

Supporting Information: Template Design for Complex Block Copolymer Patterns Using Machine Learning Method

Zhihan Liu, Yi-Xin Liu,* Yuliang Yang, and Jianfeng Li[†]

The State Key Laboratory of Molecular Engineering of Polymers,
Department of Macromolecular Science, Fudan University, Shanghai 200433, China

I. FLOW CHART OF THE MULTI-CHANNEL INPUT STRATEGY

The flow chart of the multi-channel input strategy (MCIS) is shown in Figure S1, while that for the single-channel is referred to Figure 2 in the main text.

II. NEURAL NETWORK ARCHITECTURES

As mentioned in the main text, a typical neural network model used in this work can be denoted as X-CNN_n-Res_m-SCN_Y with X = M (multi-channel) or S (single-channel), n number of CNN layers employed, m number of Resnet blocks employed and Y = ON (with SCN module) or OFF (without SCN module). SCN stands for self-conscious neural network module as explained in the following context. A list of all the NN models investigated is provided in Table S1 and Table S2.

SCN: The Self-Conscious Network (SCN) proposed in this work aims to mimick the self-monitoring function of the brain. When our brain is completing some task, our consciousness is actually somehow monitoring the inner state of our brain and this process might improve the performance of our brain. The SCN module, presented in Figure S2, plays a similar role. In this given example, the primary NN is a baseline CNN consisting of only two CNN layers. It should be noted that the baseline CNN,depicted in the brown box, is already a neural network that can be used to predict the template, and will be trained first. To monitor the inner state of the baseline CNN and modify its behavior, a secondary neural network SCN, represented by several dense layers is be added. The SCN receives data flowing into it, and sends information back to the primary model. It is worth mentioning that the SCN model can be implemented in other neural network models besides the CNN.

Residual structure: When training a very deep neural network, the gradients used to update the parameters in the early (shallow) layers can become very small. This phenomenon is known as the gradient vanishing problem [1]. To address this issue, the ResNet architecture was introduced in 2015, which involves adding the output of a convolutional layer to the input of a non-adjacent one within each convolutional block (the output and input shapes must be the same). This residual connection enables the gradients to flow more easily and allows for more effective updates to the shallow layer parameters. For more information on the ResNet block, please refer to [2].

III. DATA AUGMENTATION METHODS

Rotation & Reflection: We augment the original training set by performing rotation operations (at 90-degree intervals) and mirror-reflection operations, resulting in a new set that is eight times larger than the original.

Pattern broadening: Pattern broadening involves increasing the size of the pattern array x from 40×40 to 48×48 to provide more pattern information to the neural network, while keeping the corresponding template array y in its original size of 5×5 . To achieve this, we take advantage of the periodic boundary condition in the forward SCFT simulation, which allows us to tile the 40×40 pattern array to create a 120×120 array and then extract a 48×48 section from the center (as shown in Figure S3).

Window movement: To expand the sample size, we use a window movement technique, which increases the dataset size by a factor of six. As illustrated in Figure S4, we first tile the pattern array x and the template array y separately. Next, we place a window on the tiled arrays to generate new samples. Since we have already expanded the dataset using R&R, we can lay the window in six different ways to avoid repetition.

* lyx@fudan.edu.cn

† lijf@fudan.edu.cn

TABLE S1. Architectures of the neural network models CNN_2 (or $\text{X-CNN}_2\text{-RES}_0\text{-SCN}_Y$), $\text{X-CNN}_8\text{-RES}_0\text{-SCN}_Y$, $\text{X-CNN}_{24}\text{-RES}_0\text{-SCN}_Y$ and $\text{X-CNN}_{32}\text{-RES}_0\text{-SCN}_Y$. In these models, $(5 \times 5, 64)$ indicates a CNN layer with kernel size 5×5 and 64 features. The baseline model CNN_2 has two CNN layers and two dense layers, while the other three models have three dense layers. Please refer to the caption of Table 2 in the main text for the naming conventions of these neural network models.

NN Model	a $\text{X-CNN}_2\text{-RES}_0\text{-SCN}_Y$	b $\text{X-CNN}_8\text{-RES}_0\text{-SCN}_Y$	c $\text{X-CNN}_{24}\text{-RES}_0\text{-SCN}_Y$	d $\text{X-CNN}_{32}\text{-RES}_0\text{-SCN}_Y$
Block1	$5 \times 5, 64$	$5 \times 5, 64$ $5 \times 5, 64$	$5 \times 5, 64$ $5 \times 5, 64$ × 3	$5 \times 5, 64$ $5 \times 5, 64$ × 4
		2×2 maxpool, stride 2		
Block2	$5 \times 5, 64$	$5 \times 5, 64$ $5 \times 5, 64$	$5 \times 5, 64$ $5 \times 5, 64$ × 3	$5 \times 5, 64$ $5 \times 5, 64$ × 4
		2×2 maxpool, stride 2		
Block3		$5 \times 5, 64$ $5 \times 5, 64$	$5 \times 5, 64$ $5 \times 5, 64$ × 3	$5 \times 5, 64$ $5 \times 5, 64$ × 4
		2×2 maxpool, stride 2		
Block4		$3 \times 3, 128$ $3 \times 3, 128$	$3 \times 3, 128$ $3 \times 3, 128$ × 3	$3 \times 3, 128$ $3 \times 3, 128$ × 4
		flatten		
dense layers		dense, 512, dropout		
		dense, 1024, dropout		
	dense, 25, sigmoid			

Interface noise adding augmentation (INAA): Before training, we apply INAA to the pattern arrays in the training dataset to help the model focus more on the overall shapes of domain boundaries. The noise addition process involves two steps. In the first step, we identify pixels with a value of 1 located on the boundary between the two domains and randomly change the value of one-third of these pixels to 0. In the second step, we do the same process but with the 0 pixels changed to 1 instead, while maintaining the same proportion of the two domains. (Refer to Figure 3 in the main text for an illustration of INAA.)

IV. EARLY-STOPPING TRAINING STRATEGY AND VALIDATION ACCURACY CURVES

We employed an early-stopping strategy [1] during the training process, and determined the optimal stopping epoch using the exact-match validation accuracy curves (refer to Figure S5).

V. PERFORMANCE OF VARIOUS NN MODELS

The performance of different NN models explored in this work is shown in Figure S6 and Figure 4 in the main text.

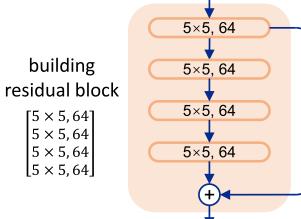
VI. INVERSE-DESIGN RESULTS

Additional inverse-design results are presented in Figure S7 (showing ‘good’ results) and Figure S8 (showing ‘bad’ results). Figure S9 highlights the importance of inverse design when the resolution of a human-designed pattern is finner than that of the template.

[1] Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press, 2016.

[2] He, K. M.; Zhang, X. Y.; Ren, S. Q.; Sun, J. Deep residual learning for image recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2016; pp 770–778.

TABLE S2. Architectures of neural network models (a) X-CNN₂₄-RES₁₂-SCN_{ON}, (b) X-CNN₃₂-RES₁₆-SCN_{ON}, (c) X-CNN₃₂-RES₁₂-SCN_{ON} and (d) X-CNN₃₂-RES₈-SCN_{ON}. The residual blocks used in these models are shown in brackets, with a typical example shown at the top of the table. All of these networks are equipped with SCN. Please refer to the caption of Table 2 in the main text for the naming conventions of these neural network models.



NN Model	a X-CNN ₂₄ -RES ₁₂ -SCN _{ON}	b X-CNN ₃₂ -RES ₁₆ -SCN _{ON}	c X-CNN ₃₂ -RES ₁₂ -SCN _{ON}	d X-CNN ₃₂ -RES ₈ -SCN _{ON}
Block1	$\begin{bmatrix} 5 \times 5, 64 \\ 5 \times 5, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 5 \times 5, 64 \\ 5 \times 5, 64 \end{bmatrix} \times 4$	$\begin{bmatrix} 5 \times 5, 64 \\ 5 \times 5, 64 \\ 5 \times 5, 64 \\ 5 \times 5, 64 \end{bmatrix} \times 2$ $\begin{bmatrix} 5 \times 5, 64 \\ 5 \times 5, 64 \end{bmatrix} \times 1$	$\begin{bmatrix} 5 \times 5, 64 \\ 5 \times 5, 64 \\ 5 \times 5, 64 \\ 5 \times 5, 64 \end{bmatrix} \times 2$
	2 × 2 maxpool, stride 2			
Block2	$\begin{bmatrix} 5 \times 5, 64 \\ 5 \times 5, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 5 \times 5, 64 \\ 5 \times 5, 64 \end{bmatrix} \times 4$	$\begin{bmatrix} 5 \times 5, 64 \\ 5 \times 5, 64 \\ 5 \times 5, 64 \\ 5 \times 5, 64 \end{bmatrix} \times 2$ $\begin{bmatrix} 5 \times 5, 64 \\ 5 \times 5, 64 \end{bmatrix} \times 1$	$\begin{bmatrix} 5 \times 5, 64 \\ 5 \times 5, 64 \\ 5 \times 5, 64 \\ 5 \times 5, 64 \end{bmatrix} \times 2$
	2 × 2 maxpool, stride 2			
Block3	$\begin{bmatrix} 5 \times 5, 64 \\ 5 \times 5, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 5 \times 5, 64 \\ 5 \times 5, 64 \end{bmatrix} \times 4$	$\begin{bmatrix} 5 \times 5, 64 \\ 5 \times 5, 64 \\ 5 \times 5, 64 \\ 5 \times 5, 64 \end{bmatrix} \times 2$ $\begin{bmatrix} 5 \times 5, 64 \\ 5 \times 5, 64 \end{bmatrix} \times 1$	$\begin{bmatrix} 5 \times 5, 64 \\ 5 \times 5, 64 \\ 5 \times 5, 64 \\ 5 \times 5, 64 \end{bmatrix} \times 2$
	2 × 2 maxpool, stride 2			
Block4	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \\ 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$ $\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 1$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \\ 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
	flatten			
dense layers	dense, 512, dropout			
	dense, 1024, dropout			
	dense, 25, sigmoid			

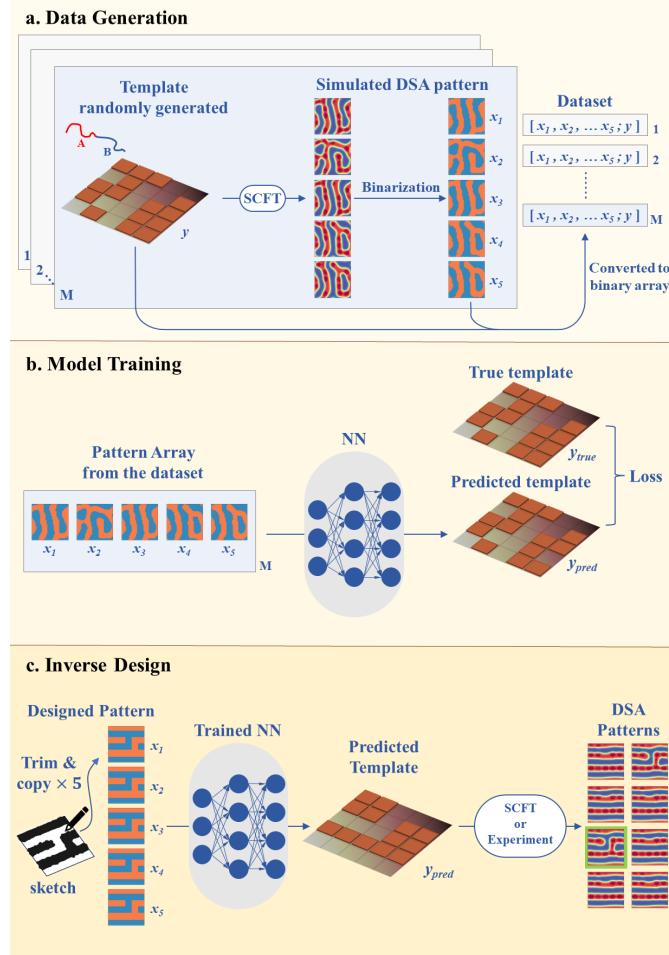


FIG. S1. Flow chart (MCIS version). (a) Data generation. A dataset is prepared by SCFT simulations, each sample of which consists of a template y and five corresponding DSA patterns $[x_1, x_2, \dots, x_5]$. The number and position of the adsorptive posts on the template are randomly generated, and SCFT is used to simulate the DSA process of A/B diblocks on the template. The template and the DSA patterns are then converted into 5×5 and 40×40 binary arrays, respectively. About 90,000 samples are prepared through this way, which will be further divided into training, validation and test datasets. (b) Model training. A multi-channel-input neural network is trained on the dataset to learn to predict the correct template for the input patterns. (c) Inverse Design. The trained NN is asked to predict a template for a designed pattern drawn by hand (five copies of the same designed pattern will be input into the five channels). The DSA patterns forms on the predicted template are expected to recover the designed pattern which can be tested by SCFT or experiments.

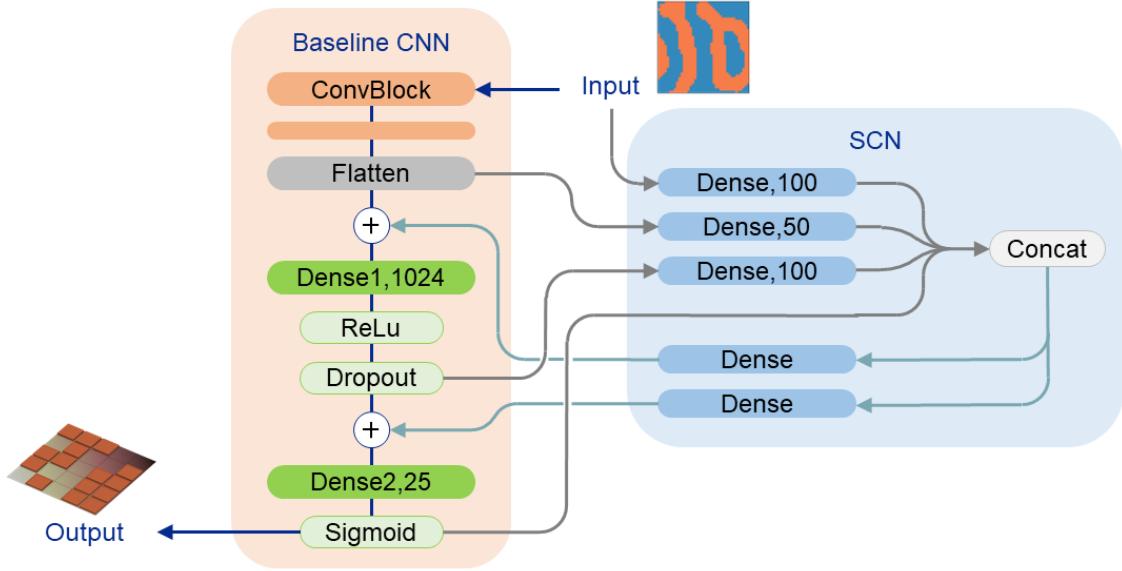


FIG. S2. Schematic diagram of self-conscious network (SCN) applied to the baseline CNN model (S-CNN₂-Res₀-SCN_{ON}).

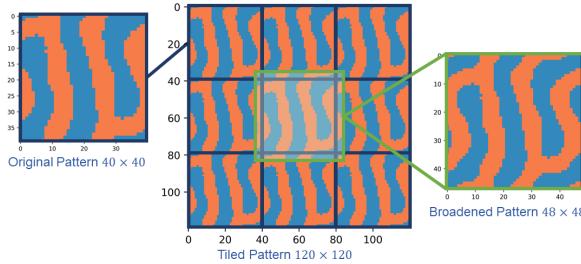


FIG. S3. Pattern Broadening. From left to right shows the original 40×40 pattern array, the 120×120 tiled array, and the 48×48 broadened pattern array cut from the center.

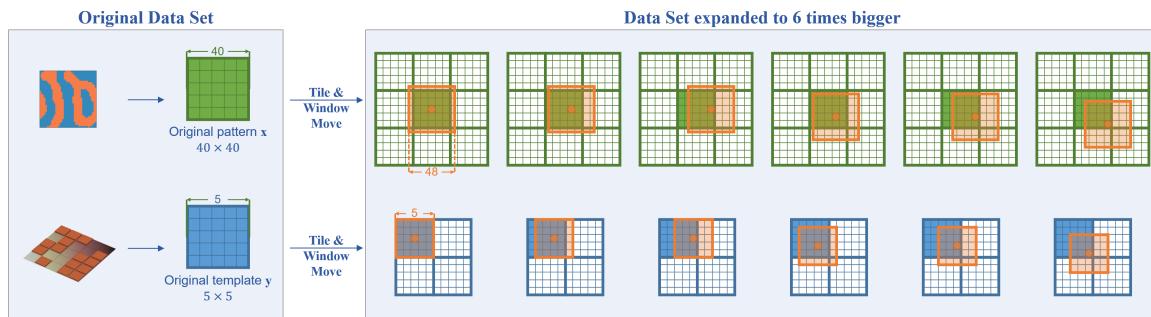


FIG. S4. Window Movement. The left panels show the original dataset arrays, with green and blue grids representing the pattern and template arrays, respectively. The arrays are tiled, and an orange window is moved across the tiled arrays to generate new samples shown in the right panels.

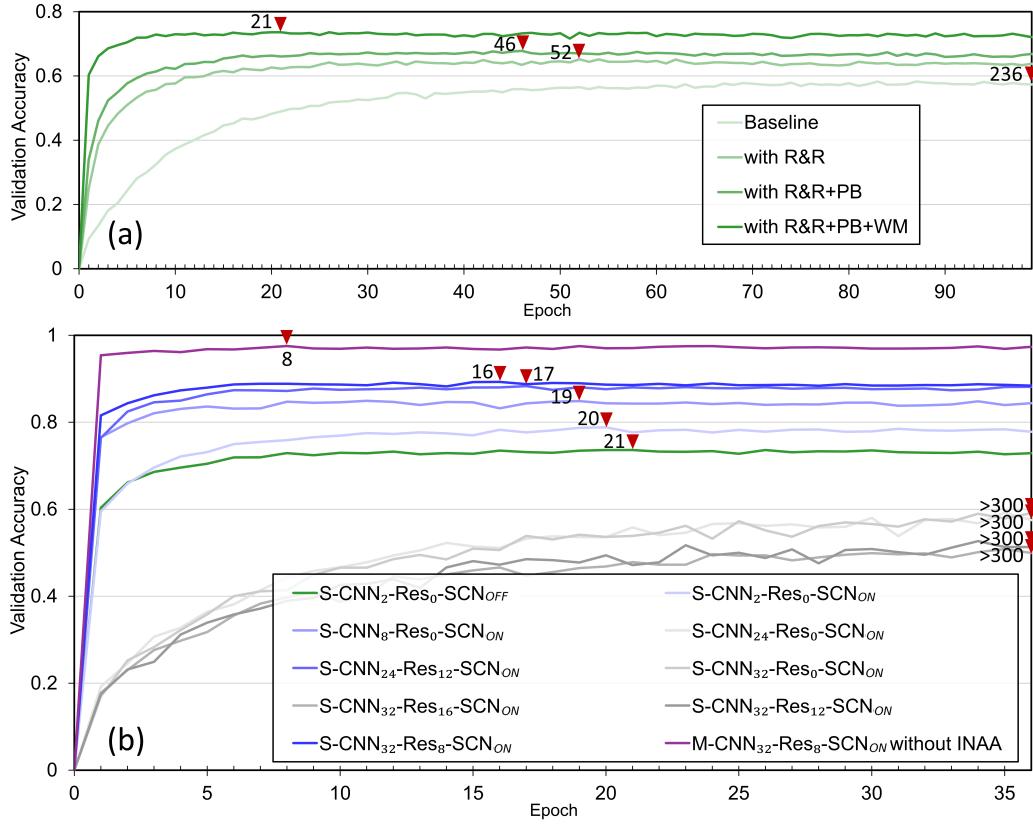


FIG. S5. Validation accuracy curve of the training process of (a) the baseline CNN model without (light green) and with various augmentation methods and (b) various deep models. The red filled triangle shows the early-stopping epoch of each trained model, where the corresponding model achieved the highest exact-match accuracy on the validation dataset on average (over four training processes).

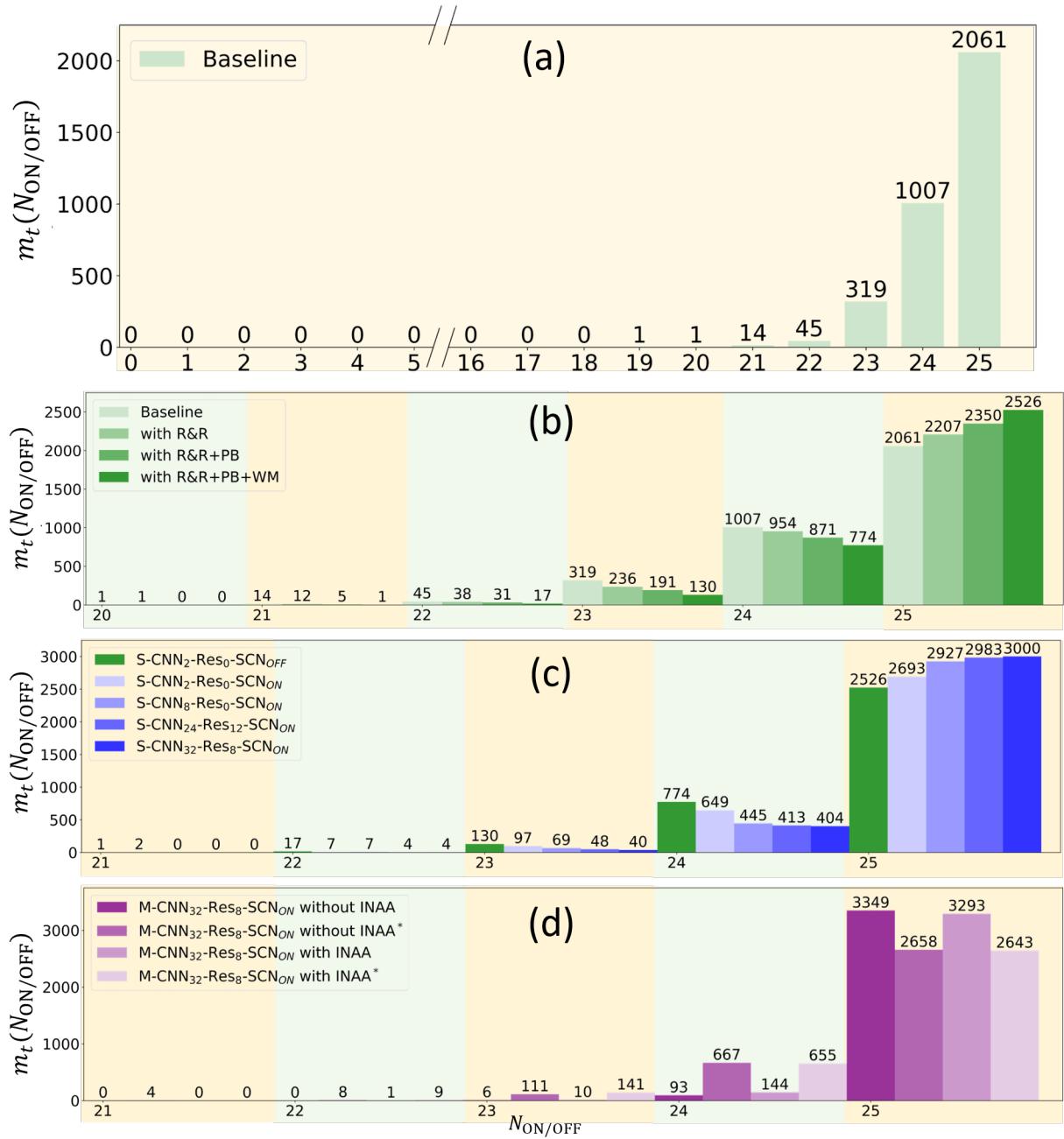


FIG. S6. Performance comparison of NN models investigated in this work in terms of $m_t(N_{ON/OFF})$, which indicates the number of DSA patterns in the test dataset that have the corresponding predicted templates with exactly $N_{ON/OFF}$ correctly-predicted ON/OFF posts. $N_{ON/OFF} = 25$ indicates the exact-match prediction. (a) shows m_t of the baseline CNN model (or S-CNN₂-Res₀-SCN_{OFF} without data augmentation). (b) shows the baseline CNN model with various data augmentation techniques (see Section III in SI). (c) shows some deepened models based on the baseline CNN model. (d) shows four models that incorporate a multi-channel input strategy with interface noise adding augmentation (INAA) or without INAA. The star symbol in the model name indicates that during evaluation, all five input channels are the same DSA patterns, while the model name without star symbol indicates that all five input DSA patterns are different. The naming rules of X-CNN_n-Res_m-SCN_Y (see Section 2.2.1 and Table S2) are as follows: X = M (multi-channel) or S (single-channel), n denotes the number of CNN layers employed, m denotes the number of Resnet blocks employed and Y = ON (with SCN module) or OFF (without SCN module).

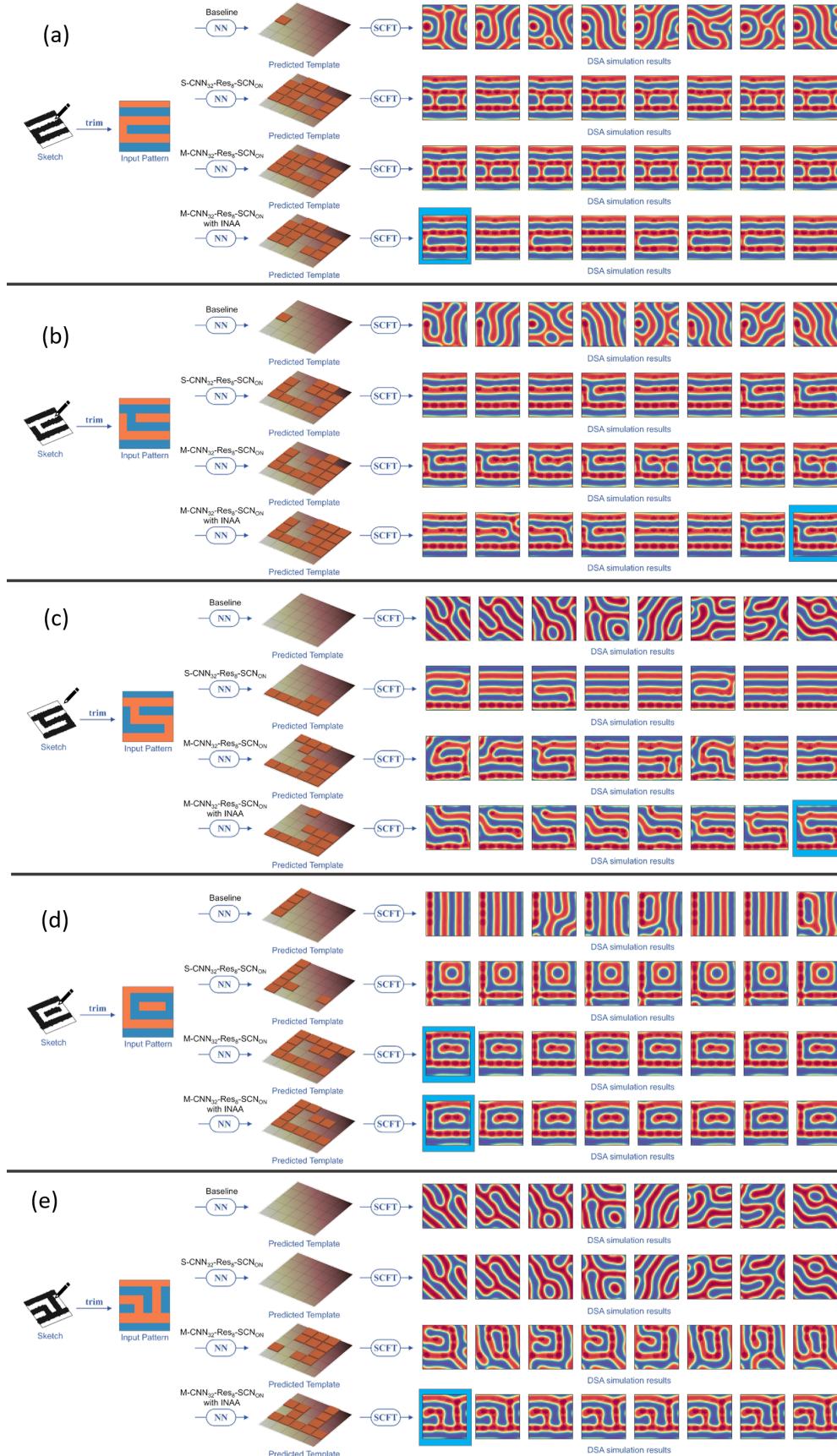


FIG. S7. Five exemplary results of the inverse design of human-designed (HD) patterns. For each HD pattern, four groups of results corresponding to four NN models are presented. The best model in this work (M-CNN₃₂-Res₈-SCN_{ON}) with INAA achieves the best performance in inverse design of all five HD patterns. The best model without INAA ranks the second which only succeeds in pattern (d). The baseline model is the least successful, with predicted templates that appear to be unrelated to the input pattern.

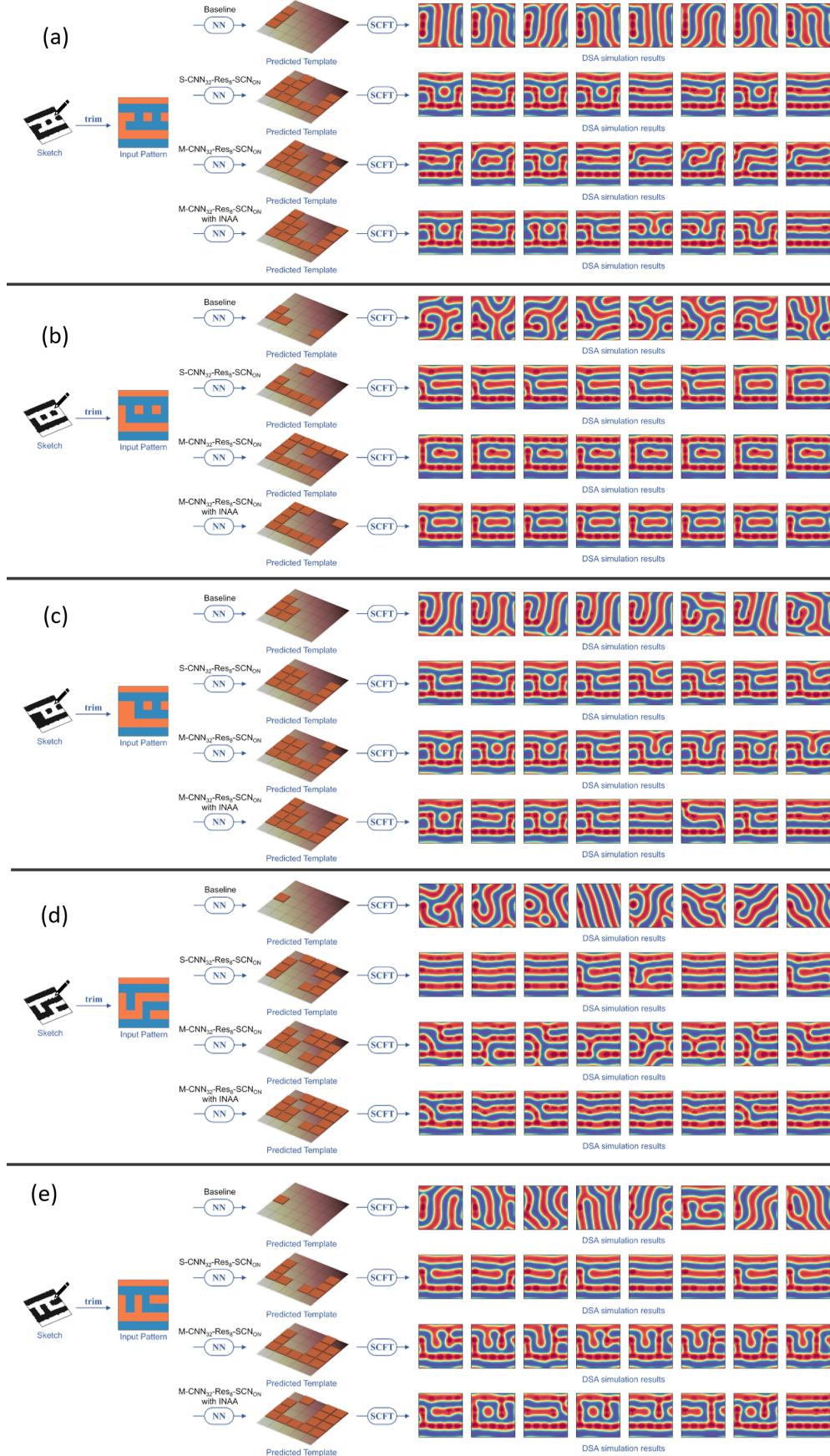


FIG. S8. Five poor results of the inverse design of human-designed (HD) patterns. For each HD pattern, four groups of results corresponding to four NN models are presented. The best model in this work (M-CNN₃₂-Res₈-SCN_{ON}) with INAA outperforms the other models and almost achieves successful inverse designs for all five patterns. The baseline model is the least successful, with predicted templates that appear to be unrelated to the input pattern. It's worth noting that some of these HD patterns may be unsolvable.

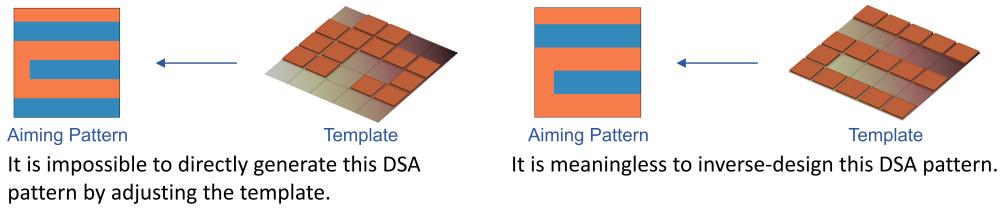


FIG. S9. If the fine structure of the aiming pattern cannot be reproduced by adjusting the guiding template, then an inverse design of the pattern is necessary (left panel). Conversely, if the aiming pattern can be generated by adjusting the template, then an inverse design of the DSA pattern is unnecessary (right panel).

Attractive potential on the A block = $\varepsilon \delta_A(\mathbf{r})$, $\varepsilon = 1.6k_B T$

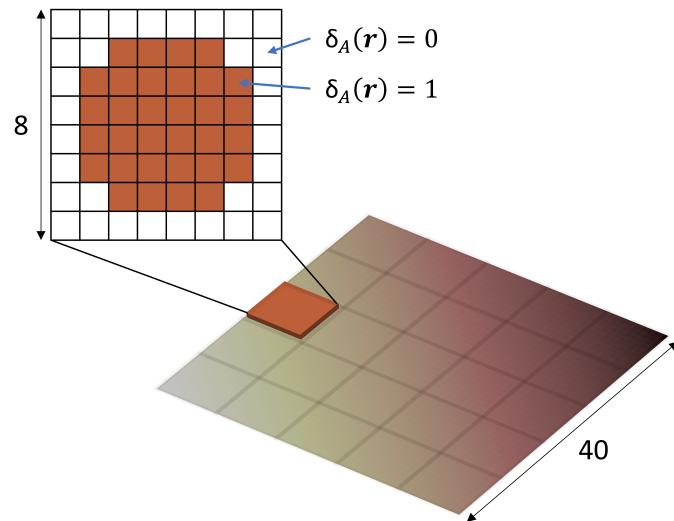


FIG. S10. Illustration of $\delta_A(\mathbf{r})$ function. Only the central lattices (brown) inside the adsorptive post are attractive to A block.

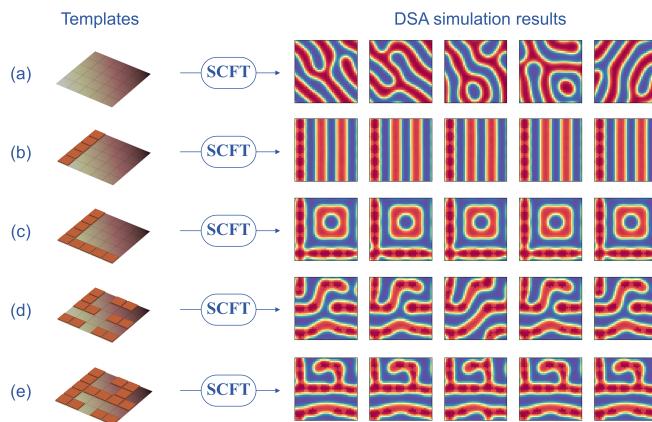


FIG. S11. Illustration of defect occurrence in microphase-separated block copolymer (BCP) patterns guided by templates with varying complexity. **a.** In the absence of template guidance, SCFT produces highly heterogeneous defect patterns. **b. and c.** With a simple arrangement of template posts, the template effectively guides the generation of regular patterns without any defects. **d. and e.** As the complexity of the templates increases, the probability of defects occurring also increases, and the same template can give rise to multiple patterns.