Optical Diffractive Neural Networks for Sub-Rayleigh Object Classification

Nico Deshler[1,⋆], *Jacob* Trzaska[1,⋆⋆], *Itay* Ozer[1], *Wenhua* He[1], *Amit* Ashok[1], and *Saikat* Guha[1]

[1] *Wyant College of Optical Sciences, University of Arizona*

Abstract. We perform image classification for objects smaller than the Rayleigh resolution limit using diffractive optical neural networks. We also compare the performance of this technique to that of a digital classifier trained on spatial mode sorting measurements and a classifier trained on direct detection measurements.

1 Introduction

All-optical diffractive neural networks (ONNs) are analog optical computing devices that can perform passive image classification at the speed of light. [1] These devices are composed of a series of learned phase masks that transform the incident electromagnetic field into a probability distribution over the classes. The phase masks are learned offline in simulation before being deployed in the real world with spatial light modulators or 3D-printed diffractive plates. Previous ONN architectures have classified handwritten digits with up to 93.39% accuracy (5-layer system), a performance comparable to the state-of-the-art classification accuracy (99.77%) achieved with convolutional neural networks.

The physics underpinning ONNs are reminiscent of the multiplane light converter (MPLC) which has been used to experimentally demultiplex a monochromatic optical field into a collection of orthogonal spatial modes. Recently, the quantum imaging community found that optical mode sorting poses a viable measurement technique for successfully estimating features of a scene that are smaller than the Rayeleigh resolution limit of the imaging system. In this work we seek to unite ONNs with the insights from the quantum imaging community to create an all-optical classifier for identifying objects beyond the diffraction limit. We also endeavor to provide a quantum information-theoretic justification for the success of this implementation. Finally we compare our ONN classifier to two alternative neural networks - one which is trained on Hermite-Gauss mode-sorting measurements, and another which is trained on direct detection measurements.

2 Theory

Consider a diffractive neural network consisting of N phase planes as shown in figure 1. The objective of this network is to classify a monochromatic input field into one of K classes. Taking $\ell = 1, 2, \ldots, N$ to be an index over phase planes, we define the optical field immediately to the left (upstream) and right (downstream) of the ℓ^{th} phase plane to be $\psi_\ell^{(-)}$ and $\psi_\ell^{(+)}$ respectively. For thin diffractive elements these fields are related to each other through a point-wise application of the phase mask shown in equation 1.

$$\psi_\ell^{(+)}(x,y) = e^{i\phi_\ell(x,y)}\psi_\ell^{(-)}(x,y) \tag{1}$$

The z-coordinates are implied by the index as the previously-defined fields coincide with the phase planes. Between adjacent planes, the field propagates according to Rayleigh-Sommerfeld diffraction. In this work, we assume that each phase element is a square with side length L so that the plane-to-plane mapping is given by equation 2

$$\psi_{\ell+1}^{(-)}(x,y) = \int_{-L/2}^{L/2}\int_{-L/2}^{L/2} dx'dy' \; \psi_\ell^{(+)}(x',y') \times \exp(ikr)\frac{z_{\ell+1}-z_\ell}{r^2}\Big(\frac{1}{2\pi r} - i\frac{1}{\lambda}\Big) \tag{2}$$

where $k = 2\pi/\lambda$ is the wavenumber, and $z_\ell, z_{\ell+1}$ are the axial positions of the adjacent planes. Equation 2 is a 2D convolution with the Huygens kernel,

$$h_\ell(x,y;x',y') = \frac{z_{\ell+1}-z_\ell}{r^2}\Big(\frac{1}{2\pi r} + \frac{1}{i\lambda}\Big)\exp(ikr) \tag{3a}$$

$$r = \sqrt{(x-x')^2 + (y-y')^2 + (z_{\ell+1}-z_\ell)^2} \tag{3b}$$

For notational convenience, we define a propagation operator $\hat{P}_\ell$ mapping the outgoing field of plane ℓ to the incoming field of plane $\ell + 1$.

$$\psi_{\ell+1}^{(-)} = \hat{P}_\ell(\psi_\ell^{(+)}) = h_\ell ** \psi_\ell^{(+)} \tag{4}$$

A visual representation of this mapping is given in figure 2. Given a known input field $\psi_1^{(-)}$ incident on the first

⋆e-mail: ndeshler@arizona.edu

⋆⋆e-mail: jtrzaska@arizona.edu

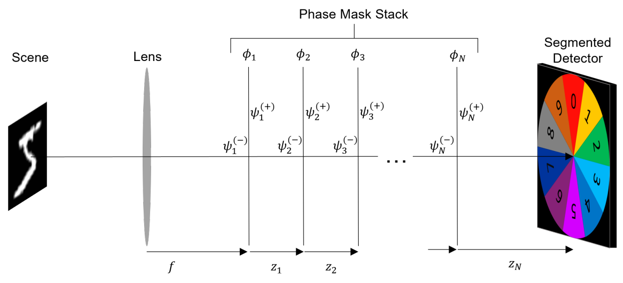

Figure 1. The system diagram for a diffractive optical neural network. The object is assumed to be at infinity and the first phase plane is taken to be at the back focal distance of the imaging system. The light propagates from plane to plane under the Rayleigh-Sommerfield diffraction equation, accruing a pointwise phase modulation at each plane. The detector is partitioned into regions of equal area, one for each class in the training data.

phase plane, the field at the detector plane ψ_D can be expressed as alternating applications of phase masking followed by propagation.

$$\psi_D = \hat{P}_N\Big(e^{i\phi_N}\hat{P}_{N-1}\Big(e^{i\phi_{N-1}}\dots\hat{P}_1\Big(e^{i\phi_1}\psi_1^{(-)}\Big)\dots\Big)\Big) \tag{5}$$

To perform classification, the detector is segmented into K mutually exclusive (non-overlapping) regions of equivalent area, one associated with each class. The relative light intensity in each segment relates to the probability of each class through the softmax function.

$$p_k = \frac{e^{\mathrm{v}_k}}{\sum_{k'} e^{\mathrm{v}_{k'}}} \tag{6a}$$

$$\mathrm{v}_k \propto \int_{\mathcal{S}_k} |\psi_D(x,y)|^2 \tag{6b}$$

where $\mathbf{v} \in \mathbb{R}^K$ is a column vector containing the cumulative intensity measured in the detector segment with domain $\mathcal{S}_k$ and $\mathbf{p} \in \mathbb{R}^K$ is the discrete probability distribution over the classes given the input field. The goal of a ONN is to optimize the phase functions ϕ_ℓ so as to minimize a given loss function that depends on the probabilities over the classes. In the next section we move towards a discrete representation of the system to facilitate employing computational machine-learning techniques.

3 Feed-Forward Network

The system is modelled as a discrete feed-forward network. Unlike traditional neural networks, optical neural networks involve purely linear operations without non-linear activation functions. In this regard, using the term 'neural network' to describe ONNs is somewhat of a misnomer as there is a strong departure from mathematical equivalence these optical systems and their digital counterparts. In fact, it is precisely the non-linear activation functions in digital neural networks which grant them power

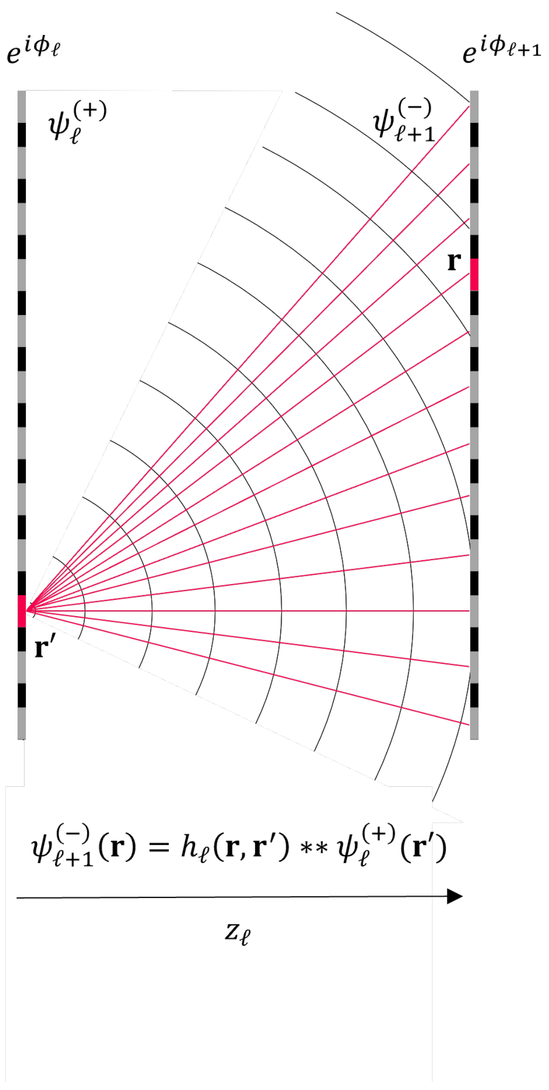

Figure 2. A depiction of the outgoing 'weights' from each diffractive neuron (phase plane pixel). Layer ℓ couples to layer $\ell + 1$ via convolution with the rayleigh-sommerfield diffraction kernel.

as generalized function estimators. While optical neural networks do not have non-linear elements, they do enjoy and additional degree of freedom from which to extract information, namely the phase of the light. That said, digital neural networks with complex-valued weights would enjoy the same degree of freedom.

3.1 Discretization

In analogy to digital neural networks, we will introduce the notion of a 'neuron' by discretizing each plane into pixels. For simplicity we model each phase plane as a square grid composed of $M \times M$ pixels so that pixel width is $\Delta = L/M$. To discretize the functions at each phase plane, we take each pixel to be the average of the continuous function over the area of the pixel centered at (x_n, y_m) where $x_n = -(L - \Delta n)/2$, $y_m = -(L - \Delta m)/2$, and $n, m = 1, \dots, M$

$$\psi_\ell^{(\pm)}[n,m] = \frac{1}{\Delta^2}\int_{x_n-\Delta/2}^{x_n+\Delta/2}\int_{y_m-\Delta/2}^{y_m+\Delta/2} dxdy\ \psi_\ell^{(\pm)}(x,y) \quad (7a)$$

$$\phi_\ell[n,m] = \frac{1}{\Delta^2}\int_{x_n-\Delta/2}^{x_n+\Delta/2}\int_{y_m-\Delta/2}^{y_m+\Delta/2} dxdy\ \phi_\ell(x,y) \quad (7b)$$

Next, we vectorize the discretized functions by converting them into $M^2 \times 1$ column-vectors. Defining the function $\mathbf{x} = \texttt{vec}(X)$ as that which takes the *column-major* vectorization of a matrix X, we have

$$\boldsymbol{\psi}_\ell^{(\pm)} = \texttt{vec}(\psi_\ell^{(\pm)}[n,m]) \quad (8a)$$

$$\boldsymbol{\phi}_\ell = \texttt{vec}(\phi_\ell[n,m]) \quad (8b)$$

Correspondingly, we put all of the Huygens convolution kernels into a $M^2 \times M^2$ matrix where each column of H_ℓ is the vectorized version of the kernel at shift location $x_{n'}, y_{m'}$.

$$H_\ell = \begin{bmatrix} | & | & & | & & | \\ \boldsymbol{h}_{11}^\ell & \boldsymbol{h}_{21}^\ell & \cdots & \boldsymbol{h}_{M1}^\ell & \cdots & \boldsymbol{h}_{MM}^\ell \\ | & | & & | & & | \end{bmatrix} \quad (9)$$

where $\boldsymbol{h}_{n'm'}^\ell = \texttt{vec}(h_\ell[x_n, y_m; x_{n'}, y_{m'}]$. This is the so-called 'Toeplitz matrix' of the convolution (Toeplitz Matrix Represenation of Convolution) Propagation of the field between adjacent planes is then simply a matrix-vector multiplication

$$\boldsymbol{\psi}_{\ell+1}^{(-)} = H_\ell \boldsymbol{\psi}_\ell^{(+)}$$

We may also define the phase mask matrices matrix $\Lambda_\ell = \texttt{diag}\Big(\exp(i\boldsymbol{\phi}_\ell)\Big)$. Then the forward model from the input field $\boldsymbol{\psi}_1^{(-)}$ to the detector $\boldsymbol{\psi}_D$ can be written entirely via matrix-vector multiplication.

$$\boldsymbol{\psi}_D = H_N\Lambda_N \cdots H_1\Lambda_1\boldsymbol{\psi}_1^{(-)} \quad (10)$$

As mentioned previously, to perform classification the detector is segmented into K mutually exclusive regions, one for each class. Our measurement of interest is the total intensity in each region. Let $S_k \in \mathbb{R}^{M\times M}$ be a binary mask for the detector pixels associated with class k. Vectorizing each mask $\mathbf{s}_k = \texttt{vec}(S_k)$ and collecting them into a matrix $S \in \mathbb{R}^{K\times M^2}$ we have

$$S = \begin{bmatrix} - & \mathbf{s}_1^T & - \\ & \vdots & \\ - & \mathbf{s}_K^T & - \end{bmatrix} \quad (11)$$

with which we can express the cumulative intensity in each region as,

$$\mathbf{v} = S\,|\boldsymbol{\psi}_D|^2 \quad (12)$$

where $|\boldsymbol{\psi}_D|^2 = (\boldsymbol{\psi}_D^* \odot \boldsymbol{\psi}_D)$. From here the probabilities are computed with the softmax function

$$\mathbf{p} = \texttt{softmax}(\mathbf{v})$$

and we choose the class with highest probability to be our final classification of the input.

3.2 Requirements for Fully-Connected Layers

For a diffractive optical networks to be 'fully-connected' we require that the inter-plane spacing be large enough that the field emitted from any part of one layer 'reaches' all parts of the adjacent layer. This amounts to ensuring that the magnitude of the normalized Rayleigh-Sommerfeld kernel is greater than a dimensionless threshold value $\tau \in [0,1]$. This threshold value is a design choice set by the user which trades off connectivity (larger τ) with light throughput (lower τ). Since the magnitude of the Rayleigh-Sommerfeld kernel is strongest at the points of minimum separation $r_{min} = \delta_z = (z_{\ell+1} - z_\ell)$ and uniformly decays for greater separations, we need only impose the threshold requirement for the largest neuron-to-neuron distance. The maximum neuron-to-neuron distance is the distance connecting opposite corners of adjacent layers $r_{max} = \sqrt{2L^2 + \delta_z^2}$. Hence a fully connected network must satisfy the inequality

$$\frac{|h(r_{max})|}{|h(r_{min})|} = \left(\frac{r_{min}}{r_{max}}\right)^3 \sqrt{\frac{\lambda^2 + (2\pi r_{max})^2}{\lambda^2 + (2\pi r_{min})^2}} \geq \tau$$

at all layers. If $L >> \lambda$, then the square root reduces to $\frac{r_{max}}{r_{min}}$ and we find the requirement,

$$\frac{r_{min}^2}{r_{max}^2} = \frac{\delta_z^2}{2L^2 + \delta_z^2} \gtrsim \tau \quad (13)$$

Note that the left hand side of equation 13 is maximal in the limit as the inter-plane spacing δ_z approaches infinity. This agrees with our intuition that as the planes get further apart, light leaving from any point at one plane will reach any point on the adjacent plane.

4 Backpropagation

4.1 Loss Function

In this work we train the diffractive network against the cross-entropy loss. Let $\mathbf{x}^{(j)} \in \mathbb{C}^{M^2\times 1}$ be a training sample of the input field $\boldsymbol{\psi}_1^{(-)}$ with one-hot class label vector $\mathbf{y}^{(j)} \in \mathbb{R}^{K\times 1}$ where we are using the superscript (j) to indicate the training sample index. Feeding $\mathbf{x}^{(j)}$ through the network results in the class probabilities $\mathbf{p}^{(j)} \in \mathbb{R}^{K\times 1}$. The cross-entropy loss is given by equation 14.

$$\mathcal{L}^{(j)} = -\sum_{k=1}^{M} y_k^{(j)} \log p_k^{(j)} = -\log p_{k_j}^{(j)} \quad (14)$$

where k_j is the index of the non-zero element in the one-hot label. We minimize the loss with respect to the phase parameters $\boldsymbol{\phi}_\ell$ by running gradient descent. The derivative of the cross-entropy loss for probabilities computed with the softmax function is [2],

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = (\mathbf{p} - \mathbf{y})^T$$

4.2 Gradient Descent

In the gradient descent algorithm we iteratively update our parameters of interest so as to minimize the loss via

$$\boldsymbol{\phi}_\ell^{[i+1]} = \boldsymbol{\phi}_\ell^{[i]} - \alpha \frac{\partial \mathcal{L}}{\partial \boldsymbol{\phi}_\ell}\bigg|_{\boldsymbol{\phi}_\ell^{[i]}} \tag{15}$$

where $\alpha > 0$ is the (tunable) learning rate and superscript $[i]$ denotes the iteration index. The derivatives of the loss with respect to the phase parameters $\boldsymbol{\phi}_\ell$ can be computed using the backpropagation algorithm. The initial partial derivatives are given in equation 16.

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\phi}_\ell} = \frac{\partial \mathcal{L}}{\partial \mathbf{v}} \frac{\partial \mathbf{v}}{\partial |\boldsymbol{\psi}_D|^2} \frac{\partial |\boldsymbol{\psi}_D|^2}{\partial \boldsymbol{\phi}_\ell} \tag{16}$$

where we have elected to exclusively consider partial derivatives involving real quantities only. Performing the chain-rule for intermediate derivatives involves complex differentiation which we can avoid for the purposes of this work.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{v}} = (\mathbf{p} - \mathbf{y})^T \tag{17a}$$

$$\frac{\partial \mathbf{v}}{\partial |\boldsymbol{\psi}_D|^2} = S \tag{17b}$$

$$\frac{\partial |\boldsymbol{\psi}_D|^2}{\partial \boldsymbol{\phi}_\ell} = 2\,\mathrm{Re}\Big\{i\ \mathtt{diag}(\boldsymbol{\psi}_D^*)\, P_{\ell\to N}\, \mathtt{diag}(\boldsymbol{\psi}_\ell^{(-)})\Big\} \tag{17c}$$

where we have defined the intermediate propagator from plane ℓ to plane N (for $\ell \le N$) as

$$P_{\ell\to N} = H_N \Lambda_N \cdots H_\ell \Lambda_\ell \tag{18a}$$

$$\boldsymbol{\psi}_D = P_{\ell\to N} \boldsymbol{\psi}_\ell^{(-)} \tag{18b}$$

The equations in 17 are each derived in appendix A. Note that the phase in equation 17c appears implicitly through the Λ_ℓ term. We may further reduce equation 17c using the property of diagonal matrices acting to the left and right

$$\begin{aligned}\frac{\partial |\boldsymbol{\psi}_D|^2}{\partial \boldsymbol{\phi}_\ell} &= 2\,\mathrm{Re}\Big\{i\ \boldsymbol{\psi}_D^* \odot P_{\ell\to N} \odot \boldsymbol{\psi}_\ell^{T(-)}\Big\} \\ &= 2\,\mathrm{Re}\Big\{i\boldsymbol{\psi}_D^* \boldsymbol{\psi}_\ell^{T(-)} \odot P_{\ell\to N}\Big\}\end{aligned}$$

Now, we look to combine all of the partial derivatives to get a closed-form for the gradient of the phase. First let us define $\mathbf{s}^T = (\mathbf{p} - \mathbf{y})^T S \in \mathbb{R}^{M^2}$. Then, the complete gradient of the loss with respect to the phase at layer ℓ is,

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{\phi}_\ell} = 2\mathbf{s}^T\,\mathrm{Re}\Big\{i\boldsymbol{\psi}_D^* \boldsymbol{\psi}_\ell^{T(-)} \odot P_{\ell\to N}\Big\} \tag{19}$$

$$= 2\,\mathrm{Re}\Big\{i\ \mathbf{s}^T\Big[\boldsymbol{\psi}_D^* \boldsymbol{\psi}_\ell^{T(-)} \odot P_{\ell\to N}\Big]\Big\} \tag{20}$$

$$\left[\frac{\partial \mathcal{L}}{\partial \boldsymbol{\phi}_\ell}\right]_j = 2\,\mathrm{Re}\left\{i \sum_k s[k]\psi_D^*[k]\psi_\ell^{(-)}[j] P_{\ell\to N}[k,j]\right\} \tag{21}$$

$$= 2\,\mathrm{Re}\left\{i\psi_\ell^{(-)}[j] \sum_k s[k]\psi_D^*[k] P_{\ell\to N}[k,j]\right\} \tag{22}$$

$$\implies \frac{\partial \mathcal{L}}{\partial \boldsymbol{\phi}_\ell} = 2\,\mathrm{Re}\Big\{i\boldsymbol{\psi}_\ell^{T(-)} \odot \Big(\mathbf{q}^T P_{\ell\to N}\Big)\Big\} \tag{23}$$

where in the last line we have defined $\mathbf{q} = \mathbf{s} \odot \boldsymbol{\psi}_D^*$. Examining the term in parenthesis we see that this appears to be something like the propagator acting on an input $\Big(\mathbf{q}^T P_{\ell\to N}\Big)^T = P_{\ell\to N}^T \mathbf{q} = \Lambda_\ell H_\ell^T \cdots \Lambda_N H_N^T \mathbf{q}$, however now we have the transpose of the Toeplitz matrices acting on the vector $\mathbf{q}$. It turns out that the convolution matrices $H_\ell^T = H_\ell$ are symmetric (See derivation in Appendix D). This allows us to write the derivative purely in terms of convolutions which drastically reduces space-complexity (i.e. we don't need to store enormous Toeplitz matrices). Therefore, we find that the complete derivative can be written as,

$$\begin{aligned}\frac{\partial \mathcal{L}}{\partial \boldsymbol{\phi}_\ell} &= 2\,\mathrm{Re}\Big\{i\boldsymbol{\psi}_\ell^{T(-)} \odot \Big(\Lambda_\ell H_\ell \cdots \Lambda_N H_N \mathbf{q})^T\Big\} \\ &= \mathtt{vec}\Big[2\ \mathrm{Re}\Big\{i\psi_\ell^{(-)}\Big[e^{i\phi_\ell}\Big(h_\ell * \Big[\cdots e^{i\phi_N}\Big(h_N * q\Big)\cdots\Big]\Big)\Big]\Big\}\Big]\end{aligned}$$

The advantage of this representation is that we have defined the complete derivative entirely in terms of convolutions. This allows us to forgo storing matrices of dimensions $M^2 \times M^2$ during the forward step which can be computationally prohibitive when the phase planes have large dimensionality.

5 Optimal Detector Segmentation

Recall that we define the cumulative intensity in a detector segment associate with class k as

$$v_k = \mathbf{s}_k^T |\boldsymbol{\psi}_D|^2 \tag{24}$$

where $\mathbf{s}_k^T$ is an indicator vector for the pixels in the segmented detector region associated with class k. This requires an ad-hoc prescription of the detector regions which places an unnecessary constraint on the network. We may instead choose to learn the detector partitions along with the phase masks. To do so, we relax the constraint that the vectors $\mathbf{s}_k^T$ have to be non-overlapping indicator vectors (i.e. composed of only ones and zeros). Instead, we let them be arbitrary real weighting functions over the detector intensity field. Note that these weighting functions

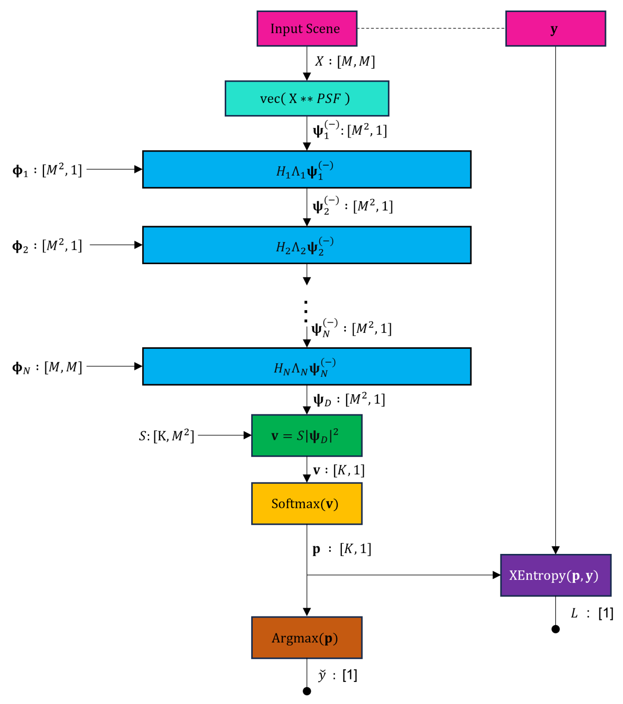

Figure 3. A block diagram showing the optical feed-forward neural network architecture. The field incident on each phase plane is transformed through the blue blocks which represent a point-wise phase modulation followed by a diffraction propagator. The remaining blocks handle the classification. We train our model to minimize the cross-entropy loss with respect to the training labels.

may be positive or negative and they need not be normalized in any way. Physically this amounts to introducing a positive (or negative) gain on the intensity measured at each pixel and summing their individual contributions. To perform this optimization we need to know the derivative of the loss with respect to the matrix of weights S,

$$\frac{\partial \mathcal{L}}{\partial S} = \frac{\partial \mathcal{L}}{\partial \mathbf{v}}\frac{\partial \mathbf{v}}{\partial S} \tag{25}$$

$$\frac{\partial \mathbf{v}}{\partial S} = I_K \otimes |\boldsymbol{\psi}_D|^2 \tag{26}$$

where $\otimes$ represents the tensor product and I_K is the $K \times K$ identity matrix. In total we have,

$$\frac{\partial \mathcal{L}}{\partial S} = (\mathbf{p} - \mathbf{y})^T \otimes |\boldsymbol{\psi}_D|^2 \tag{27}$$

6 Spatial Mode-Sorting Classifiers

Suppose we have an imaging system with point-spread function (PSF) given by $\Psi_0(x, y)$. The field at the image plane is found by convolving the PSF with the object field $X^{(j)}(x, y)$

$$\Psi^{(j)}(x, y) = \int\int dx'dy'\ \Psi_0(x - x', y - y')F^{(j)}(x', y')$$

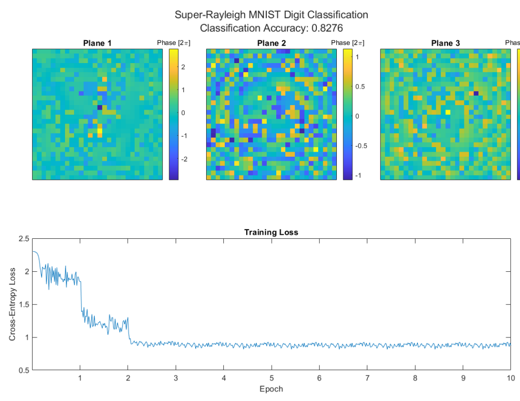

Figure 4. The learned phase masks for a 3-element system and the loss convergence over the training cycles. Each phase mask had square length of $L = 8\mu m$ and was discretized into 28×28 pixels (corresponding to the dimensionality of the MNIST dataset). The distance between adjacent planes planes was $\delta_z = 8\mu m$ giving a connectivity factor of $\frac{1}{3}$. The wavelength of light was $\lambda = 500nm$.

Consider measuring the received field by demultiplexing into the orthonormal spatial modes $\{\phi_m(x, y)\}$. The probability of detecting a photon in a given mode is

$$p_m^{(j)} = |\left\langle \phi_m \middle| \Psi^{(j)}(x, y) \right\rangle|^2$$

Assuming a shot-noise limited measurement, we may express the number of photons counted in each modal bin as a a random measurement vector with entry-wise Poisson statistics.

$$\mathbf{n}^{(j)} \sim \texttt{Poiss}(\bar{N}\mathbf{p}^{(j)})$$

After normalizing the measurements, samples of $\check{\mathbf{n}}$ becomes that transformed input data that we use to train the neural/ network.

$$\check{\mathbf{n}}^{(j)} = \mathbf{n}^{(j)} / \sum_m n_m^{(j)}$$

7 Simulated Results

7.1 Pre-Processing MNIST Dataset

The MNIST image dimensions were resized from 28×28 pixels using nearest-neighbor interpolation to occupy a certain number of rayleigh lengths for our optical system. That is the width of the image w at the input of the 4f system is $w = \alpha\sigma$ where $\sigma = 1.2197\lambda/D$. The images were then rotated the images by 180° and convolved with the point spread function

$$PSF(r) = \frac{J_1\left(\frac{2\pi R}{\lambda f} r\right)}{\frac{2\pi R}{\lambda f} r}$$

where J_1 is the Bessel function of the first-kind, R is the radius of the pinhole, f is the focal length of the imaging system and λ is the wavelength.

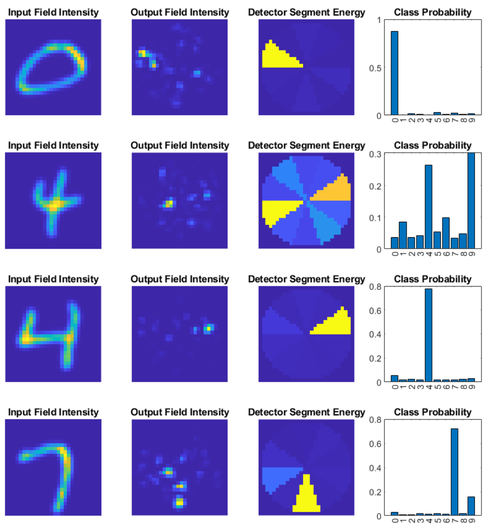

Figure 5. Examples of super-Rayleigh inputs (col. 1), detector outputs (col. 2), detector segment energies, (col. 3), and associated class probabilities (col. 4) for example test scenes. Note that the two test images for digit '4' (row 2,3) have substantially different class probability distributions. Row 2 is classified weakly as digit '9' while row 3 is classified strongly as digit '4'. We suspect that this is fundamentally because the forward model for a ONN is linear. Therefore, while humans perceptually attend to small discontinuities in the digits when classifying, the ONN model makes little distinction between shapes with minor differences in discontinuity. As a conceptual example, consider the perceptual difference between the numbers 8 and 9 in a seven-segment display (digital clock display). The 'distance' between these numbers in the display is a single line segment. While perceptually this line-segment makes a profound difference in our classification, the small 'distance' between the two numbers is (in proportion to the matrix determinant) conserved through a linear map. In the special case of a unitary map, the distances of the output are exactly equal to the distances of the input. In row 2 for instance, a small line connecting the tips of the '4' would perceptually convert the digit into a '9'. Conversely, removing the top-most edge of the '9' would perceptually convert the digit into a '4'. However the linearity of the forward model makes little distinction between presence or absence of this line segment.

8 Experimental Results

Surface	$z_{\ell+1} - z_\ell$ [mm]	Connectivity τ
PM1	-	-
PM2	556	1
PM3	556	1
Detector	850	1

Experimental Variables

1. wavelength 532 nm
2. 4f-System focal length 200 mm
3. aperture diameter 400 μm
4. SLM phase mask dimensions 300×300
5. SLM pixel pitch 8 μm
6. EMCCD dimensions
7. EMCCD pixel pitch 16 μm

9 Diffractive Neural Networks with Non-Linear χ^3 Layers

Other non-linear ONNs [?]

10 Quantum Chernoff Bound

A Backpropagation Derivatives

Here we derive equations in 17. The first two derivatives are trivial. The derivation for $\frac{\partial \mathcal{L}}{\partial \mathbf{v}}$ can be found in [2] while the derivation for $\frac{\partial \mathbf{v}}{\partial |\psi_D|^2}$ can be found in the Matrix Cookbook [3]. The third derivative $\frac{\partial |\psi_D|^2}{\phi_\ell}$ will be our focus here. We analyze the equation by components and then interpret the result to get a matrix-vector form.

$$\begin{aligned}\frac{\partial |\psi_D[p]|^2}{\phi_\ell[q]} &= \psi_D^*[p]\frac{\partial \psi_D[p]}{\partial \phi_\ell[q]} + \psi_D[p]\frac{\partial \psi_D^*[p]}{\partial \phi_\ell[q]} \\ &= 2\,\mathrm{Re}\left\{\psi_D^*[p]\frac{\partial \psi_D[p]}{\partial \phi_\ell[q]}\right\} \\ \psi_D[p] &= \sum_k P_{\ell\to N}[p,k]\psi_\ell^{(-)}[k] \\ &= \sum_k P_{\ell^{(+)}\to N}[p,k]e^{i\phi_\ell[k]}\psi_\ell^{(-)}[k]\end{aligned}$$

where we have defined $P_{\ell^{(+)}\to N} = H_N\Lambda_N\cdots H_\ell$ where the final phase operator Λ_ℓ has been omitted.

$$\begin{aligned}\implies \frac{\partial \psi_D[p]}{\partial \phi_\ell[q]} &= \sum_k P_{\ell^{(+)}\to N}[p,k]\frac{\partial e^{i\phi_\ell[k]}}{\partial \phi_\ell[q]}\psi_\ell^{(-)}[k] \\ &= \sum_k iP_{\ell^{(+)}\to N}[p,k]\delta_{kq}e^{i\phi_\ell[k]}\psi_\ell^{(-)}[k] \\ &= iP_{\ell^{(+)}\to N}[p,q]e^{i\phi_\ell[q]}\psi_\ell^{(-)}[q] \\ &= iP_{\ell\to N}[p,q]\psi_\ell^{(-)}[q]\end{aligned}$$

Therefore, the complete expression inside the Re operation is given by,

$$\psi_D^*[p]\frac{\partial \psi_D[p]}{\partial \phi_\ell[q]} = i\psi_D^*[p]P_{\ell\to N}[p,q]\psi_\ell^{(-)}[q]$$

We see that every row in the matrix $P_{\ell\to N}$ is multiplied by the entries in ψ_D^* while every column in the same matrix is multiplied by the entries in $\psi_\ell^{(-)}$. Hence we can drop the indices and compactly write the derivative,

$$\psi_D^*\frac{\partial \psi_D}{\partial \phi_\ell} = i\mathtt{diag}(\psi_D^*)P_{\ell\to N}\mathtt{diag}(\psi_\ell^{(-)})$$

In total we arrive at equation 17c

$$\frac{\partial |\psi_D|^2}{\partial \phi_\ell} = 2\,\mathrm{Re}\left\{i\mathtt{diag}(\psi_D^*)P_{\ell\to N}\mathtt{diag}(\psi_\ell^{(-)})\right\}$$

All of the partial derivatives are real, as required.

B Complex Differentiation

Implementing traditional back-propagation involves computing intermediate gradients for each operation in the forward model. In this system, all of the fields and the phase masks are complex-valued vectors. Hence we must invoke the notion of a complex derivative. For a complex number $z = x + iy$, the Cauchy-Riemann Theorem states that the derivative df/dz of a complex function $f(z) = u(x, y) + iv(x, y)$ exists if and only if its real and imaginary components are continuously differentiable and satisfy the equations

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \tag{28}$$

$$\frac{\partial u}{\partial y} = -\frac{\partial v}{\partial x} \tag{29}$$

If the derivative indeed exists, then it may be computed using,

$$f'(z) = u_x + iv_x = v_y - iu_y$$

Differentiation with respect to functions of complex vectors and matrices can be found in [4]. Interestingly, the complex derivative $\frac{d}{dz}f(z) = \frac{d}{dz}(z^* z)$ does not exist for $z \neq 0$. This suggests that we have no way of computing $\partial(\psi_D^* \psi_D)/\partial\psi_D$ as an intermediate derivative. For this reason, in equation 16 we consider the chain-rule expansion of gradients for real-valued quantities only.

C Tensor Contraction for Backpropagating Derivatives

Let A be an array of dimension $[a_1, \dots, a_n]$ and B be an array of dimensions $[b_1, \dots, b_m]$. The derivative $\partial A/\partial B$ is another array of dimension $[a_1, \dots, a_n, b_1, \dots, b_m]$ as every element in A must be differentiated with respect to every element in B. In general we will by convention set the first dimensions of the derivative array to be equal to those of the numerator in the derivative, and the last dimensions to be equal to those of the denominator. Backpropagation involves combining multiple partial derivatives as prescribed by the chain rule. To do so, implementing tensor contraction in an array-based manner is central to efficiently computing derivatives. In general, we must contract (at most) two tensors at a time to compute gradients. Here will we present the general method for doing this. Suppose we wish to compute

$$\frac{\partial A}{\partial C} = \frac{\partial A}{\partial B}\frac{\partial B}{\partial C}$$

It is ambiguous from the notation above how we are supposed to combine elements in the right hand side in order to compute the derivative tensor on the left hand side. It is revealing to look at the dimensionalities of the pieces in the equation.

$$\frac{\partial A}{\partial C} \to [a_1, \dots, a_n, c_1, \dots, c_s]$$
$$\frac{\partial A}{\partial B} \to [a_1, \dots, a_n, b_1, \dots, b_m]$$
$$\frac{\partial B}{\partial C} \to [b_1, \dots, b_m, c_1, \dots, c_s]$$

We can now imagine promoting either array to a shared higher-dimensional space by padding the end of the dimensions of $\frac{\partial A}{\partial B}$ by s singletons and pushing the dimensions of $\frac{\partial B}{\partial C}$ to the right by n singletons.

Pad Dimensionality by s

$$\frac{\partial A}{\partial B} \to [a_1, \dots, a_n, b_1, \dots, b_m, 1 \dots, 1]$$

Push Dimensionality by n

$$\frac{\partial B}{\partial C} \to [1, \dots, 1, b_1, \dots, b_m, c_1, \dots, c_s]$$

Contracting the tensors then simply amounts to taking their point-wise multiplication and summing over the shared dimensions $n + 1$ to $n + m$ which have sizes equal to the b_i's.

$$\frac{\partial A}{\partial C} = \mathtt{sum}\left(\frac{\partial A}{\partial B} \odot \frac{\partial B}{\partial C}, [n + 1, n + m]\right)$$

This is a simple algorithm for contraction of tensors involved in a chain rule computation. A more general method for contracting any two tensors requires slightly more bookkeeping as it is not obvious which dimensions are meant to be matched.

D Convolution Toeplitz Matrices are Anti-Symmetric for the Rayleigh-Sommerfeld Kernel

WLOG, the discretized convolution kernel that takes the field between sequential layers 1 and 2 is

$$h[n_1, m_1; n_2, m_2] = \frac{z_2 - z_1}{r^2}\left(\frac{1}{2\pi r} + \frac{1}{i\lambda}\right)\exp(ikr)$$
$$r = \Delta\sqrt{(n_2 - n_1)^2 + (m_2 - m_1)^2 + ((z_2 - z_1)/\Delta)^2}$$

where Δ is the pixel pitch. To define the convolution operation as a Toeplitz matrix, we linearize the indices $j_1 = (n_1, m_1)$ and $j_2 = (n_2, m_2)$ so that

$$H_{j_1 j_2} = h[n_1, m_1; n_2, m_2]$$

Therefore, the matrix elements of the transpose of the Toeplitz matrix are,

$$\begin{aligned} H^T_{j_1 j_2} = H_{j_2 j_1} &= h[n_2, m_2; n_1, m_1] \\ &= h[n_1, m_1; n_2, m_2] = H_{j_1 j_2} \end{aligned}$$

Note that exchanging the arguments of the kernel h has no effect on the value of the kernel since r is independent of whether the differences $(n_2 - n_1)$, $(m_2 - m_1)$ are positive or negative.

References

[1] X. Lin, Y. Rivenson, N.T. Yardimci, M. Veli, Y. Luo, M. Jarrahi, A. Ozcan, Science **361**, 1004 (2018), `https://www.science.org/doi/pdf/10.1126/science.aat8084`

[2] T. Kurbiel, *Derivative of the softmax function and the categorical cross-entropy loss* (2021), `https://towardsdatascience.com/derivative-of-the-softmax-function-and-the-categori`

[3] K.B. Petersen, M.S. Pedersen, *The matrix cookbook* (2012), version 20121115, `http://www2.compute.dtu.dk/pubdb/pubs/3274-full.html`

[4] A. Hjorungnes, D. Gesbert, IEEE Transactions on Signal Processing **55**, 2740 (2007)