Neural Networks 129 (2020) 103–108

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Initializing photonic feed-forward neural networks using auxiliary tasks

Nikolaos Passalis [a,*], George Mourgias-Alexandris [b], Nikos Pleros [b], Anastasios Tefas [a]

[a] Artificial Intelligence and Information Analysis Laboratory, Aristotle University of Thessaloniki, Greece
[b] Photonic Systems and Networks Research Group, Department of Informatics, Aristotle University of Thessaloniki, Greece

ARTICLE INFO

Article history:
Received 19 November 2019
Received in revised form 15 May 2020
Accepted 21 May 2020
Available online 1 June 2020

Keywords:
Photonic deep learning
Neural network initialization
Photonic activation functions

ABSTRACT

Photonics is among the most promising emerging technologies for providing fast and energy-efficient Deep Learning (DL) implementations. Despite their advantages, these photonic DL accelerators also come with certain important limitations. For example, the majority of existing photonic accelerators do not currently support many of the activation functions that are commonly used in DL, such as the ReLU activation function. Instead, sinusoidal and sigmoidal nonlinearities are usually employed, rendering the training process unstable and difficult to tune, mainly due to vanishing gradient phenomena. Thus, photonic DL models usually require carefully fine-tuning all their training hyper-parameters in order to ensure that the training process will proceed smoothly. Despite the recent advances in initialization schemes, as well as in optimization algorithms, training photonic DL models is still especially challenging. To overcome these limitations, we propose a novel adaptive initialization method that employs auxiliary tasks to estimate the optimal initialization variance for each layer of a network. The effectiveness of the proposed approach is demonstrated using two different datasets, as well as two recently proposed photonic activation functions and three different initialization methods. Apart from significantly increasing the stability of the training process, the proposed method can be directly used with any photonic activation function, without further requiring any other kind of fine-tuning, as also demonstrated through the conducted experiments.

1. Introduction

Recent advances in Deep Leaning (DL) led to state-of-the-art solutions to many difficult problems (LeCun, Bengio, & Hinton, 2015), e.g., computer vision (Stallkamp, Schlipsing, Salmen, & Igel, 2012; Szegedy, et al., 2015). However, the enormous complexity of DL models led to the development of specialized hardware tailored to accelerating both the training and inference processes using DL models. Perhaps the most commonly used hardware to achieve this is Graphics Processing Units (GPUs). Apart from using GPUs, other emerging platforms, such as Tensor Processing Units (TPUs), as well as neuromorphic hardware (Kasabov, et al., 2016; Merolla, et al., 2014; Pantone, Kendall, & Nino, 2018; Walter, Röhrbein, & Knoll, 2015), are also increasingly used to this end. Furthermore, there is an increasing interest in *photonic*-based DL implementations. Photonic DL accelerators are especially promising, being able to provide fast and low energy implementations of DL models (Shen, et al., 2017). These platforms employ light to represent signals that are appropriately manipulated through various electro-optical or purely optical components to provide the functionality of DL layers. In this way, the various signals are propagated through the photonic components near to the speed of light, while the resulting configurations can operate at extremely high frequencies.

However, most photonic architectures require significant changes in the way DL models are structured and trained. Among the most important differences with software-based DL implementations are the employed activation functions. In photonic neuromorphics the activation functions that can be actually implemented are limited by the underlying hardware. As a result, instead of using the typical activation functions developed for DL models, such as ReLU (He, Zhang, Ren, & Sun, 2015), most photonic architectures employ either sinusoidal (Tait, et al., 2017), or sigmoidal activations functions (Mourgias-Alexandris, et al., 2019), which provide bounded outputs that ensure that the employed photonic devices always operate within the appropriate limits. It is worth noting that even when ReLU-like activations are employed (Tait, et al., 2019), their behavior is significantly different from their ideal counterparts, requiring appropriately retraining the DL models before using these functions in order to ensure that the models will exhibit the same behavior when implemented on photonic hardware. However, DL models are especially sensitive to using easily saturated activation functions,

* Corresponding author.
E-mail address: passalis@csd.auth.gr (N. Passalis).

https://doi.org/10.1016/j.neunet.2020.05.024
0893-6080/

requiring a significant amount of hyper-parameter tuning, as well as deriving initialization schemes specifically designed for each activation function, e.g., Passalis, Mourgias-Alexandris, Tsakyridis, Pleros, and Tefas (2019), just to ensure that the training process will proceed smoothly and it will not prematurely converge into a bad local minimum. At the same time, most of these initialization approaches come with a number of assumptions, e.g., they employ locally linear approximations of the activation functions, and ignore the actual distribution of the data, leading to significantly lower performance compared with software-based DL models that employ other, less sensitive, activation functions (e.g., ReLU) (Passalis et al., 2019). Therefore, despite the recent advances in initialization schemes (Glorot & Bengio, 2010; He et al., 2015; Passalis et al., 2019), as well as in optimization algorithms (Tieleman & Hinton, 2012; Wang, et al., 2018), training photonic DL models remains especially challenging.

To overcome these limitations, in this paper we propose an adaptive initialization method that employs auxiliary tasks to estimate the optimal initialization variance individually for each layer of a network, instead of relying on manually calculated estimations, e.g., Glorot initialization (Glorot & Bengio, 2010), He initialization (He et al., 2015), or schemes that are adapted to photonic activations (Passalis et al., 2019). The auxiliary task employed in this paper is to train each individual layer to extract a representation that is suitable for the main task at hand just by appropriately scaling the initial weights of each layer. Therefore, the used auxiliary task is directly derived from the main task for which the model will be trained for. The term *auxiliary* task is used in order to distinguish the process of learning the initialization variance for each individual layer from the main learning task, which is to learn the parameters of all the layers simultaneously in order to facilitate the main task at hand. Furthermore, the proposed method is activation-agnostic, i.e., it can be directly used with any photonic activation function, it does not require any manual fine-tuning of any hyper-parameter or rely on manually calculating the variance for each layer. At the same time it takes into account the distribution of the input data, as well as the task at hand. The effectiveness of the proposed method, along with its ease of use, provides a solid step toward accelerating research on photonic neuromorphic architectures. It is worth noting that the proposed method is capable of training photonic neural networks, that would be otherwise especially difficult to train using conventional approaches, as it is experimentally demonstrated in Section 4. To this end, two image datasets, along with two recently proposed photonic activation functions and three different initialization methods are employed. It is worth noting that the proposed method is not limited to photonic networks, since it can be readily applied for any neural network architecture and/or activation function. However, this paper mainly focuses on photonic networks, where using such initialization approaches is critical for successfully training the corresponding networks, as we also experimentally demonstrate, since the activation functions that are employed are dictated by the behavior of the underlying photonic substrate.

The rest of the paper is structured as follows. First, the photonic activation functions considered in this paper are briefly introduced in Section 2. Next, the proposed method is presented in Section 3. Finally, the experimental evaluation is provided in Section 4, while conclusions are drawn in Section 5.

2. Photonic activation configurations

Two different photonic activation functions are considered in this paper: (a) a sinusoidal-based activation scheme (Tait, et al., 2017) and (b) an all-optical sigmoid-based activation scheme (Mourgias-Alexandris, et al., 2019).

The first one employs an Opto–Electro–Optical (O–E–O) scheme relied on a Balanced Photodetector (BPD) that is followed by a Mach–Zehnder Modulator (MZM) (Tait, et al., 2017). The electrical output of the BPD is used to control the MZM, 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 the notation P_{out} is used to refer to the output signal, P_{in} is the signal to be modulated, V_{π} refer to the voltage required for having a π phase shift and V_{RF} is the electrical control signal of the MZM, which is the actual input of the activation function. The transfer function described in (1) can be further simplified as (Passalis et al., 2019):

$$f_{sin}(x) = \begin{cases} 0, & \text{if } x \leq 0 \\ sin^2\left(\frac{\pi}{2}x\right), & \text{if } x > 0, \end{cases} \tag{2}$$

given that the activations can be always scaled before the actual hardware implementation (the appropriate scaling can be directly calculated according to the voltage V_{π}) and after taking into account that negative power cannot be represented. Note that this configuration employs electronic components, e.g., BPDs that are used to convert the optical signal to an electrical signal, which is then used to modulate a reference optical signal through an MZM. However, the O–E–O conversions is not a limiting factor for the bandwidth of the system, as the activation circuit can be implemented with state-of-the-art InP components, such as the BPD of Runge, et al. (2018) and the MZM of Lange, et al. (2018) that can support data rates up to 100Gbaud. Moreover, a similar circuit relied on a BPD followed by an optical modulator have already been implemented on a CMOS-compatible silicon photonic chip (Tait, et al., 2019), while the proper functionality of the layout has been confirmed by the experimental demonstration in feed-forward and recurrent configurations.

For the second activation scheme, an all-optical activation function is implemented as follows (Mourgias-Alexandris, et al., 2019): two cascaded wavelength converters are realized by a Semiconductor-Optical-Amplifier-assisted (SOA) Mach–Zehnder Interferometer (MZI) followed by an SOA that performs cross-gain modulation on its small-signal gain regime. The transfer function of the aforementioned activation scheme can be closely approximated by a generalized logistic function, which can be appropriately fitted to follow the actual behavior of this activation configuration, as measured through experimental observations. Therefore, the activation function is modeled as:

$$f_{sigm}(x) = A_2 + \frac{A_1 - A_2}{1 + e^{((x - x_0)/d)}}, \tag{3}$$

where the parameters of the generalized logistic function were set to $A_1 = 0.060, A_2 = 1.005, x_0 = 0.145$, and $d = 0.033$, according to the experimental findings reported in Mourgias-Alexandris, et al. (2019). Note that these values are an intrinsic property of the specific hardware implementation and, as a result, do not change according to the task at hand or when different models are used (as long as the same activation function is employed).

3. Proposed method

Consider a DL model (neural network) that consists of n layers. Each layer can be either fully connected or convolutional and it is equipped with a set of parameters (weights). To simplify the presentation of the proposed method we will first consider the case of fully connected layers. However, this is without loss of generality, since the proposed method can be trivially extended to handle convolutional layers, as we will demonstrate later. Let $\mathbf{W}_i \in \mathbb{R}^{m^{(i-1)} \times m^{(i)}}$ be the weights of the ith layer, where $m^{(i-1)}$

denote the dimensionality of the input to the layer, while $m^{(i)}$ is the number of neurons in the layer (output dimensionality). Note that the notation $m^{(0)}$ is used to refer to the dimensionality of the input to the neural network. The weighted output of a neuron can be then calculated as:

$$\mathbf{u}_i = \mathbf{W}_i^T \mathbf{y}_{i-1} + \mathbf{b}_i \in \mathbb{R}^m, \tag{4}$$

where $\mathbf{y}_{i-1} \in \mathbb{R}^{(m-1)}$ denotes the output of the previous layer and $\mathbf{b}_i$ are the biases for the neurons of the ith layer. Note again that the notation $\mathbf{y}_0$ is used to refer to the input of the neural network. The final output of the layer is then obtained by applying a nonlinear activation function $f(\cdot)$:

$$\mathbf{y}_i = f(\mathbf{u}_i) \in \mathbb{R}^m. \tag{5}$$

The initial weights of the neural layers are typically drawn from a random distribution. Using the most appropriate distribution for the initialization process is crucial. For example, if very large random values are used for initializing the network, the activation functions will be saturated, trapping, as a result, the network in a region from which it will be very difficult to recover. Most of the existing initialization approaches, e.g., He et al. (2015), Xavier (Glorot & Bengio, 2010), etc., calculate the optimal initialization variance given the size of the layers (fan-in and fan-out), as well as considering the characteristics of the employed activation function. However, as discussed in Section 1, these methods need to be carefully adapted to the characteristics of different activation functions, while they also come with assumptions that do not always hold (e.g., linear approximations are usually applied). This can limit their performance in real applications, as demonstrated in detail in Passalis et al. (2019) for a photonic sinusoidal activation functions using a wide range of different datasets, ranging from image datasets to time series datasets. For the rest of this paper, we will consider the case of Gaussian initialization, i.e., drawing the initial weights from a Gaussian distribution with zero mean and σ_i^2 variance $\mathcal{N}(0, \sigma_i^2)$. Note that this is without loss of generality, since the proposed method can be easily extended for other distributions (e.g., a uniform distribution).

Instead of manually deriving the optimal variance for different network architectures and different activation functions, we propose employing an auxiliary task for implicitly estimating the variance that should be used for initializing each layer of a network. To this end, we propose training shallow networks (up to two layers) for which the training process is typically stable and less prone to vanishing gradient phenomena. This can be better understood by observing that many stability issues arise when multiple nonlinear transformations are used, leading to vanishing gradients when the layers are not properly initialized. However, this issue affects to a significantly smaller degree networks with up to two layers. Therefore, in this paper we exploit this property to gradually estimate the best initialization variance for all the layers of the network by employing an auxiliary training task.

To this end, we first modify the network architecture by including a scaling factor $\alpha_i \in \mathbb{R}$ that allows for scaling all the weights of a layer at the same time:

$$\mathbf{y}_i = f(\alpha_i \mathbf{u}_i) \in \mathbb{R}^m. \tag{6}$$

Altering the values for the parameter α_i, while keeping the weights constant, is equivalent to altering the initialization variance of the network to $(\alpha_i \sigma)^2$, i.e., drawing the weights from $\mathcal{N}(0, (\alpha_i \sigma)^2)$. Then, we add an additional auxiliary classification layer with weights $\mathbf{W}_i^a \in \mathbb{R}^{m^{(i)} \times N_C}$ directly on top of the representation extracted from the ith layer, where N_C is the number of classes. Note that this is without loss of generality, since the layer can be directly trained for any auxiliary task, e.g., regression. Finally, all the weights of the network are kept fixed, while we are optimizing only the scaling parameter α_i and the auxiliary weights $\mathbf{W}_i^a$ using gradient descent:

$$\Delta \mathbf{W}_i^a = -\eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}_i^a}, \tag{7}$$

and

$$\Delta \alpha_i = -\eta \frac{\partial \mathcal{L}}{\partial \alpha_i}, \tag{8}$$

where η is the used learning rate and $\mathcal{L}$ is the used loss function (the cross-entropy loss was used for all the experiments conducted in this paper). Training the auxiliary layer in this way allows for simultaneously scaling the weights of the attached layer in order to facilitate the task at hand, providing in this way an implicit estimation for the optimal initialization variance.

Note that for any layer i the gradients only back-propagate through one nonlinearity and the auxiliary weights $\mathbf{W}_i^a$ before arriving to the parameter α_i. Therefore, in this way we ensure that the gradients will not diminish, allowing for effectively estimating the scale for the weights that will accommodate the classification task at hand. Also, it should be noted that the weights $\mathbf{W}_i$ of each layer are not optimized through this process, which is merely used for estimating the optimal initialization variance. To better understand why the aforementioned process works, despite relying on just a random transformation of the input data, we should consider that such random transformations, which essentially project the data in a higher dimensional space, come with strong theoretical guarantees (Huang, Zhou, Ding, & Zhang, 2011; Rahimi & Recht, 2008). In fact, it has been proven that they can effectively act as universal approximators (Huang et al., 2011; Rahimi & Recht, 2008). Therefore, the proposed method provides a way of learning the variance that should be used when such random transformations are employed in order to ensure that meaningful features are extracted without saturating the used activation functions.

The proposed variance estimation process is gradually repeated for all the layers of the network, starting from the first one until reaching the final classification layer. After calculating the most appropriate value for the variance of any layer, each layer is appropriately re-initialized and the weights $\mathbf{W}_i^a$ are discarded. Note that the proposed method can be also easily extended to handle convolutional layers. In this case, the output of a layer is not just a vector, but a higher-dimensional tensor. For example, for images, where 2D convolution is applied, the output of the ith layer would be $\mathbf{y}_i \in \mathbb{R}^{w \times h \times c}$, where w and h denote the width and height of the extracted feature map, while c refers to the number of convolutional filters. To avoid dealing with extremely large feature maps and allow for more efficiently aggregating the gradients, we propose applying the auxiliary layer after performing global sum pooling:

$$\tilde{\mathbf{y}}_i = \sum_{i=l}^{w} \sum_{m=1}^{h} [\mathbf{y}_i]_{lm} \in \mathbb{R}^c, \tag{9}$$

where the notation $[\mathbf{y}_i]_{lm} \in \mathbb{R}^c$ is used to refer to the feature vector that exists at the (l, m) position of the feature map. Sum pooling was employed instead of max pooling to ensure that all the extracted feature vectors will contribute during the back-propagation process. Note that average pooling can be used to this end as well.

The proposed method is summarized in Algorithm 1. First, the weights are initialized using any existing initialization method (line 1), e.g., Glorot initialization (Glorot & Bengio, 2010), He initialization (He et al., 2015), etc. Note that the proposed method is less sensitive on the initialization algorithm that will be used, since it can effectively recover even from bad initializations, as

Algorithm 1 Auxiliary Task-based Initialization for Photonic Feed-forward Neural Networks

Input: Initial weights $\mathbf{W}_i$, number of estimation iterations N, initial learning rate η_{init}
Output: Initialization variance for each layer σ_i^2

1: Employ any initialization method for drawing the initial weights
2: **for** i = 1 **to** n **do**
3: Initialize α_i to 1
4: Initialize the learning rate to $\eta = \eta_{init}$
5: **for** j = 1 **to** N **do**
6: Feed-forward through the network up to layer i and extract the representation $\mathbf{y}_i = f(\alpha_i \mathbf{u}_i) \in \mathbb{R}^m$
7: **if** layer i is convolutional **then**
8: Apply global sum pooling according to (9)
9: **end if**
10: Update parameters according to $\Delta\alpha_i = -\eta \frac{\partial \mathcal{L}}{\partial \alpha_i}$ and $\Delta \mathbf{W}_i^a = -\eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}_i^a}$
11: **if** $\alpha_i < 0$ **then**
12: $\alpha_i = |\alpha_i|$
13: $\eta = \eta/2$
14: **end if**
15: **end for**
16: Estimate the initialization variance $\sigma_i^2 = \alpha_i^2 Var[\mathbf{W}_i]$
17: **end for**
18: **return** Initialization variances σ_i^2

it will be further demonstrated in Section 4. Next, the most appropriate initialization variance is calculated for each of the layers of the network (lines 2 to 16). Global sum pooling is first applied for convolution layers (lines 7–9) in order to extract a compact representation, through which the gradients will back-propagate to the scaling factors. Also, note that the scaling factor α_i could also end up having negative values, especially if a learning rate larger than the optimal one is used. Even though this does not affect the variance estimation, it can lead to instabilities during the training process, since a negative scaling factor leads to flipping the sign of all the weights of a neuron. Therefore, the scaling factor was restricted to be always a positive quantity, to avoid such phenomena. To this end, we detect whenever the optimization process overshoots the minimum (line 10), ending up into a negative value and we restore the parameter to have a positive value (line 11). We also reduce the learning rate to half, whenever this happens (line 12). As it was experimentally established, this simple process allows for always starting with an initially large value for the learning rate, which is then automatically adapted to the task at hand, without having to manually fine-tune the learning rate for each different application. Finally, after N optimization iterations have been performed, the final initialization variance is calculated (line 16). The proposed process takes into account the distribution of the data used for training, since the variance is calculated using the same data as the one used for solving the actual task at hand and not only the structure of the network/activation function, in contrast with most of the existing initialization schemes (Glorot & Bengio, 2010; He et al., 2015).

The proposed method requires performing $O(n \cdot N)$ additional training epochs, where n is the number of layers of the network and N is the number of estimation iterations. Even though the cost of one training epoch for the proposed method is lower compared to a full training epoch of the network (there is no reason to feed-forward through the whole network for the earlier layers), we assume that the cost is comparable in order to simplify the conducted analysis. Also, let N_f be the number of training epochs required for training the whole network, after initializing the model (either with a static initialization approach or the proposed method). Then, the maximum possible overhead for the proposed method is $\kappa = 100 \cdot \frac{n \cdot N}{N_f}\%$.

4. Experimental evaluation

The proposed method was evaluated using two well known image datasets: (a) the Fashion MNIST dataset (Xiao, Rasul, & Vollgraf, 2017) and the CIFAR-10 dataset (Krizhevsky, Hinton, et al., 2009). The convolutional neural network architecture used for the conducted experiments is composed of 4 convolutional layers followed by 2 fully connected layers. More specifically, two convolutional layers with 32 and 64 filters of size 3×3 were employed, followed by a 2×2 average pooling layer, another two convolutional layers with 128 and 256 filters of size 3×3 and a final 2×2 average pooling layer. Next, the output of the pooling layer was flattened into a vector that was fed to a hidden layer with 512 neurons, before forwarding it to the final classification layer. This led to a network with about 2.5 million parameters for the Fashion MNIST dataset and with 3.7 million parameters for the CIFAR-10 dataset. Note that even though these networks are beyond the current capacity of existing neuromorphic hardware, the employed 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, allowing to implement significantly larger networks (Huang, Fujisawa, Ferreira de Lima, Tait, Blow, Tian, et al., 2020). All the networks were trained using the Adam optimization algorithm. The optimization ran for 50 epochs with a learning rate of $\eta = 10^{-4}$, while the networks were further fine-tuned for another 50 epochs additional iterations with a reduced learning rate of $\eta = 10^{-5}$. The proposed estimation method ran for $N = 5$ epochs with the initial learning rate set to $\eta = 0.1$.

Three different initialization approaches were evaluated: (a) He initialization (abbreviated as "He") (He et al., 2015), (b) Xavier initialization (abbreviated as "Xavier") (Glorot & Bengio, 2010) and (c) Sinusoidal initialization (abbreviated as "Sin"), which is specifically designed for the photonic sinusoidal function, as proposed in Passalis et al. (2019). Both the photonic sinusoidal activation function (Tait, et al., 2017) and the sigmoidal photonic activation function (Mourgias-Alexandris, et al., 2019) were used for the conducted experiments. For each of these functions we evaluated all the three available initialization schemes, along with the proposed method (separately combined with different initialization schemes). Finally, we also evaluated a baseline DL architecture, where the ReLU activation function was used in the place of the evaluated photonic activations.

The evaluation results are reported for the Fashion MNIST dataset in Table 1. Each experiment was repeated three times and we report the mean and standard deviation. We can draw several interesting conclusions based on the reported results. First, note that using a sub-optimal initialization scheme can significantly increase the classification error when photonic activation functions are used. For example, the error increases from around 7% (Baseline ReLU model) to over 15% when the photonic sigmoid activation is employed (regardless the initialization scheme). Furthermore, note that using an appropriately designed initialization scheme indeed reduces the classification error, since the error for the photonic sinusoidal function is reduced to 8.34% when the method proposed in Passalis et al. (2019) was employed. In all the evaluated cases employing the proposed method further reduces the error, by a significant margin, e.g., the error is reduced to 8.01% for the photonic sinusoidal function and to 8.24% for the photonic sigmoidal function. Also note that the proposed method

Table 1
Evaluation results on the Fashion MNIST dataset.

Activation	Initialization	Classification error
ReLU	He (He et al., 2015)	7.07 ± 0.06%
ReLU	Xavier (Glorot & Bengio, 2010)	6.92 ± 0.04%
Photonic Sinusoidal	Sin (Passalis et al., 2019)	8.34 ± 0.25%
Photonic Sinusoidal	He (He et al., 2015)	9.84 ± 1.12%
Photonic Sinusoidal	Proposed + He	**8.93 ± 0.73%**
Photonic Sinusoidal	Xavier (Glorot & Bengio, 2010)	8.08 ± 0.29%
Photonic Sinusoidal	Proposed + Xavier	**8.01 ± 0.29%**
Photonic Sigmoid	He (He et al., 2015)	19.35 ± 0.38%
Photonic Sigmoid	Proposed + He	**8.56 ± 0.23%**
Photonic Sigmoid	Xavier (Glorot & Bengio, 2010)	17.08 ± 0.58%
Photonic Sigmoid	Proposed + Xavier	**8.24 ± 0.31%**

Table 2
Evaluation results on the CIFAR-10 MNIST dataset.

Activation	Initialization	Classification error
ReLU	He (He et al., 2015)	16.89%
ReLU	Xavier (Glorot & Bengio, 2010)	15.21%
Photonic Sinusoidal	Sin (Passalis et al., 2019)	30.97%
Photonic Sinusoidal	He (He et al., 2015)	50.53%
Photonic Sinusoidal	Proposed + He	**21.80%**
Photonic Sinusoidal	Xavier (Glorot & Bengio, 2010)	18.10%
Photonic Sinusoidal	Proposed + Xavier	**16.43%**
Photonic Sigmoid	He (He et al., 2015)	90.00%
Photonic Sigmoid	Proposed + He	**25.27%**
Photonic Sigmoid	Xavier (Glorot & Bengio, 2010)	90.00%
Photonic Sigmoid	Proposed + Xavier	**18.97%**

leads to more stable training, since in all the evaluated cases the standard deviation is reduced when the proposed method is applied. Regarding the computational complexity, for the conducted experiments we used $n = 6$, $N = 5$ and $N_f = 100$, as described before. Therefore, the maximum theoretical overhead according to the complexity analysis conducted in Section 3 would be at most 30%. Experimentally we verified that this is indeed the upper limit, since the training time increased from 36 min (no adaptive initialization was applied) to 44 min (less than 25% overhead) after applying the proposed method. Note that the actual overhead greatly depends on the structure of the model (having more complex early layers leads to a larger overhead, but in any case the total overhead remains less than κ). It is worth noting that the overhead can be further reduced by using adaptive methods, capable of automatically stopping the training process, when a good enough estimation of the initialization variance has been obtained. For example, reducing the number of estimation epochs from $N = 5$ to $N = 1$ increases the classification error less than 0.3% for the Fashion MNIST dataset (and the photonic sinusoidal activation).

The experimental results are even more impressing for the CIFAR-10 dataset (Table 2), where in many cases it was actually impossible to train the models (e.g., using the photonic sigmoid activation and either the He or Xavier initialization). However, when the proposed method was employed the DL model actually achieved higher accuracy (16.43% error) compared to using the baseline DL model (ReLU + He initialization leads to a classification error of 16.89%). Finally, note that the proposed method is especially stable and robust, being able to recover badly initialized networks (as in the last cases reported in Table 2). It is worth stressing again that the same hyper-parameters were used for all the conducted experiments, without performing any kind of model/dataset-specific fine-tuning.

5. Conclusions

In this paper we proposed and evaluated a novel adaptive and data-driven initialization approach for photonic feed-forward DL models. The proposed method employs auxiliary tasks to estimate the optimal initialization variance for the networks, while it is also activation-agnostic, i.e., it can be directly used with any photonic activation function and allows for taking into account the actual distribution of the input data. The effectiveness of the proposed method was demonstrated using two different datasets, two recently proposed photonic activation functions, as well as three different initialization methods. It was also experimentally demonstrated that the proposed method is especially robust, allowing for recovering badly initialized networks that were almost impossible to train using conventional approaches.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The research work was supported by the Hellenic Foundation for Research and Innovation (H.F.R.I.), Greece under the "First Call for H.F.R.I. Research Projects to support Faculty members and Researchers and the procurement of high-cost research equipment grant" (Project Number: 4233)

References

Glorot, X., & Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the international conference on artificial intelligence and statistics*. (pp. 249–256).

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the international conference on computer vision*. (pp. 1026–1034).

Huang, C., Fujisawa, S., Ferreira de Lima, T., Tait, A. N., Blow, E., Tian, Y., et al. (2020). Demonstration of photonic neural network for fiber nonlinearity compensation in long-haul transmission systems. In *Proceedings of the optical fiber communication conference*.

Huang, G.-B., Zhou, H., Ding, X., & Zhang, R. (2011). Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, *42*(2), 513–529.

Kasabov, N., Scott, N. M., Tu, E., Marks, S., Sengupta, N., Capecci, E., et al. (2016). Evolving spatio-temporal data machines based on the NeuCube neuromorphic framework: design methodology and selected applications. *Neural Networks*, *78*, 1–14.

Krizhevsky, A., Hinton, G., et al. (2009). *Learning multiple layers of features from tiny images*: *Tech. rep.*, Citeseer.

Lange, S., Wolf, S., Lutz, J., Altenhain, L., Schmid, R., Kaiser, R., et al. (2018). 100 GBd intensity modulation and direct detection with an InP-based monolithic DFB laser Mach–Zehnder modulator. *Journal of Lightwave Technology*, *36*(1), 97–102.

LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436.

Merolla, P. A., Arthur, J. V., Alvarez-Icaza, R., Cassidy, A. S., Sawada, J., Akopyan, F., et al. (2014). A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, *345*(6197), 668–673.

Mourgias-Alexandris, G., Tsakyridis, A., Passalis, N., Tefas, A., Vyrsokinos, K., & Pleros, N. (2019). An all-optical neuron with sigmoid activation function. *Optics Express*, *27*(7), 9620–9630.

Pantone, R. D., Kendall, J. D., & Nino, J. C. (2018). Memristive nanowires exhibit small-world connectivity. *Neural Networks*, *106*, 144–151.

Passalis, N., Mourgias-Alexandris, G., Tsakyridis, A., Pleros, N., & Tefas, A. (2019). Training deep photonic convolutional neural networks with sinusoidal activations. *IEEE Transactions on Emerging Topics in Computational Intelligence*.

Rahimi, A., & Recht, B. (2008). Random features for large-scale kernel machines. In *Proceedings of the advances in neural information processing systems*. (pp. 1177–1184).

Runge, P., Zhou, G., Ganzer, F., Keyvaninia, S., Mutschall, S., Seeger, A., et al. (2018). 100GHz Balanced Photodetector Module. In *2018 conference on lasers and electro-optics*. (pp. 1–2).

Shen, Y., Harris, N. C., Skirlo, S., Prabhu, M., Baehr-Jones, T., Hochberg, M., et al. (2017). Deep learning with coherent nanophotonic circuits. *Nature Photonics*, *11*(7), 441.

Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2012). Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, *32*, 323–332.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., et al. (2015). Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*.(pp. 1–9).

Tait, A. N., De Lima, T. F., Zhou, E., Wu, A. X., Nahmias, M. A., Shastri, B. J., et al. (2017). Neuromorphic photonic networks using silicon photonic weight banks. *Scientific Reports*, *7*(1), 7430.

Tait, A. N., de Lima, T. F., Nahmias, M. A., Miller, H. B., Peng, H.-T., Shastri, B. J., et al. (2019). Silicon photonic modulator neuron. *Physical Review A*, *11*(6), 064043.

Tieleman, T., & Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, *4*(2), 26–31.

Walter, F., Röhrbein, F., & Knoll, A. (2015). Neuromorphic implementations of neurobiological learning algorithms for spiking neural networks. *Neural Networks*, *72*, 152–167.

Wang, C.-C., Tan, K. L., Chen, C.-T., Lin, Y.-H., Keerthi, S. S., Mahajan, D., et al. (2018). Distributed newton methods for deep neural networks. *Neural Computation*, *30*(6), 1673–1724.

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747.