Generating 3D voxelized architected materials using 3D conditional generative adversarial network

Xiaoyang Zheng [1,2], Ikumu Watanabe [1,2]

[1] Center for Basic Research on Materials, National Institute for Materials Science, 1-2-1 Sengen, Tsukuba 305-0047, Japan

[2] Graduate School of Pure and Applied Sciences, University of Tsukuba, 1-1-1 Tennodai, Tsukuba 305-8573, Japan

Emails: xyzheng1995@gmail.com (X.Z.); WATANABE.Ikumu@nims.go.jp (I.W.)

- https://doi.org/10.48505/nims.4230

Abstract

This tutorial aims to give an introduction of how to use a deep generative model, 3D conditional generative adversarial network (3D-CGAN). The 3D-CGAN can be used for the inverse design of 3D voxelized microstructures with target properties. The 3D-CGAN is trained with supervised learning using a labeled dataset. The dataset consists of a large number of geometries (3D arrays) and their corresponding properties (e.g., elastic moduli). After training, the 3D-CGAN can generate a batch of geometries using target properties at inputs. In our previous tutorial, we have demonstrated how to use CGAN for the inverse design of 2D microstructures.[1] This work is based on our previous publication for the inverse design of 3D architected materials.[2] We hope this tutorial can be useful for those who are interested in the inverse design problems of microstructures.

1. Introduction

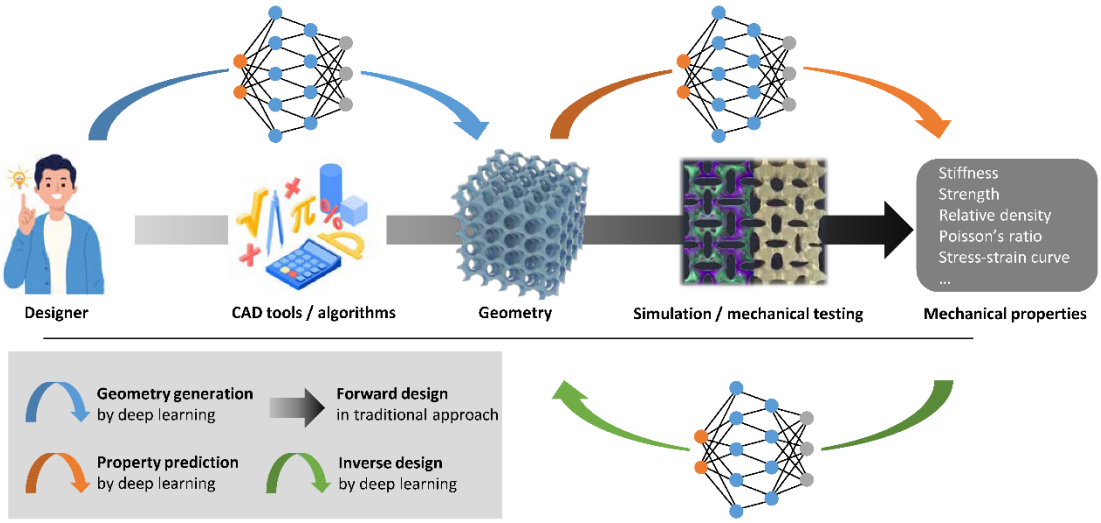

Figure 1. Deep learning in mechanical metamaterials. In traditional forward-design approach, a structure is created by experienced designers with CAD modeling tools or computational algorithms, and its mechanical properties are then theoretically predicted by FEM simulations and experimentally verified by mechanical testing. On the other hand, deep learning can speed up the forward-design approach by replacing the geometry generation and property prediction processes using ANNs. Moreover, deep learning can invert the forward-design approach with an inverse-design approach, in which a structure can be directly generated using a trained ANN taking target properties as input.[3]

Forward design is a conventional approach to design microstructures, such as architected materials, mechanical metamaterials, lattices, etc. This forward design approach follows a general process: a structure is created firstly and then its mechanical properties are investigated by finite element simulation or mechanical testing (Figure 1). The mechanical properties of designed materials will be only known after time-consumingly simulations or experiments. In contrast, deep generative models, such as GAN, enable inverse design of microstructures. In inverse design, microstructures can be automatically generated by inputting target properties to a deep generative model, which outputs corresponding geometries of microstructures. In addition, deep learning can also be used for the property prediction and geometry generation of mechanical metamaterials.[3]

In our previous tutorial, we have introduced the basics of CGAN, and how to use the CGAN for the inverse design of 2D auxetic metamaterial. As CGAN and 3D-CGAN share the similar architecture, we here only give a brief introduction of 3D-CGAN. The detailed fundamentals, usage, and guidance can be referred in the previous tutorial.[1] In the 3D-CGAN, it consists of three neural network structures: a generator, a discriminator, and a calculator (Figure 2). The generator is trained to generate the realistic geometries from latent variables (multivariate normal distribution) and user-defined labels (target properties, e.g., elastic moduli, porosity, etc.), and simultaneously aims to deceive the discriminator and the calculator. The discriminator is trained to

distinguish geometries produced by the generator from the real dataset. The calculator is trained to predict the properties of a given geometry.

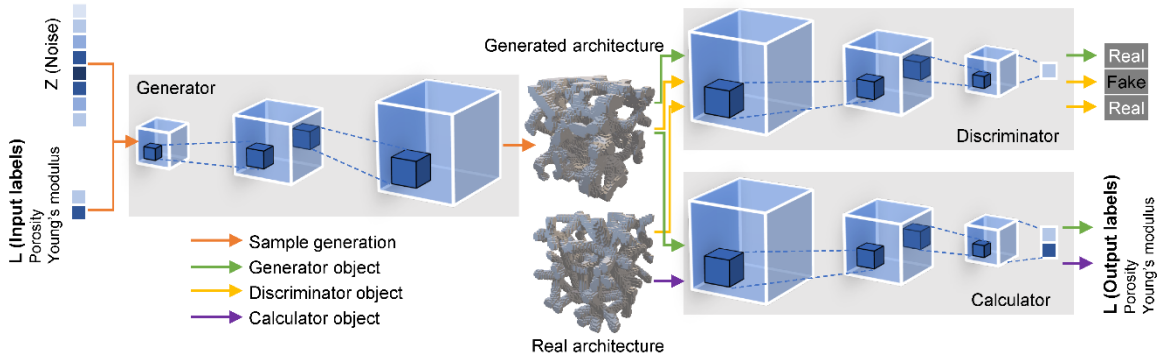

Figure 2. Architecture of 3D-CGAN

The training process is based on supervised learning, where the dataset consists of paired datapoints. In each datapoint, it is composed of a 3D geometry (with a shape of 64×64×64) and its corresponding properties (with a shape of c, where c is the total numbers of properties). In our case, we used 10,000 datapoints for the training process. The geometries were generated using Voronoi tessellation and the properties were calculated using homogenization method.

2. Procedures of CGAN training

2.1 Solver training

As the solver is independent of the generator and discriminator, we firstly train the solver with supervised learning. Follow the steps below:

a. Open *regression_with_ckpt.py* with, e.g., PyCharm
b. Change the source file in line 22 loaddata(filepath. It returns “geo” with shape of b×64×64×64 and “labels” with shape of b×c. b means the total number of geometries, and 64 means the dimension a geometry voxel. The “geo” consists of 0s and 1s, where 0 represents void part and 1 represents solid part. In “labels”, c means the number of properties for a geometry.

Note that the dataset will be divided with 80% for training and 20% for testing. (lines 197-200)

The batch size is set to 32, and training epoch is 200.

The comparison between the real and predicted values is plotted using line 257.

The performance is investigated using the mean square error (MSE) of testing dataset using line 274.

The check points (trained weights and biases) are saved using line 285.

The check points will be used to train the generator and discriminator.

2.2 Generator and discriminator training

After training the solver, we can use the saved weights of the solver to train the generator and discriminator. Follow the steps below:

a. Open CGAN_main.py with, e.g., PyCharm
b. Change the source file in line 172 mat = mat73.loadmat().
c. Change the source file of saved weights in line 166 solver.load_weights(r"/ckpt/solver_199.ckpt")
d. Change your sampling condition in line 187, condition = get_data_space(batch)

Note that the loss functions and MSE will be saved using lines 248-251.

The weighs of the generator and the discriminator will be saved using lines 254 and 255. After training, the saved weights can be used for the inverse design.

Some generated geometries will be saved using line 237, which can be read using 3d_plot_voxel.npy.

3. Conclusions

We give a tutorial of how to use 3D-CGAN for the inverse design of 3D geometries. After suitable training, the CGAN can take target properties as inputs and output corresponding 3D geometries. Also, you can find the 2D case in our previous tutorial and paper.[1,4] If you meet any problems or have any comments, just feel free to contact us via emails. Hope you can enjoy our codes.

Reference

1. Zheng X, Watanabe I. Tutorial for conditional generative adversarial network. https://doi.org/10.48505/nims.3869.
2. Zheng X, Chen TT, Jiang X, Naito M, Watanabe I. Deep-learning-based inverse design of three-dimensional architected cellular materials with the target porosity and stiffness using voxelized Voronoi lattices. Science and Technology of Advanced Materials. 2023 Dec 31;24(1):2157682.
3. Zheng X, Zhang X, Chen TT, Watanabe I. Deep Learning in Mechanical Metamaterials: From Prediction and Generation to Inverse Design. Advanced Materials. 2023 Jun 18:2302530.
4. Zheng X, Chen TT, Guo X, Samitsu S, Watanabe I. Controllable inverse design of auxetic metamaterials using deep learning. Materials & Design. 2021 Dec 1;211:110178.