Supplementary Information for

A Novel High-Speed Optical Computing Platform with Dynamic *In-situ* Reconfigurability

Yuanjia Wang[1], Xin Dong[1], Pujing Cheng[1,3], Yi Zhou[1,2,*], and Kenneth K. Y. Wong[1,4,*]

[1]*Department of Electrical and Electronic Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong, China*

[2]*College of Advanced Interdisciplinary Studies, National University of Defense Technology, Changsha, China*

[3]*Department of Electronic and Electrical Engineering, Southern University of Science and Technology, Shenzhen, China*

[4]*Advanced Biomedical Instrumentation Centre, Hong Kong Science Park, Shatin, New Territories, Hong Kong, China*

[*] *Corresponding author*

Email: yizhou1508@163.com, kywong@eee.hku.hk

S.1. Detailed HS-BONN diffractive layer connectivity expression

The mathematical descriptions of each 2D-FCNN diffractive layer with DMD modulations are illustrated here. For layer l, the input is a $M \times N$ sampled light field $\mathbf{U}^{l}$, and each input pixel in layer l is denoted by $u_{i,j}^{(l)}$:

$$\mathbf{U}^{(l)} = \begin{bmatrix} u_{1,1}^{(l)} \dots u_{1,j}^{(l)} \dots u_{1,N}^{(l)} \\ \vdots \ \ddots \ \vdots \ \ddots \ \vdots \\ u_{i,1}^{(l)} \cdots u_{i,j}^{(l)} \cdots u_{i,N}^{(l)} \\ \vdots \ \ddots \ \vdots \ \ddots \ \vdots \\ u_{M,1}^{(l)} \cdots u_{M,j}^{(l)} \cdots u_{M,N}^{(l)} \end{bmatrix}, i = 1,2, \dots, M, j = 1,2, \dots, N \tag{S1}$$

For mathematical computing, $\mathbf{U}^{(l)}$ can be expressed to be a complex-valued field combining amplitude and phase components:

$$\mathbf{U}^{(l)} = \mathbf{A}^{(l)} \exp^{j\boldsymbol{\Phi}^{(l)}}$$

$$where\ \mathbf{A}^{(l)} = \begin{bmatrix} A_{1,1}^{(l)} \dots A_{1,j}^{(l)} \dots A_{1,N}^{(l)} \\ \vdots \ \ddots \ \vdots \ \ddots \ \vdots \\ A_{i,1}^{(l)} \cdots A_{i,j}^{(l)} \cdots A_{i,N}^{(l)} \\ \vdots \ \ddots \ \vdots \ \ddots \ \vdots \\ A_{M,1}^{(l)} \cdots A_{M,j}^{(l)} \cdots A_{M,N}^{(l)} \end{bmatrix}, \boldsymbol{\Phi}^{(l)} = \begin{bmatrix} \Phi_{1,1}^{(l)} \dots \Phi_{1,j}^{(l)} \dots \Phi_{1,N}^{(l)} \\ \vdots \ \ddots \ \vdots \ \ddots \ \vdots \\ \Phi_{i,1}^{(l)} \cdots \Phi_{i,j}^{(l)} \cdots \Phi_{i,N}^{(l)} \\ \vdots \ \ddots \ \vdots \ \ddots \ \vdots \\ \Phi_{M,1}^{(l)} \cdots \Phi_{M,j}^{(l)} \cdots \Phi_{M,N}^{(l)} \end{bmatrix} \tag{S2}$$

$\mathbf{A}^{(l)}$ is the amplitude component of $\mathbf{U}^{(l)}$, with $A_{i,j}^{(l)}$ being the amplitude of $u_{i,j}^{(l)}$. $\boldsymbol{\Phi}^{(l)}$ is the phase component of $\mathbf{U}^{(l)}$, with $\Phi_{i,j}^{(l)}$ being the phase of $u_{i,j}^{(l)}$. For each $u_{i,j}^{(l)}$:

$$u_{i,j}^{(l)} = A_{i,j}^{(l)} \exp^{j\Phi_{i,j}^{(l)}} \tag{S3}$$

Considering $\mathbf{U}^{(l)}$ propagating along z-axis, the coordinate for each $u_{i,j}^{(l)}$ at l is $(x_{i,j}^{(l)}, y_{i,j}^{(l)}, z_0^{(l)})$ with a same z-axis position $z_0^{(l)}$. The output light field of layer l is denoted by $\mathbf{U}^{(l+1)}$ at layer l + 1, with each output pixel denoted by $u_{i,j}^{(l+1)}$ and a coordinate of $(x_{i,j}^{(l+1)}, y_{i,j}^{(l+1)}, z_0^{(l+1)})$.

First, $\mathbf{U}^{(l)}$ is modulated by the physically implemented weight matrix, $\mathbf{W}^{(l)}$, which is expressed to be:

$$\mathbf{W}^{(l)} = \begin{bmatrix} w_{1,1}^{(l)} \dots w_{1,j}^{(l)} \dots w_{1,N}^{(l)} \\ \vdots \ \ddots \ \vdots \ \ddots \ \vdots \\ w_{i,1}^{(l)} \cdots w_{i,j}^{(l)} \cdots w_{i,N}^{(l)} \\ \vdots \ \ddots \ \vdots \ \ddots \ \vdots \\ w_{M,1}^{(l)} \cdots w_{M,j}^{(l)} \cdots w_{M,N}^{(l)} \end{bmatrix}, w_{i,j}^{(l)} \in \{0, 1\} \tag{S4}$$

The neuron weighting of $\mathbf{W}^{(l)}$ on $\mathbf{U}^{(l)}$ is then expressed with the amplitude modulation through bitwise convolution [1]:

$$\mathbf{U}^{(l)} \circ \mathbf{W}^{(l)} = \begin{bmatrix} u_{1,1}^{(l)} w_{1,1}^{(l)} \cdots u_{1,j}^{(l)} w_{1,j}^{(l)} \cdots u_{1,N}^{(l)} w_{1,N}^{(l)} \\ \vdots \quad \ddots \quad \vdots \quad \ddots \quad \vdots \\ u_{i,1}^{(l)} w_{i,1}^{(l)} \cdots u_{i,j}^{(l)} w_{i,j}^{(l)} \cdots u_{i,N}^{(l)} w_{i,N}^{(l)} \\ \vdots \quad \ddots \quad \vdots \quad \ddots \quad \vdots \\ u_{N,1}^{(l)} w_{N,1}^{(l)} \cdots u_{M,j}^{(l)} w_{N,j}^{(l)} \cdots u_{M,N}^{(l)} w_{M,N}^{(l)} \end{bmatrix} \tag{S5}$$

Next, the diffractive fully-connections between l and $l + 1$ are given by the Rayleigh–Sommerfeld diffraction integral equation [1-2]. For all input pixels in l and an output pixel $u_{p,q}^{l+1}$, where $p = 1,2,\ldots,M, q = 1,2,\ldots,N$, the diffractive connection relationship, $\mathbf{h}_{p,q}^{(l)}$, is denoted with:

$$\mathbf{h}_{p,q}^{(l)} = \begin{bmatrix} h_{11,pq}^{(l)} \cdots h_{1j,pq}^{(l)} \cdots h_{1N,pq}^{(l)} \\ \vdots \quad \ddots \quad \vdots \quad \ddots \quad \vdots \\ h_{i1,pq}^{(l)} \cdots h_{ij,pq}^{(l)} \cdots h_{iN,pq}^{(l)} \\ \vdots \quad \ddots \quad \vdots \quad \ddots \quad \vdots \\ h_{M1,pq}^{(l)} \cdots h_{Mj,pq}^{(l)} \cdots h_{MN,pq}^{(l)} \end{bmatrix} \tag{S6}$$

where $h_{ij,pq}^{(l)}$ between each neuron $u_{i,j}^{(l)}$ and $u_{p,q}^{(l+1)}$, is given by the Rayleigh–Sommerfeld diffraction integral equation:

$$h_{ij,pq}^{(l)} = \frac{1}{2\pi} \frac{r^{(l)}}{{R_{ij,pq}^{(l)}}^2} \left(\frac{1}{2\pi R_{ij,pq}^{(l)}} + \frac{1}{j\lambda} \right) \exp\left(\frac{j2\pi R_{ij,pq}^{(l)}}{\lambda} \right) \tag{S7}$$

and $r^{(l)} = z_0^{(l+1)} - z_0^{(l)}$ is the z-axis distance between l and $l + 1$, $R_{ij,pq}^{(l)}$ is the spatial distance between $u_{i,j}^{(l)}$ and $u_{p,q}^{(l+1)}$:

$$R_{ij,pq}^{(l)} = \sqrt{\left(x_{p,q}^{(l+1)} - x_{i,j}^{(l)} \right)^2 + \left(y_{p,q}^{(l+1)} - y_{i,j}^{(l)} \right)^2 + \left(z_0^{(l+1)} - z_0^{(l)} \right)^2} \tag{S8}$$

The complete transferring relationship of each neuron between layer l and $l + 1$ is expressed to be:

$$u_{p,q}^{(l+1)} = \sum_{i=1}^{M} \sum_{j=1}^{N} \left(u_{i,j}^{(l)} w_{i,j}^{(l)} \right) \cdot h_{ij,pq}^{(l)} \tag{S9}$$

Extending to the transferring relationship to layers, $\mathbf{H}^{(l)}$ is the diffractive connection function matrix between layer l and neurons in layer $l + 1$:

$$\mathbf{H}^{(l)} = \begin{bmatrix} \mathbf{h}_{1,1}^{(l)} \cdots \mathbf{h}_{1,q}^{(l)} \cdots \mathbf{h}_{1,N}^{(l)} \\ \vdots \quad \ddots \quad \vdots \quad \ddots \quad \vdots \\ \mathbf{h}_{p,1}^{(l)} \cdots \mathbf{h}_{p,q}^{(l)} \cdots \mathbf{h}_{p,N}^{(l)} \\ \vdots \quad \ddots \quad \vdots \quad \ddots \quad \vdots \\ \mathbf{h}_{M,1}^{(l)} \cdots \mathbf{h}_{p,q}^{(l)} \cdots \mathbf{h}_{M,N}^{(l)} \end{bmatrix} \tag{S10}$$

and the fully-connected network structure of hidden layer l is described to be:

$$\mathbf{U}^{(l+1)} = (\mathbf{U}^{(l)} \circ \mathbf{W}^{(l)}) * \mathbf{H}^{(l)} \tag{S11}$$

Specifically, for the input layer, the input and output relationship is similarly expressed to be:

$$\mathbf{U}^{(1)} = (\mathbf{U}^{(\mathrm{in})} \circ \mathbf{IN}) * \mathbf{H}^{(\mathrm{in})} \tag{S12}$$

S.2. Prediction results via linear decision layer

The optical computing result, **OUT**, is captured with the high-speed camera, with a decision layer then applied on **OUT** for final predictions. The decision layer is a compact 1-layer linear FCNN, and the final prediction value for each possible category is

$$\mathbf{pred} = \left[pred_{\mathbf{1}}, pred_{\mathbf{2}}, \ldots, pred_{\mathrm{n_{catg}}}\right] \tag{S13}$$

where n_{catg} is the category number. The category decision $\hat{y}$ of the input sample, $\hat{y} \in \{1,2,\ldots,n_{\mathrm{catg}}\}$, is determined such that:

$$max(\mathbf{pred}) = pred_{\hat{y}} \tag{S14}$$

For each task, the computation load is quantified with the number of operations (OPs) required to complete the forward path through the network, including computing on both optical hidden layers and the decision layer. Considering the implemented HS-BONN has 3 hidden layers with 84480 neurons per layer, according to the OPs formular of FCNN [3], the computation load of each task on optical hidden layers is calculated to be:

$$L\left(2n_{in}n_{out}\right) = 3 \times (2 \times 84480 \times 84480) = 42.8\ \mathrm{GOPs} \tag{S15}$$

where L is the number of hidden layers, n_{in} and n_{out} are the input and output neuron number. The computation load for the linear decision layer with 10 output categories is:

$$2n_{\mathrm{in}}n_{\mathrm{out}} = 2 \times (84480 \times 10) = 1.69 \times 10^6\ \mathrm{OPs} = 1.69\ \mathrm{MOPs} \tag{S16}$$

And the proportion of the optical OPs against the total OPs is computed to be:

$$\frac{42.8 \times 10^9}{42.8 \times 10^9 + 1.69 \times 10^6} \approx 99.997\% \tag{S17}$$

S.3. Training progresses with HS-BONN

S.3.1. Training dataset preparation

The input dataset is then defined to be $\mathbb{IN} = [\mathbf{IN}^{(1)}, \mathbf{IN}^{(2)}, \mathbf{IN}^{(k)}, \ldots, \mathbf{IN}^{(S)}]$, where S is the total sample number. For each $\mathbf{IN}^{(k)}$, there is a corresponding true label, y_k, $y_k = 1,2,\ldots,n_{catg}$, indicating the correct category of the sample. The true label vector for the input dataset is denoted with $\mathbf{y} = [y_1, y_2, \ldots y_k, \ldots y_S]$.

When used for training, $\mathbb{IN}$ is further divided into two parts, $\mathbb{IN}^{(\text{train})}$ and $\mathbb{IN}^{(\text{test})}$, where $\mathbb{IN}^{(\text{train})}$ is used for NN trainings, and $\mathbb{IN}^{(\text{test})}$ is used for evaluating the NN performance. The training/testing set ratio is 80/20 in this work, according to common standard practices in NN trainings.

In each epoch of batch training, the training set $\mathbb{IN}^{(\text{train})}$ is then further divided into smaller batches $\mathbb{IN}^{(\text{train})} = [\mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \ldots, \mathbf{B}^{(m)}, \ldots \mathbf{B}^{(n_{\text{batch}})}]$, where n_{batch} is the batch number and $m = 1,2,\ldots,n_{batch}$. Each batch $\mathbf{B}^{(m)}$ is denoted as:

$$\mathbf{B}^{(m)} = \left[\mathbf{IN}^{(m,1)}, \mathbf{IN}^{(m,2)}, \ldots, \mathbf{IN}^{(m,r)}, \mathbf{IN}^{(m,B_{\text{size}})}\right], r = 1,2, \ldots, B_{\text{size}} \tag{S18}$$

and the corresponding batch labels $\mathbf{y}^{(m)}$ is:

$$\mathbf{y}^{(m)} = \left[y_1^{(m)}, y_2^{(m)}, \ldots, y_r^{(m)}, \ldots, y_{B_{\text{size}}}^{(m)}\right] \tag{S19}$$

where B_{size} is the batch size, and is selected based on the scale of $\mathbb{IN}$.

S.3.2. Training algorithms

Conventionally, the weights of a FCNN network are firstly initialized with random values, and the training process for each epoch contains following steps:

1) Forward pass: the input samples are fed into the network, with 1 batch $\mathbf{B}^{(m)}$ each time, and batch predictions $\mathbf{pred}^{(m)} = [\mathbf{pred}^{(m,1)}, \mathbf{pred}^{(m,2)}, \ldots \mathbf{pred}^{(m,r)}, \ldots, \mathbf{pred}^{(m,B_{\text{size}})}]$ are produced, with batch decisions $\hat{\mathbf{y}}^{(m)} = [\hat{y}_1^{(m)}, \hat{y}_2^{(m)}, \ldots, \hat{y}_r^{(m)}, \ldots, \hat{y}_{B_{\text{size}}}^{(m)}]$.

2) Loss calculation: the loss of the batch outputs is computed with a loss function, $\mathcal{L}()$, which quantifies how well the model is performing, and indicates the errors between the network outputs and true label. Here, the cross-entropy loss [4] is used as the loss function. For the prediction results of $\mathbf{B}^{(m)}$, the loss value, $C^{(m)}$, is computed to be

$$C^{(m)} = \mathcal{L}\left(\mathbf{y}^{(m)}, \mathbf{pred}^{(m)}\right) = -\frac{1}{B_{\text{size}}} \sum_{r=1}^{B_{\text{size}}} \log\left(P_r^{(m)}\right) \tag{S20}$$

where $P_r^{(m)}$ is the probability in outputting the correct prediction result $y_r^{(m)}$ based on $\mathbf{pred}^{(m,r)}$:

$$P_r^{(m)} = \frac{pred_{y_r^{(m)}}^{(m,r)}}{\sum_{i=1}^{n_{\text{catg}}} pred_i^{(m,r)}} \tag{S21}$$

3) Backward propagation: The gradient of the loss function is firstly calculated, with gradients denoted by *G*. For updating, Stochastic Gradient Descent [5], *SGD*(), is used to adjust the weights using the computed gradients with a learning rate, η. In this work, η is selected to be 0.04 for effective and steady weight updates.

4) Repeat to complete the epoch: Step 1) to 3) is repeated for each batch to complete the epoch.

When training BNNs, since the gradient for *sign*() function is always 0, the straight-through estimator (STE) algorithm is used to solve the gradient computing problem in backward propagation paths [6-7]. With STE, the gradient of *sign*() is estimated to be 1 across the gradient computing, such that:

$$\frac{\partial C}{\partial \mathbf{W}^{(l)}} = \frac{\partial C}{\partial sign(\mathbf{W}^{(l,\text{fp})})} \approx \frac{\partial C}{\partial \mathbf{W}^{(l,\text{fp})}} \tag{S22}$$

Hence, the original full-precision weight matrices $\mathbf{W}^{(l,\text{fp})}$ can be directly used for weight updates in back propagation, and the updated weights are then binarized again for next forward computing.

The training flowchart of each batch is given by algorithm 1. Note that all forward computing results are obtained *in-situ* with the physically implemented system, and the trained weights are also real-time updated to the physical system after each batch.

Algorithm 1 Backpropagation of HS-BONN with STE. The total layer number is *L*. $\mathcal{L}$() is the loss function with *C* being the loss value. *sign*() is the binarization function. *SGD*() is the Stochastic Gradient Descent process. η is the learning rate. *clip()* is used to restrain updated value between -1 and 1 to avoid very large values. *l* is the layer counting number. $\mathbf{A}^{(l)}$ is the middle-stage computing result of layer *l*.

Require: The current input $\mathbf{B}^{(m)}$ with true labels $\mathbf{y}^{(m)}$, where $m = 1,2,3,\ldots,n_{batch} - 1$. Current binary weight matrices $\mathbf{W}^{(l,m)}$, current full-precision weight matrices $\mathbf{W}^{(l,m,\text{fp})}$.

Ensure: Batch forward prediction outputs $\mathbf{pred}^{(m)}$, updated weight matrices $\mathbf{W}^{(l,m+1)}$.

{1. Forward propagation}

{All computing are conducted physically except decision layer.}

for *l* = *L* to 1 **do**

  if *l* = 1 **then**

    $\mathbf{A}^{(l-1)} = \mathbf{B}^{(m)}$

  end if

    $\mathbf{A}^{(l)} = \mathbf{A}^{(l-1)}\mathbf{W}^{(l,m)}$

  if *l* = *L* **then**

    $\mathbf{pred}^{(m)} = \mathbf{A}^{(l)}$

  end if

end for
{2. Back propagation}
{2.1. Computing loss and parameter gradients with STE:}
$C^{(m)} \leftarrow \mathcal{L}\,(\mathbf{y}^{(m)}, \mathbf{pred}^{(m)})$
for $l = L$ to 1 **do**
  if $l = L$ **then**
    $G_{C^{(m)}}^{(l)} \leftarrow \frac{\partial C^{(m)}}{\partial \mathbf{pred}^{(m)}}$
  end if
  $G_{sign()} \leftarrow 1$
  $G_{\mathbf{W}^{(l,m,\mathrm{fp})}}^{(l)} \leftarrow G_{sign()} \cdot G_{C^{(m)}}^{(l)} \cdot (\mathbf{A}^{(l-1)})^T$
  if $l > 1$ **then**
    $G_{C^{(m)}}^{(l-1)} \leftarrow G_{sign()} \cdot G_{C^{(m)}}^{(l)} \cdot (\mathbf{W}^{(l,m,\mathrm{fp})})^T$
  end if
end for
{2.2. Updating the parameters:}
for $l = 1$ to L **do**
  $\mathbf{W}^{(l,m+1,\mathrm{fp})} \leftarrow SGD(\mathbf{W}^{(l,m,\mathrm{fp})}, \eta, G_{\mathbf{W}^{(l,m,\mathrm{fp})}}^{(l)})$
  $\mathbf{W}^{(l,m+1,\mathrm{fp})} \leftarrow clip(\mathbf{W}^{(l,m+1,\mathrm{fp})})$
  $\mathbf{W}^{(l,m+1)} \leftarrow sign(\mathbf{W}^{(l,m+1,\mathrm{fp})})$
end for

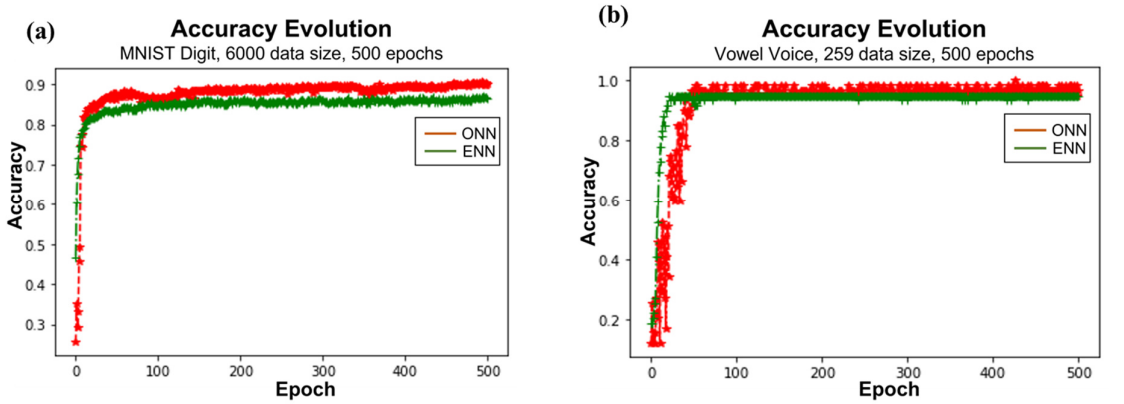

Fig. S1. (a) Accuracy evolution comparison on MNIST digit dataset, maximum accuracy: ENN 89.33%, HS-BONN 91.50%. (b) Accuracy evolution comparison on Vowel voice dataset, maximum accuracy: ENN 94.92%, HS-BONN 100%.

The accuracy evolution when training the models for MNIST digit and Vowel voice are shown as Fig. S1 (b) and (c). It can be observed that while experiencing similar evolution paths, the overall performance of HS-BONN is better than conventional same-structured ENNs throughout the training progresses.

S.4. Experimental HS-BONN system and calibration

The physically implemented HS-BONN system for experimental evaluations is presented as Fig. S2, with detailed descriptions included in Section 2.2 of the manuscript.

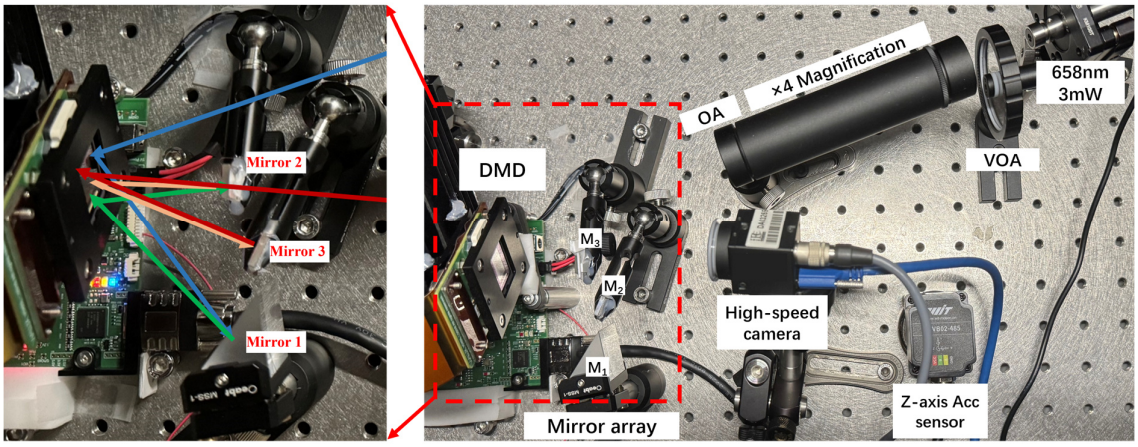

Fig. S2. The physical HS-BONN system implementation for evaluations, VOA: variable optical attenuator, OA: optical aperture, L: lens, M: mirror, In: input layer, h: hidden layer.

In the experimental setup, the core part is the realization of 4 reflections with the mirror array. To guarantee the precise reflection areas on the DMD, calibrations are required for precise forwarding paths. With the self-developed programmable calibration tool, the calibration progress is illustrated as Fig. S3.

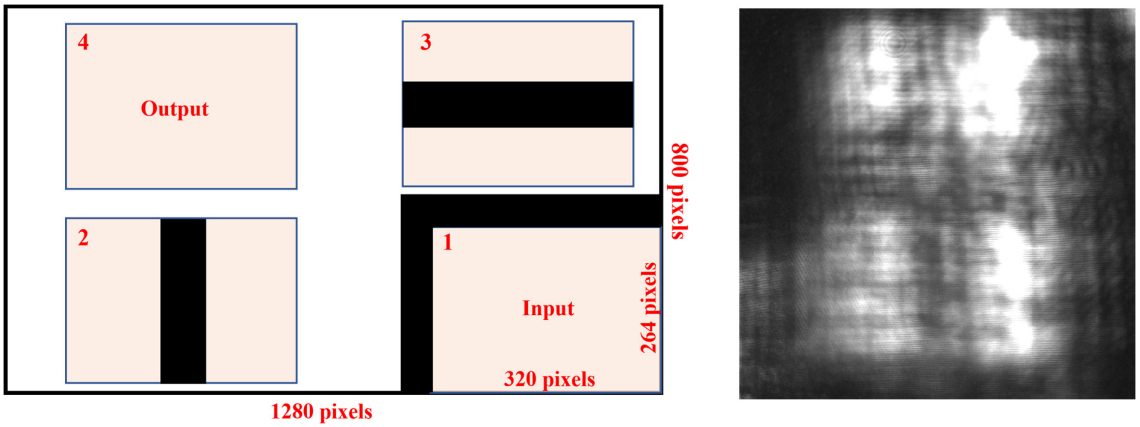

Fig. S3. Calibration of the DMD reflection path and an example of the correlated pattern.

First, in order to secure the pattern shape, the blank input pattern area is placed at one corner of the DMD illumination area, and a surrounding mask is further placed to ensure no additional interferences are input to the network. Next, by placing one vertical mask and one horizontal mask in the second and third reflection, the positions of each reflection can be observed with a cross on the final captured output from the fourth area. After careful adjustments of area boundaries, an example of the calibrated pattern is achieved, in which the effects of diffractive power-law nonlinearity can also be observed. Once calibrated, it has been proved that no additional calibration actions are required unless systemic damage is caused. The robustness of the system in long-term tasks and under environmental impacts have also been evaluated to be reliable, which is included in *Section S.6*.

Additionally, an error and exception handling mechanism are also implemented in the system. For most of the unexpected errors occurred during task conductions, such as data format exception, out of memory, and device not reachable, etc., the HS-BONN computing platform will handle the errors and reset the system without any manual operations, further enabling HS-BONN as an independent and effective versatile computing platform.

S.5. Serial/parallel modes of GPT-oriented task scheduler and training on MNIST fashion

As mentioned in the manuscript Section 2.3.1, the HS-BONN-driven GPT-oriented multi-task scheduler supports multiple serial/parallel modes for higher network capacity and conduction speed. The overall processing of the task scheduler can be divided into 2 main parts, dataset pre-processing on electronic devices (e.g., pre-processing of input samples, construction of task patterns), and the fast computing on HS-BONN. Based on this, the serial/parallel options supported by the scheduler include serial mode, hybrid mode, and parallel mode.

The selection of the modes is decided by the hardware availability. In the demonstration, one HS-BONN is realized, hence the experimental system currently supports the first 2 modes, serial mode and hybrid mode. For serial mode, all tasks are conducted in serial for both electronic processing and optical computing. Alternatively, since the all-optical forwarding path is computed at the speed of light, a hybrid mode is supported with parallel electrical pre-processing and serial HS-BONN conductions, so that the time-consumption can be further reduced.

Furthermore, consider multiple HS-BONN available in future, an all-parallel mode can also be conveniently supported with current framework, further bridging to parallel conductor scenarios.

In experimental evaluations, the hybrid mode is demonstrated for massively parallel potential. The overall dataflow chart of the hybrid mode is presented as Fig. S4.

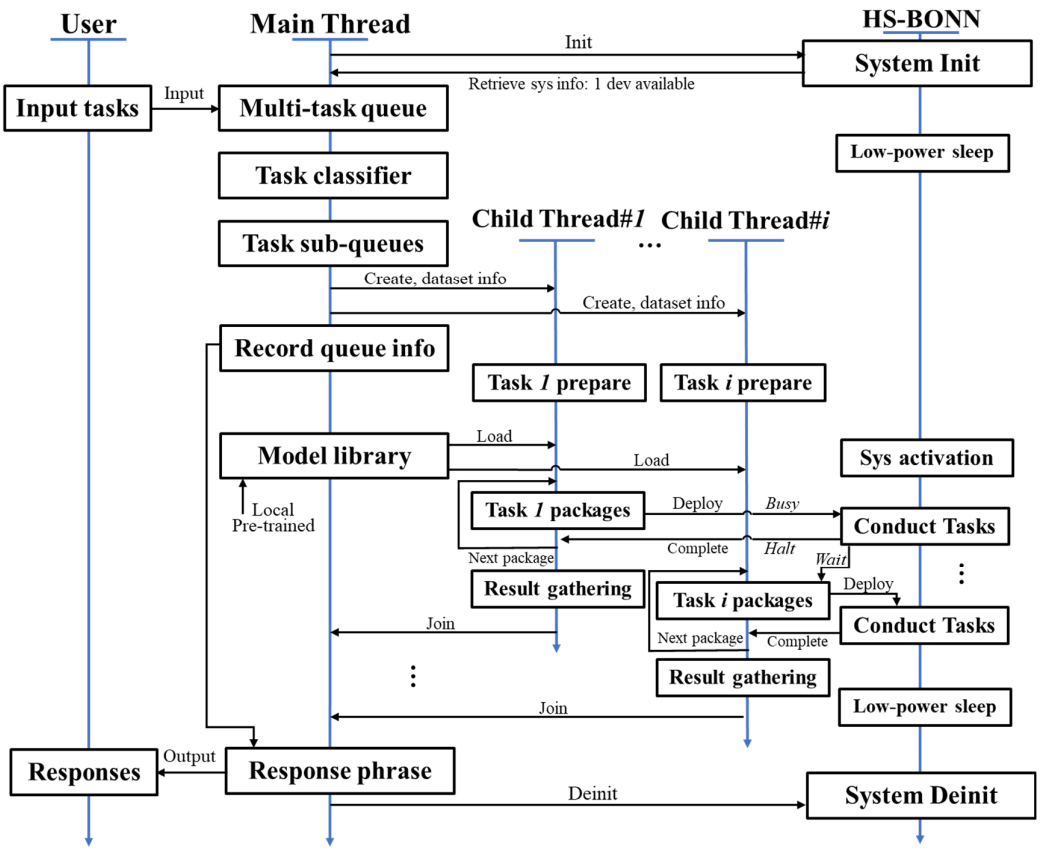

Fig. S4. The dataflow chart of the GPT-oriented image task scheduler on hybrid mode. The character `User` denotes to the operations of the application users, the character `Main Thread` denotes to the system application front-end interface, and HS-BONN is the implemented optical system as black-box computing platform.

From the user's interface, the system directly produces responses based on the input tasks, without any additional operations required.

When user inputs the tasks, which is a multi-task queue, the main thread is created on the electronic side. First, the main thread will scan the connected HS-BONN devices, and an available device list will be created based on the information retrieved (1 HS-BONN available in this case). The HS-BONN is first initialized to be ready for use.

The multi-task queue is then input to the task classifier. Since the electronic pre-processing will be conducted in parallel, the overall task queue is divided and formed into sub-task queues based on the dataset type. The original information of the tasks will be stored locally for later queue reconstruct and response phrase.

Next, for each sub-task queue, a child-thread is created for parallelly conducting the potentially time-consuming task preparations. In each child-thread, pre-trained models are

retrieved from the local model library, and task packages of the sub-task queue are constructed to be deployed on HS-BONN.

The arrangement of the serial conduction on HS-BONN is realized through two defined statues, `Halt` and `Busy`. When HS-BONN is `Halt` without tasks, child-threads can deploy tasks to HS-BONN for conductions, while marking the statues of HS-BONN to `Busy`. When another child-thread attempts to deploy tasks at this moment, the HS-BONN will respond `Wait` to until current task is completed, then mark its status back to `Halt`. While HS-BONN starts to conduct other tasks, task-finished child-threads can start to gather results and join back to the main-thread, enabling over-laps between tasks and further reduce time-consumptions.

After all tasks are completed, the main thread can then produce a summary based on the conduction results based on the recorded queue information and notify users of the final response.

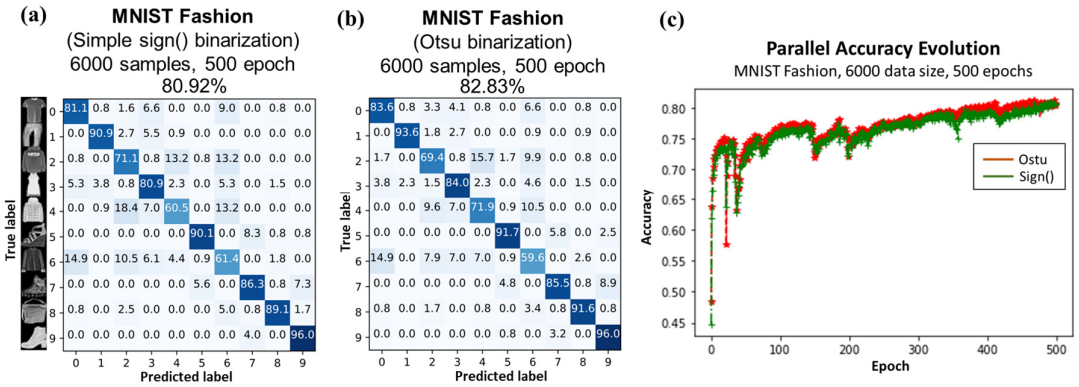

Fig. S5. Binarization algorithm comparison on MNIST fashion. (a) Confusion matrix with sign() function. (b) Confusion matrix with Otsu algorithm. (c) Accuracy evolution of the two training processes.

Specifically, when used for simultaneously model training of different datasets, the overall dataflow is similar, with the batch conduction and update of each model conducted by turns. The training of each model is controlled by one child-thread, and the update of each batch is referred as one package of the child-thread. In this way, an additional advantage of using hybrid mode is that, these model training processes will experience the same environmental condition, hence the trained results can present a direct reflection of the performance differences of HS-BONN on the simultaneously trained datasets, unifying the environment influences.

Here, the usage of hybrid mode is demonstrated through training on MNIST fashion. It contains samples of 28×28 resolution greyscale fashion item images. When evaluated on this dataset, 6000 shuffled samples are used, with 4800 samples for training, and 1200 samples for testing.

At the meantime, since both the brightness and contrast of samples in MNIST fashion can be different, the effectiveness of binarization algorithms is able to be revealed when training on it. Uutilizing the additional advantages of environmental influences cancellation brought by the parallel processing of tasks, a comparison of image binarization algorithms is also provided here between Otsu algorithm and a simple sign() function, emphasizing more system potential with future advanced binarization algorithms.

The accuracy comparison of binarization algorithms on MNIST fashion is shown as Fig. S5. It can be observed that, although with environmental interferences, a constant higher accuracy is achieved using Otsu algorithm, with an accuracy of 82.83% against 80.75% using simple sign() function, which indicates more performance potential of the system with advanced binarization algorithms.

S.6. HS-BONN long-term reliability evaluation

For HS-BONN framework, since the training and updating processes are conducted *in-situ*, it is likely that sudden light path drifts occur during long-term and large-scale tasks. Therefore, the resistance against environmental background vibrations and the resilience in disturbance recovery are crucial for reliable performances in real-case applications, especially under common industrial circumstances.

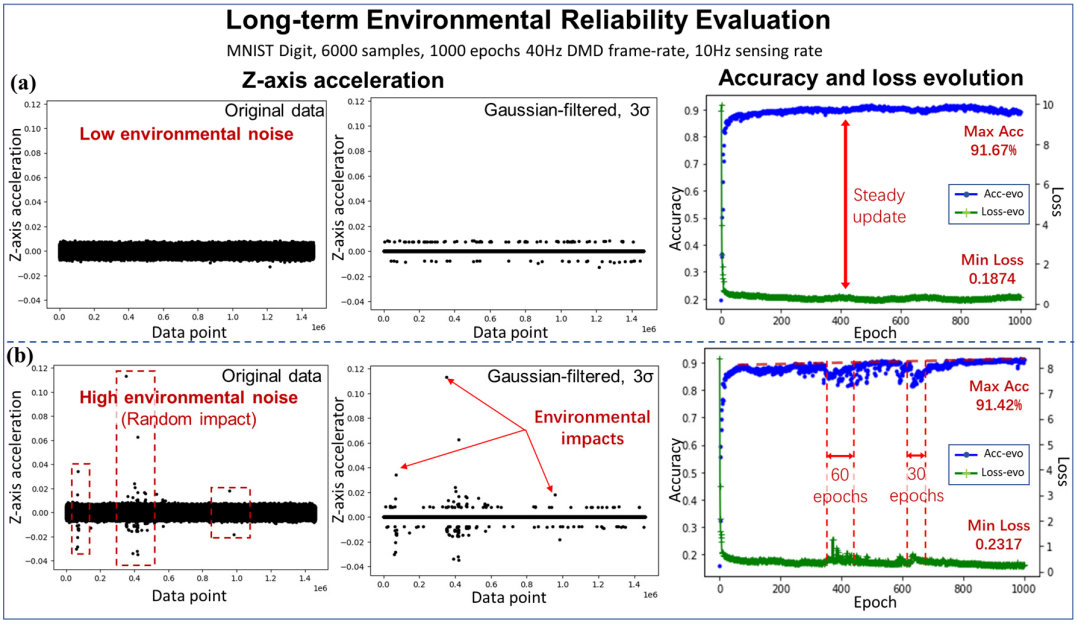

Fig. S6. Long-term environmental reliability evaluation. (a) Performance under low environmental noises. (b) Performance under high environmental noises with random impacts on the setup platform.

In the long-term reliability evaluation, the framework is implemented on an experimental damping-tuned platform (Newport RS2000), on which high-frequency vibrations can be

canceled while still sensitive to low-frequency noises. A 1000-epoch *in-situ* training task of MNIST digit with 6000 samples is conducted at a controlled speed of 40 Hz DMD switching rate for a 40-hour long-term task. During the evaluation, all other experiment setups on the same platform and surrounding environments are operated as usual, introducing potential environmental disturbances as in real application scenarios. In order to monitor and quantify the background vibrations and noises, a z-axis accelerator (WIT WT-VB02) is attached on the platform with a sampling rate of 10 Hz. The long-term environmental monitoring data, the *in-situ* training accuracy evolution curves, and the loss evolution curves of both common noise level and high noise level conditions are presented in Fig. S6 (a) and (b), respectively.

As shown in Fig. S6 (a), when no sudden environmental interferences occurred during the task, the *in-situ* training progresses steadily, achieving a maximum accuracy of 91.67% and a minimum loss of 0.1874. In Fig. S6 (b), 3 major impacts occurred during the task, which likely resulted from tool drops or large equipment movements, and can be more clearly observed by applying a Gaussian filter with 3σ threshold. For the first impact, since it occurred during the rapidly evolving period of the network training, its influence on the system was minor and no sudden accuracy/loss change is observed. The second and third impact, however, occurred during the fine-tuning period of the network progressing. While noticeable system disturbances were introduced, the framework was able to recover from the two impacts after fast adjustments of 60 and 30 epochs, which only accounts for 6% and 3% of the overall training time. With an only slightly lower accuracy of 91.42%, the reliability of our framework in long-term and large-scale tasks under common application scenarios can be verified.

S.7. Potential nonlinearity through multiple reflections on DMD

A common nonlinearity that diffractive ONNs utilize is the square-law nonlinearity introduced by capturing outputs with cameras. However, when experiencing multiple reflections on the DMD modulation area, it is reported that additional power-law nonlinearities can be introduced [8], improving the system's overall performance. Hence, the potential optical power-law nonlinearity effect with the implemented HS-BONN is experimentally evaluated here.

The nonlinearity effect is firstly assessed and verified with a *k*-nearest neighbor (KNN) classifier and neighborhood component analysis (NCA). According to the dataflow path in the HS-BONN framework, the feature component spaces of 3 MNIST digit dataset stages, **1)** original grey images without nonlinearity, **2)** with *sign()* nonlinearity defined by BNN, and **3)**

with further optical power-law nonlinearity after 4 reflections on blank DMD patterns, are identified as in Fig. S7.

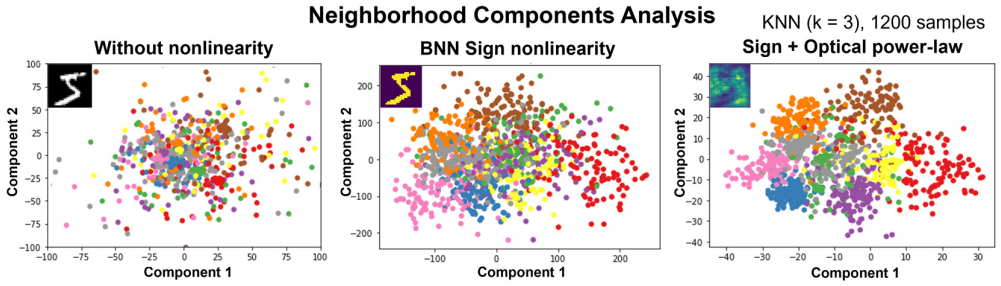

Fig. S7. NCA evaluation of different stages of applied nonlinearity via KNN on MNIST Digit: original dataset without nonlinearity, after *sign()* nonlinearity defined by BNN, and optical power-law nonlinearity with 4 reflections and all blank DMD patterns.

In the feature space, digit categories (*'0'* to *'9'*) are represented by different colors, and each component coordinate is determined such that a best accuracy is given by the stochastic nearest neighbor algorithm. For original dataset without nonlinearity, the features of all categories are coupled together and cannot be effectively separated. After inputting the dataset into the BONN framework and applying *sign()* nonlinearity defined by the structure characteristic of BNN, while a preliminary decoupling of the data categories is observed, the feature spaces are still interfering with each other. By further applying the 4 reflections on DMD, it can then be observed that more concentrated and category-separated feature spaces are achieved, visualizing the effectiveness of the nonlinearity in HS-BONN framework.

Next, a numerical evaluation is conducted through simulation with angular momentum models [9-10]. By simulating the physically implemented system with same parameters, the input and output relationship is evaluated after each reflection on the DMD.

For each reflection on the DMD, the $\mathbf{U}^{(l)}$ and $\mathbf{U}^{(l+1)}$ are captured during simulations with blank patterns on the DMD, and a log-log plot is applied between them after applying fast Fourier transform (FFT) on them. When fitting a power-law relationship between $\mathbf{U}^{(l)}$ and $\mathbf{U}^{(l+1)}$, the relationship is expressed to be:

$$\mathbf{U}^{(l+1)} = \alpha(\mathbf{U}^{(l)})^{\gamma} \tag{S23}$$

where α is the proportionality constant and γ is the power-law factor. For a linear relationship, $\gamma = 1$. After applying a log-log plot, the relationship transformed to:

$$\log\left(\mathbf{U}^{(l+1)}\right) = \gamma \log\left(\mathbf{U}^{(l)}\right) + \log\left(\alpha\right) \tag{S24}$$

which can be fitted with a linear function between $\log\left(\mathbf{U}^{(l+1)}\right)$ and $\log\left(\mathbf{U}^{(l)}\right)$, with the fitted slope being the power-law factor γ.

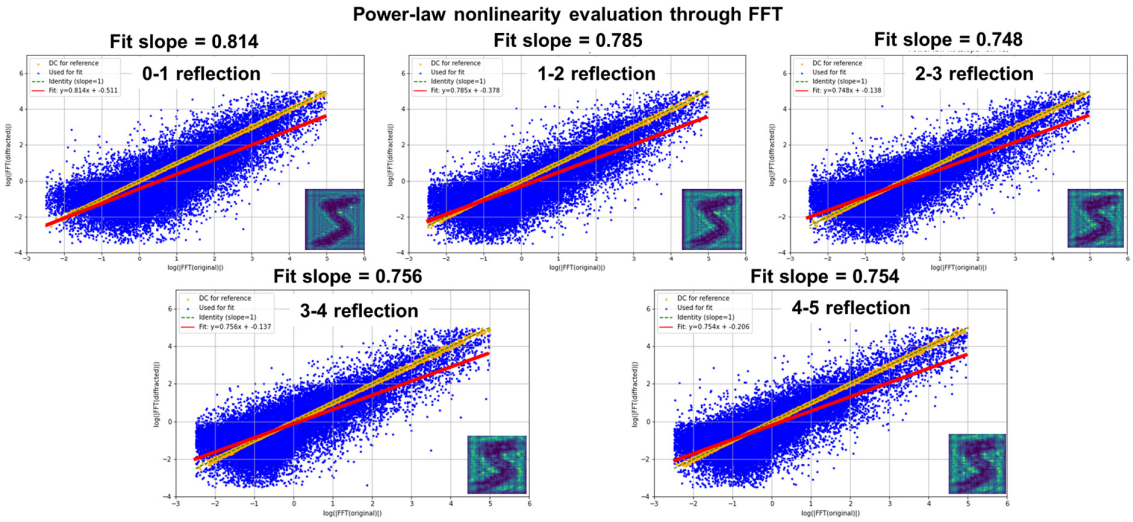

Fig. S8. The log-log plots between the input and output light field of first 5 reflections simulated from physical system implementation and parameter. The yellow line is DC signal (slope = 1) for reference, red lines are the fitted relationships.

The log-log plots and fitted linear relationships of the first 5 reflections on the DMD are presented as in Fig. S8. The yellow line represents a linear relationship (slope = 1) for reference, and red lines are the fitted relationships. It can be observed that a steady $\gamma \approx 0.75$ is obtained after the second reflection, indicating a compressive power-law nonlinearity with a power-law factor of 0.75, providing similar effects to Sigmoid or Tanh functions.

From the perspective of NN training, a compressive nonlinearity can serve as a foundational mechanism for stability and feature extraction, mirroring the behavior of standard saturating activation functions [11]. For optical systems, this can also introduce the advantages of suppressing high-intensity noise spikes, compressing the dynamic range of the input and allowing the system to resolve features across a wider range of input brightness levels.

While more theoretical analysis is further required before fully-utilizing the power-law nonlinearity in the system, this have indicated the high potential of HS-BONN on more complex tasks in future applications.

Reference

1. Zheng M, Lei S, and Jian Z, Optimize performance of a diffractive neural network by controlling the Fresnel number, *Photonics Res.* 10.11: 2667-2676 (2022).
2. Born M, Wolf E. Principles of optics: electromagnetic theory of propagation, interference and diffraction of light. *Elsevier*, (2013).
3. Sze V, et al., Efficient processing of deep neural networks: A tutorial and survey, *Proceedings of the IEEE*, 105.12: 2295-2329 (2017).

4. Mao A, Mehryar M, and Zhong, Y, Cross-entropy loss functions: Theoretical analysis and applications, *International conference on Machine learning*. pmlr, (2023).
5. Amari S, Backpropagation and stochastic gradient descent method, *Neurocomputing* 5.4-5: 185-196 (1993).
6. Courbariaux M, Yoshua B, Binarynet: Training deep neural networks with weights and activations constrained to+ 1 or-1, *arXiv preprint*, arXiv:1602.02830 (2016).
7. Yin P, et al., Understanding straight-through estimator in training activation quantized neural nets, *arXiv preprint* arXiv:1903.05662 (2019).
8. Xia F, et al., Nonlinear optical encoding enabled by recurrent linear scattering, *Nat. Photonics* **18**, 1067-1075 (2024).
9. Allen L, et al., Orbital angular momentum of light and the transformation of Laguerre-Gaussian laser modes, *Physical review A* 45.11: 8185 (1992).
10. Draine B T, and Piotr J F, Discrete-dipole approximation for scattering calculations, *Journal of the Optical Society of America A* 11.4: 1491-1499 (1994).
11. Glorot X, and Yoshua B, Understanding the difficulty of training deep feedforward neural networks, *Proceedings of the thirteenth AISTATS*. JMLR Workshop and Conference Proceedings, (2010).

Appendix

A.1. Notation table

Notation	Description
λ	Wavelength of input light.
l	Layer index.
L	Optical hidden layer number.
S	Input task number.
$\mathbb{IN}$	Input dataset.
$\mathbb{IN}^{(\text{train})}$	Training set.
$\mathbb{IN}^{(\text{test})}$	Testing set.
$\mathbf{IN}$	Input image at ONN input layer.
$IN_{i,j}$	Pixel *i*, *j* of input image.
$\mathbf{IN}^{(\text{fp})}$	Full-precision input sample.
$in_{i,j}^{(\text{fp})}$	Pixel *i*, *j* of full precision input image.
$\mathbf{IN}^{(k)}$	Image *k* of input dataset.
g^*	Optical binarization threshold given by Otsu algorithm.
$\mathbf{W}^{(l)}$	Weight matrix of layer *l*.
$w_{i,j}^{(l)}$	Weight *i*, *j* at layer *l*.
$\mathbf{W}^{(l,\text{fp})}$	Full-precision weight matrix of layer *l*.
$w_{i,j}^{(l,\text{fp})}$	Full-precision weight *i*, *j* at layer *l*.
$\mathbf{OUT}$	Optical output intensity captured by camera.
$\mathbf{pred}$	Prediction vector of decision layer.
$pred_i$	Prediction value of i^{th} category for an input.
n_{catg}	Number of sample category.
y	True label.
y_k	True label of input *k*.
$\mathbf{y}$	Label vector of input dataset.
$\hat{y}$	Predicted category.
$\mathbf{D}$	DMD total modulation matrix.
n_{batch}	Batch number in $\mathbb{B}$.
$\mathbf{B}^{(m)}$	The m^{th} in $\mathbb{B}$
B_{size}	Batch size.

$\mathbf{IN}^{(m,r)}$	r^{th} input in batch m.
$\mathbf{y}^{(m)}$	Ture label vector of batch m.
$y_r^{(m)}$	Ture label of r^{th} task in batch m.
$\hat{\mathbf{y}}^{(m)}$	Predicted categories of batch m.
$\hat{y}_r^{(m)}$	Predicted category of r^{th} task in batch m.
$\mathbf{pred}^{(m)}$	Prediction vectors of batch m.
$\mathbf{pred}^{(m,r)}$	Prediction vector of r^{th} task in batch m.
$pred_i^{(m,r)}$	Prediction value of i^{th} category for r^{th} task in batch m.
$\mathbf{W}^{(l,m)}$	Weight matrix of layer l for batch m.
$\mathbf{W}^{(l,m,\text{fp})}$	Full-precision weight matrix of layer l for batch m.
$\mathcal{L}(\ )$	Loss function.
$C^{(m)}$	Cross-entropy loss of batch m.
$P_r^{(m)}$	Correct prediction probability of r^{th} task in batch m.
η	Learning rate.
$\mathbf{A}^{(l)}$	Middle-stage computing result of layer l.
$G_{C^{(m)}}^{(l)}$	Gradient of $C^{(m)}$ at layer l.
$G_{sign()}$	Gradient of *sign()* function.
$G_{\mathbf{W}^{(l,m,\text{fp})}}^{(l)}$	Gradient of $\mathbf{W}^{(l,m,\text{fp})}$ at layer l
M, N	Matrix dimension of each optical layer.
$\mathbf{U}^{(l)}$	Light field at layer l.
$\mathbf{U}^{(\text{in})}$	Input light field to ONN system.
$\mathbf{U}^{(\text{out})}$	Output light field to ONN system.
$u_{i,j}^{(l)}$	Input pixel i, j at layer l.
$\mathbf{A}^{(l)}$	Amplitude component matrix of $\mathbf{U}^{(l)}$.
$A_{i,j}^{(l)}$	Amplitude of $u_{i,j}^{(l)}$.
$\boldsymbol{\Phi}^{(l)}$	Phase component matrix of $\mathbf{U}^{(l)}$.
$\Phi_{i,j}^{(l)}$	Phase of $u_{i,j}^{(l)}$.
$(x_{i,j}^{(l)}, y_{i,j}^{(l)}, z_0^{(l)})$	Spatial coordinate of $u_{i,j}^{(l)}$.
$\mathbf{H}^{(IN)}$	Diffractive connection matrix between input layer and hidden layer 1.
$\mathbf{H}^{(l)}$	Diffractive connection matrix between hidden layer l and $l+1$.
$\mathbf{h}_{p,q}^{(l)}$	Diffractive connection matrix between layer l and $u_{p,q}^{(l+1)}$.
$h_{ij,pq}^{(l)}$	Diffractive connection between $u_{i,j}^{(l)}$ and $u_{p,q}^{(l+1)}$.
$r^{(l)}$	z-axis distance between layer l and $l+1$.
$R_{ij,pq}^{(l)}$	Spatial distance between $u_{i,j}^{(l)}$ and $u_{p,q}^{(l+1)}$.
OPs	Computing operation number.
n_{in}	Number of hidden layer input neurons.
n_{out}	Number of hidden layer output neurons.
α	Proportionality constant of a power-law relationship.
γ	Power-law factor of a power-law relationship.