Neural networks for topology optimization

Article *in* Russian Journal of Numerical Analysis and Mathematical Modelling · September 2017

DOI: 10.1515/rnam-2019-0018

CITATIONS
193

READS
932

2 authors, including:

Ivan Oseledets
Skolkovo Institute of Science and Technology
346 PUBLICATIONS **10,320** CITATIONS

SEE PROFILE

Ivan Sosnovik and Ivan Oseledets*

Neural networks for topology optimization

https://doi.org/10.1515/rnam-2019-0018
Received February 11, 2019; accepted May 21, 2019

Abstract: In this research, we propose a deep learning based approach for speeding up the topology optimization methods. The problem we seek to solve is the layout problem. The main novelty of this work is to state the problem as an image segmentation task. We leverage the power of deep learning methods as the efficient pixel-wise image labeling technique to perform the topology optimization. We introduce convolutional encoder-decoder architecture and the overall approach of solving the above-described problem with high performance. The conducted experiments demonstrate the significant acceleration of the optimization process. The proposed approach has excellent generalization properties. We demonstrate the ability of the application of the proposed model to other problems. The successful results, as well as the drawbacks of the current method, are discussed.

Keywords: Deep learning, topology optimization, image segmentation.

MSC 2010: 49M99, 78M50, 65N99

Topology optimization solves the layout problem with the following formulation: how to distribute the material inside a design domain such that the obtained structure has optimal properties and satisfies the prescribed constraints? The most challenging formulation of the problem requires the solution to be binary, i.e., it should state whether there is a material or a void for each of the parts of the design domain. One of the common examples of such an optimization is the minimization of elastic strain energy of a body for a given total weight and boundary conditions. Initiated by the demands of automotive and aerospace industry in the 20^{th} century, topology optimization has spread its application to a wide range of other disciplines: e.g. fluids, acoustics, electromagnetics, optics and combinations thereof [5].

All modern approaches for topology optimization used in commercial and academic software are based on finite element methods. SIMP (Simplified Isotropic Material with Penalization), which was introduced in 1989 [3], is currently a widely spread simple and efficient technique. It proposes to use penalization of the intermediate values of density of the material, which improves the convergence of the solution to binary. Topology optimization problem could be solved by using BESO (Bi-directional evolutionary structural optimization) [14, 25] as an alternative. The key idea of this method is to remove the material where the stress is the lowest and add material where the stress is higher. The more detailed review is given in Section 1.

For all of the above-described methods, the process of optimization could be roughly divided into two stages: general redistribution of the material and the refinement. During the first one, the material layout varies a lot from iteration to iteration. While during the second stage the material distribution converges to the final result. The global structure remains unchanged and only local alteration could be observed.

In this paper, we propose a deep learning based approach to speeding up the most time-consuming part of a traditional topology optimization methods. The main novelty of this work is to state the problem as an image segmentation task. We leverage the power of deep learning methods as an efficient pixel-wise image labeling technique to accelerate modern topology optimization solvers. The key features of our approach are the following:

- acceleration of optimization process;
- excellent generalization properties;
- absolutely scalable techniques.

Ivan Sosnovik, University of Amsterdam, The Netherlands; Skolkovo Institute of Science and Technology, Moscow 121205, Russia

***Corresponding author: Ivan Oseledets,** Skolkovo Institute of Science and Technology, Moscow 121205, Russia; Marchuk Institute of Numerical Mathematics, Moscow 119333, Russia. E-mail: i.oseledets@skoltech.ru

1 Topology optimization problem

Current research is devoted to topology optimization of mechanical structures. Consider a design domain $\Omega : \{\omega_j\}_{j=1}^N$, filled with a linear isotropic elastic material and discretized with square finite elements. The material distribution is described by the binary density variable x_j that represents either absence (0) or presence (1) of the material at each point of the design domain. Therefore, the problem, that we seek to solve, can be written in mathematical form as:

$$\begin{cases} \min\limits_{\boldsymbol{x}} \; c(\boldsymbol{u}(\boldsymbol{x}), \boldsymbol{x}) = \sum_{j=1}^{N} E_j(x_j) \boldsymbol{u}_j^T \boldsymbol{k_0} \boldsymbol{u}_j \\ \text{s.t.} \;\; V(\boldsymbol{x})/V_0 = f_0 \\ \boldsymbol{KU} = \boldsymbol{F} \\ x_j \in \{0; 1\}, \quad j = 1, \dots, N \end{cases} \tag{1.1}$$

where c is a compliance, $\boldsymbol{u_j}$ is the element displacement vector, $\boldsymbol{k_0}$ is the element stiffness matrix for an element with unit Young's modulus, $\boldsymbol{U}$ and $\boldsymbol{F}$ are the global displacement and force vectors, respectively, and $\boldsymbol{K}$ is the global stiffness matrix; $V(\boldsymbol{x})$ and V_0 are the material volume and design domain volume, respectively; f_0 is the prescribed volume fraction.

The discrete nature of the problem makes it difficult to solve. Therefore, the last constraint in (1.1) is replaced with the following one: $x_j \in [0; 1], \; j = 1, \dots, N$. The most common method for topology optimization problem with continuous design variables is so-called SIMP or power-law approach [3, 16]. This is a gradient-based iterative method with the penalization of non-binary solutions, which is achieved by choosing Young's modulus of a simple but very efficient form:

$$E_j(x_j) = E_{\min} + x_j^p (E_0 - E_{\min}). \tag{1.2}$$

The exact implementation of SIMP algorithm is out of the scope of the current paper. The updating schemes, as well as different heuristics, can be found in excellent papers [4, 6, 10, 19, 22]. The topology optimization code in Matlab is described in detail in [2, 20] and the Python implementation of SIMP algorithm is represented in [11].

Standard half MBB-Beam problem is used to illustrate the process of topology optimization. The design domain, constraints, and loads are represented in Fig. 1. The optimization of this problem is demonstrated in Fig. 2. During the initial iterations, the general redistribution of the material inside of the design domain is performed. The rest of the optimization process includes the filtering of the pixels: the densities with intermediate values converge to binary values and the silhouette of the obtained structure remains almost unchanged.

2 Learning topology optimization

As it was illustrated in Section 1, it is enough for the solver to perform a few number N_0 of iterations to obtain the preliminary view of a structure. The fraction of non-binary densities could be close to 1, however, the global layout pattern is close to the final one. The obtained image I could be interpreted as a blurred image of a final structure, or an image distorted by other factors. The thing is that there are just two types of objects on this image: the material and the void. The image I^*, obtained as a result of topology optimization does not contain intermediate values and, therefore, could be interpreted as the mask of image I. According to this notation, starting from iteration N_0 the process of optimization $I \rightarrow I^*$ mimics the process of image segmentation for two classes or foreground-background segmentation.

We propose the following pipeline for topology optimization: use SIMP method to perform the initial iterations and get the distribution with non-binary densities; use the neural network to perform the segmentation of the obtained image and converge the distribution to $\{0, 1\}$ solution.

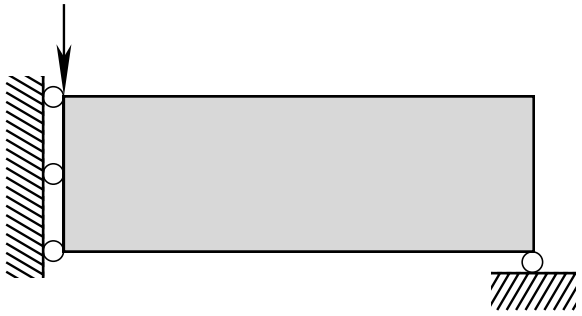
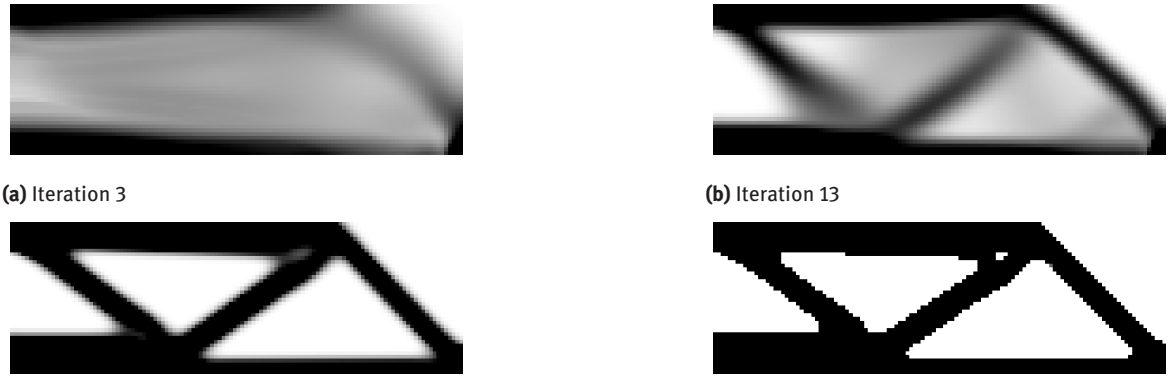


Fig. 1: The design domain, boundary conditions, and external load for the optimization of a half MBB beam.



(a) Iteration 3

(b) Iteration 13

(c) Iteration 30

(d) Iteration 80

Fig. 2: The process of optimization of half MBB beam with SIMP method for 120 × 40 mesh. Black — 1, white — 0.

2.1 Architecture

Here we introduce the *Neural Network for Topology Optimization* — deep fully-convolutional neural network aimed to perform the convergence of densities during the topology optimization process. The input of the model is two grayscale images (or a two-channel image). The first one is the density distribution X_n inside of the design domain which was obtained after the last performed iteration of topology optimization solver. The second input is the last performed update (gradient) of the densities $\delta X = X_n - X_{n-1}$, i.e., the difference between the densities after the nth iteration and $(n-1)$th iteration. The output of the proposed model is a grayscale image of the same resolution as an input, which represents the predicted final structure. The architecture of our model mimics the common for the image segmentation hourglass shape. The proposed model has an encoder network and a corresponding decoder network, followed by a final pixel-wise classification layer. The architecture is illustrated in Fig. 3.

The encoder network consists of 6 convolutional layers. Each layer has kernels of size 3×3 and is followed by ReLU nonlinearity. The first two layers have 16 convolutional kernels. This block is followed by the pooling of the maximal element from the window of size 2×2. The next two layers have 32 kernels and are also followed by MaxPooling layer. The last block consists of 2 layers with 64 kernels each.

The decoder network copies the architecture of the encoder part and reverses it. The MaxPooling layers are replaced with Upsampling layers followed by the concatenation with features from the corresponding low-level layer as it is performed in U-Net [18]. The pooling operation introduces the invariance of the subsequent network to small translations of the input. The concatenation of features from different layers allows one to benefit from the use of both the raw low-level representation and significantly encoded parameterization from the higher levels. The decoder is followed by the Convolutional layer with 1 kernel and sigmoid activation function. We included 2 Dropout layers [12] as the regularization for our network.

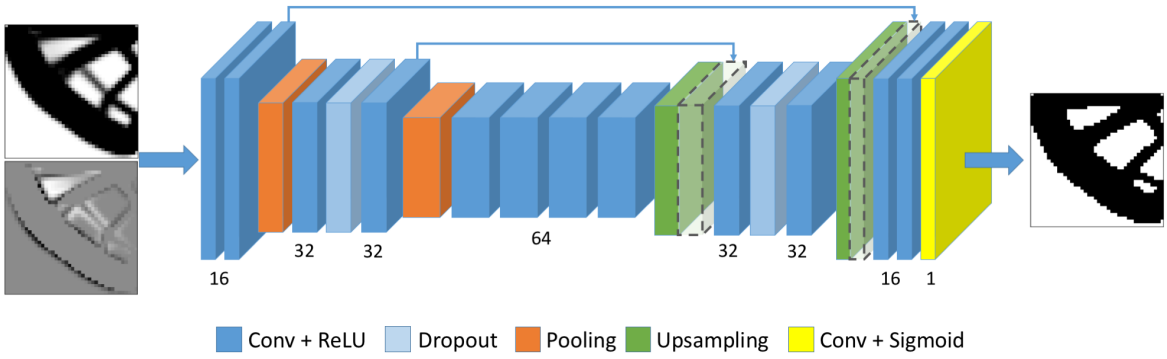


Fig. 3: The architecture of the proposed neural network for topology optimization. All kernels are of size 3×3. The number of kernels is represented by the number at the bottom of the layer. Blue arrows and opaque boxes represent the concatenation of the features from different layers.

The width and height of the input image could vary, however, they must be divisible by 4 in order to guarantee the coherence of the shapes of tensors in the computational graph. The proposed neural network has just 192,113 parameters.

2.2 Dataset

To train the above-described model, we need example solutions to system (1.1). To collect a large dataset from the real-life examples is difficult or even impossible. Therefore, we use synthetic data generated by using ToPy [11] — an open source solver for 2D and 3D topology optimization, based on SIMP approach.

To generate the dataset we sampled the pseudo-random problem formulations and performed 100 iterations of standard SIMP method. Each problem is defined by the constraints and the loads. The strategy of sampling is the following:

- The number of nodes with fixed x and y translations and the number of loads are sampled from the Poisson distribution:

$$N_x \sim P(\lambda = 2)$$
$$N_y, N_{\mathrm{L}} \sim P(\lambda = 1).$$

- The nodes for each of the above-described constraints are sampled from the distribution defined on the grid. The probability of choosing the boundary node is 100 times higher than that for an inner node.
- The load values are chosen as -1.
- The volume fraction is sampled from the Normal distribution $f_0 \sim \mathcal{N}(\mu = 0.5, \sigma = 0.1)$.

The dataset and the related code are available at https://github.com/ISosnovik/top. The obtained dataset has 10,000 objects. Each object is a tensor of shape $100 \times 40 \times 40$: 100 iterations of the optimization process for the problem defined on a regular 40×40 grid.

2.3 Training

We used dataset, described in Section 2.2, to train our model. During the training process we 'stopped' SIMP solver after k iterations and used the obtained design variables as an input for our model. The input images were augmented with transformations from group D4: horizontal and vertical flips and rotation by 90 degrees. k was sampled from some certain distribution $\mathcal{F}$. Poisson distribution $P(\lambda)$ and discrete uniform distribution

$U[1, 100]$ are of interest to us. For training the network we used the objective function of the following form:

$$\mathcal{L} = \mathcal{L}_{\text{conf}}(X_{\text{true}}, X_{\text{pred}}) + \beta \mathcal{L}_{\text{vol}}(X_{\text{true}}, X_{\text{pred}}) \tag{2.1}$$

where the confidence loss is a binary cross-entropy:

$$\mathcal{L}_{\text{conf}}(X_{\text{true}}, X_{\text{pred}}) = -\frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \left[X_{\text{true}}^{ij} \log(X_{\text{pred}}^{ij}) + (1 - X_{\text{true}}^{ij}) \log(1 - X_{\text{pred}}^{ij}) \right] \tag{2.2}$$

where $N \times M$ is the resolution of the image. The second summand in (2.1) represents the volume fraction constraint:

$$\mathcal{L}_{\text{vol}}(X_{\text{true}}, X_{\text{pred}}) = (\bar{X}_{\text{pred}} - \bar{X}_{\text{true}})^2. \tag{2.3}$$

We used ADAM [13] optimizer with default parameters. We halved the learning rate once during the training process. The implementation in Python is available at https://github.com/ISosnovik/nn4topopt. For neural networks, we used Keras [8] with TensorFlow [1] backend. NVIDIA Tesla K80 was used for deep learning computations. The training of a neural network from scratch took about 80-90 minutes.

3 Results

The goal of our experiments is to demonstrate that the proposed model and the overall pipeline are useful for solving topology optimization problems. We compare the performance of our approach with standard SIMP solver [11] in terms of the accuracy of the obtained structure and the average time consumption.

We report two metrics from common image segmentation evaluation: Binary Accuracy and Intersection over Union (IoU). Let n_l, $l = 0, 1$, be the total number of pixels of class l. The ω_{tp}, $t, p = 0, 1$, is a total number of pixels of class t predicted to belong to class p. Therefore:

$$\text{Bin. Acc.} = \frac{\omega_{00} + \omega_{11}}{n_0 + n_1}, \qquad \text{IoU} = \frac{1}{2}\Big[\frac{\omega_{00}}{n_0 + \omega_{10}} + \frac{\omega_{11}}{n_1 + \omega_{01}} \Big]. \tag{3.1}$$

We examine 4 neural networks with the same architecture but trained with different policies. The number of iterations after which we 'stopped' SIMP algorithm was sampled from different distributions. We trained one neural network by choosing discrete uniform distribution $U[1, 100]$ and another three models were trained with Poisson distribution $P(\lambda)$ with $\lambda = 5, 10, 30$.

3.1 Accuracy and performance

We conducted several experiments to illustrate the results of the application of the proposed pipeline and the exact model to mechanical problems. Figure 4 demonstrates that our neural network restores the final structure while being used even after 5 iterations. The output of the model is close to that of SIMP algorithm. The overall topology of the structure is the same. Furthermore, the time consumption of the proposed method, in this case, is almost 20 times smaller.

Neural networks trained with different policies produce close results: models preserve the final structure up to some rare pixel-wise changes. However, the accuracy of these models depends on the number of the initial iterations performed by SIMP algorithm. Tables 1 and 2 summarize the results obtained in the series of experiments. The trained models demonstrate the sufficiently more accurate results comparing to the thresholding applied after the same number of iterations of SIMP method. Some models benefit when they are applied after 5-10 iterations, while others demonstrate better result in the middle or at the end of the process. The proposed pipeline could significantly accelerate the overall algorithm with minimal reduction in accuracy, especially when CNN is used at the beginning of the optimization process.

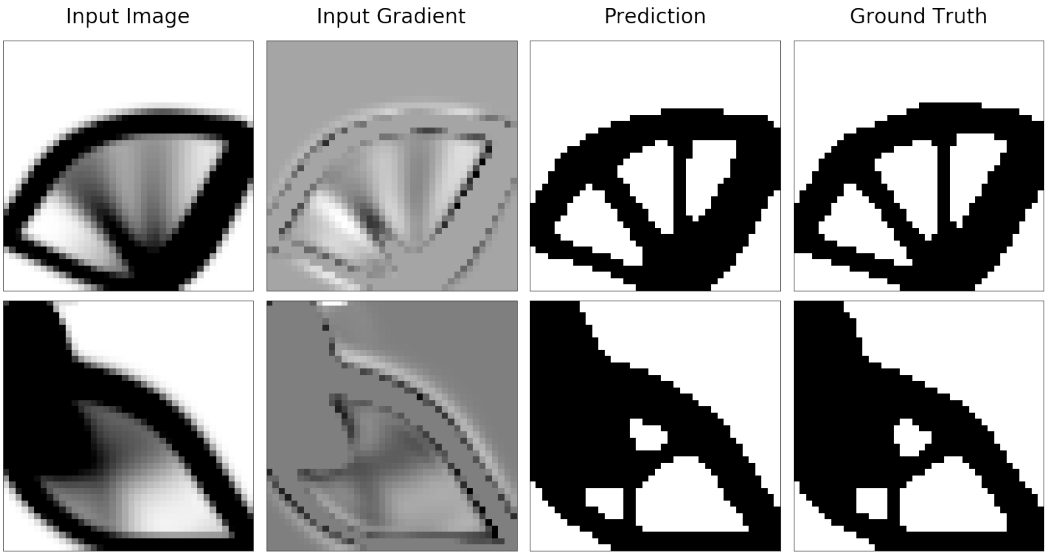


Fig. 4: Top: SIMP is stopped after 8 iterations, binary accuracy 0.96, mean IoU 0.92; Bottom: solver is stopped after 5 iterations, binary accuracy 0.98, mean IoU 0.95.

Tab. 1: Binary Accuracy of the proposed method and the standard one on the mechanical dataset.

	Iteration								
Method	**5**	**10**	**15**	**20**	**30**	**40**	**50**	**60**	**80**
Thresholding	92.9	95.4	96.5	97.1	97.7	98.1	98.4	98.6	98.9
CNN $P(5)$	**95.8**	97.3	97.7	97.9	98.2	98.4	98.5	98.6	98.7
CNN $P(10)$	95.4	**97.6**	**98.1**	98.4	98.7	98.9	99.0	99.0	99.0
CNN $P(30)$	92.7	96.3	97.8	**98.5**	**99.0**	**99.2**	**99.4**	**99.5**	99.6
CNN $U[1, 100]$	94.7	96.8	97.7	98.2	98.7	99.0	99.3	99.4	**99.6**

Tab. 2: Intersection over Union (IoU) of the proposed method and the standard one on the mechanical dataset.

	Iteration								
Method	**5**	**10**	**15**	**20**	**30**	**40**	**50**	**60**	**80**
Thresholding	86.8	91.2	93.3	94.3	95.6	96.3	96.8	97.3	97.9
CNN $P(5)$	**92.0**	94.7	95.4	96.0	96.5	96.9	97.1	97.3	97.4
CNN $P(10)$	91.1	**95.3**	**96.4**	96.9	97.4	97.8	98.0	98.0	98.1
CNN $P(30)$	86.4	92.9	95.7	**97.0**	**98.1**	**98.5**	**98.8**	**99.0**	99.2
CNN $U[1, 100]$	90.0	93.9	95.5	96.4	97.5	98.1	98.6	98.8	**99.2**

The neural network which used discrete uniform distribution during the training process does not demonstrate the highest binary accuracy and IoU comparing to other models till the latest iterations. However, this model allows one to outperform the SIMP algorithm with thresholding throughout the optimization process.

3.2 Transferability

This research deals with the application of neural networks to the topology optimization of minimal compliance problems. Nevertheless, the proposed model does not rely on any prior knowledge of the nature of the problem. Despite the fact that we used mechanical dataset during the training, other types of problems from topology optimization framework could be solved by using the proposed pipeline. To examine the generaliza-

Tab. 3: Binary Accuracy of the proposed method and the standard one on heat conduction dataset. Models were trained on the minimal compliance dataset.

	Iteration								
Method	**5**	**10**	**15**	**20**	**30**	**40**	**50**	**60**	**80**
Thresholding	97.5	98.4	98.8	99.1	99.4	**99.6**	**99.7**	**99.8**	**99.9**
CNN $P(5)$	98.1	98.7	99.0	99.2	99.4	99.5	99.6	99.7	99.7
CNN $P(10)$	**98.1**	98.8	99.0	99.2	99.4	99.5	99.6	99.7	99.8
CNN $P(30)$	97.3	**99.0**	**99.2**	**99.4**	**99.5**	99.6	99.7	99.7	99.8
CNN $U[1, 100]$	97.8	98.8	99.1	99.3	99.5	99.6	99.7	99.7	99.8

Tab. 4: IoU of the proposed method and the standard one on heat conduction dataset. Models were trained on the minimal compliance dataset.

	Iteration								
Method	**5**	**10**	**15**	**20**	**30**	**40**	**50**	**60**	**80**
Thresholding	95.1	96.8	97.6	98.1	98.8	**99.2**	**99.4**	**99.6**	**99.9**
CNN $P(5)$	96.2	97.5	98.0	98.4	98.8	99.0	99.2	99.3	99.5
CNN $P(10)$	**96.3**	97.6	98.1	98.4	98.9	99.1	99.3	99.4	99.5
CNN $P(30)$	94.8	**98.0**	**98.5**	**98.7**	**99.0**	99.2	99.3	99.4	99.5
CNN $U[1, 100]$	95.7	97.7	98.2	98.6	98.9	99.2	99.3	99.4	99.6

tion properties of our model, we generated the small dataset of heat conduction problems defined on 40×40 regular grid. The exact solution and the intermediate densities for the problems were obtained in absolutely the same way as it was described in Section 2.

The conducted experiments are summarized in Tables 3 and 4. During the initial part of the optimization process, the results of the pre-trained CNNs are more accurate than this of thresholding. Our model approximates the mapping to the final structure precisely when the training dataset and the validation dataset are from the same distribution. However, it mimics updates of SIMP method during the initial iterations even when CNN is applied to another dataset. Therefore, this pipeline could be useful for the fast prediction of the rough structure in various topology optimization problems.

The neural network described in Section 2 is fully-convolutional, i.e., it consists of Convolutional, Pooling, Upsampling, and Dropout layers. The architecture itself does not have any constraints on the size of the input data. In this experiment, we checked the scalable properties of our method. The model we examined has been trained on the original dataset with square images of size 40×40. Figure 5 visualizes the result of the application of CNN to the problems defined on grids with different resolution. Here we can see that changes in the aspect ratio and reasonable changes in the resolution of the input data do not affect the accuracy of the model. Pre-trained neural network successfully reconstructs the final structure for a given problem. Significant changes of the size of the input data require additional training of the model because the typical size of a common patterns changes with the increase of the resolution of an image. Nevertheless, demonstrated cases did not require one to tune neural network and allowed to transfer model from one resolution to another.

4 Related work

The current research is supposed to be the first one which utilizes deep learning approach for the topology optimization problem. It is inspired by the recent successful application of deep learning to problems in computational physics. Greff et al. [9] used the fully-connected neural network as a mapping function from the nano-material configuration and the input voltage to the output current. The adaptation of restricted Boltzmann machine for solving the Quantum Many-Body Problem was demonstrated in [7]. Mills et al. [15] used the machinery of deep learning to learn the mapping between potential and energy, bypassing the need to

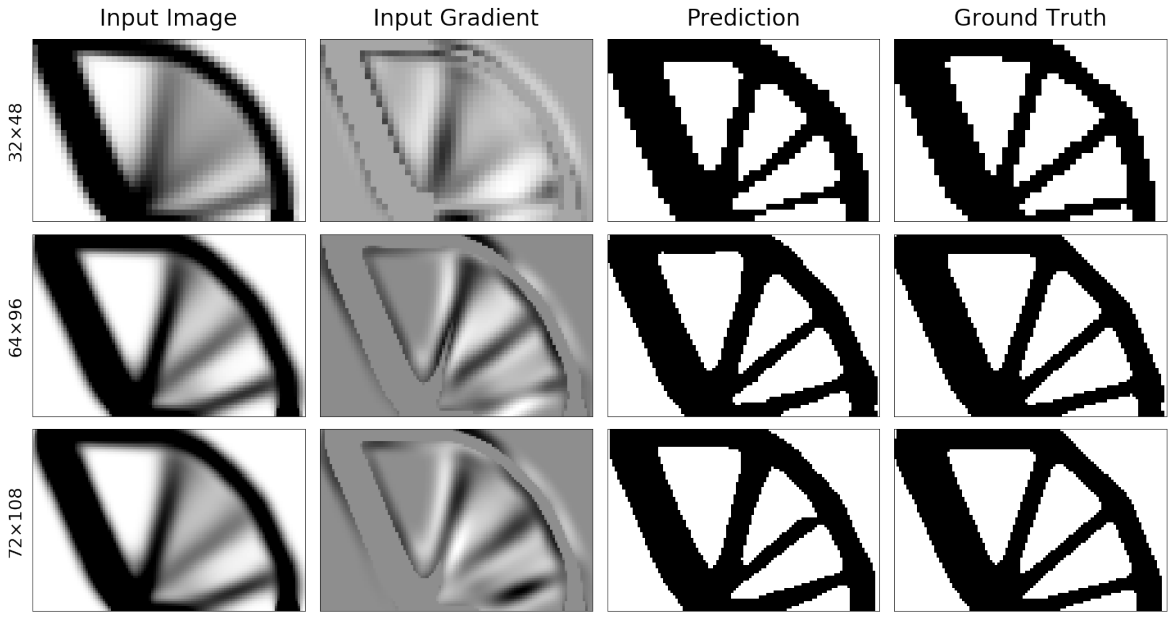


Fig. 5: Results of the application of the proposed CNN to the problems defined on grids with resolution and aspect ratio different from that of the training dataset.

numerically solve the Schrödinger equation and the need for computing wave functions. Tompson et al. [23] and Kiwon et al. [24] accelerated the process of modelling of liquids by the application of neural networks. The paper [21] demonstrates how a deep neural network trained on quantum mechanical density functional theory calculations can learn an accurate and transferable potential for organic molecules. The cutting-edge research [17] shows how generative adversarial networks could be used for simulating 3D high-energy particle showers in multi-layer electromagnetic calorimeters.

5 Conclusion

In this paper, we proposed a neural network as an effective tool for the acceleration of topology optimization process. Out model learned the mapping from the intermediate result of the iterative method to the final structure of the design domain. It allowed us to stop SIMP method earlier and significantly decrease the total time consumption.

We demonstrated that the model trained on the dataset of minimal compliance problems could produce the rough approximation of the solution for other types of topology optimization problems. Various experiments have shown that the proposed neural network transfers successfully from the dataset with a small resolution to the problems defined on the grids with better resolution.

Funding: This study was supported by the Ministry of Education and Science of the Russian Federation (grant 14.756.31.0001).

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin et al., Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

[2] E. Andreassen, A. Clausen, M. Schevenels, B. S. Lazarov, and O. Sigmund, Efficient topology optimization in MATLAB using 88 lines of code. *Struct. Multidiscip. Optimiz.* **43** (2011), No. 1, 1–16.
[3] M. P. Bendsøe, Optimal shape design as a material distribution problem. *Struct. Multidiscip. Optimiz.* **1** (1989), No. 4, 193–202.
[4] M. P. Bendsøe, *Optimization of Structural Topology, Shape, and Material*. Springer, **414**, 1995.
[5] M. P. Bendsøe, E. Lund, N. Olhoff, and O. Sigmund, Topology optimization-broadening the areas of application. *Control Cybern.* **34** (2005), 7–35.
[6] B. Bourdin, Filters in topology optimization. *Int. J. Numer. Meth. Engrg.* **50** (2001), No. 9, 2143–2158.
[7] G. Carleo and M. Troyer, Solving the quantum many-body problem with artificial neural networks. *Science* **355** (2017), No. 6325, 602–606.
[8] F. Chollet et al, Keras. GitHub, 2015. URL: https://github.com/fchollet/keras.
[9] K. Greff, R. M. J. van Damme, J. Koutnik, H. J. Broersma, J. Mikhal, C. P. Lawrence, W. G. van der Wiel, and J. Schmidhuber, Using neural networks to predict the functionality of reconfigurable nano-material networks. *Int. J. Advances in Intelligent Systems* **9** (2017), No. 3–4, 339–351.
[10] A. A. Groenwold and L. F. P. Etman, A simple heuristic for gray-scale suppression in optimality criterion-based topology optimization. *Struct. Multidiscip. Optimiz.* **39** (2009), No. 2, 217–225.
[11] W. Hunter et al., ToPy–Topology optimization with Python, 2017. URL: https://github.com/williamhunter/topy.
[12] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.
[13] D. Kingma and J. Ba, Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
[14] C. Mattheck and S. Burkhardt, A new method of structural shape optimization based on biological growth. *Int. J. Fatigue* **12** (1990), No. 3, 185–190.
[15] K. Mills, M. Spanner, and I. Tamblyn, Deep learning and the Schrödinger equation. *arXiv preprint arXiv:1702.01361*, 2017.
[16] H. P. Mlejnek, Some aspects of the genesis of structures. *Struct. Multidiscip. Optimiz.* **5** (1992), No. 1, 64–69.
[17] M. Paganini, L. de Oliveira, and B. Nachman, CaloGAN: simulating 3D high energy particle showers in multi-layer electromagnetic calorimeters with generative adversarial networks. *arXiv preprint arXiv:1705.02355*, 2017.
[18] O. Ronneberger, P. Fischer, and T. Brox, U-net: Convolutional networks for biomedical image segmentation. *Int. Conf. on Medical Image Computing and Computer-Assisted Intervention*. Springer, Munich, 2015, pp. 234–241.
[19] O. Sigmund, On the design of compliant mechanisms using topology optimization. *J. Structural Mechanics* **25** (1997), No. 4, 493–524.
[20] O. Sigmund, A 99 line topology optimization code written in Matlab. *Struct. Multidiscip. Optimiz.* **21** (2001), No. 2, 120–127.
[21] J. S. Smith, O. Isayev, and A. E. Roitberg, ANI-1: an extensible neural network potential with DFT accuracy at force field computational cost. *Chemical Science* **8** (2017), No. 4, 3192–3203.
[22] K. Svanberg and H. Svärd, Density filters for topology optimization based on the Pythagorean means. *Struct. Multidiscip. Optimiz.* **48** (2013), No. 5, 859–875.
[23] J. Tompson, K. Schlachter, P. Sprechmann, and K.Perlin, Accelerating Eulerian fluid simulation with convolutional networks. *arXiv preprint arXiv:1607.03597*, 2016.
[24] K. Um, X. Hu, and N. Thuerey, Liquid splash modeling with neural networks. *arXiv preprint arXiv:1704.04456*, 2017.
[25] Yi M. Xie and G. P. Steven, A simple evolutionary procedure for structural optimization. *Computers&Structures* **49** (1993), No. 5, 885–896.