

Supplementary Information for

Robust freeform metasurface design based on progressively growing generative networks

Fufang Wen[†], Jiaqi Jiang[†] and Jonathan A. Fan*

[†]These two authors contribute equally to this work.

*Email: jonfan@stanford.edu

This PDF file includes:

- Protocol of PGGAN metasurface optimization
- Impact of the self-attention mechanism
- Computational cost calculation
- Images of generated devices
- Comparison with the stretched training set
- Benchmark of RCWA simulation with FDTD simulation
- Figure S1 – S5
- Table S1 – S3

Protocol of PGGAN metasurface optimization

The PGGAN metasurface optimization process contains three parts: training data preparation, PGGAN network growth, and training data growth.

1. Training data preparation

To create the initial sparse training set for the basic GAN and PGGAN, we consider devices that sparsely sample the design parameter space. The training set devices sample the wavelength space in increments of 200 nm and the deflection angle space in increments of 10 degrees (see black solid boxes in Figure 4(a)). For each wavelength-deflection angle pair, we generate 40 initial devices comprising randomly generated dielectric distributions, perform 350 iterations of adjoint-based topology optimization for each device, and keep the top 50% devices (in terms of efficiency) for the training set. For each iteration of gradient-based topology optimization, six individual device simulations per iteration are performed: forward simulations of dilated, intermediate, and eroded patterns and their corresponding adjoint simulations. More details of the topology optimization process can be found in Ref. [31]. In total, the training set contains 600 high-efficiency metasurfaces.

2. PGGAN network growth

The PGGAN network growth process for a single cycle of PGGAN training is summarized in Figure S1 and Table S1 and can be divided into 4 stages. In the first stage, we train the initial fixed network at a spatial resolution of 8 x 16. For each subsequent stage, we spend 5000 training iterations to gradually double the spatial resolution and another 5000 iterations to stabilize the network at the new resolution. Once we reach the full resolution of 64 x 128, we spend 10000 iterations to stabilize the final network architecture.



Figure S1. Examples of device patterns in the training set at different spatial resolutions.

Stage	Resolution	Iteration
1	8 x 16	0k - 5k
	8 x 16 → 16 x 32	5k - 10k
2	16 x 32	10k - 15k
	16 x 32 → 32 x 64	15k - 20k
3	32 x 64	20k - 25k
	32 x 64 → 64 x 128	25k - 30k
4	64 x 128	30k - 40k

Table S1. PGGAN training scheme. Resolutions and number of iterations are shown.

Detailed generator and discriminator architecture information at each stage of network growth is presented in Table S2. The input parameters to the generator include the device wavelength, deflection angle, and an 8 dimensional noise vector. During each increase in resolution, upsample and convolution layers are added to the generator and downsample and deconvolution layers are added to the discriminator. For both generator and discriminator, we employ Adam optimizer with a batch size of 128, the learning rate of 0.001, β_1 of 0, and β_2 of 0.9. We employ the Wasserstein loss with a gradient penalty, where lambda for the discriminator loss is 10. It is noted that the basic GAN architecture is the same as the PGGAN at its final resolution.

In the network, a final deconvolution layer is added at the end of the generator to transform the tensor output to a 2-dimensional image, and an initial convolution layer is added at the beginning of the discriminator to transform the image input into a tensor. For example, in stage 2, a final deconvolution layer with an input size of $32 \times 8 \times 32$ and output size of $1 \times 8 \times 32$ is added to the generator, and an initial convolution layer with input size of $1 \times 8 \times 32$ and output size of $32 \times 8 \times 32$ is added to the discrimination. For conciseness, this is not shown in the tables. In practice, we did not add self-attention layer for the full resolution layer (32×128) due to the huge amount of memory requirement.

Generator

Stage				Layer type	Activation	Input shape	Output shape
4	3	2	1	Fully-connected	LReLU	8+2	512
				Fully-connected	LReLU	512	4096
				Reshape		4096	$64 \times (4 \times 16)$
				Deconv (5×5)	LReLU	$64 \times (4 \times 16)$	$32 \times (8 \times 32)$
				Self-attention	ReLU	$32 \times (8 \times 32)$	$32 \times (8 \times 32)$
				Deconv (5×5)	LReLU	$32 \times (8 \times 32)$	$16 \times (16 \times 64)$
				Self-attention	ReLU	$16 \times (16 \times 64)$	$16 \times (16 \times 64)$
				Deconv (5×5)	LReLU	$16 \times (16 \times 64)$	$8 \times (32 \times 128)$
				Deconv (5×5)		$8 \times (32 \times 128)$	$1 \times (32 \times 128)$

Discriminator

Stage				Layer type	Activation	Input shape	Output shape
4	3	2	1	Conv (5×5)		$1 \times (32 \times 128)$	$8 \times (32 \times 128)$
				Conv (5×5)	LReLU	$8 \times (32 \times 128)$	$16 \times (16 \times 64)$
				Conv (5×5)	LReLU	$16 \times (16 \times 64)$	$32 \times (8 \times 32)$
				Self-attention	ReLU	$32 \times (8 \times 32)$	$32 \times (8 \times 32)$
				Conv (5×5)	LReLU	$32 \times (8 \times 32)$	$64 \times (4 \times 16)$
				Self-attention	ReLU	$64 \times (4 \times 16)$	$64 \times (4 \times 16)$
				Reshape		$64 \times (4 \times 16)$	4096
				Fully-connected	LReLU	4096+2	512
				Fully-connected	LReLU	512	512
				Fully-connected		512	1

Table S2. Detailed network architecture information. Both deconvolution and convolution layer are followed immediately by a leaky ReLU with leakiness 0.2 as well as a batch normalization. The deconvolution/convolution layers have a filter size of 5 x 5 and stride of 2. The network resolutions are indicated in brackets. The size of the first dimension of the output image is doubled because of symmetry.

3. PGGAN training set growth

We perform PGGAN training set growth at the end of each PGGAN training cycle. During training set growth, we consider wavelengths ranging from 500 nm to 1300 nm, in increments of 50 nm, and a deflection angle ranging from 35 degrees to 85 degrees, in increments of 5 degrees, and we generate 2000 devices operating at each of these wavelength-deflection angle pairs. For each device, we calculate the efficiencies of its dilated, intermediate, and eroded form using the RCWA EM solver and determine the efficiency performance metric of the device, as defined in the main text. Devices that have efficiency metrics higher than the average efficiency metric of the training set, for a given wavelength-deflection angle pair, are added to the training set. At the same time, we also remove devices that have relatively low overall performance, which can impede the ability of the PGGAN to optimally train. In this process, we identify the highest efficiency device in the training set for a given wavelength-deflection angle pair and we remove devices that have efficiencies lower than 85% of this highest efficiency value. This method of training set refinement ensures that the performance of the training set, as quantified by the average weighted efficiency, improves over the course of network training.

We note that this sampling of deflection angles and wavelengths is relatively dense and that many of these wavelength-deflection angle pairs are not covered in the initial training set. We add the five best PGGAN-generated devices for each of these wavelength-deflection angle pairs (not covered by the initial training set) to the training set during the first training set growth step.

Impact of the self-attention mechanism

To show the effectiveness the self-attention mechanism, we train the PGGAN with and without the self-attention mechanism in the final training cycle. The result is shown in Figure S2. The addition of self-attention boosts the average PGGAN-produced device efficiency by about 10%.

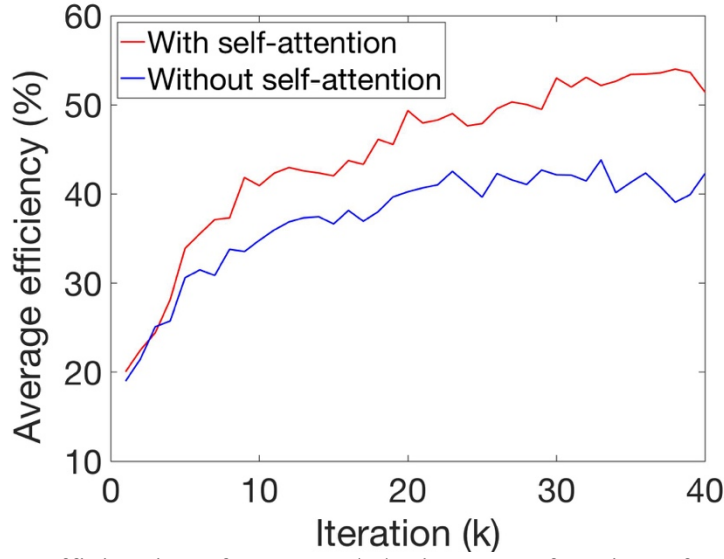


Figure S2. Average efficiencies of generated devices as a function of training iteration for the PPGAN with the self-attention mechanism (red) and PPGAN without the self-attention mechanism (blue), during the final PPGAN training cycle.

Computational cost calculation

In this section, we further break down the computation cost of the optimization methods featured in Figure 4 of the main text.

1. Gradient-based topology optimization

In testing device performance, we consider operating wavelengths ranging from 500 nm to 1300 nm, in increments of 50 nm, and deflection angles ranging from 35 degrees to 85 degrees, in increments of 5 degrees, which lead to 17×11 wavelength-deflection angle pairs. For each wavelength-deflection angle pair, we perform 350 iterations of adjoint-based topology optimization on a total of 50 initial random devices [S1]. Figure S3 shows the efficiency histograms of these devices, which exhibit a wide range of device efficiencies and demonstrates the need for optimizing a few tens of structures to achieve particularly high performing devices. The total number of simulations required to optimize these 9,350 devices is $9,350 \times 350 \times 6 = 19,635,000$.

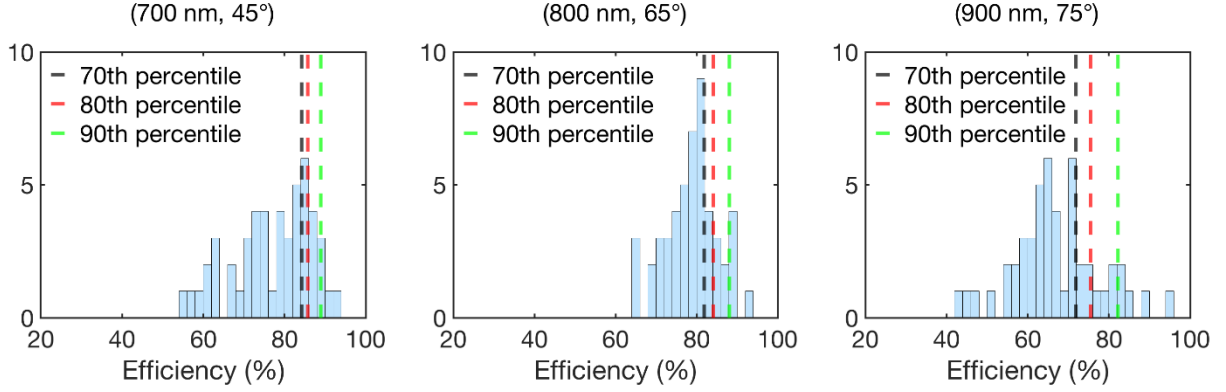


Figure S3. Results of 350 iterations gradient-based topology optimization for different number of initial randomly generated devices. Devices operating at (700 nm, 45°), (800 nm, 65°) and (900 nm, 75°) are shown as examples. 70th, 80th and 90th percentiles are shown in each plot.

2. PGGAN

The preparation of the initial training set utilizes 350 iterations of adjoint-based topology optimization for 1,200 randomly generated devices, which requires $1,200 \times 350 \times 6 = 2,520,000$ simulations. For each training set update at the end of each training cycle, we simulate the weighted efficiencies of 2,000 devices at 17×11 wavelength-deflection angle pairs. After the last PGGAN training cycle, we generate and evaluate 2,000 devices for each wavelength-deflection angle pair. The total computational cost includes the preparation of the initial training set, the training set updates, and the evaluation of the PGGAN-generated devices from a fully trained network. The total number of simulations is: $2,520,000 + 3 \times 3 \times 17 \times 11 \times 2,000 + 3 \times 17 \times 11 \times 2,000 = 7,008,000$ simulations. These simulations can be partitioned as a one-time cost to train the network ($2,520,000 + 3 \times 3 \times 17 \times 11 \times 2,000 = 5,886,000$) and simulations required to evaluate generated devices from the final network ($3 \times 17 \times 11 \times 2000 = 1,112,000$).

Images of generated devices

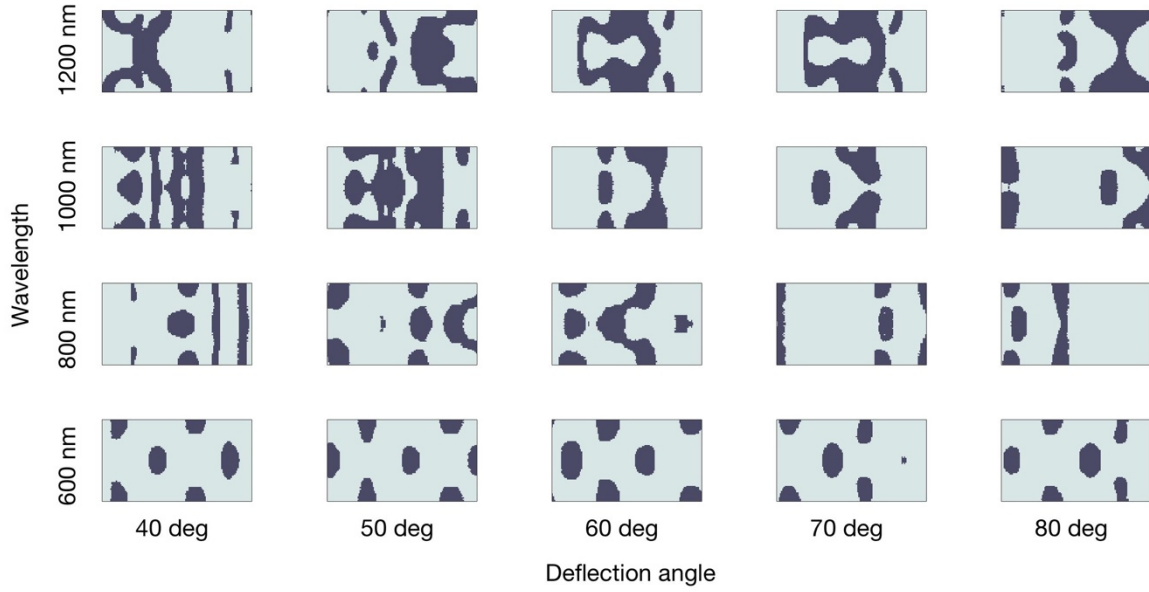


Figure S4. Examples of PPGAN-generated devices for different wavelengths and deflection angles. The PPGAN is able to properly interpolate devices with operating parameters outside the initial training set, and it is able to produce a diversity of structural layouts for different wavelength-deflection angle pairs.

Comparison with the stretched training set

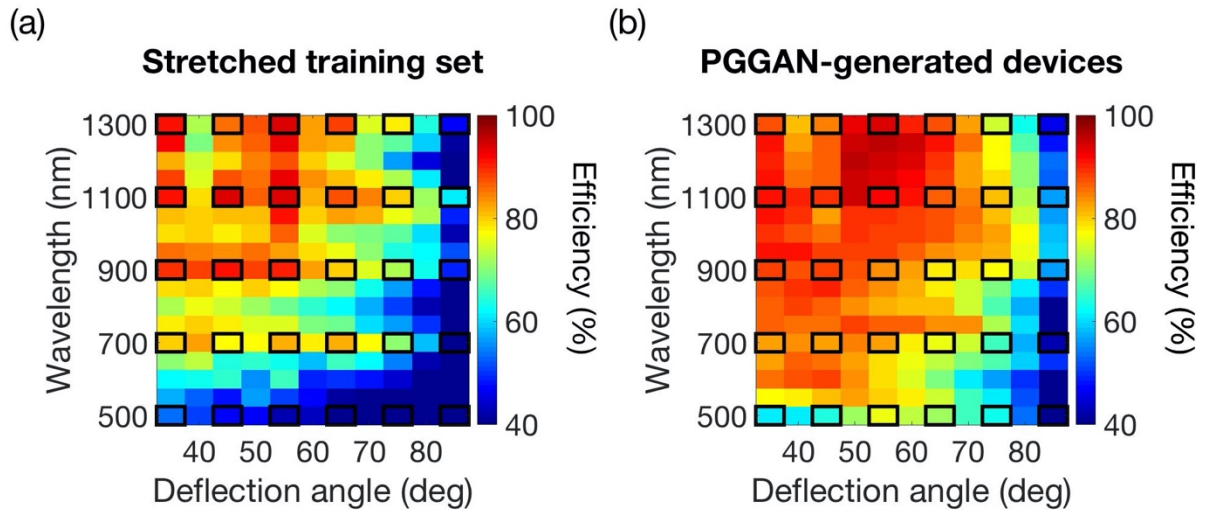


Figure S5. (a) Plot of the highest device efficiencies from the stretched training set for differing wavelength and deflection angle pairs. The devices in the training set have been geometrically scaled to operate over differing wavelengths and deflection angles. (b) Plot of the highest device efficiencies generated by PPGAN.

For each wavelength and deflection angle pair, we test the performance of all devices in the training set and choose the best one. In Figure S5, the stretched training set plot shows that the efficiencies are in general higher at parameters covered by the training set (a, black rectangles) and lower at the nearby points. In contrast, the PGGAN efficiency plot shows that the efficiency is similar for parameters covered by the training set (b, black rectangles) and at points well outside the training set. As a result, the PGGAN is better at interpolation.

Benchmark of RCWA simulation with FDTD simulation

To confirm the RCWA simulation is accurate for freeform metgratings, we compare the result of Reticolo RCWA [S2] and full-wave simulation Lumerical FDTD [S3]. The patterns used in RCWA and FDTD are pixelated images with same resolution. The comparison is shown below. RCWA and FDTD converge well.





Pattern	Wavelength (nm)	Deflection angle (degree)	Efficiency (FDTD)	Efficiency (RCWA)
	900	40	85.4%	85.7%
	1000	50	87.0%	87.7%
	1100	40	90.0%	90.0%
	1200	50	95.0%	94.3%

Table S3. Benchmark of Reticolo RCWA simulation and Lumerical FDTD simulation.

References

[S1] Jiang, J.; Sell, D.; Hoyer, S.; Hickey, J.; Yang, J.; Fan, J. Free-Form Diffractive Metagrating Design Based on Generative Adversarial Networks. *ACS Nano* 2019, 13, 8872–8878.

[S2] Hugonin, J.P.; Lalanne, P. Reticolo software for grating analysis. Institut d'Optique, Orsay, France. 2005.

[S3] Solutions FD. Lumerical solutions inc. Vancouver, Canada. 2003.