Supporting Information

A Deep Learning Approach for Objective-Driven All-Dielectric Metasurface Design

Sensong An[1], Clayton Fowler[1], Bowen Zheng[1], Mikhail Y. Shalaginov[2], Hong Tang[1], Hang Li[1], Li Zhou[1], Jun Ding[3], Anuradha Murthy Agarwal[2], Clara Rivero-Baleine[4], Kathleen A. Richardson[5], Tian Gu[2], Juejun Hu[2], Hualiang Zhang[1,]*

[1]Department of Electrical & Computer Engineering, University of Massachusetts Lowell, Lowell, Massachusetts 01854, USA

[2]Department of Materials Science & Engineering, Massachusetts Institute of Technology, Cambridge, Massachusetts 02139, USA

[3]Shanghai Key Laboratory of Multidimensional Information Processing, East China Normal University, Shanghai 200062, China

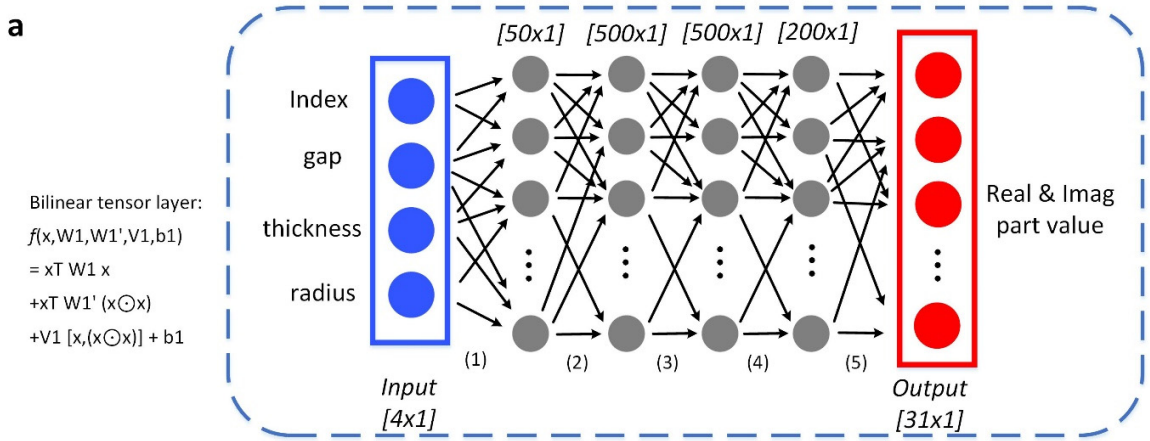
[4]Lockheed Martin Corporation, Orlando, Florida 32819, USA

[5]CREOL, University of Central Florida, Orlando, Florida 32816, USA

In this Supporting Information, we provide further details on network modeling, network training and additional examples showcasing the performances of different networks. This Supplementary Information comprises the following Sections:

Section I – Detailed information of the network architecture

Fig. S1 illustrates the network architecture of the proposed PNN and meta-filter design network, with the data flow details included. As shown in Fig. S1a, the PNN consists of a bilinear tensor layer, followed by four consecutive fully-connected layers. A ReLU activation function is applied to the output tensor of each layer except for the last one. The real and imaginary parts of the complex transmission coefficient are predicted using two independent networks, which share the same network architecture.



Full **PNN** architecture:

(1) W1,W1'=[50x4x4], V1=[50x8], b1=[50x1], y1=ReLU(*f*(x,W1,W1',V1,b1))
(2) W2=[50x500], b2=[500x1], y2=ReLU(W2 × y1 + b2)
(3) W3=[500x500], b3=[500x1], y3=ReLU(W3 × y2 + b3)
(4) W4=[500x500], b4=[500x1], y4=ReLU(W4 × y3 + b4)
(5) W5=[500x31], b5=[31x1], y5=W5 × y4 + b5

Full **meta-filter** design network architecture:

(1) W1=[31x500], b1=[500x1], y1=ReLU(W1 × x +b1)
(2) W2=[500x500], b2=[500x1], y2=ReLU(W2 × y1 + b2)
(3) W3=[500x500], b3=[500x1], y3=ReLU(W3 × y2 + b3)
(4) W4=[500x200], b4=[200x1], y4=ReLU(W4 × y3 + b4)
(5) W5=[200x31], b5=[31x1], y5=W5 × y4 + b5

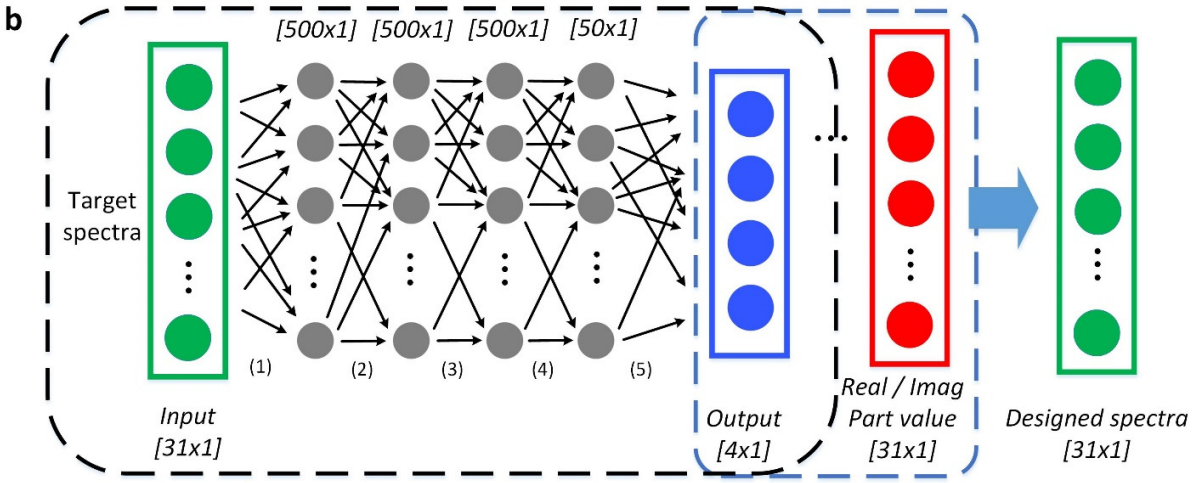


Figure S1. The network architectures of the PNN & meta-filter design network. a Illustration of the prediction neural network. Details, including four fully-connected hidden layers and five weight arrays, are given in the figure. **b** Detailed network structures for the training of the meta-filter design network. The designs are generated by the meta-filter design network (circled in black dot line). The pre-trained PNN (circled in blue dot line) is cascaded to the meta-filter design network to evaluate the optical performances of generated designs.

The complete network constructed for the training of the meta-filter design network is illustrated in Fig. S1b. After the PNNs for the real & imaginary parts are constructed and well-trained, values of the weight and bias arrays are fixed, saved, and then cascaded to the meta-filter design network for meta-filters' performance evaluation, as shown in the blue dot circle in Fig. S1b. After one target spectra is fed into the meta-filter design network (outlined by a black dash-line contour), corresponding design parameters are generated and then evaluated by the well-trained PNN, where the designed spectra are finally obtained and compared with the target. Differences between them are minimized through the training.

Section II – Hyperparameters used in the DNN training process

Hyperparameters used in the training for both PNN and meta-filter networks are shown in Table S1. The hardware consists of a quad-core CPU with 3.5 GHz clock speed, 64 Gigabytes of RAM and two NVidia 1080Ti GPUs. As shown in the table, after 100,000 iterations, the average test set error stabilized at 0.00035 and 0.00023 for the real and imaginary part prediction networks, respectively. With the current hardware setup, the training takes 48 hours for both PNNs before their error rates stabilize. The meta-filters design networks, on the other hand, take more training time due to their more complicated network structures. Compared to the PNNs, they have 4 more layers of hidden neurons included in their structure and are thus more time-consuming for each iteration. All three meta-filter networks are well-trained after 50,000 iterations, each requiring 40 hours. This network is later fed with data from PNN dataset and trained, the small test error of 5.3% also supports the conclusion that the training is completed and the relatively large errors with previous Gaussian targets actually manifest the inherent difficulty of achieving increasingly complex filter functions.

Table S1. Hyperparameters used in the training of PNN and meta-filter design network

Hyperparameters	**PNN (real)**	**PNN (Imaginary)**	**Mata-filter design network**		
			Single Gaussian	Dual-Gaussian	PNN dataset
Training set size	*35000*	*35000*	*20000*	*20000*	*35000*
Test set size	*15000*	*15000*			*15000*
Optimizer	Adam	Adam	Adam		
Learning rate	10^{-6}	10^{-6}	10^{-5}		
Dropout rate	*0.3*	*0.3*	/		
Batch size	*200*	*200*	*200*		
Batch Norm.	No	No	Yes		
Nonlinear activations	ReLU	ReLU	Sigmoid		
Iterations	*100000*	*100000*	*50000*	*50000*	*50000*
Time taken	*48* h	*48* h	*40* h	*40* h	*70* h
Error (train)	*0.000048*	*0.000063*	*10.24%*	*32.05%*	*5.3%*
Error (test)	*0.00035*	*0.00023*			

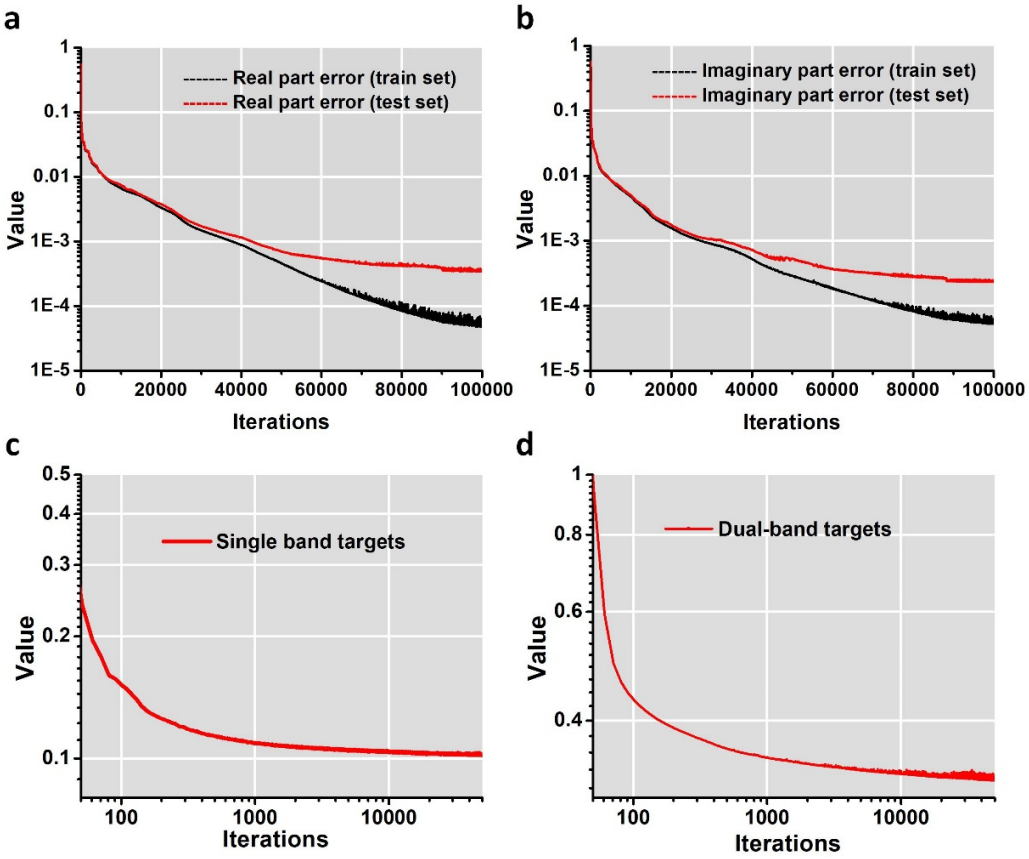


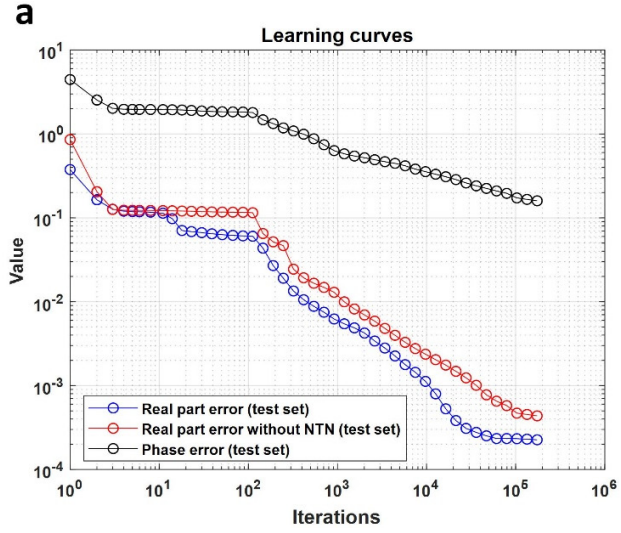
Figure S2. Learning curves of the PNN & meta-filter design networks. Showing in the figures are the learning curves of **a** real part PNN; **b** imaginary part PNN; **c** single-band Gaussian meta-filters design network and **d** dual-band Gaussian meta-filter design network.

Learning curves recording the error history during the training of each network are shown in Fig. S2. As shown in Figs. S2a and S2b, the differences between the training data error (in black) and the test data error (in red) kept increasing during the training process, which is caused by unavoidable overfitting. Overfitting occurs when a statistical model follows a particular set of data too closely (in this case the training data), and may therefore fail to fit additional data (test data) or predict future observations (data that are not included in the dataset) reliably. To minimize this effect and ensure accuracy of the network for test dataset (which is benchmarked as the actual accuracy), we applied a dropout rate of 0.3 to each layer in the PNN. The term "dropout" refers to dropping out units in a neural network, with a dropout rate of 0.3, 30% of the hidden neurons in each hidden layer are randomly eluded when the data is passed on from the previous layer to the next. The ReLU function that was applied to each layer also helped to reduce the overfitting effect. After applying these measures, the overfitting effect is minimized, along with the test data error.

Section III – Ablation analysis to verify the necessity of NTN layers and real & imaginary component prediction approach

To address the importance of the NTN layers and the real & imaginary component prediction approach, we performed an ablation analysis by constructing several additional differing neural networks and trained them with the same dataset. For a fair comparison, two additional neural networks were built: one replaced the NTN layer with an ordinary fully connected layer containing 1x50 neurons, the other one kept the same network architecture but used phase responses over the 30-60 THz frequency band as the training targets. These two networks, together with the network proposed in the paper, are trained with the same dataset, including 35,000 training data samples and 15,000 test data samples. After 200,000 iterations of training, learning curves of these three networks are shown in Fig. S3.

a



b

Error	with NTN (iterations)	without NTN (iterations)
10^{-1}	14	128
10^{-2}	452	1196
10^{-3}	10697	36550

Figure S3. Ablation analysis on the effects of NTN layers and real & imaginary component predicting method. a learning curves of the three different neural networks. **b** Numbers of iterations needed for each network to converge to specified error levels. Both axes are set in log scale for a better view.

Firstly, by replacing the NTN layer with a conventional fully-connected layer, the converging speed was significantly reduced. For the purpose of comparison, we marked the numbers of iterations needed for each network to converge to error values of 10^{-1}, 10^{-2} and 10^{-3} (Fig. S3b). The network with the NTN layer is able to converge more than 3 times faster than the conventional fully-connected network. Secondly, it is extremely hard for the neural network using phase responses as targets to converge. As shown in Fig. S3a, the prediction error of the neural network using phase responses directly is still relatively high (>0.1) even after 100,000 iterations of training (Fig. S3a). To further illustrate how these error values correspond to prediction results, we chose some designs from the test dataset and evaluated their phase responses using the PNN and the neural network using phase responses as targets, respectively. The results are shown in Fig. S4.

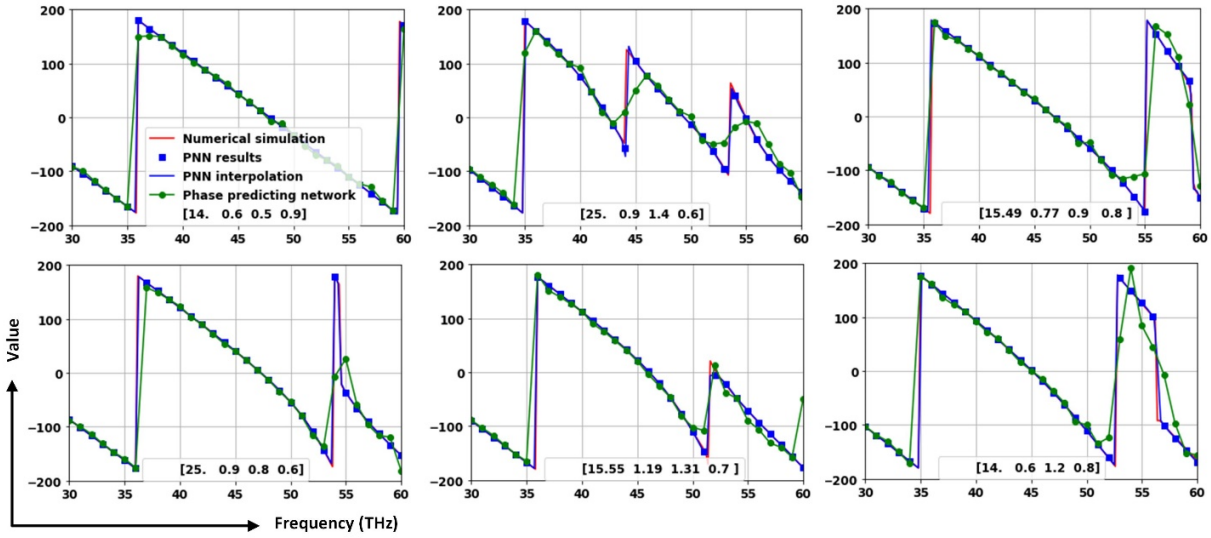


Figure S4. Performance of the proposed PNN and the neural network using phase responses as targets to converge. Red curves are the numerical simulation results from CST. Blue dots represent the phase responses predicted with the PNN and blue curves are the corresponding interpolated spectral phase responses. The phase responses generated by the neural network directly using phase responses are marked in green. All design parameters including permittivity, gap (μm), thickness (μm) and radius (μm) are given as insets.

As shown in Fig. S4, it is hard for the neural network directly using phase responses to predict where the phase jumps would happen and the exact values of the phase jumps. As a result, it simply averaged the phase responses of all 35,000 training data samples, which caused the "rounded" phase responses (in green) around resonant frequencies as shown in Fig. S4. The proposed PNN based on real and imaginary part predictions does not suffer from this issue. As shown in Fig. S4, the prediction results from the PNN (in blue) are in good agreement with the numerical simulation results (in red) across the whole frequency band.

The ablation analysis results have clearly shown that adopting the NTN layer and the real & imaginary component training method accelerated the training process and increased the prediction accuracy.

Section IV – Computation efficiency comparison

We performed several tests to compare the efficiencies of the "forward" PNN and "inverse" design networks with those of traditional design methods. As shown in Table S2, we measured the time-taken for all the designs mentioned in the paper. Since time taken with single PNN prediction or numerical simulation varies from structure to structure, we took an average time based on over 100 results for each method. Depending on the skill and luck of the designer, the time-taken for conventional methods can be very different. To be consistent, here we chose the brute-force method, which is simply performing simulations on all parameter-combination-possible and then picking the best design. More specifically, we counted the number of possibilities for each design parameter based on the fabrication capability (in this case it is assumed to be 0.01 μm), then multiply the numbers together to derive the total number of simulations needed for brute force scan. The total time is then estimated by multiplying this number with the average-time-taken for a one-time numerical simulation.

From the results showing in Table S2, huge computation time savings are achieved by replacing the conventional method with the DNN-based schemes. Interestingly, compared to the meta-atom design networks whose model generators are built with evolutionary algorithms, the meta-filter design network that used a DNN-based model generator is even more time-efficient, indicating the superiority of pure DNN-based design networks.

Table S2. Time comparisons between DNN-based and brutal force methods

		With DNN-based method	With numerical simulations (brute force scan)	Ratio
One-time EM response prediction	**Cylinder meta-atom**	*34.4* ms	*19.4* s	*500*
	H meta-atom	*35.4* ms	*22* s	*600*
Meta-atom design	**Design in Fig. 3**	*22* s	*1.3* million simulations *277* days (estimation)	10^6
	Design in Fig. 4	*253* s	*130* million simulations *77* years (estimation)	10^7
Meta-filter design	**Designs in Fig. 5**	*4.8* ms	*1.3* million simulations *277* days (estimation)	5×10^9

Section V – Additional samples of the PNN

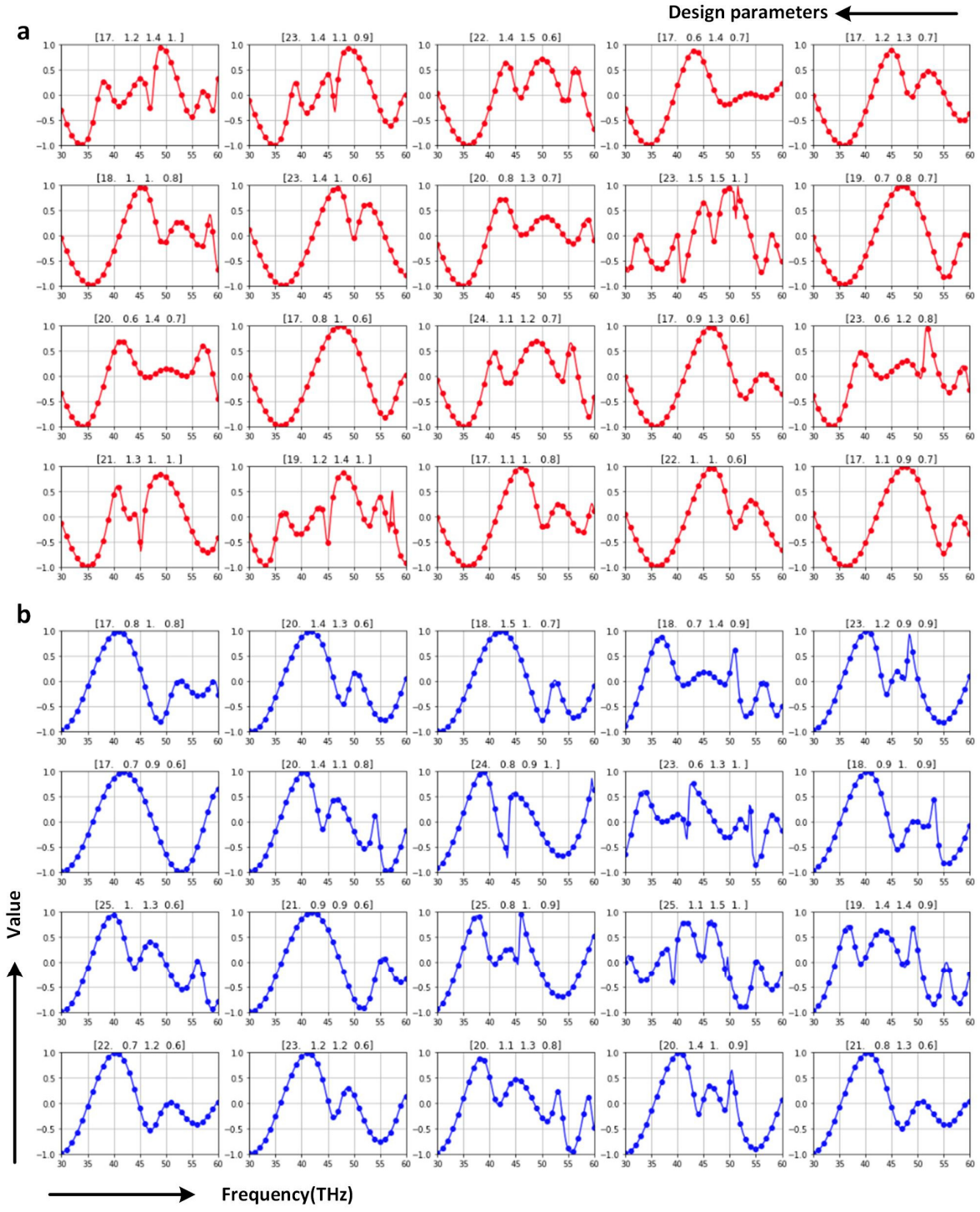


Figure S5. Additional prediction examples generated with the PNN. a Real part prediction examples. **b** Imaginary part prediction examples. In each panel, the dots represent the PNN predicted values. The lines depict numerically simulated values obtained with the commercial frequency domain solver (CST Microwave Studio). Randomly-generated design parameters, including permittivity, gap (μm), thickness (μm) and radius (μm) are shown at the top of each panel.

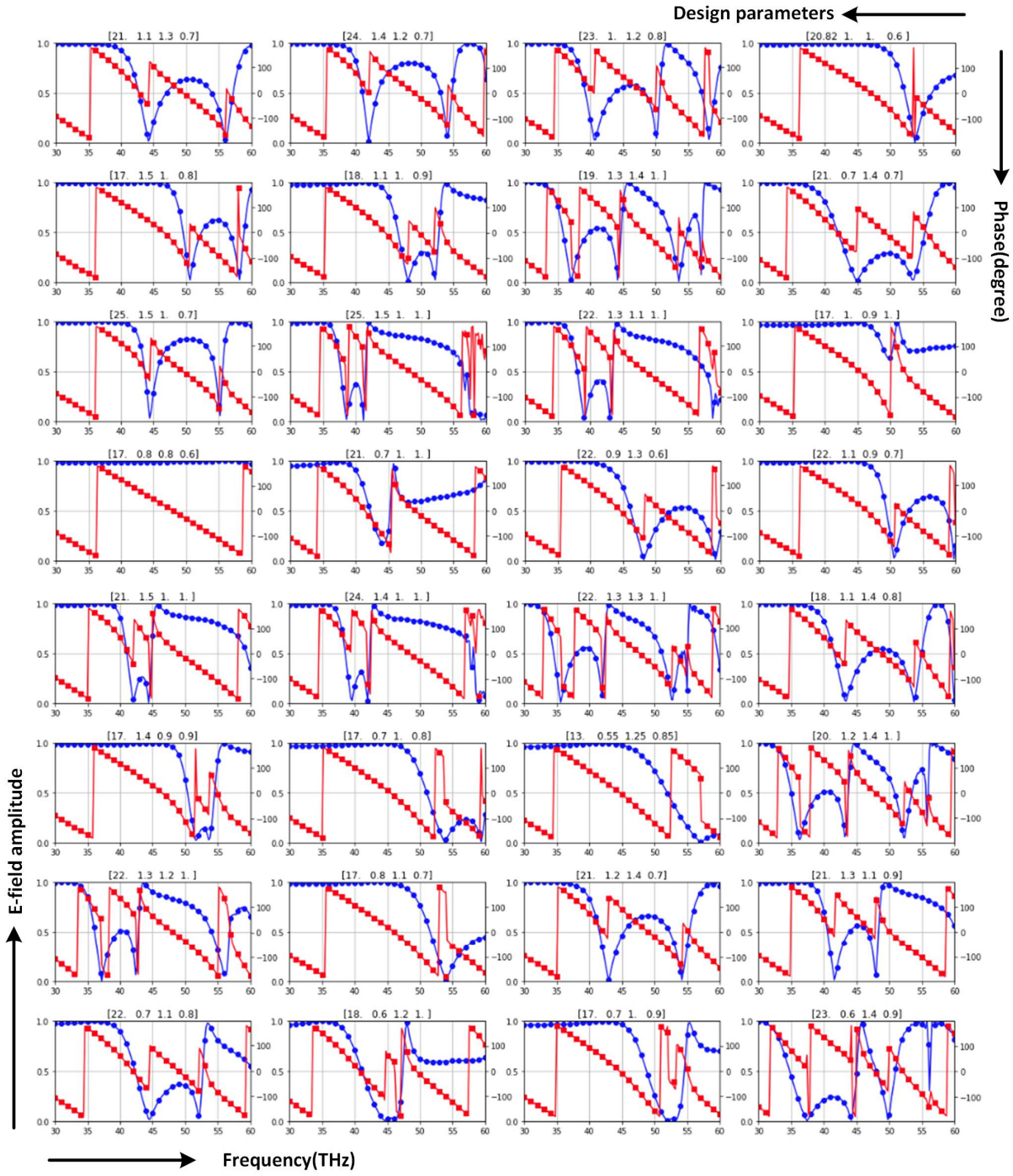


Figure S6. Additional amplitude and phase spectra examples generated from the PNN compared to simulation results. The dots are E-field amplitude (blue) and phase (red) values calculated following the method in equation 1 of the main text, using real & imaginary part values generated from the PNN. Curves are simulation results from the numerical simulation tool CST. Randomly-generated design parameters including permittivity, gap (μm), thickness (μm) and radius (μm) are shown at the top of each panel.

Section VI – Additional samples of the PNN for H-shaped structures

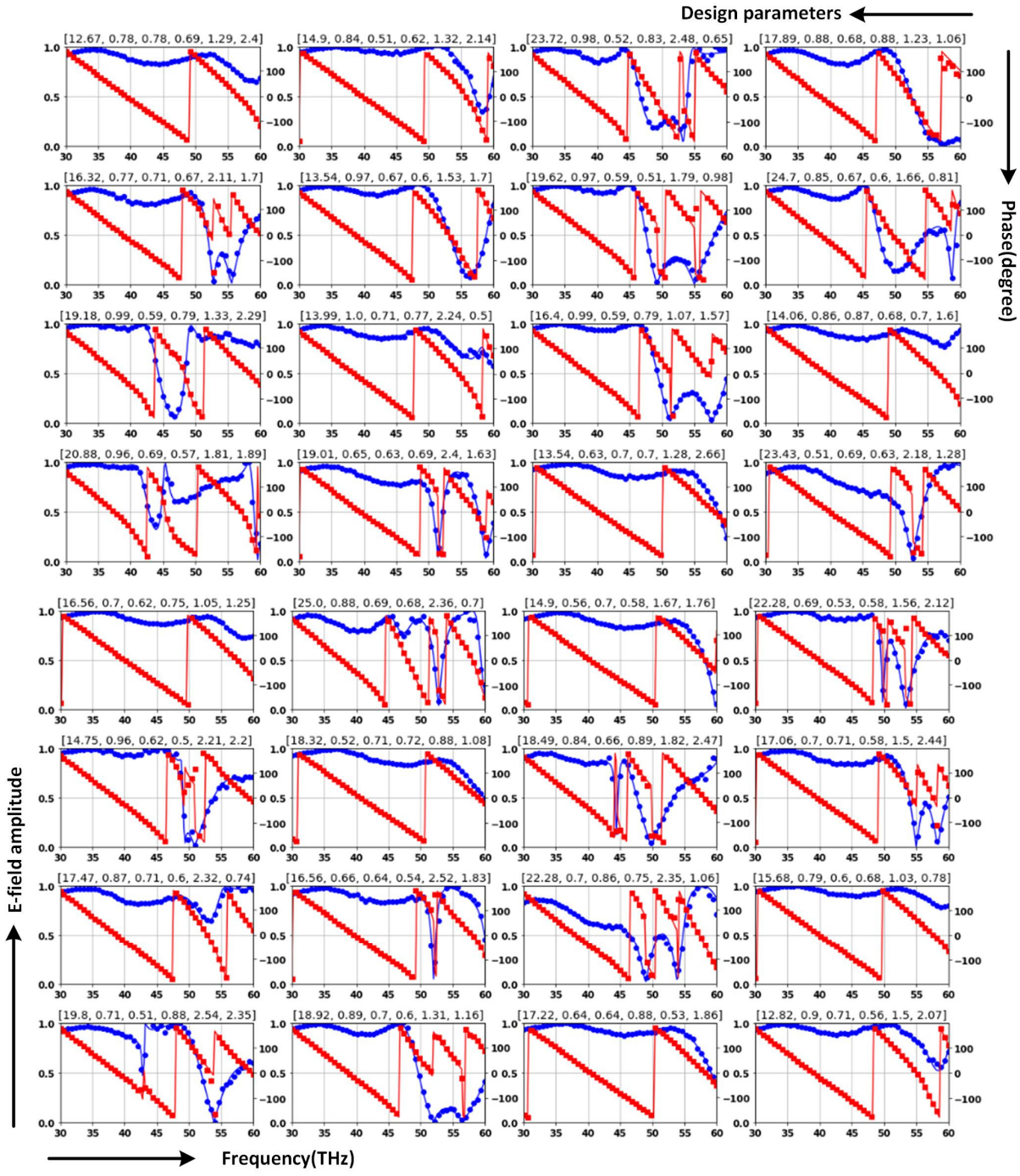


Figure S7. Additional amplitude and phase spectra examples generated from the PNN for H-shaped structures. The dots are E-field amplitude (blue) and phase (red) values calculated following the method in equation 1, using real & imaginary part values generated from the PNN. Curves are simulation results from CST. Randomly-generated design parameters including permittivity, meta-atom thickness (μm), Lx, Lx_1, Ly and Ly_1 (μm) are shown at the top of each panel.

Section VII – Generalization to the modeling of more complicated meta-atoms

To further demonstrate the PNN's potential in modeling more complicated meta-atoms, we constructed a neural network dealing with nearly freeform all-dielectric meta-atoms. As shown in Fig. S8, after adding 4 convolutional layers and training with over 30,000 meta-atom data points, the network is able to converge well and generate accurate phase and amplitude predictions. Here we randomly picked some sample meta-atoms from the test dataset and plotted their spectrum response predicted by the PNN and CST, respectively.

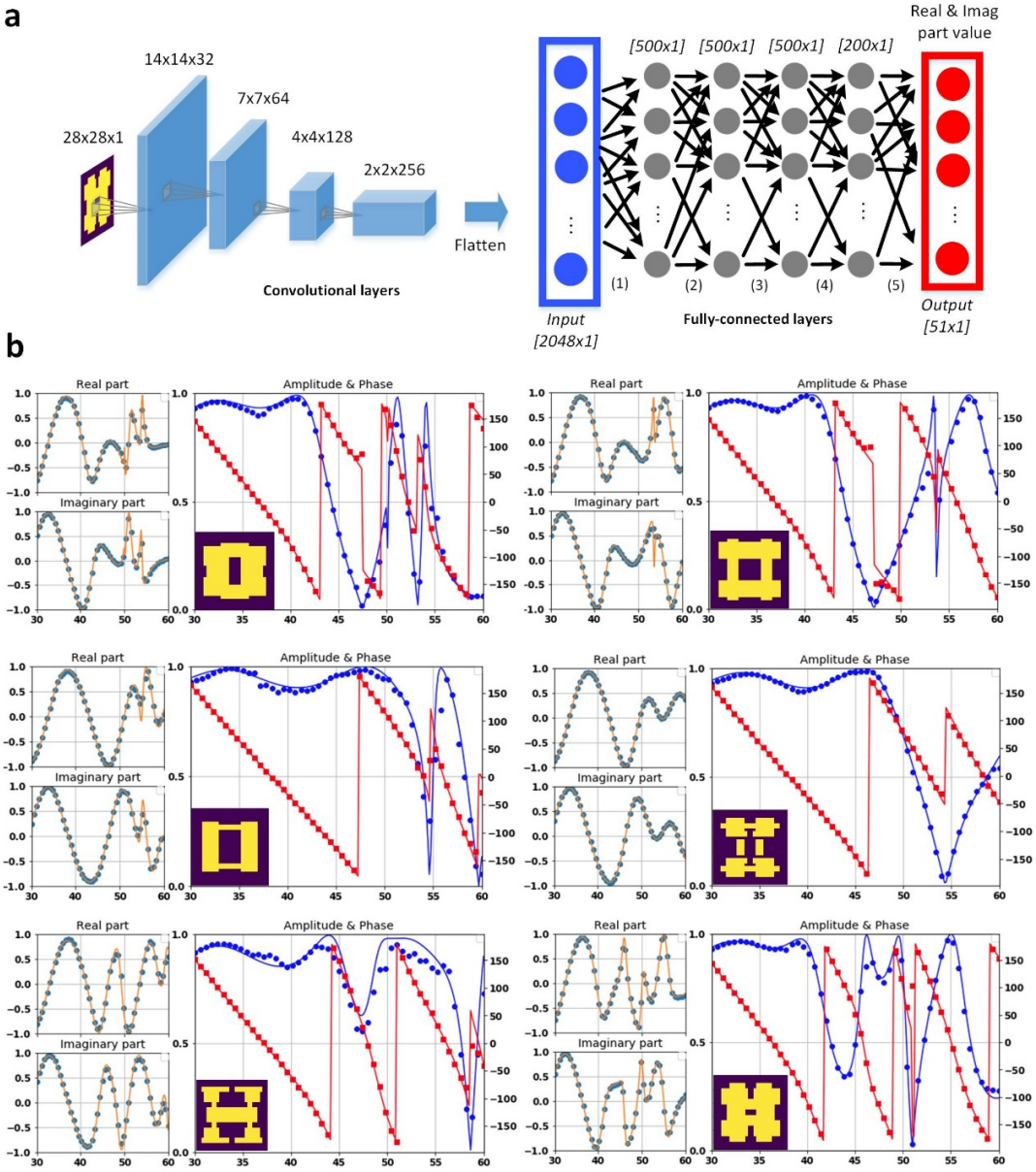


Figure S8. Generalization of the modeling approach to nearly free-form shaped meta-atoms. a Network architecture. Four consecutive convolutional layers were added to the original network to process more complex

shape inputs (treated as 28x28 pixels pattern). The flattened output was further processed with 4 layers of fully-connected layers before the real and imaginary components of the input structure were generated. **b** Examples demonstrating the PNN performance. Small subplots shown on the left are the real and imaginary parts of each meta-atom's transmission coefficient. The red curves shown in the large subplots represent the phase profiles, while the blue curves refer to the amplitude responses. Dots represent data generated by the PNN, while solid curves are data obtained from numerical simulations. Shapes of each meta-atom are given in the insets, while their heights, periodicity and refractive index are fixed at 1 μm, 2.8 μm and 5, respectively, to simplify the problem. All meta-atoms presented were randomly selected from the test data.

Section VIII – Additional samples of the meta-filter design network

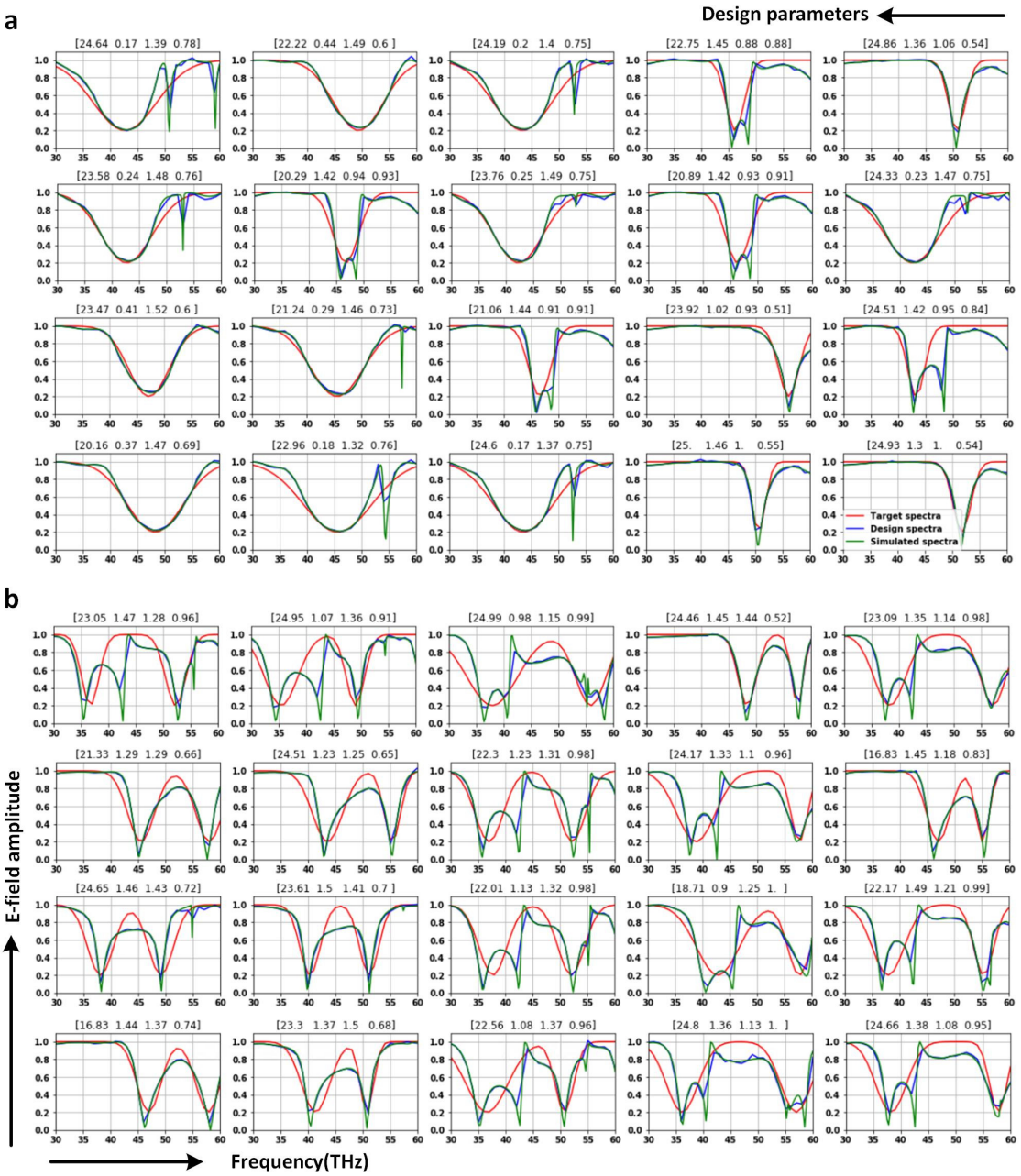


Figure S9. Additional design examples employing the meta-filter design network. a Samples from the single-band target dataset. **b** Samples from the dual-band target dataset. Red curves are target filter spectral responses, and the

blue curves are the PNN-predicted filter spectral responses based on the designs given by the design network, the green curves are the CST-simulated amplitude responses. All design parameters including dielectric constant, gap (μm), radius (μm) and thickness (μm) are shown at the top of each subplot.

Section IX – PNN vs. interpolation

The PNN is a far more sophisticated, precise, and powerful tool compared to interpolation algorithms for two reasons. First, a well-trained PNN offers superior performance in making predictions based on the same prior information. To demonstrate this, we find a unique method to compare the PNN's performance with a built-in interpolation function from the numerical computing tool MATLAB. As shown in Fig. S10a, we first find the 27 groups of data (in blue circles) that are evenly distributed in the parameter range: $\text{gap} \in [1.2, 1.4], \text{thickness} \in [1, 1.2]$ and $\text{radius} \in [0.6, 0.8]$ (all in μm), with a spacing of 0.1 μm for each parameter. Then we randomly chose two test parameter combinations within this parameter space (indicated by the green and red triangles) and predict their transmission coefficients with the PNN and the interpolation tool, respectively. Since these 27 groups are the only training datasets existing in this specific parameter space, the PNN and the interpolation tool have "equal knowledge" about data within this range. Using a spacing of 0.01 μm for each parameter, we created $21 \times 21 \times 21$ (a total of 9,261) query points (blue dots in Fig. S10b) and performed two types of interpolations (linear and cubic) for each query point (including the two test samples). According to the results shown in Figs. S10c and S10d, spectra obtained by the interpolation tools (blue squares and green triangles) are much less accurate compared to the PNN predictions, in particularly at the short wavelength (high frequency) end of the spectrum. In contrast, the PNN-generated spectrum (red circles) maintains high accuracy across the entire 30-60 THz band.

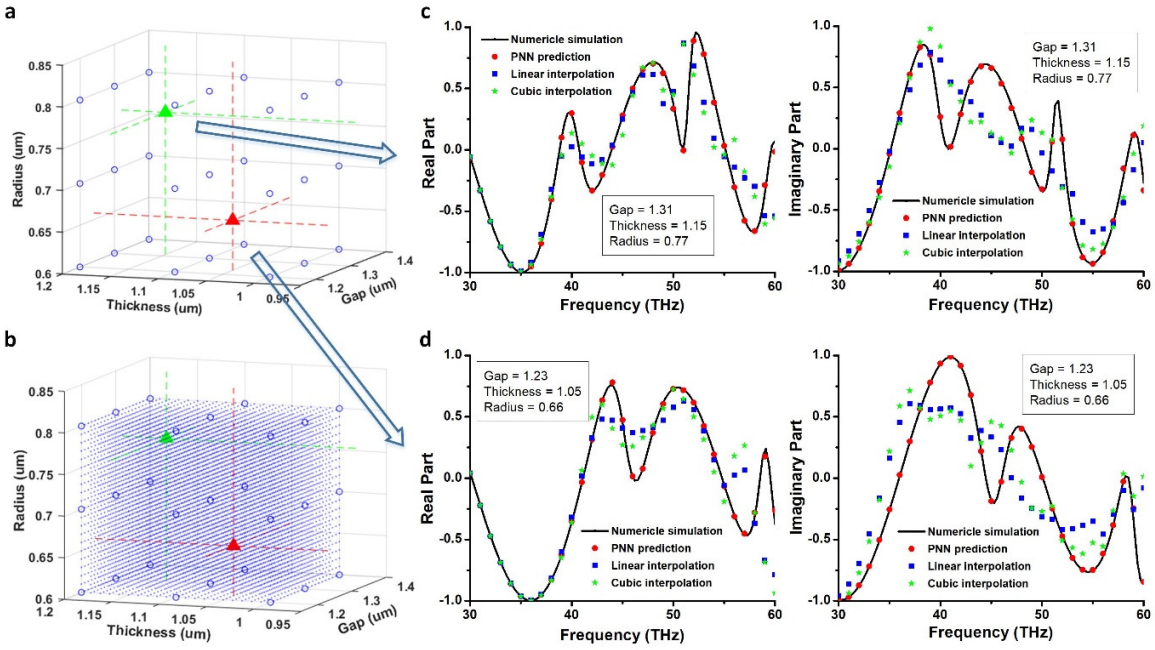


Figure S10. Comparison between PNN and data interpolation. a The parameter scope chosen to perform the comparison. Blue circles indicate the 3 by 3 by 3 data points for interpolation. Green and Red triangles indicate the locations of the two test samples in this parameter scope. All data samples have the same dielectric constant of 24. **b** The meshed parameter scope with 21 by 21 by 21 query points. Both test parameter combinations are covered in these query points. **c** Prediction results with the PNN (red circles), linear interpolation (blue dots), cubic interpolation

(green stars) and numerical simulation results (black lines) for the first test data point (green triangle in a and b) **d** PNN, interpolation and simulation results for the second (red triangle in a and b) test data point. Dimensions of these two test samples are given as insets.

The second reason is that the proposed PNN can also be used for extrapolation to make predictions outside the parameter space of the training data set. Ordinary extrapolation requires assumptions by a designer about the physical behavior outside of the data set (linear, polynomial, etc.), which are unlikely to hold over a significant data range. Moreover, the best choice of model may not be intuitively clear, particularly for multivariable problems. The PNN is better informed than a traditional designer when it comes to making accurate extrapolation predictions, because it can draw much more information from the training data to unravel intrinsic physical behavior of the system. We explored the PNN's "out-of-range" prediction capacity by feeding it with meta-atom structures with one or more parameters residing outside of the preset training data range. As shown in Fig. S11, the PNN retains excellent prediction accuracy when the inputs are not too far beyond the training data set boundaries. This interesting discovery indicates the DNN-based method's potential for uncovering the hidden physical mechanisms behind the large amount of input data. Nonetheless, since the PNN's performance deteriorates as the inputs move far away from the preset data range, it is important that certain boundaries of the collected data, such as fabrication limits, system requirements and design interests should be carefully determined before the network is constructed.

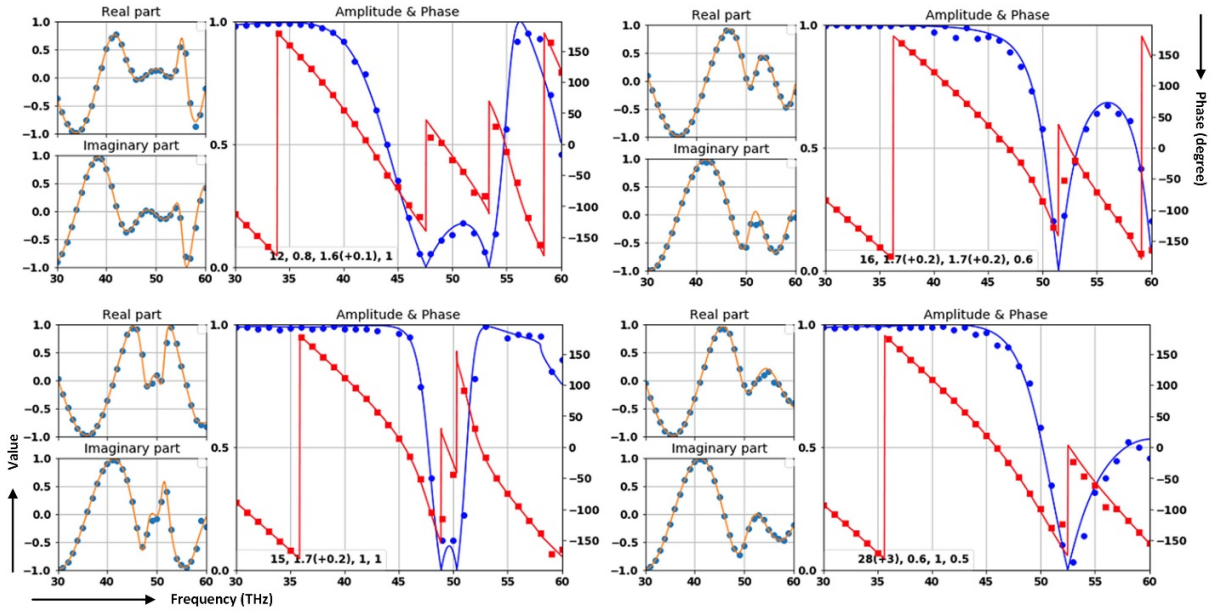


Figure S11. Examples of the proposed PNN with test data outside of the learning range. Smaller subplots shown on the left are the real and imaginary parts of each meta-atom's transmission coefficient. The red curves shown in the larger subplots represent the phase profiles, while the blue curves refer to the amplitude responses. All dots represent data generated by the PNN, while solid curves are data obtained from the numerical simulation tool. Design parameters including the dielectric constant, gap (μm), thickness (μm) and radius (μm) of each meta-atom are given in the insets. All four meta-atoms presented have one or more parameters residing outside of the preset training data range. Values in the parentheses represent the distance from the preset data boundary.