Machine Learning With Neuromorphic Photonics

Thomas Ferreira de Lima, *Student Member, IEEE*, Hsuan-Tung Peng, *Student Member, IEEE*, Alexander N. Tait, *Member, IEEE*, Mitchell A. Nahmias, Heidi B. Miller, Bhavin J. Shastri, *Member, IEEE*, and Paul R. Prucnal, *Life Fellow, IEEE*

(Invited Tutorial)

***Abstract*—Neuromorphic photonics has experienced a recent surge of interest over the last few years, promising orders of magnitude improvements in both speed and energy efficiency over digital electronics. This paper provides a tutorial overview of neuromorphic photonic systems and their application to optimization and machine learning problems. We discuss the physical advantages of photonic processing systems, and we describe underlying device models that allow practical systems to be constructed. We also describe several real-world applications for control and deep learning inference. Finally, we discuss scalability in the context of designing a full-scale neuromorphic photonic processing system, considering aspects such as signal integrity, noise, and hardware fabrication platforms. The paper is intended for a wide audience and teaches how theory, research, and device concepts from neuromorphic photonics could be applied in practical machine learning systems.**

***Index Terms*—Deep learning, machine learning, more-than-Moore computing, neuromorphic photonics, nonlinear programming, optimization, photonic hardware accelerator, photonic integrated circuits, photonic neural networks, silicon photonics, wavelength-division multiplexing (WDM).**

I. Introduction

A. Scope of This Paper

THIS paper is intended for both *machine learning* (ML) researchers interested in how photonics can accelerate machine learning tasks, and *neuromorphic photonics* (NP) researchers exploring how to best relate device metrics to system-level processing benchmarks. In other words, it attempts to create a bridge to computer engineering so that neuromorphic photonics can be understood as a viable option for neuromorphic computing by the high-performance computing and signal processing communities. Here, we argue that the field of neuromorphic photonics is ripe for expansion, given that critical concepts that enable neuromorphic computing with photonics have already been demonstrated. We hope this paper offers an introductory but detailed overview as well as a glimpse at the future of this emerging field.

B. Motivation: Machine Learning Outlook and Role of Optics

Artificial Intelligence (AI) has always captured our imagination. AI has the potential to drastically change almost every aspect of our lives through new medical treatments, new assistive robots, intelligent modes of transportation, and much more. Inspired by the human brain and spurred by the advances in deep learning, the past six years has seen a renaissance in AI. IBM [1], [2], HP [3], Intel [4], and Google [5], [6], have all shifted their core technological strategies from "mobile first" to "AI first". Deep learning with artificial neural networks (ANNs) [7] have expanded from image recognition [8]–[11] to translating languages [12], generating realistic speech indistinguishable from that of a human [13], and beating humans at highly complex strategy games like Go [14]. The general consensus amongst the scientific and private sector community is that three factors will drive the future advance of AI: better algorithms, more training data, and the amount of compute power available for training. While there has been no shortage of innovative architecture variants for these neural networks nor data to train them, the most pressing bottleneck for AI is now processing power (Fig. 1). Over the last six years, the amount of compute power required to train state-of-the-art AI has been doubling every 3.5 months [15]. For instance, Googles AlphaGo AI requires 1920 CPUs and 280 GPUs, which translates into massive power consumption, reaching around $3000 USD in electric bill per game. Training neural networks also takes a considerable amount of computational time. For example, image classification tasks with residual neural networks (ResNet-200) requires 8 GPUs and takes more than three weeks of training to achieve classification error rates at around 20.7% [11]. Traditional CPUs, GPUs and even neuromorphic electronics (IBM TrueNorth [2] and Google TPU [5]) have improved both energy efficiency and speed enhancement for learning (inference)

Manuscript received November 7, 2018; revised December 16, 2018 and January 30, 2019; accepted February 25, 2019. Date of publication March 7, 2019; date of current version March 27, 2019. This work was supported in part by the National Science Foundation (NSF) Enhancing Access to the Radio Spectrum (EARS) program (Award 1642991). The work of B.J. Shastri and H.B. Miller was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). (*Corresponding author: Thomas Ferreira de Lima.*)

T. F. de Lima, H.-T. Peng, M. A. Nahmias, and P. R. Prucnal are with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA (e-mail: tlima@princeton.edu; hpeng@princeton.edu; mnahmias@princeton.edu; prucnal@princeton.edu).

A. N. Tait is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA, and also with the National Institute of Standards and Technology, Boulder, CO 80305 USA (e-mail: atait@ieee.org).

H. B. Miller is with the Department of Physics, Engineering Physics & Astronomy, Queen's University, Kingston, ON KL7 3N6, Canada (e-mail: miller.heidi@queensu.ca).

B. J. Shastri is with the Department of Electrical Engineering, Princeton University, Princeton, NJ 08544 USA, and also with the Department of Physics, Engineering Physics & Astronomy, Queen's University, Kingston, ON KL7 3N6, Canada (e-mail: shastri@ieee.org).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/JLT.2019.2903474

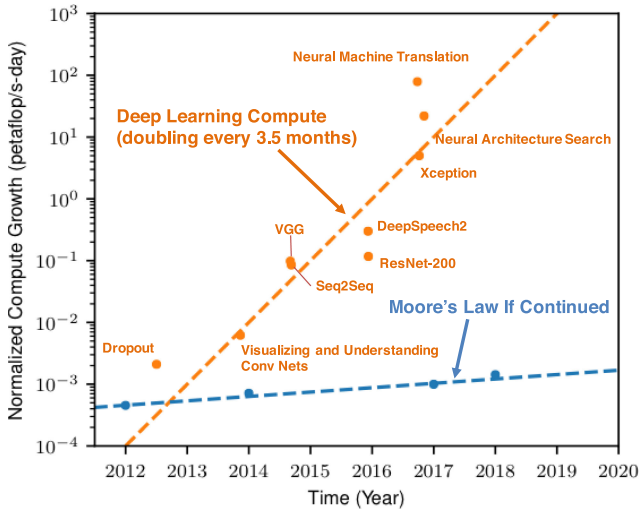

Fig. 1. High-performance computing is dominated by deep learning which is quickly saturating available compute growth [15]. The orange dots show the total amount of compute, normalized to petaflop/s-day, that was used to train each of selected neural network architectures [9], [11], [42]–[49]. The blue dots show the trend of Moore's law. (A petaflop/s-day is the number of operations of performing 10^{15} operations per second for one day, which in total is 8.64×10^{19} operations).

tasks. However, electronic architectures face fundamental limits as Moores law is slowing down [16]. Furthermore, moving data electronically on metal wires has fundamental bandwidth and energy efficiency limitations [17], thus remaining a critical challenge facing deep learning hardware accelerators [18]. Photonic processors can significantly outperform electronic systems that fundamentally depend on interconnects. Silicon photonic waveguides bus data at the speed of light. The associated energy costs are currently on the order of femtojoules per bit [19] and, in the near future, attojoules per bit [20], [21]. Aggregate bandwidths continue to increase by combining multiple wavelengths of light (i.e., wavelength-division multiplexing (WDM)), theoretically topping out at 10 Tb/s per single-mode waveguides using 100 Gb/s per channel and up to 100 channels. On-chip scaling of many-channel dense WDM (DWDM) systems may be possible with comb generators in the near future [22].

Recently, there has been much work on photonics processors to accelerate information processing and reduce power consumption using: artificial neural networks [23]–[27], spiking neural networks [28]–[35], and reservoir computing [36]–[39]. By combining the high bandwidth and efficiency of photonic devices with the adaptive, parallelism and complexity attained by methods similar to those seen in the brain, photonic processors have the potential to be at least ten thousand times faster than state-of-the-art electronic processors while consuming less energy per computation [40], [41].

However, even though the analog operations at the core of neuromorphic photonic processors exhibit greater efficiency and speed compared to digital hardware, most of AI tasks, including those shown in Fig. 1, need to be interfaced with electronic systems, thereby costing significant overheads in these metrics. As a result, neuromorphic photonics should not be understood as a one-size-fits-all deep learning hardware accelerator, but as a promising way of executing complex tasks using artificial neural networks to process gigahertz parallel signals in real-time for which digital hardware would have been unsuitable. Some of these photonic neural networks can also be trained using traditional deep learning algorithms, even though they are not designed to accommodate the diversity of deep neural networks commonly studied by computer scientists.

C. How Neuromorphic Photonics Can Improve Machine Learning

The primary goal of neuromorphic photonics is to physically emulate a neural network in real-time as efficiently and as fast as possible. In machine learning terms, this corresponds to performing the equivalent of *inference*. Secondarily, it provides a way to accelerate training algorithms for neural networks, aiding in the process of *training*, which requires much more advanced implementation and computing power. Ordinarily, in machine learning, training a network is performed in data-center-grade hardware over long periods of time. The trained network is then deployed to the "edge" of computing networks, which use smaller hardware, e.g. mobile phones, IoT devices or embedded systems, for inference and analytics collection. In this paper, we will focus on using neuromorphic photonics for enhancing *inference* tasks because 1) it forms the compute basis for all machine learning and 2) it maps well to pure photonic hardware.

a) Inference Acceleration: Neural networks have been used in many real-world tasks primarily due to the accelerating progress in computational efficiency of electronics. Many of these tasks require a neural network inference engine at its core, performing real-time classification, pattern matching, nonlinear optimization or even sophisticated control algorithms, especially in robotics (Sec. II-A). In these applications, training can be performed separately and in low-frequency intervals, like a "software upgrade", and is not crucial for the well-functioning of the task.

Model-Predictive Control is an example of such an application explored thoroughly throughout the paper. It is based on solving a multi-variable quadratic optimization problem with linear constraints (Eq. 1) in order to compute the next actuation step. The computation time of the solution is critical and determines the latency of the controller (lower is better). The lower the latency, the more the controller can accommodate fast-changing, unstable systems [50]. State-of-the-art electronic controllers can offer latency on the order of 10 ms [51], compared to 10ns order-of-magnitude that the neuromorphic photonic circuit that we will detail in Sec. II-B can achieve.

b) Parallelism: As mentioned earlier, neuromorphic photonics combines the interconnect advantages of photonics and the computation efficiency of electronics for emulating neural networks in hardware. Most technological breakthroughs mentioned in Sec. I-B rely on making the memory and the processing units close to each other and as distributed as possible. That is the operational principle of GPUs and TPUs, which are specialized to execute expensive matrix-multiplication operations across large arrays of data. The compiler slices the data and distributes them into separate units operating in parallel. Similarly, in neuromorphic photonics, the compute power is physically distributed across the neural network, with each neuron performing small bits of computation in parallel. The configuration memory

of the network is distributed in the tuning of optical components, which perform weighted addition inside each *photonic neuron* (Sec. IV-B).

c) Passive Interconnects: Photonic neurons are passively interconnected with optical waveguides, each bussing data with ~4 THz bandwidth. Moreover, these interconnects are passive and *switchless*, requiring no dynamic power to route data between neurons. They are also *clockless*, with information flowing from one layer of neurons to another at the speed of light, enabling sub-nanosecond latency between subsequent layers. Finally, they are *scalable*, because the cost of moving data stays virtually constant with increasing distances due to low waveguide loss.

d) Ultrafast Optoelectronics: Each neuron can use ultrafast optoelectronic devices as a nonlinear unit [52]: e.g. excitable lasers with sub-nanosecond pulses [28] or modulators with tens of GHz speed [23] (cf. Sec. IV-C). In this strategy, neural networks can enjoy the energy efficiency of real-time analog signal processing while being able to encode information with digital amplitudes (e.g. spikes) thereby being robust to noise accumulation.

D. Envisioning a Neuromorphic Processor

In neuromorphic photonics, there is an isomorphism between the analog artificial neural networks and the underlying photonic hardware, which allows continuous functions to be fully represented in an analog way. An analog representation of information avoids overhead energy consumption and speed reduction caused by sampling and digitization into binary streams processed by clocked logic gates. But because of this analog representation, we cannot dissociate the information that flows through the neural network from the photonic physics that impacts distortion, noise and loss. This prevents computer engineering researchers from obtaining a good understanding of the trade-offs and constraints of neuromorphic photonics. At the same time, it also makes it very hard for device engineers to understand how individual device metrics can affect the performance of an entire application.

In this paper, we seek to weave a thread between these two extremes. On the one hand, we propose a processor architecture with an abstraction hierarchy that can be understood by computer engineers and machine learning experts (Sec. V). On the other hand, we will consider what physical effects have an important role in building neuromorphic processors based on an integrated photonics platform (Sec. VI).

e) Neuromorphic Processor Architecture: At this stage of photonic manufacturing and packaging platforms, we envision a photonic integrated circuit containing a reconfigurable neural network dedicated to performing inference tasks (*Processor Core*, Fig. 10), surrounded by auxiliary devices such as laser sources, waveform generators, photodetector arrays, and erbium-doped fiber amplifiers. This core is controlled by a command & control micro-controller, an autonomous circuit whose task is to guarantee that the processor core is running according to its program. The overall processor is interfaced with the real world by a conventional digital controller, e.g. FPGA, CPU and RAM, which stores programs, learning algorithms, data, and higher-level commands for the low-level micro-controller.

f) Hardware Considerations: Analog hardware are known to introduce corrupting noise to signals. The presence of noise affects how much optical power is necessary to sustain a certain signal-to-noise ratio (SNR) (hardware metric) and a certain performance benchmark (application metric). Limited dynamic range of analog neurons and imprecision of weight elements can cause signal distortions that must be accounted for by an accurate hardware model. In addition, loss must be compensated with amplification to maintain cascadability of neural layers. These non-idealities must be taken into account at a systemic level, with individual devices working in concert to compensate them. For example, networks can be trained using extra neurons in order to mitigate inaccuracies caused by noisy signals. Training algorithms can also take into account global hardware constraints and fabrication variations. This means that trade-offs in speed, power consumption and size are computed at a system level and are therefore application-specific – it *cannot* be an extrapolation of individual devices' performance.

g) Photonic Platforms: Integration platforms for photonics also dictate how practical and how efficient neuromorphic photonic circuits can be (Sec. VII). The most mature technology is *silicon photonics*, whose high-volume manufacturing allows for the most repeatable and robust platform for photonic circuits. Using silicon as a substrate also enables greater compatibility with digital electronic technology, allowing more compact solutions for neuromorphic hardware. A great disadvantage of silicon photonics is the reliance on external lasers, typically built in III–V platforms, which require difficult and expensive co-packaging solutions. There are many applications driving the research community to find an industry-compatible solution for lasers-on-silicon, with good candidates such as III–V/Si hybrid fabrications, or quantum dot lasers grown directly on silicon. Industrial experts predict enabling innovations in the next five years that will allow neuromorphic photonic processors to be fabricated in a single die.

E. Organization of the Paper

This introduction walks through the intersection between machine learning (ML) and neuromorphic photonics (NP). The rest of the paper is organized as follows: *Applications* → *Artificial Neural Networks* → *Photonic Physics* → *Processor Architecture* → *Hardware Considerations* → *Compatible Platforms* → *Design & Simulation*.

Specifically, Section II describes computing tasks suitable for NP, with a particular example of model-predictive control (Appendix A). Section III provides a model for an artificial neural network compatible with neuromorphic photonic hardware, and gives an intuitive example of how neural networks can perform general analog computations. In Section IV, we comment on key properties of photonic devices that render them as uniquely suitable candidates for ultrafast neuromorphic computing. Section V introduces a processor architecture that interfaces a photonic integrated circuit with a general-purpose computer (Appendix B contains a more formal hardware description). In Section VI, we introduce hardware constraints and trade-offs

that only exist at the neural network level. In Section VII, we provide an roadmap of fabrication platforms for consecutive generations of neuromorphic photonic hardware. Finally, in Section VIII we overview methods for accurate circuit simulation of an NP chip, which are necessary for layout validation and functional verification of NP processors. The appendices provide very important concepts, but contain more technical details and requires deeper background in other areas, and for that reason, they were separated from the main text of the paper.

Because of the wide scope of this paper, we chose to present the concepts in each section at an introductory level. The sections are generally organized in increasing order of specificity and complexity. The reader will find more detail in the references or appendices reviewed therein. That said, some sections, such as V, VI, VIII and the appendices, sit at the cutting-edge of the field and, as a result, should be read more as a roadmap than a review.

II. Applications of Neuromorphic Photonics

The introduction presented neuromorphic photonics as a means to build ultrafast, reconfigurable hardware that is capable of solving real-world machine learning tasks. The processor architecture that will be introduced in this paper is mostly suited for analog-dominant, highly-parallel, high-speed applications for which digital processing struggles to process in real-time. As a result, this section discusses a few tasks that take particular advantage of the low-latency and high-parallelism of photonics. We identify three main classes of tasks, and then we go into the details of a quadratic program problem as an example task revisited throughout the paper.

A. Classification of Tasks

These tasks can be categorized into three: nonlinear programming, feedforward inference, and feedback control.

Nonlinear programming refers to optimization problems with nonlinear objective function and/or nonlinear constraints. These problems are computationally very expensive for digital computers, so applications that depend on nonlinear programming are limited to low-speed tasks. Neural networks (NNs) can solve some of these problems with a specially-configured recurrent neural network, such as a "Hopfield network". [53]–[55]. There are many problems that can be translated into a well-defined quadratic program (QP), and here we will study an "iterative" problem that requires a QP solution per time step.

Feedforward inference refers to problems that require a machine to compute functions of inputs as quickly as possible, e.g. tracking the location of an object via radar. These problems are typically well-behaved in the sense that the outputs change continuously with small changes to inputs. But they also include classification tasks in which the outputs represent discrete answers or decisions for different inputs. The latter problems are very common in modern machine learning and are prevalent in most AI systems.

Feedback control is the most challenging because it relies upon interaction with a changing outside environment. It is similar to feedforward inference, except that the network must be reconfigured as a result of the output of the neural computation, therefore requiring recurrent connections and sometimes short-term memory. Alluding to the tracking example above, this feedback can allow the network to keep tracking the object even in the presence of vibration or partial views. Neuromorphic photonics can enable new applications because there is no general-purpose hardware capable of dealing with microsecond environmental variations.

B. Nonlinear Programming: Quadratic Programming, Model-Predictive Control

There are a number of high-speed control problems – for example, controlling plasma in aircraft actuators, fusion power plants, guiding of drones, etc. – that are currently bottlenecked by the ability to perform high-speed, low-latency computations. Model-predictive control (MPC) is an advanced technique to control complex systems, and is widely used in chemical plants and refinaries [51]. MPC outperforms traditional PID control methods because it estimates effects of possible actions a few steps in the future, but that adds a lot of complexity to the control law. As a result, MPC has lacked computational tractability at speeds higher than kHz because of the limitations of electronics [51]. Thanks to photonics, these limitations can be overcome and the control law can be computed in up to hundreds of MHz. Here, we show that a neuromorphic processor can enable MPC for plants with sub-microsecond stability timescales.

To demonstrate the idea of high-speed control with MPC, instead of considering the control problem of chemical plants, we consider an example of tracking a moving target with matching speed while respecting constraints on position and acceleration. Suppose that the moving target is in a two-dimensional space (say x-y plane) with the reference trajectory $(y(t), x(t)) = (t, 2\sin(t))$, and the goal is to approach the moving target under the constraints $|y| < 1, |a_x| < 4, |a_y| < 4$, where t is time and a_x, a_y are the acceleration in x and y direction respectively. The goal of the neuromorphic photonic processor is to sense the target's position and control the acceleration of the tracker.

Translating this task into a neuromorphic processor configuration requires three steps (Fig. 2). First, we mathematically map the MPC task into a quadratic program (QP). Second, we compute the optimal configuration of a recurrent neural network capable of converging to the solution of the QP problem in real time. Finally, based on the network parameters, we configure a photonic neural network that emulates the mathematical model. The derivations are given in Appendix A.

a) Step 1: Mathematically, this problem is equivalent to solving an optimization problem with quadratic objective function and a set of linear constraints [56] at discrete time steps:

$$\begin{aligned} &\min_{\vec{\mathcal{X}}} \frac{1}{2}\vec{\mathcal{X}}^T \mathbf{P}\vec{\mathcal{X}} + \vec{q}^T\vec{\mathcal{X}} \\ &\text{subject to: } \mathbf{G}\vec{\mathcal{X}} \leq \vec{h}, \end{aligned} \tag{1}$$

where $\vec{\mathcal{X}}$ represents the change of the control variables (i.e. the change of acceleration); $\mathbf{P}$ models the tracker's response to these changes; $\vec{q}$ represents the relative position and velocity between tracker and target; $\mathbf{G}$ is similar to $\mathbf{P}$ and represents which parts

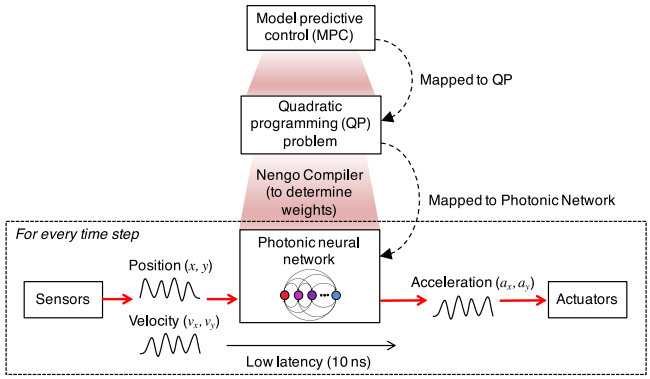

Fig. 2. Schematic figure of the procedure to implement the MPC algorithm on a neuromorphic photonic processor. Firstly, map the MPC problem to QP. Then, construct a QP solver with continuous-time recurrent neural networks (CT-RNN). Finally, build a neuromorphic photonic processor to implement the CT-RNN. The details of how to map MPC to QP, and how to construct a QP solver with CT-RNN are given in Appendix A.

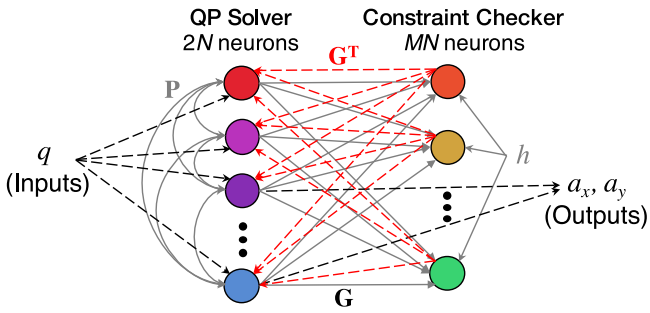

Fig. 3. Schematic figure of construction of a QP solver with CT-RNN. In this example, $N = 3$, which is the prediction horizon, $M = 6$, which is the number of inequalities, and 2 is the vector dimension.

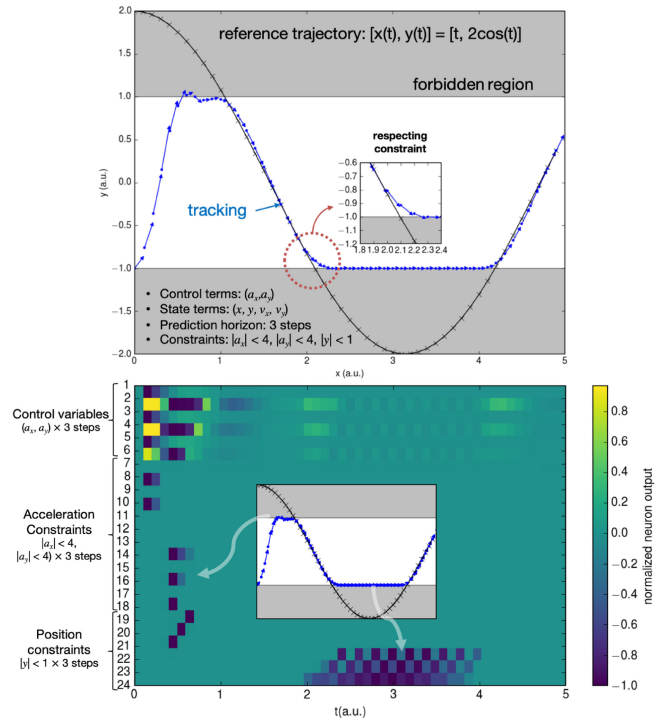

Fig. 4. (top) The trajectory of the moving target is shown in the black curve, and the blue dots and blue arrows are the simulated results of the position and velocity of the tracker at each time step respectively. The inset shows that the controller predicts a constraint violation and starts turning the tracker to avoid violating the acceleration's constraint. (bottom) The "constraint checker" neurons fire around $t = 0.5$ and between $t = 2$ and $t = 4$, inhibiting the output of the "QP solver neurons" such that the outcome of the system does not violate the acceleration and position constraints, respectively.

of the motion of the tracker's we want to constrain; and finally $\vec{h}$ represents how far we are from violating the constraints.

b) Step 2: Equation 1 can be solved using a network with 24 neurons with the configuration shown in Fig. 3. These are enough to accommodate a prediction horizon of 3 steps, the number of steps in the future modeled by MPC. The neurons can be separated into two populations. The first population consisting of 6 neurons represents the control variables a_x, a_y in the prediction window. The second population formed by the other 18 neurons represents the constraints of the system (6 constraint inequalities $\times$ 3 steps of prediction horizon). The first population is configured with a weight matrix based on $\mathbf{P}$, which is fixed between steps, and bias vector based on the vector $\vec{q}$, which varies between steps. The second population is configured with $\mathbf{G}^T$ (fixed) and $\vec{h}$ (variable), so that it is activated only near the constraint boundaries (Fig. 3), inhibiting the first population of neurons (Fig. 4). In this scheme, the first population of neurons converges to the solution of Eq. 8.

c) Step 3: Here, we show the tracking task and the simulated behavior the recurrent network in Fig. 4. The key to accelerate the MPC algorithm is to solve the QP problem quickly. Photonic neural networks offer a convergence time in the order of 10 ns, about a million times faster than the state-of-the-art electronic devices (~10 ms) [57]. Sections III, IV, and V explore in more detail how model-predictive control can be programmed in a neuromorphic photonic processor and this example will be revisited later.

III. Artificial Neural Networks

Three key elements are present in artificial neural networks: a nonlinear networkable node (neuron), interconnection (network) and information representation (coding scheme). A simple but common model of a neuron is shown in Fig. 5. In this model, neural networks are interconnected in a weighted directed graph, in which the connections are called synapses. The input of a neuron (s_j) carries a weighted sum of the outputs from other neurons (x_i). Then, the neuron integrates the sum over time and produces a nonlinear response, called an *activation function*, which typically looks like a threshold function. The output is broadcast to all succeeding nodes in the network. As we will see in Sec. VI, the weights can be positive (excitatory) or negative (inhibitory), but must be finite, and can be represented by a real number. The interconnections of the network can be described by a *weight matrix*. Programming the network can be done via a *training* algorithm, which understands how real-valued variables (information) can be encoded and decoded on the network, and optimizes the weight matrix

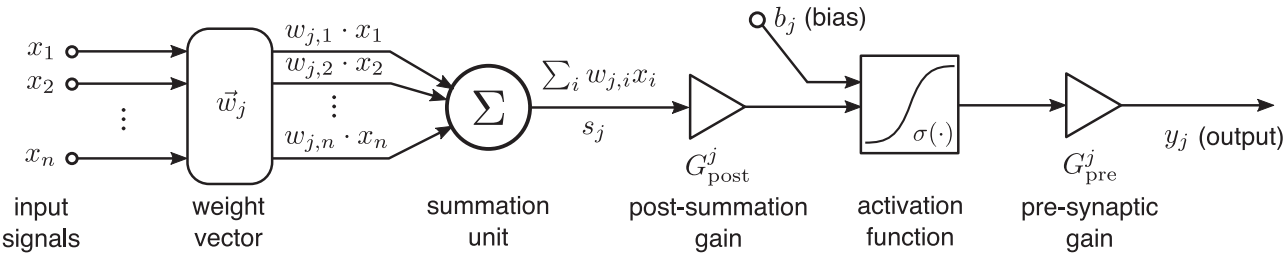

Fig. 5. Building blocks of an artificial neuron with simple synaptic model. The output is to be connected to other neurons, forming a neural network. This model can be described by Eq. 2.

so that the decoded output matches the expected outputs in a training set.

Most neural networks used in machine learning have continuous activation functions, which have achieved state-of-the-art classification accuracy in a wide variety of tasks [58]. These networks can be theoretically demonstrated to be universally able to carry out digital and analog computations [59]. More recently, *spiking* neurons were investigated following experimental results from neuroscience, which realized that some tasks solved by our brain cannot be solved by conventional neural networks at the same speed. Spiking neural networks have a richer modeling capacity and higher power efficiency and have inspired research in electronic [60], [61] and photonic spike processing [52]. There has been many investigations into a variety of optical neuron systems [62]–[64]. Recently, we demonstrated an integrated excitable laser acting as a spiking neuron [28], [65], [66] capable of integrating multichannel sub-nanosecond signals and 'firing' spikes when incoming pulses correlated in time, which is exactly the functionality that conventional networks cannot reproduce with the same efficiency. In this paper, we will focus on conventional neural networks, which can be modeled with a simple first-order ODE (Eq. 2), where $y_j(t)$ is the output of the j^{th} post-synaptic neuron, $x_i(t)$ is the input from the i^{th} pre-synaptic neuron, τ_j is the time constant of the j^{th} neuron, $w_{j,i}$ is the weight of connection from the i^{th} to j^{th}neuron, b_j is the bias of the j^{th} neuron, G^j_{pre}, G^j_{post} are the pre- and post-synaptic gain of the j^{th} neuron, and $\sigma_j(x)$ is the activation function of the j^{th} neuron node. Modeling and processing with spiking neural networks is far more complex, and it is out of the scope of this paper. SNNs are still under active research by computational neuroscientists, whose goal is to understand how cognitive skills emerge from the self-organization of spiking neural networks [67], [68].

$$\tau \dot{y}_j = -y_j + G^j_{\text{pre}} \left[\sigma_j \left(b_j + G^j_{\text{post}} \sum_i w_{j,i} x_i \right) \right] \quad (2)$$

A. Strategies of Photonic Neuromorphic Hardware

As discussed in Sec. I-D, there is a physical isomorphism between ANNs and the photonic hardware. In it, information flows through the hardware in the analog domain, susceptible to propagation loss and noise accumulation. This problem is solved in digital systems by representing information in binary electrical signals, which can be thresholded by logic gates and its errors eliminated by error correction codes. But in an analog neuromorphic processor, analog data manipulation requires a careful consideration of a large number of physical trade-offs, constraints, and dependencies as systems are designed.

Therefore, we need to model neurons under hardware-realistic assumptions. These assumptions are very common in machine learning and at the same time very amenable for photonics.

- Synaptic weights are simply real-valued weights without any temporal filters. Since signals are analog, they can only be weighted by values between -1 and 1. This can be overcome by adjusting the gain of a post-synaptic amplifier.
- Each weight will have a certain precision expressed in bits, e.g. 4-bit weights can have 16 independent values from -1 to 1. This is because weight control circuitry are limited by the calibration complexity and DAC precision.
- There is no synaptic plasticity.
- Summation is completely linear.
- The bandwidth of the weighted sum s_j is at most as large as the bandwidth of individual signals x_i.
- The nonlinear node has to obey a simple hysteresis-free first-order ODE with short-memory, such as Eq. 2, or in the case of spiking, with a short refractive period.
- The neuron model is subject to various sources of noise, as detailed in Sec. VI-A, which must be taken into account during training.

B. Programming the Network

The processing capabilities of the network rest more in the connectivity and weights than the transfer function of the nonlinear units. As a rule of thumb, any activation function, so long as it is well-behaved (e.g. continuous, bounded and monotonic), are candidates for neurons which can be trained to accommodate the requires transformation. The price to be paid for having simple neurons is that bigger networks are necessary. Fig. 6 shows an example of a single neural layer encoding the value of $x \in [0, 1]$, and the weights necessary to produce representations of x, x^2 and x^3. Then we can use a linear combination of those weights to represent any third-order polynomial in $f(x) = ax^3 + bx^2 + cx + d$. Note that although the figure only shows one example of polynomial approximation, multiple polynomials can be approximated in parallel. NNs can be

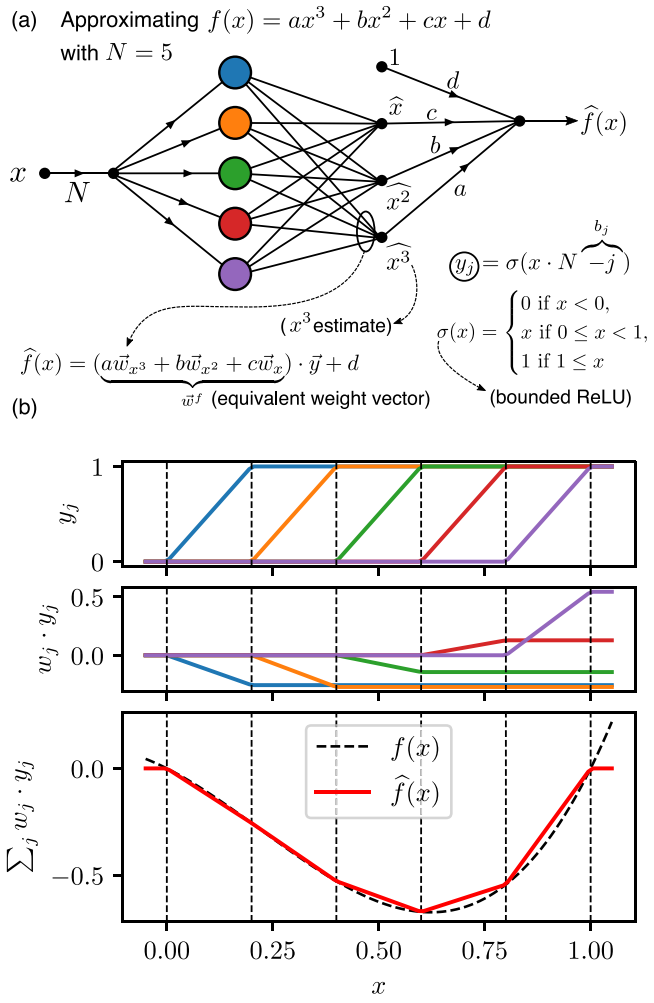

Fig. 6. Example of polynomial approximation ($f(x) = 3x^3 - x^2 - x$) by one layer of neurons with a bounded ReLU activation function. Here, $N = 5$ neurons "interpolate" a polynomial curve in five points in the interval [0,1]. The weight matrix computation was done in two steps, to ease understanding. First, we computed optimal weights for interpolating x, x^2, x^3, and second, used these weight vectors to produce an equivalent weight vector expression for estimating any third-order polynomial $f(x)$.

made to approximate any such functions to any degree of precision provided that enough neurons are present [69].

h) Training: In the example above, we can *train* the weights by a supervised learning algorithm. It goes like this: we generate one million random samples of $x_i \in [0, 1]$, then we compute $f(x_i)$, and store the pairs into a *training set* in one million rows. The training task consists in finding the right neural network shape and the right weight matrix that best approximates the function $f(\cdot)$. Because we knew the relation $f(\cdot)$, we were able to manually craft a neural network that solves the task (see Fig. 6), but this approach also works for unknown functions f. The good news is that for known functions and known network shape, it is possible to deterministically *compile* the best weight matrix for the job with a software called Nengo [69].

i) Inference: Once the network is trained, it is ready to *execute* what is is programmed for (e.g. computing $f(x)$ with low latency). In AI, this is called *inference*. Inference requires different hardware metrics than training. The primary metric in inference is latency, as opposed to updating weight values. Photonic neurons could outperform at inference especially since their latency is roughly determined by the speed of light.

C. Learning: Online and Offline

Whenever there is a change in the nature of the data being treated by the machine, it is necessary to *retrain* the network to prepare for these changes. This is called *learning*. It can be done online, i.e. gradually being reconfigured as it performs inference, or offline, where a separate computer retrains the network based on a batch of new training data and halts the inference hardware to reconfigure it.

Neuromorphic computing generally follows a slow-learning principle, which distinguishes it from a deep learning hardware accelerator. That means that the reconfiguration rate (learning) is much slower than the data rate (processing). As a result, learning algorithms can be significantly sophisticated and implemented in a co-integrated digital circuit with dedicated memory. The applications listed in Sec. II-A all require a slow-learning rate, e.g. milliseconds, but fast processing, e.g. nanoseconds.

Online learning can be performed by an iterative update rule, which evaluates some characteristic property of the output of the network (or neurons) and then computes a gradient for the weight matrix. Computing the gradient involves many inferences, yet it itself does not require ultrafast hardware. Thus, learning can be implemented by a dedicated circuit co-integrated with the inference circuit. Co-integration is important because learning rules are more complex than inference, so they will likely not be implemented by pure photonics. Software neural networks can implement online learning at the expense of reduced inference speed, but hardware neural networks require dedicated circuit elements in the processor, requiring significantly more size, weight, and power. Section V presents neuromorphic processor architectures that enable new opportunities for learning in neuromorphic photonics.

IV. Photonic Physics

A. O/E/O

Photonics is uniquely great for creating parallel channels of communication that do not interfere with each other. This is mainly due to the non-interacting nature of photons. This same property is the one that makes "computing" with light so hard. Because of that, we cannot employ all the tools we learn in the quantum mechanics of solid state semiconductors to manipulate photons as well as we do electrons. For the same reason that electronic digital gates are possible, electronic interconnects have limited performance in communication. Due to cross-talk, two parallel wires cannot transmit twice as much information as an individual wire.

Neuromorphic photonics takes advantage of both approaches and offers optoelectronic hardware that can support high-throughput, scalability, and reconfigurability at the same time. This results in an *O/E/O* approach, in which inputs are optically multiplexed and weighted, but by summing them,

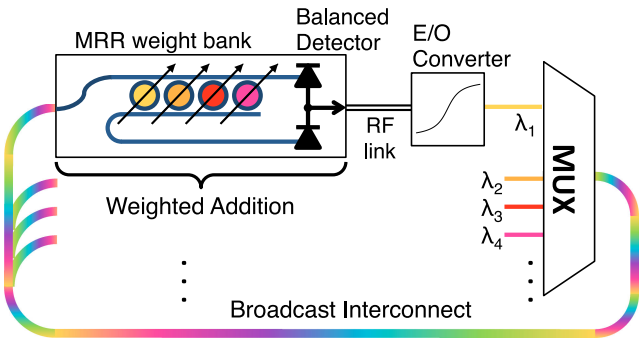

Fig. 7. Concept of an integrated broadcast-and-weight network [35]. A microring resonator (MRR) weight bank provides the key functionality to configure connection strengths in the analog wavelength-multiplexed (WDM) network. Tuning each MRR between the on- and off-resonance states determines how much of a given WDM channel is split between 2 ports of a balanced photodetector. The detected signal drives an electro-optic (E/O) converter, such as a laser, which generates a new optical signal at a unique wavelength. Figure is taken from [70].

get converted into a high-speed electronic photocurrent (O/E, Fig. 7), which then drives an electrically-driven light source (E/O) [41] (Fig. 5). In 2014, we proposed an architecture called broadcast-and-weight [35] based on this principle that was examined experimentally in [70]. Weights implemented by microrings were shown to exhibit weight accuracy and precision of 5.1 bits [71].

B. WDM Channels

Guided light waves can have multiple independent degrees of freedom, or 'modes': wavelength, polarization, spatial modes. Optical fibers optimize these degrees of freedom to increase data transmission capacity.

A neuromorphic photonic network requires the ability to weight individual channels prior to summation at the "soma".[1] Thus, signals representing the outputs of individual neurons must be individually addressed in one of these channels. And synaptic connections to a particular neuron must be implemented with multiplexing and demultiplexing circuits that interconnect the networks.

There have been many proposals for creating neuromorphic synaptic weights with wavelength-division multiplexing (WDM) or spatial-division multiplexing (SDM). The approach our group adopts is WDM, in which all channels coexist in a single waveguide attached to all neural units. We use microring resonator (MRR) weight banks to provide simultaneous filtering and weighting functionality. The total number of WDM channels available in a single waveguide is limited by the finesse of microring weights and photodetector bandwidth, plus an insertion loss to weightability ratio derived in [70]. Resonators typical of silicon photonic platforms with finesse of 368 [72] could support 108 channels in a one-pole configuration or 367 channels using the two-pole enhancement, which we showed in [73].

[1] Soma is the main body of the neuron, which aggregates the sum of all synaptic inputs.

This fan-in number can be understood as the total number of input synaptic weights for each neuron. Because all channels are in a single waveguide (called a broadcast inteconnect waveguide cf. Fig. 7), these signals can be broadcast to all neural units attached to it. However, as networks are usually not all-to-all-connected, the total number of neurons could be much greater. This enables multi-level broadcast hierarchy, facilitating very complex and realistic types of networks [74].

C. Nonlinear Dynamics

The third component of the neuron is a nonlinear unit, capable of applying the function $\sigma(\cdot)$ (see Sec. III). The kind of nonlinearity in the transfer function can be divided into spiking or non-spiking, leading to two schemes of neural networks: continuous-time (CT) neural networks and spiking neural networks (SNNs). Neither scheme requires specific transfer functions between input and output (see Sec. V), so long as they have a nonlinear transfer function (for example a ReLU or a sigmoid function). This nonlinearity is also important for noise suppression and cascadability (see Sec. VI-B).

There is a range of optoelectronic devices that are capable of displaying neuron-like behavior. They range from electrically injected excitable lasers to all-optical bistable laser cavities. [52]. In order to be compatible with a neural model, however, we need an electrically-injected single-wavelength light source (E/O). This can be in the form of a standard laser or modulator, or an excitable laser. For microfabrication purposes, this device needs to be co-integrated with the networking circuit, e.g. in silicon photonics. We note that choosing a different light-generation device, for example optically injected lasers, is possible, but requires rethinking the entire network architecture from scratch, since that removes the advantage of using a photodetector for WDM summing while generating current. There is no simple way to perform all-optical summation of several WDM signals, therefore a new networking architecture, different from that of B&W, must be constructed to use other multiplexing schemes (see Sec. IV-B).

j) Example 1: MRR Modulator for Artificial Neural Networks: Recently, Tait *et al.* showed the first silicon-photonic modulator neuron, depicted in Fig. 8. This paper [23] is the first to demonstrate a photonic neuron that is compatible with both silicon photonics and a well-defined network architecture that implements broadcast-and-weight with tunable spectral filter banks. By showing bistability in a neuron that can drive itself (as an autapse), it therefore follows that the neuron can drive other identical neurons, thereby completing the picture of a silicon photonic network fully compatible with mainstream silicon photonic foundry platforms. The results on the right of Fig. 8 depict the modulator neuron's response to two inputs (A and B) at different wavelengths. For addition, they are sent into the same port of the neuron's balanced PD; for subtraction, they are sent into complementary ports. It demonstrates that the neuron is capable of fan-in, converting two inputs at different wavelengths into one output at one wavelength. Depending on the bias, the neuron can have a linear transfer function (i.e., $A + B$ or $A - B$) or a rectifying transfer function (i.e., $(A + B)^2$ or

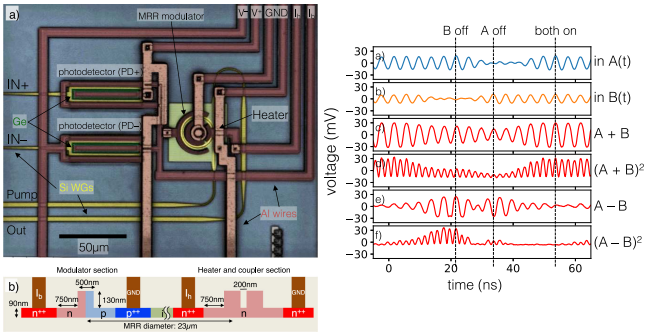

Fig. 8. Left: (a) False color confocal micrograph of a silicon microring (MRR) modulator neuron. Two photodetectors (PDs) are electrically connected to an MRR modulator, resulting in an O/E/O transfer function. (b) Cross-section of the MRR modulator with embedded PN modulator and N-doped heater. Right: Modulator neurons performing burst addition and two channel rectification with two separate inputs (wavelengths) illustrating excitatory and inhibitory behavior. From [23].

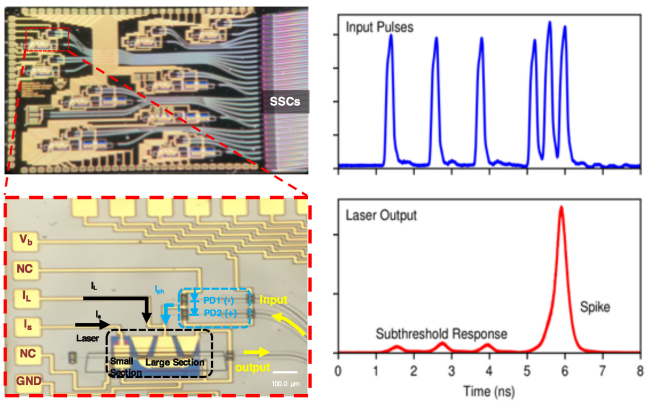

Fig. 9. Left: Micrograph of an excitable laser. The chip is an indium phosphide-based device fabricated by Heinrich Hertz Institute. I_L, I_s are the current put into large and small section respectively. The photocurrent I_{ph} generated by PD2 flows into the large section under a reverse bias condition. The output of the two-section DFB and the input of PD2 travel through waveguides coupled to benchtop instruments via a V- groove fiber array. Right: Nonlinear response of the excitable laser. Reproduced from [28].

$(A-B)^2$). The fact that input optical signals affect changes in the output optical signal is significant because that output could, in principle, be fed to other neurons; furthermore, the fact that multiple signals can be "weighted" by positive and negative values and their sum then influencing the output is an indicator that the MRR neuron can be networked with multiple inputs and outputs.

k) Example 2: Excitable Laser for Spiking Neural Networks: One design of using an excitable laser as a neuron's nonlinear unit is shown in Fig. 9 [28]. The excitable laser consists of two electrically isolated and optically coupled distributed feedback (DFB) sections. This current-pumped semiconductor laser can be excited or inhibited by a perturbation in its injected current, which is from the photodetectors attached to it. This was the first demonstration of the photodetector-driving concept (proposed in [75]) applied to excitable lasers. As shown on the right side of Fig. 9, the laser's excitable behavior allows it to exhibit both

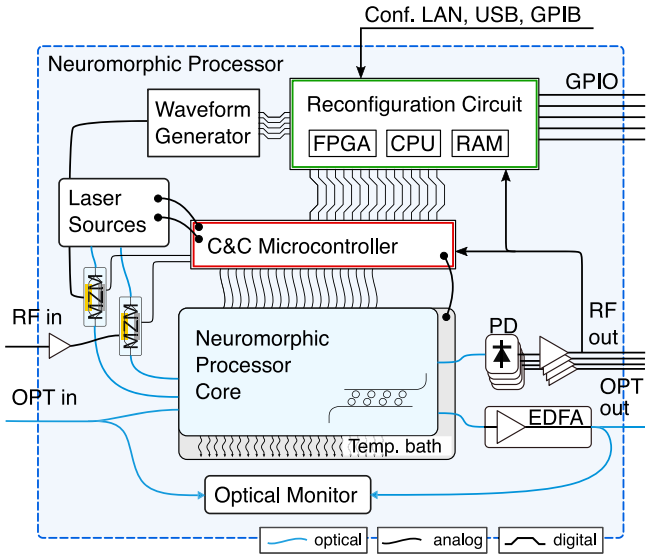

Fig. 10. Simplified schematics of a Neuromorphic Processor. Thanks to integrated laser sources and photodetectors, it can input and output RF signals directly as an option to optically-modulated signals. The waveform generator allows for programming arbitrary stimulus that can be used as part of a machine learning task.

integration and thresholding, which are the main functionalities of a spiking neuron. This platform is the first step to construct a spiking neural network.

V. Neuromorphic Processor Architecture

This section describes a vision for how to create a useful neuromorphic processor and how it can be interfaced with a general-purpose computer from a user's perspective to achieve specific applications. First, we need to take a step back and understand the differences between programming for general-purpose processors and for application specific computing with application-specific integrated circuits (ASICs). A CPU processes a series of computation instructions in an undecided amount of time and is not guaranteed to be completed. Neural networks, on the other hand, can process data in parallel and in a deterministic amount of time.

Unlike conventional processors, the concept of a 'fixed' instruction set on top of which computer software can be developed is not useful for ASIC hardware. Here, the neuromorphic processor is composed of custom components with specific applications and specific instructions, which cannot generalize to a common software language. As a result, a hardware description language (HDL) is more appropriate because it describes the intended behavior of a hardware in real-time. In Sec. V-A, we will discuss the need for a high-level processor specification that users can interface with, while offering more details, including a prototype circuit definition of an ideal 'neuron' written in an HDL, in Appendix B. This is followed by Sec. V-B, in which we describe how to compose different circuits to build a neuromorphic photonic processor.

A. Processor Firmware Specifications

When we speak of neural networks, there are two layers of abstraction: physical and behavioral. The physical layer contains the set of neuromorphic circuits necessary for emulating neural networks. The behavioral layer describes how information is encoded, transformed, and decoded as it flows along a network, and how the network should learn new behavior from new information. The former will be discussed in Section VIII, and the latter explored here.

As neuromorphic processors attain higher levels of technological readiness, they need to be understod by potential users without background in integrated photonics. New hardware development must invariably be done in tandem with software specification. Here, HDLs are useful because they describe circuits in a way that a computer can understand and simulate. At the same time it is a *specification*, which gives hardware engineers the freedom to implement it in different platforms. It creates an abstraction hierarchy that breaks down the circuit into different levels of detail, from a large structure such as a digital memory module e.g., in a conventional processor, to the individual transistor level.

This abstraction is necessary to allow integrated photonics professionals to be able to build neuromorphic processors *to spec*. It also allows them to simulate speed and power consumption before sending a chip layout for manufacture – these metrics depend not only on the performance of individual photonic devices inside a chip, but also more importantly on system tradeoff choices.

As an example, suppose that a particular neural network that executes an inference task can be implemented using a conventional or a spiking neural network. Both of these networks require different coding schemes, but could be used to accomplish the same task with different efficiencies and speed. It is obvious that these coding schemes require different hardware, but they also require different control algorithms and network configuration. That is why it is important to be able to express the function of the neuromorphic circuit without fixing the hardware, as is exemplified in Table III (Appendix B).

B. Architecture Components

l) Processor Core: A "photonic neuron" is a device containing three sub-unit: a weighting, a summing, and a nonlinear unit (see Fig. 5), that can be scalably networked with other neurons. Because of this network aspect, we defined this interpretation of a neuron as a *processing-network node* (PNN) [41, Sec. 2]. In this paper, the word 'neuron' in the context of photonics must be understood as a PNN. Using WDM-compatible neurons, there is a compatible networking architecture that uses a broadcast-and-weight protocol for interconnections [35], [41, Sec. 4]. The general idea is that a single broadcast waveguide loop can hold N independent WDM channels which can be interfaced by any photonic neuron (corresponding to a unique wavelength) attached to it. These broadcast loops can be connected with each other in a cellular-network hierarchy, by reusing the wavelength spectrum of silicon photonics (Sec. VII-A). Readily available silicon-photonic foundries can already implement all of the components of a high-density broadcast-and-weight system [23], [25] containing $\sim 10^4$ weights/mm^2 with current routing overhead (200%). This corresponds to an equivalent of 10 TMAC/s/mm^2[2] processing power with 30 fJ/MAC efficiency, for 7-bit analog MACs. This is in comparison to digital electronic architectures currently under development, which are in the 0.5 TMAC/mm^2—pJ/MAC range [28]. But as detailed in Sec. VI-E, these metrics can only be properly compared when tallying the aggregate size and power consumption of the full processor architecture, not just the MAC computations, especially if its data stream is not analog. A photonic integrated circuit (PIC) containing a reconfigurable PNN lies at the *core* of a neuromorphic photonic processor's architecture (Fig. 10 (A)). It can have internal or external laser sources, depending on the platform (Sec. VII), and electrical or optical I/O, but it needs to be controlled by an electronic circuit, referred to here as a *Command & Control* circuit.

m) Command & Control Circuit: Beyond being susceptible to fabrication variations (as discussed in Sec. VI-D), PICs are sensitive to thermal fluctuations and electronic damage. The *Command & Control* circuit (Fig. 10(B)) ensures that the inference circuit is well calibrated and run as intended. It takes a desired set of weights, and then synthesizes information from external laser parameters and embedded optical power monitors to achieve these weights. The fundamental technology for this control has been previously demonstrated in silicon photonics [71], [76]. This micro-controller has a very high analog DC I/O count because each electronically-controlled weight in the neuromorphic photonic processor requires a unique analog input, therefore it must contain an analog-to-digital interface that can be configured digitally by a *reconfiguration* circuit. Circuits based on this design should be able to reprogram about 10000 weights in less than one millisecond, a subject of current research.

n) Reconfiguration Circuit: The reconfiguration circuit (Fig. 10 (C)) receives instructions from a CPU, live-data from the environment and the state of the command and control (C&C) circuit and makes decisions about how the network is to be configured in real-time. It is best implemented with a combination of interconnected FPGA, CPU, and RAM modules.[3] This circuit acts as the boundary between the photonic engineers and the digital hardware programmers. Therefore, it must be the one that receives the instructions (synthesized and assembled from an HDL program) and takes care of not only configuring the core processor but also handling training, on-line learning, and digital and analog interconnects.

o) Interfacing with the Real World: The entire processing unit described here is being useful to a bigger application. For example, the MPC task requires high-speed RF inputs and outputs, as shown in Fig. 10, and the neuromorphic processor is capable of completing the task in the analog domain, without the need

[2]MAC: multiply-and-accumulate operations, i.e., operations of the form $a = a + w \times x$ for signals x, weight w, and accumulate variable a. The performance of such systems are typically measured in MAC/s or J/MAC.

[3]FPGA: Field-programmable gate array. CPU: Central processing unit. RAM: Random-access memory. These are common modules in modern digital hardware.

for expensive and power-hungry high-speed digital-analog conversion. Furthermore, the processor cannot be insulated from the rest of the plant; it might need different control solutions depending on temperature conditions, time-of-day, humidity or even human-made decisions. An example is temperature gradient control: in addition to causing all the resonances to shift in silicon photonic elements, they can also cause linewidth and gain spectra shifts in lasers. In some cases, they can completely change lasing conditions or cause some wavelengths to turn off, due to limitations such as gain clamping. The system must be designed to account for that, gathering as much information as possible from the environment and from onboard sensors. As a result, it is crucial to maintain a high-bandwidth communication link with a computer motherboard, represented as GPIO in Fig. 10.

VI. Hardware Considerations

A. Signal-to-Noise Ratio

Are noisy signals and noisy circuits a problem for neural networks? Neuroscience has shown that the neural circuits in our brain operate under a tremendously noisy environment, and yet it has clearly been robust to noise. The main reason for this is that brains use redundant neural circuits to encode and process information, so that damage in one neural pathway can be corrected and properly compensated by downstream circuits in the cortex. The secondary reason is that neural algorithms themselves do not have the same expectations of exactitude as digital algorithms. With that as an inspiration, one can design neural circuits that are specifically robust to noise. Noise-aware training can be used to prepare an ideal network for noisy data [77]. The particular case in which the networks themselves are imperfect are discussed in Sec. VI-D.

The sources of noise in hardware can be modeled mathematically as additive noise terms in Eq. 2. The training procedure should take this noise into account. Curiously, in some machine learning techniques, noise and defects are often artificially added to neural networks in order to prevent overfitting effects.

$$\tau\dot{y}_j = -y_j + N_{\mathrm{pre}} + G^j_{\mathrm{pre}} \cdot \cdot \left[\sigma_j\left(b_j + G^j_{\mathrm{post}}\sum_i (w_{j,i} + N_w)x_i + N_{\mathrm{post}}\right) + N_{\mathrm{E/O}}\right] \tag{3}$$

where:

- N_w: Weight precision, originated from electronic fluctuations from the C&C circuit.
- N_{post}: Post-summation amplifier noise. In neuromorphic photonics, this can correspond to a transimpedance amplifier (TIA) placed in the RF link between O/E and E/O 7. This noise is dependent on optical intensity (e.g. shot-noise) and can be modelled as Gaussian.
- N_{pre}: Pre-synaptic amplifier noise. In neuromorphic photonics, this can be generated via amplified spontaneous emission (ASE) of the optical amplifier. This can be modelled as dependent on the fan-out of the neuron.
- $N_{\mathrm{E/O}}$: Nonlinear-unit noise. In neuromorphic photonics, this can be generated by the relative intensity noise (RIN) of a laser source. This can be modelled as dependent on the average E/O bias and average laser source power.

B. Cascadability

An important requirement in neuromorphic hardware is the notion of cascadability: the ability of one neuron to excite and communicate with a number of other neurons. This number is called *fan-out*. Many all-optical and optoelectronic elements exhibit nonlinear input-output transfer functions, but this does not mean they can drive other like devices. Optics faces special challenges in satisfying the critical requirements of cascadability and fan-in [78], [79].

The need for cascadability stems directly from the isomorphism between analog artificial neural networks and the underlying photonic hardware, as discussed in Sec. I-D. This means that taking advantage of neuromorphic photonics requires a one-to-one correspondence between every neuron in a neural network and their hardware counterpart. For example, if a neural network classification task requires 5 layers with 10 neurons each, one needs to use hardware with at least 50 neurons to implement it. An alternative would be to use the same 10 neurons to emulate a 5-layer deep network, iteratively storing its output, reconfiguring weights to represent the next layer, and reinjecting inputs, repeating 5 times. This approach is undesirable because although it uses fewer neurons, the limited memory bandwidth and power efficiency would impose both latency and power overhead.

p) Gain Cascadability: Gain cascadability means the ability of one neuron excited with a certain strength to evoke an equivalent response in a downstream neuron. In other words, the differential gain must be greater than the fan-out. High-gain optical-to-optical nonlinearity is difficult to achieve using nonlinear optics. In optical devices based on semiconductor modulation or Kerr effect, the output signal (probe) affects the material properties in the same way as the input signal (pump). This necessitates weak probes and very small pump-to-probe gains (e.g. the fiber neurons in [80], [81]). One approach to increase nonlinearity strength is with resonant cavities.

q) Physical Cascadability: In addition to having enough gain, the output must be of the same physical format as the input. Resonant devices often impose constraints on wavelengths such that the output wavelength is necessarily different from the input [31], [63], [82]. While one such neuron might be able to drive another, the second then cannot drive the first. Optical signals have a phase degree of freedom that also must be accounted for. Neurons whose behavior changes depending on phase [83] can only be robust and repeatable if they introduce ways to regenerate the output phase. Some interconnect schemes are phase-dependent, which means they would require neurons that regenerate optical phase into a known state in order to cascade through a second interconnect layer [26]. Ultrafast devices that can regenerate phase have yet to be proposed. Cascadability conditions can be met with an O/E/O signal pathway that can accept inputs at any wavelength and output at any desired wavelength [35], [75], [84], [85].

When the physical cascadability condition is met, the neuron should be able to drive itself. A methodology for demonstrating gain and physical cascadability, employed in [23], [25], [33], is to connect a neuron to itself and show that two different stable states can be maintained. Here, we assume that the implicit condition of input-output isolation is satisfied. This concept means that the output generated by one neuron should not disrupt its input (which can be shared with other neurons).

An interesting feature of cascadable optical neurons that also have input-output isolation is that one can directly instantiate a recurrent neural network (RNN), because the optical signals can be directly fed back to the network without the need for memory storage. As an example, in the application described in Sec. II-B, the recurrent network's output is accessed only after it has converged to a solution, i.e. after many "iterations".

r) Noise Cascadability: In analog neuromorphic processors, neuron variables are represented by physical variables, such as the power envelope of a lightwave signal. When an optical signal splits, it gets weaker. With enough attenuation of the signal, noise begins to corrupt the signal. This process determines the maximum fan-out possible to maintain signal integrity. This limitation can be modelled mathematically as constraints in the configuration parameters of the network. For example, $\sum_i |w_{ji}| < I_j/I_0$, where w_{ji} here indicates a weight value and I_j indicates the strength (or laser power) of neuron j and I_0 a certain noise floor.

Noise cascadability has been thoroughly studied before [86]. Assuming that this noise is generated in an uncorrelated way, there are two methods for avoiding it to propagate across a network. The first is using multiple redundant neurons to encode the same signal, because signals add linearly, but noise is suppressed via the central limit theorem. ($\sum_i^n S + N_i \rightarrow n \cdot S + \sqrt{n} \cdot N$)

The second way is to make use of the transfer function of the nonlinear unit to effectively have a negative noise figure. The main idea is that the signal and the noise undergo different gains. This concept is clearly illustrated in Fig. 9, where only perturbations above the noise floor, such as a train of spikes, can trigger an output spike. If we can take advantage of that property, then there is no need to build redundant circuits for the sake of mitigating noise.

Noise can be mathematically modelled as in VI-A, and its effect on training can be compensated for, but the noise cascadability metric provides us with a figure of merit for neuromorphic hardware that is useful for benchmarking purposes.

C. Dynamic Range

The other challenge with analog signal processing in general is dynamic range, which is defined roughly as the ratio between the maximum and the minimum tolerable amplitudes of a signal. Analog devices have a fixed dynamic range, which needs to be taken into account (and not violated!) during the training step. Mathematically, this can be written as global constraint equation, e.g. $\forall j \ \sum_i w_{ji} \cdot x_i < I_{\max}/I_0$, where $I_{\max}/I_0$ is the dynamic range of the neuron's input photodetector.

Similarly, its output (see Fig. 5), y_j, is represented by a physical quantity (optical power), and is limited below by optical noise and above by maximum laser power.

D. Training Imperfect Networks

Neuromorphic hardware are prone to have manufacturing and environmental variations that are not always possible to be corrected by hardware circuits or quality assurance. It might also be that after extended usage, the same neuromorphic hardware experiences a drift in its internal parameters. It might also be possible that a neuromorphic hardware offered to the programmer contains neurons with different activation functions that closely resemble each other but are not exact.

These features must be taken into account by the training algorithm. So it is important that the statistics and the parameters described in this section be known to the assembler/trainer. Machine learning researchers have demonstrated methods to train neural networks with varying degrees of weight precision [87], with binary weights and activation functions [88], and with finite weight magnitudes as well [89].

This all means that two separate neuromorphic processors should be able perform the same functionality, despite fabrication variations, by pre-correcting the variations at the assembly step. Should a processor become too adrift, with enough deviation from the "normality", one could halt the assembly step to prevent malfunction or further damages to the photonic substrate.

E. Electronics vs. Photonics

Energy consumption is especially critical for scaling computing systems to larger processing densities, since modern digital chips today are largely limited by thermal dissipation limits. When looking at the bottlenecks of computing systems – and high performance computing (HPC) systems, in particular – there are two primary sources of energy consumption: data movement, and performing linear operations such as matrix multiplications. In highly parallel processors, i.e., the Google TPU [6] or an FPGA - data movement can be as large as 85% (or more) of the total energy cost.

Photonics has the potential to address these bottlenecks directly. Electronic interconnects dissipate power according to their capacitance, which scales linearly with the length of the wire. In contrast, photonic communication channels are nearly dissipationless outside of the fixed E/O and O/E conversion cost, and scale in a nearly distance-independent way. They can also carry a vast amount of information per cross sectional area, a property well known by the photonic interconnects community that can allow for bandwidth densities currently unheard of in the electronic domain up to this point [79].

Secondly, photonic linear operations are also nearly dissipationless, requiring Mach Zehnder arrays [26] or spectral filtering [35] for fully reconfigurable linear computations. This is in stark contrast to digital electronic systems, which require individual processing units for each linear operation, coupled with a communication system that allows for message passing between each node.

Although a detailed analysis of such trade-offs is outside the scope of this work, a simple example can be used to illustrate the power of using neuromorphic photonics to compute linear operations. Since the act of performing each operation is non-dissipative, one only pays the cost of generating and receiving the signals. For an $N \times N$ matrix operation, this cost scales with N rather than N^2 . Current digital systems require more than a half of a picojoule of energy for every MAC operation [6], [90]. The cost of an optical transceiver in an optical communication channel can easily be 1 pJ/bit or less [91]. If we use similar machinery to perform matrix operations (wherein each bit, in the previous case, becomes a time slot for a vector input and output) and use $N = 128$ wavelengths, our effective energy consumption would be less than 10 fJ. This is more than 50 times greater in energy efficiency than current state-of-the-art in digital electronics, without significant architectural changes to what is already available.

This practical number is far from fundamental. In particular, optical communication channels are expected to be pushed down to the low femtojoule range as more efficient photonic devices are employed with electronics scaled to smaller nodes [21]. This means that, since we can effectively divide the Joule/MAC efficiency by N as N increases, the fundamental limit is sufficiently in the low attojoule range.

In order to directly reap the benefits of this energy efficiency, we assume that the input and output of the processor are analog. Otherwise, A/D or D/A[4] data conversion costs could overwhelm energy savings of this approach, as well as limit total throughput but not its latency. Because of these uncertainties, a full comparison between neuromorphic photonics and electronics requires 1) an application or task to be evaluated, 2) identification of which devices are used and how much power they consume, and 3) and a dynamic vs. static power scalability analysis. There are still a number of practical problems that must be addressed before this is achievable (cf. Fig. 11) but there is a great deal of promise for the future efficiency of using neuromorphic photonics to perform computations over current approaches today.

VII. Fabrication Platforms

The ultimate goal of a neuromorphic processor is to be compatible and interfaced with the rest of the computer, i.e. it needs to be self-contained, robust to temperature fluctuations and vibrations, and with only electrical I/O. Therefore, devices on a neuromorphic processor (including light sources, passives, and actives) must all function together.

Currently, there is no single commercially available fabrication platform that can simultaneously offer high-quality devices for WDM weighting, high-speed photodetection, light generation, and low-power transistors on a single die; state-of-the-art devices in each of these categories use different photonic materials (silicon nitride, germanium, indium phosphide, gallium arsenide, etc.) with incongruous fabrication processes (silicon-on-insulator, CMOS, FinFETs, and others.)

[4]Analog-to-Digital and vice versa.

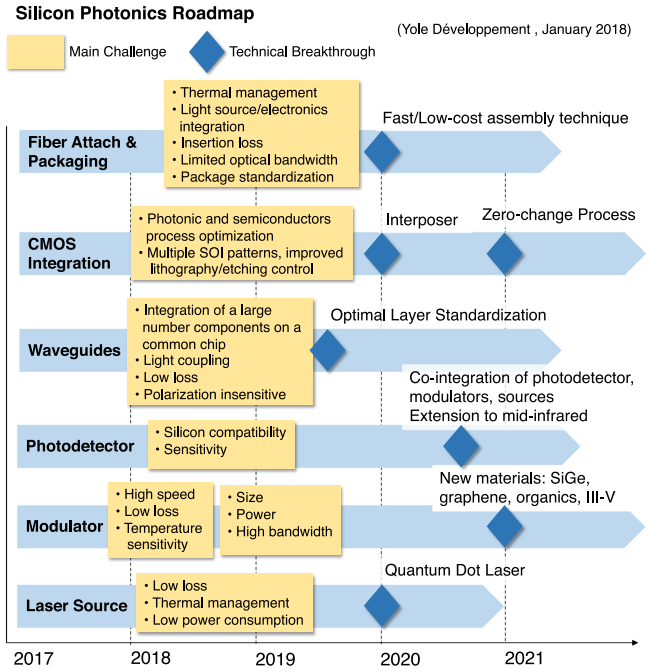

Fig. 11. Technological challenges facing silicon photonics. CMOS and laser source integration in addition to advanced packaging techniques will facilitate self-contained neural networks, while waveguide, photodetector, and modulator development will aid photonic neural network commercialization. Adapted from Ref. [92].

Here, we discuss a road-map of fabrication platforms for consecutive generations of neuromorphic photonic hardware, keeping in mind the commercial viability of each.

A. Silicon Photonics

Silicon is a natural candidate for a photonic neural network substrate because it is CMOS-compatible, which mediates low-cost, high-volume manufacturing and integration with electronics while utilizing silicon's transparency at optical communication wavelengths (i.e. 1270-1625 nm.)

Due to the wide range of applications for silicon photonics [93], it is currently at the maturity level of electronics in the 1980's [92]. Monolithic silicon photonic wafers can include high-speed active and passive elements such as modulators, photodetectors, and microring filters, but for full silicon photonics integration, package design must include a) silicon photonic die design, b) parallel waveguide interconnect technology, c) chip-to-waveguide assembly, d) thermal management, and e) electronic logic element integration with photonics ICs [94]. However, manufacturers do not currently assemble electrical/thermal elements and chip-to-fiber interconnects, among other challenges shown in Fig. 11 [94].

CMOS technology either requires bulk silicon substrates, or thin silicon-on-insulator (SOI) wafers, with the former being greater in supply and economic efficiency. Silicon photonics usually requires thick SOI wafers with a relatively lower supply chain that is more expensive. Electronics feature sizes are smaller than photonics, so fabrication is commercially

infeasible. However, these hurdles are being overcome with technology such as "zero-change" SOI platforms, in which all photonic devices are manufactured according to electrical foundry design flow, allowing transistors and photonic devices to be fabricated on the same chip, but with lasers off chip [95], [96].

B. III–V + Silicon Photonics

Waveguides and modulators, which form the backbone of photonic neural networks, are made of silicon, which has a refractive index ranging from 1.45 (oxide index) to as high as 3.48 (silicon index) [97] at the wavelength of 1550 nm that can be controlled thermally, electrically, mechanically (strain), or chemically (doping). Although silicon lasers are infeasible at room temperature due to its indirect bandgap, the large span of silicon's refractive index allows efficient evanescent coupling to waveguides made of III–V materials due to phase matching conditions at similar refractive indices. Using external lasers with monolithic silicon photonics requires precise alignment of light to the waveguide, which is difficult and expensive, so presently, it is not commercially viable [98]. Using solely a III–V platform would neglect silicon's advantages. A laser source manufactured directly on chip has therefore been a prime objective for silicon photonics, and silicon/III–V hybrid lasers are a key ingredient in spiking neural networks [28]. The two current approaches involve either III-V to silicon wafer bonding (heterogeneous integration) or butt-coupling with precise assembly (the hybrid approach) [99], [100].

s) Heterogeneous Integration: In a series of steps, lasers on III–V wafers are aligned and bonded to SOI wafers. SOI wafers implement passive rib waveguides by etching the top silicon layer, which is then optimized with additional steps for coupling to III–V waveguides on wafers that are later aligned and bonded on top of the SOI wafers [101].

Though companies such as Kaiam, Luxtera, and Kotura/ Mellanox have developed short-term solutions [94], lasers still require greater power efficiency, lower packaging cost, and better heat flow management in order to be economically viable for industrial application.

t) Quantum Dot Lasers: One potential solution to this problem is growing lasers on silicon. Typically, this results in defects at the interface between III–V materials and silicon. However, quantum dot (QD) lasers can alleviate these detrimental effects because QDs operate independently of one another. Therefore, the lasers can be grown directly onto silicon, but fabrication reliability does not currently reach commercial standards [102].

While present technology does not support commercially viable laser integration with silicon, the increase in demand for silicon photonics in addition to promising research in QD lasers grown on silicon substrates does point towards a future with a commercial photonic neural network fabricated on silicon/ III–V chips, as demonstrated by (Fig. 11).

VIII. Design & Simulation

A successful circuit design should be able to predict complex system behavior in the presence of optical, electrical, or thermal stimuli [103]. The silicon photonics design process usually involves device design and simulation, circuit design and simulation, layout, verification, tape-out and mask preparation. For operational neuromorphic technology, a physical model and hardware simulation is necessary. Here, 'simulation' means a way to accurately compute the performance of an existing hardware in which individual components have been studied, fabricated, validated and optimized for neuromorphic processing.

A. Hardware Model of the Neuron

In digital electronic development, it is well known that trace lengths and shapes, metal pad geometry, and heat dissipation can lead to impedance mismatch, unintended low-pass filtering and overall system performance degradation. Photonic devices on the same chip can also affect each other, primarily because they are much more sensitive to heat then electronic gates. As a result, an accurate simulator that takes parasitics into account can help impose appropriate constraints in the final route and placement steps during layout.

However, software has not yet been developed to fully simulate a complex photonic circuit in this way. Typically, there are four different approaches to circuit simulation, described by Bogaerts and Chrostowski [103]: *a*) The first approach is to separate the electronic and photonic circuit simulators entirely, but it is not suitable for neuromorphic photonics because it lacks the ability to deal with the intrinsic optical-electrical-optical conversion detailed in Sec. IV-A. *b*) Another approach involves exchanging signals between simulators, which is sufficient for \-of-concept demonstrations but its computational requirement scales badly with larger networks. *c*) A third approach is to map photonic circuits to an electronic equivalent circuit and reuse the tools developed over the years for analog circuit simulation. This proved to be several orders of magnitude faster then than the previous approach [104], but it requires manual modeling of layout-level parasitics at a schematic level, which is fundamentally incompatible with electronic design automation (EDA) philosophy (see Sec. VIII-D). *d*) Finally, the photonics and electronics can be implemented in the same simulator. Current tools still do not offer full-scale optoelectronic simulation of large circuits, but private companies are making progress in this area [105].

In parallel to the development of simulation tools, we advocate that neuromorphic photonics should make use of a hardware abstraction layer, discussed in Sec. V-A and Appx. B, which will allow for a physically accurate neural network simulation without the need to capture the physics of individual optoelectronics devices.

B. Layout Floorplanning

Without automation, the current layout floorplanning approach is to optimize device placement with human intuition. Once the processor schematic is fixed, the layout designer can layout a neural network under the constraints given by the systems engineer.

For example, suppose that the same abstract neural network can be implemented in hardware in two ways that achieve the

same overall results. One way uses 'neurons' that are close together, clustered in one side of the chip with shorter traces to bond pads, and the other decides to use neurons scattered across the chip. In the first scenario, the network run with lower latency and higher speed due to the fact that the interconnects were simpler, but it would be more noisy and imprecise, because of thermal crosstalk and higher power requirements since control is more difficult. In the second scenario, the independent neurons would be less noisy and operate under tighter power budgets, but speed would be compromised, as well as lower accessibility to other parts of the chip.

The systems engineer can work with the layout designer to optimize the chip for lower power or raw speed, but in our experience, human layout design is expensive and time consuming, leading invariably to sub-optimal layouts. Automating that task will allow the systems engineer to exercise greater control over the performance of the chip.

C. Modular Layout & Simulation

While simulations do not exist for entire photonic networks, silicon photonics does provide infrastructure for laying out and simulating individual optoelectronic devices procedurally. This approach is based on programmable cells, or "PCells", programmed in scripting languages paired with software such as SKILL in Cadence, Ample in Mentor Graphics Pyxis, SPT in Phoenix Software, Python, which is used in IPKISS, KLayout, and Synopsys PyCell Studio, and Tcl, which is used in Synopsys, and Mentor Graphics or Matlab [103]

PCells such as microring modulators and filters are included in many silicon photonic process-design kits, and can be independently tested and verified experimentally by different research groups. With experimentally-validated data, a realistic *parametric* model can be written in a hardware-description language. More importantly, with the appropriate implementation, these models can be composed to form larger devices with some guarantees of performance, e.g. a 'neuron' PCell (cf. Appx. B).

D. Spiking Neural Network Simulation: A Case Study

Here we showcase a concrete example of how a spiking neural network can be physically simulated in the context of a concrete machine learning task.

As mentioned earlier, we have previously proposed an equivalent circuit-based simulator for two-section photonic excitable lasers (SIMPEL). By mapping the excitable laser's rate equations into a circuit model, SPICE analysis can be used as an efficient and accurate engine for numerical calculations, capable of generalization to a variety of different types of laser neurons found in literature [104]. On the other hand, compilers such as *nengo* [106] translate a task to a neural network configuration with minimal training via a Neural Engineering Framework (NEF). NEF is a methodology that organizes a neural network to compute desired neural functions from its inputs values and specified properties, solving for connection weight matrices between neural layers [107].

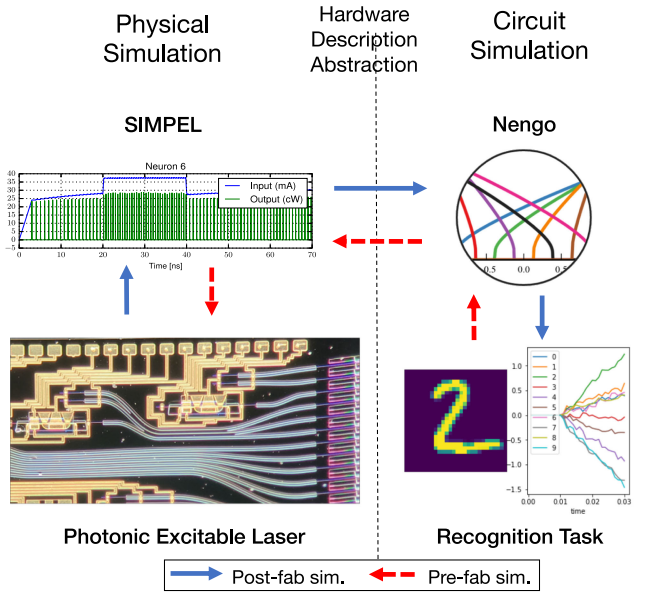

Fig. 12. An example of a hardware simulation procedure. For the post-fabrication simulation, given a photonic excitable neural network hardware design, we use a physical model simulator SIMPEL, which can simulate the behavior of the device and translate the physical parameters into neural network variables. Then, we plug the information from SIMPEL into Nengo to evaluate how accurately the hardware completes the task.

This process is depicted in Fig. 12, which shows the direction of simulation steps typically performed pre- and post-fabrication. In the case of a *modulator-based neuron*, there is a direct signal analogy between the network and the circuit model, explored in ref. [25]. However, for a spiking neuron, both simulators need to agree on the encoding of information in train pulses. In this case, it is typical to use the pulse-frequency to represent the strength of the signal (a strategy called 'rate-coding'). With that information, we know how to generate and collect signals to and from spiking photonic neural networks.

Similar to sub-circuits in FPGAs, photonic neural networks need to be assembled with optimized speed and power consumption, requiring a pre-fabrication simulation as shown in Fig. 12. For the pre-fabrication simulation, given a specific task that we want the hardware to implement, we use *nengo* to transform the problem into neural networks. Then, we plug the neural network parameters into the physical model simulator. The simulator will calculate the estimate power consumption, total computation time, etc. Based on this result, we can design our hardware with optimized physical parameters and performance estimates.

IX. Conclusion

This article walked through the important developments in the nascent field of *Neuromorphic Photonics* [40]. Researchers around the world have recognized the potential that photonics has to offer for ultrafast neuromorphic computing, but there is still a wide gap of understanding between integrated photonic device engineers and machine learning experts about how it can be integrated in a system. By presenting an overview of the field while zooming into specific details and models, we hope

this article clarifies how a neuromorphic processor can be built. Proof-of-concept devices and systems have already been demonstrated, and the field has reached the boundaries of Computer Engineering. This paper provided a roadmap for expanding research in the direction of transforming neuromorphic photonics into a viable and useful candidate for accelerating neuromorphic computing.

Appendix A
Photonic Neural Network to Solve MPC

The MPC algorithm can only be implemented if the feedback control loop latency is orders of magnitude smaller than the update rate. Thus, a neuromorphic photonic processor with a high-speed processing rate paves a way for the implementation of MPC algorithms. Here, we demonstrate how to implement an MPC algorithm on a neuromorphic photonic processor mathematically. The procedure can be divided in three steps:

a) Map an MPC algorithm to a quadratic programming (QP) problem mathematically [56]
b) Construct a QP solver with a continuous-time recurrent neural network (CT-RNN) [108]
c) Build the recurrent neural network (QP solver) in a photonic system

a) Map MPC to QP: Consider a discrete-time system with state variable $\hat{x}$, control input sequence of the system $\hat{u}$, and the measured output (observables) $\hat{z}$. At the k^{th} time step, dynamics of the system can be described as:

$$\begin{aligned}\hat{x}(k+1|k) &= \mathbf{A}\hat{x}(k) + \mathbf{B}\hat{u}(k)\\ \hat{z}(k) &= \mathbf{C}\hat{x}(k),\end{aligned} \tag{4}$$

where the notation $\hat{x}(k+i|k)$ represents the predicted result of state variable $\hat{x}$ at time step $k+i$. $\mathbf{A}, \mathbf{B}, \mathbf{C}$ are the constant matrices given by the system's equation of motion. For the example described in Sec. II, Eq. 4 can be written as:

$$\begin{aligned}\underbrace{\begin{bmatrix} x(t+\Delta t)\\ y(t+\Delta t)\\ v_x(t+\Delta t)\\ v_y(t+\Delta t)\end{bmatrix}}_{\hat{x}(k+1|k)} &= \underbrace{\begin{bmatrix} 1 & 0 & \Delta t & 0\\ 0 & 1 & 0 & \Delta t\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} x(t)\\ y(t)\\ v_x(t)\\ v_y(t)\end{bmatrix}}_{\hat{x}(k)}\\ &\quad + \underbrace{\begin{bmatrix} \Delta t^2/2 & 0\\ 0 & \Delta t^2/2\\ \Delta t & 0\\ 0 & \Delta t\end{bmatrix}}_{\mathbf{B}} \underbrace{\begin{bmatrix} a_x(t)\\ a_y(t)\end{bmatrix}}_{\hat{u}(k)}\\ \underbrace{\begin{bmatrix} x(t)\\ y(t)\end{bmatrix}}_{\hat{z}(k)} &= \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\end{bmatrix}}_{\mathbf{C}} \underbrace{\begin{bmatrix} x(t)\\ y(t)\\ v_x(t)\\ v_y(t)\end{bmatrix}}_{\hat{x}(k)},\end{aligned} \tag{5}$$

where the state variable is $\hat{x} = [x, y, v_x, v_y]^T$, the control variable is $\hat{u} = [a_x, a_y]^T$, and the measured output is $\hat{z} = [x, y]^T$, and $\mathbf{A}, \mathbf{B}, \mathbf{C}$ matrices are the corresponding terms shown in Eq. 5.

The reference trajectory is denoted by $\hat{r}(k)$, and the objective function is calculated in the form of:

$$\begin{aligned}V(k) &= \sum_{i=H_w}^{H_p} \|\hat{z}(k+i|k) - \hat{r}(k+i|k)\|^2\\ &\quad + \sum_{i=0}^{H_u-1} \|\Delta\hat{u}(k+i|k)\|^2,\\ &= \|\mathcal{Z}(k) - \mathcal{T}(k)\|^2 + \|\Delta\mathcal{U}(k)\|^2\end{aligned} \tag{6}$$

where H_w is the horizon window, H_p is the prediction horizon, H_u is the control horizon, $\Delta\hat{u}(k+i|k) = \hat{u}(k+i|k) - \hat{u}(k+i-1|k)$, and $\mathcal{Z}(k), \mathcal{T}(k), \Delta\mathcal{U}(k)$ are defined as:

$$\begin{aligned}\mathcal{Z}(k) &= [\hat{z}(k+H_w|k), \hat{z}(k+H_w+1|k), ..., \hat{z}(k+H_p|k)]^T\\ \mathcal{T}(k) &= [\hat{r}(k+H_w|k), \hat{r}(k+H_w+1|k), ..., \hat{r}(k+H_p|k)]^T\\ \Delta\mathcal{U}(k) &= [\Delta\hat{u}(k|k), \Delta\hat{u}(k+1|k), ..., \Delta\hat{u}(k+H_u-1|k)]^T\end{aligned}$$

The goal is to find the optimal $\Delta\mathcal{U}(k)_{opt}$ such that we can update the system variables and minimize the objective function. Minimizing $V(k)$ can be reduced to a QP problem [56]

$$\begin{aligned}&\min_{\vec{\mathcal{X}}} \frac{1}{2}\vec{\mathcal{X}}^T\mathbf{P}\vec{\mathcal{X}} + \vec{q}^T\vec{\mathcal{X}}\\ &\text{subject to: } \mathbf{G}\vec{\mathcal{X}} \le \vec{h},\end{aligned} \tag{7}$$

where $\vec{\mathcal{X}} = \Delta\mathcal{U}(k)$, $\mathbf{P}$ and $\vec{q}$, independent of $\Delta\mathcal{U}(k)$, are calculated by replacing terms in Eq. 4 to Eq. 6. $\mathbf{G}$ and h are given by the constraints of the system.

b) Construct a QP Solver with CT-RNN: The model of a continuous-time recurrent neural network (CT-RNN) can be expressed as a set of ordinary differential equations coupled through a weight matrix. For the i^{th} neuron in the CT-RNN, the time evolution equation is:

$$\tau_i\dot{y}_i = -y_i(t) + \sigma_i\left(\sum_{j=1}^{n} w_{j,i}x_j(t) + b_i\right), \tag{8}$$

where $y_i(t)$ is the output of the i^{th} post-synaptic neuron, $x_j(t)$ is the input from the j^{th} pre-synaptic neuron, τ_i is the time constant of the i^{th} neuron, $w_{j,i}$ is the weight of connection from the j^{th} to i^{th}neuron, b_i is the bias of the i^{th} neuron, and $\sigma_i(x)$ is the activation function of the i^{th} neuron node. Note the similarity between Eq. 8 and Eq. 2.

For simplicity $\sigma_i(x)$ can be a simple ReLU function:

$$\sigma_i(x) = \begin{cases}\alpha x = \alpha\sum_{j=1}^{n} w_{ji}x_j + J_i + I_{i,sig}, & \text{if } x \ge 0\\ 0, & \text{otherwise,}\end{cases} \tag{9}$$

where α is a positive constant given by the neuron's characteristic, J_i is the baseline activation term of the i^{th}neuron, $I_{i,sig}$ is the external input signal of the i^{th} neuron. The the relation of bias b_i to J and I_{sig} is $b_i = (J_i + I_{i,sig})/\alpha$. At the steady state of the CT-RNN, plugging Eq. 9 into Eq. 8, we find that:

$$(Id - \alpha W)\vec{y} = \vec{I}_{sig} + \vec{J}, \tag{10}$$

TABLE I
MAP BETWEEN NEURAL NETWORK ARCHITECTURE AND QP PROBLEM PARAMETERS

Neural Network Parameters	QP solver Neurons	Constraint Neurons
Weight Matrix	$(Id - \mathbf{P})/\alpha$	$\mathbf{G}/\alpha$
External Inputs	$-\vec{q}$	$-\vec{h}$
DC Bias of Neuron	$\mathbf{P}\vec{y_0}$	$-\mathbf{G}\vec{y_0}$

where Id and W are the identity matrix and the weight matrix of the network, respectively, $\vec{y} = [y_1, y_2, ..., y_n]^T$, $\vec{I}_{sig} = [I_{1,sig}, I_{2,sig}, ..., I_{n,sig}]^T$, and $\vec{J} = [J_1, J_2, ..., J_n]^T$. We can choose $\vec{J}$ as a baseline activation of a state $\vec{y_0}$ such that we have:

$$\vec{J} = (Id - \alpha W)\vec{y_0} \equiv \mathbf{P}\vec{y_0}. \tag{11}$$

Here, we identify $\mathbf{P}$ as $Id - \alpha W$. Thus, now the steady state solution of neurons can be expressed as:

$$\mathbf{P}(\vec{y} - \vec{y_0}) = \vec{I}. \tag{12}$$

If we identify $\vec{y} - \vec{y_0}$ as $\mathcal{X}$, and $\vec{I}$ as $-q$, then we get the solution of Eq. 7 under the unconstrained case (i.e. $\mathbf{G} = 0$). So given $\mathbf{P}, \vec{q}$, we can construct $W = \frac{1}{\alpha}(Id - \mathbf{P}), \vec{I} = -\vec{q}$, and the steady state solution of the neural network $\vec{y}$ will the the solution of QP shifted by $\vec{y_0}$. The result of $\vec{y_0}$ can be recovered from Eq. 11.

To deal with the constraints in the system, we construct another set of neurons to give the penalty to the QP solver neurons when the constraints are violated. The second population of neurons are connected to the QP solver neurons as shown in Fig. 3. The connection from the QP solver neuron is given by $\mathbf{G}$. For constraint neurons, the weight matrix $W = \mathbf{G}/\alpha$, the baseline activation is $\vec{J} = -\mathbf{G}\vec{y_0}$, and the external input to the neurons is $\vec{I} = -\vec{h}$. With this construction, the input to the constraint neurons will turn into:

$$W\vec{y} + \vec{b} = G/\alpha\vec{y} + (\vec{J} + \vec{I}_{sig})/\alpha = \frac{G(\vec{y} - \vec{y_0}) - \vec{h}}{\alpha}. \tag{13}$$

Notice that $\mathbf{G}(\vec{y} - \vec{y_0}) - \vec{h}$ represents the constraint of the system $\mathbf{G}\mathcal{X} - \vec{h}$. Thus, the constraint neurons will fire only if the constraint is violated. The penalty given by constraint neurons is fed back to QP solver neurons with the connection shown in the red dashed arrows in Fig. 3. The relation between Eq. 7 and the neural network can be summarized in Table.I:

c) Build the CT-RNN in a Photonic System: The photonic neuron architecture is discussed in Section IV. As shown in Fig. 7, each of the neuronal characteristics can be implemented by photonic devices. The activation function of a neuron node can be implemented by an excitable laser or a modulator. The weighted connection in a CT-RNN can be implemented by MRR weight banks. The bias of a neuron consists of DC and AC parts. The DC bias is controlled by the DC current injected to the excitable laser or modulator, and the AC perturbation is provided by the photocurrents resulting from optical external input. Here, in Table II, we provide a map for each term in the CT-RNN and their photonic counterparts.

TABLE II
MAP BETWEEN CT-RNN NEURAL NETWORKS AND PHOTONIC NEURAL NETWORKS

	CT-RNN Terminology	Photonics Equivalent
τ_i	Time constant of post-synaptic node	CMOS Amplifier RC time
y_i	Output of post-synaptic node	Actual averaged optical output power
w_{ji}	Weight of connection from pre- to post-synaptic node	Connection weights as calibrated by MRR Weight Banks [35]
$\sigma(x)$	Activation function of neuron	Nonlinear Transfer function for laser or modulator
b_j	Bias of pre-synaptic node	DC current injection and the photocurrent caused by external optical input

APPENDIX B
FORMALIZING HARDWARE DESCRIPTION

As explained in Sec. V-A, a hardware description language (HDL) is the most appropriate way of documenting a machine-readable specification for optoelectronic circuits, distancing hardware engineers from systems engineers. While being instrumental to the hierarchization of electronic circuits development, this tool has not been used in the optoelectronics community until very recently, with the development of Analox-Mixed Signal (AMS) extension of VHDL and Verilog, the two most popular HDLs [109], [110]. Here, we will use VHDL-AMS as an example, but the arguments can be expanded into other languages as well.

d) Design Automation: A VHDL-AMS code can be *synthesized*[5] directly into low-level logic gates, analog circuits, optoelectronic circuits, and a netlist. These must be placed and routed in a layout, a process called *implementation*. This process can be automated via *schematic-driven layout* [103, Sec. 4], or performed manually. More importantly, a physical simulator can then evaluate the performance of the circuit as it is laid out on chip and *back-annotate* the schematic, informing the circuit simulator about loss, timing, voltage drops, etc. This is called *Layout vs. Schematic (LVS)*. This abstraction is necessary to allow integrated photonics professionals to be able to build neuromorphic processors *to spec*.

e) High-Level Synthesis: Despite its high level of abstraction, writing HDL code by hand can become unwieldy to hardware engineers. That is why computer engineers created higher-level languages (e.g. *SystemC*) that can accommodate an algorithmic description of a behavior, which could then be synthesized to HDL. Electrical engineering students are already familiar with this concept when they program FPGA-based development kits or Arduinos. A high-level synthesizer *transcompiles*[6] their code into an HDL program and a Assembly machine code.[7] The HDL program is assembled into instructions for the embedded FPGA, and the machine code is sent directly to a memory unit that is read by a central processing unit (see Fig. 10). An abstraction like this would be very useful for neuromorphic computing in general, but in particular to neuromorphic photonics, since there is little

[5]Synthesis is the equivalent of compilation in programming languages.
[6]Transcompile means creating new source code, not necessarily at lower level.
[7]Assembly code contains instructions targeted to a particular CPU architecture.

TABLE III
VHDL-AMS (PSEUDO)CODE FOR THE NEURON MODEL DESCRIBED IN SECTION III. THIS CODE DEFINES AN IDEAL INTERFACE THAT CAN BE SIMULATED AND USED BY SOFTWARE ENGINEERS, AND IT SERVES AS SPECIFICATION FOR HARDWARE ENGINEERS

```
ENTITY neuron IS
  GENERIC (
    fan_in : NATURAL := 4;
    gain_post : REAL := 1;
    gain_pre : REAL := 1;
    transfer_function : FUNC := logistic_function
  );
  PORT (
    SIGNAL signal_in :
      in OPTICAL_MULTICHANNEL; -- multiplexed inputs
    SIGNAL signal_out:
      out OPTICAL_SINGLECHANNEL; -- single-channel output
    TERMINAL weight_in :
      in ELECTRICAL_DCARRAY(fan_in-1 DOWNTO 0);
    TERMINAL bias_in :
      in ELECTRICAL_HS;  -- high-speed bias
    TERMINAL ground_dc : ELECTRICAL_DC;  -- DC ground
    TERMINAL ground_hs :
      ELECTRICAL_HS;  -- high-speed ground
    TERMINAL vcc : ELECTRICAL_POWER;
    TERMINAL vdd : ELECTRICAL_POWER
  );
END ENTITY neuron;

ARCHITECTURE ideal of neuron IS
    QUANTITY v_w ACROSS i_w THROUGH weight_in to ground_dc;
    QUANTITY v_b ACROSS i_b THROUGH bias_in to ground_hs;
    VARIABLE s : REAL := 0;
    VARIABLE tau: REAL := 100E-9
    -- other specific quantities
BEGIN
    -- begin continuous-time process
    assert len(signal_in) <= fan_in
    -- pseudocode for weighted addition
    s := sum(weight_in[i] * signal_in[i]);
    signal_out'DOT == - (signal_out +  -- Neuron's ODE
        gain_pre * transfer_function(gain_post * s)) / tau
END ARCHITECTURE ideal;
```

user-friendly literature and knowledge available about photonic components compared to electronic circuits.

f) Field-Programmability: Another feature of high-level synthesis is that *users* of neuromorphic chips can reprogram the processor by editing the high-level behavioral SystemC code. Depending on the kind of application used (cf. Sec. II), for example, users might write a SystemC program that transcompiles to an HDL requiring recurrent or feedforward networks with continuous or spiking neurons, coupled with Assembly code with instructions for controlling a plant, optimizing a mathematical formula, or connecting to a network of processors.

g) Continuous Operation: Here, we envision the neuromorphic processor as plugged into a motherboard and belonging to a bigger computer or distributed network. In this model, an *operating system* (OS) is responsible for managing the processor's I/O and coordinate its execution with other conventional processors or computers. This OS should be able to manage complex operations such as turning the processor core on and off, storing/loading the state of the controller, or, in a low-energy/energy-saving mode, temporarily shut off lasers while waiting for more data in the pipeline.

ACKNOWLEDGMENT

The authors would like to thank Dr. David Rosenbluth and his team at Lockheed Martin's Advanced Technology Labs (ATL) for discussions on model predictive control.

REFERENCES

[1] P. A. Merolla *et al.*, "A million spiking-neuron integrated circuit with a scalable communication network and interface," *Science*, vol. 345, no. 6197, pp. 668–673, 2014. [Online]. Available: http://www.sciencemag.org/content/345/6197/668.full.pdf
[2] S. K. Esser *et al.*, "Convolutional networks for fast, energy-efficient neuromorphic computing," *Proc. Nat. Acad. Sci.*, vol. 113, no. 41, pp. 11 441–11 446, 2016. [Online]. Available: http://www.pnas.org/content/113/41/11441
[3] M. D. Pickett, G. Medeiros-Ribeiro, and R. S. Williams, "A scalable neuristor built with Mott memristors," *Nat. Mater.*, vol. 12, no. 2, pp. 114–117, 2013.
[4] M. Davies *et al.*, "Loihi: A neuromorphic manycore processor with on-chip learning," *IEEE Micro*, vol. 38, no. 1, pp. 82–99, Feb. 2018.
[5] A. Graves *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, pp. 471–476, Oct. 2016. [Online]. Available: https://doi.org/10.1038/nature20101
[6] N. P. Jouppi *et al.*, "In-Datacenter performance analysis of a tensor processing unit," 2017. [Online]. Available: http://arxiv.org/abs/1704.04760
[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, May 2015. [Online]. Available: http://dx.doi.org/10.1038/nature14539
[8] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds., Curran Associates, Inc., 2012, pp. 1097–1105. [Online]. Available: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf
[9] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014. [Online]. Available: http://arxiv.org/abs/1409.1556
[10] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2015, pp. 1–9.
[11] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2016, pp. 770–778.
[12] Y. Wu *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," arXiv 1609.08144, Sep. 2016.
[13] Y. Leviathan, "Google duplex: An AI system for accomplishing real-world tasks over the phone." [Online]. Available: https://ai.googleblog.com/2018/05/duplex-ai-system-for-natural-conversation.html
[14] D. Silver *et al.*, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–489, Jan. 2016. [Online]. Available: http://dx.doi.org/10.1038/nature16961
[15] D. Amodei and D. Hernandez, "AI and compute." [Online]. Available: https://blog.openai.com/ai-and-compute
[16] M. M. Waldrop, "The chips are down for Moore's law," *Nature News*, vol. 530, no. 7589, p. 144, 2016.
[17] D. A. B. Miller, "Device requirements for optical interconnects to silicon chips," *Proc. IEEE*, vol. 97, no. 7, pp. 1166–1185, Jul. 2009.
[18] Y. 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*, vol. 52, no. 1, pp. 127–138, Jan. 2017.
[19] E. Timurdogan, C. M. Sorace-Agaskar, J. Sun, E. Shah Hosseini, A. Biberman, and M. R. Watts, "An ultralow power athermal silicon modulator," vol. 5, Jun. 2014, Art. no. 4008. [Online]. Available: http://dx.doi.org/10.1038/ncomms5008
[20] S. Khan, Z. Ma, J. Jeon, C. J. Lee, and V. J. Sorger, "Sub 1-volt graphene-based plasmonic electroabsorption modulator on silicon," in *Frontiers in Optics 2017*, Opt. Soc. Amer., 2017, p. FM2A.4. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=FiO-2017-FM2A.4
[21] D. A. B. Miller, "Attojoule optoelectronics for low-energy information processing and communications," *J. Lightw. Technol.*, vol. 35, no. 3, pp. 346–396, Feb. 2017.
[22] B. Stern, X. Ji, Y. Okawachi, A. L. Gaeta, and M. Lipson, "Battery-operated integrated frequency comb generator," *Nature*, vol. 562, no. 7727, pp. 401–405, 2018. [Online]. Available: https://doi.org/10.1038/s41586-018-0598-9
[23] A. N. Tait, *et al.*, "A silicon photonic modulator neuron," arXiv preprint, arXiv:1812.11898, Dec. 2018.

[24] T. W. Hughes, M. Minkov, Y. Shi, and S. Fan, "Training of photonic neural networks through in situ backpropagation and gradient measurement," *Optica*, vol. 5, no. 7, pp. 864–871, Jul. 2018. [Online]. Available: http://www.osapublishing.org/optica/abstract.cfm?URI=optica-5-7-864

[25] A. N. Tait *et al.*, "Neuromorphic photonic networks using silicon photonic weight banks," *Sci. Rep.*, vol. 7, no. 1, 2017, Art. no. 7430. [Online]. Available: https://doi.org/10.1038/s41598-017-07754-z

[26] Y. Shen *et al.*, "Deep learning with coherent nanophotonic circuits," *Nat. Photon.*, vol. 11, pp. 441–446, Jun. 2017. [Online]. Available: http://dx.doi.org/10.1038/nphoton.2017.93

[27] J. M. Shainline, S. M. Buckley, R. P. Mirin, and S. W. Nam, "Superconducting optoelectronic circuits for neuromorphic computing," *Phys. Rev. Appl.*, vol. 7, Mar. 2017, Art. no. 034013. [Online]. Available: https://link.aps.org/doi/10.1103/PhysRevApplied.7.034013

[28] H. T. Peng, M. A. Nahmias, T. Ferreira de Lima, A. N. Tait, B. J. Shastri, and P. R. Prucnal, "Neuromorphic photonic integrated circuits," *IEEE J. Sel. Topics Quantum Electron.*, vol. 24, no. 6, pp. 1–15, Nov. 2018.

[29] I. Chakraborty, G. Saha, and K. Roy, "A photonic in-memory computing primitive for spiking neural networks using phase-change materials," 2018. [Online]. Available: http://arxiv.org/abs/1808.01241

[30] Y. Zhang, S. Xiang, X. Guo, A. Wen, and Y. Hao, "Polarization-resolved and polarization-multiplexed spike encoding properties in photonic neuron based on VCSEL-SA," *Sci. Rep.*, vol. 8, no. 1, 2018, Art. no. 16095. [Online]. Available: https://doi.org/10.1038/s41598-018-34537-x

[31] T. Deng, J. Robertson, and A. Hurtado, "Controlled propagation of spiking dynamics in vertical-cavity surface-emitting lasers: Towards neuromorphic photonic networks," *IEEE J. Sel. Topics Quantum Electron.*, vol. 23, no. 6, pp. 1–8, Nov. 2017.

[32] B. Romeira, R. Avó, J. L. Figueiredo, S. Barland, and J. Javaloyes, "Regenerative memory in time-delayed neuromorphic photonic resonators," *Sci. Rep.*, vol. 6, Jan. 2016, Art. no. 19510. [Online]. Available: http://dx.doi.org/10.1038/srep19510

[33] B. J. Shastri, M. A. Nahmias, A. N. Tait, A. W. Rodriguez, B. Wu, and P. R. Prucnal, "Spike processing with a graphene excitable laser," *Sci. Rep.*, vol. 6, Jan. 2016, Art. no. 19126. [Online]. Available: http://dx.doi.org/10.1038/srep19126

[34] A. Aragoneses, S. Perrone, T. Sorrentino, M. C. Torrent, and C. Masoller, "Unveiling the complex organization of recurrent patterns in spiking dynamical systems," *Sci. Rep.*, vol. 4, Apr. 2014, Art. no. 4696. [Online]. Available: http://dx.doi.org/10.1038/srep04696

[35] 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. Lightw. Technol.*, vol. 32, no. 21, pp. 4029–4041, Nov. 2014.

[36] F. Duport, A. Smerieri, A. Akrout, M. Haelterman, and S. Massar, "Fully analogue photonic reservoir computer," *Sci. Rep.*, vol. 6, Mar. 2016, Art. no. 22381. [Online]. Available: http://dx.doi.org/10.1038/srep22381

[37] D. Brunner, M. C. Soriano, C. R. Mirasso, and I. Fischer, "Parallel photonic information processing at gigabyte per second data rates using transient states," *Nat. Commun.*, vol. 4, Jan. 2013, Art. no. 1364. [Online]. Available: http://dx.doi.org/10.1038/ncomms2368

[38] K. Vandoorne *et al.*, "Experimental demonstration of reservoir computing on a silicon photonics chip," *Nat. Commun.*, vol. 5, Mar. 2014, Art. no. 3541. [Online]. Available: http://dx.doi.org/10.1038/ncomms4541

[39] L. Larger *et al.*, "Photonic information processing beyond turing: An optoelectronic implementation of reservoir computing," *Opt. Express*, vol. 20, no. 3, pp. 3241–3249, Jan. 2012. [Online]. Available: http://dx.doi.org/10.1364/OE.20.003241

[40] P. R. Prucnal and B. J. Shastri, *Neuromorphic Photonics*. Boca Raton, FL, USA: CRC Press, May 2017.

[41] T. Ferreira de Lima, B. J. Shastri, A. N. Tait, M. A. Nahmias, and P. R. Prucnal, "Progress in neuromorphic photonics," *Nanophotonics*, vol. 6, no. 3, pp. 577–599, Jan. 2017. [Online]. Available: http://www.degruyter.com/view/j/nanoph.2017.6.issue-3/nanoph-2016-0139/ nanoph-2016-0139.xml

[42] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Improving neural networks by preventing co-adaptation of feature detectors," *CoRR*, vol. abs/1207.0580, 2012. [Online]. Available: http://arxiv.org/abs/1207.0580

[43] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," *CoRR*, vol. abs/1311.2901, 2013. [Online]. Available: http://arxiv.org/abs/1311.2901

[44] V. Mnih *et al.*, "Playing atari with deep reinforcement learning," *CoRR*, vol. abs/1312.5602, 2013. [Online]. Available: http://arxiv.org/abs/1312.5602

[45] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *CoRR*, vol. abs/1409.3215, 2014. [Online]. Available: http://arxiv.org/abs/1409.3215

[46] D. Amodei *et al.*, "Deep speech 2: End-to-end speech recognition in english and mandarin," *CoRR*, vol. abs/1512.02595, 2015. [Online]. Available: http://arxiv.org/abs/1512.02595

[47] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *CoRR*, vol. abs/1610.02357, 2016. [Online]. Available: http://arxiv.org/abs/1610.02357

[48] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *CoRR*, vol. abs/1611.01578, 2016. [Online]. Available: http://arxiv.org/abs/1611.01578

[49] Y. Wu *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *CoRR*, vol. abs/1609.08144, 2016. [Online]. Available: http://arxiv.org/abs/1609.08144

[50] G. Stein, "Respect the unstable," *IEEE Control Syst. Mag.*, vol. 23, no. 4, pp. 12–25, Sep. 2003.

[51] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Trans. Control Syst. Technol.*, vol. 18, no. 2, pp. 267–278, Mar. 2010. [Online]. Available: http://ieeexplore.ieee.org/document/5153127/

[52] P. R. Prucnal, B. J. Shastri, T. Ferreira de Lima, M. A. Nahmias, and A. N. Tait, "Recent progress in semiconductor excitable lasers for photonic spike processing," *Advances Opt. Photon.*, vol. 8, no. 2, pp. 228–299, Jun. 2016. [Online]. Available: https://www.osapublishing.org/abstract.cfm?URI=aop-8-2-228

[53] J. J. Hopfield and D. W. Tank, "Computing with neural circuits: A model." *Science*, vol. 233, no. 4764, pp. 625–633, 1986.

[54] M. P. Kennedy and L. O. Chua, "Neural networks for nonlinear programming," *IEEE Trans. Circuits Syst.*, vol. 35, no. 5, pp. 554–562, May 1988.

[55] Y. M. Pan and J. J. Wang, "Two neural network approaches to model predictive control," in *Proc. Amer. Control Conf.*, 2008, pp. 1685–1690.

[56] J. M. Maciejowski, *Predictive control: With Constraints*. Harlow, U.K.: Prentice Hall, 2002.

[57] J. L. Jerez, G. A. Constantinides, and E. C. Kerrigan, "An FPGA implementation of a sparse quadratic programming solver for constrained predictive control," in *Proc. FPGA*, New York, NY, USA: ACM, 2011, pp. 209–218.

[58] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic, "Deep learning applications and challenges in big data analytics," *J. Big Data*, vol. 2, no. 1, p. 1, Feb. 2015. [Online]. Available: https://doi.org/10.1186/s40537-014-0007-7

[59] W. Maass, "Networks of spiking neurons: The third generation of neural network models," *Neural Netw.*, vol. 10, no. 9, pp. 1659–1671, 1997.

[60] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," in *Proc. Advances Neural Inf. Process. Syst. 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., Curran Associates, Inc., 2015, pp. 1117–1125. [Online]. Available: http://papers.nips.cc/paper/5862-backpropagation-for-energy-efficient-neuromorphic-computing.pdf

[61] F. Akopyan *et al.*, "TrueNorth: Design and tool flow of a 65 mW 1 million neuron programmable neurosynaptic chip," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 34, no. 10, pp. 1537–1557, Aug. 2015.

[62] H. J. Wünsche, O. Brox, M. Radziunas, and F. Henneberger, "Excitability of a semiconductor laser by a two-mode homoclinic bifurcation," *Phys. Rev. Lett.*, vol. 88, Dec. 2001, Art. no. 023901.

[63] S. Barbay, R. Kuszelewicz, and A. M. Yacomotti, "Excitability in a semiconductor laser with saturable absorber," *Opt. Lett.*, vol. 36, no. 23, pp. 4476–4478, Dec. 2011.

[64] F. Selmi, R. Braive, G. Beaudoin, I. Sagnes, R. Kuszelewicz, and S. Barbay, "Relative refractory period in an excitable semiconductor laser," *Phys. Rev. Lett.*, vol. 112, no. 18, 2014, Art. no. 183902.

[65] H.-T. Peng, M. A. Nahmias, T. F. de Lima, A. N. Tait, B. J. Shastri, and P. R. Prucnal, "Temporal dynamics of an integrated laser neuron," in *Proc. IEEE Photon. Conf. (IPC)*, Reston, VA, USA, 2018, pp. 1–2.

[66] M. A. Nahmias *et al.*, "A teramac neuromorphic photonic processor," in *Proc. IEEE Photon. Conf. (IPC)*, Sep. 2018, pp. 1–2.

[67] R. Brette *et al.*, "Simulation of networks of spiking neurons: A review of tools and strategies," *J. Comput. Neurosci.*, vol. 23, no. 3, pp. 349–398, Dec. 2007. [Online]. Available: https://doi.org/10.1007/s10827-007-0038-6

[68] E. M. Izhikevich, "Neural excitability, spiking and bursting," *Int. J. Bifurcation Chaos*, vol. 10, pp. 1171–1266, 2000.

[69] T. C. Stewart and C. Eliasmith, "Large-scale synthesis of functional spiking neural circuits," *Proc. IEEE*, vol. 102, no. 5, pp. 881–898, May 2014.

[70] A. N. Tait *et al.*, "Microring weight banks," *IEEE J. Sel. Topics Quantum Electron.*, vol. 22, no. 6, pp. 312–325, Nov. 2016. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7479545http://ieeexplore.ieee.org/document/7479545/

[71] A. N. Tait *et al.*, "Feedback control for microring weight banks," *Opt. Express*, vol. 26, no. 20, pp. 26422–26443, Oct. 2018. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-26-20-26422

[72] Q. Xu, D. Fattal, and R. G. Beausoleil, "Silicon microring resonators with 1.5-μm radius," *Opt. Express*, vol. 16, no. 6, pp. 4309–4315, 2008.

[73] A. N. Tait, A. X. Wu, T. F. de Lima, M. A. Nahmias, B. J. Shastri, and P. R. Prucnal, "Two-pole microring weight banks," *Opt. Lett.*, vol. 43, no. 10, pp. 2276–2279, May 2018. [Online]. Available: http://ol.osa.org/abstract.cfm?URI=ol-43-10-2276

[74] D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world' networks," *Nature*, vol. 393, no. 6684, pp. 440–442, 1998. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/9623998

[75] M. A. Nahmias *et al.*, "An integrated analog O/E/O link for multi-channel laser neurons," *Appl. Phys. Lett.*, vol. 108, no. 15, 2016, Art. no. 151106. [Online]. Available: http://scitation.aip.org/content/aip/journal/apl/108/15/10.1063/1.4945368

[76] A. N. Tait, T. Ferreira de Lima, M. A. Nahmias, B. J. Shastri, and P. R. Prucnal, "Multi-channel control for microring weight banks," *Opt. Express*, vol. 24, no. 8, pp. 8895–8906, Apr. 2016. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-24-8-8895

[77] M. L. Seltzer, D. Yu, and Y. Wang, "An investigation of deep neural networks for noise robust speech recognition," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, IEEE, May 2013, pp. 7398–7402. [Online]. Available: http://mi.eng.cam.ac.uk/{∼}yw293/pdfs/ICASSP2013a.pdfhttp://ieeexplore.ieee.org/document/6639100/

[78] R. W. Keyes, "Optical logic-in the light of computer technology," *Optica Acta: Int. J. Opt.*, vol. 32, no. 5, pp. 525–535, 1985.

[79] J. W. Goodman, "Fan-in and fan-out with optical interconnections," *Optica Acta: Int. J. Opt.*, vol. 32, no. 12, pp. 1489–1496, 1985. [Online]. Available: http://dx.doi.org/10.1080/713821684

[80] M. Hill, E. E. E. Frietman, H. de Waardt, G.-D. Khoe, and H. Dorren, "All fiber-optic neural network using coupled SOA based ring lasers," *IEEE Trans. Neural Netw.*, vol. 13, no. 6, pp. 1504–1513, Nov. 2002. [Online]. Available: http://dx.doi.org/10.1109/TNN.2002.804222

[81] D. Rosenbluth, K. Kravtsov, M. P. Fok, and P. R. Prucnal, "A high performance photonic pulse processing device," *Opt. Express*, vol. 17, no. 25, pp. 22 767–22 772, Dec. 2009. [Online]. Available: http://dx.doi.org/10.1364/OE.17.022767

[82] M. A. Nahmias, B. J. Shastri, A. N. Tait, and P. R. Prucnal, "A leaky integrate-and-fire laser neuron for ultrafast cognitive computing," *IEEE J. Sel. Topics Quantum Electron.*, vol. 19, no. 5, pp. 1–12, Sep. 2013.

[83] T. V. Vaerenbergh *et al.*, "Cascadable excitability in microrings," *Opt. Express*, vol. 20, no. 18, pp. 20292–20308, Aug. 2012. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-20-18- 20292

[84] K. Nozaki *et al.*, "Ultracompact O-E-O converter based on fF-capacitance nanophotonic integration," in *Proc. Conf. Lasers Electro-Optics*, Opt. Soc. Amer., 2018, p. SF3A.3. [Online]. Available: http://www.osapublishing.org/abstract.cfm?URI=CLEO_SI-2018-SF3A.3

[85] S. Buckley *et al.*, "All-silicon light-emitting diodes waveguide-integrated with superconducting single-photon detectors," *Appl. Phys. Lett.*, vol. 111, no. 14, 2017, Art. no. 141101. [Online]. Available: https://doi.org/10.1063/1.4994692

[86] A. Jayakumar and J. Alspector, "A cascadable neural network chip set with on-chip learning using noise and gain annealing," in *Proc. IEEE Custom Integr. Circuits Conf.*, IEEE, 1992, pp. 19.5.1–19.5.4. [Online]. Available: http://ieeexplore.ieee.org/document/5727383/

[87] P. W. Hollis, J. S. Harper, and J. J. Paulos, "The effects of precision constraints in a backpropagation learning network," *Neural Comput.*, vol. 2, no. 3, pp. 363–373, 1990. [Online]. Available: https://doi.org/10.1162/neco.1990.2.3.363

[88] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or −1," *CoRR*, vol. abs/1602.02830, 2016. [Online]. Available: http://arxiv.org/abs/1602.02830

[89] J. R. McDonnell, "Training neural networks with weight constraints," in *Proc. 1st Annu. Conf. Evol. Program.*, 1992. [Online]. Available: http://www.dtic.mil/dtic/tr/fulltext/u2/a264665.pdf

[90] A. Aimar *et al.*, "Nullhop: A flexible convolutional neural network accelerator based on sparse representations of feature maps," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 3, pp. 644–656, Mar. 2019.

[91] M. Georgas, J. Leu, B. Moss, C. Sun, and V. Stojanović, "Addressing link-level design tradeoffs for integrated photonic interconnects," in *Proc. IEEE Custom Int. Circuits Conf. (CICC)*, 2011, pp. 1–8.

[92] E. Mounier and J.-L. Malinge, "Silicon photonics reaches tipping point, with transceivers shipping in volume," Yole, Tech. Rep., vol. 13, no. 3, Apr./May 2018.

[93] N. Izhaky *et al.*, "Development of CMOS-compatible integrated silicon photonics devices," *IEEE J. Sel. Topics Quantum Electron.*, vol. 12, no. 6, pp. 1688–1698, Nov. 2006.

[94] D. Patterson, "The future of packaging with silicon photonics," *Chip Scale Rev.*, pp. 14–15, 18, 20–21, 23–25, Feb. 2017.

[95] A. H. Atabaki *et al.*, "Integrating photonics with silicon nanoelectronics for the next generation of systems on a chip," *Nature*, vol. 556, no. 7701, pp. 349–354, 2018. [Online]. Available: https://doi.org/10.1038/s41586-018-0028-z

[96] V. Stojanović *et al.*, "Monolithic silicon-photonic platforms in state-of-the-art CMOS SOI processes," *Opt. Express*, vol. 26, no. 10, pp. 13106–13121, May 2018. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-26-10-13106

[97] P. Dong *et al.*, "Novel integration technique for silicon/III–V hybrid laser," *Opt. Express*, vol. 22, no. 22, pp. 26854–26861, Nov. 2014. [Online]. Available: http://www.opticsexpress.org/abstract.cfm?URI=oe-22-22-26854

[98] A. Y. Liu and J. Bowers, "Photonic integration with epitaxial III–V on silicon," *IEEE J. Sel. Topics Quantum Electron.*, vol. 24, no. 6, pp. 1–12, Nov. 2018.

[99] D. Thomson *et al.*, "Roadmap on silicon photonics," *J. Opt.*, vol. 18, no. 7, 2016, Art. no. 073003. [Online]. Available: http://stacks.iop.org/2040-8986/18/i=7/a=073003

[100] D. Liang and J. E. Bowers, "Recent progress in lasers on silicon," *Nat. Photon.*, vol. 4, pp. 511–517, Jul. 2010. [Online]. Available: https://doi.org/10.1038/nphoton.2010.167

[101] G. Duan *et al.*, "Hybrid III–V on silicon lasers for photonic integrated circuits on silicon," *IEEE J. Sel. Topics Quantum Electron.*, vol. 20, no. 4, pp. 158–170, Jul. 2014.

[102] A. Y. Liu, S. Srinivasan, J. Norman, A. C. Gossard, and J. E. Bowers, "Quantum dot lasers for silicon photonics [invited]," *Photon. Res.*, vol. 3, no. 5, pp. B1–B9, Oct. 2015. [Online]. Available: http://www.osapublishing.org/prj/abstract.cfm?URI=prj-3-5-B1

[103] W. Bogaerts and L. Chrostowski, "Silicon photonics circuit design: Methods, tools and challenges," *Laser Photon. Rev.*, vol. 12, no. 4, Apr. 2018, Art. no. 1700237. [Online]. Available: http://doi.wiley.com/10.1002/lpor.201700237

[104] B. J. Shastri, M. A. Nahmias, A. N. Tait, B. Wu, and P. R. Prucnal, "Simpel: Circuit model for photonic spike processing laser neurons," *Opt. Express*, vol. 23, no. 6, pp. 8029–8044, Mar. 2015. [Online]. Available: http://dx.doi.org/10.1364/OE.23.008029

[105] G. Lamant *et al.*, "Schematic driven simulation and layout of complex photonic IC's," in *Proc. IEEE Int. Conf. Group IV Photon. GFP*, Nov. 2016, pp. 92–93.

[106] T. Bekolay *et al.*, "Nengo: A python tool for building large-scale functional brain models," *Frontiers Neuroinformat.*, vol. 7, p. 48, 2014. [Online]. Available: https://www.frontiersin.org/article/10.3389/fninf.2013.00048

[107] T. C. Stewart, "A technical overview of the neural engineering framework," Centre for Theoretical Neuroscience, Tech. Rep., 2012. [Online]. Available: http://compneuro.uwaterloo.ca/publications/stewart2012d.html

[108] A. Cichocki and R. Unbehauen, *Neural Networks for Optimization and Signal Processing*. New York, NY, USA: Wiley, 1993.

[109] M. Briere, L. Carrel, T. Michalke, F. Mieyeville, I. O'Connor, and F. Gaffiot, "Design and behavioral modeling tools for optical network-on-chip," in *Proc. Des. Autom. Test Europe Conf. Exhib.*, vol. 1, no. c, pp. 738–739, 2004. [Online]. Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1268955

[110] F. Pêcheux, C. Lallement, and A. Vachoux, "VHDL-AMS and Verilog-AMS as alternative hardware description languages for efficient modeling of multidiscipline systems," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 24, no. 2, pp. 204–224, Feb. 2005.

Authors' biographies not available at the time of publication.