This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

Training Deep Photonic Convolutional Neural Networks With Sinusoidal Activations

Nikolaos Passalis, George Mourgias-Alexandris, Apostolos Tsakyridis, Nikos Pleros, and Anastasios Tefas

***Abstract*—Deep learning (DL) has achieved state-of-the-art performance in many challenging problems. However, DL requires powerful hardware for both training and deployment, increasing the cost and energy requirements and rendering large-scale applications especially difficult. Recognizing these difficulties, several neuromorphic hardware solutions have been proposed, including photonic hardware that can process information close to the speed of light and can benefit from the enormous bandwidth available on photonic systems. However, the effect of using these photonic-based neuromorphic architectures, which impose additional constraints that are not usually considered when training DL models, is not yet fully understood and studied. The main contribution of this paper is an extensive study on the feasibility of training deep neural networks that can be deployed on photonic hardware that employ sinusoidal activation elements, along with the development of methods that allow for successfully training these networks, while taking into account the physical limitations of the employed hardware. Different DL architectures and four datasets of varying complexity were used for extensively evaluating the proposed method.**

***Index Terms*—Photonic neural networks, sinusoidal activations, deep learning.**

I. Introduction

DEEP Learning (DL) has revolutionized machine learning providing state-of-the-art solutions to several difficult problems [1], ranging from computer vision to natural language processing. DL models are composed of several "processing" layers that extract increasingly complex features. It has been demonstrated that increasing the *depth* of a model usually positively impacts its accuracy, given that the model is appropriately designed and regularized, leading to an enormous increase in the complexity of DL models. For example, while early DL models were restricted to 10-20 layers, recently proposed state-of-the-art models have more than 1,000 layers [2]. However, training DL models require powerful and specialized hardware. In fact, much of the progress in deep learning has been fueled by the development of specialized and powerful hardware for this task [1].

Manuscript received December 26, 2018; revised April 19, 2019; accepted June 3, 2019. *(Corresponding author: Nikolaos Passalis.)*

N. Passalis is with the Artificial Intelligence and Information Analysis Laboratory, Aristotle University of Thessaloniki, 541 24 Thessaloniki, Greece. He is now with the Faculty of Information Technology and Communication Sciences, Tampere University, 33800 Tampere, Finland (e-mail: passalis@csd.auth.gr).

A. Tefas is with the Artificial Intelligence and Information Analysis Laboratory, Aristotle University of Thessaloniki, 541 24 Thessaloniki, Greece (e-mail: tefas@csd.auth.gr).

G. Mourgias-Alexandris, A. Tsakyridis, and N. Pleros are with the Photonic Systems and Networks Research Group, Aristotle University of Thessaloniki, 541 24 Thessaloniki, Greece (e-mail: mourgias@csd.auth.gr; atsakyrid@csd.auth.gr; npleros@csd.auth.gr).

Digital Object Identifier 10.1109/TETCI.2019.2923001

Apart from training DL models, the inference speed and computational/energy requirements are even more important, since the successful large-scale deployment of DL models critically relies on them.

Several neuromorphic hardware solutions have been proposed to overcome the aforementioned limitations [3]–[9]. Neuromorphic hardware provides several advantages by lowering the energy requirements of deploying neural networks and improving the inference speed. Even though these solutions are not yet widely used, they hold the credentials for replacing the currently predominantly used generic hardware accelerators, such as GPUs and Tensor Processing Units (TPUs) [10]. Perhaps among the most promising solutions for providing hardware implementations of deep neural networks is using *photonics* [11]. This allows for overcoming many of the limitations of the currently used generic hardware accelerators by replacing the electronic-based information processing components with optical ones. This allows for having photonic-based neuromorphic architectures that can either be used to implement spiking neural networks or to accelerate the inference for traditional artificial neural networks. In this paper we study the use of photonic hardware for the latter, since deep artificial neural networks currently exhibit state-of-the-art performance.

In photonic-based neuromorphic architectures optical signals are used to represent the input to a neural network. This optical signal is then manipulated and processed using the appropriate components, providing in this way the functionality of neurons. This allows the signal to propagate near to the speed of light and the neurons to operate at extremely high frequencies, while the enormous bandwidth of optical components provide a great potential for parallelizing various tasks. These provides significant advantages over the currently used solutions, often outperforming them by several order of magnitude [11]. It is worth noting that very efficient and fast processing engines have been already demonstrated in the context of reservoir computing [12], along with optical deep neural networks [13]. Also, quite recently, Photonic Integrated Circuits (PICs) have been employed for implementing integrated weighting banks, allowing for moving towards fully integrated photonic neural network that employ sinusoidal activation elements [14].

Even though neuromorphic hardware can provide significant performance benefits, i.e., improve the speed, power, and energy consumption, it always comes with additional limitations and constraints over artificial neural networks that will be simulated, i.e., deployed on general purpose hardware, instead of being actually implemented on hardware, e.g., [15], [16]. For

2471-285X © 2019 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on July 14,2020 at 12:07:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

these neuromorphic architectures there is no generic training approach, since each architecture exhibits its own particularities [15]–[20]. Therefore, the training algorithms must be carefully adapted to the needs of each application, ensuring that a) the transfer functions of the various components is appropriately modeled, b) the trained network behaves correctly and stays within the imposed hardware limits (weights, activations, etc.) and c) the various sources of noise are taken into account during the training process.

Similar constraints also exist for photonic-based neuromorphic architectures. For example, the activation functions that are usually used in DL, e.g., *ReLU* [21], cannot be directly implemented in photonic-based neuromorphic hardware [14], [22], [23]. To provide the functionality of non-linear activation functions several combinations of photonic and electronic components have been proposed in the literature. For example, in [14], a Mach-Zehnder Modulator (MZM) [24] is used in order to appropriately modulate an optical signal based on the output of a neuron, leading to the following transfer function:

$$P_{out} = P_{in} \sin^2 \left(\frac{\pi}{2} \frac{V_{RF}}{V_{\pi}} \right), \tag{1}$$

where P_{out} is the output signal, P_{in} is the Continuous Wave (CW) signal to be modulated, V_{π} is the voltage needed for achieving a π phase shift and V_{RF} is the input to the activation function. In this paper, we also consider this activation configuration. It is also worth noting that this method requires using balanced photodetectors in order to convert the optical signal (output of a neuron) to an electrical signal. This signal is then employed to modulate a reference optical signal, while a diode is used at the output of balanced layout for ignoring negative voltage signals. This configuration can operate at high frequencies, despite using electronic components, due to the use of Germanium-based photodetectors [25] and the significant progress witnessed in high speed RF modulator driver [26] and Transimpedance Amplifier (TIA) IC circuits [27].

Apart from the previous problem, additional training difficulties arise when non-linear activation functions, such as the sigmoid function, are used, since the training process can slow down (or even completely halt), due to several phenomena, such as vanishing of the input and/or back-propagated signals [21]. Furthermore, note the periodic and highly non-linear behavior of the activation function described in (1), which significantly differs from the traditional activation functions that are usually employed in DL. Neurons that are implemented in photonic hardware also come with additional constraints. The response of neurons must be bounded, since physical systems work within a specific power range. Also, the effect of various noise sources must be taken into account and the transfer function of the various components of the system must be accurately modeled to ensure that the implemented neural network will behave as expected. Note that the aforementioned limitations arise only with neuromorphic implementations, since software (simulation-based) implementations ensure that the network will always work in a noise-free environment, while the precision of floating point numbers is usually more than enough for representing the weights/activations of neural networks, ensuring that the network will always behave as it was designed.

The main contributions of this work are the following: a) An appropriate initialization scheme is derived based on the transfer function of the photonic configuration used to implement the activation function in the employed neuromorphic architecture. This initialization scheme ensures that the input signals will not diminish and the training process will proceed smoothly. b) A regularization method, which ensures that the activations of the network will remain within the working range of the used hardware, is also proposed. The same approach can be additionally used to control the range of the weights of the network, if such constraints are also imposed by the hardware. c) The effect of different types of noise on the training and inference process is studied and it is demonstrated that it is possible to train robust photonic neural networks that can withstand significant amounts of noise, if the characteristics of the noise are known beforehand and taken into account during the training process. d) Finally, the effectiveness of the proposed approach is experimentally verified using a wide range of different configurations, networks and datasets.

The rest of this paper is structured as follows. First, the related work is briefly discussed in Section II, while an introduction to neural networks and the employed photonic-based neuromorphic architecture is provided in Section III. Then, the limitations imposed by photonic hardware are introduced in Section IV and the proposed methods for training deep neural networks on such hardware are derived. Next, the effect of various parameters, design choices and methods on training neural networks that are to be implemented on photonic hardware are extensively evaluated in Section V, while conclusions are drawn in Section VI.

II. Related Work

This work is related to training neural networks for neuromorphic hardware [15]–[19], [28]–[31]. Note that the term neuromorphic is not consistently used in the literature. In this work, the term neuromorphic is used to refer to dedicated hardware architectures that directly implement the functionality of neurons, usually by employing analog circuits and/or devices.

There is no generic approach for training networks for neuromorphic hardware, since each neuromorphic architecture exhibits its own particularities that need to be considered when designing the corresponding networks. For example, in [17], the behavior of metal-oxide memristors, e.g., their large variability due to the manufacturing processes that are currently used, is considered for designing networks that can be trained *in situ*, i.e., using the actual neuromorphic hardware for the training process, while a similar approach for *ex-situ* training, i.e., training using external hardware and then programming the neuromorphic hardware, which is also the approach followed in this paper, is presented in [15]. The use of quantum computers and high-performance low-power memristive hardware is also examined in [19]. A way of mapping convolutional neural networks (CNN) to neuromorphic chips, such as the TrueNorth, is presented in [18], while the use of hardware that only supports spiking neural network architectures is thoroughly considered

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on July 14,2020 at 12:07:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

in [16]. To the best of our knowledge this is the first in-depth study on the feasibility of training deep neural networks for photonic hardware that employ sinusoidal activations [14], while taking into account the constraints imposed by the hardware.

The weight regularization approaches proposed in this paper have been also examined previously in various occasions [32]–[34], but in a different context. Most of the existing regularization methods, such as dropout [35], aim to reduce over-fitting phenomena that occur during the training process, and, as a result, achieve better generalization. On the other hand, the methods proposed and evaluated using simulations in this paper have a completely different purpose: they aim to impose the constraints dictated by the neuromorphic hardware used for deploying the network. Employing noise during the training process is also related to various regularization and denoising methods, i.e., denoising autoencoders [36]. However, again the motivation in this work is different: instead of trying to avoid over-fitting or make the network more robust to noisy input signals, the network must be trained to withstand noise in all of each layers. This noise does not originate from the input signal, but it is an intrinsic characteristic of the hardware that is used for its implementation.

III. Neural Networks and Photonic Hardware

A neuron, which is the building block of a neural network, receives its input, denoted by a vector $\mathbf{x}^{(l)} \in \mathbb{R}^{n_l}$, where n_l denotes the input dimensionality. Note that the input of the neural network may be directly given to the neuron, i.e., the neuron receives the raw data, or the processed output of another layer of neurons can be used as input. The layer from which the neuron receives the data is denoted by "(l)". Therefore, the input data are denoted by $\mathbf{x}^{(0)}$, while for each succeeding layer the input is denoted by $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \ldots$. Then, the weights $\mathbf{w}_i^{(l)} \in \mathbb{R}^{n_{l-1}}$ are multiplied with the input and summed together. The activation vector $\mathbf{u}^l \in \mathbb{R}^{n_l}$ for the l-th layer is compiled as:

$$\mathbf{u}^{(l)} = \mathbf{W}^{(l)}\mathbf{x}^{(l-1)} + \mathbf{b}^{(l)} \in \mathbb{R}^{n_l}, \tag{2}$$

where n_l denotes the number of neurons in this layer, $\mathbf{W}^{(l)} = [\mathbf{w}_1^{(l)}, \mathbf{w}_2^{(l)}, \ldots, \mathbf{w}_{n_l}^{(l)}] \in \mathbb{R}^{n_l \times n_{l-1}}$ denotes the matrix with the weights for the l-th layer and $\mathbf{b} \in \mathbb{R}^{n_l}$ denotes the bias vectors for the corresponding layer. Then, the output for each neuron is calculated by passing the activation through a non-linearity $f(\cdot)$:

$$\mathbf{x}^{(l)} = f(\mathbf{u}^{(l)}) \in \mathbb{R}^{n_l}. \tag{3}$$

Note that the term "output" and "activation" are not used consistently in the literature. In this paper, the term *output* refers to the final output of a neuron *after* applying the activation function $f(\cdot)$, while the term *activation* refers to the quantity $\mathbf{u}^{(l)}$, i.e., the neuron's output *before* applying the activation function. Multiple layers of neurons can be stacked together to form deep neural networks. Also, note that other types of layers apply the weights in different ways, e.g., convolutional layers [2]. Neural networks are usually trained using the back-propagation algorithm [33],

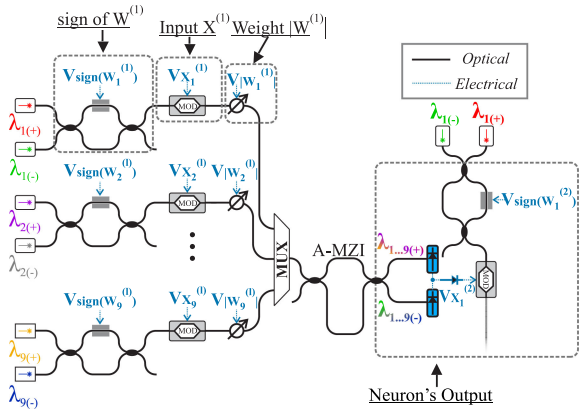


Fig. 1. Photonic layout implementing a single neuron/convolutional filter.

which optimizes the parameters of the network toward minimizing a loss function $\mathcal{L}$, which measures how well the network fits the task at hand.

Figure 1 depicts the photonic layout for a single neuron of the employed architecture that can be used either for implementing dense layers or convolutional filters. For each of the 9 separate inputs, 2 different lasers emitting at different wavelengths $\lambda_{i(+)}$ and $\lambda_{i(-)}$ are employed to represent positive and negative numbers and are injected as input CW signals into a 2×2 Mach-Zehnder Interferometer (MZI) switch. The role of this 2×2 MZI switch is to correspond the sign of weight with a laser at specific wavelength and can either rely on an electro-optic or thermo-optic switching effect: whenever a negative weight value is employed, a zero voltage-level is applied at the MZI forcing the MZI to operate in its "Cross"-state, so that the $\lambda_{i(-)}$ and the $\lambda_{i(+)}$ CW signals emerge at the upper and the lower output ports of the switch, respectively. In case a positive weight value has to be used, then a V_π voltage gets applied at the MZI switch, discarding the $\lambda_{i(-)}$. The $\lambda_{i(+)}$ signal will exit the switch through its upper output port and will get forwarded to the electro-optic modulation stage of every neuron. After exiting the MZI switch, the selected laser beam is injected into an electro-optic Mach-Zehnder Modulator (MZM), driven by the proper $V_{X_i^{(1)}}$ electrical input signal, imprinting in this way the input ($X_i^{(1)}$) onto an optical signal with proportional power $P_{X_i^{(1)}}$. This signal enters then a Variable Optical Attenuator (VOA) that can be realized again by means of an electro-optic or thermo-optic MZI switch driven by an electrical signal, having the role of realizing the $|W_i^{(1)}|$ by attenuating properly the signal $P_{X_i^{(1)}}$, as a result the signal $W_i^{(1)} \cdot P_{X_i^{(1)}}$. At this point, it is worth to mention that the MZIs which are responsible for realizing the $P_{X_i^{(1)}}$ and the $|W_i^{(1)}|$ have been operated at the linear region of their sinusoidal transfer function in order to translate precisely the numerical value of $X_i^{(1)}$ and $|W_i^{(1)}|$ to the corresponding optical power. Following the weighting stage, all 9 signals are combined into a single waveguide via an optical multiplexer (MUX), while the following Asymmetric MZI (A-MZI) is employed for separating the positive- and negative-weighted signals, forwarding to

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on July 14,2020 at 12:07:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

upper port and lower port of A-MZI all the wavelengths that carry positive- and negative-weighted signals, respectively. Afterwards, at the input of upper photodiode occurs the summation of positive-weighted signals and at the lower photodiode the summation of negative weighted signals. The expression below describes how the optical power of positive- and negative-weighted signals is summed at the input of the corresponding photodiode (pd):

$$P_{pd_{(+|-)}-input} = \sum_{i=1}^{n} W_i^{(l)} \cdot P_{X_i^{(l)}(+|-)} \tag{4}$$

The balanced topology of 2 photodiodes allows for the subtraction of positive- and negative-weighted signals, while the diode at the output of this circuit discards the negative voltages $V_{X_i^{(2)}}$. Using this signal to drive a MZM where 2 CWs at $\lambda_{i(+)}$ and $\lambda_{i(-)}$ are injected to the input, at the output of MZM will occur the output of the photonic activation unit. Now the MZM is operated with $V_{X_i^{(2)}}$ voltages that cause $\frac{\pi}{2}$ phase-shift, utilizing the non-linear region of its transfer function. The transfer function of the photonic activation unit depicted in Fig. 1 can be now described, by employing the transfer function of MZM described in (1), as:

$$P_{activ-out} = \begin{cases} 0, & \text{if } V_{X_i^{(l)}} \leq 0 \\ P_{\lambda i(+|-)} \cdot \sin^2\left(\frac{\pi}{2}\frac{V_{X_i^{(l)}}}{V_\pi}\right), & \text{if } V_{X_i^{(l)}} > 0 \end{cases}, \tag{5}$$

This topology allows for every neuron to produce an output signal that is used as input for the subsequent optical neuron. It is worth noting this architecture holds the credentials of getting integrated onto a single photonic chip by utilizing components and circuits already offered by CMOS-compatible silicon Photonic Integrated Circuit (PIC) technology.

IV. Training Deep Photonic Neural Networks

The proposed methods for training deep neural networks for neuromorphic photonic hardware are analytically derived in this Section, after considering the limitation imposed by the employed photonic hardware. The simplest case will be considered for the scope of the analysis, i.e., a fully connected neuron. This is without loss of generality, since it is trivial to extend the obtained results and methods for other types of layers as well, as demonstrated in Section V.

A. Neural Networks and Photonic Hardware

Photonic hardware can be used to implement the multiplication between the input and the weights, along with the summation that leads to the activation of the neurons $\mathbf{u}^{(l)}$, as discussed in Section III. However, note that in a physical system there are restrictions regarding the scale of the activations, since there are power/energy constraints that cannot be exceeded. This behavior is simulated in the conducted experiments by using a hard cut-off threshold on the activations:

$$\mathbf{u}^{(l)} = \min(\max(\mathbf{u}^{(l)}, -u_{\max}), u_{\max}), \tag{6}$$

where u_{max} is the maximum absolute activation value supported by the hardware and the $\max(\cdot)$ and $\min(\cdot)$ functions are applied element-wise. Similarly, one can also constraint the weights of the network as $\mathbf{W}^{(l)} = \min(\max(\mathbf{W}^{(l)}, -w_{max}), w_{max})$, where w_{max} is the absolute maximum value for the weights that is supported by the hardware.

Then, the activations must be fed to a non-linear device that acts as the activation function of the network. As already discussed in Section III, the layout proposed in [14] along with a diode is used in this work, i.e., a modulator controlled by the output current of a photodetector-based circuit is employed to provide the functionality of a non-linear activation function. The transfer function of the employed photonic activation can be simplified as:

$$f(u) = \begin{cases} 0, & \text{if } u \leq 0 \\ \sin^2(\frac{\pi}{2}u), & \text{if } u > 0 \end{cases}, \tag{7}$$

after taking into account the physical limitations of the involved components and that the activation u can be easily scaled accordingly before implementing the network (according to the voltage V_π required to have a phase shift of π that corresponds to optimal modulation in optical interferometric-based modulators). As also discussed in the relevant literature [31], [37], employing an activation scheme adapted to the actual transfer function of the device at hand is of crucial importance to avoid vanishing gradient phenomena and ensure the smooth training of deep neural networks.

Furthermore, different types of noise exist in a real hardware implementation. The proposed neuromorphic photonic layout is based on a photonic component portfolio, including laser sources, waveguides, modulators, Mach-Zehnder interferometers, multiplexers and photodiodes, which has been extensively studied in optical communication systems by employing the Additive White Gaussian Noise (AWGN) model [38]. Following these findings, we also employ a Gaussian noise source to model the noise that exists in hardware implementations. Therefore, during the testing, an AWGN source is used to stochastically corrupt the activations as $\tilde{\mathbf{u}}^{(l)} = \mathbf{u}^{(l)} + \mathcal{N}(0, \sigma^2)$, where $\mathcal{N}(\mu, \sigma^2)$ is a normal distribution with mean μ and standard deviation σ.

B. Initialization Scheme for Sinusoidal Activation Functions

In this Section, the characteristics of the initialization process are derived by requiring the variance of the input signals to kept constant thought the various layers of the networks. This will allow the information to arrive to the employed loss function, as well as ensures the smooth flow of gradients [31], [37].

The notation $w^{(l)}$ is used to denote the random variable used for initializing the weights of the l-th layer, while the notation $u^{(l)}$ and $x^{(l)}$ is employed to refer to the random variables that correspond to the outputs $\mathbf{u}_i^{(l)}$ and $\mathbf{x}_i^{(l)}$ of the l-th layer. The elements of $\mathbf{W}^{(l)}$ share the same distribution, are mutually independent and must have zero mean, i.e., $E[w^{(l)}] = 0 \ \forall \ l$. Furthermore, the variables $w^{(l)}$ and $x^{(l)}$ are assumed to be independent, while the elements in $\mathbf{x}^{(l)}$ to be mutually independent and drawn from $x^{(l)}$. A first-order Taylor expansion around u_0 is used to approximate the behavior of the employed function in order to simplify

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on July 14,2020 at 12:07:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

the analysis. Therefore, the following function will be used for the rest of the conducted analysis:

$$\hat{f}(u) = \begin{cases} c_1(u - u_0) + c_0, & \text{if } u > 0 \\ 0, & \text{if } u \leq 0 \end{cases}. \tag{8}$$

where $c_1 = f'(u_0) = \frac{1}{2}\pi \sin(\pi u_0)$ and $c_0 = f(u_0) = \sin^2(\frac{1}{2}u_0)$. The feed-forward case will be examined. The variance of the output of the l-th layer is calculated as:

$$\begin{aligned} Var[u^{(l)}] &= Var[\mathbf{u}_k^{(l)}] \\ &= Var\left[\sum_{i=1}^{n_{l-1}} \mathbf{W}_{ki}^{(l)} \mathbf{x}_i^{(l-1)}\right] \\ &= \sum_{i=1}^{n_{l-1}} Var\left[w^{(l)}\right] E\left[(\mathbf{x}_i^{(l-1)})^2\right]. \end{aligned} \tag{9}$$

By initializing the bias terms to 0 and assuming that $w^{(l)}$ is symmetric around zero, then it can be easily derived that $E[u^{(l)}] = 0$, as well as that $u^{(l)}$ is also symmetric around zero. Also, note that the activation function maps half of its values to 0. Then, $E[(\mathbf{x}_i^{(l-1)})^2]$ can be calculated as:

$$\begin{aligned} &E[(\mathbf{x}_i^{(l-1)})^2] \\ &= \frac{1}{2}c_1^2 E[(\mathbf{u}_i^{(l-1)})^2] = \frac{1}{2}c_1^2 Var[u^{(l-1)}], \end{aligned} \tag{10}$$

by employing the approximation given in (8) for the other half. Therefore, the variance for the activations of the l-th layer is calculated as:

$$\begin{aligned} Var[u^{(l)}] &= \frac{1}{2}c_1^2 \sum_{i=1}^{n_{l-1}} Var[w^{(l)}] Var[u^{(l-1)}] \\ &= \frac{c_1^2}{2} n_{l-1} Var[w^{(l)}] Var[u^{(l-1)}] \\ &= Var[x^{(0)}] \prod_{i=1}^{l-1} \frac{c_1^2}{2} n_{i-1} Var[w^{(i)}], \end{aligned} \tag{11}$$

by plugging (10) into (9). Recall that $\mathbf{x}^{(0)}$ denotes the neural network's input and n_0 is the input dimensionality. Therefore, the product that appears in (11) must be equal to 1. This ensures that the variance will be kept constant across the layers. Therefore, the variance of the distribution used for initializing the weights of the l-th layer should be:

$$Var[w^{(i)}] = \frac{2}{c_1^2 n_{i-1}}, \tag{12}$$

to ensure that the variance will be kept constant.

If a normal distribution is used for initializing the i-th layer, then its standard deviation should be set to $\sigma = \sqrt{Var[w^{(i)}]} = \frac{1}{\sigma_1}\sqrt{\frac{2}{n_{i-1}}}$, by employing the target variance $Var[w^{(i)}]$ calculated in (12). If a uniform distribution is used, then the weights should be sampled from the interval $[-\alpha, \alpha]$, where $\alpha = \sqrt{3Var[w^{(i)}]} = \frac{1}{\sigma_1}\sqrt{\frac{6}{n_{i-1}}}$, since $Var[x] = \frac{1}{3}\alpha^2$ for x drawn uniformly from $[-\alpha, \alpha]$.

The variance of the back-propagated signals can be similarly derived. However, as proposed in [39], either of them can be used interchangeably in most of the cases. Furthermore, note it is straightforward to adapt the proposed initialization scheme to other types of layers, e.g., convolutional layers, by correctly setting n_{i-1} to the *fan-in* of each neuron [39].

C. Hardware-Guided Neural Network Regularization

Apart from the initialization, there are also several other constraints that arise from the use of photonic hardware, as discussed in Section IV-A. First, the activations of the neurons must be restricted into a pre-determined range. To achieve this, an activation regularizer is employed in this work:

$$\mathcal{L}_a = \sum_{l=1}^{L} ||max(0, |\mathbf{u}^{(l)}| - u_{max})||_2^2, \tag{13}$$

where $||\mathbf{x}||_2$ is the l^2 norm of a vector $\mathbf{x}$ and u_{max} is the maximum absolute value that a neuron can output. It is easy to verify that this regularizer penalizes the network whenever the activation of a neuron is out of the working range of the hardware. In this way, the weights of the networks are modified to avoid producing large activation values. Alternatively, l^1 regularization, which also encourages more sparse solutions, can be also employed. The interested reader is referred to [32] for a more in depth comparison between these two approaches. The weights of the network can be regularized similarly using the same approach:

$$\mathcal{L}_w = \sum_{l=1}^{L} ||\max(0, |\mathbf{W}^{(l)}| - w_{\max})||_F^2, \tag{14}$$

where the maximum function is applied element wise and $||\mathbf{W}||_F$ denotes the Frobenius norm of matrix W. Note the close relationship of (14) to the well-known l^2 regularization [33], which is used to regularize various machine learning models and avoid over-fitting. Therefore, the following loss function is employed for training the networks:

$$\mathcal{L} = \mathcal{L}_{class} + \alpha_a \mathcal{L}_a + \alpha_w \mathcal{L}_w, \tag{15}$$

where α_a is the weight of the activation regularizer, α_w is the weight of the weight regularizer and $\mathcal{L}_{class}$ is the classification loss used for the training, typically the cross-entropy loss [33].

Finally, to make the trained networks more robust to noise, the same noise that exists in the hardware device (simulation in the conducted experiments) was also added to the activation. For example, for AWGN the activations are calculated as:

$$\tilde{\mathbf{u}}^{(l)} = \mathbf{u}^{(l)} + \mathcal{N}(0, \sigma^2), \tag{16}$$

where σ is the standard deviation of the noise used for the training. Similarly, different types of noise can be considered, if the characteristics of the noise are known beforehand. Note that the proposed approaches are orthogonal to other regularization techniques, such as dropout [35], which aim to reduce over-fitting instead of ensuring that the trained network will stay within the specifications of the employed photonic hardware. Of course,

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on July 14,2020 at 12:07:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

the proposed method can be readily combined with these regularization methods and further improve the performance of the trained networks, as also demonstrated in the conducted experiments.

V. Experimental Evaluation

Experimental Setup: Four different datasets are used in this paper: three image dataset, the MNIST dataset [40], the Fashion MNIST dataset [41] and the CIFAR10 dataset [42], as well as a high frequency limit order book dataset that contains more than 4 million limit order event, the FI-2010 dataset [43], [44]. The MNIST and the Fashion MNIST datasets were normalized to the range 0...1, while the CIFAR-10 and the FI-2010 dataset were normalized using z-score normalization, i.e., to have zero mean and unit variance.

Three different deep learning convolutional architectures were used: a) a 4-layer 2D convolutional network and b) a 6-layer 2D convolutional network and c) a 3 layer 1D convolutional neural network. The first network (abbreviated as "CNN-1") is composed of two convolutional layers (3×3 kernels) with 32 and 64 filters, followed by two fully connected layers with 512 and 10 neurons respectively. Before and after the first fully connected layer dropout ($p = 0.5$) is used. Average pooling with kernel size 2×2 is used after each convolutional layer (note that currently there is no method proposed in the literature for implementing max pooling in photonic networks). The second network, abbreviated as "CNN-2", follows a similar architecture, i.e., four convolutional layers with 32, 64, 128 and 256 filters (average pooling is used after the second and the fourth layers) are used, while the fully connected layer is composed of 1024 neurons. Again, dropout is used before and after the first fully connected layer. The CNN-3 network is composed of a 1D convolutional layer with 64 filters, followed by two fully connected layers with 512 and 3 neurons respectively. The same architectures were used for all the experiments regardless the employed activation function and/or initialization scheme to ensure a fair comparison between the evaluated methods. Furthermore, all the models were implemented using the PyTorch library (more details regarding each experiment, e.g., optimizer, learning rate, etc., are given in the corresponding section), while they were trained using a mid-range GPU accelerator.

The cross entropy loss, together with the softmax activation function, is used for training the networks. Note that if we are interested solely in predicting the most probable class, then the softmax function can be removed during the deployment, since the softmax function does not alter the relative ordering between the predictions. Unless otherwise stated, the networks were initialized using a uniform distribution. Finally, u_0 was set to $\frac{1}{4}$ for all the conducted experiments, since this value consistently yielded the best results.

Initialization Scheme Evaluation: First, the proposed initialization scheme (called "Photonic Sinusoidal Initialization" - PSI - in the rest of this paper) is compared to the well-known Xavier initialization (as proposed in [37]) in Fig. 2. Stochastic gradient descent (the learning rate was set to $\eta = 0.1$) with momentum of 0.9 was used for these experiments. The

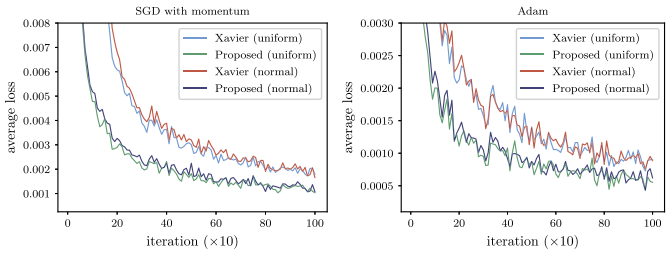


Fig. 2. Learning curves for two different initialization schemes (Xavier [37] and proposed), two different random distributions (uniform and normal) and two different optimizers (SGD with momentum and Adam).

TABLE I
Evaluation on the MNIST Dataset

Model	Optimizer	Train error (%)	Test error (%)
ReLU	SGD	0.78	1.00
Photonic + Xavier	SGD	0.81	1.04
Photonic + PSI	SGD	**0.68**	**0.99**
ReLU	RMSProp	0.54	0.89
Photonic + Xavier	RMSProp	0.72	1.02
Photonic + PSI	RMSProp	**0.39**	**0.98**
ReLU	Adam	0.49	0.79
Photonic + Xavier	Adam	0.59	1.00
Photonic + PSI	Adam	**0.37**	**0.78**

The optimization ran for 10 epochs (learning rate set to 0.01 for SGD and 0.001 for RMSProp and Adam) with batch size of 128.

baseline "ReLU" model refers to using the CNN-1 network with the ReLU activation and the appropriate initialization scheme [39], while the "Photonic + Xavier" and "Photonic + PSI" refers to using the CNN-1 network with the Xavier and the proposed initialization approach respectively. PSI leads to significantly faster convergence than the Xavier initialization regardless the distribution used for drawing the weights and the employed optimizer (SGD or Adam [45]). Then, the improved behavior of PSI was also confirmed in the experimental results provided in Table I using the MNIST dataset and the CNN-1 architecture. Three different optimizers were employed for the evaluation: SGD with momentum, RMSProp [46] and Adam [45]. The proposed combination of the employed photonic activation and initialization scheme improves the performance over using the Xavier initialization regardless the used optimizer. The Adam optimizer led to the best results and, as a result, is employed for the rest of the conducted experiments. Also, note that the photonic activation also leads to slightly improved performance over the baseline network with the ReLU function. This can possibly attributed to the non-linear behavior of the photonic function for positive values compared to the linear behavior of the ReLU function (for the same range), which can be especially beneficial for smaller networks. However, at the same time, this non-linear behavior can also cause vanishing gradients, depending on the used network and dataset, as discussed earlier.

Similar conclusions can be also drawn from the experiments conducted using the Fashion MNIST dataset, reported in

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on July 14,2020 at 12:07:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE II
EVALUATION ON THE FASHION MNIST DATASET

Model	Network	Train error (%)	Test error(%)
ReLU	CNN-1	10.61	10.28
Photonic + Xavier	CNN-1	12.62	12.28
Photonic + PSI	CNN-1	**11.84**	**11.71**
ReLU	CNN-2	5.62	6.56
Photonic + Xavier	CNN-2	6.83	7.44
Photonic + PSI	CNN-2	**6.12**	**7.11**
Photonic + Xavier	ResNet*	90.00	90.00
Photonic + PSI	ResNet*	**15.27**	**14.48**

The optimization ran for 20 iterations (learning rate set to 0.001) with batch size of 128, followed by 20 additional iterations (learning rate set to 0.0001). ResNet* refers to a variant of the ResNet-18 model [2] that was adapted to the task at hand after removing the Batch Normalization layers and the last residual block.

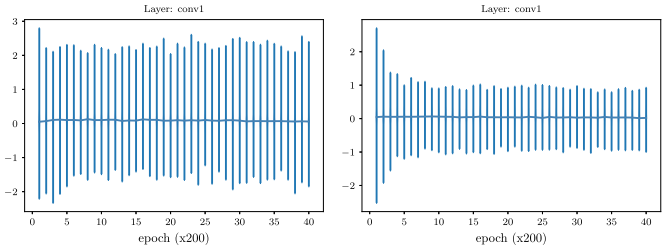
TABLE III
EFFECT OF USING ACTIVATION REGULARIZATION

Reg. Type	Reg. weight (α_a)	Train error (%)	Test error (%)
-	0	4.24	4.97
l^1	10^{-6}	2.06	2.66
l^1	10^{-5}	0.17	**0.62**
l^1	10^{-4}	0.22	0.73
l^1	10^{-3}	0.38	0.88
l^2	10^{-6}	1.97	2.69
l^2	10^{-5}	0.22	**0.67**
l^2	10^{-4}	0.31	0.79
l^2	10^{-3}	0.54	0.86

The evaluation was conducted on the MNIST dataset and the "CNN-1" network on a simulator that supports activations within the range $-1 \ldots 1$.

Table II, where the proposed initialization scheme also always improves both the training and testing error over the Xavier initilization. Note that the first model (CNN-1) was under-fitting the data, while the second one (CNN-2) was more powerful leading to better performance. To evaluate the ability of the proposed method to train deeper architectures, we also conducted an additional experiment using an architecture based on the ResNet-18 model (as described in Table II). The effect of PSI is even more obvious in this case, since it was not possible to successfully train this architecture without using the proposed initialization method, despite employing a state-of-the-art optimization method (Adam [45]). Note that even though the ResNet model is successfully trained using PSI, it achieves lower accuracy compared to the other CNN architectures. This highlights the need for models that are specifically designed for the needs of photonic neuromorphic hardware, which often exhibits different characteristics than general purpose accelerators.

Hardware-guided Regularization Evaluation: First, the effect of using the activation regularizer was examined. The activations of the network were clipped to $-1 \ldots 1$. The experimental results are reported in Table III, while the networks were trained for 20 epochs. Different choices for the activation regularizer hyper-parameter α_a and regularizer (l^1 or l^2) were evaluated. It is clearly demonstrated that using activation regularization,



(a) (b)

Fig. 3. Activations distribution for the first convolutional layer of the "CNN-1" network during the training process without using regularization (a) and using l^2 regularization (b).

TABLE IV
EFFECT OF VARYING THE ACTIVATION RANGE OF THE NEURONS

Activation range	No regularization	Activation regularization (l^2, $\alpha_a = 10^{-5}$)
1.1	5.35	**0.67**
1.0	4.97	**0.67**
0.9	7.60	**0.65**
0.8	9.79	**0.70**

The test classification error of the "CNN-1"(MNIST dataset) is reported (%).

which takes into account the limitations of the hardware used to deploy the network, improves the accuracy of network since the error drops from 4.97%, when no regularization is used, to 0.62 (l^1 regularization) and 0.67% (l^2 regularization) when the proposed regularization approach is employed. Both regularization approaches (l^1 and l^2) work equally well, given that the weight of the regularizer α_α is appropriately tuned. For all the networks, the activations were restricted to the range supported by the hardware during the evaluation. For the regularized networks, the weights were also restricted to the range supported by the network during the training. The effect of using regularization is also demonstrated in Fig. 3, where the output distribution of a convolutional layer is plotted. When no regularization is used, the activations range from -2 to 3. On the other hand, when regularization is used ($\alpha_a = 10^{-3}$) the weights are restricted into the appropriate range (-1 to 1) and they are restrained as the training progresses. The possibility of training networks that work in even more narrow activation ranges was also considered and the corresponding results are reported in Table IV, where different ranges (between 0.8 and 1.1) were used for the evaluation. Note that even though the error significantly increases when no regularization is used and the activations are severely constrained (9.97% classification error), using the proposed activation regularization scheme significantly increases the accuracy of the network.

The range of the weights can be similarly constrained using the weight regularizer presented in Section IV-C. The results using different weights for the regularizer α_w are presented in Table V. The weights were restricted into the $-0.5 \ldots 0.5$ range (larger weights were clipped during the simulation of the network) and the training process ran, as before, for 20 epochs.

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on July 14,2020 at 12:07:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE V
EFFECT OF VARYING THE WEIGHT REGULARIZER (WEIGHTS CLIPPED TO 0.5) ON THE MNIST DATASET USING THE "CNN-1" NETWORK

Reg. weight (α_w)	**Train error (%)**	**Test error (%)**
0	2.68	3.40
10^{-4}	1.75	2.23
10^{-3}	0.68	1.29
10^{-2}	0.21	0.80
10^{-1}	0.18	**0.75**
0.5	0.17	0.79

TABLE VI
EFFECT OF USING DIFFERENT TYPES OF NOISE ON THE LAYERS OF THE "CNN-1" NETWORK

Noise	σ/α	**Regular Training**	**Noisy Training**
Gaussian (add., $\mu = 0$)	0.10	1.23	**0.98** (1.28)
Gaussian (add., $\mu = 0$)	0.15	2.54	**1.23** (1.24)
Gaussian (add., $\mu = 0$)	0.20	5.35	**1.43** (1.34)
Gaussian (add., $\mu = 0$)	0.25	35.56	**1.59** (1.48)
Gaussian (add., $\mu = 0$)	0.30	43.41	**1.91**(1.69)
Gaussian (mult., $\mu = 1$)	0.50	2.38	**1.85** (4.86)
Gaussian (mult., $\mu = 1$)	0.60	2.76	**2.02** (3.21)
Gaussian (mult., $\mu = 1$)	0.70	4.51	**2.17** (2.98)
Gaussian (mult., $\mu = 1$)	0.80	8.54	**3.06** (3.13)
Gaussian (mult., $\mu = 1$)	0.90	18.20	**3.27** (3.26)
Gaussian (mult., $\mu = 1$)	1.00	33.16	**3.83** (3.76)
Uniform (add.)	0.10	0.86	**0.84** (4.01)
Uniform (add.)	0.20	1.45	**1.08** (1.59)
Uniform (add.)	0.30	6.29	**1.08** (1.25)
Uniform (add.)	0.40	25.45	**1.43** (1.41)
Uniform (add.)	0.50	47.16	**1.92** (1.56)
Uniform (mult.)	0.50	1.55	**1.15** (7.89)
Uniform (mult.)	1.00	2.75	**1.94** (3.91)
Uniform (mult.)	1.50	10.27	**2.89** (2.99)
Uniform (mult.)	2.00	68.10	**3.32** (3.55)

The MNIST dataset was used for the evaluation. Both the regular training and the proposed *noise-aware* training are evaluated. The classification error on the test set is reported (%). The parameter σ refers to the standard deviation of the Gaussian noise, while the parameter α refers to the range of the uniform noise, i.e., $[-\alpha, \alpha]$ for the additive noise and $[1-\alpha, 1+\alpha]$ for the multiplicative noise. The *cross-noise* classification error, i.e., the error obtained when the models are evaluated using a different type of noise than the one on which they were trained on, e.g., training on Gaussian noise and evaluating on uniform noise, is also reported in parenthesis. The underlined noise parameter corresponds to the noise used for this cross-noise evaluation.

Again, employing regularization seems to improve the accuracy of the network by allowing it to better adapt to the used hardware.

Furthermore, the effect of corrupting the activations using different types of noise (additive Gaussian, multiplicative Gaussian, additive uniform and multiplicative uniform) is evaluated in Table VI. Even though typically the AWGN model is employed for studying the effect of noise in many photonic components [38], we also considered different types of noise to demonstrate the ability of the proposed method to successfully train models that can withstand different types of noise. Any of the evaluated noise sources can have a devastating effect on the accuracy of the network, especially when its power is high enough. However, training the network by employing the proposed *noise-aware* scheme can significantly increase its robustness. For example, when very high levels of noise are used, e.g., additive Gaussian with $\sigma = 0.3$ or additive uniform with $\alpha = 0.5$, the classification error rises above 40%. However, when the same noise is employed during the training, which allows the network to adapt to the actual characteristics of the hardware implementation, the error drops to below 2%. Furthermore, we also evaluated the performance of the proposed noisy training approach using an additional challenging setup. In this setup the model was trained using a specific type of noise, e.g., Gaussian or uniform, but during the testing another type of noise, e.g., uniform or Gaussian (respectively), was used. This setup (cross-noise evaluation) aims to evaluate the ability of the trained model to withstand noise sources which were unknown during the training. These results are reported in parentheses in Table VI. Again, the proposed method improves the accuracy of the network over not using the proposed training approach. Note that the underlined noise parameter corresponds to the noise used for this cross-noise evaluation, e.g., for the additive Gaussian noise the results reported in parentheses correspond to evaluation with uniform additive noise with $\alpha = 0.30$.

TABLE VII
COMPARING THE PROPOSED TRAINING APPROACH TO DIFFERENT EXISTING INITIALIZATION APPROACHES

Dataset	**Network**	**Xavier [36]**	**He [39]**	**PSI+**
MNIST	CNN-1	4.20	6.95	**0.75**
Fashion-MNIST	CNN-1	15.42	16.82	**11.88**
Fashion-MNIST	CNN-2	12.94	11.48	**7.87**
CIFAR10	CNN-2	17.67	22.65	**16.91**

The classification error (%) on three different datasets (test set) is reported.

Finally, the ability of the proposed methods to tackle all the aforementioned issues, i.e., activation limits, weights limits and noise, at the *same time* is evaluated in Table VII. The proposed method (PSI combined with the aforementioned photonic adaptations) is compared to two other well known initialization approaches, i.e., Xavier [37] and He [39]. The activation limit was set to $u_{max} = 1.1$, the weight limit to $w_{max} = 1$, while additive Gaussian noise with $\sigma = 0.05$ was used. Several conclusions can be drawn from the results reported in Table VII. First, note that the Xavier initialization works better than He in most of the cases. However, neither of these approaches alone is enough for successfully training a network that can be deployed on photonic hardware. On the other hand, when the proposed adaptations are employed, the performance of the networks is significantly improved, e.g., for the MNIST dataset the error drops from 6.95% to 0.75%. Similar conclusions can be also drawn for all the evaluated networks, confirming the importance of adapting the training process to the characteristics of the actual neuromorphic architecture that will be employed. Finally, the proposed approach was also evaluated on a large scale high frequency limit order book dataset, the FI-2010 [43], [44], using the CNN-3 architecture (Table VIII). The first prediction horizon was used for the evaluation (please refer to [44] for details regarding the evaluation setup and metrics). Again, the proposed method allows for improving the prediction precision over both the Xavier and He initialization, only with minimal loss of precision compared to a baseline network with the ReLU activation.

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on July 14,2020 at 12:07:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE VIII
EVALUATION ON A HIGH FREQUENCY LIMIT ORDER BOOK DATASET

Method	F1	Cohen's κ
Baseline CNN	52.34 ± 2.39	0.2815 ± 0.0369
Photonic CNN + Xavier [36]	50.86 ± 2.58	0.2631 ± 0.0387
Photonic CNN + Kaiming [39]	50.75 ± 2.55	0.2612 ± 0.0393
Photonic CNN + PSI+	$\mathbf{50.98 \pm 2.36}$	$\mathbf{0.2699 \pm 0.0356}$

VI. CONCLUSION AND FUTURE WORK

Photonic hardware has the potential of providing significant performance and energy improvements over conventional hardware for deploying deep learning models. However, directly deploying a trained neural network into photonic hardware is not possible due to several physical restrictions and differences between the ideal components and the hardware actually used for the implementation. The models must be trained while taking into account the behavior of the physical components that will be used for the actual implementation. To the best of our knowledge, this paper presents the first in-depth study on training deep neural networks that can be deployed on photonic hardware. However, further research is needed to answer many questions that arise from this study. For example, is it possible to directly adapt a trained neural network for deployment on photonic hardware without training it from scratch, since training from scratch is computationally intensive and requires a significant amount of time, especially for larger datasets. Finally, is it possible to develop and co-design architectures and activation functions that better fit the photonic hardware, such as a photonic equivalent of the ReLU function, that will allow for training even deeper neural networks, since existing photonic activations are still prune to vanishing gradient phenomena for deeper architectures?

REFERENCES

[1] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vision Pattern Recognit.*, 2016, pp. 770–778.

[3] S. H. Jo, T. Chang, I. Ebong, B. B. Bhadviya, P. Mazumder, and W. Lu, "Nanoscale memristor device as synapse in neuromorphic systems," *Nano Lett.*, vol. 10, no. 4, pp. 1297–1301, 2010.

[4] G. Indiveri *et al.*, "Neuromorphic silicon neuron circuits," *Front. Neuroscience*, vol. 5, 2011, Art. no. 73.

[5] 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.

[6] S. Wen, H. Wei, Z. Zeng, and T. Huang, "Memristive fully convolutional network: An accurate hardware image-segmentor in deep learning," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 5, pp. 324–334, Oct. 2018.

[7] P. Wijesinghe, A. Ankit, A. Sengupta, and K. Roy, "An all-memristor deep spiking neural computing system: A step toward realizing the low-power stochastic brain," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 5, pp. 345–358, Apr. 2018.

[8] X. Shi, Z. Zeng, L. Yang, and Y. Huang, "Memristor-based circuit design for neuron with homeostatic plasticity," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 5, pp. 359–370, Oct. 2018.

[9] A. M. Zyarah and D. Kudithipudi, "Neuromorphic architecture for the hierarchical temporal memory," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 3, no. 1, pp. 4–14, Feb. 2019.

[10] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," in *Proc. Annu. Int. Symp. Comput. Architecture*, 2017, pp. 1–12.

[11] Y. Shen *et al.*, "Deep learning with coherent nanophotonic circuits," *Nature Photon.*, vol. 11, no. 7, pp. 441–446, 2017.

[12] K. Vandoorne *et al.*, "Experimental demonstration of reservoir computing on a silicon photonics chip," *Nature Commun.*, vol. 5, 2014, Art. no. 3541.

[13] X. Lin *et al.*, "All-optical machine learning using diffractive deep neural networks," *Science*, vol. 361, no. 6406, pp. 1004–1008, 2018.

[14] A. N. Tait *et al.*, "Neuromorphic photonic networks using silicon photonic weight banks," *Scientific Rep.*, vol. 7, 2017, Art. no. 7430.

[15] C. Yakopcic, R. Hasan, and T. M. Taha, "Memristor based neuromorphic circuit for ex-situ training of multi-layer neural network algorithms," in *Proc. Int. Joint Conf. Neural Netw.*, 2015, pp. 1–7.

[16] E. Hunsberger and C. Eliasmith, "Training spiking deep networks for neuromorphic hardware," 2016, *arXiv:1611.05141*.

[17] M. Prezioso, F. Merrikh-Bayat, B. Hoskins, G. Adam, K. K. Likharev, and D. B. Strukov, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, pp. 61–64, 2015.

[18] S. K. Esser, R. Appuswamy, P. Merolla, J. V. Arthur, and D. S. Modha, "Backpropagation for energy-efficient neuromorphic computing," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1117–1125.

[19] T. E. Potok *et al.*, "A study of complex deep learning networks on high-performance, neuromorphic, and quantum computers," *ACM J. Emerg. Technol. Comput. Syst.*, vol. 14, no. 2, 2018, Art. no. 19.

[20] N. Rathi and K. Roy, "STDP-based unsupervised multimodal learning with cross-modal processing in spiking neural network," *IEEE Trans. Emerg. Topics Comput. Intell.*, to be published.

[21] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2011, pp. 315–323.

[22] M. Miscuglio *et al.*, "All-optical nonlinear activation function for photonic neural networks," *Opt. Mater. Express*, vol. 8, no. 12, pp. 3851–3863, 2018.

[23] G. Mourgias-Alexandris, A. Tsakyridis, N. Passalis, A. Tefas, K. Vyrsokinos, and N. Pleros, "An all-optical neuron with sigmoid activation function," *Opt. Express*, vol. 27, no. 7, pp. 9620–9630, 2019.

[24] S. Pitris *et al.*, "O-band energy-efficient broadcast-friendly interconnection scheme with Sipho Mach-Zehnder Modulator (MZM) & arrayed waveguide grating router (AWGR)," in *Proc. Opt. Fiber Commun. Conf.*, 2018, pp. Th1G–5.

[25] M. Pantouvaki *et al.*, "Active components for 50 Gb/s NRZ-OOK optical interconnects in a silicon photonics platform," *J. Lightw. Technol.*, vol. 35, no. 4, pp. 631–638, Feb. 15, 2017.

[26] H. Ramon *et al.*, "70 Gb/s 0.87 pJ/bit GeSi EAM driver in 55 nm SiGe BiCMOS," in *Proc. Eur. Conf. Opt. Commun.*, 2018, pp. 1–3.

[27] J. Lambrecht *et al.*, "56-Gb/s silicon optical receiver using a low-noise fully-differential transimpedance amplifier in SiGe sicmos," in *Proc. Eur. Conf. Opt. Commun.*, 2018, pp. 1–3.

[28] L. Danial, N. Wainstein, S. Kraus, and S. Kvatinsky, "Breaking through the speed-power-accuracy tradeoff in ADCS using a memristive neuromorphic architecture," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 5, pp. 396–409, Oct. 2018.

[29] I. Chakraborty, D. Roy, and K. Roy, "Technology aware training in memristive neuromorphic systems for nonideal synaptic crossbars," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 2, no. 5, pp. 335–344, Oct. 2018.

[30] H. Liang *et al.*, "Memristive neural networks: A neuromorphic paradigm for extreme learning machine," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 3, no. 1, pp. 15–23, Feb. 2019.

[31] N. Passalis, G. Mourgias-Alexandris, A. Tsakyridis, N. Pleros, and A. Tefas, "Variance preserving initialization for training deep neuromorphic photonic networks with sinusoidal activations," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2019, pp. 1483–1487.

[32] A. Y. Ng, "Feature selection, l 1 vs. l 2 regularization, and rotational invariance," presented at the Int. Conf. Mach. Learn., 2004, Paper no. 78.

[33] S. S. Haykin, *Neural Networks and Learning Machines*. Hamilton, ON, Canada: McMaster Univ. Hamilton, 2009.

[34] S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 1135–1143.

[35] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.

[36] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proc. Int. Conf. Mach. Learn.*, 2008, pp. 1096–1103.

[37] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2010, pp. 249–256.

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on July 14,2020 at 12:07:17 UTC from IEEE Xplore. Restrictions apply.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

[38] R.-J. Essiambre, G. Kramer, P. J. Winzer, G. J. Foschini, and B. Goebel, "Capacity limits of optical fiber networks," *J. Lightw. Technol.*, vol. 28, no. 4, pp. 662–701, 2010.
[39] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. IEEE Int. Conf. Comput. Vision*, 2015, pp. 1026–1034.
[40] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
[41] H. Xiao, K. Rasul, and R. Vollgraf, " Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv:1708.07747*.
[42] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," University of Toronto, Toronto, ON, Canada, Tech. Rep. 1., 2009.
[43] A. Ntakaris, M. Magris, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Benchmark dataset for mid-price prediction of limit order book data," *J. Forecasting*, 2017, *arXiv:1705.03233*.
[44] N. Passalis, A. Tefas, J. Kanniainen, M. Gabbouj, and A. Iosifidis, "Temporal bag-of-features learning for predicting mid price movements using high frequency limit order book data," *IEEE Trans. Emerg. Topics Comput. Intell.*, pp. 1–12, 2018. Early Access, doi: 10.1109/TETCI.2018.2872598.
[45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representations*, 2015, pp. 1–13.
[46] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural Netw. Mach. Learn.*, vol. 4, pp. 26–31, 2012.

Nikolaos Passalis is a Postdoctoral Researcher at the Tampere University, Tampere, Finland. He received the B.Sc. degree in informatics in 2013, the M.Sc. degree in information systems in 2015, and the Ph.D. degree in informatics in 2018, from the Aristotle University of Thessaloniki, Thessaloniki, Greece. He has coauthored 45 journal and conference papers and contributed one chapter to one edited book. His research interests include deep learning, information retrieval, and computational intelligence.

George Mourgias-Alexandris received the B.S. degree in electrical and computer engineering, in 2016 from the University of Thessaly, Volos, Greece. At the beginning of 2018, he received M.S degree in computer system networking from Aristotle University of Thessaloniki, Thessaloniki, Greece, where he is currently working toward the Ph.D. degree. During the Summer of 2018, he was a research intern at the Optics Laboratory of Microsoft Research, Cambridge, UK, working on optical network technologies for novel designs of regional and wide area networks. His research interests include optical communications and optical computing.

Apostolos Tsakyridis received the B.S. degree from the Department of Electrical and Computer Engineering, University of Thessaly, Volos, Greece, in 2016. His undergraduate thesis concerned the implementation of optimizing code transformations in a high-level synthesis compiler. He is currently working towards the master's degree in networks communications and systems architectures with the Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece. His research interests include optical communications, neuromorphic photonics, and optical memories.

Nikos Pleros received the Diploma and Ph.D. degree in electrical and computer engineering from National Technical University of Athens, Athens, Greece, in 2000 and 2004, respectively. In September 2007, he joined the faculty of the Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, Greece, where he is currently an Assistant Professor. He has authored/coauthored more than 250 archival journal publications and conference presentations including several invited contributions. He has held positions of responsibility at several major conference committees including ECOC, OFC, and SPIE Photonics West. His research interests include optical interconnect technologies and architectures, photonic integrated circuit technologies, optical technologies for disaggregated data center architectures and high-performance computing, silicon photonics and plasmonics, optical signal processing, optical switching, as well as fiber-wireless technologies, network architectures and protocols for 5G mobile networks. He has coordinated several FP7 and H2020 European projects including ICT-STREAMS, PlasmoFab, RAMPLAS, PLATON and 5G-PHOS, while he has participated as partner in more than ten additional projects. He was a recipient of the 2003 IEEE Photonics Society Graduate Student Fellowship granted to 12 Ph.D. candidates worldwide in the field of photonics, while he was proud to (co-) supervise three more fellowship winners (Dr. D. Fitsios in 2014, Dr. C. Vagionas in 2016, and Dr. P. Maniotis in 2017) during their Ph.D. He was also a recipient of the 15th prize in the Greek Mathematical Olympiad. He is a member of the IEEE Photonics Society and the IEEE Communications Society.

Anastasios Tefas received the B.Sc. and Ph.D. degrees in informatics, from the Aristotle University of Thessaloniki, Thessaloniki, Greece, in 1997 and 2002, respectively. Since 2017, he has been an Associate Professor with the Department of Informatics, Aristotle University of Thessaloniki. From 2008 to 2017, he was a Lecturer and Assistant Professor with the same University. From 2006 to 2008, he was an Assistant Professor with the Department of Information Management, Technological Institute of Kavala. From 2003 to 2004, he was a temporary Lecturer with the Department of Informatics, University of Thessaloniki. From 1997 to 2002, he was a Researcher and Teaching Assistant with the Department of Informatics, University of Thessaloniki. . His current research interests include computational intelligence, deep learning, pattern recognition, statistical machine learning, digital signal and image analysis and retrieval and computer vision.

He participated in 18 research projects financed by national and European funds. He has coauthored 100 journal papers, 215 papers in international conferences, and contributed eight chapters to edited books in his area of expertise. More than 4900 citations have been recorded to his publications and his H-index is 36 according to Google Scholar

Authorized licensed use limited to: Aristotle University of Thessaloniki. Downloaded on July 14,2020 at 12:07:17 UTC from IEEE Xplore. Restrictions apply.