applied optics

Knowledge distillation circumvents nonlinearity for optical convolutional neural networks

Jinlin Xiang,[1,2] Shane Colburn,[2] Arka Majumdar,[2,3,5] and Eli Shlizerman[3,4,*]

[1]Department of Mechanical Engineering, University of Washington, Seattle, Washington 98195, USA
[2]Department of Electrical & Computer Engineering, University of Washington, Seattle, Washington 98195, USA
[3]Department of Physics, University of Washington, Seattle, Washington 98195, USA
[4]Department of Applied Mathematics, University of Washington, Seattle, Washington 98195, USA
[5]e-mail: arka@uw.edu
*Corresponding author: shlizee@uw.edu

Received 8 July 2021; revised 1 February 2022; accepted 7 February 2022; posted 7 February 2022; published 14 March 2022

In recent years, convolutional neural networks (CNNs) have enabled ubiquitous image processing applications. As such, CNNs require fast forward propagation runtime to process high-resolution visual streams in real time. This is still a challenging task even with state-of-the-art graphics and tensor processing units. The bottleneck in computational efficiency primarily occurs in the convolutional layers. Performing convolutions in the Fourier domain is a promising way to accelerate forward propagation since it transforms convolutions into elementwise multiplications, which are considerably faster to compute for large kernels. Furthermore, such computation could be implemented using an optical $4f$ system with orders of magnitude faster operation. However, a major challenge in using this spectral approach, as well as in an optical implementation of CNNs, is the inclusion of a nonlinearity between each convolutional layer, without which CNN performance drops dramatically. Here, we propose a spectral CNN linear counterpart (SCLC) network architecture and its optical implementation. We propose a hybrid platform with an optical front end to perform a large number of linear operations, followed by an electronic back end. The key contribution is to develop a knowledge distillation (KD) approach to circumvent the need for nonlinear layers between the convolutional layers and successfully train such networks. While the KD approach is known in machine learning as an effective process for network pruning, we adapt the approach to transfer the knowledge from a nonlinear network (*teacher*) to a linear counterpart (*student*), where we can exploit the inherent parallelism of light. We show that the KD approach can achieve performance that easily surpasses the standard linear version of a CNN and could approach the performance of the nonlinear network. Our simulations show that the possibility of increasing the resolution of the input image allows our proposed $4f$ optical linear network to perform more efficiently than a nonlinear network with the same accuracy on two fundamental image processing tasks: (i) object classification and (ii) semantic segmentation.

https://doi.org/10.1364/AO.435738

1. INTRODUCTION

Convolutional neural network (CNN) architectures are well known for their ability to compute visual tasks [1–5]. Many of these tasks require fast processing of real-time signals. In autonomous navigation, for example, a network must be capable of identifying obstacles with different textures and lighting conditions in real time. While CNNs are instrumental in providing high accuracy for such tasks, the time that it takes for the input to propagate through the trained network (forward propagation time) is large and precludes real-time operation. The main reason for such inefficiency is the computational complexity of CNNs, which is $\mathcal{O}(HWk^2)$, where H and W are the height and width of an image frame and $K_2 = k \times k$ is the size of the convolutional kernel. The challenge of enhancing the computational performance has driven significant development of electronic hardware that is dedicated to computing convolutions, with graphics processing units (GPUs) and tensor processing units (TPUs), which deliver an order of magnitude acceleration. Even with such dedicated hardware, however, effective computation times are still suboptimal and become too large for many applications, especially when high-resolution inputs are processed. For example, when applied to ResNet-18 for autonomous navigation, the Jetson Nano, NVIDIA hardware development kit, achieves at most a five frames per second (fps) rate for images of dimensions 1000×1000 [6].

Since convolution is the most time-consuming operation in CNNs, a possible gain in computational efficiency can be achieved by implementing the CNN in the Fourier domain (spectral domain) [7,8]. The fast Fourier transform (FFT) operation has the complexity of $\mathcal{O}(HW\log(HW)$ for the images and the kernels. In the Fourier domain, the convolution is transformed into an elementwise product with only $\mathcal{O}(HW)$ operations. While promising, the approach does not boost the forward propagation time in practice. Due to nonlinearities that follow most of the layers in a CNN, the spectral approach ends up including a large number of costly forward and inverse Fourier transforms between successive layers. Optimizations of network architectures to be compatible with spectral operations were proposed, such as FCNN and Clebsch–Gordan nets [9,10]. These architectures, however, appear to suffer from a reduction in classification accuracy for complex visual tasks. For example, FCNN reaches less than half of state-of-the-art accuracy on CIFAR-10. Another branch of research developed in parallel is the introduction of spectral pooling layers for the purpose of adapting the spatially applied max-pooling operation to the spectral domain [11–13]. These networks still entail nonlinear activations that require conversions between the spatial and spectral domains. Removing these nonlinear functions dramatically reduces the computational complexity of spectral CNNs, which comes at the cost of reduced performance.

Another promising approach for accelerating computation is the development of optical neural networks (ONNs) that could replace electronic hardware [14]. ONNs utilize the inherent parallelism of light, which enables passive manipulation of massive amounts of 2D data at the speed of light [15–18]. Thus, ONNs can offer time of computation that is nearly instantaneous in comparison with those provided by the best electronic hardware available. Specifically, a lens can perform a Fourier transform with $\mathcal{O}(1)$ complexity [19]. In recent years, this promise has resulted in various ONNs for MNIST classification based on a sequence of diffractive masks in the terahertz regime [17], vector-matrix multiplication using integrated photonics [20,21], and hybrid optical-electronic networks leveraging the inherent Fourier transform property of lenses exploited in a $4f$ architecture [18]. While impressive in their own right, the demonstrated ONNs to date have been limited in terms of the complexity of the scenes on which they operate, which have mostly been limited to low-resolution images with simple features like MNIST digits. When applied to more complex scenes (e.g., CIFAR-10), these implementations exhibit low classification accuracy.

A major contributing factor to these limitations is that these networks lack the optical implementation of nonlinear activation and were mostly constrained to linear operations in the optical domain [18]. Achieving an optical nonlinearity requires large optical power, a high-quality-factor resonator, exotic materials, or a combination thereof. While some nonlinear functions are readily available, such as the square operation imparted by a detector, it is unclear how effective it is for achieving comparable performance with that of the ReLU nonlinearity, which is the most common nonlinear activation for CNNs [22,23]. If the nonlinearity is implemented electronically, as a subsequent layer, the electronic signal needs to be converted back to the optical domain to enable additional optical processing. Such repeated signal transductions significantly increase the power consumption and latency, thus obviating the benefits of an ONN versus a more traditional electronic implementation [18].

To compensate for the lack of nonlinearity in an ONN as well as to find an optimal level of performance for a fully linear spectral network, here we develop a knowledge distillation (KD) training methodology to transfer the information from a nonlinear network (teacher) to a SCLC (student). Originally, KD training was introduced for pruning of networks, i.e., knowledge from a large teacher network is transferred to a less complex student model [24,25]. Trained with the KD approach, the student network typically converges faster and obtains better performance than it would achieve without the KD training. A common example for successful KD training is object classification in images. In this problem, the teacher classifies images and provides “soft labels” to the student during training along with the actual labels. A Kullback–Leibler (KL) divergence loss between the soft labels and the student model predictions is then optimized to take into account the teacher’s predictions [26]. KD training is a generic approach and was applied to a variety of problems such as semantic segmentation in which soft labels are used for classification of each pixel [27,28].

In this paper, we adapted the KD training framework to circumvent the need for nonlinearity by “distilling the nonlinearity” from the nonlinear, teacher CNN, to the linear counterpart SCLC, which can be easily implemented using free-space optics. We find that, for boosting student accuracy, the teacher and student networks are required to be as architecturally similar as possible. The KD approach allows the student network to achieve state-of-art performance, exceeding that of previous training methods or networks in the spectral domain, and is also easily amenable to optical implementation because light propagation can be naturally described in Fourier space. The adapted KD training enables us to design a hybrid optical-electronic architecture for the SCLC network, where the optics serve as a linear front end processing unit connected to an electronic back end that typically includes the last layer that corresponds to a specific task. We train this student network with KD training and demonstrate the performance of such a network on two common problems in which CNNs are leading computational methods: object classification and object segmentation. We show that the KD-trained SCLC can achieve performance easily surpassing that of a linear network trained with a standard training approach and nearing the performance of the nonlinear network.

2. METHODS

A. Knowledge Distillation

Knowledge distillation is a machine learning method introduced for compression of neural networks [24]. In particular, it defines the transfer of knowledge from a large neural network model, called the *teacher*, to a small model, called the *student*. We show the schematic flow of KD in Fig. 1. KD assumes that the teacher model is already trained and performs the given task with high accuracy. Then, the student model is trained with both the conventional approach of minimization of the loss between the model prediction and the training data (*student loss*) and, in addition, compares the outcome of the student model with the teacher model through temperature loss (*temp loss*) using the KL divergence [4]. The optimization of both losses is performed through backpropagation, which adjusts the parameters of the student model using stochastic gradient descent [29] or ADAM [30] optimization.

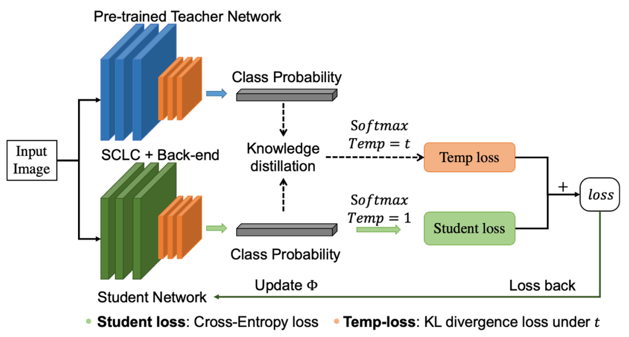

Fig. 1. Illustration of KD training. Student network (green-bottom) parameters Φ are updated according to an interpolation of two losses: the student loss (cross-entropy loss between data and the student model output) and temp loss (cross-entropy loss or KL divergence) between the teacher network (blue-top) and student class distribution with a temperature parameter T.

It was shown that student models trained with the KD approach can achieve significantly higher accuracy than the same model trained on the data alone without a teacher network [26]. The advantage of KD stems from the proposal to compute the temperature loss, which treats the prediction of the teacher model as "soft labels" that inform the student model. In particular, conventional learning considers exclusively the labels in the training data as "hard labels" and for tasks such as classification computes the probabilities distribution vector p^{hl}, where each element p_i^{hl} in the vector corresponds to the probability of the current input belonging to the class i. The softmax function is used to compute probabilities

$$p_i^{\mathrm{hl}} = \exp(z_i) / \sum_j \exp(z_j), \tag{1}$$

where z_i are the student logits after the last fully connected layer. Training with hard labels is a sensitive process, especially for compact networks such as the student model. This typically results in inefficient networks and convergence to poor local optima. To overcome this limitation, KD proposes to add soft labels generated by the teacher model. These labels are the probabilities that the teacher model generates. In particular, for each input, at the same time of computing p^{hl} with the student model, KD computes the soft probabilities vector, p^{sl}, with the teacher model according to

$$p_i^{\mathrm{sl}} = \exp(y_i/T) / \sum_j \exp(y_j/T), \tag{2}$$

where y_i are the logits of the teacher after the last fully connected layer. Through p^{sl}, the teacher model contributes the probability estimates to the student model. This knowledge is unavailable from the data alone and is helpful for the student by providing extra information of the similarities between classes. The similarities are important since these indicate the effective knowledge of the teacher model and subsequently assist in achieving a similar knowledge and performance in the student model. The two probability distributions p^{sl} and p^{hl} are taken into account (as an interpolation) to compute the overall loss used in the training of the student model.

The probabilities contributed by the teacher are defined as soft since the softmax function has a softening parameter, T, named as the distillation temperature. The success of KD depends on the choice of T. In a well-trained teacher model, the correct class has a much higher probability than other classes. For a low value of T, the probability of the correct class will approach 1, while probabilities of other classes will be negligible and will not influence the training, due to p^{sl} being similar to p^{hl}. On the other hand, when T is too high, p_i^{sl} will approach a uniformly distributed vector and will lose the distinction between the correct and incorrect classes. It is therefore important to choose T in between these two extreme cases such that p^{sl} would pass the similarities detected by the teacher to the student.

In practice, the distillation of knowledge from the teacher to the student is particularly effective when the differences in the overall architecture between the two networks are minimal. In network compression, it is typically the case that the architectural blocks are kept the same, and the only change that is implemented is the reduction of the number of neurons in each block. This leads us to the proposition of the implementation of KD between nonlinear and linear CNNs, where, besides exclusion of nonlinear components, the linear network will be kept as similar as possible to the nonlinear one.

B. Spectral CNN Linear Counterpart (SCLC)

We propose to construct a spectral CNN linear counterpart (SCLC), the "student network," to a spectral nonlinear CNN, the "teacher network," based on a free-space optical $4f$ architecture. The teacher network takes the shape of a common CNN designed for generic tasks, e.g., image classification or object segmentation. In the case of classification, we consider CNNs that are structured with multiple layers of repeating operations of convolution, nonlinearity (ReLU), and max-pool (selecting the maximum value), as demonstrated in Fig. 2. In the case of object segmentation, we consider a CNN of a "U" shape [31], where the input passes through similar operations of convolution, ReLU, and max-pool, called convolution layers; in addition, the output of each convolutional layer contracts the input dimension to a "bottle-neck" layer, called "skip connections," from which the representation is expanded with a set of inverse operations, such as transposed convolution, ReLU, and max-pool, altogether called the "transposed convolution layers." To measure the performance of the proposed architectures, we concatenate them with a back end that corresponds to either a classification task (softmax fully connected back end layer) or a segmentation task (sequence of transposed convolution back end layers).

To obtain a network model that is readily realizable with optical components, the network requires a conversion to the spectral domain, such that the input into the model is the Fourier transform of the input image, and the operations such as convolution and pooling are implemented in the spectral domain [18]. Furthermore, the model should not include nonlinearities, since computing those will require an inverse transform and transition from optical to electronic components.

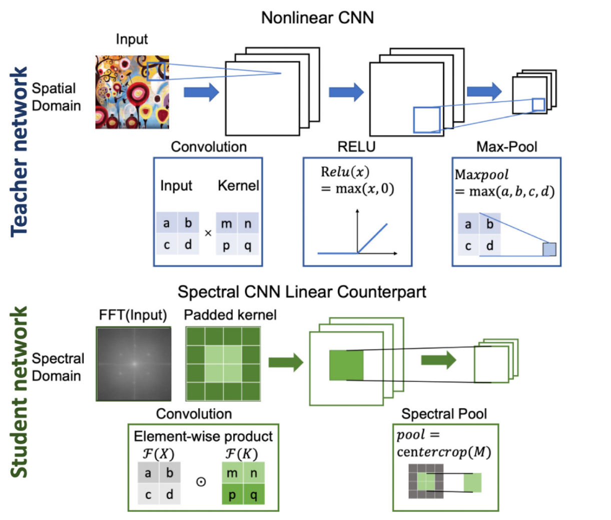

Fig. 2. Nonlinear CNN (top) and the proposed substitute, spectral CNN linear counterpart (SCLC) (bottom). Top: example of nonlinear CNN with layers that include operations of convolution, nonlinear RELU activation, and max-pool. Bottom: SCLC of the CNN shown in top row. The convolution corresponds to the elementwise product in the spectral domain. The RELU operation is excluded. The max-pool layer is represented by a center-crop operation in the spectral domain.

Moreover, as explained, the nonlinear activations will be difficult to implement optically. We describe the building blocks of such a proposed model below.

1. *Convolution in the SCLC*

The convolution is performed as the elementwise product between the input images represented in the spectral domain and the kernels padded to the same dimensions as the inputs. The input into the convolution is a 4D tensor (S, batch size; C, number of input channel; H, height of the image; W, width of the image), and the 2D convolution operation with a stride of 1 is calculated as

$$y(i,j) = x * k = \sum_{h=0}^{H-1}\sum_{w=0}^{W-1} x(h,w) \times k(i-h, j-w), \quad (3)$$

where x is the input image, k is the kernel, and $*$ indicates the convolution operation. We implement the convolution in SCLC in the spectral domain by using an FFT and an elementwise product

$$Y = \mathcal{F}(x * k) = \mathcal{F}(x) \odot \mathcal{F}(k). \quad (4)$$

The function $\mathcal{F}$ denotes the FFT operation, and $\odot$ indicates the elementwise product, which requires the input and the kernels to have the same dimension $H \times W$. During training, the spectral convolution kernel is updated according to

$$\sigma_X = \nabla_X \mathcal{L}|_{X=X_0} = \sigma_Y \odot K_0,$$

$$\delta_k = \nabla_K \mathcal{L}|_{K=\mathcal{F}(0)} = \sigma_Y \odot X_0,$$

$$k_1 = k_0 + \lambda[\mathcal{F}^{-1}(\delta_K)], \quad (5)$$

where σ_X and σ_Y are the errors from the previous and next layers; X_0 and K_0 are, respectively, the input and kernel in the forward propagation; ∇_K and ∇_X are the gradient operators w.r.t. the kernel and input; and $\mathcal{L}$ is the loss function. The updates to the elements in the kernel are computed by applying an inverse FFT (i-FFT) and multiplying by the learning rate λ.

Once the network has been trained, the implementation with spectral convolution components has significant benefits for forward propagation latency for a given input. Indeed, assuming an input image size of (H, W) and square kernel size of (k, k), the complexity of one spatial convolution in the image domain is $\mathcal{O}(HWk^2)$. On the other hand, the spectral convolution composed of elementwise products is of complexity $\mathcal{O}(HW)$. In the optical setup, elementwise operations can be performed in parallel; thus, the runtime, being independent of H and W, is further reduced to $\mathcal{O}(1)$. In addition, the computational complexity of an FFT is $\mathcal{O}(HW\log(HW))$, which brings the overall complexity for the spectral convolution to be $\mathcal{O}(HW\log(HW))$ [10,32,33]. As the proposed network is in the spectral domain, the FFT and i-FFT transforms are needed to be applied at the beginning and at the end of the network for images only, such that the complexity of the intermediate layers will only apply FFT to kennels, which greatly reduces the transformation times. Additionally, significant acceleration occurs in the optical implementation, since the spectral transforms can be achieved through phase transforming components at the speed of light. The combination of convolution via forward and inverse transforms in the optical domain would correspond to instantaneous forward propagation running time of $\mathcal{O}(1)$, when compared with the electronic running time for all layers of the network.

2. *Pooling in the SCLC*

Pooling in the spectral domain needs to take into consideration both mimicking the effect of the standard max-pool used in CNNs and to be practically implementable with optical components. In a nonlinear CNN, common pooling operations are average-pooling or max-pooling, which select the average of the elements or the maximum element, respectively, from the region of the feature map covered by the pooling filter shown in Fig. 2. The purpose of the pooling layers is to reduce the dimensions of the feature maps and to find the representative elements to be transferred forward from the feature map. We propose to substitute the max-pool with a different pooling function, called "spectral pool," which will enable similar functionality in the spectral domain. In particular, we propose to replace the pooling layer with a linear, low-pass filter in the spectral domain. The forward and backward propagation in the layer are defined as

$$y = \mathcal{F}^{-1}(\mathrm{CROP}(\mathcal{F}(x), (H', W'))) \quad (6)$$

$$z = \mathcal{F}^{-1}(\mathrm{PAD}(\mathcal{F}(y/x^*), (H, W))), \quad (7)$$

where the CROP operation in forward propagating keeps the center part of the spectral feature map and reduces the total size from (H, W) to (H', W'). The PAD operation in backward propagation matches the dimension of gradient outputs from (H', W') to (H, W) by padding zeros instead of cropped elements. The $\mathcal{F}$ and $\mathcal{F}^{-1}$ are applied if the networks are in the

Table 1. Comparison of Forward Propagation Computational Complexity between Nonlinear CNN, SCLC Implemented on Electronic Hardware, SCLC Implemented on Optical Hardware

Structure	Nonlinear CNN	SCLC (Computational)	OSCLC (Optical)
Convolution layers	$\mathcal{O}(HWk^2)$	Spatial: $\mathcal{O}(HW\log(HW))$ Spectral: $\mathcal{O}(HW\log(HW))$	$\mathcal{O}(1)$
Pooling layers	$\mathcal{O}(HWk^2)$	Spatial: $\mathcal{O}(k^2\log(k^2))$ Spectral: $\mathcal{O}(1)$	$\mathcal{O}(1)$

spatial domain. These operations are similar to max pooling since the idea of max-pool is to find the representative element in each kernel, while it goes over the feature map and effectively selects the most descriptive features. The spectral-pool drops the high frequencies in the Fourier domain such that it achieves a less noisy output and captures the dominant features, similar to the effect of the conventional max-pool operation. It was shown that training with spectral pooling has better convergence properties compared with max-pooling [11]. Furthermore, spectral pool is of lower computational complexity and is inherently amenable to optical implementation with $\mathcal{O}(1)$ complexity (see Table 1).

Typically, an activation function, such as ReLU, tanh, or σ is applied to extract the features after convolution to avoid extremities in neural units' values and to maintain network stability. These nonlinearities, however, are nearly impossible to achieve in a practical optical implementation. Thus, the SCLC skips the nonlinear activation function to make the network structure amenable to optical implementation. The lack of nonlinearity will be circumvented by the application of KD to maximize the performance achieved using linear operations only.

C. Knowledge Distillation in the SCLC

KD training requires a teacher and student model both performing the same task and exhibiting similarity in their architectures. In previous applications, KD training was applied to model pruning, where the teacher and student models have exactly the same structure with the student having a fraction of the units of the teacher. Here, we extend KD application and consider the teacher and the student models with a similar number of units; however, the student is an adapted version of the operations without the nonlinear activation and has a revised pooling operator. In particular, the teacher model is a pre-trained nonlinear CNN, which is a state-of-the-art system for that particular task. The student network is the SCLC that corresponds to that system with the architectural changes described in the previous sections. The objective of KD training is to optimize the total loss subject to updating the weights, Φ, of the SCLC student model only. The total loss is a linear combination of the temperature loss and student loss. The weights are updated according to implementation of backpropagation that optimizes the total loss.

We select the temperature loss to be the KL function between the soft labels ($p^{\mathrm{sl},t}$) from the pre-trained nonlinear CNN model (*teacher*) and predictions ($p^{\mathrm{sl},s}$) from the SCLC (*student*), both distilled with the same temperature T. We chose KL loss over cross-entropy because it includes an extra penalty on the direction of the loss, which facilitates convergence. The student loss is the standard cross-entropy loss between the data labels and SCLC probabilities (p^{hl}). The total loss is then calculated as a weighted summation of the two losses:

$$\mathrm{Loss}(x,\Phi)=\alpha\mathcal{L}_C(y,p^{\mathrm{hl}})+(1-\alpha)\mathcal{L}_K \times((p^{\mathrm{sl},t};T=\tau),(p^{\mathrm{sl},s};T=\tau)), \quad (8)$$

where x corresponds to the input, y is the training data, Φ are student model weights, $\mathcal{L}_C$ is the cross-entropy loss function, $\mathcal{L}_K$ is the KL divergence loss function, p^{hl} corresponds to the student model hard predictions, $p^{sl,s}$ corresponds to the student predictions under given teacher model probabilities $p^{\mathrm{sl},t}$, and α is the weighting parameter.

D. Optical Implementation of the SCLC (OSCLC)

In this section, we explore how the proposed SCLC network can potentially be implemented using free-space optics, i.e., optical SCLC (OSCLC). The choice of free-space optics is motivated by the large number of information channels available to us in such an implementation [23]. There are three components of the OSCLC that need to be considered for optical implementation: convolutional layers, spectral pooling, and summation of different channels.

Each convolutional layer can be implemented by a $4f$ correlator architecture to further accelerate the forward propagation speed (Fig. 3) [15,34–36]. A typical $4f$ correlator architecture comprises two lenses of equal focal length spaced apart at $2f$ distance and with input and output planes located in the front and the back focal planes of the first and second lenses, respectively. The first lens produces a Fourier transform of the input scene at the focal plane. A complex-values phase-mask placed in that focal plane provides the pointwise multiplication implementing the convolution, and the second lens performs the inverse Fourier transform. Therefore, a $4f$ correlator architecture is able to function as an equivalent architecture for a single channel of a linear spectral CNN counterpart. We note that, by using coherent light and phase-only spatial light modulators, we can handle the negative weights. By creating a lenslet array and an array of convolutional phase-masks, we can parallelize all the convolutional operations. Thus, the computational complexity decreases to $\mathcal{O}(1)$ for any operations in the spectral domain and will not grow exponentially with the input image resolution/pixels. Based on our simulation, with 1 mm center-to-distance between 3 mm focal length lenses in an array, we have negligible cross-talk between the channels [18]. The spectral pooling in the OSCLC can also be implemented by a low-pass filter, where we block high-frequency components in the Fourier plane. Essentially, we can use a $4f$ correlator, with an amplitude mask in the Fourier plane. The highest-frequency components in the image correspond to the largest wavevectors, which, in the Fourier plane, are distributed farther from the optical axis. By placing a circular aperture in the Fourier plane, we will block frequency components that reside outside of the central aperture, thus enabling us to perform spectral pooling

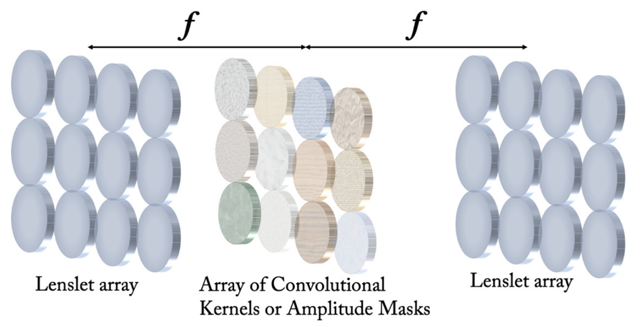

Fig. 3. Using a lenslet array and array of phase and amplitude masks, we can implement all the convolutions in a parallel fashion. We can also implement spectral pooling in such a structure by placing an amplitude mask in the Fourier plane. The lenslet array and convolutional masks can be implemented using meta-optics.

optically. This approach is also amenable to variants of our defined low-pass spectral pooling operator. For example, we could implement a high-pass filter by inverting the trasmittance of the Fourier plane mask to be opaque at the center and transparent outside this region. Alternatively, if frequency components from the whole range at the Fourier plane are desired, we could instead use a mask based on concentric transparent annuli, which corresponds to a spectral pooling operator that passes frequency components only at certain frequency intervals. An additional consideration that must be made with this architecture is the field of view of the lenses in the lenslet array. In order to limit crosstalk between the different convolution channels, the field of view will be constrained so that the convolutions at the output plane will be nonoverlapping; this is achievable using an array of field stops positioned at the input plane.

The summation of different channels is also a basic principle in deep neural networks and is widely used in object classification and segmentation. If there are no summations after elementwise products, the channels will grow exponentially, requiring a massive number of kernels after only a few convolutional layers, which makes it impossible for real optical implementations. Although fewer kernels in the convolution layers can alleviate this challenge, the overall accuracy, especially in complex scenarios, would suffer. Such a summation can be implemented using various techniques used for coherent beam combining [37,38]. We note that, while free-space optics tends to be bulky and prone to misalignment, recent demonstrations of meta-optics and volume optics [39,40] exhibit complicated free-space optics in a compact form factor, possibly in a monolithic fashion mitigating any misalignment. We note that, in the current paper, for simulation as assumed perfect alignment and aberration-free optics, which are ideal conditions, and future works will explore analysis of the proposed architecture with realistic optics and their robustness against experimental imperfections.

3. COMPUTATIONAL EXPERIMENTS

We evaluate our proposed SCLC on two distinct tasks involving image inputs: (i) object classification and (ii) object segmentation. The object classification task associates each image with a class that corresponds to the object photographed in the image. For example, the task could be to classify whether the image depicts a cat, dog, or car. Object segmentation is a pixel-level classification task, which aims to determine for each pixel in a given image to which class it belongs. Effectively, object segmentation partitions the input image into regions that correspond to classes. For example, in the case of a photograph of a person, regions would correspond to the background, face, hair, etc. Our evaluation consists of determining the accuracy and the forward propagation runtime of the SCLC and OSCLC and compare them with that of a standard CNN.

In our experiment, we select AlexNet as the standard CNN and the teacher with with input dimensions of $3 \times 224 \times 224$ [where the first dimension corresponds to R,G,B channels and the other two are height (H) and width (W) of the image]. The linear counterpart of AlexNet (SCLC) front end is when nonlinearities are removed resulting in five convolutional layers, each followed by a spectral pooling layer, which is then connected to a fully connected back end of $256 \times 6 \times 6$ dimensions. The input into the SCLC front end is the Fourier transform of the input image and corresponds to the same dimension $3 \times 224 \times 224$ spectral representation of the input image. Each convolutional layer in the spectral domain corresponds to an elementwise tensor product between the input (previous layer output) and the Fourier transform of the kernel padded to the same size of the input. Following AlexNet choice of kernels, the tensors in each layer are $32 \times 224 \times 224$, $64 \times 113 \times 113$, $128 \times 54 \times 54$, $256 \times 27 \times 27$, and $256 \times 13 \times 13$, respectively (see Table 2 for the summary). The input to the back end is a vector that corresponds to a flattened output tensor of the front end of dimensions $256 \times 6 \times 6$, i.e., a vector of 9216.

In image segmentation experiments, we adopt the linear counterpart of a simplified U-Net with input size of 960×640 or smaller. This U-Net has three components: 1) convolution layers; 2) transposed convolution layers; and 3) skip connections. The SCLC implements only the convolution layers [component (1)]. The other two components are intended to be implemented with a standard electronic back end. The implementation of the convolution layers is similar to image classification problems. The convolutional layers contain four double SCLC layers (where each double layer results in two sequential SCLC layers) and four spectral pooling layers.

We test the performance metrics on a variety of common benchmarks for each task. For object classification, we perform computational experiments on (i) Kaggle's cats and dogs challenge [41], (ii) Cifar-10 [42], and (iii) HIGH-10 (a subset of ImageNet) [43]. For object segmentation, we perform computational experiments on (i) Kaggle's carvana image masking challenge (cars segmentation) [44], (ii) face recognition [45] and (iii) VOC2012 [46]. Training of all networks is performed on a Tesla P100 GPU with the Google Colab platform. For all networks, the initial learning rate for student networks is 0.0001 with momentum 0.9 and weight decay 0.0005. We set the batch size according to the available memory: 16 for the classification task and 1 for the segmentation task. We use the cross-entropy loss to calculate the student loss, and KL divergence to calculate the temp loss.

Table 2. Architecture of the Network Backbone

AlexNet				SCLC		
Layer	**Input**	**Kernel**	**Output**	**Layer**	**Input**	**Elementwise Tensor**
Convolution1	3 × 224 × 224	3 × 64 × 11 × 11	64 × 55 × 55	SCLC1	3 × 224 × 224	3 × 32 × 224 × 224
Pooling1	64 × 55 × 55	[3,3,64]	64 × 27 × 27	Pooling1	32 × 224 × 224	32 × 113 × 113
Convolution2	64 × 27 × 27	64 × 192 × 5 × 5	192 × 27 × 27	SCLC2	32 × 113 × 113	32 × 64 × 113 × 113
Pooling2	192 × 27 × 27	[3,3,192]	192 × 13 × 13	Pooling2	64 × 113 × 113	64 × 55 × 55
Convolution3	192 × 13 × 13	192 × 384 × 5 × 5	384 × 13 × 13	SCLC3	64 × 55 × 55	64 × 128 × 55 × 55
Pooling3	384 × 13 × 13	[3,3,384]	384 × 13 × 13	Pooling3	128 × 55 × 55	128 × 27 × 27
Convolution4	384 × 13 × 13	384 × 256 × 13 × 13	256 × 13 × 13	SCLC4	128 × 27 × 27	128 × 256 × 27 × 27
Pooling4	256 × 13 × 13	[3,3,256]	256 × 13 × 13	Pooling4	256 × 27 × 27	256 × 13 × 13
Convolution5	256 × 13 × 13	256 × 256 × 3 × 3	256 × 13 × 13	SCLC5	256 × 13 × 13	256 × 256 × 13 × 13
Pooling5	256 × 13 × 13	[3,3,256]	256 × 6 × 6	Pooling5	256 × 13 × 13	256 × 6 × 6

Back End		
Layer	Input	Output
FC1	9216	1024
FC2	1024	256
FC3	256	10

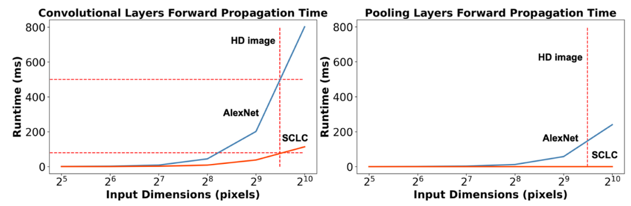

Fig. 4. Comparison of forward propagation time between AlexNet (blue line) and its SCLC (red line). Left: Propagation time through a single convolutional layer for variable image dimensions. Right: Propagation time through a single pooling layer for variable image dimensions. The dashed lines mark the resolution of an HD image and its corresponding runtime in AlexNet and SCLC.

A. Spectral Counterpart Layer Efficiency

We first investigate the runtime of forward propagation when variable input dimensions (32 × 32 to 1024 × 1024) are considered for AlexNet (nonlinear CNN) versus SCLC, and show our results in Fig. 4. We evaluate forward propagation time for convolutional layers and pooling layers separately. Forward propagation time of convolutional layers in AlexNet grows exponentially with input size. The runtime of the SCLC grows exponentially as well but with a significantly slower rate. For example, for high-definition (HD) input size image, the runtime of the SCLC convolution corresponds to 100 ms versus 500 ms for AlexNet convolution. For pooling layers, AlexNet forward propagation runtime grows exponentially with input size with a smaller rate than the convolution. For the SCLC, due to the pooling being spectral, the runtime is constant and independent.

B. Object Classification Task

We evaluate our proposed SCLC on different classification datasets to estimate the accuracy that SCLC can achieve when trained with KD. We compare the accuracy with accuracy of the teacher network, AlexNet. The first data set that we consider is the Kaggle cats and dogs challenge [41], which consists of 125,000 images of dimensions 96 × 96 associated with two classes: a cat or a dog. An additional data set that we consider is Cifar-10 [42], which consists of 60,000 images of dimension of 32 × 32, including 10 classes. The third data set that we consider is High-10, which is a subset of ImageNet [43] and consists of approximately 10,000 annotated images, equally distributed 10 classes with images of resolution 500 × 300. During training of AlexNet, all images are resized or cropped to dimensions of 224 × 224 to match AlexNet's input size. To evaluate the performance of the SCLC in classification, we add a single fully connected back end layer to it. The back end layer will be implemented in electronics for the OSCLC since the last layer will include nonlinearity. We employ the KD approach to train the SCLC with AlexNet being the teacher network; when the training converges, we test SCLC variants against AlexNet.

In particular, we compare the SCLC trained with and without the KD approach and a variation of AlexNet with a square nonlinearity that could be realized optically using detectors. We show the results of the comparison for the three benchmarks in Table 3. We observe that, for all benchmarks, KD training significantly enhances the accuracy of the classification achieved by the SCLC (by 12.5% on average). Indeed, KD contribution appears to be essential in generating an SCLC network with robust accuracy. On Kaggle cats and dogs classification, SCLC achieves 90.60% (6% below the accuracy of AlexNet); on Cifar-10 classification, it achieves 80.80% (5% below the accuracy of AlexNet) and on HIGH-10 classification it achieves 81.4% (≈ 12% below the accuracy of AlexNet). These results are encouraging since the SCLC trained with standard training has a much bigger gap of 16%, 20%, and 23% between its accuracy and the accuracy of AlexNet. The KD approach appears to close this gap by more than half for all benchmarks. Furthermore, when the RELU nonlinearity is modified to square nonlinearity (SQ-AlexNet), both KD and standard training do not result with sufficiently accurate network. This is because the square nonlinearity will magnify the model's parameters. This observation indicates that the KD approach is effective in regimes where

Table 3. Forward Propagation Accuracy and Processing-Time of Different Network Variants for Classification Tasks[a]

Dataset	Model (input: 224 x 224)	Accuracy	Runtime (ms/img)	Optical Runtime (ms/img)
Cats versus Dogs	AlexNet	96.10%	350.66	–
	SCLC	79.50%	11.05	0.61
	SCLC + KD	90.60% (+11.10%)	11.05	0.61
	SQ-Nonlinear	51.70%	12.04	0.61
	SQ-Nonlinear + KD	51.72%	12.04	0.61
Cifar-10	AlexNet	85.09%	350.66	–
	SCLC	65.45%	11.05	0.61
	SCLC + KD	80.80% (+15.35%)	11.05	0.61
High-10	AlexNet	93.95%	350.66	–
	SCLC	70.12%	11.05	0.61
	SCLC + KD	81.40% (+11.28%)	11.05	0.61

[a]AlexNet: nonlinear baseline network (no optical implementation), SCLC: The linear counterpart of AlexNet.

the network architecture of the student is kept as close to the teacher as possible.

Notably, to match AlexNet input dimensions, all experiments were implemented with the same input image resolution of 224 × 224. We therefore explore with the HIGH-10 data set (which includes higher-resolution images) how accuracy varies if the input resolution is increased. For each input image, its resolution is downsampled to 224 × 224 to train the teacher (AlexNet). Then, the student (SCLC/OSCLC) is trained under the supervision of the teacher model. We show in Fig. 5 that increasing the resolution of the input increases the accuracy of both the teacher and student models (with a linear rate). SCLC trained with higher-resolution inputs can surpass the accuracy of the teacher network trained with a lower-resolution data set, e.g., the SCLC with input of $2^7 \times 2^7$ dimensions performs similarly (≈80%) to the teacher with input of $2^6 \times 2^6$ dimensions. While the accuracy of the two is similar, the runtime of the SCLC is expected to be more favorable, since the teacher's runtime grows exponentially. It is impossible to continually increase the input dimension for nonlinear models. Higher-resolution inputs require higher computational power and longer processing time, which makes it impractical for mobile GPUs or any real-time usages. The computational efficiency and computational power increase at much lower rates or are even constant when increasing the input dimension. Indeed, we show that for 80% accuracy both the SCLC and OSCLC are at least five times faster than (≈5 ms and ≈ 1 ms) than AlexNet (≈25 ms).

We further demonstrate the efficient runtime of the SCLC compared with AlexNet in Table 3 Columns 3 and 4 for input size of 224 × 224. While AlexNet runtime for a single image is 350 ms, SCLC runtime drops by an order of magnitude to 11 ms for a single image. Simulated runtime of OSCLC drops by another order of magnitude to 0.6 ms. The estimation of OSCLC forward propagation runtime includes three parts:

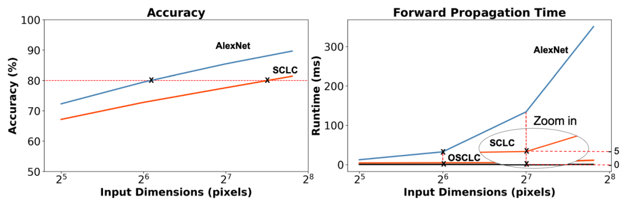

Fig. 5. Accuracy and forward propagation time for classification task on High-10 data set. Left: AlexNet and SCLC accuracy for varying input resolution. Right: Forward propagation runtime of AlexNet, SCLC, and OSCLC for varying input resolution.

Table 4. Ablation Studies of Pooling Components, SCLC, Back End, and KD Training

Structure	Accuracy
Max pooling	68.41%
Spectral pooling	70.12% (+1.71%)
Back end only	41.40%
SCLC (front end) + back end	70.12% (+28.72%)
SCLC (front end) + back end + KD	81.40% (+11.28%)

1) running time of the optical structure; 2) transduction time between optics and electronics; and 3) running time of the electronic back end. Since light propagation is short, ≈ ps, the main contribution comes from the signal transduction (≈ 0.32 ms for 100 kb images via USB 3.0 protocol at a rate of 2500 Mbit/s) and the back end propagation time (≈ 0.28 ms for a single image on a GPU (Tesla P100).

To demonstrate how different components contribute to SCLC performance, we investigated ablations of the SCLC architecture (Table 4). We first compared spectral pooling with max pooling in the SCLC. We find that spectral pooling corresponds to faster convergence and to 1.71% higher accuracy than max pooling. From our observation and classical image processing studies, the information in the natural images is concentrated in the lower frequencies, while higher frequencies tend to encode noise [11]. In addition, we test the contribution of the front end (SCLC), the back end and the KD training of them. When only the back end (linear layer and softmax function) is considered, the accuracy is 41%. Adding the front end to it and training the model on the same data set corresponds to an increase of ≈ 30% such that the accuracy becomes 70.12%. Further application of KD training of both the front end and the back end boosts the accuracy and in convergence, such that the accuracy is enhanced by another ≈ 10% to 81.4%.

C. Object Segmentation Task

In addition to image classification, we explore the construction of an SCLC for object segmentation as shown in Fig. 6. For such a task, a standard CNN is of a U-shape with a sequence of convolutions, transposed convolutions and skip connections. During training, we choose the teacher network to be a U-Net [31]. KD training corresponds to pixel-level loss for both soft predictions and soft labels. The front end for this task is the sequence of contracting convolutions while the back end corresponds to a sequence of transposed convolutions.

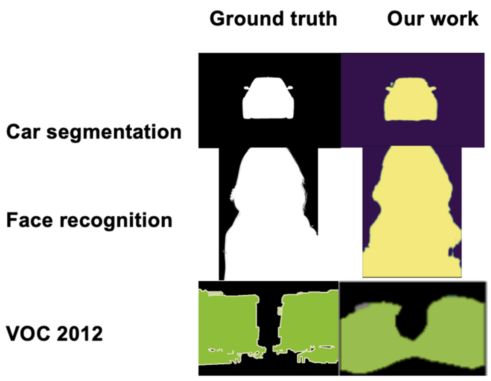

Fig. 6. Examples of object segmentation for the considered benchmarks of (i) car segmentation, (ii) face recognition, and (iii) VOC2012. Left to right: Input image from data sets, ground truth, our work (SCLC/OSCLC).

Table 5. Accuracy and Forward Propagation Rate (Frames per Second) for Object Segmentation Task Tested on Three Benchmarks

Dataset	Network	Accuracy	Rate (Higher Is Better) (fps)
Car Segmentation	U-Net	98.02%	7.36
	SCLC	97.1%	12.7
Face Recognition	U-Net	95.58%	9.38
	SCLC	91.39%	15.6
VOC2012	U-Net	75.51%	7.36
	SCLC	62.21%	12.7

The first benchmark that we consider is the Kaggle carvana image masking challenge, which consists of 5088 cars with an original resolution of 1920×1280. The data set is randomly split into 4580 and 508 images for training and testing, respectively. Our second benchmark is the face recognition data set, which consists of 2000 images with 1700 for training and 300 for testing. Our third benchmark is VOC2012, which consists of 2913 images of original resolution 500×375, which includes 20 classes and one background class. The images in all data sets are downsampled to 960×640 or smaller due to limitations of GPU memory.

Our results are shown in Table 5. For both car and face data sets, we observe that the SCLC performs relatively well and obtains accuracy that falls from that of the teacher U-Net by only a few percent. We observe that consideration of higher-resolution inputs corresponds to enhanced accuracy in classification experiments. For the VOC2012 data set, which includes lower-resolution images and has more segmentation classes, both the teacher and SCLC perform with rather low accuracy below 80%. This example demonstrates that, in such problems, it is crucial to consider the fully available image resolution. We compare the computational efficiency of the SCLC against the teacher in terms of frames per second rate and find that the SCLC is approximately $2\times$ faster than U-Net. While such a speedup is more modest than of the image classification task, the rate is closer to a real-time operation rate (30 fps) or alternatively allows for consideration of larger input images that may correspond to enhanced segmentation with the same frame rate as that of the teacher. The main contribution to SCLC runtime is the electronic back end that consists of transposed convolution layers, which have a computational complexity of $\mathcal{O}(HWk^2)$ (compared with image classification back end, which is a fully connected single layer).

4. DISCUSSION

The SCLC network is a multilayered model in spectral space, where each layer consists of a matrix multiplication and spectral pooling (prior to the back end). Theoretically, the convolution layers and their corresponding operation in the spectral space can be implemented as a single matrix that implements an elementwise product operation with the input. In particular, the convolutional kernels in the spectral domain correspond to an elementwise product between the Fourier transform of the input and the Fourier transform of the kernel padded to the same size of the input image. The elementwise product yields a matrix for each layer, and the product of the sequence of matrices generates a single matrix reflecting elementwise product with the input:

$$O = x \odot k_1 \odot \ldots \odot k_i \odot \ldots \odot k_n, \; i = 1, 2, \ldots n, \qquad (9)$$

where O is the output, x is the input image, and the k_i is the padded parameter in each layer. The spectral pooling is also reduced to a single operation, where it crops the center part of this matrix:

$$\begin{aligned} O_{\mathrm{CROP}} &= \mathrm{CROP}(O) \\ &= \mathrm{CROP}(x \odot k_1 \odot \ldots \odot k_i \odot \ldots \odot k_n), \quad i = 1, 2, \ldots n. \end{aligned} \qquad (10)$$

Such a setup could be instrumental in implementation of the optical network (OSCLC) for operation on given inputs (i.e., forward inference). Notably, such a structure would not be applicable during training. The reason stems from the fact that each element in the outcome matrix incorporates multiple parameters that originally correspond to multiple layers and being efficiently trained using the KD approach that we propose in this work. As our experiments indicate, for effective approximation of the performance of the baseline AlexNet by KD, the SCLC needs to have as similar structure as possible to the original nonlinear network structure (i.e., same number of layers). Therefore, an approach such as KD is not directly applicable to train the much smaller number of elements in the outcome single matrix.

We also studied deeper and more complex teacher networks than AlexNet, such as VGG16 and ResNet18, to examine whether the KD could distill "additional knowledge" from these networks and improve the performance of the SCLC counterpart of AlexNet. Table 6 shows the results of training the SCLC with three different teacher networks: AlexNet, VGG16, and ResNet18 on the High-10 data set. As can be observed, even when the SCLC student is trained with a better performing teacher, SCLC still has the best performance when the teacher network is the network from which it was originated (i.e., AlexNet). These results reaffirm the need for the student and the teacher networks to be as architecturally similar as possible to boost student accuracy.

While we have shown that the SCLC counterpart reaches accuracy and potential speedup through OSCLC in forward

Table 6. Accuracy of AlexNet SCLC with Different Teacher Models

Teacher Model	SCLC Accuracy
ResNet18	78.50%
VGG16	79.80%
AlexNet	81.40%

inference, these results are for networks that are not too deep, such as AlexNet. When the number of layers increases to a much deeper network, such performance is not guaranteed to persist. The reason for this limitation is that it is unclear how to perform the normalization or skip connections operations in the optical setting. Notably, since optical setup would be mostly applicable to real-time processing situations, operations such as batch-normalization may not be required since the batch size for these applications will be of size 1. However, as deep learning literature indicates, incorporation of normalization is critical for improving the network's overall performance.

5. CONCLUSIONS

Over the last decade, various CNNs have been developed and deployed as robust systems for ubiquitous visual tasks. While such incorporation in multiple applications is remarkable, scaling up CNNs for tasks that require high speed and energy efficiency entail significant challenges. This is due to the fact that the computational runtime of convolutional layers increases exponentially with input size. Such a constraint eventually leads to CNNs exceeding the limit of power consumption; as such, they limit the accuracy and rate at which the system can perform. It is often the case that these limitations prohibit real-time deployment of CNNs. Representation of CNNs in the spectral domain in conjunction with an optical platform that supports parallel, elementwise product computations has the potential to overcome the exponential increase in computation runtime. However, since transitions from optical to electronic systems are computationally inefficient, it is required to compute the additional, typically nonlinear layers of CNNs with the same optical platform. Unfortunately, performing such operations with optical components is currently impractical. To that end, we propose a spectral linear version of CNN (SCLC) that is also optically realizable (OSCLC) with free-space optical components and investigate whether it can serve as an effective counterpart for CNN. While it is indeed possible to design such a system, we show that the accuracy significantly drops when nonlinear activations are not present. To address the gap in accuracy between CNN and its SCLC, we propose a novel training approach based on knowledge distillation (KD). The approach assists in training the SCLC by involving the original CNN in the training as the teacher in addition to the standard optimization with the training data. We show that such training is promising in circumventing the need for a nonlinearity and represents a generic training procedure that could be applied to various CNNs and their respective SCLCs. In particular, we consider two visual tasks and six benchmarks and show that KD training applied to training a SCLC and an OSCLC boosts its accuracy by more than half of the gap created by exclusion of nonlinearities. We further show that increasing input dimensions has almost no impact on computational efficiency for OSCLC; however, increasing dimensions is able to further close the accuracy gap or even surpass the accuracy of CNN operating with low resolution inputs.

While our focus here is on realizing an optically viable architecture for an SCLC, it is notable that our analysis and experiments indicate a computational benefit for an SCLC trained with KD even for standard electronic components such as GPU. For the SCLC, we find that keeping the back-end layer, which is implemented electronically, as minimal as possible is critical to fully leverage the benefits of optical parallelism and its speed. Indeed, as we demonstrate in our simulations, in the image classification task, in which the back end consists of a single layer, optical forward propagation reduces the runtime for standard input image dimensions by two orders of magnitude. In the object segmentation task, the back end is more complex. It consists of transposed convolutions; thus, while a $2\times$ computational speedup is achieved, it is not as significant as the speedup in the case of the classification task given its increased complexity of effectively performing many classification tasks throughout an image. We expect that this reduced benefit for object segmentation can be mitigated with further innovation in the design of meta-optics by exploiting spatially large kernels that are computationally costly in software and hyperspectral kernels that can extract information optically that is never exploited in a traditional network architecture in which colors can only be separated into RGB bins. Furthermore, our investigation indicates that key components of CNNs, such as convolution and pooling, can indeed be adapted to the free-space optical domain; for such adaptation, the training process needs to be an integrated training procedure that involves both the original CNN, its adapted counterpart, and the electronic back end. We show that KD training presents a plausible integrated training procedure for these purposes. We note that such end-to-end design is already being pursued in optics, including the emerging field of meta-optics. Adaptation of additional components alongside further development of KD-based training may pave the way toward optical-electronic deployment of CNNs to provide a further leap in enhancement of CNN performance for complex and real-time tasks.

Funding. National Science Foundation HDR Institute Accelerated AI Algorithms for Data-Driven Discovery (Grant No. OAC-2117997); Washington Research Foundation; National Science Foundation (Grant No. 2127325).

Acknowledgment. ES acknowledges the support of National Science Foundation HDR Institute Accelerated AI Algorithms for Data-Driven Discovery Grant No. OAC-2117997, Washington Research Foundation Fund and the partial support by the Departments of Applied Mathematics, Electrical & Computer Engineering, the Center of Computational Neuroscience (CNC), and the eScience Center at the University of Washington. AM acknowledges the support of National Science Foundation Grant No. 2127235, and AM and SC acknowledge the support by Washington Research Foundation Fund, UW Reality Lab, Facebook, Google, Amazon, and Futurewei.

Disclosures. The authors declare no conflicts of interest.

Data availability. The data that support the findings of this study are available within the article. The code for KD training of the SCLC is available in Ref. [47].

REFERENCES

1. E. Kakkava, N. Borhani, Y. Pu, C. Moser, and D. Psaltis, “Image classification and reconstruction through multimode fibers by deep neural networks,” in *Conference on Lasers and Electro-Optics Pacific Rim (CLEO-PR)* (IEEE, 2018), pp. 1–2.
2. K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 770–778.
3. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2015), pp. 1–9.
4. I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning* (MIT, 2016).
5. K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” arXiv:1409.1556 (2014).
6. S. Mittal, “A survey on optimized implementation of deep learning models on the NVIDIA Jetson platform,” J. Syst. Archit. **97**, 428–442 (2019).
7. M. Mathieu, M. Henaff, and Y. LeCun, “Fast training of convolutional networks through FFTs,” arXiv:1312.5851 (2013).
8. N. Vasilache, J. Johnson, M. Mathieu, S. Chintala, S. Piantino, and Y. LeCun, “Fast convolutional nets with FBFFT: A GPU performance evaluation,” arXiv:1412.7580 (2014).
9. R. Kondor, Z. Lin, and S. Trivedi, “Clebsch–Gordan nets: a fully Fourier space spherical convolutional neural network,” in *Advances in Neural Information Processing Systems* (2018), pp. 10117–10126.
10. H. Pratt, B. Williams, F. Coenen, and Y. Zheng, “FCNN: Fourier convolutional neural networks,” in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (Springer, 2017), pp. 786–798.
11. O. Rippel, J. Snoek, and R. P. Adams, “Spectral representations for convolutional neural networks,” in *Advances in Neural Information Processing Systems* (2015), pp. 2449–2457.
12. B. Guan, J. Zhang, W. A. Sethares, R. Kijowski, and F. Liu, “SpecNet: spectral domain convolutional neural network,” arXiv:1905.10915 (2019).
13. J. Li, S. You, and A. Robles-Kelly, “A frequency domain neural network for fast image super-resolution,” in *International Joint Conference on Neural Networks (IJCNN)* (IEEE, 2018), pp. 1–8.
14. G. Wetzstein, A. Ozcan, S. Gigan, S. Fan, D. Englund, M. Soljačić, C. Denz, D. A. Miller, and D. Psaltis, “Inference in artificial intelligence with deep optics and photonics,” Nature **588**, 39–47 (2020).
15. L. Cutrona, E. Leith, C. Palermo, and L. Porcello, “Optical data processing and filtering systems,” IRE Trans. Inf. Theory **6**, 386–400 (1960).
16. J. Bueno, S. Maktoobi, L. Froehly, I. Fischer, M. Jacquot, L. Larger, and D. Brunner, “Reinforcement learning in a large-scale photonic recurrent neural network,” Optica **5**, 756–760 (2018).
17. 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**, 1004–1008 (2018).
18. S. Colburn, Y. Chu, E. Shilzerman, and A. Majumdar, “Optical frontend for a convolutional neural network,” Appl. Opt. **58**, 3179–3186 (2019).
19. J. W. Goodman, *Introduction to Fourier Optics* (Roberts and Company, 2005).
20. J. Feldmann, N. Youngblood, M. Karpov, H. Gehring, X. Li, M. Stappers, M. Le Gallo, X. Fu, A. Lukashchuk, A. Raja, J. Liu, C. D. Wright, A. Sebastian, T. J. Kippenberg, W. H. P. Pernice, and H. Bhaskaran, “Parallel convolutional processing using an integrated photonic tensor core,” Nature **589**, 52–58 (2021).
21. Y. Shen, N. C. Harris, S. Skirlo, M. Prabhu, T. Baehr-Jones, M. Hochberg, X. Sun, S. Zhao, H. Larochelle, D. Englund, and M. Soljačić, “Deep learning with coherent nanophotonic circuits,” Nat. Photonics **11**, 441–446 (2017).
22. D. Englund, A. Majumdar, M. Bajcsy, A. Faraon, P. Petroff, and J. Vučković, “Ultrafast photon-photon interaction in a strongly coupled quantum dot-cavity system,” Phys. Rev. Lett. **108**, 093604 (2012).
23. A. Ryou, J. Whitehead, M. Zhelyeznyakov, P. Anderson, C. Keskin, M. Bajcsy, and A. Majumdar, “Free-space optical neural network based on thermal atomic nonlinearity,” Photon. Res. **9**, B128–B134 (2021).
24. G. Hinton, O. Vinyals, and J. Dean, “Distilling the knowledge in a neural network,” arXiv:1503.02531 (2015).
25. A. Mishra and D. Marr, “Apprentice: using knowledge distillation techniques to improve low-precision network accuracy,” arXiv:1711.05852 (2017).
26. M. Phuong and C. Lampert, “Towards understanding knowledge distillation,” in *International Conference on Machine Learning* (2019), pp. 5142–5151.
27. Q. Li, S. Jin, and J. Yan, “Mimicking very efficient network for object detection,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 6356–6364.
28. Y. Liu, K. Chen, C. Liu, Z. Qin, Z. Luo, and J. Wang, “Structured knowledge distillation for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 2604–2613.
29. L. Bottou, “Stochastic gradient descent tricks,” in *Neural Networks: Tricks of the Trade* (Springer, 2012), pp. 421–436.
30. D. P. Kingma and J. Ba, “Adam: a method for stochastic optimization,” arXiv:1412.6980 (2014).
31. O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention* (Springer, 2015), pp. 234–241.
32. J.-H. Lee, M. Heo, K.-R. Kim, and C.-S. Kim, “Single-image depth estimation based on Fourier domain analysis,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 330–339.
33. A. Kappeler, S. Ghosh, J. Holloway, O. Cossairt, and A. Katsaggelos, “PtychNet: CNN based Fourier ptychography,” in *IEEE International Conference on Image Processing (ICIP)* (IEEE, 2017), pp. 1712–1716.
34. D. Psaltis, D. Brady, X.-G. Gu, and S. Lin, “Holography in artificial neural networks,” in *Landmark Papers on Photorefractive Nonlinear Optics* (World Scientific, 1995), pp. 541–546.
35. T. Lu, S. Wu, X. Xu, and T. Francis, “Two-dimensional programmable optical neural network,” Appl. Opt. **28**, 4908–4913 (1989).
36. D. Psaltis, D. Brady, and K. Wagner, “Adaptive optical networks using photorefractive crystals,” Appl. Opt. **27**, 1752–1759 (1988).
37. M. Prossotowicz, A. Heimes, D. Flamm, F. Jansen, H.-J. Otto, A. Budnicki, A. Killi, and U. Morgner, “Coherent beam combining with microlens arrays,” Opt. Lett. **45**, 6728–6731 (2020).
38. I. Fsaifes, L. Daniault, S. Bellanger, M. Veinhard, J. Bourderionnet, C. Larat, E. Lallier, E. Durand, A. Brignon, and J.-C. Chanteloup, “Coherent beam combining of 61 femtosecond fiber amplifiers,” Opt. Express **28**, 20152–20161 (2020).
39. T. D. Gerke and R. Piestun, “Aperiodic volume optics,” Nat. Photonics **4**, 188–193 (2010).
40. A. Zhan, S. Colburn, C. M. Dodson, and A. Majumdar, “Metasurface freeform nanophotonics,” Sci. Rep. **7**, 1673 (2017).
41. “Kaggle dogs vs. cats redux: Kernels edition,” 2016, https://www.kaggle.com/c/dogs-vs-cats.
42. A. Krizhevsky, “Learning multiple layers of features 847 from tiny images,” Technical Report (2009).
43. A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in Neural Information Processing Systems* (2012), Vol. **25**, p. 1097.
44. “Carvana image masking challenge,” 2017, https://www.kaggle.com/c/carvana-image-masking-challenge.
45. X. Shen, X. Tao, H. Gao, C. Zhou, and J. Jia, “Deep automatic portrait matting,” in *European Conference on Computer Vision* (Springer, 2016), pp. 92–107.
46. M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, “The PASCAL visual object classes challenge 2012 (VOC2012) results,” 2012, http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.
47. J. Xiang, S. Colburn, A. Majumdar, and E. Shlizerman, “SCLCKD training repository,” GitHub, 2022, https://github.com/shlizee/SCLCKDTraining.