 / 29.7
causeway_bridge_low_128x128_0.1							
							
0.03 / 31.7	0.028 / 31.7	0.0 / 30.8	0.184 / 36.5	0.003 / 30.4	0.0 / 30.3	0.015 / 30.8	0.125 / 34.1
causeway_bridge_low_256x256_0.08							
							
0.043 / 37.2	0.205 / 43.0	0.0 / 35.7	0.322 / 47.1	0.006 / 35.2	0.0 / 35.0	0.02 / 35.7	0.289 / 45.0
two_level_bridge_64x64_0.2							
							
0.01 / 19.0	0.008 / 18.9	0.0 / 18.8	0.048 / 19.7	0.0 / 18.5	0.0 / 18.5	0.002 / 18.5	0.032 / 19.1
two_level_bridge_128x128_0.16							
0.017 / 22.7	0.0 / 22.4	0.0 / 22.3	0.114 / 24.9	0.0 / 21.9	0.003 / 21.9	0.016 / 22.2	0.048 / 22.9
two_level_bridge_256x256_0.12							
							
0.006 / 28.9	0.004 / 28.8	0.0 / 28.7	0.256 / 36.1	0.0 / 27.7	0.007 / 27.9	0.037 / 28.7	0.214 / 33.6
free_suspended_bridge_64x64_0.15							
0.069 / 19.2	0.0 / 17.9	0.073 / 19.3	0.245 / 22.3	0.016 / 18.1	0.0 / 17.8	0.007 / 18.0	0.04 / 18.6
free_suspended_bridge_128x128_0.1							
							
0.019 / 25.0	0.0 / 24.5	0.137 / 27.9	0.813 / 44.4	0.0 / 23.9	0.005 / 24.0	0.041 / 24.8	0.145 / 27.3

Typical sample				Best of ensemble			
CNN-LBFGS	**MMA**	**OC**	**Pixel-LBFGS**	**CNN-LBFGS**	**MMA**	**OC**	**Pixel-LBFGS**
free_suspended_bridge_256x256_0.075							
0.0 / 30.4	0.021 / 31.1	0.2 / 36.5	1.594 / 78.9	0.0 / 28.9	0.005 / 29.0	0.132 / 32.7	0.338 / 38.6
free_suspended_bridge_256x256_0.05							
0.0 / 53.6	0.089 / 58.4	0.236 / 66.3	2.197 / 171.5	0.0 / 50.2	0.007 / 50.6	0.217 / 61.1	0.482 / 74.4
anchored_suspended_bridge_64x64_0.15							
0.0 / 21.6	0.013 / 21.9	0.088 / 23.5	0.046 / 22.6	0.003 / 21.1	0.0 / 21.1	0.002 / 21.1	0.042 / 21.9
anchored_suspended_bridge_128x128_0.1							
0.017 / 28.8	0.0 / 28.3	0.053 / 29.8	0.178 / 33.3	0.004 / 27.5	0.0 / 27.4	0.03 / 28.2	0.171 / 32.1
anchored_suspended_bridge_256x256_0.075							
0.026 / 34.2	0.0 / 33.3	0.131 / 37.7	0.457 / 48.6	0.0 / 32.4	0.016 / 32.9	0.104 / 35.8	0.364 / 44.2
anchored_suspended_bridge_256x256_0.05							
0.0 / 60.9	0.039 / 63.2	0.151 / 70.1	0.625 / 99.0	0.0 / 57.7	0.037 / 59.8	0.127 / 65.0	0.404 / 81.0
canyon_bridge_64x64_0.16							
0.018 / 14.3	0.0 / 14.0	0.025 / 14.4	0.018 / 14.3	0.002 / 13.6	0.0 / 13.6	0.005 / 13.7	0.036 / 14.1
canyon_bridge_128x128_0.12							
							
0.001 / 17.7	0.086 / 19.2	0.0 / 17.7	0.075 / 19.0	0.0 / 16.9	0.008 / 17.1	0.02 / 17.3	0.088 / 18.4
canyon_bridge_256x256_0.1							
0.025 / 19.1	0.0 / 18.7	0.038 / 19.4	0.211 / 22.6	0.0 / 18.3	0.007 / 18.5	0.045 / 19.2	0.179 / 21.6
canyon_bridge_256x256_0.05							
							
0.0 / 51.7	0.009 / 52.2	0.04 / 53.8	0.562 / 80.8	0.0 / 47.1	0.072 / 50.5	0.086 / 51.1	0.347 / 63.4

Typical sample				Best of ensemble			
CNN-LBFGS	**MMA**	**OC**	**Pixel-LBFGS**	**CNN-LBFGS**	**MMA**	**OC**	**Pixel-LBFGS**
thin_support_bridge_64x64_0.3							
0.0 / 54.9	0.058 / 58.0	0.143 / 62.7	0.122 / 61.5	0.0 / 53.6	0.0 / 53.6	0.005 / 53.9	0.01 / 54.2
thin_support_bridge_128x128_0.2							
							
0.0 / 78.7	0.026 / 80.8	0.065 / 83.8	0.331 / 104.7	0.0 / 75.7	0.006 / 76.1	0.018 / 77.0	0.054 / 79.8
thin_support_bridge_256x256_0.15							
							
0.007 / 97.1	0.0 / 96.5	0.018 / 98.2	0.379 / 133.0	0.0 / 91.9	0.036 / 95.2	0.069 / 98.2	0.224 / 112.5
thin_support_bridge_256x256_0.1							
							
0.028 / 170.8	0.0 / 166.2	0.115 / 185.3	0.594 / 264.9	0.0 / 156.3	0.063 / 166.2	0.117 / 174.7	0.349 / 210.9
drawbridge_64x64_0.2							
							
0.069 / 14.8	0.0 / 13.9	0.008 / 14.0	0.149 / 15.9	0.018 / 13.8	0.0 / 13.6	0.011 / 13.7	0.03 / 14.0
drawbridge_128x128_0.15							
							
0.013 / 18.2	0.004 / 18.0	0.0 / 17.9	0.196 / 21.5	0.019 / 17.0	0.0 / 16.7	0.036 / 17.3	0.063 / 17.8
drawbridge_256x256_0.1							
							
0.0 / 26.5	0.001 / 26.5	0.044 / 27.7	0.797 / 47.7	0.009 / 24.5	0.0 / 24.3	0.096 / 26.6	0.178 / 28.6
hoop_32x64_0.25							
							
0.0 / 0.8	0.051 / 0.8	0.159 / 0.9	0.402 / 1.1	0.0 / 0.8	0.0 / 0.8	0.008 / 0.8	0.014 / 0.8

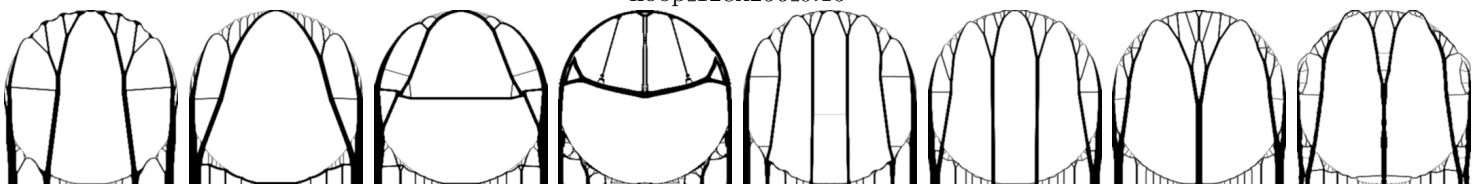
Typical sample				Best of ensemble			
CNN-LBFGS	**MMA**	**OC**	**Pixel-LBFGS**	**CNN-LBFGS**	**MMA**	**OC**	**Pixel-LBFGS**

hoop_64x128_0.2



0.0 / 0.9	0.063 / 0.9	0.183 / 1.1	0.648 / 1.5	0.012 / 0.9	0.0 / 0.9	0.035 / 0.9	0.068 / 0.9

hoop_128x256_0.15



0.0 / 1.1	0.079 / 1.2	0.252 / 1.4	1.198 / 2.5	0.007 / 1.1	0.0 / 1.1	0.041 / 1.1	0.144 / 1.2

dam_64x64_0.2

0.036 / 6.3	0.0 / 6.1	0.037 / 6.3	0.141 / 6.9	0.007 / 5.8	0.0 / 5.8	0.002 / 5.8	0.039 / 6.0

dam_128x128_0.15

0.047 / 7.9	0.0 / 7.6	0.071 / 8.1	0.307 / 9.9	0.008 / 7.3	0.0 / 7.2	0.03 / 7.4	0.066 / 7.7

dam_256x256_0.05

0.0 / 32.9	0.133 / 37.3	0.024 / 33.7	1.818 / 92.8	0.0 / 29.7	0.009 / 30.0	0.054 / 31.3	0.291 / 38.3

dam_256x256_0.1

0.0 / 11.5	0.054 / 12.1	0.069 / 12.3	0.653 / 19.0	0.004 / 10.7	0.0 / 10.7	0.05 / 11.2	0.203 / 12.8

Typical sample				Best of ensemble			
CNN-LBFGS	**MMA**	**OC**	**Pixel-LBFGS**	**CNN-LBFGS**	**MMA**	**OC**	**Pixel-LBFGS**

dam_256x256_0.2

0.021 / 4.5	0.0 / 4.4	0.113 / 5.0	0.287 / 5.7	0.007 / 4.4	0.0 / 4.4	0.041 / 4.5	0.142 / 5.0

ramp_64x64_0.3

0.041 / 2.9	0.0 / 2.8	0.014 / 2.8	0.04 / 2.9	0.008 / 2.8	0.0 / 2.8	0.004 / 2.8	0.013 / 2.8

ramp_128x128_0.2

0.029 / 4.5	0.0 / 4.4	0.033 / 4.5	0.1 / 4.8	0.012 / 4.3	0.0 / 4.2	0.004 / 4.3	0.077 / 4.6

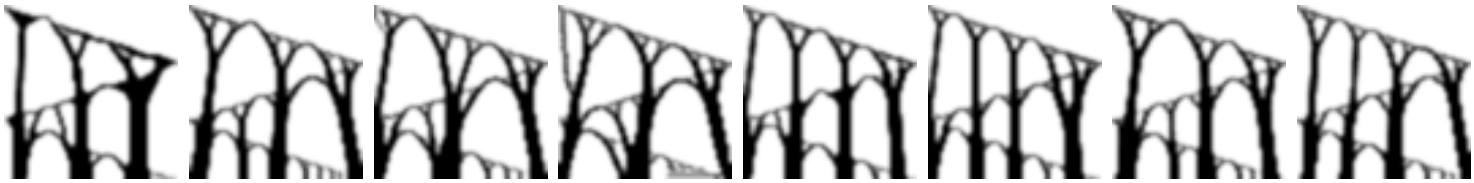
ramp_256x256_0.2

0.013 / 3.9	0.0 / 3.9	0.053 / 4.1	0.2 / 4.7	0.001 / 3.8	0.0 / 3.8	0.032 / 3.9	0.105 / 4.2

ramp_256x256_0.1

0.02 / 10.0	0.0 / 9.8	0.066 / 10.4	0.371 / 13.4	0.0 / 9.4	0.012 / 9.5	0.069 / 10.1	0.205 / 11.4

staircase_64x64_0.3



0.058 / 1.8	0.0 / 1.7	0.042 / 1.8	0.118 / 1.9	0.0 / 1.7	0.0 / 1.7	0.007 / 1.7	0.014 / 1.7

Typical sample				Best of ensemble			
CNN-LBFGS	**MMA**	**OC**	**Pixel-LBFGS**	**CNN-LBFGS**	**MMA**	**OC**	**Pixel-LBFGS**

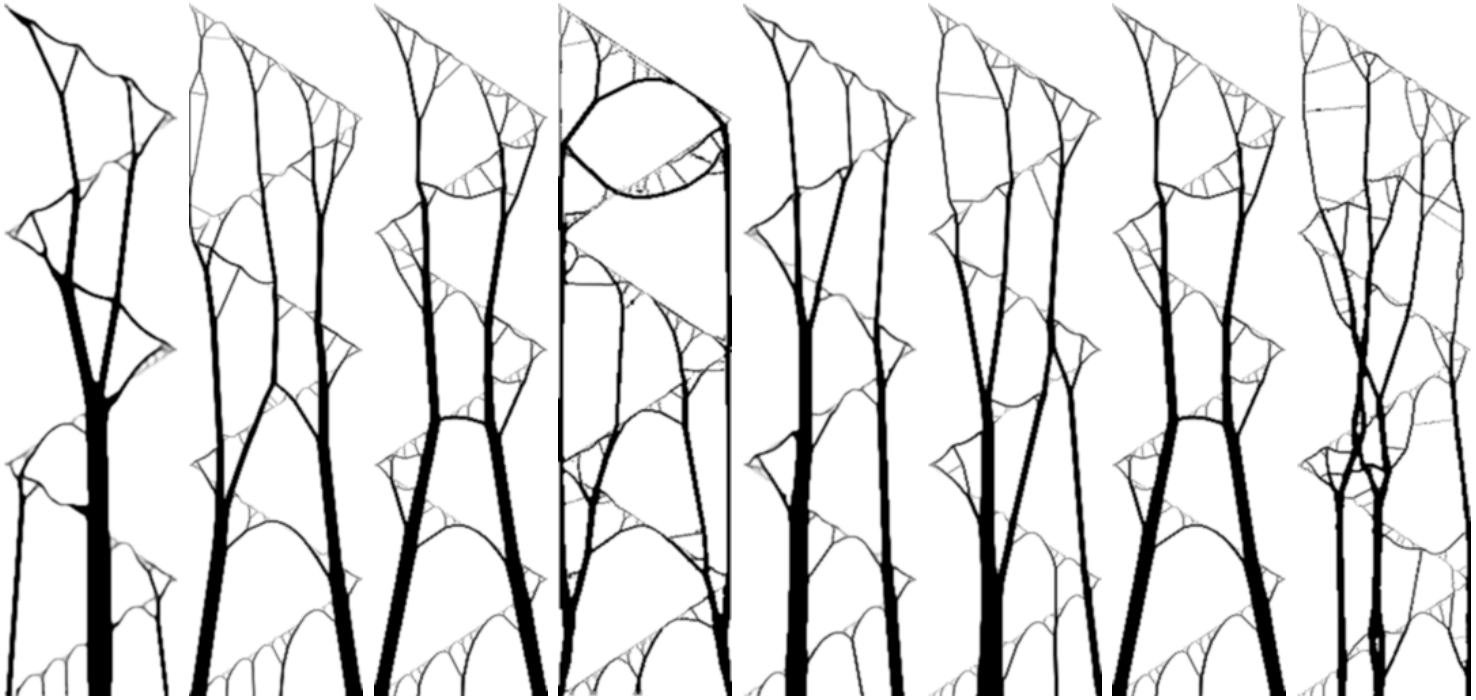
staircase_128x128_0.2

0.066 / 2.8	0.0 / 2.6	0.095 / 2.9	0.206 / 3.1	0.004 / 2.6	0.0 / 2.6	0.036 / 2.7	0.118 / 2.9

staircase_256x256_0.15

0.009 / 3.3	0.0 / 3.3	0.066 / 3.5	0.377 / 4.5	0.0 / 3.2	0.033 / 3.3	0.084 / 3.4	0.242 / 3.9

staircase_128x512_0.15

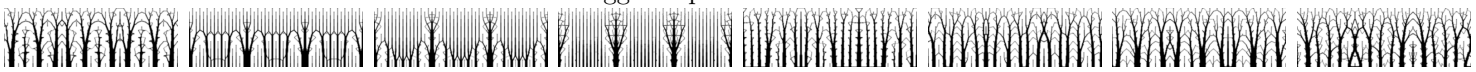


0.0 / 16.6	0.009 / 16.7	0.013 / 16.8	0.54 / 25.5	0.0 / 15.7	0.047 / 16.4	0.072 / 16.8	0.328 / 20.8

staggered_points_64x64_0.3

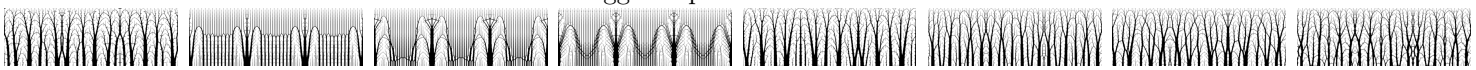
0.069 / 11.4	0.0 / 10.6	0.005 / 10.7	0.039 / 11.0	0.006 / 10.5	0.0 / 10.4	0.019 / 10.6	0.053 / 10.9

staggered_points_128x128_0.3

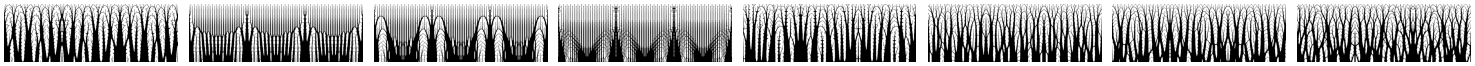
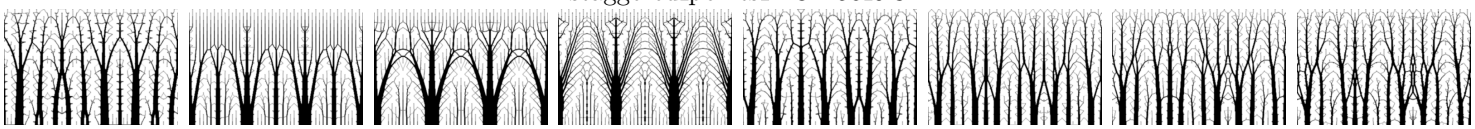
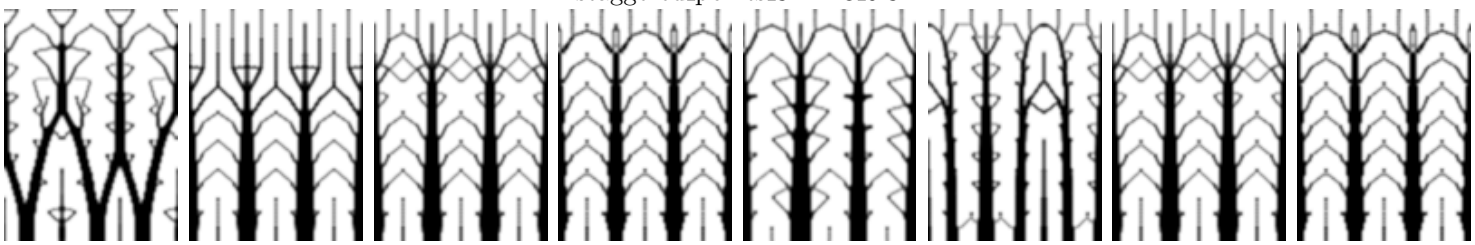
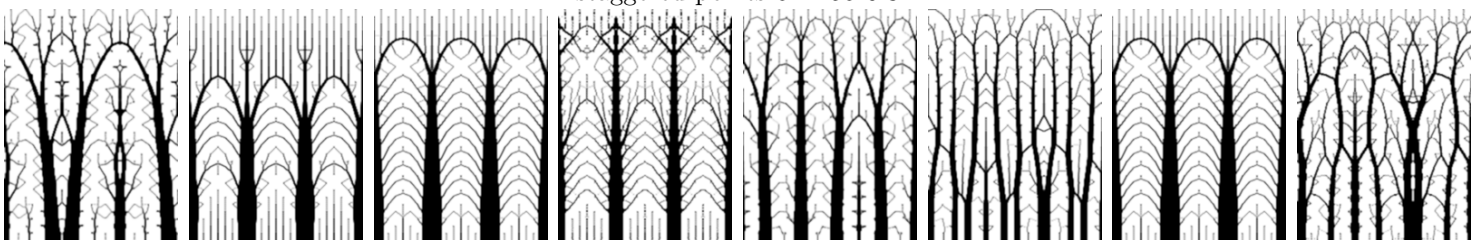
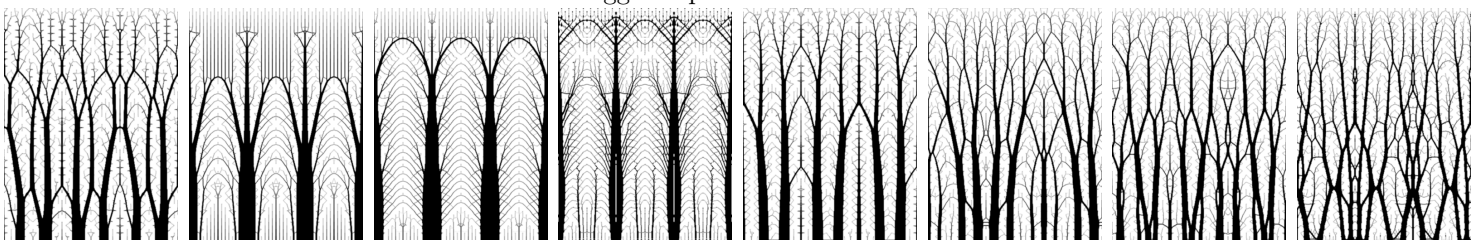


0.0 / 8.2	0.005 / 8.2	0.041 / 8.5	0.116 / 9.1	0.0 / 7.6	0.029 / 7.9	0.053 / 8.0	0.124 / 8.6

staggered_points_256x256_0.3

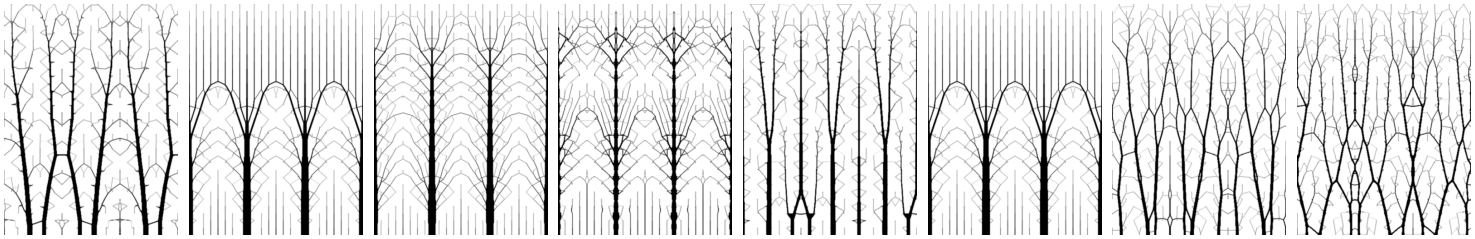


0.0 / 6.5	0.017 / 6.7	0.127 / 7.4	0.262 / 8.3	0.002 / 6.4	0.0 / 6.4	0.025 / 6.6	0.092 / 7.0

Typical sample				Best of ensemble			
CNN-LBFGS	**MMA**	**OC**	**Pixel-LBFGS**	**CNN-LBFGS**	**MMA**	**OC**	**Pixel-LBFGS**
staggered_points_256x256_0.5							
							
0.0 / 2.9	0.028 / 3.0	0.038 / 3.0	0.153 / 3.4	0.0 / 2.9	0.002 / 2.9	0.017 / 2.9	0.056 / 3.1
staggered_points_64x128_0.3							
0.045 / 18.4	0.0 / 17.6	0.073 / 18.9	0.185 / 20.9	0.007 / 17.2	0.0 / 17.1	0.029 / 17.6	0.068 / 18.3
staggered_points_128x256_0.3							
							
0.008 / 14.0	0.0 / 13.8	0.086 / 15.0	0.286 / 17.8	0.012 / 13.5	0.0 / 13.3	0.02 / 13.6	0.086 / 14.5
staggered_points_32x128_0.3							
							
0.097 / 45.4	0.008 / 41.7	0.0 / 41.3	0.016 / 42.0	0.021 / 41.3	0.0 / 40.4	0.022 / 41.3	0.039 / 42.0
staggered_points_64x256_0.3							
							
0.026 / 31.3	0.001 / 30.5	0.0 / 30.5	0.181 / 36.0	0.0 / 29.8	0.008 / 30.0	0.023 / 30.5	0.086 / 32.4
staggered_points_128x512_0.3							
							
0.0 / 24.7	0.012 / 24.9	0.045 / 25.8	0.293 / 31.9	0.0 / 23.9	0.021 / 24.4	0.041 / 24.8	0.117 / 26.7

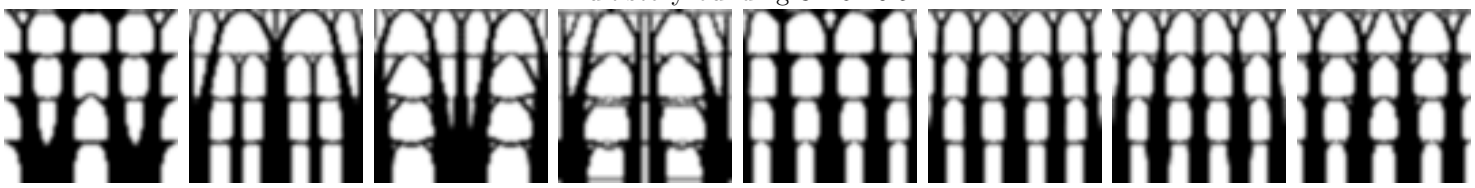
Typical sample				Best of ensemble			
CNN-LBFGS	**MMA**	**OC**	**Pixel-LBFGS**	**CNN-LBFGS**	**MMA**	**OC**	**Pixel-LBFGS**

staggered_points_128x512_0.15



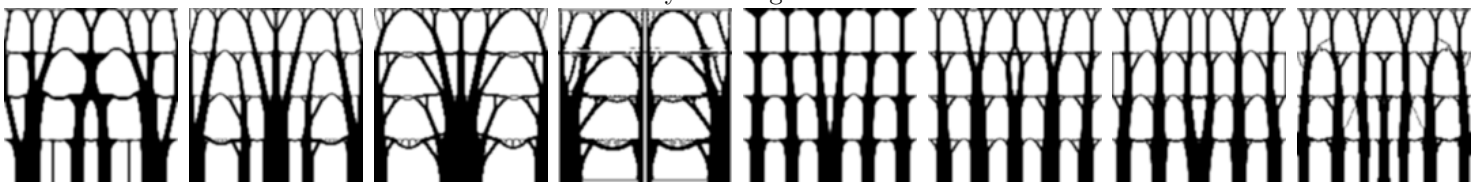
0.025 / 74.8	0.0 / 73.0	0.094 / 79.8	0.4 / 102.2	0.0 / 70.0	0.043 / 73.0	0.095 / 76.6	0.239 / 86.7

multistory_building_32x64_0.5



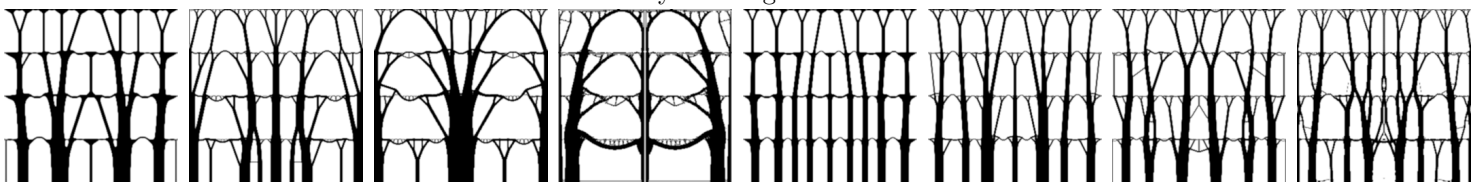
0.023 / 41.5	0.0 / 40.6	0.034 / 42.0	0.12 / 45.5	0.011 / 39.6	0.0 / 39.1	0.008 / 39.4	0.019 / 39.9

multistory_building_64x128_0.4



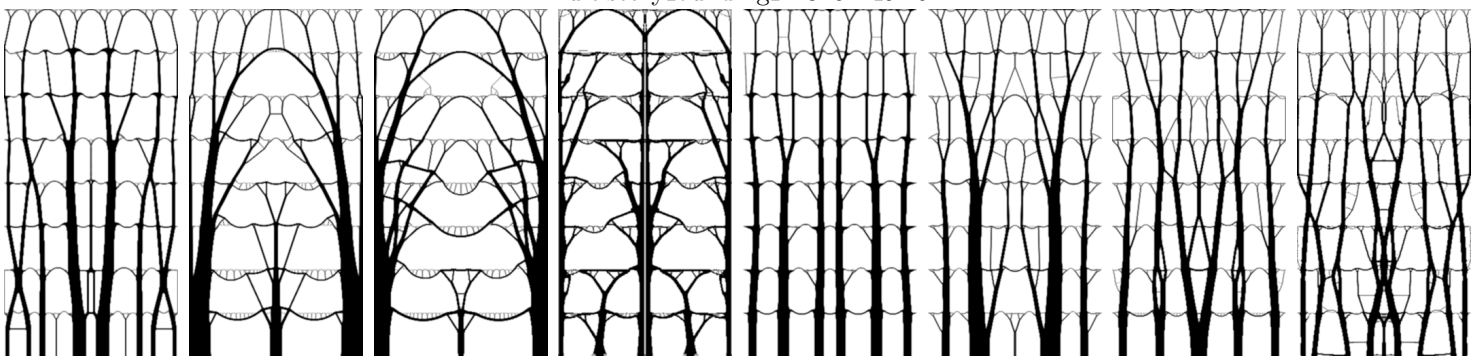
0.028 / 49.5	0.0 / 48.1	0.103 / 53.1	0.369 / 65.9	0.009 / 46.8	0.0 / 46.4	0.005 / 46.6	0.046 / 48.5

multistory_building_128x256_0.3



0.0 / 63.0	0.002 / 63.1	0.107 / 69.7	0.501 / 94.5	0.008 / 60.2	0.0 / 59.8	0.049 / 62.7	0.096 / 65.5

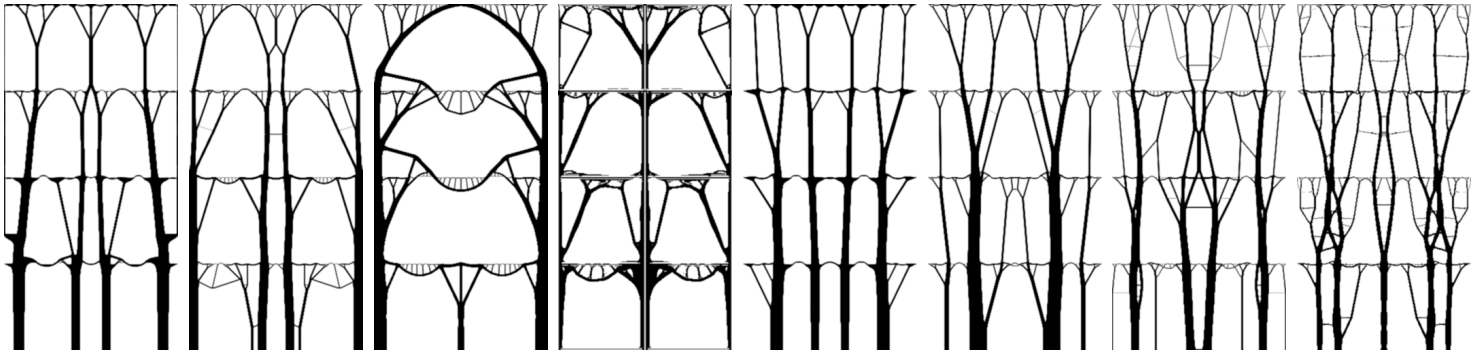
multistory_building_128x512_0.25



0.0 / 493.6	0.069 / 527.8	0.267 / 625.4	0.532 / 756.4	0.0 / 470.0	0.031 / 484.7	0.053 / 494.8	0.173 / 551.2

Typical sample				Best of ensemble			
CNN-LBFGS	**MMA**	**OC**	**Pixel-LBFGS**	**CNN-LBFGS**	**MMA**	**OC**	**Pixel-LBFGS**

multistory_building_128x512_0.2



0.002 / 188.6	0.0 / 188.1	0.367 / 257.2	0.916 / 360.4	0.0 / 181.8	0.007 / 183.1	0.089 / 198.0	0.206 / 219.3