Supplementary for

Spectral Transfer-learning-based Metasurface Design Assisted by Complex-valued Deep Neural Network

Yi Xu,[1] Fu Li,[1] Jianqiang Gu,[1*] Zhiwei Bi,[2] Bing Cao,[2*] Quanlong Yang,[3*] Jiaguang Han,[4] Qinghua Hu,[2] and Weili Zhang[5]

[1]Tianjin University, Center for Terahertz Waves and College of Precision Instrument and Optoelectronics Engineering, Key Laboratory of Optoelectronic Information Technology, Ministry of Education, 300072 Tianjin, People's Republic of China.
[2]Tianjin University, College of Intelligence and Computing, 300072 Tianjin, People's Republic of China.
[3]Central South University, Hunan Key Laboratory of Nanophotonics and Devices, School of Physics and Electronics, Changsha 410083, Hunan, China.
[4]Guilin University of Electronic Technology, Guangxi Key Laboratory of Optoelectronic Information Processing, School of Optoelectronic Engineering, Guilin, 541004, China.
[5]Oklahoma State University, School of Electrical and Computer Engineering, Stillwater, OK 74078, United States.

*gjq@tju.edu.cn

Supplementary 1. Details of the CDNN and its real-valued counterparts

1.1 Training of the source model

The source model is constructed by using the open-source framework Pytorch. The neural network has 11 hidden layers, [50, 500, 500, 500, 500, 500, 500, 500, 500, 500, 200] neurons for each layer in sequence and complex Rectified Linear Unit ($\mathbb{C}$ReLU) as the nonlinear activation function. We randomly initialize the real and imaginary parts of weights and biases with a normal distribution and train the network from scratch. The Adam optimizer with a mini-batch size of 512 and a learning rate of 1×10^{-5} is used.

It should be clarified that the choices of hyperparameters have a great influence on the learning curves. And the hyperparameter configuration is practicable as long as it leads to a stable convergence of the training process and a low value of the testing loss. The above-mentioned configuration is adopted in this work based on training results.

1.2 Complex building blocks

All the building blocks, functions and techniques are based on complex-valued representations and operations here.

1. Complex activation function called $\mathbb{C}$ReLU.

We use the $\mathbb{C}$ReLU as the non-linear activation function because it satisfies the Cauchy-Riemann equations when both the real and imaginary parts are at the same time either strictly positive or strictly negative:

$$\mathbb{C}\mathrm{ReLU}(z) = \mathrm{ReLU}(\mathrm{Re}(z)) + i\mathrm{ReLU}(\mathrm{Im}(z)), \tag{S1}$$

where z is a complex variable, $z = x + iy, x, y \in \mathbb{R}$.

Another applicable non-linear activation function is given by the following equation:

$$\mathbb{C}\mathrm{PolarReLU}(z) = \mathrm{ReLU}(\mathrm{Re}(z)) \cdot \exp\big(i\mathrm{ReLU}(\arg(z))\big), \tag{S2}$$

where $\arg(z) = \arctan\left(\frac{y}{x}\right), \in (-\pi,\pi]$.

Both of them can be good candidates for the nonlinear activation function of CDNN because they have the ability to discriminate the complex information in different quadrants. In our work we use $\mathbb{C}$ReLU for more direct operation on the real and imaginary parts of the complex numbers.

2. Backpropagation of the CDNN.

The objective and activation functions are differentiable with respect to the real and imaginary parts of each complex parameter in the network, which enables the backpropagation in the CDNN. Loss functions are usually real numbers, then the chain rule for complex variables functions is as follows:

$$\nabla_L(t) = \frac{\partial L}{\partial t} = \frac{\partial L}{\partial r} + i\frac{\partial L}{\partial s}$$

$$= \frac{\partial L}{\partial x}\frac{\partial x}{\partial r} + \frac{\partial L}{\partial y}\frac{\partial y}{\partial r} + i\left(\frac{\partial L}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial L}{\partial y}\frac{\partial y}{\partial s}\right)$$

$$= \frac{\partial L}{\partial x}\left(\frac{\partial x}{\partial r} + i\frac{\partial x}{\partial s}\right) + \frac{\partial L}{\partial y}\left(\frac{\partial y}{\partial r} + i\frac{\partial y}{\partial s}\right)$$

$$= \frac{\partial L}{\partial \operatorname{Re}(z)}\left(\frac{\partial \operatorname{Re}(z)}{\partial \operatorname{Re}(t)} + i\frac{\partial \operatorname{Re}(z)}{\partial \operatorname{Im}(t)}\right) + \frac{\partial L}{\partial \operatorname{Im}(z)}\left(\frac{\partial \operatorname{Im}(z)}{\partial \operatorname{Re}(t)} + i\frac{\partial \operatorname{Im}(z)}{\partial \operatorname{Im}(t)}\right)$$

$$= \operatorname{Re}\left(\nabla_L(z)\right)\left(\frac{\partial \operatorname{Re}(z)}{\partial \operatorname{Re}(t)} + i\frac{\partial \operatorname{Re}(z)}{\partial \operatorname{Im}(t)}\right) + \operatorname{Im}\left(\nabla_L(z)\right)\left(\frac{\partial \operatorname{Im}(z)}{\partial \operatorname{Re}(t)} + i\frac{\partial \operatorname{Im}(z)}{\partial \operatorname{Im}(t)}\right) \tag{S3}$$

where L is the loss function, t is another complex variables, $t = r + is, r, t \in \mathbb{R}$, and $z = \tilde{f}(t)$.

1.3 Prediction results of CDNN

Several test samples in the source dataset are randomly presented here in **Figure S1**.

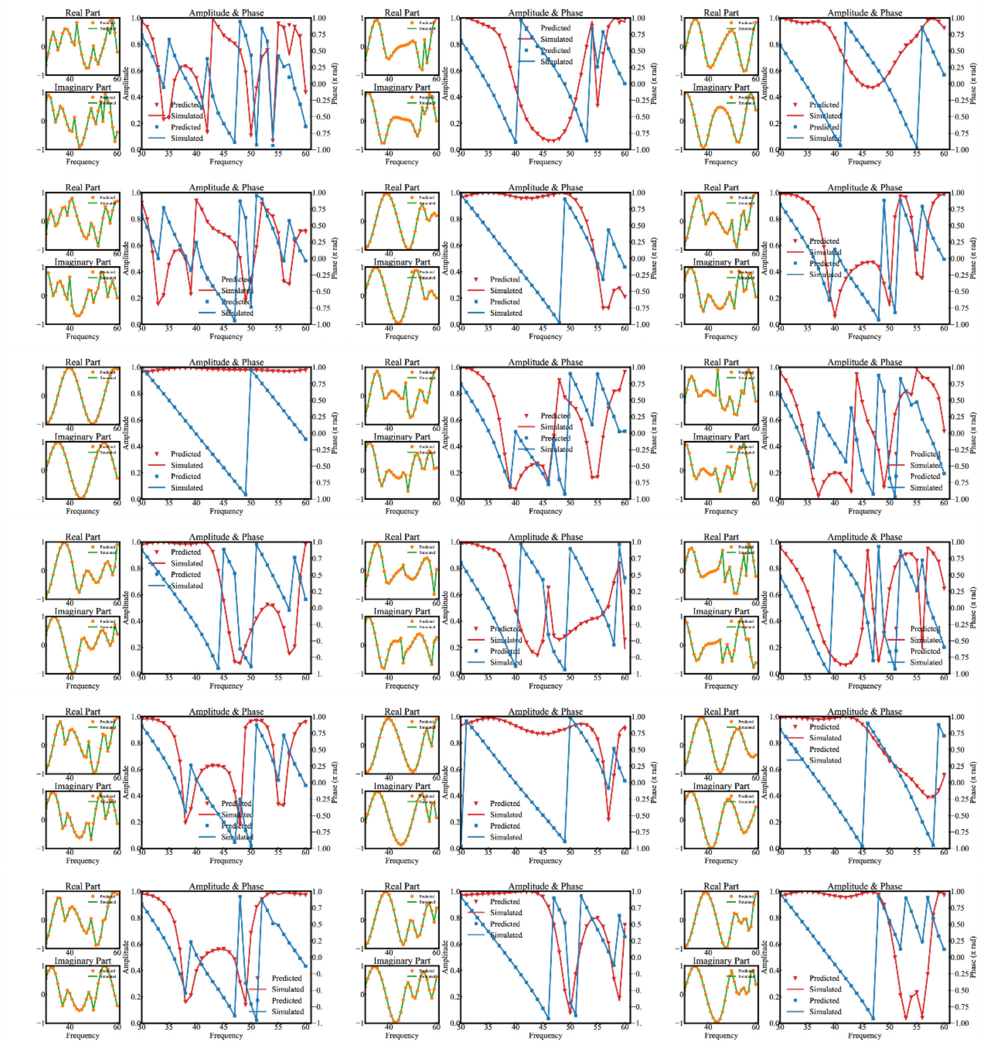

Figure S1. Examples demonstrating the CDNN performance.

1.4 Detailed information for the real-valued deep neural networks

In order to prove the superiority of the CDNN, we trained the real part and the imaginary part of the transmission coefficients as the control groups, respectively. The real deep neural network (RDNN) and the imaginary deep neural network (IDNN) have the same architecture and dataset as the CDNN, whereas all the building blocks, functions and parameters are based on real-valued operations and representations. Specifically, the forward predicting MSEs are as follows:

$$MSE(\mathrm{real}) = \frac{1}{m}\sum_{i=1}^{m}\left[\left(\mathrm{Re}(\widehat{\widetilde{\boldsymbol{T}}}) - \mathrm{Re}(\widetilde{\boldsymbol{T}})\right)^2\right], \tag{S4}$$

$$MSE(\mathrm{imag}) = \frac{1}{m}\sum_{i=1}^{m}\left[\left(\mathrm{Im}(\widehat{\widetilde{\boldsymbol{T}}}) - \mathrm{Im}(\widetilde{\boldsymbol{T}})\right)^2\right], \tag{S5}$$

As a result, the overall test errors are both ~7.5×10^{-5} after 50000 epochs as shown in **Figure S2**. And several test samples in the source dataset are randomly presented here. Obviously, according to the geometric interpretation provided by the complex plane for complex numbers, i.e. $|z|^2 = x^2 + y^2$, when $MSE(\mathrm{real}) + MSE(imag) = MSE$ in **eq 1**, the CDNN is competitive in performance with their real-valued counterparts. And if $MSE < MSE(\mathrm{real}) + MSE(imag)$, we can say that the CDNN has the superior performance compared with their real-valued counterparts.

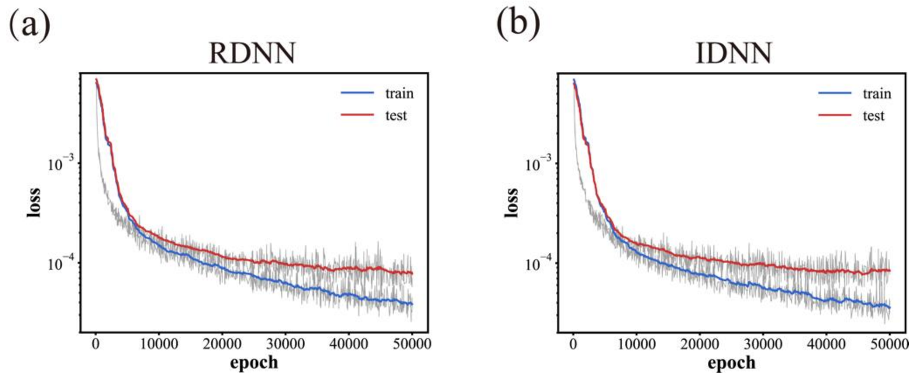

Figure S2. Learning curves of the two real-valued deep neural networks together as the control group of the CDNN. (a) and (b) are RDNN and IDNN that both take the loss value as the function of epoch. The smoothed train losses (blue curves) and test losses (red curves) are shown upon the original learning curves (light gray).

Supplementary 2. Training of the transfer model

The transfer model has the same construction as the source model but some different hyperparameters of a mini-batch size of 128 and a learning rate of 5×10^{-6}. Also, to alleviate the overfitting effect, L^2 regularization is adopted for weight parameters (bias parameters are not included) and the weight decay rate is set as 5×10^{-5}.

It should be clarified that the reason why the numbers of lines in three strategies are different in Fig. 4a is that some cases coincide with each other: 1) $k = 0$ in “frozen-none” and $k = 0$ in “hybrid-transfer” (i.e. direct learning); 2) $k = 10$ in “frozen-none” and $k = 0$ in “copy-all”; 3) $k = 10$ in “copy-all” and $k = 10$ in “hybrid-transfer”, respectively. The same cases are only draw once.

Supplementary 3. Details of the CAPSO-based hybrid inverse model

Particle swarm optimization (PSO) is a kind of evolutionary algorithm, designed by imitating the social foraging behavior of birds. And CAPSO is short for the conditioned adaptive PSO, adding the constraints and adaptive inertia weights which will be discussed as follows. The optimization starts from the random population within the search area. They look for the optimal solution by acting both independently and collaboratively. The fitness is used to evaluate the quality of the solution and guide each individual particle to update the position by adjusting its velocity through iterations. And the best personal position (*pbest*) and the best global position (*gbest*) update constantly until the end criterion is reached.

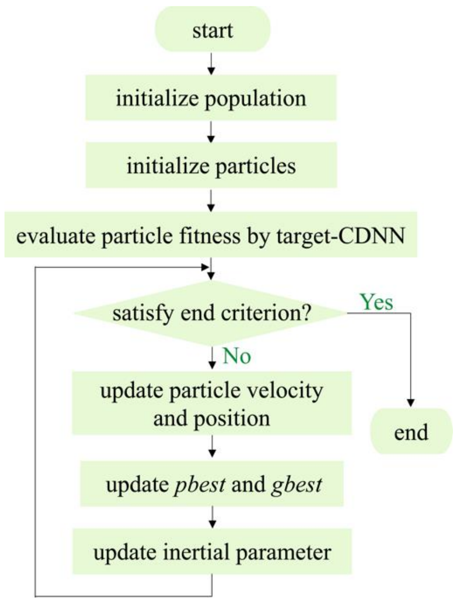
start
initialize population
initialize particles
evaluate particle fitness by target-CDNN
satisfy end criterion?
Yes
No
update particle velocity and position
end
update *pbest* and *gbest*
update inertial parameter

Figure S3. Flowchart of the conditioned adaptive particle swarm optimization (CAPSO) algorithm.

In our CAPSO, each particle is a four-dimensional tensor $\boldsymbol{x}_i = [\varepsilon_i, g_i, h_i, r_i]$, $1 \leq i \leq N$, where i is an integer, and $N = 200$ is the total number of particles in the population. Considering that the meta-atoms used to form a metasurface generally have a fixed lattice size, a restriction is added to the initialization of the population: $2r + g = P$,where $P = 150\ \mu m$ is the period of the target metalens. Also, to alleviate the coupling among adjacent unit cells, we set another constraint as $g > h/10$. The velocity tensor has the same dimension as the position tensor $\boldsymbol{x}_i$ used to update $\boldsymbol{x}_i$,which is $\boldsymbol{v}_i = [v_{i1}, v_{i2}, v_{i3}, v_{i4}]$, and each element is in the range of $[-v_{max}, v_{max}]$. The population is then input into the trained target-CDNN, which takes the place of CST software to calculate current EM coefficients. Then the fitness value is determined by the fitness function, which is defined as the loss as shown in **Eq 2** based on the target EM coefficients, to evaluate the quality of particles. The particle with the best position is the meta-atom with the dimension that has the most similar EM responses as the target. At this time, it is judged whether the end criterion is satisfied, that is, whether the maximum number of iterations is reached in our PSO algorithm. If yes, jump out of the loop and output the optimal parameter and the corresponding fitness value; if not, proceed to the next iteration, in which $\boldsymbol{v}_i$ and $\boldsymbol{x}_i$ of the particles are updated according to the following equations:

$$v_{ij}^{k+1} = wv_{ij}^{k} + c_1 rand_1\left(pbest_{ij}^{k} - x_{ij}^{k}\right) + c_2 rand_2\left(gbest_{ij}^{k} - x_{ij}^{k}\right), \tag{S6}$$

$$x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1}, \tag{S7}$$

where w is the inertia weight, $c_1 = 1.5$ and $c_2 = 2.5$ are cognitive and social rates, respectively. $rand1$ and $rand2$ are used to generate random numbers uniformly distributed between 0 and 1. k is the current iteration round. **Eq S6** and **S7** indicate that the velocity is actually the distance and direction that the particle will move in the next iteration. **Eq S6** includes three items:

1) inertia part. It is composed of the inertia weight and the particle's own velocity, which indicates that the next movement of the particle depends on its previous motion state. The non-linear dynamic inertia weight coefficient formula is used here to adaptively balance the global search and local improvement ability of the PSO algorithm as follows:

$$w_i^{k+1} = w_{min} + (w_{max} - w_{min})\frac{f_i^k - f_{min}^k}{f_{avg}^k - f_{min}^k}, \tag{S8}$$

where $w_{max} = 0.9$ and $w_{min} = 0.4$ are the maximum and minimum of w, f is the current fitness value of the particle, f_{min} and f_{avg} are the mean and minimum fitness value of all the particles, respectively.

2) Cognitive part. It represents the thinking of the particle itself, that is, the part of the particle's own experience, which can be understood as the distance and direction between the particle's current position and its own historical optimal position.

3) Social part. It represents the information sharing and cooperation between particles, which is derived from the experience of other qualified particles in the swarm and can be understood as the distance and direction between the current position of the particle and the historical best position of the swarm.

In our CAPSO, we assigned $c_2 > c_1$, which indicates that particles are more likely to move toward the best location of the swarm.

For each particle, its current fitness value is compared with the best position it had ever experienced. If the current is smaller, it will be taken as the best personal position. All the current values of *pbest* and *gbest* are compared, and *gbest* is updated.

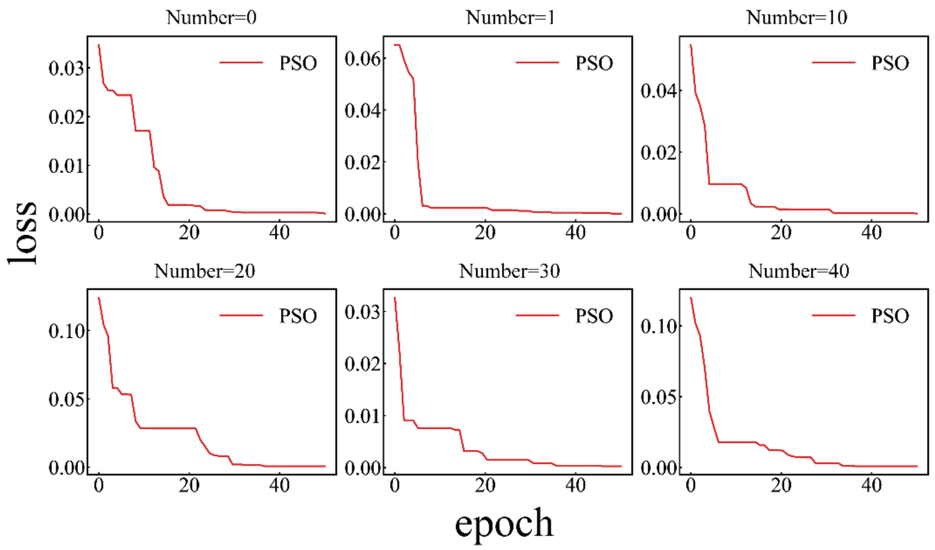

Figure S4. Convergence trajectories of CAPSO for six selected meta-atoms.

Supplementary 4. Detailed information for the designed metalens

Table S1 lists the details of the optimal 41 meta-atoms from the edge to the center of the metalens at each location derived by the inverse model, which are utilized to achieve a terahertz metalens working at 0.95 THz with a designed focal length of 20 mm.

Num ber	permittivity ϵ	radius $\boldsymbol{r}$	height $\boldsymbol{h}$	gap $\boldsymbol{g}$	phase (π rad)	amplitu de(a.u.)	fitness value
40	14.8958	73.4154	62.3973	38.2923	-0.4226	0.5025	0.000956
39	12.3959	42.1214	57.5118	53.9393	-0.6926	0.4975	0.000779
38	15.7753	27.9887	82.8054	61.0056	-0.9554	0.5005	0.000891
37	17.6039	27.5766	82.5549	61.2117	0.7866	0.5002	0.000069
36	14.4675	44.5239	59.7618	52.7381	0.5360	0.5005	0.000317
35	16.8846	50.8084	62.7127	49.5958	0.2914	0.5001	0.000045
34	15.3000	74.3422	88.0551	37.8289	0.0524	0.5025	0.001802
33	15.9980	63.4489	76.4050	43.2755	-0.1781	0.4998	0.000076
32	17.0048	24.5140	38.9071	62.7430	-0.4029	0.5000	0.000163
31	13.8215	28.5336	43.8894	60.7332	-0.6214	0.4983	0.000634
30	13.8738	41.7405	51.9259	54.1298	-0.8332	0.4999	0.000135
29	15.1697	29.6376	87.8139	60.1812	0.9615	0.4981	0.000620
28	15.1000	41.1182	50.4460	54.4409	0.7632	0.4824	0.005612
27	14.1599	43.5752	59.5675	53.2124	0.5708	0.5000	0.000156
26	15.7340	52.8102	83.3944	48.5949	0.3857	0.5000	0.000112

25	16.7908	67.6577	63.0376	41.1712	0.2073	0.4969	0.000936
24	16.3948	62.8581	65.0291	43.5710	0.0356	0.4989	0.000326
23	18.8266	69.2454	73.7300	40.3773	-0.1287	0.4985	0.000975
22	15.0330	82.8158	79.2444	33.5921	-0.2874	0.5000	0.000073
21	12.8982	70.3403	74.4250	39.8298	-0.4382	0.5006	0.000578
20	11.4534	52.6684	67.1337	48.6658	-0.5830	0.5001	0.000149
19	11.4428	37.4814	60.8193	56.2593	-0.7204	0.4998	0.000110
18	16.6226	27.2567	77.3806	61.3716	-0.8511	0.4993	0.000344
17	13.3719	35.9049	53.1872	57.0476	-0.9746	0.4992	0.000237
16	16.4804	29.0597	84.6219	60.4701	0.9087	0.5093	0.002807
15	18.0515	26.3955	80.6526	61.8023	0.7990	0.4983	0.000544
14	16.1279	42.6820	48.2692	53.6590	0.6963	0.4988	0.000358
13	18.3060	46.5453	44.4834	51.7274	0.6007	0.5018	0.000547
12	13.3695	37.0506	62.0445	56.4747	0.5120	0.5013	0.000467
11	15.8661	53.9055	88.5541	48.0473	0.4304	0.5003	0.000123
10	14.9480	32.8033	50.8088	58.5983	0.3556	0.5011	0.000480
9	17.3103	35.4797	90.8764	57.2602	0.2883	0.5000	0.000087
8	18.0049	57.1266	70.6637	46.4367	0.2277	0.4988	0.000439
7	16.9420	70.4929	65.5674	39.7535	0.1745	0.5007	0.000249
6	16.0321	71.6491	72.6703	39.1754	0.1290	0.4977	0.001462
5	15.8621	64.9650	67.6018	42.5175	0.0890	0.4998	0.000058
4	16.3302	62.7078	64.7292	43.6461	0.0568	0.4971	0.001043
3	15.1655	62.4863	71.2675	43.7568	0.0321	0.5010	0.000324
2	16.9134	47.7333	75.7231	51.1334	0.0142	0.5015	0.000445
1	14.8374	71.7000	89.3223	39.1500	0.0035	0.4996	0.000123
0	14.4581	65.3860	80.7976	42.3070	0.0001	0.5005	0.000246

Table S1. Geometrical parameters and corresponding electromagnetic responses and fitness values of the optimal meta-atoms used to build the proposed metalens.

Supplementary 5. Effect of amplitude weight factor on the performance of the metalens

To verify the effect of the amplitude weight factor on the performance of the metasurface, we set different values of η in **eq 2** and conducted the inverse model on the source dataset with other conditions remaining the same. The metalens is designed to focus the incident light to $z = 200$ μm with a diameter of 162 μm and a lattice size of 2 μm. As shown in **Figure S5**, different amplitude weight factors lead to variations in the peak intensity of the focus and the shift of the focal length, where the case that amplitude is not considered corresponds to the lowest peak intensity and the largest focal shift. The results indicate that the amplitude also plays a role in the phase-gradient metasurface, which further declares the significance of CDNN that can predict the complex transmission coefficients containing the comprehensive information of amplitude and phase.

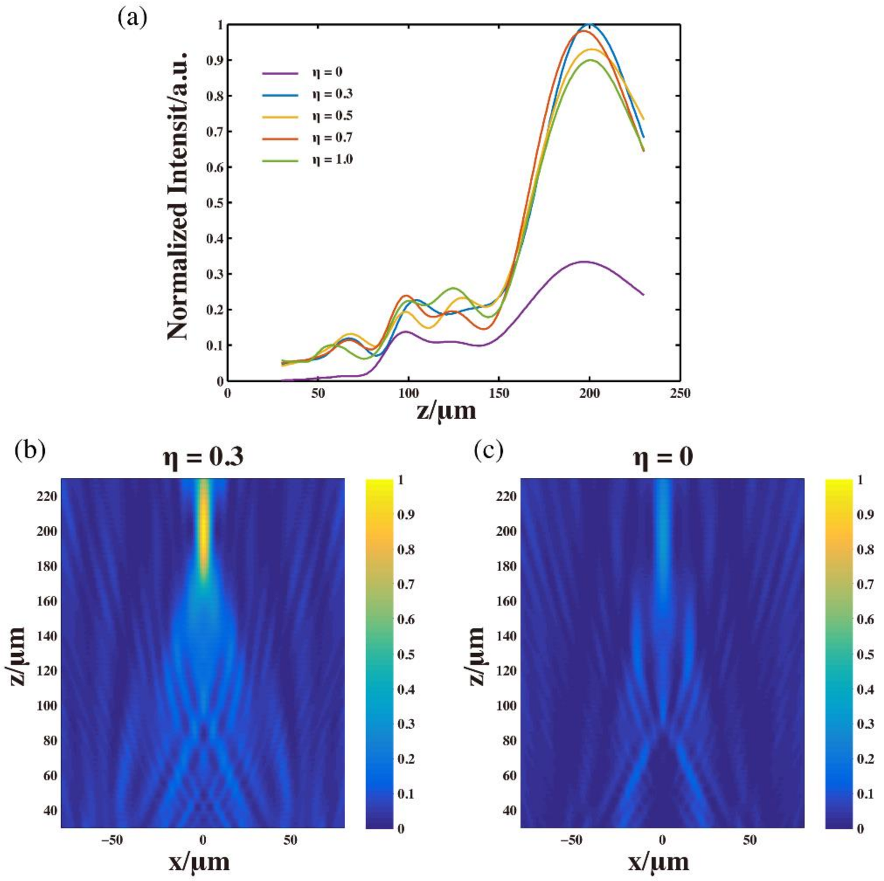

Figure S5. Characterization of the metalens designed by the hybrid inverse model in the source frequency band. (a) Normalized intensity profiles along the propagation direction at $x = 0$. Different colors represent different values of the weight factor η as the legend shows. (b) and (c) are normalized intensity distributions of the designed metalens along the propagation plane with $\eta = 0.3$ and $\eta = 0$, respectively.