Optics EXPRESS

Multi-directional beam steering using diffractive neural networks

I. U. IDEHENRE[1,2] AND M. S. MILLS[1,*]

[1]Air Force Research Laboratory, 2179 12th St., Wright-Patterson Air Force Base, OH 45433-7718, USA
[2]Azimuth Corporation, 2970 Presidential Dr. # 200, Fairborn, OH 45324, USA
[*]matthew.mills.25@us.af.mil

Abstract: The modern-day resurgence of machine learning has encouraged researchers to revisit older problem spaces from a new perspective. One promising avenue has been implementing deep neural networks to aid in the simulation of physical systems. In the field of optics, densely connected neural networks able to mimic wave propagation have recently been constructed. These diffractive deep neural networks (D^{2}NN) not only offer new insights into wave propagation, but provide a novel tool for investigating and discovering multi-functional diffractive elements. In this paper, we derive an efficient GPU-friendly D^{2}NN methodology based on Rayleigh-Sommerfeld diffraction. We then use the implementation to virtually forge cascades of optical phase masks subject to different beam steering conditions. The input and output conditions we use to train each D^{2}NN instance is based on commercial electro-optic modulated waveguide systems to encourage experimental follow-on. In total, we analyze the beam steering efficacy of 27 individual D^{2}NN instances which explore different permutations of input sources, mask cascades, and output steering targets.

1. Introduction

The core of all machine learning (ML) techniques rely on the algorithmic parsing of data in order to uncover abstracted patterns. From the emergent models, future determinations or predictions can be evaluated, often with a surprising degree of both accuracy and generality. Under the umbrella of ML, many successful modalities have been developed over the years ranging from rule-based approaches [1,2], nonlinear regression methods [3], natural language processing algorithms [4,5], and knowledge-based representations [6,7], to agent-based computing [8,9] and perception systems based on ambient intelligence [10]. The recent resurgence in ML techniques can be attributed to modern advancements in large data collection infrastructures, cloud computing, and the serendipitous applicability of GPUs for massively parallel computation [11–13]. One of the fastest growing approaches has been deep learning [14]. Simply put, deep learning is a process of parameter fitting accomplished by nesting statistical function calls on raw data. If one repeatedly composes these transformations on initial data while constraining to a desired output (i.e. supervising), a multi-layered abstract representation correlating raw data to prescribed outputs can be obtained. Amazingly with deep learning, the resulting high-level data representation is automatically retrieved in the sense that the optimal transformations can be uncovered without requiring specific domain expertise. This fact is illustrated particularly well in a deep net auto-encoder [15–17] where image compression schemes can be uncovered without prior knowledge to the inner workings of standard image compression algorithms. The success of deep learning indubitably stems from this profound ability to abstract data, and has since demonstrated impressive results in the fields of image recognition [12,18], speech recognition [5,19], self-driving vehicles [20,21], and consumer tailored advertising [22,23]. The ramifications of deep learning are so ubiquitous that it is now virtually present in everyday modern life through computers, cameras, smartphones, and the Internet of Things [24]. By virtue of the scientific community, the workflow of designing a deep neural network is now streamlined. Open source

#400364
Journal © 2020
https://doi.org/10.1364/OE.400364
Received 25 Jun 2020; revised 7 Aug 2020; accepted 8 Aug 2020; published 18 Aug 2020

programs such as TensorFlow [25] and pyTorch [26], equip the researcher with a toolkit for seamlessly setting up deep network topologies while ensuring the most efficient methods for training.

The abstract data representations accreted by deep neural network training suggests harnessing them for physics based simulations. One can imagine implementing this functionality in two possible ways; the first is to procure a database of simulation results and train a network capable of guiding follow-up computations. An example of this can be found in studies requiring density functional theory. The numerical computations involved are expensive and time-consuming; the ability to pre-screen material classes of interest with deep neural networks greatly increases the chance of identifying an optimal material composition [27]. In a similar vein, the configuration space of possible optical metasurface compositions and structures can be effectively navigated through neural network advisory [28,29]. The second way is to teach a machine learning algorithm to mimic or aid the computational method itself. A prime example is in the study of partial differential equations which often describe the dynamics of physical processes. Convolutional neural networks, for instance, offer a means to aid in their numerical computation [30,31]. Such an ability has given a fresh perspective in studying famously intractable problem spaces such as fluid dynamics [32–34]. The first suggestion and experimentation for exploiting the field of optics to mimic neural network behavior worked with a system of light emitters/detectors and holograms to act as the nodes and edges of the network respectively [35–38]. This approach had the specific perspective of using light-matter interactions to "perform the computation" that a dense neural network on a computer would produce.

More recently, this perspective has been flipped. Due to an intriguing isomorphism between the Huygens-Fresnel Principle and the architecture of a dense neural network, the modern computational capabilities of deep neural networks can instead be harnessed to mimic wave propagation [39,40]. This so-called diffractive deep neural network (D^2NN) not only offers new insights into materials design but gives a novel tool for investigating multi-functional diffractive elements. In this paper, we derive an efficient D^2NN methodology and use it to design cascades of optical phase masks in order to perform multi-directional beam steering. The input and output training data is derived from a commercially based end-firing waveguide system to encourage experimental realization. We investigate in detail the effects of permuting multiple illumination sources, multiple cascaded phase masks, and multiple target output intensities. Our investigation provides a general protocol to follow when subjecting phase-based devices to multiple wave propagation constraints.

2. Methodology

2.1. Rayleigh-Sommerfeld diffraction through a cascade of phase masks

The Huygens-Fresnel principle is an analysis method for wave propagation valid for both the near and far field. The principle states that all spatial points on an existing primary wavefront can be treated as sources of secondary spherical radiators. The total superposition of these secondary wavelets properly determines the evolved wavefront at subsequent planes. This physical description can be isomorphic to a densely connected neural network under proper constraints. The transverse spatial discretization of the complex electric field, the phase evolution of the secondary spherical wavelets, the direction of propagation, the nonlinear effects and free EM sources, directly correspond to the number of neurons in a layer, connecting weights, number of layers in the network, activation functions, and bias values respectively (Fig. 1).

Mathematically, the wave propagation illustrated in Fig. 1(a) can be described by the Rayleigh-Sommerfeld diffraction integral [38]. The discrete form of this integral is commonly

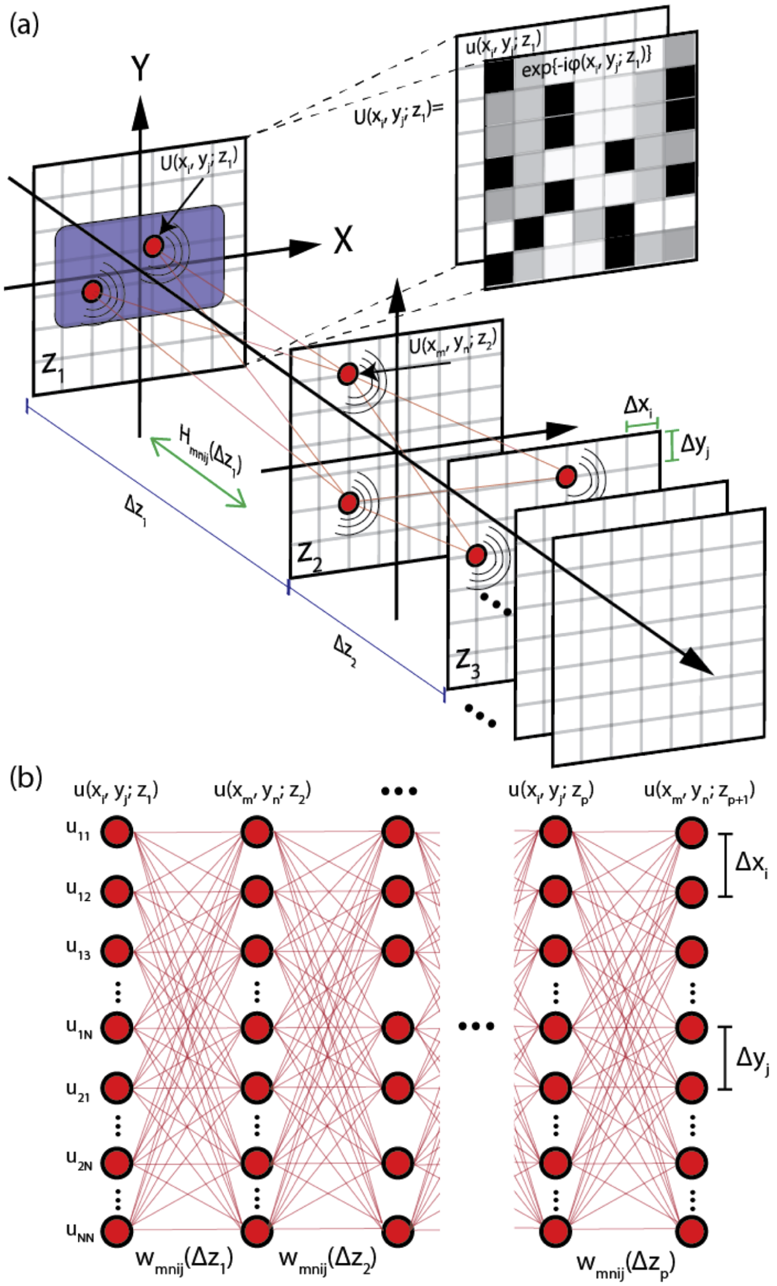

Fig. 1. The isomorphism between the Huygens-Fresnel principle and a densely connected neural network. (a) Every discrete point in an initial transverse plane acts as secondary spherical resonators. The evolution of the electric field at each point in a further plane is determined by the superposition of these contributions. (b) Every discrete amplitude in an initial transverse plane represents a neuron in a layer. The evolution of the electric field along with its initial phase value determines the weights which connect to a further layer. The neurons in this further layer are determined by the summation of these contributions.

used for its numerical computation:

$$\begin{aligned} U(x_m, y_n; z_{p+1}) &= \sum_{i}^{N}\sum_{j}^{N} H_{mnij}(\Delta z_p) U(x_i, y_j; z_p)\Delta x \Delta y \\ &= \sum_{i}^{N}\sum_{j}^{N} \left(\frac{e^{\hat{\imath} k r_{mnij}}}{2\pi r_{mnij}}\right)\left(\frac{\Delta z_p}{r_{mnij}}\right)\left(\frac{1}{r_{mnij}} - \hat{\imath} k\right) U(x_i, y_j; z_p)\Delta x \Delta y \end{aligned} \tag{1}$$

where $U(x_i, y_j; z_p) = u(x_i, y_j; z_p)\exp\{-\hat{\imath}\phi(x_i, y_j; z_p)\}$ is the total electric field located at a general propagation distance, z_p. For the purposes in this manuscript, we choose to describe this total field by a complex-valued field, $u(x_i, y_j; z_p)$, and a separated contribution from a thin phase-only mask element placed at z_p, $\exp\{-\hat{\imath}\phi(x_i, y_j; z_p)\}$. The total field is discretized in both transverse coordinates, $\{x_i, y_j\}$, by N points with resolution, $\{\Delta x, \Delta y\}$, and are tracked with object plane indices, $\{i, j\}$. The resolution, defined as $\Delta x = \Delta x_i = x_{i+1} - x_i$ and $\Delta y = \Delta y_j = y_{j+1} - y_j$, are constant in a given plane. Identical to the aforementioned situation, $U(x_m, y_n; z_{p+1}) = u(x_m, y_n; z_{p+1})\exp\{-\hat{\imath}\phi(x_m, y_n; z_{p+1})\}$ is the total electric field located at a further propagation distance, z_{p+1}, subject to another phase mask, $\exp\{-\hat{\imath}\phi(x_m, y_n; z_{p+1})\}$, discretized in both transverse coordinates, $\{x_m, y_n\}$, by N points tracked by image plane indices $\{m, n\}$. These two electric fields are connected by a transfer function, $H_{mnij}(\Delta z_p)$, which encodes the spherical phase accumulation of each secondary resonator during propagation. The transfer function is dependent on the radial distance between points in the z_p and the z_{p+1} planes and is given by $r_{mnij} = \sqrt{(x_i - x_m)^2 + (y_j - y_n)^2 + \Delta z_p^2}$ where $\Delta z_p = z_{p+1} - z_p$. Finally, $k = 2\pi n/\lambda$ is the wavenumber with wavelength, λ, refractive index of the medium, n, and $\hat{\imath} = \sqrt{-1}$ (script '$\hat{\imath}$' is used to avoid conflict with the iterator 'i'). When wanting to propagate further from the z_{p+1} to the z_{p+2} plane, the object and image plane indices can simply be re-assigned to the former and latter respectively.

2.2. D^{2}NN forward propagation isomorphism

If we reform Eq. (1) in a contrived way, we are able to draw a direct analogy between the Rayleigh-Sommerfeld description of diffraction and the mathematics of a dense neural network (Fig. 1(b)):

$$\begin{aligned} u(x_m, y_n; z_{p+1}) &= f\left(\sum_{i}^{N}\sum_{j}^{N} w_{mnij}(\Delta z_p) u(x_i, y_j; z_p) + b_{ij}(z_p)\right) \\ w_{mnij}(\Delta z_p) &= e^{-\hat{\imath}\phi(x_i, y_j; z_p)} \times \left(\frac{e^{\hat{\imath} k r_{mnij}}}{2\pi r_{mnij}}\right)\left(\frac{\Delta z_p}{r_{mnij}}\right)\left(\frac{1}{r_{mnij}} - \hat{\imath} k\right)\Delta x \Delta y. \end{aligned} \tag{2}$$

When flattened, the two-dimensional complex-valued field without the phase mask contribution, $u(x_i, y_j; z_p)$, represents the nodes in the p^{th} layer of a dense neural network while the weight factor, $w_{mnij}(\Delta z_p)$, represents the edge values which connect nodes between layers p and $p+1$. Note that the weight factor, as defined, accounts for the phase mask contribution at z_p, the transverse coordinate discretization, and the Rayleigh-Sommerfeld diffraction kernel. A bias term, $b_{ij}(z_p)$, is added to the weighted sums and then wrapped in the activation function, f. In the propagation picture, the bias term represents contributions to the field from current sources and the activation function can accompany nonlinear effects. For our purposes, we assume no current sources, $b_{ij}(z_p) = 0$ and a linear activation function, $f = x$. Computing Eq. (2) results in the complex electric field, $u(x_m, y_n; z_{p+1})$, at the further propagation distance, z_{p+1}. The new two-dimensional complex amplitude is flattened and stored as the node values at the $p+1$ layer. This process is then iterated until the final image plane is obtained. Note that unlike standard neural networks, the nodes and weight factors are complex numbers which cannot be arbitrarily adjusted. Instead, they contain physical meaning defined by contributions from the Rayleigh-Sommerfeld diffraction kernel, the phase mask element located at z_p, and the physical spacing of the discrete sampling. For this reason, the adjustment of weights during training must be done in accordance with the physics and can be interpreted as changes in wavelength, refractive index, Δz, or phase mask pixel heights depending on which of these are locked constant in the program.

There are several ways to solve Eq. (2) including direct evaluation. However, a number of Fast Fourier Transform (FFT) based approaches have been developed that are accurate, less resource intensive, and most importantly faster than a direct approach. These FFT methods fall into two main categories: the angular spectrum methods (FFT-AS) [38,41,42] and the direct integration (FFT-DI) [41,42] methods. The latter is the approach implemented in this manuscript, and takes advantage of the fact that for sufficiently large problems, convolution is more efficiently and easily done in the frequency domain. The incompatibility between the cyclic convolution performed by FFT-DI and the linear convolution of Rayleigh-Sommerfeld kernel is resolved by zero-padding the boundaries of the input field, $u(x_i, y_j; z_p)$ [41]. We note that although FFT-DI is used in this publication, any approach such as FFT-AS that solves the Rayleigh-Sommerfeld diffraction integral is also suitable for use.

2.3. D^{2}NN backpropagation isomorphism

Equipped with the D^{2}NN forward propagation method of section 2.2, one can now compute the evolution of an electric field through a series of planes each subject to a thin phase mask element in a format compatible with deep learning. The next essential step is to provide an iterative procedure which constrains an input field to a desired output intensity. This constraint will lead to an iterative adjustment of the layer weights; if the wavelength, refractive index, and spacing parameters are held constant, changes in the layer weights can be correlated to adjustments in the pixel-by-pixel values of the phase masks at each layer. In this way, the phase delays imposed by the phase masks can be optimized. In general, this procedure is able to accompany multiple inputs fields connected to multiple output target intensities. Each input and target output pair become the training data set fed into the D^{2}NN.

The iterative procedure to be defined in our context is known as backpropagation. Succinctly put, backpropagation is an algorithmic process which computes the multi-dimensional gradient of a neural network loss function with respect to the layer weights. In the case of D^{2}NN, dictated by Eq. (2), the gradients are evaluated with respect to the phases introduced by the masks at each layer. We define a training set of size Q where each $q \in Q$ contains an input field, $u_q(x_i, y_j; z_1)$, and a corresponding target output intensity, $\bar{I}_q(x_i, y_j; z_{\mathscr{P}})$. This notation is meant to elucidate that the input field is set at an initial $z_1 = 0$ plane and is computed through a finite amount of layers, $\{z_1, z_2, z_3, \ldots, z_{\mathscr{P}}\}$, with the final field occurring at $z_p = z_{\mathscr{P}}$. Thus, the input field in conjunction with Eq. (2) is used to obtain the actual output intensity at $z_{\mathscr{P}}$, $I_q(x_i, y_j; z_{\mathscr{P}}) = |u_q(x_i, y_j; z_{\mathscr{P}})|^2$. For each data pair in Q, this actual output intensity will differ from the supervised target output intensity. The pixel-by-pixel error is given by:

$$\varepsilon_{qij} = I_q(x_i, y_j; z_{\mathscr{P}}) - \bar{I}_q(x_i, y_j; z_{\mathscr{P}}). \tag{3}$$

The neural network loss function to be minimized can then be defined as the mean squared error (MSE) of these pixel-by-pixel errors summed over each of the training data pairs:

$$MSE = \frac{1}{QN^2} \sum_{q}^{Q} \sum_{i}^{N} \sum_{j}^{N} \varepsilon_{qij}^2. \tag{4}$$

We seek phase mask functions at each plane prior to the output plane which minimize Eq. (4):

$$\{\phi(x_i, y_j; z_1), \phi(x_i, y_j; z_2), \ldots, \phi(x_i, y_j; z_{\mathscr{P}-1})\} = \arg\min(MSE). \tag{5}$$

Because of the formalism in use, these phase mask values can be obtained via gradient descent as outlined in Ref [39]. The derivative of each and every phase mask pixel can be derived with respect to the MSE:

$$\frac{dMSE}{d\phi(x_s,y_t;z_v)} = \frac{4}{QN^2}\sum_q^Q\sum_i^N\sum_j^N \varepsilon_{qij}\Re\left(u_q^*(x_i,y_j;z_{\mathscr{P}})\frac{du_q(x_i,y_j;z_{\mathscr{P}})}{d\phi(x_s,y_t;z_v)}\right). \tag{6}$$

Where the dummy subscripts, (s,t,v), are able to mirror any of the values of the subscripts (i,j,p) can. All terms in Eq. (6), except $du_q(x_i,y_j;z_{\mathscr{P}})/d\phi(x_s,y_t;z_v)$, are computed and stored during the forward propagation algorithm. The remaining unknown term, the derivative of the computed output field with respect to any particular phase mask pixel, can be readily obtained by modifying the weights in the forward propagation algorithm with an appropriate aperture function, $a_{stij}(z_v,z_p)$:

$$\begin{gathered}\widetilde{u}(x_m,y_n;z_{p+1}) = \widetilde{f}\left(\sum_i^N\sum_j^N \widetilde{w}_{mnij}(\Delta z_p)\widetilde{u}(x_i,y_j;z_p)+\widetilde{b}_{ij}(z_p)\right)\\ \widetilde{w}_{mnij}(\Delta z_p) = a_{stij}(z_v,z_p)w_{mnij}(\Delta z_p)\\ a_{stij}(z_v,z_p) = \begin{cases}-j & x_i = x_s, y_j = y_t, z_p = z_v\\ 0 & x_i \neq x_s, y_j \neq y_t, z_p = z_v\\ 1 & z_p \neq z_v\end{cases}.\end{gathered} \tag{7}$$

Again we assume that the medium is source free, $\widetilde{b}_{ij}(z_p) = 0$, and undergoes linear propagation, $\widetilde{f} = x$. This aperture function incorporates effect of differentiating the Rayleigh-Sommerfeld with respect to the parameter $\phi(x_s,y_t;z_v)$ as discussed in the supplemental material of Ref. [39]. Using the modified weight function, Eq. (7) can be used to calculate the derivatives of the field where $\widetilde{u}(x_m,y_n;z_P) = du_q(x_i,y_j;z_P)/d\phi(x_s,y_t;z_v)$ and thus calculate the gradient for all phase parameters in any plane. Because we have maintained the same formalism as Eq. (2), Eq. (7) can also be solved using FFT based techniques (specifically FFT-DI). The difference is that now in order to calculate the gradients, a large number of FFT-DI calculations must be performed; that is, one must adjust the aperture function based on the each phase parameter $\phi(x_s,y_t;z_v)$ and re-compute. However, it should be noted that the derivative calculations are independent from one another and can be performed in parallel. To maximize speed, all calculations of the gradients were performed in parallel using Nvidia RTX-2080Ti GPUs.

3. Problem setup for multi-directional beam steering

3.1. Input source model

The procedures described in the section 2 create a cascade of phase-masks such that an arbitrary number of input and output intensity patterns can be connected after propagation. Using a similar protocol, D^{2}NNs were first demonstrated in the context of digit classification [39,43]. The networks essentially categorized input electric fields with intensity patterns resembling handwritten digits. Each input field was supervised to an output intensity bin representing the digits 0-9. Since the output field intensity for each number was assigned a unique non-overlapping region of space, the output field intensity could be used to interpret the input with use of detectors. Such an approach needs to be modified to work in the contexts of beam steering. To do so, we must prudently provide training data in the form of input fields with corresponding steering output intensity states. Additionally, we must stage an experimental scenario where switching between different input fields can be optically fast.

We propose the use of an end-firing waveguide optical phased array [44] shown in Fig. 2 which allows for nanosecond time scale switching. The device is a one-dimensional array of waveguides where each component can be electro-optically modulated (EOM) on an integrated platform. Each port is approximated as a point source with equal amplitude at a plane located at z_0 with a phase delay of $\phi_q(\chi_\ell, \psi_\ell; z_0)$ corresponding to the ℓ^{th} port at location $\{\chi_\ell, \psi_\ell\}$. In this way, the output from the array can be modeled using using the Rayleigh-Sommerfeld approach introduced in Eq. (1) where the phase delay of each port is analogous to a pixel on an adjustable phase mask. The accuracy of treating each port like a point source is contingent on the distance Δz_1 and the width of the port w_{port} being such that the fresnel number $N_F = w_{port}^2/\lambda\Delta z_1$ is significantly less than unity (i.e. in the far field) and the Rayleigh resolution limit $r_{limit} = 1.22\lambda\Delta z_1/N\Delta x$ is slightly below waveguide port size (i.e. $r_{limit} \lessapprox w_{port}$) [38]. If such a condition is met, the electric field at the end of the EOM ports $u_q(x_i, y_j; z_0)$ is given by the formula:

$$\begin{aligned} u_q(x_i, y_j; z_0) &= A_q \sum_{\ell}^{N_\ell} \delta_{x_i, y_j}^{\chi_\ell, \psi_\ell} \\ w_{q,mnij}(\Delta z_p) &= e^{-\hat{\imath}\phi_q(x_i, y_j; z_p)} \times \left(\frac{e^{\hat{\imath}kr_{mnij}}}{2\pi r_{mnij}}\right)\left(\frac{\Delta z_p}{r_{mnij}}\right)\left(\frac{1}{r_{mnij}} - \hat{\imath}k\right)\Delta x \Delta y \\ \delta_{x_i, y_j}^{\chi_\ell, \psi_\ell} &= \begin{cases} 1 & x_i = \chi_\ell, y_j = \psi_\ell \\ 0 & otherwise \end{cases} \end{aligned} \tag{8}$$

where A_α is a normalization factor to be determined, N_ℓ is the total number of ports, and $w_{q,mnij}(\Delta z_p)$ is weight factor for the q^{th} target. The weight factor $w_{q,mnij}(\Delta z_p)$ is used in place of $w_{mnij}(\Delta z_p)$ in Eq. (2) to calculate the field $u_q(x_m, y_n; z_1)$.

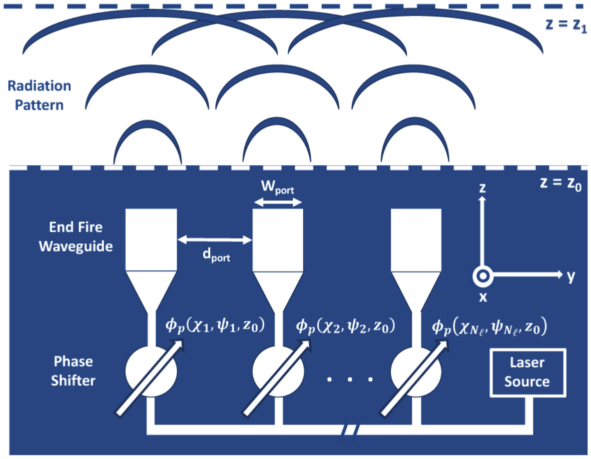
z = z₁
Radiation Pattern
z = z₀
W_port
End Fire Waveguide
d_port
z
y
x
$\phi_p(\chi_1, \psi_1, z_0)$
$\phi_p(\chi_2, \psi_2, z_0)$
$\phi_p(\chi_{N_\ell}, \psi_{N_\ell}, z_0)$
Phase Shifter
. . .
Laser Source

Fig. 2. Basic scheme of end-firing waveguide electro-optic modulated phased array. Under the correct conditions, each individual port acts as a point source at the plane, z_0. The phase delays of each port can be electro-optically adjusted leading to different input fields at z_1.

3.2. Input source optimization

Our main objective now is to select phase delay values associated with each port of the EOM array and for each target $\phi_q(\chi_\ell, \psi_\ell; z_0)$ so that the output field intensity of a given input/target pair $I_q(x_i, y_j; z_{\mathscr{P}})$ after passing through the D^2NN will be less likely to overlap with any other member of the dataset. In other words, we want to identify phase delays on each port such that the resulting set of generated input field intensities are as orthogonal as possible. We propose a simple method for selecting phase delays based on minimizing the overlap between the input field intensities $I_q(x_i, y_j; z_0)$. It can be inferred from the optimization algorithm, that when a significant amount of optical energy of the various input fields falls on the same pixel of a phase mask, they compete to influence the value of the phase delay. As discussed in Ref [45], a D^2NN performs poorly if the field amplitudes (and by extension the intensity) are not well separated physically. While the D^2NN can be configured to minimize the errors under such a condition, it is simply not possible to steer the optical energy of a given input towards the desired direction without a significant amount leaking into the regions assigned to other inputs. This effect, referred to as “ghosting", has been observed in fabricated D^2NN [39,43]. As such, the best approach is for the EOM array to generate fields $u_q(x_i, y_j; z_1)$ that have as little overlap in field intensity (or amplitude) as possible. In short, what we seek are values of $\phi_q(\chi_\ell, \psi_\ell; z_0)$ that minimize overlap between all possible combination of two separate input intensities. The overlap, measured via an overlap integral (OI) between intensities $I_\alpha(x_i, y_j; z_1)$ and $I_\beta(x_i, y_j; z_1)$ is given as:

$$OI_{\alpha\beta} = \sum_i^N \sum_j^N I_\alpha(x_i, y_j; z_1) I_\beta(x_i, y_j; z_1) \Delta x \Delta y \tag{9}$$

we define the figure of merit is as the mean of the overlap integrals (MOI) across all possible combinations of target intensities:

$$MOI = \frac{Q!}{2N^2\,(Q-2)!} \sum_\alpha^Q \sum_{\beta>\alpha}^Q OI_{\alpha\beta}. \tag{10}$$

To simplify analysis the value A_q is chosen so that the overlap integral of a given intensity with itself is unity (i.e. $OI_{qq} = 1$),

$$A_q = OI_{qq}^{-\frac{1}{2}}. \tag{11}$$

By doing this both the overlap integrals and MOI are bounded between 0 and unity. We seek to minimize Eq. (10) with respect to the phase delays in plane 0

$$\{\phi_q(\chi_0, \psi_0; z_0), \phi_q(\chi_1, \psi_1; z_0), \ldots, \phi_q(\chi_Q, \psi_Q; z_0)\} = \arg\min(MOI). \tag{12}$$

The derivative of the MOI with respect to the phase delay of the ℓ^{th} port and α^{th} target is given as:

$$\frac{dMOI}{d\phi_\alpha(\chi_\ell, \psi_\ell; z_0)} = \frac{Q!}{N^2\,(Q-2)!} \sum_{\beta\neq\alpha}^Q \sum_i^N \sum_j^N \left[\Re\left(u_\alpha^*(x_i, y_j; z_1) \frac{du_\alpha(x_i, y_j; z_1)}{\phi_\alpha(\chi_\ell, \psi_\ell; z_0)} \right) \right. \tag{13}$$
$$\left. \times I_\beta(x_i, y_j; z_1) \Delta x \Delta y \right].$$

It can be seen that Eq. (13) has a similar form to Eq. (6). In this case the unknown value is the derivative which can be solved as before by setting $\widetilde{u}(x_m, y_n; z_P) = du_\alpha(x_i, y_j; z_1)/\phi_\alpha(\chi_\ell, \psi_\ell; z_0)$ and $w_{mnij}(\Delta z_p) = w_{q,mnij}(\Delta z_p)$. As we found previously, solving for the derivative in Eq. (13) allows for the calculation of gradients to optimize the values of the phases. The derivative calculations are independent from one another and can be performed in parallel, which was done on a GPU.

Along with optimizing the values $\phi_q(\chi_\ell, \psi_\ell; z_0)$ we can also optimize the port position values $\{\chi_\ell, \psi_\ell\}$. In this case we resort to a numerical approach of solving for $\phi_q(\chi_\ell, \psi_\ell; z_0)$ using various values of $\{\chi_\ell, \psi_\ell\}$ and selecting the one with the lowest MOI value. Because of the nature of the EOM array all values of Ψ_ℓ can be assumed to exist along the center of the y-axis y_c for all values of ℓ, which eliminates a variable. To further reduce the complexity, it assumed that the EOM array is symmetric about the x-axis (i.e. centered at x_c), the width of the waveguide at the end of the EOM array w_{port} is constant (in addition to the other criterion), and that the spacing between two adjacent ports d_{port} is constant. These restrictions serve to reduce the number of possible variables from $N_\ell\,(N_\ell - 1)$ to 1. While it it is possible that the optimal result occurs when these conditions are not true, this direct numerical approach would become impractical for large values of Q and N_ℓ.

3.3. Input parameters for all simulations

In the approaching section, we analyze the beam steering capabilities of D^2NN instances under various conditions. The important varied parameters were the number of ports (N_ℓ), the number of steering targets (Q), and the number of phase masks ($\mathscr{P}$). In total, 27 different diffractive neural networks were trained covering every permutation of $\{N_\ell, Q, \mathscr{P}\}$; for each, the end-firing waveguide arrangement was configured to have optimized input fields, or equivalently, a minimized MOI. Each simulation also required several parameters to be defined and held static. These are now described: a wavelength of $\lambda = 1$ µm was chosen, and the refractive index of the system assumed propagation through air, $n \approx 1$. The grid spacing resolution $\Delta x, \Delta y = 25$ µm and the number of square grid points $N \times N = 400 \times 400$ determined the size of the simulation window to be 10 mm × 10 mm. Our choice was to define a simulation window centered at $x_c = y_c = 5$ mm. This gave window bounds of $[0, 2 * x_c]$ and $[0, 2 * y_c]$. These values, together with w_{port}, Δz_1 and λ, result in a Fresnel number of $N_F = .01$ and a Rayleigh limit of $r_{limit} = 1.22$. This ensured that we were within our input source assumptions. When optimizing the value of d_{port}, the search ranged from $d_{port,min} = 1$ µm to $d_{port,max} = 100$ µm in increments of $\Delta d_{port} = 1$ µm. The iterative gradient descent search used in our optimization was initialized with random inputs for $\phi_q(\chi_\ell, \psi_\ell; z_0)$ or $\phi(x_s, y_t; z_v)$. Each gradient descent search was performed over 100

Table 1. Summary of the parameters used in our simulations. Most parameters were held static for every D^2NN instance. Importantly, the number of phase masks ($\mathscr{P}$), steering targets (Q), and EOM ports (N_ℓ) were varied between three values. All permutations were simulated to obtain the $3^3 = 27$ different D^2NN instances.

Static Parameter Description	Symbol	Value
Window Center for x Axis	x_c	5 mm
Window Center for y Axis	y_c	5 mm
Resolution of x	Δx	25 µm
Resolution of y	Δy	25 µm
Number of Simulation Points	$N \times N$	400 × 400
Distance from EOM Phase Array to First Mask	Δz_0	1 cm
Distance Between Masks	$\Delta z_{p>0}$	10 cm
Distance from Final Mask to Output Plane	$\Delta z_{\mathscr{P}}$	1 m
EOM Port Width	w_{port}	10 µm
EOM Optimal Port Spacing Search Range	$[d_{port,min}, d_{port,max}]$	[1 µm, 100 µm]
EOM Optimal Port Spacing Search Resolution	Δd_{port}	1 µm
Wavelength of Light	λ	1 µm
Refractive Index	n	1
Varied Parameter Description	**Symbol**	**Set of Values**
Number of Phase Masks	$\mathscr{P}$	{1,2,3}
Number of Steering Targets	Q	{3,4,5}
Number of EOM Ports	N_ℓ	{3,4,8}

generations, with the lowest MOI or MSE result selected and stored. Additionally, the iterative gradient descent search for the optimal d_{port} and $\phi_q(\chi_\ell, \psi_\ell; z_0)$ was repeated for each port-target combination. The values of d_{port} and $\phi_q(\chi_\ell, \psi_\ell; z_0)$ with the lowest MOI was selected and used as inputs for the corresponding D^2NN instance. All simulation input parameters used in this manuscript are gathered and summarized in Table 1.

4. Multi-directional beam steering results

In this section, we present the beam steering results of 27 different retrieved phase mask cascades. These phase elements were designed by training D^2NNs following the framework laid out in section 2 in the context of the experimentally proposed active system of section 3. Due to the difficulty of displaying the outputs for each of the 27 different simulations, we have chosen to present a detailed walk-through of the simplest case. We then follow with the outputs related to few more complicated cases before discussing the aggregated results and patterns that emerged.

4.1. Detailed walk-through: three ports, three steering states

The three port EOM array with a three target requirement ($N_\ell = 3$, $Q = 3$) is the least complicated source and target combination within our simulation set. Before we train this diffractive neural network, we must identify the optimum d_{port} and $\phi_q(\chi_\ell, \psi_\ell; z_0)$ for this arrangement. Implementing $N_\ell = 3$, Fig. 3 plots the lowest MOI as a function of port width for each Q obtained via the iterative approach defined in section 3.2. From this, one can identify an overall minimal value for $Q = 3$ of ≈ 0.214 occurring at $d_{port} = 15$ µm. This optimal port spacing is then used in conjunction with the protocol defined in section 3.1 to obtain the three port phase functions which will generate three minimally overlapping input electric fields. Figure 3 demonstrated a MOI varying insignificantly with d_{port} for all $N_\ell = 3$ cases; however, the minimum MOI increased with Q. This correlation indicated that as the number of desired steering states increases, it becomes progressively harder to define a unique set of non-overlapping input fields. A general corollary arises from this: the amount of ghosting will always increase with increasing Q. Consider, for example, the illustrative case of a $N_\ell = 3$ port system commanded to produce an unreasonable $Q = 100$ steering states. Because three electro-optic modulated ports would fail at producing 100 spatially distinct non-overlapping intensity patterns, the MOI would be large. In fact, the MOI would be close to unity, meaning that the optimal steering arrangement would consist of energy spread uniformly in the image plane. In other words, the concept of steering becomes lost.

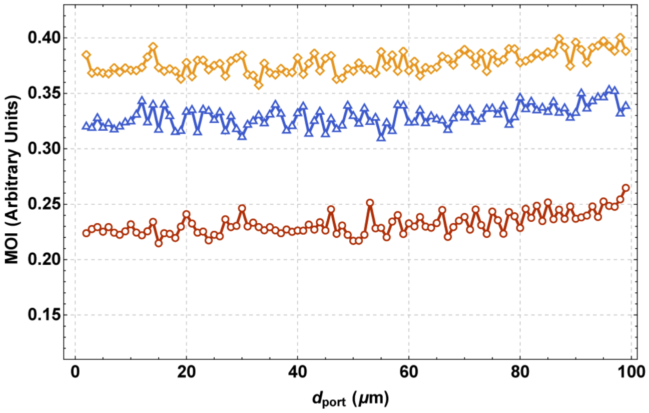

Fig. 3. MOI vs d_{port} for $N_\ell = 3$ port EOM array with $Q = 3$ target states (○), $Q = 4$ target states (△), and $Q = 5$ target states (◇).

Equipped with the optimal d_{port} = 15 µm and three $\phi_q(\chi_\ell, \psi_\ell; z_0)$ values, Eq. (8) is then used to generate three input fields at the z_1 plane. Each input field is assigned a corresponding target output intensity (Fig. 4). The target intensities chosen were single 2 mm x 2 mm squares in the output plane $z_{\mathscr{P}}$ such that all optical energy was focused in an isolated region (Fig. 4(a)). Outside the square target region, the intensity was zeroed, thus encouraging the desired condition of minimal ghosting during neural network training. The generated input intensities from the three port phase array appear similar (Fig. 4(b)), but a close inspection reveals a physical shift in the bright and dark bands consistent with minimal intensity overlap (Fig. 4(c)).

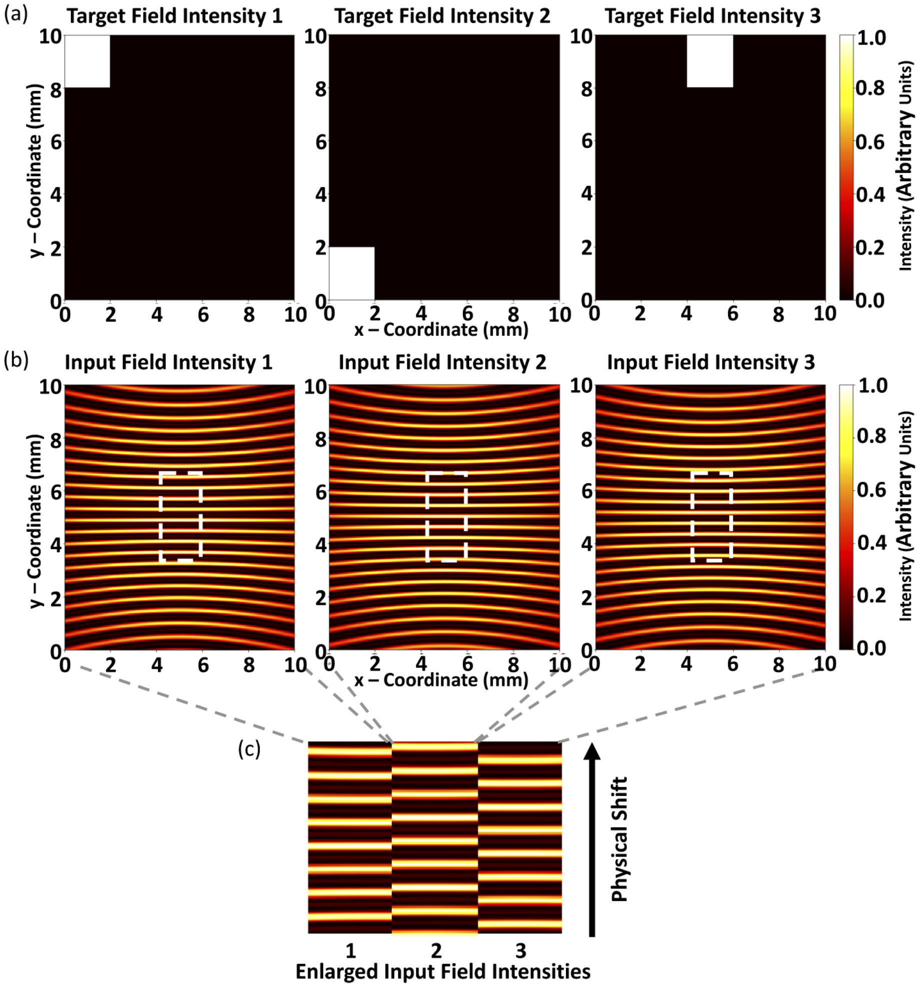

Fig. 4. (a) The target state output intensities. (b) The corresponding input field intensities generated by a N_ℓ = 3 port EOM array Q = 3 targets, (c) and an enlarged section of each input fields showing the physical shift in the otherwise similar interference patterns.

The pairs of input and output field intensities of Fig. 4 constitute training data used to converge three separate D^2NN instances. We remind the reader that after reaching convergence, the network layers come to represent cascaded phase masks which optimally connect the input and output intensity training data. Each of these three instances investigated the use of one, two, or three hidden layers. Figure 5 displays the cascade of phase masks that were retrieved. Examination of the masks result in both intuitive and unexpected characteristics. In the $\mathscr{P}$ = 1 case, the single retrieved phase mask spatially converged to an array of distinct Fresnel lenslets (Fig. 5(a)). These periodic lenses had the same center-to-center spacing as the bright and dark bands of the input intensity fields; furthermore, the general orientation of the lenslets were biased every three rows such that energy should be steered efficiently in the target directions depending on the physical shift discussed in Fig. 4. In the $\mathscr{P} = 2$ case, the D^2NN qualitatively appeared to lessen the modulation depth and lensing power of the individual lenslets (Fig. 5(b)). Interestingly, the lenslet array appeared only in the first phase mask while the second layer formed larger "steering zones" in the approximate shape of the 2 mm × 2 mm squares; however, further zooming into these zones did not reveal any intuitive structuring. Bringing in a third hidden layer seemed to have little effect on the first mask (Fig. 5(c)); instead, the steering zones discussed in the $\mathscr{P} = 2$ case appear to split into a coarse and fine adjustment with visible grating-like structures forming. Generally speaking, we find that the increasing the number of hidden layers, $\mathscr{P}$, gives a filtering-like or factoring-like flavor to the resulting phase masks. This result is reminiscent of the feature-filters that often arise when studying convolutional neural networks.

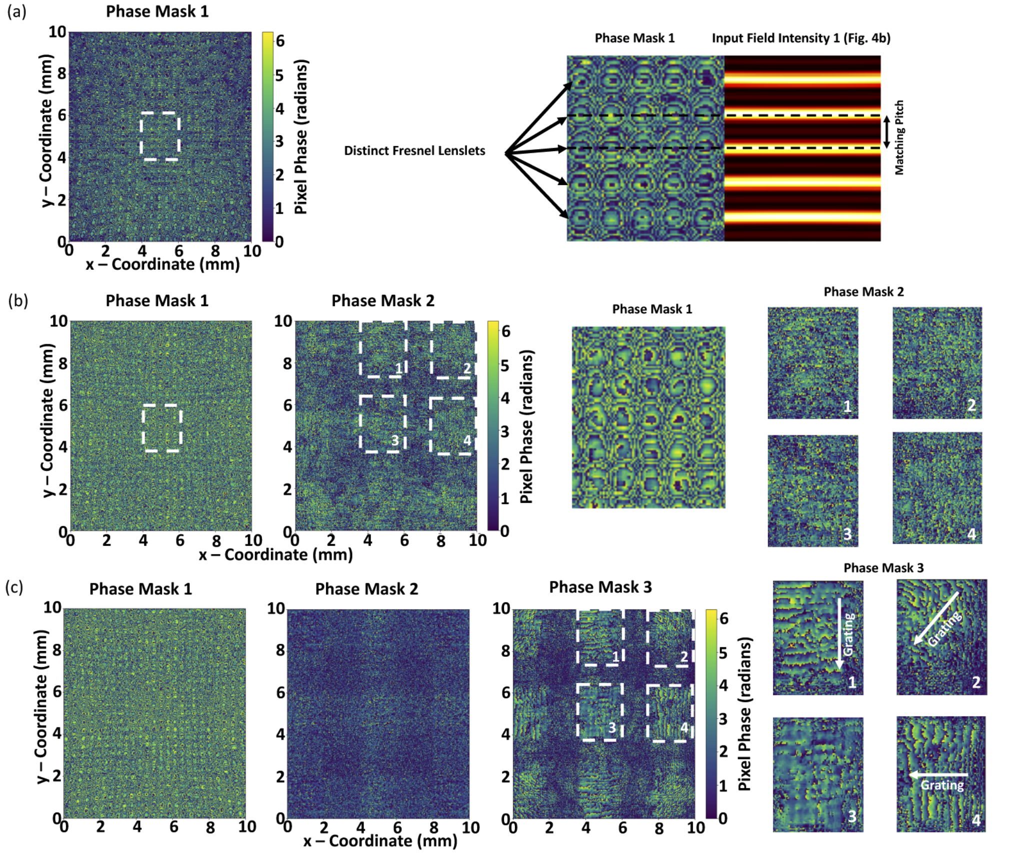

Fig. 5. D^2NN phase masks generated for the three port, three target EOM array ($N_\ell = 3$, $Q = 3$) system at planes z_1, z_2, and z_3. From top to bottom are the results for (a) $\mathscr{P} = 1$ mask , (b) $\mathscr{P} = 2$ masks, (c) and $\mathscr{P} = 3$ masks. Emergent features include an array of Fresnel lenslets with a periodicity matching the input intensities and grating-like structures

The output fields for the $N_\ell = 3$, $Q = 3$ case resulting from each calculated phase mask cascade, $\mathscr{P}$, are displayed in Fig. 6. All output field values have been normalized so that the largest intensity value in the window is unity. Qualitatively, the calculated output intensities recreated the desired target profiles of Fig. 4. A direct result was that as the number of hidden layers increased, the D^2NN were better able to simulate the target profile. To elucidate: the ghosting effect seen on all three targets in Fig. 6(a), is notably reduced in Figs. 6(b) and (c). As we see later, this trend was generally true across all values of Q simulated. In this system, however, we found a significant amount of optical energy (> 58%) scattered outside the target region. While some of the scatter was concentrated into wrong beam-steering zones, most was uniformly dispersed throughout the simulation window with very low signal.

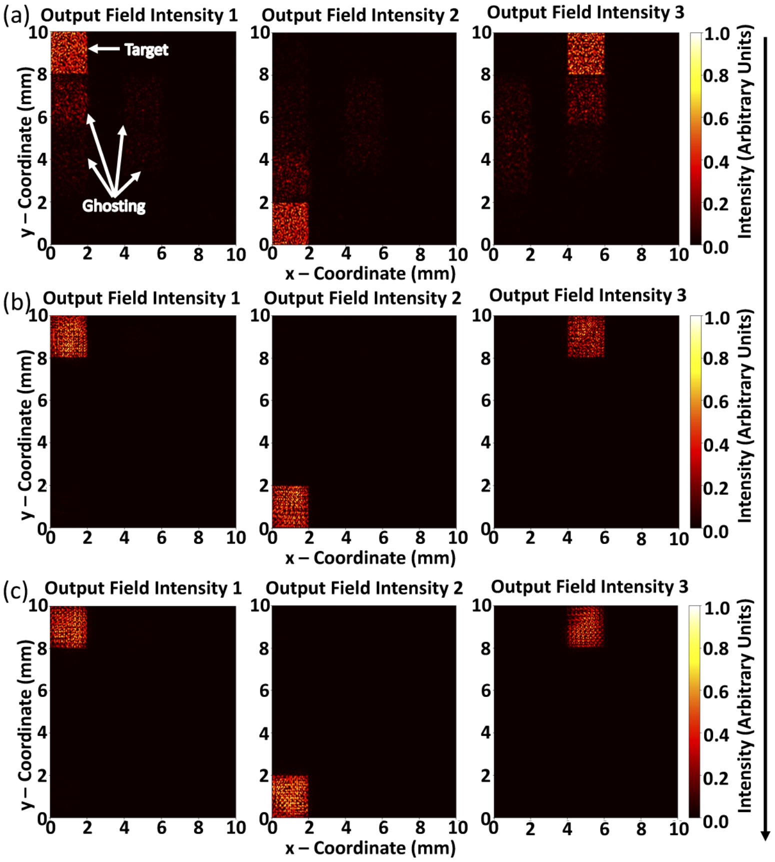

Fig. 6. Resulting output fields of the D^2NN for a three port, three target system ($N_\ell = 3$, $Q = 3$). Results of (a) $\mathscr{P} = 1$ mask , (b) $\mathscr{P} = 2$ masks, (c) and $\mathscr{P} = 3$ masks. We highlight the off-target scattering or ghosting in the $\mathscr{P} = 1$ case. This ghosting effect is mitigated with increasing $\mathscr{P}$ with most of the scatter dispersed throughout the background.

4.2. Dependence on ports: five steering states

Of the 27 different D^2NN instances we trained, the $Q = 5$ scenarios represented the most steering constraints we imposed. As before, the process of finding the optimal EOM conditions, defining the input and output training set, and converging a D^2NN was performed. The optimization plots for MOI vs d_{port} for the $N_\ell = \{4, 8\}$ cases are displayed in Fig. 7 for each Q. We found a similar MOI pattern of behavior as the three port simulation. Namely, as Q increased the larger the average overlap decreased; however, unlike before, there was a clear dependence of MOI on the value of d_{port}. Larger values of d_{port} resulted in larger MOI values, so that the optimum generally favored a smaller but specific gap between ports. The optimum values for the four and eight port systems produced *MOI* values that were lower than the three port counterparts. The lowest *MOI* values for a given number of targets was always observed in the eight port simulation. This result is consistent with the expected behavior of phased arrays, as the number of elements/ports is directly indicative of how well it can superimpose a unique non-overlapping optical pattern.

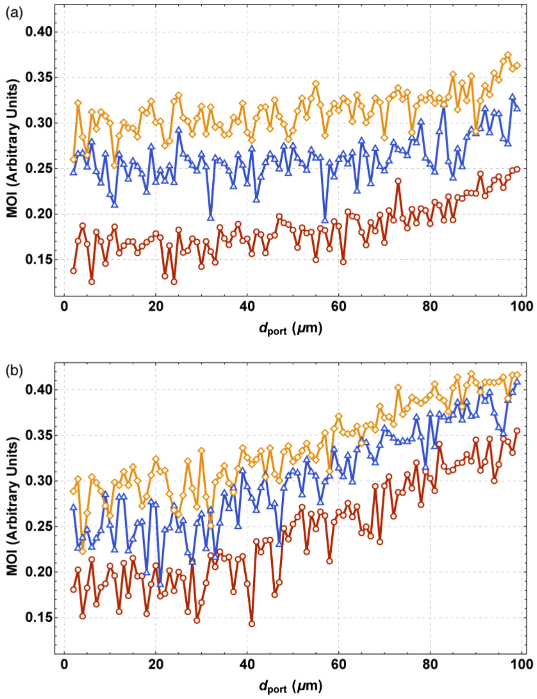

Fig. 7. *MOI* vs d_{port} for (a) a four port EOM array and (b) eight port EOM array for three target states (○), four target states (△), and five target states (◇).

We then took the optimized input fields for the three, four, and eight port system and assigned five unique non-overlapping 2 mm × 2 mm square targets (Fig. 8(a)). Using $\mathscr{P} = 2$ hidden layers, three D^2NN instances were then trained for the different values of N_ℓ (Figs. 8(b)-(d)). We choose to present the two masks scenario here because the decrease in ghosting with N_ℓ is apparent in each row. When comparing the data across all ports, the primary benefit of increasing N_ℓ was a decrease in average *MOI* value. This in turn increased the average fraction of power on target.

4.3. Aggregate results and emergent patterns

We remind the reader that for each and every D^2NN instance, we had to first find the optimum d_{port} and $\phi_q(\chi_\ell, \psi_\ell; z_0)$ which was dependent on N_ℓ and Q. The *MOI* computation only involved the EOM phase array and therefore did not depend on the number of cascaded phase masks, $\mathscr{P}$. Extracting values from Figs. 3 and 7, Table 2 orders the optimal d_{port} spacing and minimal *MOI* for each situation we studied. We observed as a general trend that lower *MOI* values resulted when a higher number of ports and lower number of targets were used; however, the lowest *MOI* was not observed in the $N_\ell = 8$, $Q = 3$ case as would be expected. This could have been the result of our iterative gradient descent search missing the optimal port value or the assumption of a uniform d_{port} value breaking down.

After training the full 27 D^2NN instances, output field intensities were calculated like in Figs. 6 and 8. Ideally, each target steering state is perfectly recreated, resulting in 100% of the input power falling within the appropriate 2 mm × 2 mm square. We quantified the phenomenon by

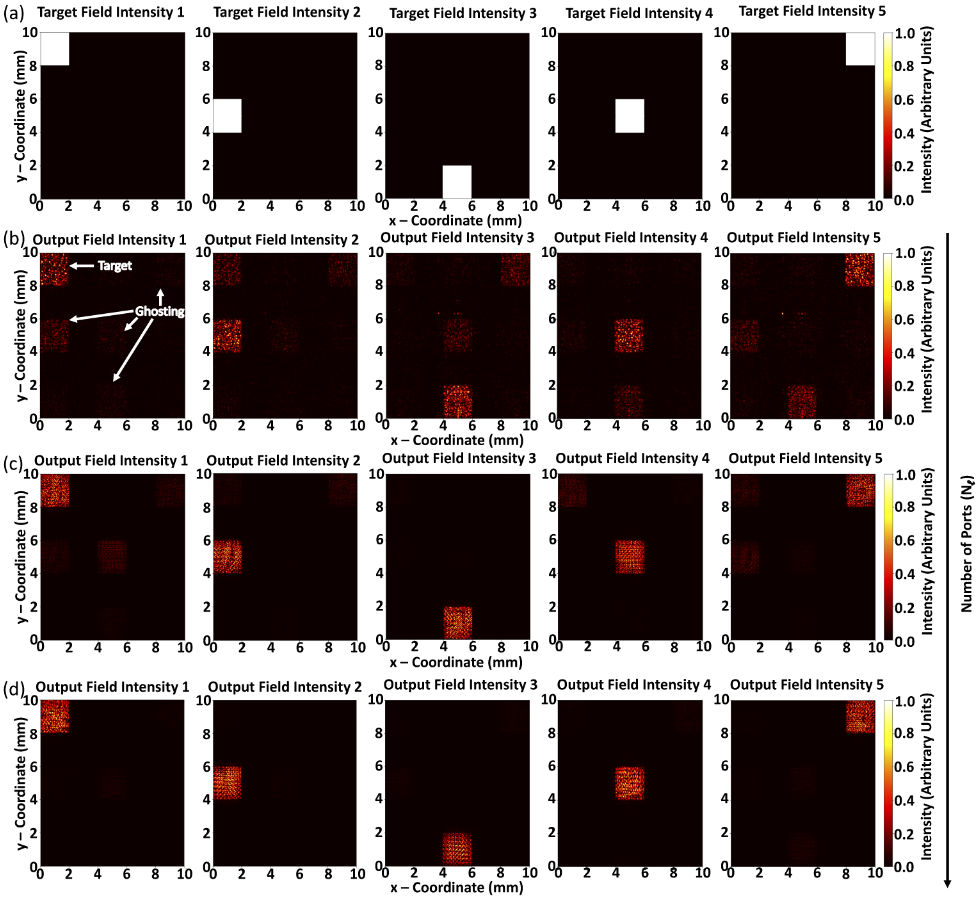

Fig. 8. (a) $Q = 5$ target states with the respective output fields of the D^2NN using $\mathscr{P} = 2$ phase masks. Results for (b) $N_\ell = 3$, (c) $N_\ell = 4$, and (d) $N_\ell = 8$ port EOM array.

Table 2. Tabulated number of ports and target steering states with corresponding optimal d_{port} in order of lowest *MOI*. A lower *MOI* directly corresponded to a smaller amount of ghosting.

N_ℓ	Q	d_{port} (μm)	MOI
4	3	24	.126
8	3	41	.143
8	4	21	.186
4	4	57	.193
3	3	15	.214
8	5	4	.223
4	5	11	.254
3	4	55	.309
3	5	33	.357

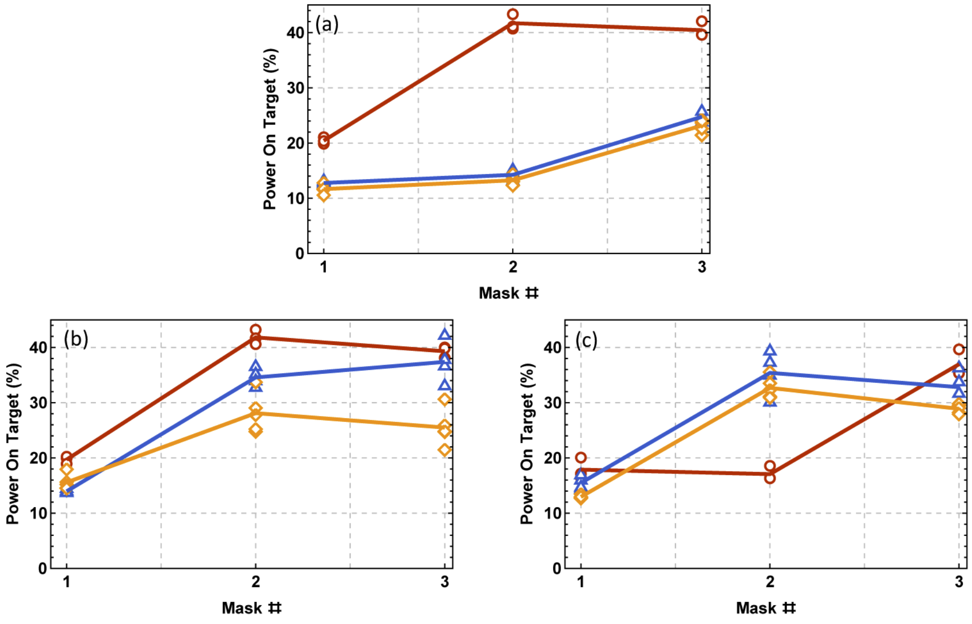

Fig. 9. Fraction of power in target steering region vs the number of mask for 3 port(a), 4 port (b), and 8 port (c) EOM waveguide array for 3 target states (○), 4 target states (△), and 5 target states (◇). A piecewise linear fit for each respective target steering state (—,—,—) that passes through the average fraction of power on targets for each mask number is also shown. Each mark represents one of 3, 4, or 5 target states.

calculating the fraction of power impinging on this designated square target region versus the total power in the simulation window for every scenario. For the $N_\ell = 3$ case, the fraction of power on target vs. $\mathscr{P}$ for every steering number Q is plotted in Fig. 9(a). The results have two key implications: one, increasing the number of masks increases the amount of power that can be focused in the target region; two, as the number of simulated targets increases, the more difficult it is to confine the optical energy into the target region. In this three port case, the best results were observed for the three target system when at least two masks were used ($\sim 40\%$ power on target). The worst results occurred when a single mask was used to guide to five steering states ($< 12\%$ power on target). This corroborates the trend we observed in Fig. 3.

The calculated fraction of power in the target regions for all $N_\ell = 4$ and $N_\ell = 8$ cases are given in Figs. 9(b) and (c). The two trends observed in the three port system were held true for the four and eight port system as well. As before, increasing the number of masks increased the fraction of power in the targeted region (decreased the ghosting). The effect, however, gave diminishing returns after two layers. On average, the addition of a third layer either showed little ghosting improvement or slightly increased the scattering. This latter behavior is likely due to the the fact that the best set of masks were simply not found by our iterative gradient descent search. Overall, the key findings are as follows:

1. We found a consistent correlation between the ability of D^2NN to steer optical energy into a specific region and the mean intensity overlap of the input fields. In general lower MOI values resulted in D^2NNs that could steer more optical energy (up to ~40%) into the designated steering state.

2. The ability of EOM optical arrays to produce low MOI is contingent on the number of ports and the number of target steering states. In general, the higher the number of target steering states, the larger the MOI. This can be mitigated however, by increasing the number of ports as we found this lowers the MOI for a given number of targets.

3. The D^2NN ability to steer the optical energy of optimized inputs to the target region consistently benefited from cascading more phase masks. While a two gave the most returns in our investigation, a third mask slightly increased the power impinging on the target region in most of the cases.

5. Conclusions

In this manuscript we have demonstrated how the combination of EOM arrays and D^2NN can be designed for use in beam steering. By exploiting the computational capabilities of D^2NN as image classifiers, we were able to simulate a device that could take unique inputs from a user controlled optical source (EOM array) and steer its optical energy to designated regions of space. Our study develops a general procedure for selecting the optimal outputs of the EOM array to allow for efficient steering by the D^2NN and builds on the work done in previous studies with neural networks. We examined how the performance of the device, as measured by the fraction of energy steered into a given region, would vary with parameters related to the number of individual sources in the EOM arrays, number of target output intensities, and number of cascading phase mask in the D^2NN. We observed that in general the more elements the EOM array has the easier it is to design a D^2NN that can produce the desired target intensity profile. Additionally, the simulated systems performed best when at least two phase mask were used, consistent with what has been reported in previous studies. Interestingly, recognizable features such as fresnel lenslets and gratings structures emerged spotaneously with the increasing number of layers. For a given phase mask in the system, only one of the aforementioned features could be recognized with the other features either not being present at all or existing in one of the other phase mask in the system. This indicates that the D^2NN are dividing up the filtering task in a manner similar to their electronic based electronic deep neural network counterparts. The key difference in the D^2NN case is that the physics of light give us useful incite into the nature of the structures being generated.

We note that the method used for calculating the gradients was brute force and as a consequence is relatively slow compared to other routines in the algorithm such as forward propagation and determination of the errors. As discussed in the Sec. 2.3, calculating the derivative of the field with respect to the phase of a given pixel is analogous to placing a $-j$ phase mask and an aperture in the system that blocks out all other pixels in the plane of interest ($a_{stij}(z_v, z_p)$ term in Eq. (7)) and then calculating the Rayleigh-Sommerfeld integral. To obtain derivatives for every pixel on every mask we must shift the aperture to the relevant position and recalculate the integral equation again. The process can be impractically slow for large values of Q and $\mathscr{P}$. However, there is hope in the sense that we should be able to obtain the derivatives while only having to calculate the Rayleigh-Sommerfeld integral once. Heuristically we know this is true because if we eliminate the shifting aperture but retain the $-j$ phase mask, the output will simply be the linear super-position of all the derivatives from every pixel in any given plane. In essence, without the aperture we would be calculating all the derivatives at once. Unfortunately, we are interested in the individual values of gradients with respect to an individual pixel and not the sum of the individual gradients of all pixels. So, we resort to using the shifting aperture function $a_{stij}(z_v, z_p)$ as a spatial filter to block out the contributions to the derivative from other pixels. We believe adjoint sensitivity approaches, like those used in more rigorous full wave FDTD simulations, may be the key to determining how to more efficiently calculate gradients. In those problems it is imperative that the calculation of gradients, which involve 2 at minimum calculations of the full wave solver, are minimized. Adjoint sensitivity analysis has been used to minimize simulations to at most 2 full wave calculations. We believe based on the heuristic argument presented above, that adjoint sensitivity analysis can be applied to D^2NN, to limit the number of simulations necessary for the gradients to 2 calculations of the Rayleigh Sommerfeld integral.

Direct applications of our work include optical phase array based LADAR systems such as those currently being explored in autonomous vehicle research. The architecture presented here is capable of performing 2D beam steering using only a 1D phase array sidestepping the technical hurdle of fabricating 2D phase arrays. Furthermore, our methods provides a direct path to getting the best performance for a limited number of elements present in the phase array and a given number of desired target steering states. While we found it difficult to increase the steering states without degrading the steering capabilities of the D^2NN, one possible solution may be to incorporate additional wavelengths taking advantage of optical dispersion in both the EOM array and the D^2NN. Our study was limited to a single laser wavelength, but the methods outlined here can easily be extended to incorporate more values accounting for the requisite dispersive effects of the material/structures used in the EOM waveguides and D^2NN phase masks. With wavelength acting as a new degree of freedom, it should be possible to design a single EOM arrray and D^2NN system that can beam steer through both the color and spatial structure of the field input.

Funding

Air Force Research Laboratory (FA8650-16-D-5404-0013).

Acknowledgments

We would like to thank members of the LMISOM group and the PACE team for useful discussions.

Disclosures

The authors declare no conflicts of interest.

References

1. S. M. Weiss and N. Indurkhya, “Rule-based Machine Learning Methods for Functional Prediction,” J. Artif. Intell. Res. **3**, 383–403 (1995).
2. S. Sette and L. Boullart, “Implementation of genetic algorithms for rule based machine learning,” Eng. Appl. Artif. Intell. **13**(4), 381–390 (2000).
3. J. H. Friedman and W. Stuetzle, “Projection Pursuit Regression,” J. Am. Stat. Assoc. **76**(376), 817–823 (1981).
4. C. D. Manning, H. Schütze, and G. Weikurn, “Foundations of Statistical Natural Language Processing,” in *SIGMOD Record* (2002).
5. R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in *Proceedings of the 25th international conference on Machine learning*, (2008), pp. 160–167.
6. F. Neri and L. Saitta, “Knowledge representation in machine learning,” in *European Conference on Machine Learning*, vol. 784 LNCS (Springer, 1994), pp. 20–27.
7. E. Davis, “Knowledge Representation,” in *International Encyclopedia of the Social & Behavioral Sciences: Second Edition*, (Elsevier Inc., 2015), pp. 98–104.
8. L. Monostori, J. Váncza, and S. R. Kumara, “Agent-based systems for manufacturing,” CIRP Annals - Manuf. Technol. **55**(2), 697–720 (2006).
9. M. Luck, P. Mcburney, C. Preist, C. Guilfoyle, S. Bergamaschi, P. Davidsson, F. Dignum, P. Edwards, M. Klusch, D. Kudenko, S. Moss, P. Petta, V. Roth, C. Sierra, and F. Zambonelli, *Agent Technology: Enabling Next Generation Computing (A Roadmap for Agent Based Computing with)* (AgentLink, 2003).
10. M. Gams, I. Y. H. Gu, A. Härmä, A. Mu noz, and V. Tam, “Artificial intelligence and ambient intelligence,” J. Ambient Intell. Smart Environ. **11**(1), 71–86 (2019).
11. K. C. Morris, C. Schlenoff, and V. Srinivasan, “Guest Editorial A Remarkable Resurgence of Artificial Intelligence and Its Impact on Automation and Autonomy,” IEEE Trans. Automat. Sci. Eng. **14**(2), 407–409 (2017).
12. R. Uetz and S. Behnke, “Large-scale object recognition with CUDA-accelerated hierarchical neural networks,” in *2009 IEEE international conference on intelligent computing and intelligent systems*, vol. 1 (IEEE, 2009), pp. 536–541.
13. D. Strigl, K. Kofler, and S. Podlipnig, “Performance and scalability of GPU-based convolutional neural networks,” in *2010 18th Euromicro Conference on Parallel, Distributed and Network-based Processing*, (IEEE, 2010), pp. 317–324.
14. Y. Lecun, Y. Bengio, and G. Hinton, “Deep learning,” Nature **521**(7553), 436–444 (2015).
15. W. Wang, Y. Huang, Y. Wang, and L. Wang, “Generalized Autoencoder: A Neural Network Framework for Dimensionality Reduction DeepVision: Deep Learning for Computer Vision 2014,” in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, (2014), pp. 490–497.

16. F. Jiang, W. Tao, S. Liu, J. Ren, X. Guo, and D. Zhao, "An End-to-End Compression Framework Based on Convolutional Neural Networks," IEEE Trans. Circuits Syst. Video Technol. **28**(10), 3007–3018 (2018).
17. C. Dong, Y. Deng, C. C. Loy, and X. Tang, "Compression artifacts reduction by a deep convolutional network," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2015 Inter (2015), pp. 576–584.
18. D. Ciregan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *2012 IEEE conference on computer vision and pattern recognition*, (IEEE, 2012), pp. 3642–3649.
19. G. E. Dahl, D. Yu, S. Member, L. Deng, and A. Acero, "Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition," IEEE Transactions on audio, speech, and language processing **20**(1), 30–42 (2012).
20. S. Jana, Y. Tian, K. Pei, and B. Ray, "DeepTest: Automated testing of deep-neural-network-driven autonomous cars," in *Proceedings of the 40th international conference on software engineering*, vol. 2018-May (2018).
21. M. Bojarski, P. Yeres, A. Choromanska, K. Choromanski, B. Firner, L. Jackel, and U. Muller, "Explaining How a Deep Neural Network Trained with End-to-End Learning Steers a Car," arXiv preprint arXiv:1704.07911 pp. 1–8 (2017).
22. J. Chen, B. Sun, H. Li, H. Lu, and X. S. Hua, "Deep CTR prediction in display advertising," in *Proceedings of the 24th ACM international conference on Multimedia*, (2016), Figure 2, pp. 811–820.
23. R. Wang, G. Fu, B. Fu, and M. Wang, "Deep & cross network for ad click predictions," in *Proceedings of the ADKDD'17*, (2017), August 2017, pp. 1–7.
24. S. Bhatt, F. Patwa, and R. Sandhu, "An Access Control Framework for Cloud-Enabled Wearable Internet of Things," in *2017 IEEE 3rd International Conference on Collaboration and Internet Computing (CIC)*, vol. 2017-Janua (IEEE, 2017), pp. 328–338.
25. M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng, "TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems," arXiv preprint arXiv:1603.04467 (2016).
26. A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in neural information processing systems*, (NIPSF, 2019), NeurIPS.
27. K. Choudhary, M. Bercx, J. Jiang, R. Pachter, D. Lamoen, and F. Tavazza, "Accelerated Discovery of Efficient Solar Cell Materials Using Quantum and Machine-Learning Methods," Chem. Mater. **31**(15), 5900–5908 (2019).
28. E. S. Harper, M. N. Weber, and M. S. Mills, "Machine accelerated nano-targeted inhomogeneous structures," in *2019 IEEE Research and Applications of Photonics in Defense Conference (RAPID)*, IEEE (IEEE, 2019), pp. 1–5.
29. E. S. Harper, J. P. Vernon, M. S. Mills, and E. J. Coyle, "Inverse Design of Broadband Highly Reflective Metasurfaces using Neural Networks," Phys. Rev. B **101**(19), 195104 (2020).
30. L. Ruthotto and E. Haber, "Deep Neural Networks Motivated by Partial Differential Equations," J. Math. Imaging Vis. **62**(3), 352–364 (2020).
31. J. Sirignano and K. Spiliopoulos, "DGM: A deep learning algorithm for solving partial differential equations," J. Comput. Phys. **375**, 1339–1364 (2018).
32. T. P. Miyanawala and R. K. Jaiman, "An Efficient Deep Learning Technique for the Navier-Stokes Equations: Application to Unsteady Wake Flow Dynamics," arXiv preprint arXiv:1710.09099 (2017).
33. M. Raissi, A. Yazdani, and G. E. Karniadakis, "Hidden Fluid Mechanics: A Navier-Stokes Informed Deep Learning Framework for Assimilating Flow Visualization Data," arXiv preprint arXiv:1808.04327 (2018).
34. J. N. Kutz, "Deep learning in fluid dynamics," J. Fluid Mech. **814**, 1–4 (2017).
35. D. Psaltis and N. Farhat, "Optical information processing based on an associative-memory model of neural nets with thresholding and feedback," Opt. Lett. **10**(2), 98–100 (1985).
36. D. Psaltis, D. Brady, X. G. Gu, and S. Lin, "Holography in artificial neural networks," Nature **343**(6256), 325–330 (1990).
37. K. Wagner and D. Psaltis, "Optical neural networks: an introduction by the feature editors," Appl. Opt. **32**(8), 1261–1263 (1993).
38. J. W. Goodman, *Introduction to Fourier Optics* (Roberts and Company Publishers, 2005), 2nd ed.
39. X. Lin, Y. Rivenson, N. T. Yardimci, M. Veli, Y. Luo, M. Jarrahi, and A. Ozcan, "All-optical machine learning using diffractive deep neural networks," Science **361**(6406), 1004–1008 (2018).
40. T. Yan, J. Wu, T. Zhou, H. Xie, F. Xu, J. Fan, L. Fang, X. Lin, and Q. Dai, "Fourier-space Diffractive Deep Neural Network," Phys. Rev. Lett. **123**(2), 023901 (2019).
41. F. Shen and A. Wang, "Fast-Fourier-transform based numerical integration method for the Rayleigh-Sommerfeld diffraction formula," Appl. Opt. **45**(6), 1102–1110 (2006).
42. C. Buitrago-Duque and J. Garcia-Sucerquia, "Non-approximated Rayleigh–Sommerfeld diffraction integral: advantages and disadvantages in the propagation of complex wave fields," Appl. Opt. **58**(34), G11–G18 (2019).
43. D. Mengu, Y. Luo, Y. Rivenson, and A. Ozcan, "Analysis of diffractive optical neural networks and their integration with electronic neural networks," IEEE J. Sel. Top. Quantum Electron. **26**(1), 1–14 (2020).

44. M. R. Kossey, C. Rizk, and A. C. Foster, "End-fire silicon optical phased array with half-wavelength spacing," APL Photonics **3**(1), 011301 (2018).
45. S. Zheng, X. Zeng, L. Zha, H. Shangguan, S. Xu, and D. Fan, "Orthogonality of diffractive deep neural networks," arXiv preprint arXiv:1811.03370 pp. 1–5 (2018).