Compressed Meta-Optical Encoder for Image Classification: Supplementary Information

Anna Wirth-Singh[a,1,*], **Jinlin Xiang**[b,*], **Minho Choi**[b,*], **Johannes E. Fröch**[a,b], **Luocheng Huang**[b], **Shane Colburn**[b], **Eli Shlizerman**[c,b], **Arka Majumdar**[a,b,2]

[a]Department of Physics, University of Washington, Seattle, WA 98195, USA
[b]Department of Electrical and Computer Engineering, University of Washington, Seattle, WA 98195, USA
[c]Department of Applied Mathematics, University of Washington, Seattle, WA 98195, USA
[*]These authors contributed equally to this work.

S1 Electronic Training Details: Dataset, Networks and Implementation Details

Dataset: The MNIST dataset includes $70,000$ grayscale images of handwritten digits, ranging from 0 to 9. Each image (28×28 pixels) represents a single digit. MNIST includes two sets: a training set, which includes $60,000$ images, and a test set, encompassing the remaining $10,000$ images.

Network Selection: For network selection, the MNIST dataset typically does not require a complex CNN or multiple layers. Here, we implemented a compressed hybrid network, primarily due to concerns about underfitting in small networks and the effects of noise and misalignments inherent in multiple-layer optical systems. Specifically, each optical convolutional layer introduces noise, and after several layers, this could culminate in failures in multi-layer systems. We conducted a practical experiment to simulate the performance of single and five convolutional layers by adding Gaussian noises ($\mathcal{N}(0,1)$) to each kernel. In Figure S1, we observed that such noise significantly diminishes the performance of 5 convolutional layers, notably reducing accuracy to 64% for the MNIST dataset and 32% for ImageNet, respectively. Therefore, compressing multi-layer CNNs into a single layer emerges as the proper strategy for optical implementation.

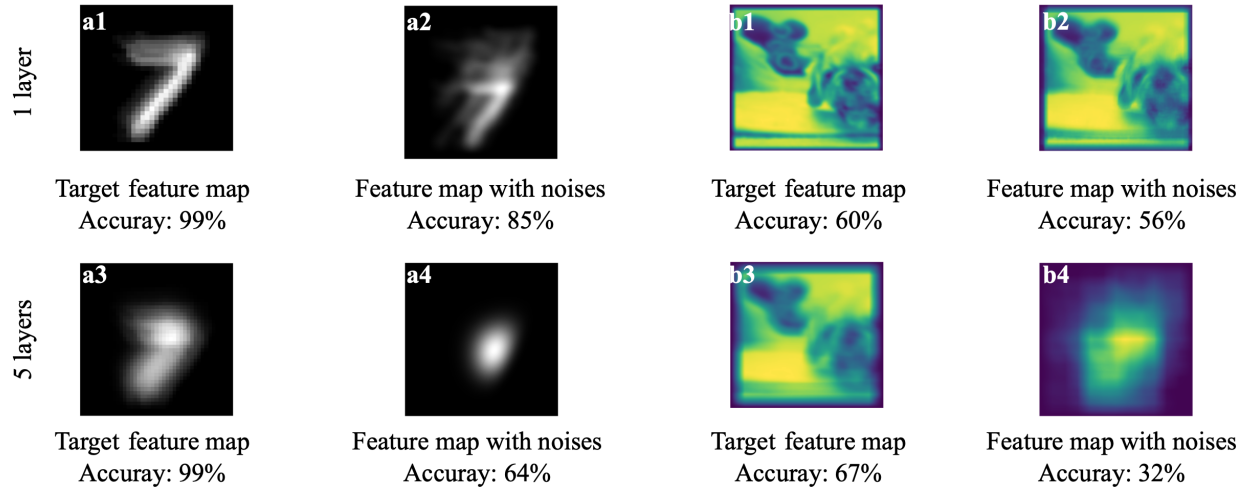

Fig S1 Network selection experiments: a1-a4 are the MNIST dataset, and b1-b4 are the ImageNet dataset.

The Table S1 compares three CNN architectures: Modified AlexNet, LeNet, a Compressed CNN, and a Hybrid Optical-Electronic CNN. We employ a modified version of the AlexNet as the teacher

network. This network is composed of five convolutional layers, three fully connected layers, and several nonlinear activation layers. The input for the AlexNet is determined by the MNIST dataset, which is 28 by 28 pixels. LeNet is designed for MNIST with smaller parameter size compared to other CNN models, such as AlexNet based models. In contrast, the student network is purely linear, containing only two layers: one convolutional layer and one fully connected layer. The dimensions of each kernel are 6 by 6. We also set the kernel size of the compressed model as 6 by 6, to align with the kernel size used in AlexNet. The Hybrid CNN combines optical and electronic components, maintaining a single convolutional layer and introducing a calibration layer before the final fully connected layer. The convolution channel is 16, including 8 positive kernels and 8 negative kernels. We merge positive and negative kernels and then proceed to the pooling layer. While LeNet is smaller, this approach may limit our contributions. Since LeNet is small and customized for the MNIST dataset, it is not feasible to transfer the compressed model to more complex datasets, such as CIFAR or ImageNet. However, if our method can compress AlexNet, it will provide a solid baseline for further work on CIFAR or ImageNet datasets. Therefore, we start with a modified AlexNet.

Modified AlexNet		**LeNet**		**Compressed CNN**		**Hybrid Optical-Electronic CNN**	
Layer	Kernel	Layer	Kernel	Layer	Kernel		
Convolution1	1×64×6×6	Convolution1	1×6×5×5	Convolution	1×8×6×6	Convolution	1×16×6×6
Convolution2	64×192×5×5	Convolution2	6×16×5×5				
Convolution3	192×256×3×3						
Convolution4	256×384×3×3						
Convolution5	384×8×3×3						
Average Pooling	[8,6,6]	Average Pooling	[16,6,6]	Average Pooling	[8,6,6]	Average Pooling	[8,6,6]
Layer	Weights	Layer	Weights	Layer	Weights	Layer	Weights
FC1	288 → 1024	FC1	576 → 120	FC	288 → 10	Calibration	288 → 288
FC2	1024 → 256	FC2	120 → 84			FC	288 → 10
FC3	256 → 10	FC3	84 → 10				

Table S1 Architecture of the network backbone

Kernel Selection: Previous work has indicated that the size of the convolutional kernel affects the performance of convolutional neural networks (CNNs) in image classification challenges [28]. Experimental results show that CNN architectures with convolutional kernels around 6×6 tend to be more successful compared to other settings, including 3×3, 5×5, 7×7, and 9×9. However, the difference in performance is minor in the MNIST dataset, within a 1% margin. Therefore, we select a 6×6 kernel size to capture more detailed features from the teacher network and avoid using larger kernel sizes in AlexNet, e.g., 11×11. This is because larger kernel sizes, which correspond to more complex kernels, pose more challenges for meta-optical implementation.

Complexity and Energy Analysis: We employ Multiply-and-Accumulate (MAC) operations as a metric to analyze the computational complexity of the three different models presented. There are two types of layers where MAC operations are pivotal: convolution layers and fully connected layers. The following formula is utilized to calculate the number of MAC operations in a convolutional layer:

$$\mathit{MAC}_{convolution} = N^2 \times k^2 \times \#\mathit{kernels} \times C_{in}, \tag{1}$$

where N^2 is the dimension of the input, K^2 is the dimension of the kernel, $\#\mathit{kernels}$ is the number of kernels and C_{in} is input channel. This formula assumes that the stride is 1 and uses the same

size padding. For fully connected layers, the computation is more straightforward. The number of MAC operations is determined by the product of the number of input and output nodes:

$$MAC_{FC} = m \times n, \tag{2}$$

where m corresponds to the size of the input vector, and n is the size of the output vector. We don't count the pooling or resize into total MAC calculation since they are less computationally intensive than convolution operations [41].

	AlexNet	**Compressed CNN**	**Hybrid Optical-Electronic CNN**
Convolution1	64×6×6×28×28	8×6×6×28×28	0
Convolution2	64×192×5×5×3×3		
Convolution3	192×256×3×3×3×3		
Convolution4	256×384×3×3×3×3		
Convolution5	384×8×3×3×3×3		
FC1	288 ×1024	288 × 10	288 × 288
FC2	1024 × 256		288 × 10
FC3	256 × 10		
Sum	$17,323,520$ (100%)	$228,672$ (1.32%)	$85,824$ (0.49%)

Table S2 Multiply-and-accumulate (MAC) operations in network

We summarize the MAC consumption for each model in Table S2, which includes each layer in the AlexNet, Compressed CNN, and Hybrid Optical-Electronic CNN architectures. For AlexNet, the convolutional layers require a substantial number of operations. The first layer alone involves applying 64 kernels of size 6×6 to a 28×28 input feature map. As the layers progress, the number of kernels increases, reaching 256 at the last convolutional layer, accounting for 17 million operations at 8-bit precision. In a modern digital system, one MAC operation consumes approximately 1pJ, making the total energy consumption about 17μJ [19]. The Compressed CNN significantly reduces the number of MAC operations, employing only 8 kernels of size 6x6, which reflects a 98.7% reduction in computational requirements compared to AlexNet. This streamlining is also evident in the fully connected layers, where the Compressed CNN requires operations only between 288 and 10 nodes, a contrast to the thousands of operations in AlexNet's layers. Finally, the total total energy consumption is about $228n$J.
However, the Hybrid Optical-Electronic CNN stands out, requiring 0 MAC operations in its convolutional layer, due to processing images with the optical component. Its fully connected layers include one calibration layer and then mirror the simplified structure of the Compressed CNN. This configuration results in a total energy consumption of about $85nJ$, which not only represents a 98.7% reduction from AlexNet but also a 62.5% decrease compared to the Compressed CNN.

Training Details: During the knowledge distillation training process, we utilize knowledge distillation loss (Equation 3) to optimize the student networks. We choose stochastic gradient descent (SGD) with an initial learning rate of 0.001 for 80 epochs. This learning rate is reduced by a factor of 10 after every 20 epochs. For calibration training, we randomly select 6000 images, which represent only 10% of the complete dataset. We employ a Mean Squared Error (MSE) loss function to calibrate the distance between the outputs of the convolutional layers from the teacher networks and those of the student network with an initial learning rate of 0.001 for 40 epochs. We implement

existing networks based on the publicly available official code and train all models on two 2080Ti GPUs.

S2 Explanation of PSF and Convolution

The point spread function (PSF) of a lens-like imaging system describes how the system focuses point sources. Simply, for an input point source, $PSF(x, y)$ is the amplitude of the electric field measured in the image plane. A perfect imaging system focuses light from a point source back to a point, mathematically represented by a delta function $\delta(x, y)$. More generally, for realistic lenses which have a PSF of finite size, we can denote $PSF(x, y)$ to describe the point spread function. Further, the PSF may be spatially-varying; that is, a point source located at (x_1, y_1) may produce $PSF_1(x_1, y_1)$ which is different (not simply translated) from $PSF_2(x_2, y_2)$ which is a measurement of a different point source located at (x_2, y_2). However, here we assume that a spatially invariant PSF is sufficient to describe our optical system and therefore are concerned with only a single $PSF(x, y)$ for each optic.
In Fig. S2a, we illustrate an input point source which produces an arbitrary example PSF in Fig. S2b. In a realistic lens system, the PSF is broadened from a delta function and may or may not be symmetric. For a more complex input object (for example, a two-dimensional image rather than a single point source), we can represent the light intensity in the object plane as an array of point sources denoted $O(x, y)$, illustrated in Fig. S2c. Then, the intensity in the image plane $I(x, y)$ produced by the optic is

$$I(x_0, y_0) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} O(x, y) PSF(x_0 - x, y_0 - y)\, dxdy \tag{3}$$

For the case of a perfect lens $PSF(x, y) = \delta(x, y)$, we retrieve the expected output

$$I(x_0, y_0) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} O(x, y)\delta(x_0 - x, y_0 - y)\, dxdy = O(x_0, y_0) \tag{4}$$

The convolution of continuous one-dimensional functions $f(x)$ and $g(x)$, denoted $f * g$, is defined by [42]

$$(f * g)(x_0) = \int_{-\infty}^{\infty} g(x) f(x_0 - x)\, dx \tag{5}$$

and is straightforwardly extended to two-dimensional continuous functions as

$$(f * g)(x_0, y_0) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} g(x, y) f(x_0 - x, y_0 - y)\, dxdy \tag{6}$$

Therefore, noting the similarities between Eqns. 3 and 6, we observe $I(x_0, y_0) = (O*PSF)(x_0, y_0)$. That is, the image produced by an optic is the input object convolved with the lens PSF.
Analogously, the two-dimensional discrete convolution is defined

$$(f * g)[n, m] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} f[i, j] \cdot g[n - i, m - j] \tag{7}$$

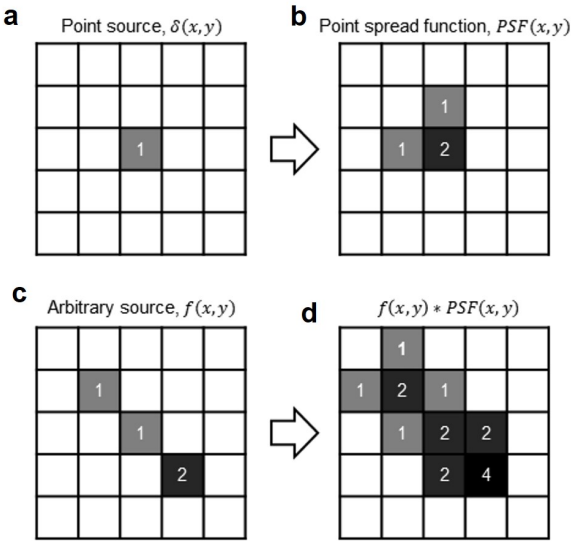

Fig S2 PSF and Convolution illustrated with discrete matrices. (a) Illustrates a point source. (b) Represents an arbitrary PSF which may be obtained by imaging (a) through a realistic lens system. (c) An arbitrary source, or input image, which is an array of point sources. (d) The expected image produced by imaging the arbitrary source (c) through the lens system with PSF shown in (b).

And therefore we have, for discrete input $O[n,m]$ and discrete point spread function $PSF[n,m]$,

$$I[n,m] = \sum_{j=-\infty}^{\infty} \sum_{i=-\infty}^{\infty} O[i,j] \cdot PSF[n-i, m-j] \tag{8}$$

Figure S2d demonstrates a discrete convolution between an arbitrary input source (Fig. S2c) and optic PSF (S2b).
In a convolutional neural network (CNN), the discrete convolution operation is used to extract features from an input image. The input is convolved with optimized kernels (or "filters") to produce an output with reduced dimensionality that is fed to the next layer in the network. In this work, we perform this convolution optically, with the (discrete) kernel analogous to our (continuous) optic PSF.

S3 Meta-optic Specifications

In designing the optics, there are a wide range freedoms to choose parameters such as operating wavelength, optics size, desired PSF size, focal length, and meta-optic scatterer. We considered these choices carefully explain our rationale here, but many other possible configurations may also yield viable results.
The hybrid CNN can be adapted to operate at any wavelength; in this case, we chose 525 nm to best align with the available light sources and camera sensitivity. The camera used in experiment is Allied Vision Prosilia GT1930C with 5.86 μm per pixel resolution. Based on these factors, we set the simulation grid size to 586 nm such that 10 simulation pixels = 1 camera pixel, and the simulation grid size is comparable to the operating wavelength. For phase mask propagation, deeply subwavelength pixel size is unnecessary and would increase the computational load of designing

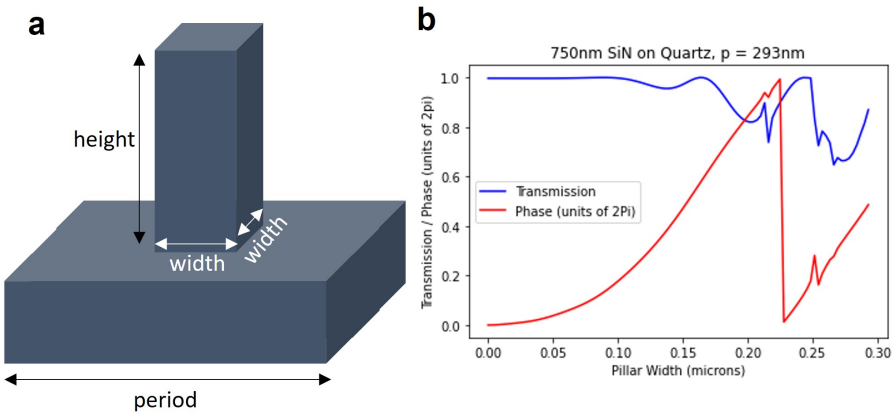

Fig S3 Scatterer response. (a) Diagram of the scatterer unit cell. The unit cell consists of a rectangular pillar of fixed height and variable width, sitting on a lattice of fixed period. (b) The simulated unit cell phase and transmission as the pillar width is changed, for fixed wavelength of 525 nm. A subset of unit cells with widths ranging from 80 nm to 225 nm were chosen to provide 0 to 2π phase coverage (red) which high transmission (blue).

the phase masks. The meta-optic unit cells, however, are situated on a deeply subwavelength lattice size of 293 nm (exactly 1/2 simulation pixel size).

The optimized phase are physically realized as arrays of sub-wavelength scatterers. To be evenly divisible by the simulation grid size, we chose scatterer periodicity of 293 nm. The chosen scatterers are 750 nm tall square SiN pillars on quartz substrate, with pillar widths ranging from 80 nm to 225 nm, as illustrated in Figure S3(a). The quartz substrate functions primarily as structural support for the pillars and is transparent at the wavelength of interest. We calculate the phase and transmission response of these scatterers using rigorous coupled wave analysis (RCWA), specifically the S4 implementation [40], as shown in Figure S3(b). By adjusting the scatterer width, phase shifts covering a 0 to 2π range can be achieved with high transmission. This phase-pillar width response was then used to map the optimized phase masks to physical geometries. In this case, each phase mask pixel (586 nm) corresponds to a 2×2 block of scatterers. The individual pillars are shown in high-resolution SEM images of the fabricated devices in Fig. S4).

Each optic was designed to produce an image of a particular convolutional kernel as its PSF. As described in the main text, each electronic convolutional kernel is a 6×6 matrix which contains both positive and negative values. Since negative values cannot be represented when measuring the amplitude of the electromagnetic field on the camera sensor, we separated the kernels into two matrices, each also 6×6, one containing only the positive values and the other containing the (absolute value of) negative values. We present these electronic kernels, separated into positive and negative parts, in Fig. S5(a). The desired PSF was defined by shrinking one of these kernel matrices to the desired size and padding the remaining space with zeroes. So that the PSF remains point-like, it is desirable to shrink the PSF image as small as possible (down to a minimum resolution where one PSF "pixel" = one camera pixel). However, such a design would not be robust; to guard against misalignments, we defined the desired PSF to be such that each PSF "pixel" would correspond to 2x2 camera pixels and therefore the desired PSF image is 70.32 μm $\times$ 70.32 μm in physical size.

Finally, we chose the size of each kernel optic to be 468.8 μm (800 simulation pixels). A larger optic allows for more light collection and therefore higher SNR, but also undesirably increases the

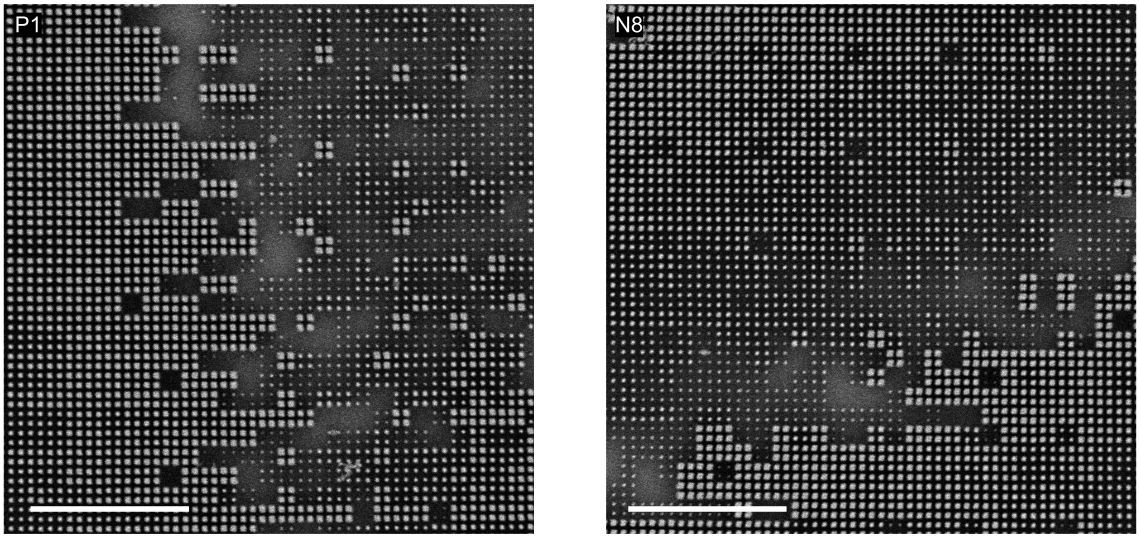

Fig S4 High-resolution SEM images of the scatters from the meta-optics. (a) Positive-1 meta-optics. (b) Negative-8 meta-optics. Scale bar: 5 μm.

total optic footprint, so the chosen size reflects a tradeoff between these two factors. The focal length was chosen to be 2.4 mm based on our ability to place the optics as near the camera sensor as possible while allowing room to adjust the focal length and alignment to obtain the best image.

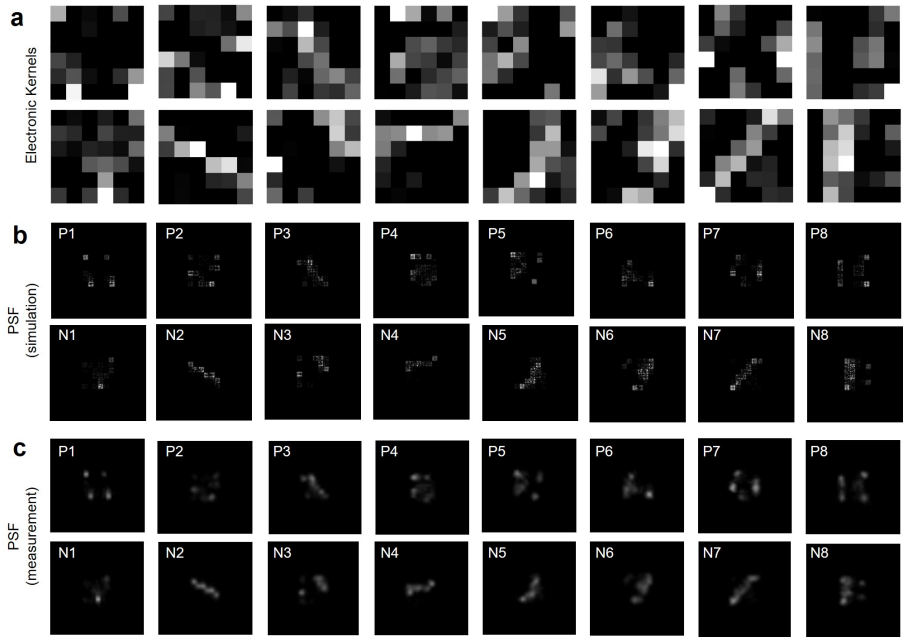

Fig S5 The convolutional kernels and corresponding PSFs.(a) The positive (top) and negative (bottom) parts of all eight convolutional kernels. Each kernel is a 6 by 6 matrix. (b) The simulated PSF results at imaging distance 2.4 mm for the positive (top) and negative (bottom) sub-optics. (c) The experimentally measured PSF.

S4 Phase Mask Optimization

Once the desired PSF is defined, we aim to determine the phase required to produce it, modeling the meta-optic as a phase mask 2.4 mm in front of the desired plane. This is effectively a holography problem; both the input light and the desired PSF are specified by electric field amplitude only, and the complex phase in a particular intermediate plane must be determined to map between the two. This problem is solvable by the well-known Gerchberg-Saxton (GS) algorithm [30]. Here, we provide the details of the implementation. All simulations were done at a single wavelength (525 nm) and on a lattice grid of 586 nm.

The designed phase masks corresponding to the positive and negative parts of the eight convolutional kernels are shown in Fig. S6(a). To implement the phase retrieval, we initialize the phase mask guess ϕ as constant 0 phase and the input light U as a plane wave (constant amplitude 1). Therefore, the initial field after it passes through the optic is $U_O = Ue^{i\phi}$. Our implementation of the iterative steps is based on that in [43,44]:

1. Propagate the initial field U_O to the image plane to obtain U_I.
2. Calculate $U_I' = |PSF|e^{i*\Theta(U_I)}$, where $|PSF|$ is the amplitude of the desired point spread function and $\Theta(U_I)$ is the phase of the complex field U_I.
3. Backward propagate U_I' to the object plane to obtain U_O'.
4. Calculate $U_O'' = |U_O|e^{i*\Theta(U_O')}$, where $|U_O|$ is the amplitude of the field in the object plane; here, this is a plane wave.

The process is iterated by inputting U_O'' to U_O, ultimately retrieving the final phase $\phi_{final} = \Theta(U_O'')$. This process is quick to converge, requiring only 20 iterations to retrieve the desired phase to our satisfaction. In this process, the propagation function is Fourier-based band-limited angular spectrum propagation [32].

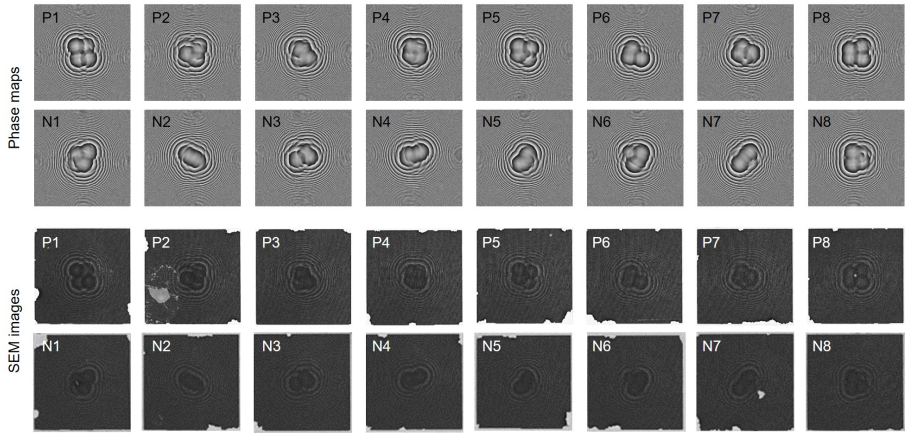

Fig S6 The designed meta-optics. (a) Optimized phase maps for each sub-optic. The phase values vary between 0 and 2π and each optic is 470 μm$\times$ 470 μm in size. (b) SEM images of all designed optics.

Additionally, we developed our own phase mask optimization code in TensorFlow and fabricated a second set of optics corresponding to these phase masks. The experimental PSFs and an example convolution from both sets of optics are shown in Fig. S8(b) and (c). While both optics produce the

desired PSFs and perform the desired convolution, the optics designed using the GS method exhibit brighter and slightly clearer images on the camera. Therefore, we present only the GS-optimized results in the main text and present the TensorFlow results here as supplementary information.
For the TensorFlow based method, the Adam optimizer was used to optimize the phase mask. We model the input light U as a plane wave and again use the angular spectrum method to propagate the field to the desired plane. To facilitate faster convergence, we initialized the phase mask guess ϕ as the phase of the backward propagated PSF. Then, we iteratively updated the phase mask through the following steps:

1. Calculate the initial field $U_O = Ue^{i\phi}$, where U is the input plane wave and ϕ is the phase mask guess.

2. Propagate the initial field U_O to the image plane to obtain U_I.

3. Calculate the loss between the simulated image intensity $|U_I|$ and the desired field intensity $|PSF|$. To calculate the loss, we first L2 normalize both $|U_I|$ and $|PSF|$. We then calculate the difference between these two quantities, take the absolute value, and sum up the values in the difference matrix.

4. Update the phase mask.

This optimization continued for 500 iterations at a learning rate of 0.05. The phase masks designed using the TensorFlow method are qualitatively similar to those designed using the GS method in that they both exhibit lobe-like center structures and lens-like outer rings, but the TensorFlow optics exhibit slightly more randomness. We hypothesize that the TensorFlow-based optics exhibit more destructive interference than the GS-based optics, which is supported by the observation that the TensorFlow-based optics produce dimmer images.

S5 Experiment Setup

In our case, multiple convolutional meta-optics (16 in total; 8 for positive and 8 for negative) are placed on a single wafer, while the light is propagating along a perpendicular direction, utilizing all three-dimensional space and making the experiment setup compact and robust. Figure S7 represents the experimental setup for both PSF and convoluted image measurement from the meta-optics. We considered the light out of single-mode fiber as a point source as the mode size of the single-mode fiber is about 4μm, which is far smaller than the image size on the micro-display, 8mm. Therefore, by measuring the images from single-mode fiber light source passing through the meta-optics, we can achieve the PSFs of the meta-optics on the camera. And, by replacing the single-mode fiber light source to the micro-display, we can achieve the convoluted images of the display on the camera.
Here, the distance between the light source, i.e. single-mode fiber or the micro-display, and the meta-optics is about 90mm, and the distance between the meta-optics and camera is about 2.4mm. Each of 16 convolutional meta-optics has a size of 468.8μm, and the spacing between the meta-optics is $234.4\mu m$, which results in the total size of the whole 16 meta-optics of a 2 row as $5.4 \times 1.2mm^2$. As the camera (GT-1930C) that we captured both PSFs and convoluted images has a much larger sensor than the array of 16 meta-optics, we can simultaneously measure either PSFs and convoluted images from all meta-optics all together as shown in Figure S8. Thanks to the

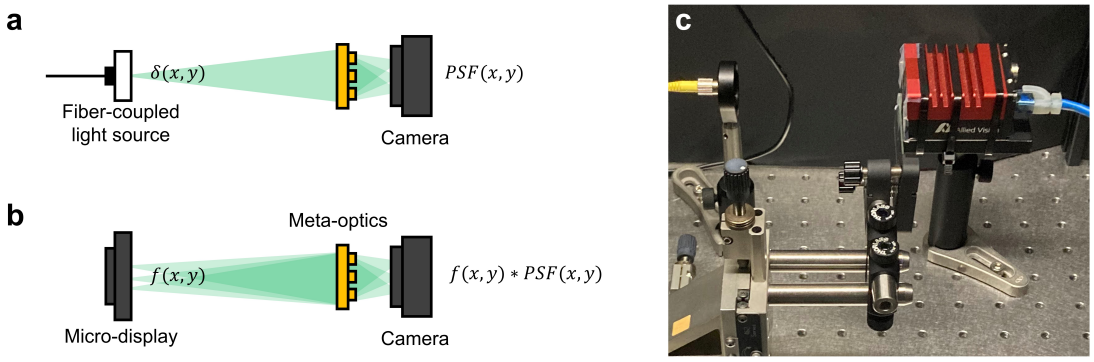

Fig S7 Experiment setup. (a) Schematics of measuring PSFs from the meta-optics. (b) Schematics of measuring convoluted images from the meta-optics. (c) Picture of the experimental setup for measuring PSFs.

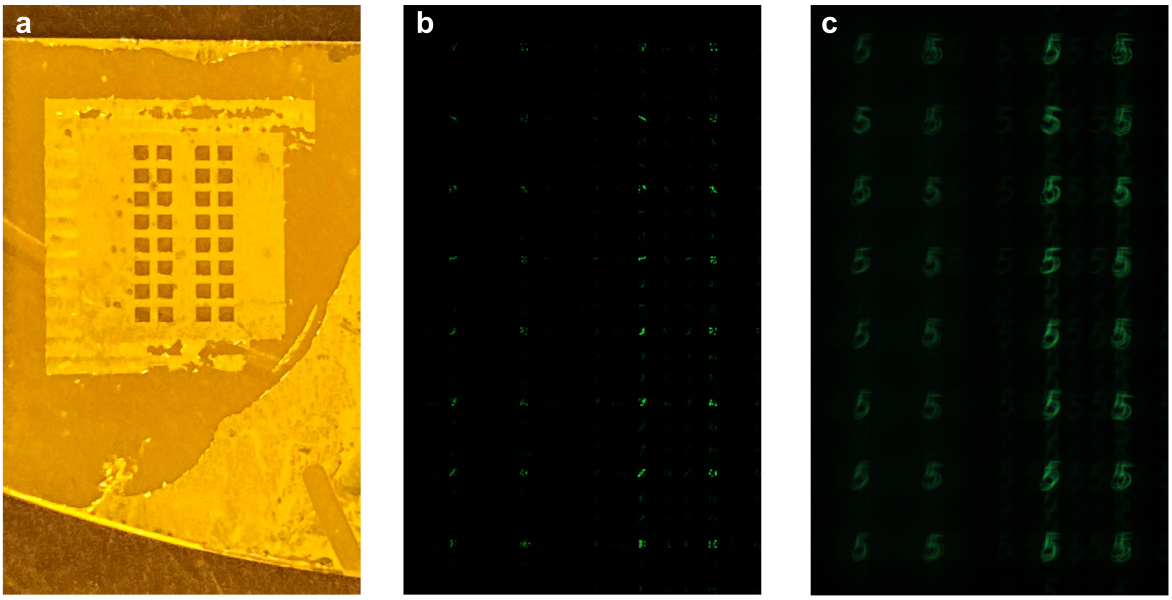

Fig S8 Multiple convolutional meta-optics on a single wafer. (a) Picture of the fabricated chip with a metallic aperture. (b) Measured PSFs of the meta-optics in a single shot. (c) Measured convoluted images from one of the MNIST datasets in a single shot. For both PSFs and convoluted images, 16 PSFs on the left are from the meta-optics designed by Tensorflow, and the other 16 PSFs on the right are from the meta-optics designed by the GS-algorithm, which were used with the computational backend for the classification experiment.

parallel measurements of all meta-optics, we can extremely reduce the total time for saving the convoluted images of all 10,000 MNIST datasets which are simply limited by the exposure time for each capture, which was 0.5 sec for this work. A Python script was used to automate the experiment.

The optics work under coherent and incoherent illumination. For PSF measurements, we used a laser source for the clearest results. However, for the classification experiment, we used incoherent illumination. The fact that this approach works with incoherent illumination is a great benefit, since the system could be applied to classification real-world imaging without the need for specialized

optics.

S6 Extended Optical Experiment Results

We present further comparison between the convoluted images obtained in the experiment and the electronic convolution Fig. S9. For a particular convolutional kernel (number 4), we illustrate the breakdown into positive and negative parts, as well as the net kernel (positive minus negative) in Fig. S9a. In Fig. S9b, we illustrate the optical experiment results obtained with the optic corresponding to this particular kernel. The negative image is computationally subtracted from the positive image to obtain the net result, shown at the right. Further, in Fig. S9c, we compare the net electronic convolution (top row) to the net optical experiment convolution (bottom row). The results illustrate the effectiveness of the optical convolution, albeit with slight noise and differences which are accounted for by the calibration function. To quantitatively assess these differences, we calculated the average cosine similarity between the electronic convolution outputs and optical experiment outputs to be 0.78 (a cosine similarity of 1 indicates that the two are identical) based on Equation 9. After the calibration layer, the average cosine similarity between the calibrated optical outputs and electronic convolution outputs is 0.96.

$$\textit{cosine similarity}\ (A, B) = \frac{AB}{\|A\|\|B\|}, \tag{9}$$

Where our input are two grayscale images, we reshape them into vectors A and B, and then calculate the cosine similarity.

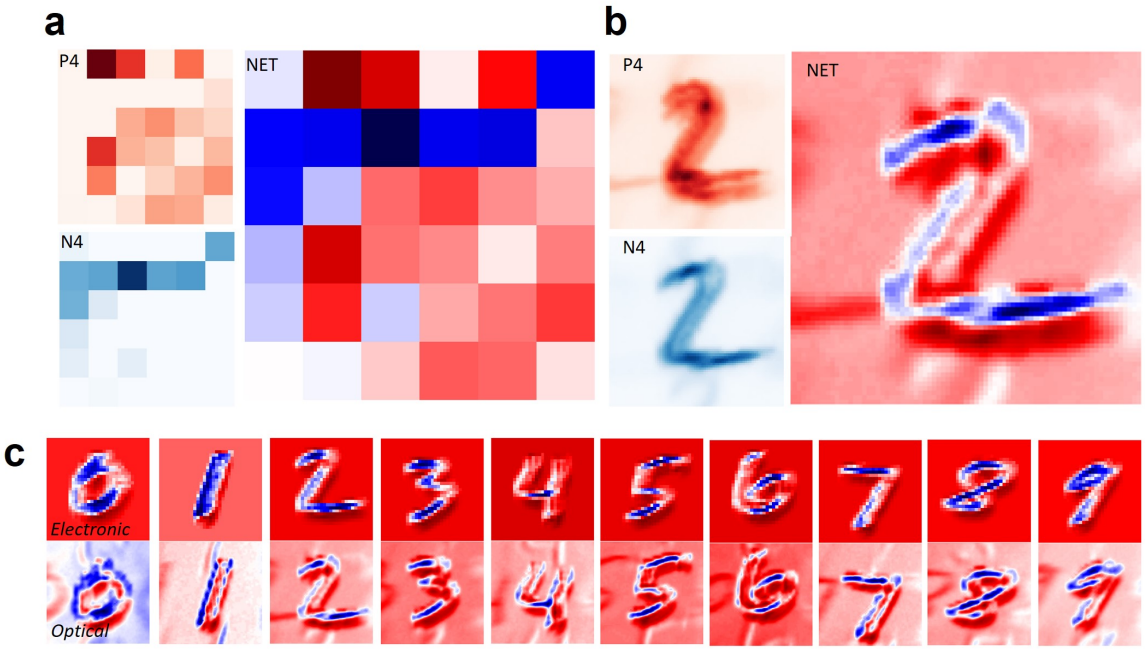

Fig S9 A sample of electronic and optical convolution results. (a) As an example, we plot convolutional kernel number 4, broken up into positive and negative parts as well as the net kernel with both positive and negative values. (b) The corresponding optical experiment convolution results for this kernel's positive (top) and negative (bottom) optics, as well as the computationally combined (right) net convolution. (c) A sample of MNIST digits convolved electronically (top) versus optically (bottom).