Science Advances
AAAS

advances.sciencemag.org/cgi/content/full/6/17/eaaz4169/DC1

Supplementary Materials for

Designing complex architectured materials with generative adversarial networks

Yunwei Mao, Qi He, Xuanhe Zhao*

*Corresponding author. Email: zhaox@mit.edu

Published 24 April 2020, *Sci. Adv.* **6**, eaaz4169 (2020)
DOI: 10.1126/sciadv.aaz4169

This PDF file includes:

Supplementary Text

Topology generation algorithm

Our topology generation algorithm is schematically shown in **Fig. S1**. The topology generation process is correlated with the desired symmetry of the unit. For convenience, we choose a relatively simple symmetry p4 as an example. We mark the total number of pixel in the unit as N_{pixel}, and the desired porosity of the final architectured material as ϕ. The start point of our generation procedure is a unit filling with solid pixels, as displayed in **Fig. S1 (A)**. The task of generating an architectured material with desired porosity then be simplified as transform $N_{void} = N_{pixel}\phi$ solid pixels into void pixels.

We then randomly seed N_{seed} void seeds into an element of the unit. The spatial distribution of void seeds in the unit can be mapped from the spatial distribution of void seeds in an element by the symmetry manipulations, i.e. **Fig. S1 (B)**. These N_{seed} void seeds in an element of the unit will randomly grow into void phases by a boundary-etching algorithm (see the next paragraph and **Fig. S2** for more details). An intermediate configuration of the unit is shown in **Fig. S1 (C)**. Notice that i). the sizes of void phases in the element can be different from each other; and ii). the shapes of void phases in the element can be different from each other. These randomness features are rooted in our boundary-etching algorithm. The growth of the void phases will be terminated when the total number of pixels in the void phases N_{void} reaches the $N_{void} = N_{pixel}\phi$ constraint. The configuration reached the constraint is shown in **Fig. S1 (D)**.

Even though the porosity constraint is satisfied, this configuration is usually not path-connected. There are many isolated solid phases which are not connected with the percolated solid phase of the configuration. In order to have a path-connected configuration, we need merge these isolated solid phases to the percolated solid phase. The configuration satisfying the porosity

constraint will go through a domain-check algorithm. The domain-check algorithm detects solid phases, and marks the solid pixels belonging to a solid domain with a unique domain ID. **Fig. S1 (E)** is colored in the domain IDs. For example, the void phases are with a domain ID zero, and the percolated solid phase is with a domain ID one. Other solid phases will be with a domain ID larger than one. After such identification, these small isolated solid phases are merged to the surface of the percolated solid phase, to form a path-connected configuration. **Fig. S1 (F)** displays the final unit of the architecture material.

The boundary-etching algorithm for a void phase growth is shown in **Fig. S2. Fig. S2 (A)** displays the void phase (pixels in white) surrounding by solid phase (pixels in black) at a certain. To grow the void phase, we first identify the void-solid boundary. This void-solid boundary is highlighted in **Fig. S2 (B)** in green. The new voids will appear by etching the solid pixels next to the void-solid boundary. Such new voids are shown in **Fig. S2 (C)** in yellow. Both the number and the position of solid pixels that turn into voids are random. After forming these new voids, the void phase grows (the gray figure in **Fig. S2 (C)**). This boundary-etching procedure will be repeated until the void phase reaches a certain size.

Details of machine learning setups

The structure of datasets

Our database is composed by 17 separated datasets. Each dataset corresponds to a particular symmetry. In **Fig. S3** the color bars represent our database, while each color bar is for a dataset corresponding to a particular symmetry. In each data-point in each dataset, such as the showed symmetries p4g and p6m in **Fig. S3**, we store three items: normalized Young's modulus E_{mean}/E_{HS}, the isotropy Ω, and the configuration. In order to take the symmetry information

into account while we construct the dataset, only the pixel information in the element will be stored. More specifically, the pixel information in the element will be stored as a vector for each architectured material.

As we see in **Fig. S3**, the dimension of the vector, which representing the architectured material, depends on several factors: the resolution of the computational domain (the gray rectangles in **Fig. S3**) and the symmetry. Typically, the higher the resolution of the computational domain, the vector will possess a higher dimension; the higher the symmetry, the vector will have a decreased dimension. **Table S1** summarizes the details of the resolutions of the computational domain and dimensions of the configuration vector for each symmetry that we used in our database.

GANs setups

In this work, the architecture of deep convolutional generative adversarial network is adopted. Overall, the generator is identified through supervised learning, while the discriminator learns without supervision. For the neural networks structures, the generator is composed by five layers. The first layer is a fully connected layer composed by 1024 neuros to receive the data. The second layer is also a fully connected layer composed by 1600 neuros. The third and fourth layers are a convolutional and are composed by 64 and 32 neuros, respectively. The last deconvolutional layer is associated with a tanh activation function to produce configurations with bounded pixel values. For the input of the generator, the $\tilde{E}_{mean}/E_{HS}$ is used as the label attached to each of the configuration for the supervised learning of the discriminator and Ω is used as the threshold to screen out anisotropic configurations in advance. For the output of the generator, the size of the last layer of the neural network depends on the symmetry groups, e.g. *2500 for p1*,

and 132 for p6m. We have separately trained different GANs for different symmetry groups. The overall structure of discriminator is similar with that of generator. In the discriminator, the last convolutional layer has a sigmoid activation function appended to produce probabilities between 0 and 1, while the other convolutional layers are all associated with batch normalization operations and leaky rectified linear unit activations. The goal of the discriminator is to detect fake generated data, i.e. low-modulus configurations, so the discriminative neural network is trained to minimize the final classification error. A discriminator in this problem effectively functions as a fast finite-element solver to map out the relationship between pixel matrices of elements and their corresponding probability to be the high elastic-modulus configurations. The discriminator could speed up the whole training process without introducing another costly finite-element solver (to calculate the newly generated configurations).

While the theoretical optimality of GANs is Nash equilibrium, the global optimality or sufficiently good local optimality is not guaranteed. In order to improve it, we include several terms into the loss function. Specifically, the total loss consists of three major components: i). adversarial loss that evaluates the performance of generator and discriminator and ii) style transfer loss that imposes morphological constraints to the generated microstructures. We will explain one by one in the following section.

1. The adversarial loss is essentially the optimization objective of GANs, expressed as

$$L_{GAN} = \max_{G}\left(\min_{D} E(G,D)\right) = E_{X\sim p_{data}}[\log D(x)] + E_{X\sim p_z}\left[\log\left(1 - D(G(z))\right)\right]$$

Note that the min-max training process essentially wants the generator G to minimize the loss the let D maximizes it.

2. The style transfer loss is

$$L_{style} = (\phi_G - \phi_A)^2$$

with ϕ_G and ϕ_A are the porosity of generated configurations and real configurations, respectively.

Thus, the total loss is

$$L(G, D) = L_{GAN} + \alpha L_{style}$$

with α is the moderating weights that prevent the style transfer loss from diminishing to zero or overwhelming the GANs adversarial loss. α is set to be 0.03 in our study. Furthermore, Adam optimizer is applied in training by setting the learning rate as 0.0001. The batch size for training is set to be 32.

Fig. S4 gives the details of the machine learning training process. **Fig. S4(A)** shows the accuracy versus epoch for models with different training data densities. The accuracy for the model with 0.1 million data points actually does not improve over 100 epochs. Models with larger databases (0.4 and 0.8 million) can substantially increase the accuracy over a few epochs (e.g., 20 epochs). To be conservative, we choose 1 million data points in the training process.

Overfitting could happen in the training process. We adopt the early stop method to suppress the overfitting, and the results is shown in **Fig. S4(B).** In the models with early stop, the accuracies for both training set and the test set are similar to each other.

Additional architectured materials that achieve the HS bound

In the main text, we show several representative configurations for architectured materials that achieve the HS bound within a wide porosity range from 0.1 to 0.8. After the GANs-model is trained, it can generate enormous number of architectured materials that achieve the HS bound. Here we show more architectured materials for such an extreme design. All architectured

materials satisfy i). $\Omega \leq 0.05$ and ii). $\tilde{E}_{mean}/E_{HS} \geq 0.9$. **Fig. S5** is for $\phi = 0.05$. In this case, the void phases are very small and most of them are pixel-level (the number of void pixels is limited). Thus, the shape/size of void itself is not very intriguing compared with the spatial pattern of void phases. With this consideration, we only show a limited number of architectured materials. As the porosity increases, both the shape/size of void and the spatial pattern of void phases become important. Thus, a larger number of architectured materials are displayed for illustrating this point. **Fig. S6** is for $\phi = 0.15$, **Fig. S7-S8** is for $\phi = 0.25$, **Fig. S9-S10** is for $\phi = 0.35$, **Fig. S11-S12** is for $\phi = 0.45$, and **Fig. S13** is for $\phi = 0.55$. As the porosity further increases, the shape/size of the void phase is really stretched out (the number of solid pixels is limited) and majorly only the spatial pattern of void phases matters. This fact is rooted in the fact that our resolution of the computational domain is fixed. If we enhance the resolution of the computational domain, more possible architectured materials can be discovered. Nevertheless, with the current setup for resolution (**Table S1**), **Fig. S14** and **Fig. S15** is for $\phi = 0.65$, and $\phi = 0.75$, respectively.

Experimental and finite-element simulation setups

Experimental setup

In the above-mentioned Section 3, we provide a large dataset consists of over a thousand 2D architectured materials created by generative adversarial networks (GANs). The experiments were carried out to verify that these isotropic configurations achieved the optimal Young's modulus.

To prepare the suitable testing samples, we need their CAD models first. Each 3D sample model was built from a 2D 1-0 configurations generated by the GANs system. The 1-0 matrix

was mapping into the black-and-white geometry to represent solid phase and void phase in architectured materials. The binary number "1" represented the solid phase and "0" represented the void phase, which was plotted in the final configuration as black and white pixel, respectively. After the black-and-white configuration was generated, we imported the configuration into Solidworks software. In Solidworks sketch module, we need to use simple geometry, i.e. closed shape with lines and curves, to fully represent the original black-and-white pixels, since the cutting route for laser cutter need to be smooth. The cubic spline interpolation was adopted here to fit the solid-void boundary. Thereafter, the simplified geometry was linear patterned in x and y direction to make a full architectured material, then pad the 2D architectured material in z direction by 1.5mm. The size of architectured material is $50mm \times 50mm$. Appended with shoulders on the two sides, the whole 3D testing sample was established. The full batch of 3D model used in experiments are displayed in **Fig. S17.**

Fabrication on the testing samples

Acrylic plates (thickness 2.25mm) were used to fabricate the testing samples. The CAD testing dataset were uploaded to the laser cutter (Epilog Mini/Helix; Epilog Laser) and cut multiple times with minimal laser energy, so as to avoid weight loss by melting acrylic. The resolution of Epilog Laser is limited to 200um, and the micro holes in the architectured materials cannot be formed concisely, so we choose 3×3 to be equivalent to periodic boundary condition. **Fig.S18** also shows the 3×3 experimental setup is sufficient to derive effective mean Young's modulus. It converges fast and have little difference with 6×6 case, which is assumed to be closer to the periodic boundary condition. This claim is valid in different porosities. In experimental setup, during the stretching process, the shoulders are constrained to fixed

boundary condition, i.e. the constrained uniaxial tension tests. The sample undergoes neither plain-strain nor plain-stress deformation. The comparation between the plain-stress case, plain-strain case and constrained uniaxial tension tests are displayed in **Fig. S18.** The experimental data lie between the plain-stress and plain-strain case, and it could be converted to equivalent plain-strain result.

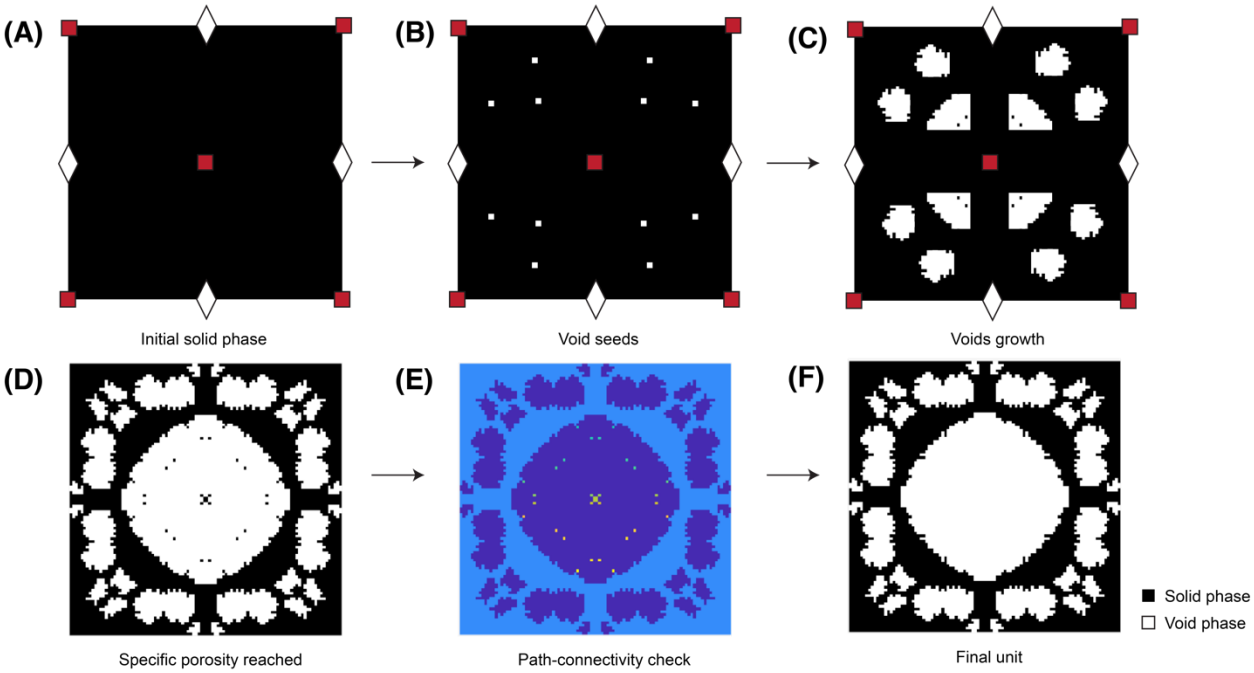

Fig. S1. The region-growing based algorithm for topology generation. (A). A unit is filled with all solid pixels at the initial stage. (B). Void seeds are randomly assigned in an element of the unit. (C). The void seeds randomly grow into void phases by a boundary-etching algorithm. (D). The growth is terminated when the total number of void pixels N_{void} reaches $N_{pixel}\phi$. (E). The domain-check algorithm detects isolated solid pixels. (F). The isolated solid pixels are randomly merged to the surface of the percolated solid phase to form a path-connected configuration.

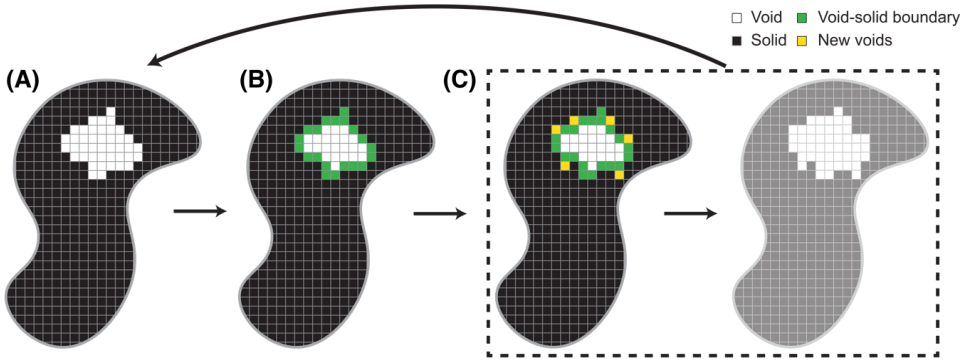

Fig. S2. The boundary-etching algorithm for the growth of a void phase. (A) The initial void phase (pixels in white) is surrounded by solid phase (pixels in black). (B). Highlight the void-solid boundary (pixels in green). (C). New voids randomly grow by etching the solid pixels next to the boundary and the whole process repeats.

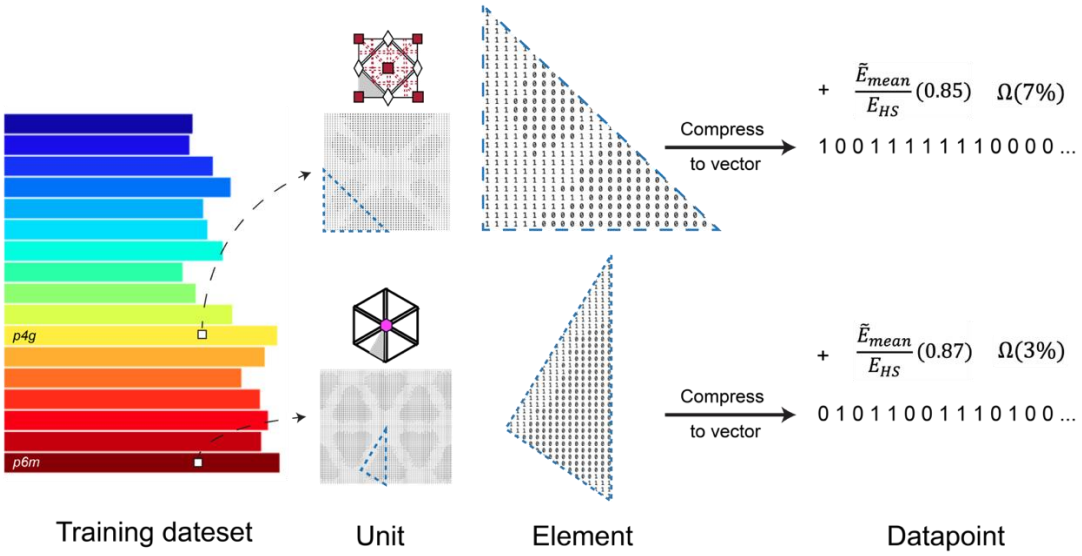

Fig. S3. The structure of datasets.

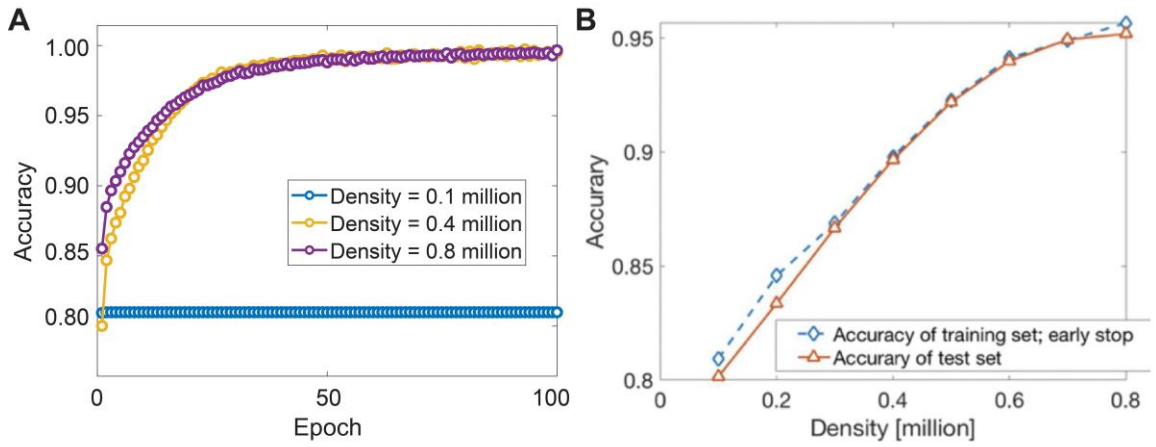

Fig. S4. Details of the machine-learning training process. (A). Training set accuracy versus epoch for models with different training data densities. (B). Early stop to avoid overfit for models with different training data densities. In the models with early stop, the accuracies for both training set and the test set are similar to each other.

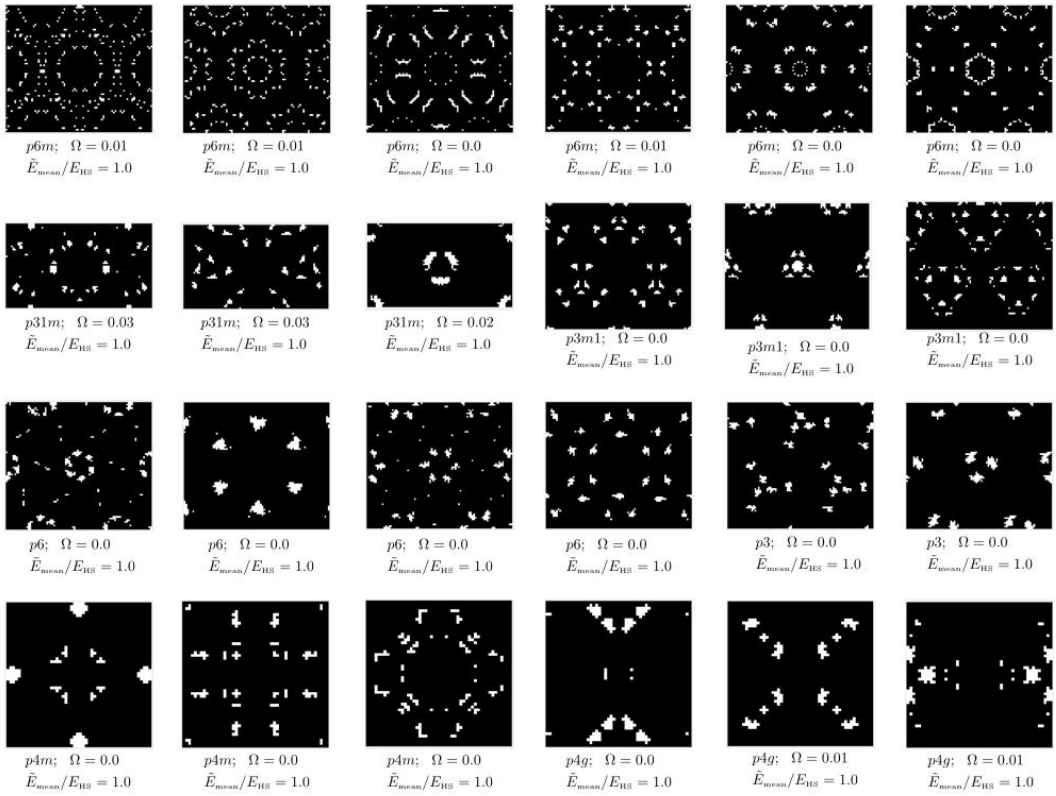

Fig. S5. The architectured materials achieved HS bound for $\boldsymbol{\phi = 0.05}$.

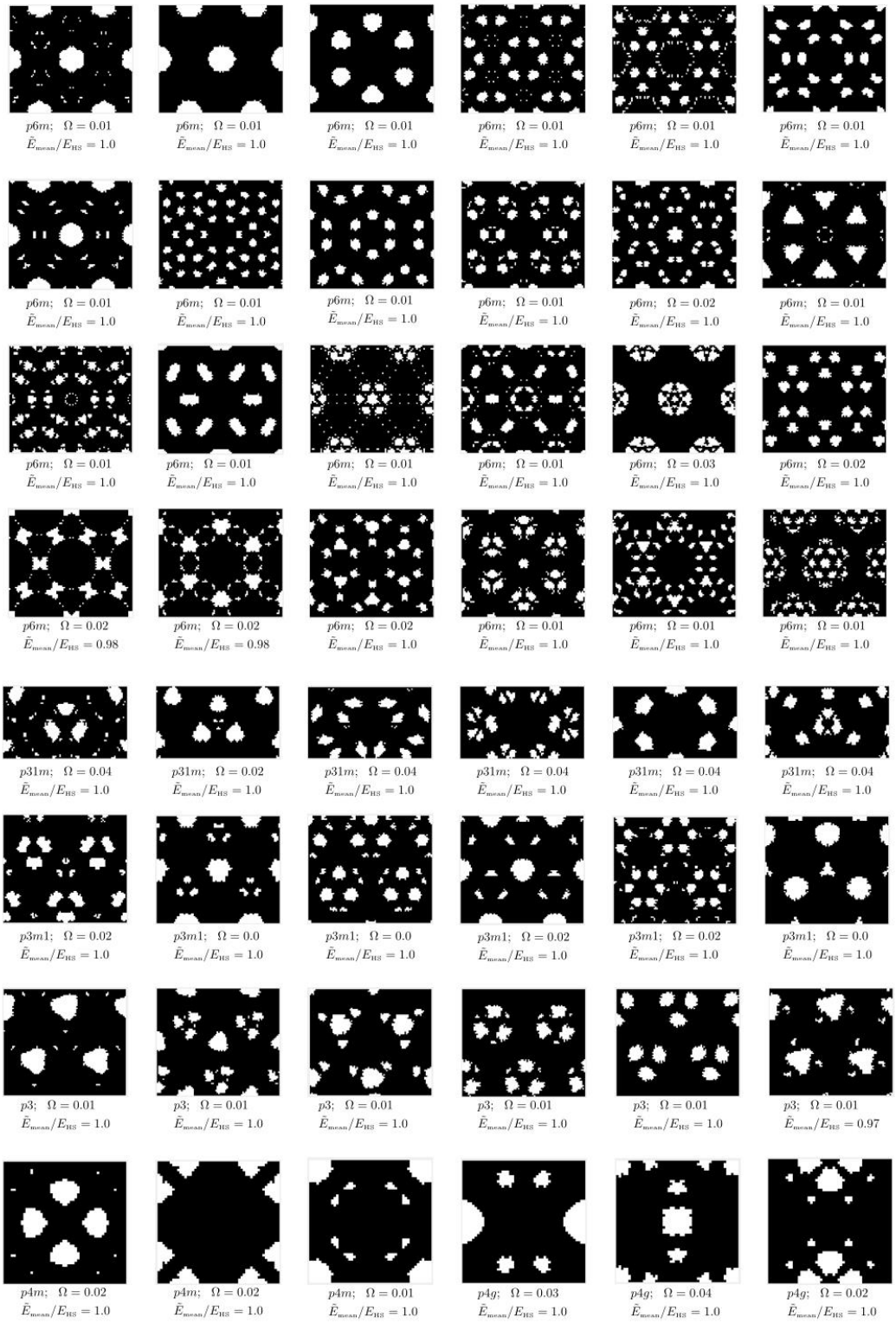

Fig. S6. The architectured materials achieved HS bound for $\boldsymbol{\phi = 0.15}$.

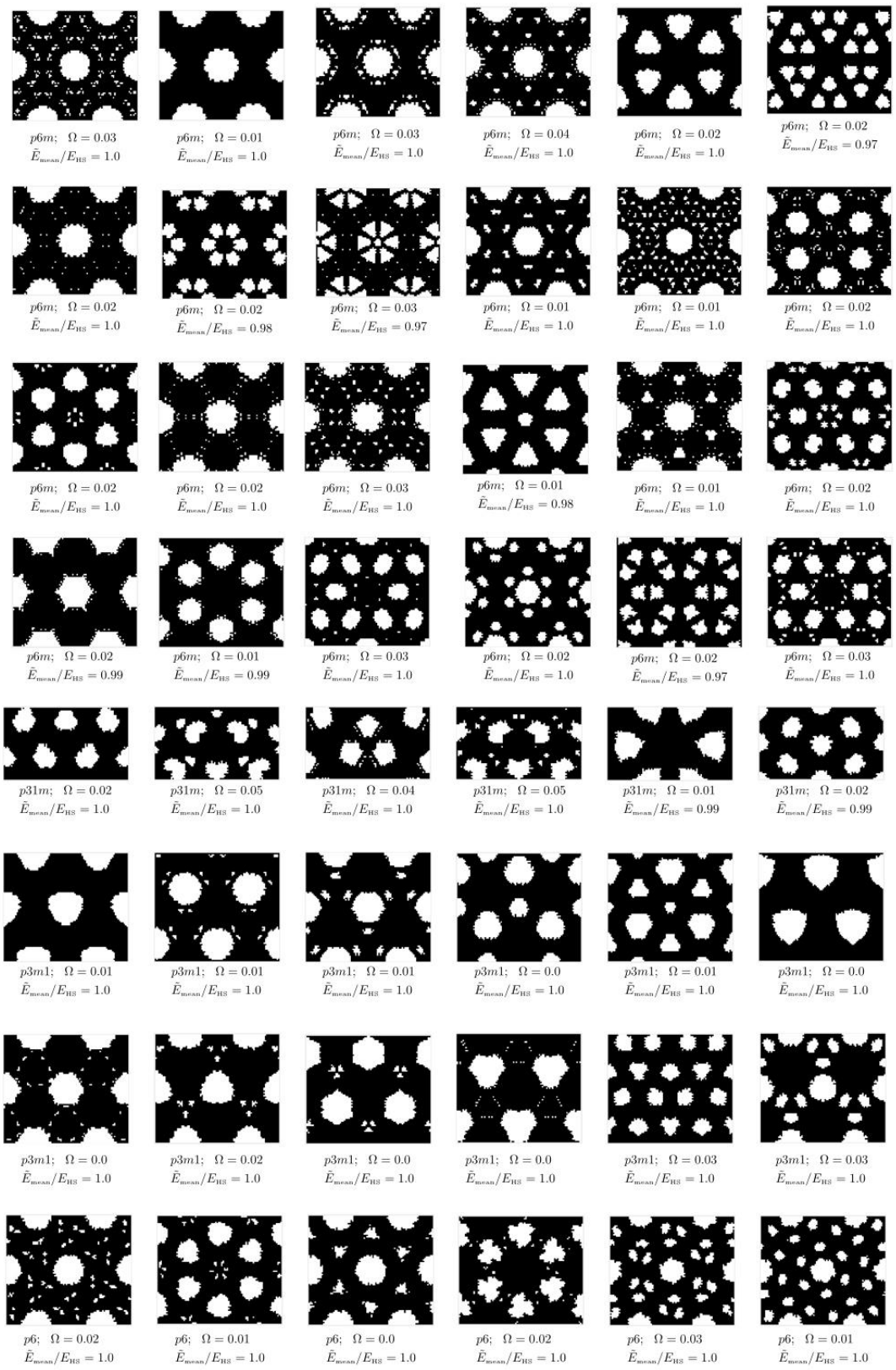

Fig. S7. The architectured materials achieved HS bound for $\boldsymbol{\phi = 0.25}$.

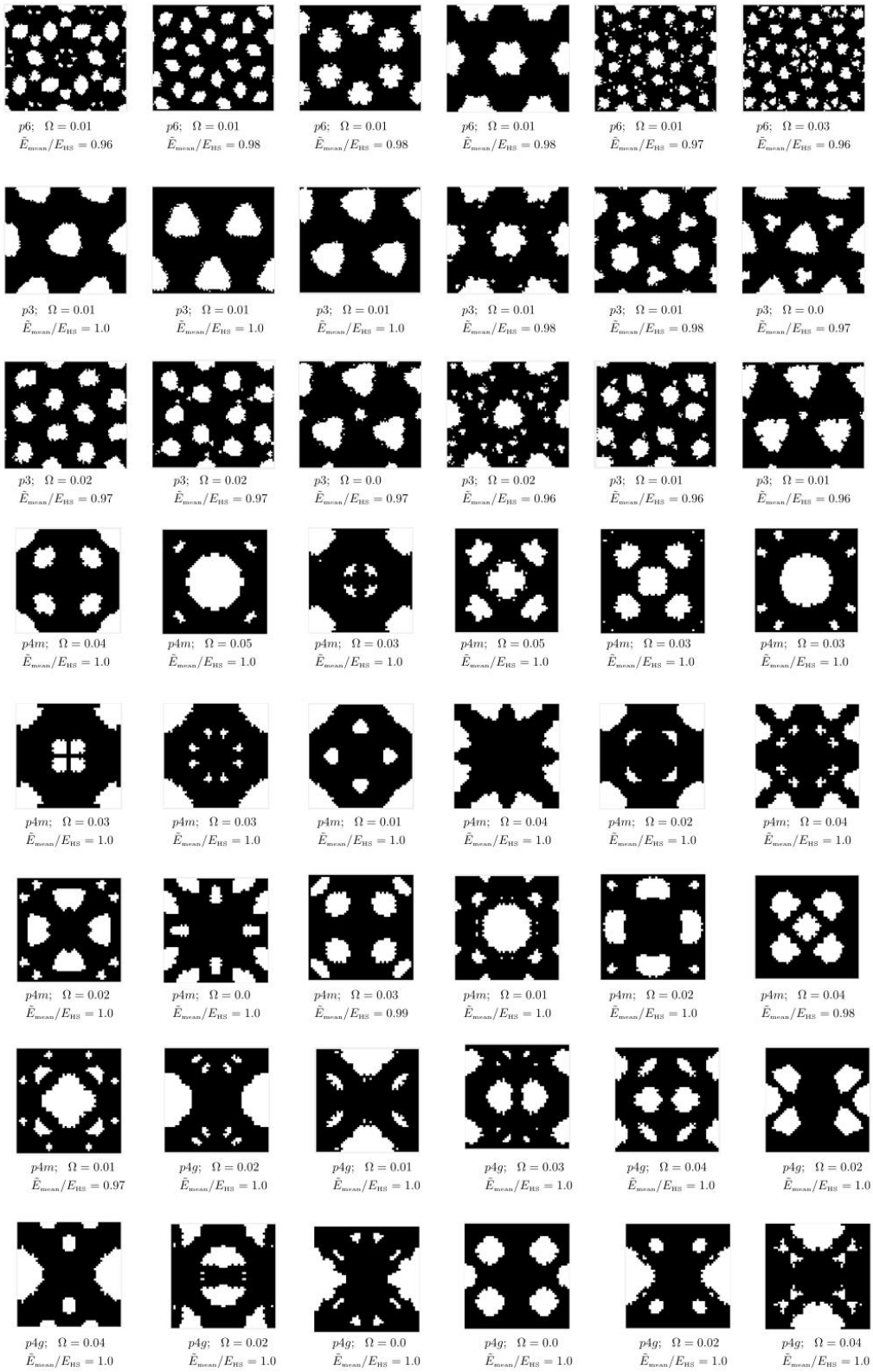

Fig. S8. The architectured materials achieved HS bound for $\boldsymbol{\phi = 0.25}$ (***continued***).

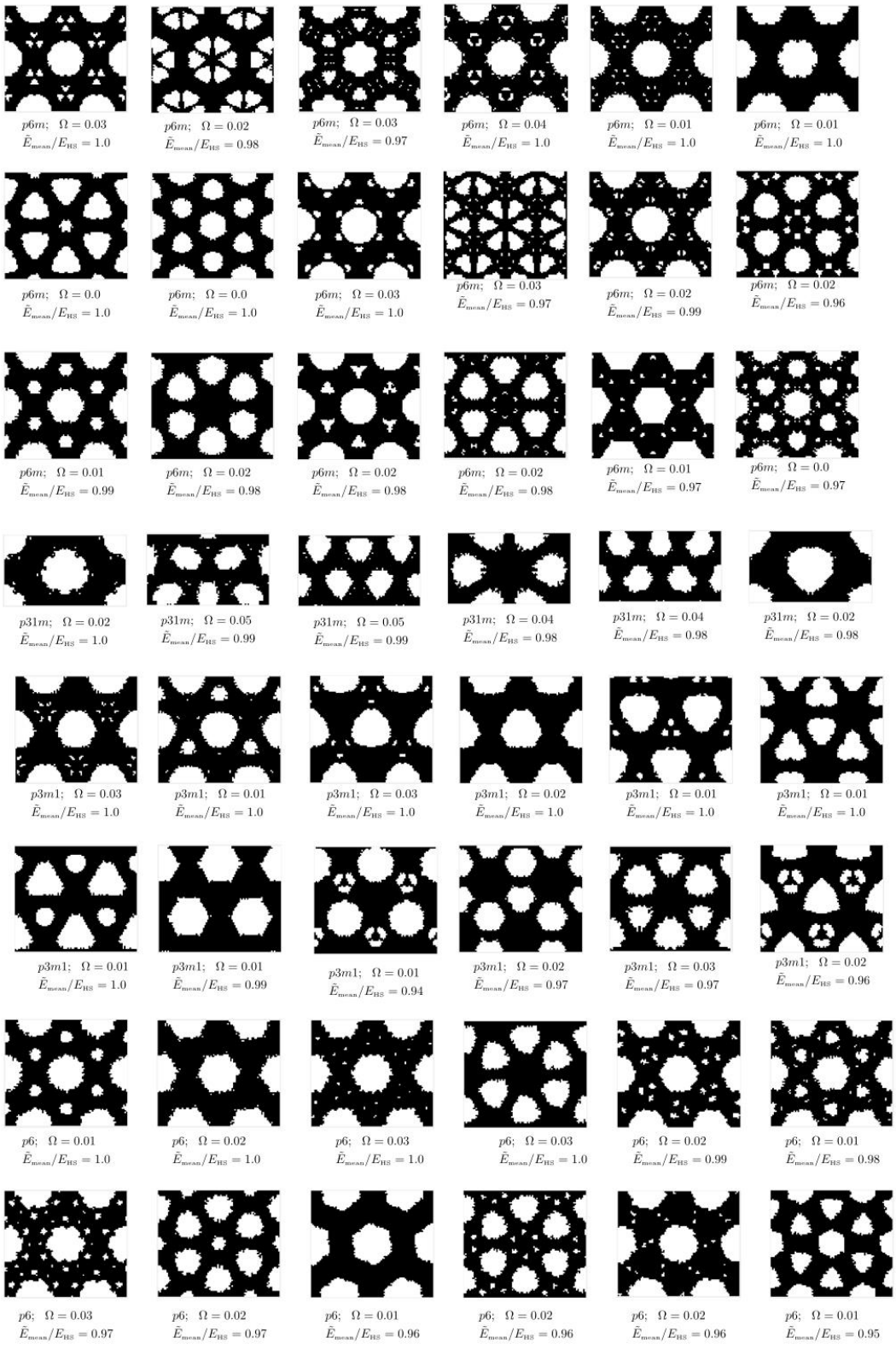

Fig. S9. The architectured materials achieved HS bound for $\boldsymbol{\phi = 0.35}$.

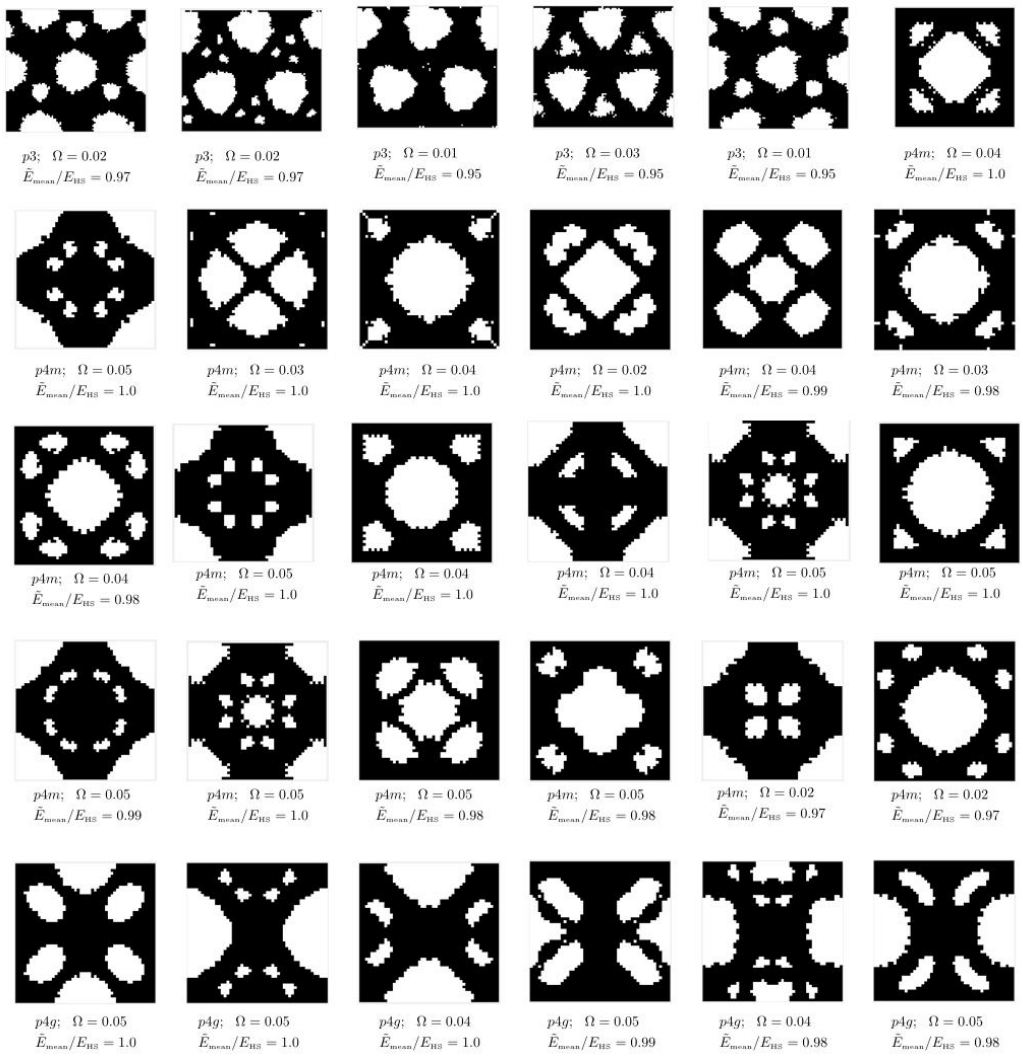

Fig. S10. The architectured materials achieved HS bound for $\boldsymbol{\phi = 0.35}$ (***continued***).

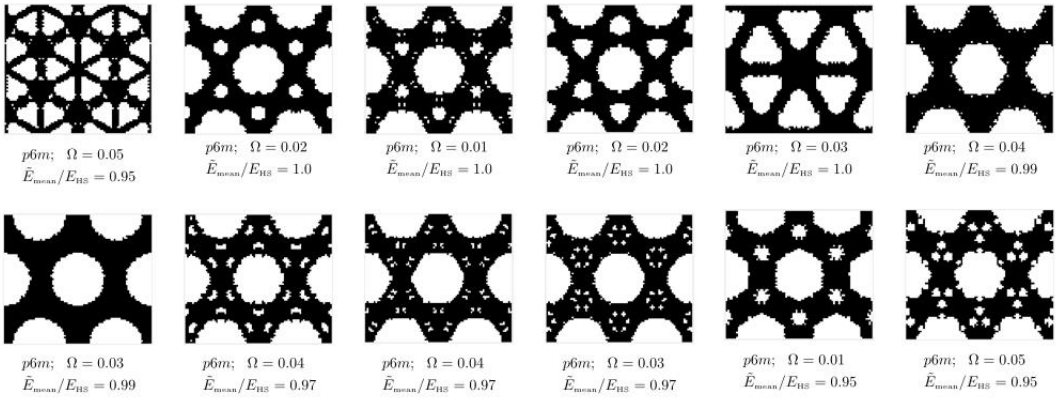

Fig. S11. The architectured materials achieved HS bound for $\boldsymbol{\phi = 0.45}$.

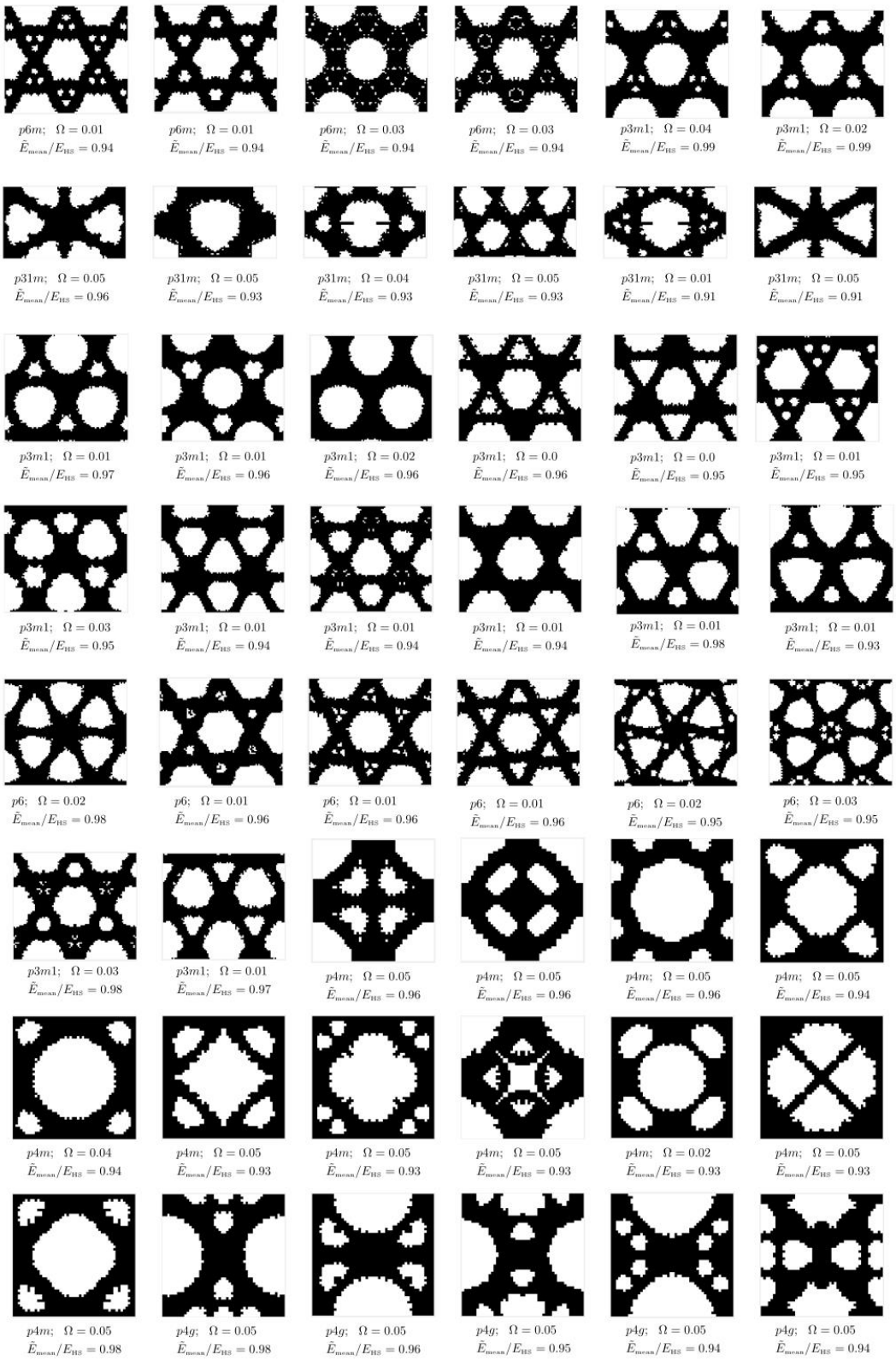

Fig. S12. The architectured materials achieved HS bound for $\boldsymbol{\phi = 0.45}$ (***continued***).

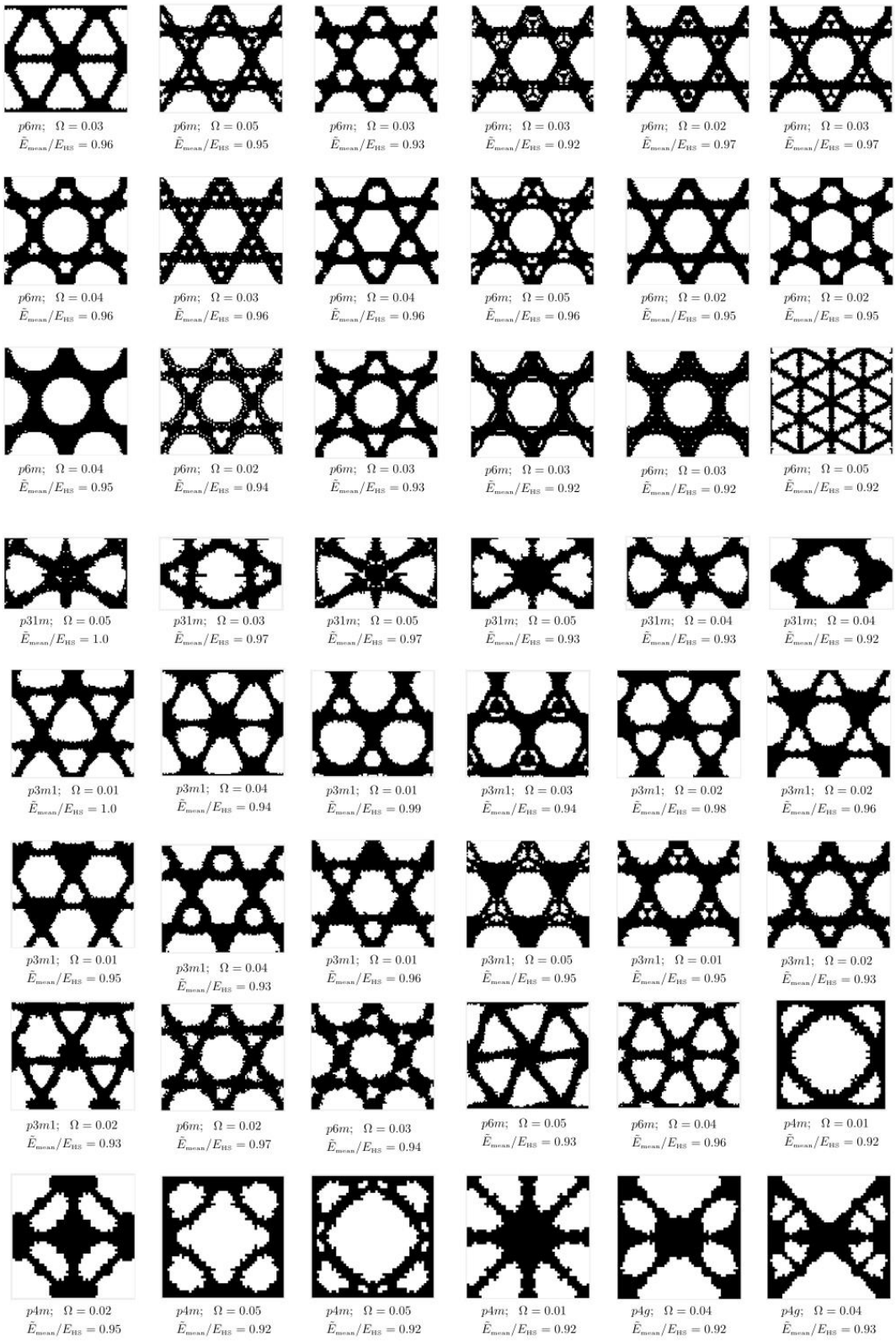

Fig. S13. The architectured materials achieved HS bound for $\boldsymbol{\phi = 0.55}$

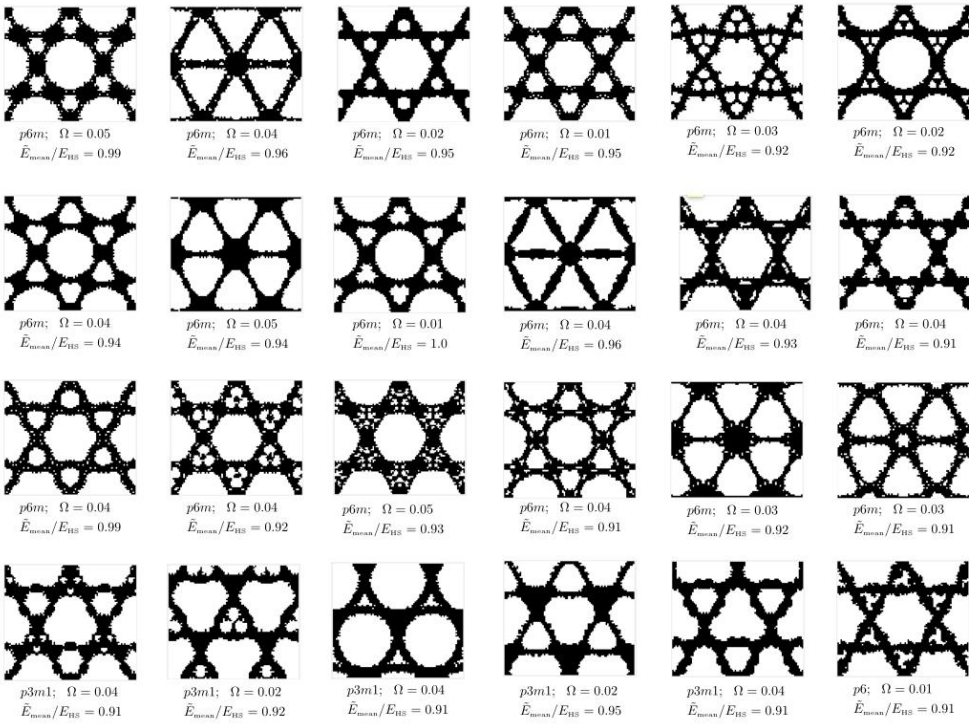

Fig. S14. The architectured materials achieved HS bound for $\boldsymbol{\phi = 0.65}$

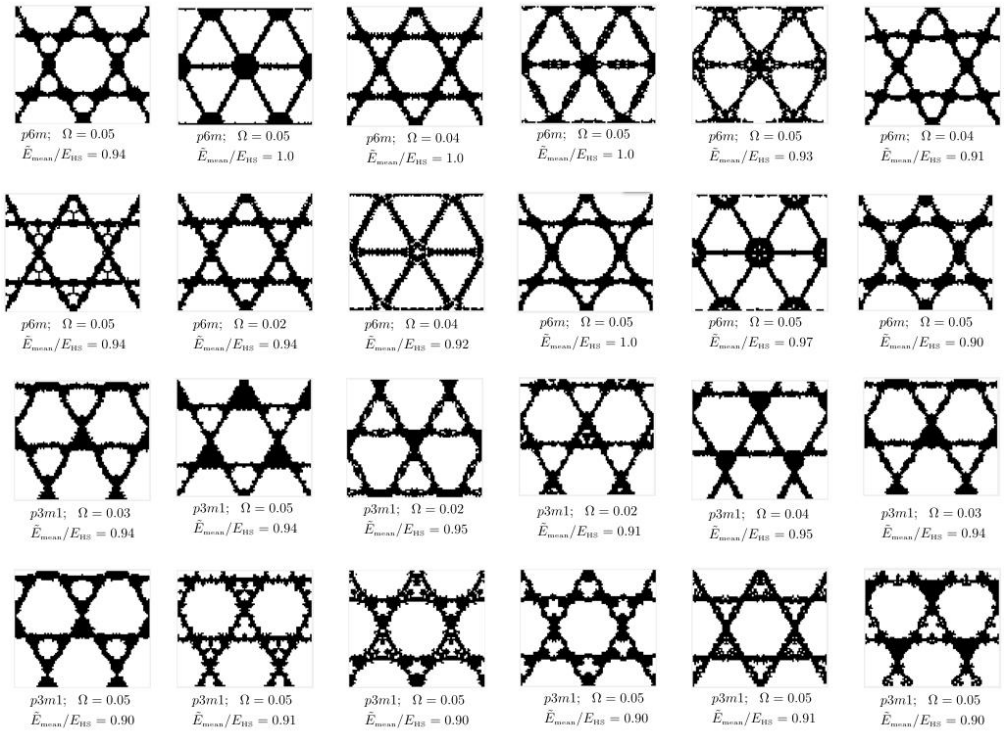

Fig. S15. The architectured materials achieved HS bound for $\boldsymbol{\phi = 0.75}$

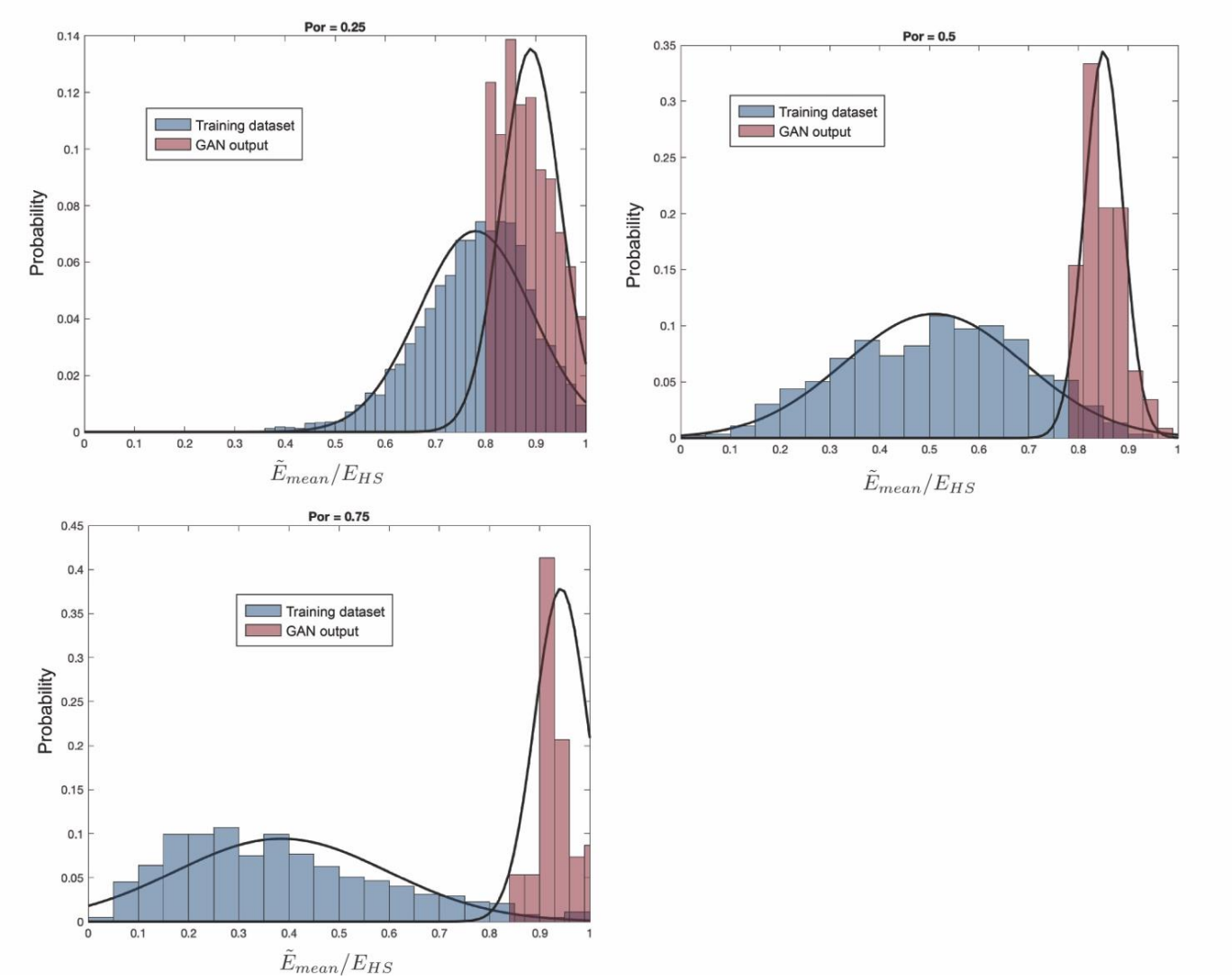

Fig. S16. The statistic distribution plot compared the training dataset with the GANs output within *p6m* symmetry group ($\boldsymbol{\Omega < 5}$%). Porosity ($\boldsymbol{\phi}$) changes from 0.25, 0.5 to 0.75.

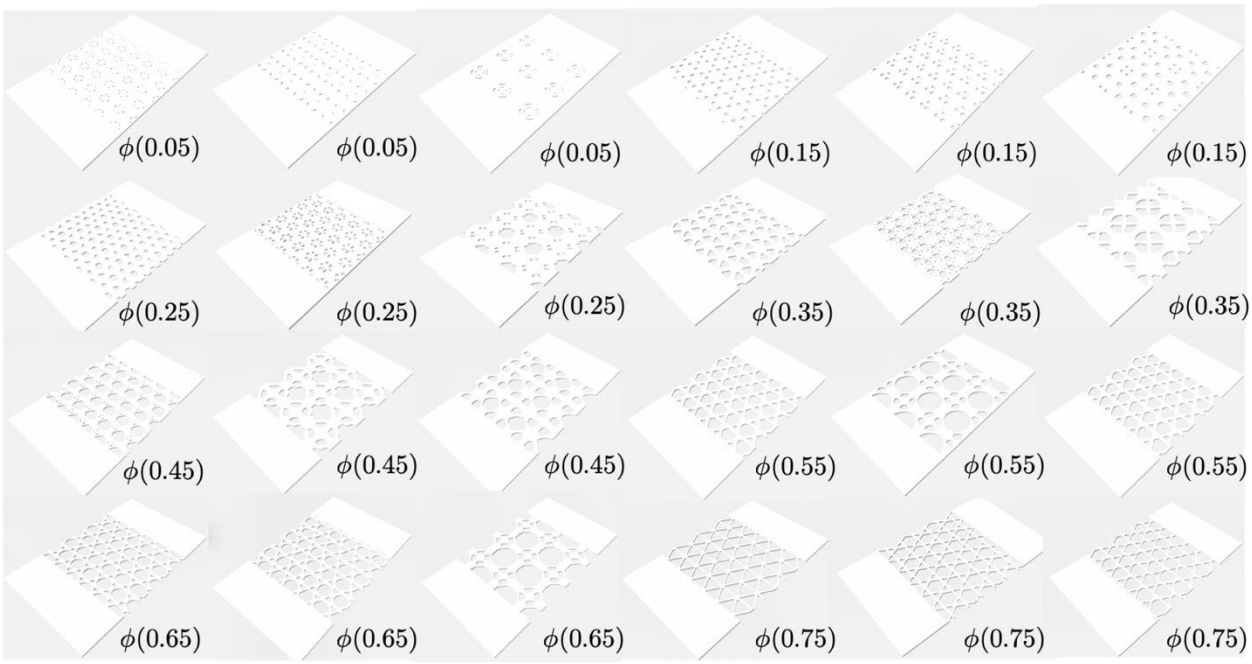

Fig. S17. Architectured materials 3D model in uniaxial tensile test with different porosities ϕ

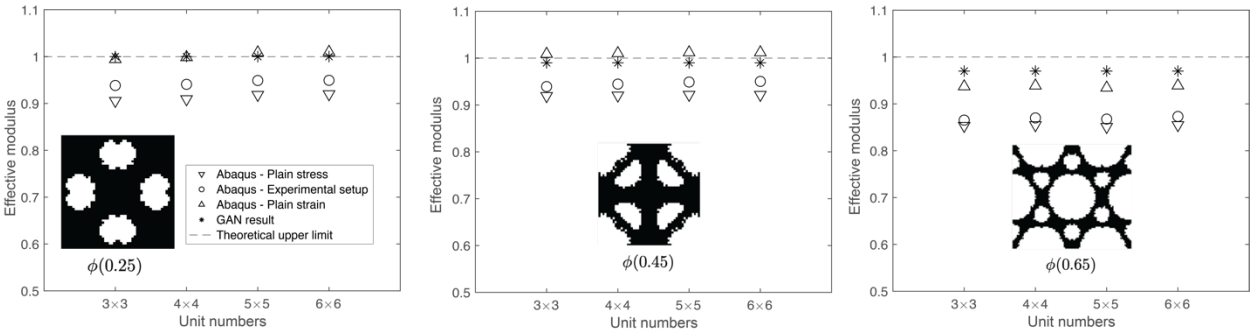

Fig. S18. Effective mean Young's modulus with different repetition of units' number in various porosity

Table S1. Resolutions of the computational domain, unit, and element.

Symmetry	*p1*	*p2*	*pg*	*pm*	*pgg*	*pmg*	*pmm*	*cm*
Dimension	50x50	50x50	50x50	50x50	50x50	50x50	50x50	50x50
Pixels in compute-domain	2500	2500	2500	2500	2500	2500	2500	2500
Pixels in an element	2500	1250	1250	1250	625	650	625	600

Symmetry (continued)	*cmm*	*p4*	*p4g*	*p4m*	*p3*	*p6*	*p3m1*	*p31m*	*p6m*
Dimension	50x50	50x50	50x50	50x50	86x75	86x75	86x75	86x50	86x75
Pixels in compute-domain	2500	2500	2500	2500	6450	6450	6450	4300	6450
Pixels in an element	300	625	325	325	528	264	264	363	132