On-Chip Optical Convolutional Neural Networks

Hengameh Bagherian,[1,*] Scott Skirlo,[1] Yichen Shen,[1,†]
Huaiyu Meng, [2] Vladimir Ceperic, [1,3] and Marin Soljačić[1]

[1] *Massachusetts Institute of Technology, Department of Physics, Cambridge, MA 02139, USA*
[2] *Research Laboratory of Electronics, Massachusetts Institute of Technology, Cambridge, MA 02139, USA*
[3] *Faculty of Electrical Engineering and Computing, University of Zagreb, Unska 3, 10000 Zagreb, Croatia*
[*] *hengameh@mit.edu*
[†] *ycshen@mit.edu*

Abstract: Convolutional Neural Networks (CNNs) are a class of Artificial Neural Networks (ANNs) that employ the method of convolving input images with filter-kernels for object recognition and classification purposes. In this paper we propose a photonics circuit architecture which could consume a fraction of energy per inference compared with state of the art electronics.

1. Introduction

Exploration of neuromorphic computing architectures began in the late 1950s with the invention of the perceptron, which functioned as a binary classifier with a linear decision boundary [1]. The preceptron worked well for certain tasks, but further progress was hindered by a lack of understanding on how to handle multilayer versions. Progress on neuromorphic computing for image processing accelerated rapidly in the 1990s, when LeCun et al. pioneered using back-propagation on an architecture based on convolving images with kernels, known as Convolutional Neural Networks (CNNs) [2–4]. This architecture consists of successive layers of convolution, nonlinearity, downsampling, followed by fully connected layers (see Fig. 1a). The key to the success of CNNs was that convolution and downsampling handled the translation invariance of image features efficiently, while the multiple layers allowed greater flexibility in training than the few-layer approaches.

Although the CNN architecture successfully managed to implement digit classification at human performance levels and compared favorably to other machine learning techniques, it was not until improvements in processing speeds and the creation of large human-labeled image databases from the Internet, that the full potential of CNNs became apparent [5]. Using GPU-accelerated backpropagtion, AlexNet achieved record breaking results on ImageNet for a thousand categories using a CNN architecture composed of five convolutional layers and three fully connected layers [5, 6]. Following AlexNet's lead, modern CNNs of dozens or hundreds of layers, and hundreds of millions to billions of parameters, can achieve better than human level performance in many image classification tasks [7, 8]. Recent breakthroughs with DeepLearning such as playing Atari games [9], by combining reinforcement-learning and CNNs, have convinced many that these networks are some of the best tools for a new machine learning golden age with applications ranging from pedestrian detection for self-driving cars to biomedical image analysis [10–16].

A big part of this success story was the advent of GPU-acceleration for large matrix-matrix multiplications, which are the essential and most time intensive step of back-propagation in CNN training. Despite significant gains, training large CNNs takes weeks utilizing large clusters of GPUs. More practically, GPU-accelerated CNN inference is still a computationally intensive task, making image analysis of the vast majority of the image and video data generated by the Internet very difficult. Youtube itself, in 2015 experienced uploads of 300 hours of video every minute [17]; this would require a cluster of 18000 Nvidia Titan X GPUs to process continuously with CNNs, drawing 4.5 Megawatts, with the hardware costing tens of millions of US dollars [18].

Given that this is just one company and that video traffic is predicted to grow to be 80% of the Internet by 2020 [19], this problem is going to get harder and will far outpace the current

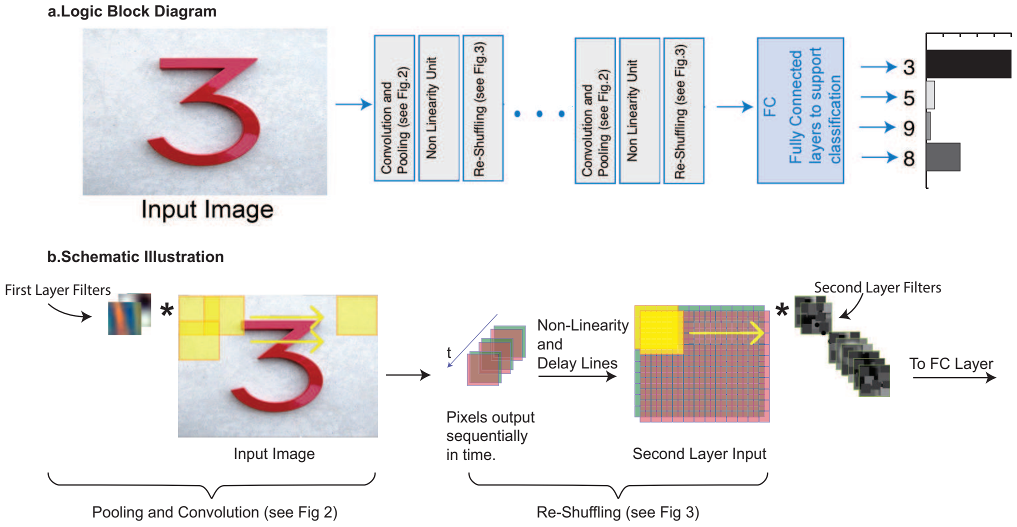


Fig. 1. Convolutional Neural Net (CNN) Architecture. a. Logic Block Diagram: The input image, number 3 shown here, is passed through successive layers of convolution and pooling, nonlinearities (see Fig. 2 for further description), and re-shuffling of the pixels (see Fig. 3 for further description). A final fully connected layer maps the last stage of convolution output to a set of classification outputs. b Schematic Illustration: First part of CNN implements convolution of the image with a set of smaller filters. These produce a sequence of kernel-patch dot products which are passed through a nonlinearity and are re-shuffled into a new d-dimensional image, where d is the number of filters in the first layer. The process is then repeated on this new image for many subsequent layers.

computing paradigms, requiring investment in specialized neuromorphic *hardware* architectures. There are many proposals and experimental demonstrations to accomplish this through analog circuits, digital ASIC designs, FPGAs, and other electronic technologies [20–26].

Our work follows a long history of optical computing such as optical implementations of unitary matrix multiplication, optical memory, all optical switching, optical interconnects, and even recent works on optical neuromorphic architectures such as photonic spike processing and reservoir computing [27–36]. We focus primarily on integrated photonics as a computation platform because it provides the highest raw bandwidth currently available of any technology that is mass manufacturable and has standardized components.

2. Architecture

As depicted in the block diagram form in Fig. 1a, and pictorially in Fig. 1b, the CNN algorithm consists of several main steps, each of which we need to execute optically. First the image is convolved with a set of kernels. The output is a new image with dimensions $[(W-K)/S+1] \times [(W-K)/S+1] \times [d]$, where W is the original image width, K is the kernel dimension, S is the convolution stride and d is the number of kernels. Next, the new image is subject to pooling, where the image produced by convolution is further downsampled by selecting the maximum value of a set of pixels (max pooling) or taking their average (average pooling). After pooling, a nonlinearity is applied to each pixel of the downsampled image. This nonlinearity can consist of the rectified linear unit (ReLU), sigmoid, tanh, or other functions. Following nonlinearity, the entire process is repeated with new sets of kernels and the same nonlinearity. After a number of these convolution layers, a fully connected neural network is applied to the output to perform the final processing steps for classification.

We begin our discussion of an optical kernel convolution, with the description of a related GPU algorithm [37]. One of the chief advantages of utilizing GPUs for machine learning is their ability to execute large matrix-matrix multiplications rapidly. For fully connected neural networks, it is obvious how this capability can lead to large speedups. To see how this works for CNNs, we first depict the conversion of an image into a set of "patches", the same dimension as the kernels in Fig. 2a. In a GPU algorithm, these patches can be converted into a "patch matrix" by vectorizing and stacking each patch. The patch matrix can then be efficiently multiplied by a "kernel matrix", formed by vectorizing and stacking each kernel. The output is a matrix composed of kernel-patch dot products which can then be "re-patched" for multiplication by the next layer's kernel matrix.

A recent work [32] utilized networks of MZIs and variable loss waveguides for optical matrix multiplication for fully connected neural networks. Here we propose using such a network of MZIs and variable loss waveguides to implement the kernel matrix multiplication, as depicted in Fig. 2b. The patch matrix takes the form of a sequence of coherent optical pulses whose amplitude encodes the intensity of each patch pixel from an image. In turn, each output of the photonic circuit will correspond to a time series of Kernel-patch dot products with the amplitude and phase (0 or π) of each pulse encoding the output of the computation. Fig. 2b depicts this process schematically, where the photonic circuit outputs at different times have been drawn in their corresponding locations in the next layer's image. Furthermore, because MZIs take up a large chip space, by doing this time multiplexing method, we are minimizing the number of MZIs we use to be equal to the number of degrees of freedoms in the kernal matrices – the theoretical limit for any hardware implementation using weight-stationary data flow scheme [26] [1]

In addition to implementing convolution with kernel matrices, the photonic circuit in Fig. 2 is also providing two other functions: pooling and nonlinearity. Pooling (or downsampling) is realized by taking the stride of the convolution to be greater than one. This step occurs when the patches are formed from the input image. This provides a necessary information filtering function required to reduce the image to only a few bits of information identifying its class. CNNs working this way have been shown to have the same performance as those using max or average pooling [38].

Finally, optical or electrical nonlinearity and repatching is applied to each output of the circuit, providing the remaining ingredient for a complete CNN layer. In the following section, we will discuss two different ways of carrying out such nonlinearity and repatching, and compare their performances.

3. Physical Implementation

3.1. Full Optical Setup

Figure 4 shows a full implementation of the envisioned architecture using optical matrix multiplication, optical nonlinearity and optical delay lines. To illustrate the fundamental features of the photonic integrated circuit more clearly we have omitted the required optical amplifiers for each convolution layer, and some parts of the optical matrix multiplication.

The first part of the circuit consists of an optical interference unit. The matrix M_i encodes the kernels for a given convolution layer. From SVD decomposition we know that $M_i = U\Sigma V$, where V and U are unitary and Σ is some real diagonal matrix. U and V are implemented through Reck-encoding of the MZI matrix, and Σ through tunable waveguide loss [32, 39–41]. An incorrect realization of the unitary matrices (among many other possible errors) will degrade

[1] Another natural way of carrying out convolutions in optical domain is using lens-spacial-light-modulator(SLM)-lens system. The advantage of such a system is that one can feed in the entire image in one clock (instead of feeding through multiple patches in our system). However, the disadvantage for such approach is SLM modulation speed is 6 orders of magnitude slower than an integrated modulator, and for each kernel one would need a separate lens-SLM-lens system. For a typical CNN, each layer contains hundreds of kernels, which will effectively make the system to be bulky.

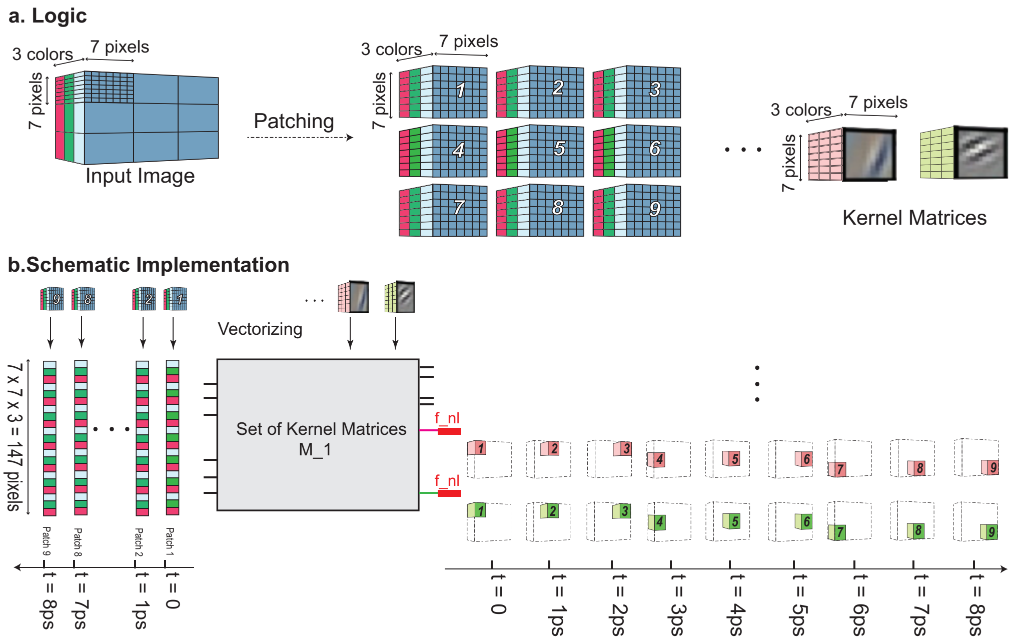


Fig. 2. General Optical Matrix Multiplication a. Logic: The pixels of the input image on the left (21 x 21 x 3 colors) are grouped into smaller patches, which have the same dimension as the kernels of the first layer (depicted on the right-hand side). b. Schematic Implementation: Each of these patches is reshaped into a single column of data that is sequentially fed, patch by patch, into the optical interference unit. Signal propagation of the optical data column through the unit implements a dot product of the first layer kernels with the patch input vector. The result is a time series of optical signals whose amplitude is proportional to the dot products of the patches with the kernels. Each output port of the optical interference unit corresponds to a separate time series of dot products associated with a given kernel. Optical nonlinearity is applied to each output port of the optical interference unit.

a network's inference capability, which we explore in the Appendix through a numerical example. In Fig. 4 we have omitted the optical implementations of Σ and V for simplicity.

The next stage consists of optical nonlinearity applied to each output waveguide of the MZI matrix. As suggested in ref. 31, optical nonlinearity can be realized by using graphene, dye, or semiconductor saturable absorbers [42–46]. Additionally ref. 31 showed that the nonlinear response of graphene saturable absorbers has a suitable functional form for training neural networks. The power budget required for operating AlexNet, utilizing a graphene saturable absorber nonlinearity operating at $\sim$ 0.05mW per waveguide input powers is detailed in the Appendix B [47,48].

We find in the appendix there that a single inference can be computed with P_{inf} power, which is given by:

$$1.26 \cdot 10^{11} \Delta t P_0 = P_{inf} \tag{1}$$

Taking $P_0 \sim 0.05$mW, $\Delta t = \frac{1}{f} = 1/3$ ns, where $f = 3$ GHz is the throughput (from the maximum delay-line bandwidth), we find that we require 2 mJ per inference which is of the same order of magnitude as that of an electronic setup demonstrated in Appendix C. Furthermore the optical implementation is 30 times faster than GPU-enabled inference with AlexNet (See Appendix C for calculations).

Finally a single convolution layer ends with repatching logic consisting of a tree of 3dB splitters feeding into variable length delay lines. We propose to re-patch through a set of optical delay

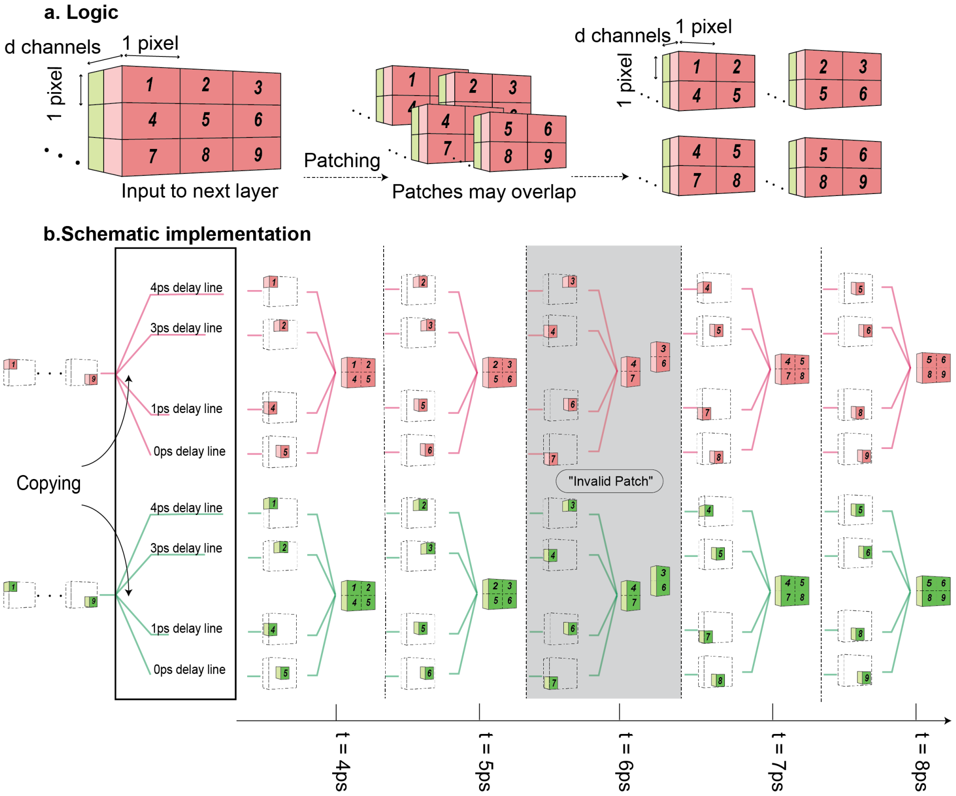


Fig. 3. Optical Delay Lines for repatching a. Logic: The output kernel dot products from the first layer (right hand side Fig2.b) are depicted as cubes on the left in Fig3. Each cube is labeled with the timestep the corresponding kernel dot product was computed. The right hand side indicates the repatching procedure necessary to convert the set of kernel dot products on the left hand-side into input patches the same size as the next layer's kernels. b. Schematic Implementation: The optical delay lines are designed such that a sequence of kernel dot products can be reshuffled in time to form a new patch the same size as the next layer's kernels. Each delay line is connected through 3-dB splitters to the original signal line, allowing the data to be copied and then delayed for synchronization. The reshuffling procedure produces valid patches only at specific sampling times. The grayed out section at t=6ps indicates an invalid sampling interval, where the patches have partially wrapped around.

lines and splitters. The requirements for the splitting and delay procedure can be understood from Fig. 3a. Here we depict the time sequence of kernel-patch dot products from Fig. 2b, as a single image, with each pixel of the image labeled with the timeslot associated with the computation. This image needs to be converted into four patches on the right of Fig. 3a, which will need to appear as a time sequence input for the next optical matrix multiplication. How this is accomplished is illustrated in Fig. 3b. Here a given output from the previous layer is split into four separate waveguides and subject to different delays. Each delay line is selected such that at a given time, the outputs from the previous layer are synchronized in time and form a new patch for input into the next layer's kernel matrix. Since in this particular example we are forming two by two patches, a delay line of one time unit is required for the top right signal to arrive at the same time as the top left signal. Further delay lines of three and four time units are required for the bottom left and right signals to arrive with these. We illustrate the formation of new patches on the right hand side of Fig. 3b, where at specific times we have formed the four desired patches from Fig. 3a. Note that the grayed out section indicates a sampling time when an invalid patch is formed, that is the wrong set of pixels have arrived simultaneously. Since the original length of the patch matrix input is nine time units long, and since there are only four patches for input into the next system, there will be five such invalid sampling times in the period of the original input signal. The exact length of the delay lines for each layer are determined (see Appendix E), and the maximum delay line length is $\sim 5000\Delta t$. In this approach, the total power consumption is solely determined by the optical power needed to trigger optical nonlinearity, and ranges from a minimum value of 2mJ/image (a detailed power calculation is described in Appendix B).

To get a general sense of the engineering difficulties, if one use this setup to carry out conventional convolutional neural networks, delay lines have been engineered 1 ns long with a 3 GHz bandwidth using 200 ring resonators on 0.2mm^2 area [49–52]. Assuming we are using this technology, and $\Delta t \sim 1$ns, this will require 5000 such delay units or 5μs of delay. Since there are 256 outputs for the final AlexNet layer [6], we have to assume we would require at least this many maximum delay line chips with a total area of $\sim$ 500cm^2. Engineering and integrating delay lines of such length is a substantial engineering challenge.

3.2. Optical-Electronic Hybrid Setup

It is possible to replace the photonics circuits that are difficult to implement by their electronic counterparts. Due to the slow conversion of analog to digital signals and vice-versa, we propose that electronic counterparts operate in the analog domain (the schematic data flow of such system is illustrated in Fig. 5). Otherwise, the speed gained by photonics circuits could be offset by slow conversion. Furthermore, such conversion would require additional electronic circuits and overall power consumption could increase. The non-linearity needed in the convolution neural networks can be implemented by simple CMOS circuits such as [53, 54]. Since the voltage-current characteristic of a diode resembles ReLU, even a simple diode can be used as a nonlinear activation function. The long optical delay lines can be replaced by e.g. memristors [55], capacitance based analog memories [56,57]. The benefits of such hybrid architecture are reduced overall size, simpler implementation and on-chip integration of the system. A detailed analysis of future work on this system and its components should focus on increasing the bandwidth. This task is critical for maximizing the inference rate of the system, shortening the delay line length, and decreasing the power consumption per inference. However every component, including the optical nonlinearity, MZIs, optical detectors, and signal generators, can potentially limit the bandwidth. Currently we are limited in bandwidth by the present state-of-the-art compact delay lines [51]. Beyond this, a significant limiting factor will be optical detectors which operate up to 120 GHz [58]. A detailed derivation of the energy consumed by the hybrid setup is found in Appendix D.

4. Conclusion

We have outlined the operation of a novel photonics integrated circuit that is capable of implementing high speed and low power inference for CNNs [2]. The system we outlined is related to a recently published work which investigated fully connected optical neural networks [32]. Here we bridge the gap between the fully connected optical neural network implementation and an optical CNN through the addition of precisely designed optical delay lines connecting together successive layers of optical matrix multiplication and nonlinearity. This new platform, should be

[2] As we were completing this work for publication, another article was posted on the Arxiv, suggesting a related optical CNN implementation [59].

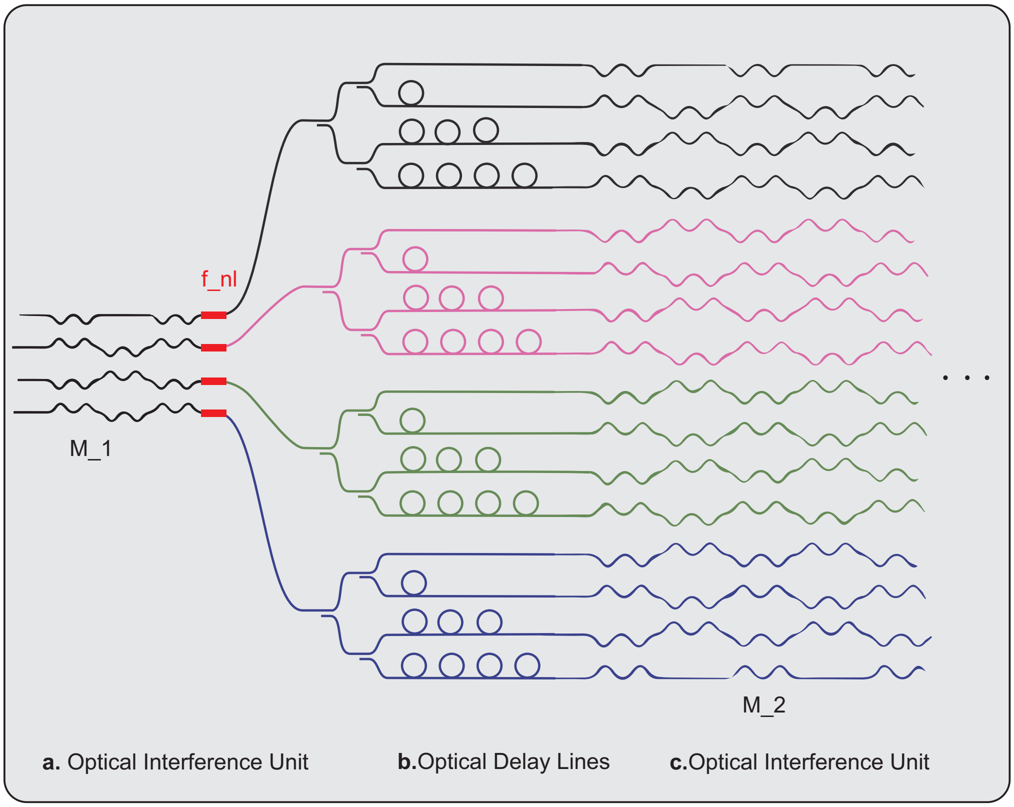


Fig. 4. Illustration of optical interference unit with delay lines a. Optical interference unit: In the first stage an optical interference unit is used to implement a kernel matrix M_1 which processes the patches from the original image. The red segments on the output of M_1 are optical nonlinearity. b. Optical Delay Lines: In the second stage, optical delay lines properly reform the sequence of kernel dot products into new patches for input into the second kernel matrix M_2. c.Optical interference unit: In the third stage the next optical interference unit is used to implement M_2 (partially depicted here). For clarity the actual number of inputs and outputs have been reduced and the attenuator stage and subsequent additional optical interference units have been omitted from M_1 and M_2. Additionally we have omitted optical amplifiers required in each layer which boost the power sufficiently to trigger the optical nonlinearity.

able to perform on the order of a million inferences per second, at 2mJ power levels per inference, with the nearest state of the art ASIC competitor operating 30 times slower and requiring the same total power [26]. This system could play a significant role in processing the thousands of terabytes of image and video data generated daily by the Internet. Although the implementation of the photonics circuit proposed in this manuscript would be a substantial engineering challenge, the benefits of successfully realizing it are difficult to understate. In the meantime one could replace the photonic sub-circuits that are difficult to implement by their electronic counterparts.

5. Acknowledgements

We are grateful to Yann LeCun for his comments on an earlier version of this work, and encouraging us to look into this direction. We acknowledge useful discussions with Dirk Englund and John Joannopoulos. Research was sponsored by the Army Research Office and was

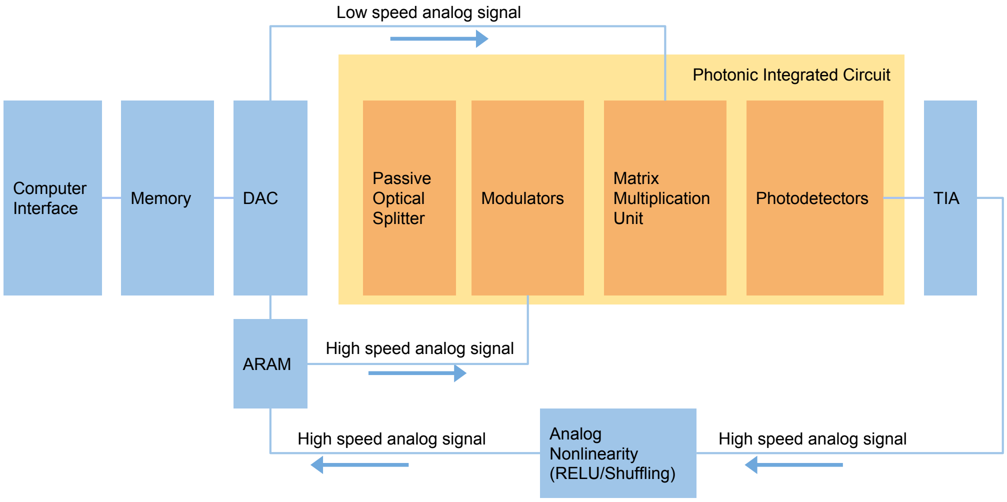


Fig. 5. The photonic integrated circuit (PIC) consists of four stages. An input laser is first split equally. Then the light will go through an array of optical amplitude modulators. Afterwards, the modulated optical signal will pass through the optical matrix multiplication unit (OMMU). Finally, the outputs are fed into an array of photodetectors and converted to photocurrents. On the electronic side, the system first receive data from an interface to external computational device and store them in a memory. The data are then converted to analog. The matrix weights are applied to the OMMU through a low speed link. Afterwards, the images information are transfered to a high speed analog memory (ARAM) before sent to optical modulators. The outputs of PIC are typically weak, so a transimpedance amplifier (TIA) array is used. A high speed ASIC circuit is used to perform shuffling and nonlinear operations, such as RELU or sigmoid. Afterwards, the rearranged images are sent back to the ARAM and ready for the next cycle of matrix multiplication. The image data will go through a large number of cycles in analog domain before being converted back to digital. Therefore, no high speed analog to digital conversion is necessary.

accomplished under Cooperative Agreement Number W911NF-18-2-0048.

Appendix A: Analysis of MZI phase encoding and error

Although the analog nature of our optical CNN can allow for high precision computation, it also suffers significantly more than equivalent digital architectures from error propagation. There are many sources of error within our system that include variable waveguide loss, variable optical nonlinearity, variable amplification, and shot noise. We do not discuss these here, but for the purposes of providing a basic model of error propagation through the circuit, we calculate the classification error from incorrect phase settings of a Reck-encoded MZI matrix [40, 41, 60].

To estimate how errors in MZI phase settings effect classification accuracy, a simulation was done on a toy model of a digit recognition CNN. Our toy CNN was comprised of two convolution layers and two fully connected layers and was trained on the MNIST digit recognition data set. After training the reference CNN (which had a classification accuracy of 97% on the test data set), the kernel matrix for each layer was exported to a phase extractor program that calculated the phase settings necessary to implement the unitary component of these matrices with MZIs (i.e. U and V from SVD decompostion of $M = U\Sigma V$).

We describe how we can calculate a phase encoding for any real valued unitary matrix. By applying a rotation matrix $T_{i,j}(\theta)$ to real unitary matrix U, with appropriate θ we can null elements in the ith row and jth column, where $T_{ij}(\theta)$, with $i = j - 1$, is an identity matrix with elements $T_{ii}, T_{ij}, T_{ji}, T_{jj}$ replaced by a two by two rotation matrix:

$$U' = \begin{bmatrix} 1 & & 0 \\ & \boxed{\begin{matrix} \cos(\theta_{j-1,j}) & \sin(\theta_{j-1,j}) \\ -\sin(\theta_{j-1,j}) & \cos(\theta_{j-1,j}) \end{matrix}} & \\ 0 & & 1 \end{bmatrix} \begin{bmatrix} u_{1,1} & \cdots & u_{1,j-1} & 0 & 0 \\ & \cdot & & \boxed{\begin{matrix} u_{i,j} \\ u_{i+1,j} \end{matrix}} & \begin{matrix} \cdot \\ \cdot \end{matrix} \\ & & \cdot & & \\ & & & d_k & 0 \\ & & & & d_N \end{bmatrix} \tag{2}$$

To determine θ in $T_{i,j}(\theta)$ such that the ijth element of the matrix is nulled, θ must satisfy the following equation:

$$\begin{aligned} u'_{i,j} = \cos(\theta_{j-1,j})u_{i,j} + \sin(\theta_{j-1,j})u_{i+1,j} &= 0 \\ \tan(\theta_{j-1,j}) &= \frac{-u_{i,j}}{u_{i+1,j}} \end{aligned} \tag{3}$$

If we apply these rotations starting with the first element of the far-right column and working downwards we find that:

$$T_{N,N-1}T_{N,N-2}\cdots T_{N,2}T_{N,1}U(N) = \left[\begin{array}{c|c} U(N-1) & 0 \\ & \vdots \\ \hline 0\cdots & a \end{array}\right] \tag{4}$$

Now we note that the block matrix $U(N-1)$ can undergo the same process, and thus after $(N-1) + (N-2)\cdots = \frac{N(N-1)}{2}$ rotations the right hand side will turn into a diagonal real matrix D (this is only true if U is real, which it is for conventional CNNs). In total U can be written in the following way [40] [41]:

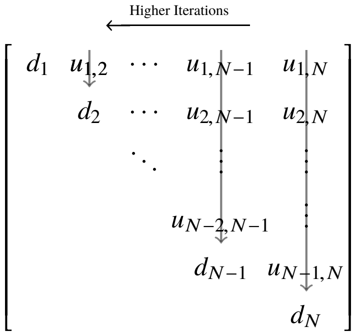
$$U = (T_{2,1}T_{3,2}T_{3,1}T_{4,3}\cdots T_{N,N-1}\cdots T_{N,1})^{-1}D \tag{5}$$

We can extract a matrix of phase encodings Θ with the following pseudocode:

```
function Phase_Extractor

for i from N to 2 :
        for j from 1 to i :
                $\Theta_{j,j+1} = \tan^{-1}(\frac{-U_{j,i}}{U_{j+1,i}})$
                Update $U = T_{j,j+1}(\Theta_{j,j+1})U$
        end for
end for
return $\Theta$
```

The algorithm nulls the elements of a given unitary matrix starting from $U_{1,N}$ and moving downward until reaching a diagonal element, upon which it moves to the next column to the left. The below schematic shows the order in which the algorithm nulls the elements and extracts the corresponding phases:



Each element of the resulting phase matrix Θ is randomly perturbed with a distribution given by $p(\Delta\Theta_{i,j}) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left[-\frac{\Delta\Theta_{i,j}^2}{2\sigma^2}\right]$. This perturbed matrix is used to create a new U' (in $M = U'\Sigma V'$), which is in turn used to generate the kernel matrices of a perturbed CNN. The results of this perturbed network's digit classification on MNIST are then compared to those of the trained unperturbed CNN such that the network is assumed to be "error free" if it gives the same results as the reference network, not if it has 100% correct classification. These results are plotted in Fig. 6.

We find that for an error distribution with $\sigma > 0.01$, the performance of the perturbed network is significantly degraded relative to the unperturbed version. This corresponds to about 8-bit accuracy in the phase settings, which has been achieved in the previous work [32] on fully connected optical neural networks. These results are promising, but larger CNNs need to be examined with this method to assess their tolerance to the phase setting errors, and the other errors mentioned briefly above.

Appendix B: Power consumption for optical implementation of AlexNet

To ensure proper operation of the saturable absorption nonlinearity, the input power needs to be of the order of $P_0 \sim 0.05$ mW per waveguide for graphene to trigger the nonlinear effect [47, 48]. This requires that the output in each layer be amplified to that signal level, since the average power level is divided every layer by waveguide splitters in the repatching logic. To accommodate this, optical amplifiers are needed at every layer to boost the patch inputs appropriately. Given that standard integrated InP amplifiers operating in the IR have a wall plug efficiency of 10% or more [61], this means that the power dissipated by the network will scale as $0.5\frac{mW}{waveguide}$.

Since we are feeding $55 \times 55 = 3025$ patch units at a rate of $\sim$ 3GHz each optical nonlinearity unit requires $E = 3025 \cdot 10\frac{P_0}{f} \approx 5 \cdot 10^{-10}$J/waveguide for each signal. Consequently if we sum the number of waveguides in each layer and multiply by this number, we should get an idea of the power requirements for this type of integrated photonic circuit:

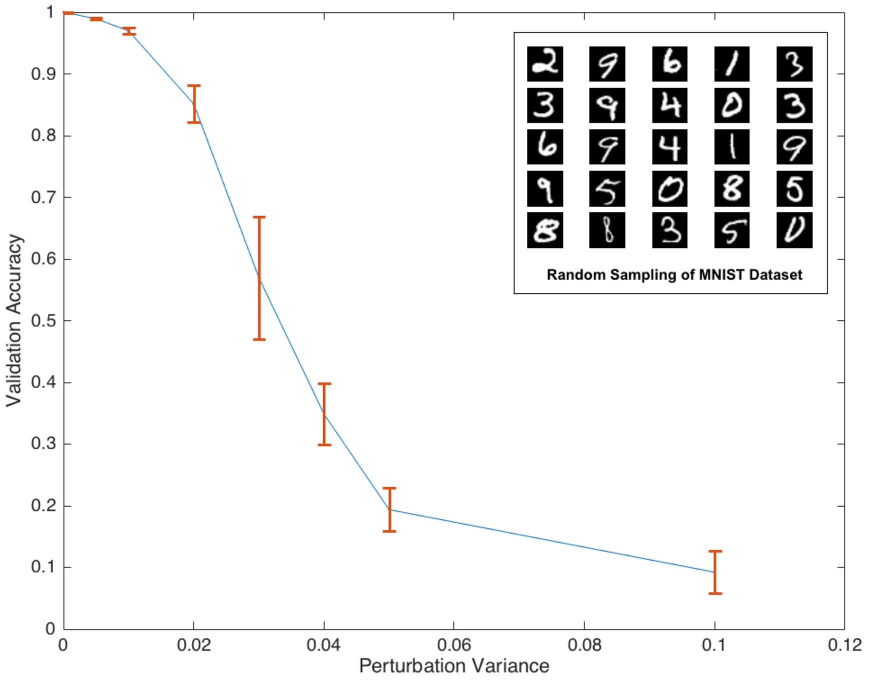


Fig. 6. Performance of digit recognition network on MNIST dataset vs. training parameter perturbation. After the matrix of phases, Θ, has been returned by the phase extractor algorithm, the entries are perturbed and the resulting phase matrix is used in composing a new unitary matrix of weights that builds up a second CNN with desired perturbations. The perturbed CNN is tested and its inference performance on the MNIST dataset is then compared to those of the unperturbed CNN to analyze the error.

	# of ReLU units	# of Input Patches	Layer Energy Consumption(J)
1st Conv	96	55×55	1452000×10^{-10}
2nd Conv	256	55×55	3872000×10^{-10}
3rd Cnv	384	55×55	5808000×10^{-10}
4th Conv	384	55×55	5808000×10^{-10}
5th Conv	256	55×55	3872000×10^{-10}
1st FC	4096	1[3]	20480×10^{-10}
2nd FC	4096	1	20480×10^{-10}
3rd FC	1000	1	5000×10^{-10}
Total Energy Consumption			$20,857,960 \times 10^{-10}$

[3] The number of input patches decreases to one for the fully connected layers because we employ electro-optic

, this yields a total energy consumption per inference of 2 mJ for AlexNet [6]. We can further rewrite this as $1.26 \cdot 10^{11} \Delta t P_0$.

Appendix C: Power consumption for electronic implementation of AlexNet

Electronic computers consume a fixed amount of energy per floating point operation. Since data movement (i.e. data transfer between hard drive and RAM, etc.) represents additional significant "overhead" for memory intensive algorithms like CNNs, and minimizing this is the chief objective of new neuromorphic digital architectures, calculating the total number of floating point operations required for AlexNet gives us a good estimation of the best case performance for a digital implementation of that algorithm:

	Kernel Size	Number of Input Patches	Number of Kernels	Layer FLOPs
1st Conv	$11 \times 11 \times 3$	55×55	96	105415200×2
2nd Conv	$5 \times 5 \times 96$	27×27	256	447897600×2
3rd Conv	$3 \times 3 \times 256$	13×13	384	149520384×2
4th Conv	$3 \times 3 \times 384$	13×13	384	224280576×2
5th Conv	$3 \times 3 \times 384$	13×13	256	149520384×2
1st FC	$13 \times 13 \times 256$	1×1	4096	177209344×2
2nd FC	$1 \times 1 \times 4096$	1×1	4096	16777216×2
3rd FC	$1 \times 1 \times 4096$	1×1	1000	4096000×2
Total FLOPs				$2,549,433,408$

, electronic computers have an average performance rate of $112 \frac{\text{TFLOPs}}{\text{s}}$ at 250 W [62], so the lower bound on the total energy consumed by a digital computer (ASIC, GPU, CPU) running AlexNet is $2,549,433,408$ FLOPs $\times (112 \frac{\text{TFLOPs}}{\text{s}})^{-1} \times 250$ W ≈ 5.7mJ [6]. In reality, the power consumption per image is a lot higher, because not all layers (e.g. fully connected layer) can be carried out at TPU's peak performance.

To compare the temporal performance of an electronic setup with that of a full optical implementation the time interval over which each circuit processes an image is calculated. For the electronic circuit, dividing the total number of FLOPs by the performance rate provides us with the amount of time it takes to process one image: $2,549,433,408$ FLOPs $\times (112 \frac{\text{TFLOPs}}{\text{s}})^{-1} \approx 2.2 \times 10^{-5}$s. Meanwhile, the optical counter part of the image processing time is derived from dividing the total number of patches within an image by the throughput of the system, 3025 unit patches $\times (3\text{GHz})^{-1} \approx 10^{6}$s, revealing that the optical setup is 30 times faster than its electronic counterpart.

Appendix D: Power consumption for optical-electronic hybrid implementation of AlexNet

The bulk of the energy consumption in the optical-electric hybrid implementation is from analog electronics. Number of analog operations for each layer can be obtained by kernel size × number of input pads × 2.

converters to extract one valid patch from the convolutional layer output. In principle this technique could be employed to extract and re-emit only valid patches from previous layers, but we do not discuss this here, because we are primarily interested in an all-optical implementation.

	Kernel Size	Number of Input Patches	Analog Operations
1st Conv	$11 \times 11 \times 3$	55×55	$1.10 \times 10^6 \times 2$
2nd Conv	$5 \times 5 \times 96$	27×27	$1.75 \times 10^6 \times 2$
3rd Conv	$3 \times 3 \times 256$	13×13	$3.89 \times 10^5 \times 2$
4th Conv	$3 \times 3 \times 384$	13×13	$5.84 \times 10^5 \times 2$
5th Conv	$3 \times 3 \times 384$	13×13	$5.84 \times 10^5 \times 2$
1st FC	$13 \times 13 \times 256$	1×1	$4.32 \times 10^4 \times 2$
2nd FC	$1 \times 1 \times 4096$	1×1	4096×2
3rd FC	$1 \times 1 \times 4096$	1×1	4096×2
Total			8.96×10^6

Major source of power consumption in analog circuits are .0transimpedance amplifiers, optical modulator drivers and analog memories. Typical off-the-shelf components have 20 pJ/sample, 100 pJ/sample, respectively. Analog memory power consumption is estimated using typical CMOS capacitor:

$$\begin{aligned} \text{Energy per bit} &= CV^2 \times \text{number of capacitors} \\ &= 100fF \times (1V)^2 \times 10^3 \\ &= 100pJ \end{aligned} \tag{6}$$

Therefore, the total analog energy for each image is around $(20 + 100 + 100) \times 8.96 \times 10^6 =$ 2mJ/image .

Laser power can be calculated by:

$$\begin{aligned} \text{Energy per bit} &= \text{laser power} \times \text{total number of patches}/\text{modulation speed} \\ &= 2W \times 4264/10^9 \\ &= 8.5\mu J \end{aligned} \tag{7}$$

This is negligible compared with analog electronic energy consumption.

Recent research [63] shows that, with monolithic integration of electronic circuits into the photonic network, it is possible to dramatically reduce the energy consumption. Modulators cost as little as 5fJ/sample. Additionally, the analog memory could potentially be designed in such a way that only a small fraction of the data are moved for each cycle. As a result, the total energy consumption in the optical-electronic hybrid implementation can be reduced substantially. The energy consumption for each image could be as low as 45nJ/image.

Appendix E: Delay Line Length Calculation

The calculation to derive each delay line length begins with the following definitions:

Column Delay, $\Delta t_j :=$ The delay time required for two adjacent pixels in the same row to arrive at the same time.

Row Delay, $\Delta T_j :=$ The delay time required for two adjacent pixels in the same column to arrive at the same time.

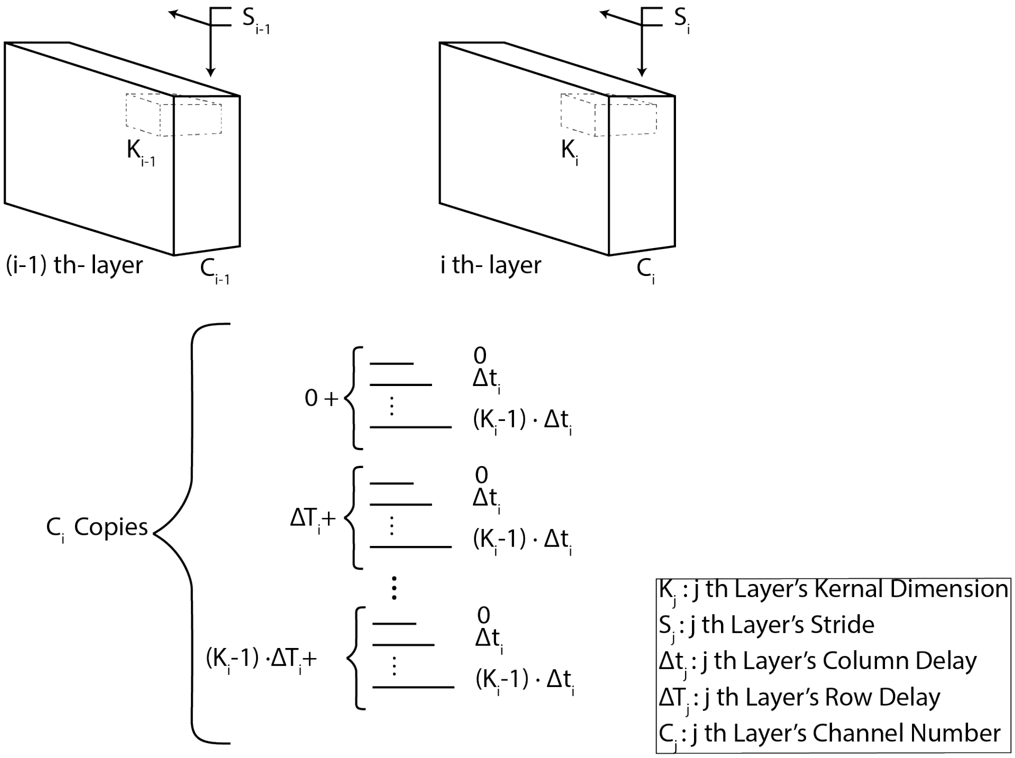

S_{i-1}
K_{i-1}
(i-1) th- layer
C_{i-1}
S_i
K_i
i th- layer
C_i
C_i Copies
0 +
0
Δt_i
(K_i-1) · Δt_i
ΔT_i+
0
Δt_i
(K_i-1) · Δt_i
(K_i-1) ·ΔT_i+
0
Δt_i
(K_i-1) · Δt_i
K_j : j th Layer's Kernal Dimension
S_j : j th Layer's Stride
Δt_j : j th Layer's Column Delay
ΔT_j : j th Layer's Row Delay
C_j : j th Layer's Channel Number

Fig. 7. Delay lines connecting the $(i-1)th$ and $i\ th$ layers of a CNN.

An example of Column delay is the delay line, with $\Delta t_j = 1ps$, required for the top right signal to arrive at the same time as the top left one on the right of Fig. 3a. The delay line, with $\Delta T_j = 3ps$, required for the top right signal to arrive at the same time as the bottom right one on the right of Fig. 3a. is an example of a row delay.

Looking at Fig. 7 one can write the following recursive formulas for row and column delay lengths:

$$\begin{aligned} \Delta t_j &= \Delta t_{j-1} \times S_{j-1} \\ \Delta T_j &= \Delta T_{j-1} \times S_{j-1} \end{aligned} \tag{8}$$

And the maximum delay line length for each layer can be calculated as :

$$\text{Max Delay Length} = (K_j - 1) \times (\Delta T_j + \Delta t_j) \tag{9}$$

The initial values for Column Delays, Δt_0, and Row Delays, ΔT_0, are derived using the properties of the input image and the first convolution layer. Assuming that the input image has dimensions of $W \times W$, zero-padding of P, and is being fed into the system with a frequency f we get :

$$\begin{aligned} \Delta t_1 &= \frac{1}{f} \\ \Delta T_1 &= \frac{1}{f} \times (\frac{W - K_0 + 2P}{S_0} + 1) \end{aligned} \tag{10}$$

Using the above formulas the delay line length for AlexNet is calculated in Table 1:

	Dimension	Kernel Size	Stride	Δt	ΔT	Delay Line Length
1st-ConvLayer	55×55	5×5	2	$\frac{1}{f}$	$55 \times \frac{1}{f}$	$4 \times (55 + 1)\frac{1}{f} = \frac{224}{f}$
2nd-ConvLayer	27×27	3×3	2	$\frac{2}{f}$	$55 \times \frac{2}{f}$	$2 \times (55 + 1)\frac{2}{f} = \frac{224}{f}$
3rd-ConvLayer	13×13	3×3	2	$\frac{4}{f}$	$55 \times \frac{4}{f}$	$2 \times (55 + 1)\frac{4}{f} = \frac{448}{f}$
4th-ConvLayer	13×13	3×3	1	$\frac{8}{f}$	$55 \times \frac{8}{f}$	$2 \times (55 + 1)\frac{8}{f} = \frac{896}{f}$
5th-ConvLayer	13×13	13×13	NA	$\frac{8}{f}$	$55 \times \frac{8}{f}$	$12\times(55+1)\frac{8}{f} = \frac{5376}{f}$

Table 1. AlexNet Delay Lines being fed to the system at frequency f.

References

1. F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain." Psychol. review **65**, 386 (1958).
2. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proc. IEEE **86**, 2278–2324 (1998).
3. G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," science **313**, 504–507 (2006).
4. I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning* (MIT Press, 2016).
5. O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, "ImageNet Large Scale Visual Recognition Challenge," Int. J. Comput. Vis. (IJCV) **115**, 211–252 (2015).
6. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems,* (2012), pp. 1097–1105.
7. K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision,* (2015), pp. 1026–1034.
8. 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.
9. V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," Nature **518**, 529–533 (2015).
10. Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," Nature **521**, 436–444 (2015).
11. J. Schmidhuber, "Deep learning in neural networks: An overview," Neural networks **61**, 85–117 (2015).
12. D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," Nature **529**, 484–489 (2016).
13. R. Benenson, M. Omran, J. Hosang, and B. Schiele, "Ten years of pedestrian detection, what have we learned?" arXiv preprint arXiv:1411.4304 (2014).
14. A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," Nature **542**, 115–118 (2017).
15. K.-H. Yu, C. Zhang, G. J. Berry, R. B. Altman, C. Ré, D. L. Rubin, and M. Snyder, "Predicting non-small cell lung cancer prognosis by fully automated microscopic pathology image features," Nat. Commun. **7** (2016).
16. H. Li, Z. Lin, X. Shen, J. Brandt, and G. Hua, "A convolutional neural network cascade for face detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* (2015), pp. 5325–5334.
17. B. Brouwer, "Youtube now sees 300 hours of video uploaded every minute," Online unter: http://www. tubefilter. com/2014/12/01/YouTube-300-hours-video-per-minute/(22.10. 2015) (2014).
18. D. Kang, J. Emmons, F. Abuzaid, P. Bailis, and M. Zaharia, "Optimizing deep cnn-based queries over video streams at scale," arXiv preprint arXiv:1703.02529 (2017).
19. I. Cisco, "Cisco visual networking index: Forecast and methodology, 2015–2020," CISCO White paper (2016).
20. C. Mead, "Neuromorphic electronic systems," Proc. IEEE **78**, 1629–1636 (1990).
21. C.-S. Poon and K. Zhou, "Neuromorphic silicon neurons and large-scale neural networks: challenges and opportunities," Front. neuroscience **5**, 108 (2011).
22. M. Horowitz, "1.1 computing's energy problem (and what we can do about it)," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International,* (IEEE, 2014), pp. 10–14.
23. A. Shafiee, A. Nag, N. Muralimanohar, R. Balasubramonian, J. P. Strachan, M. Hu, R. S. Williams, and V. Srikumar, "Isaac: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars," in *Proceedings of the 43rd International Symposium on Computer Architecture,* (IEEE Press, 2016), pp. 14–26.
24. J. Misra and I. Saha, "Artificial neural networks in hardware: A survey of two decades of progress," Neurocomputing **74**, 239–255 (2010).
25. P. Merolla, J. Arthur, F. Akopyan, N. Imam, R. Manohar, and D. S. Modha, "A digital neurosynaptic core using embedded crossbar memory with 45pj per spike in 45nm," in *Custom Integrated Circuits Conference (CICC), 2011 IEEE,* (IEEE, 2011), pp. 1–4.
26. Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," IEEE J. Solid-State Circuits (2016).
27. K. Nozaki, T. Tanabe, A. Shinya, S. Matsuo, T. Sato, H. Taniyama, and M. Notomi, "Sub-femtojoule all-optical switching using a photonic-crystal nanocavity," Nat. Photonics **4**, 477–483 (2010).
28. C. Ríos, M. Stegmaier, P. Hosseini, D. Wang, T. Scherer, C. D. Wright, H. Bhaskaran, and W. H. Pernice, "Integrated all-photonic non-volatile multi-level memory," Nat. Photonics **9**, 725–732 (2015).
29. J. Sun, E. Timurdogan, A. Yaacobi, E. S. Hosseini, and M. R. Watts, "Large-scale nanophotonic phased array," Nature **493**, 195–199 (2013).
30. C. Sun, M. T. Wade, Y. Lee, J. S. Orcutt, L. Alloatti, M. S. Georgas, A. S. Waterman, J. M. Shainline, R. R. Avizienis, S. Lin *et al.*, "Single-chip microprocessor that communicates directly using light," Nature **528**, 534–538 (2015).
31. T. Tanabe, M. Notomi, S. Mitsugi, A. Shinya, and E. Kuramochi, "Fast bistable all-optical switch and memory on a silicon photonic crystal on-chip," Opt. letters **30**, 2575–2577 (2005).

32. Y. Shen, N. C. Harris, S. Skirlo, M. Prabhu, T. Baehr-Jones, M. Hochberg, X. Sun, S. Zhao, H. Larochelle, D. Englund *et al.*, "Deep learning with coherent nanophotonic circuits," Nat. Photonics **11**, 441–446 (2017).
33. A. N. Tait, M. A. Nahmias, Y. Tian, B. J. Shastri, and P. R. Prucnal, "Photonic neuromorphic signal processing and computing," in *Nanophotonic Information Physics,* (Springer, 2014), pp. 183–222.
34. A. N. Tait, M. A. Nahmias, B. J. Shastri, and P. R. Prucnal, "Broadcast and weight: an integrated network for scalable photonic spike processing," J. Light. Technol. **32**, 3427–3439 (2014).
35. P. R. Prucnal, B. J. Shastri, T. F. de Lima, M. A. Nahmias, and A. N. Tait, "Recent progress in semiconductor excitable lasers for photonic spike processing," Adv. Opt. Photonics **8**, 228–299 (2016).
36. K. Vandoorne, P. Mechet, T. Van Vaerenbergh, M. Fiers, G. Morthier, D. Verstraeten, B. Schrauwen, J. Dambre, and P. Bienstman, "Experimental demonstration of reservoir computing on a silicon photonics chip," Nat. communications **5** (2014).
37. S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, and E. Shelhamer, "cudnn: Efficient primitives for deep learning," arXiv preprint arXiv:1410.0759 (2014).
38. J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," arXiv preprint arXiv:1412.6806 (2014).
39. N. C. Harris, G. R. Steinbrecher, M. Prabhu, Y. Lahini, J. Mower, D. Bunandar, C. Chen, F. N. Wong, T. Baehr-Jones, M. Hochberg *et al.*, "Quantum transport simulations in a programmable nanophotonic processor," Nat. Photonics **11**, 447–452 (2017).
40. M. Reck, A. Zeilinger, H. J. Bernstein, and P. Bertani, "Experimental realization of any discrete unitary operator," Phys. Rev. Lett. **73**, 58–61 (1994).
41. W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer, and I. A. Walmsley, "An optimal design for universal multiport interferometers," (2016).
42. Z. Cheng, H. K. Tsang, X. Wang, K. Xu, and J.-B. Xu, "In-plane optical absorption and free carrier absorption in graphene-on-silicon waveguides," IEEE J. Sel. Top. Quantum Electron. **20**, 43–48 (2014).
43. M. Soljačić, M. Ibanescu, S. G. Johnson, Y. Fink, and J. Joannopoulos, "Optimal bistable switching in nonlinear photonic crystals," Phys. Rev. E **66**, 055601 (2002).
44. R. W. Schirmer and A. L. Gaeta, "Nonlinear mirror based on two-photon absorption," JOSA B **14**, 2865–2868 (1997).
45. Q. Bao, H. Zhang, Z. Ni, Y. Wang, L. Polavarapu, Z. Shen, Q.-H. Xu, D. Tang, and K. P. Loh, "Monolayer graphene as a saturable absorber in a mode-locked laser," Nano Res. **4**, 297–307 (2011).
46. A. Selden, "Pulse transmission through a saturable absorber," Br. J. Appl. Phys. **18**, 743 (1967).
47. Q. Bao, H. Zhang, Z. Ni, Y. Wang, L. Polavarapu, Z. Shen, Q.-H. Xu, D. Tang, and K. P. Loh, "Monolayer graphene as a saturable absorber in a mode-locked laser," Nano Res. **4**, 297–307 (2010).
48. H. Li, Y. Anugrah, S. J. Koester, and M. Li, "Optical absorption in graphene integrated on silicon waveguides," Appl. Phys. Lett. **101**, 111110 (2012).
49. G. Lenz, B. Eggleton, C. K. Madsen, and R. Slusher, "Optical delay lines based on optical filters," IEEE J. Quantum Electron. **37**, 525–532 (2001).
50. T. Tanabe, M. Notomi, E. Kuramochi, A. Shinya, and H. Taniyama, "Trapping and delaying photons for one nanosecond in an ultrasmall high-q photonic-crystal nanocavity," Nat. Photonics **1**, 49–52 (2007).
51. F. Xia, L. Sekaric, and Y. Vlasov, "Ultracompact optical buffers on a silicon chip," Nat. photonics **1**, 65–71 (2007).
52. M. S. Rasras, C. K. Madsen, M. A. Cappuzzo, E. Chen, L. T. Gomez, E. J. Laskowski, A. Griffin, A. Wong-Foy, A. Gasparyan, A. Kasper *et al.*, "Integrated resonance-enhanced variable optical delay lines," IEEE photonics technology letters **17**, 834–836 (2005).
53. M. Carrasco-Robles and L. Serrano, "A novel minimum-size activation function and its derivative," Trans. Cir. Sys. **56**, 280–284 (2009).
54. D. Vrtaric, V. Ceperic, and A. Baric, "Area-efficient differential gaussian circuit for dedicated hardware implementations of gaussian function based machine learning algorithms," Neurocomputing **118**, 329–333 (2013).
55. M. Payvand, J. Rofeh, A. Sodhi, and L. Theogarajan, "A cmos-memristive self-learning neural network for pattern classification applications," in *Proceedings of the 2014 IEEE/ACM International Symposium on Nanoscale Architectures,* (ACM, New York, NY, USA, 2014), NANOARCH '14, pp. 92–97.
56. M. Hock, A. Hartel, J. Schemmel, and K. Meier, "An analog dynamic memory array for neuromorphic hardware," 2013 Eur. Conf. on Circuit Theory Des. (ECCTD) pp. 1–4 (2013).
57. K. Soelberg, R. L. Sigvartsen, T. S. Lande, and Y. Berg, "An analog continuous-time neural network," Analog. Integr. Circuits Signal Process. **5**, 235–246 (1994).
58. L. Vivien, A. Polzer, D. Marris-Morini, J. Osmond, J. M. Hartmann, P. Crozat, E. Cassan, C. Kopp, H. Zimmermann, and J. M. Fédéli, "Zero-bias 40gbit/s germanium waveguide photodetector on silicon," Opt. express **20**, 1096–1101 (2012).
59. A. Mehrabian, Y. Al-Kabani, V. J. Sorger, and T. El-Ghazawi, "Pcnna: A photonic convolutional neural network accelerator," arXiv preprint arXiv:1807.08792 (2018).
60. D. A. Miller, "Perfect optics with imperfect components," Optica **2**, 747–750 (2015).
61. P. W. Juodawlkis, J. J. Plant, L. J. Missaggia, K. E. Jensen, and F. J. O'Donnell, "Advances in 1.5-μm ingaasp/inp slab-coupled optical waveguide amplifiers (scowas)," in *Lasers and Electro-Optics Society, 2007. LEOS 2007. The 20th Annual Meeting of the IEEE,* (IEEE, 2007), pp. 309–310.
62. N. Corporation, "NVIDIA TESLA V100," Online unter: https://www.nvidia.com/en-us/data-center/tesla-

v100/(26.01. 2018) (2018).

63. M. T. Wade, J. M. Shainline, J. S. Orcutt, C. Sun, R. Kumar, B. Moss, M. Georgas, R. J. Ram, V. Stojanovic, and M. A. Popovic, "Energy-efficient active photonics in a zero-change, state-of-the-art CMOS process," (2014), pp. 1–3.