Feasibility Study of Training Neural Networks Using Photonics

T.J.C.Tubbing,[1] R. Stabile,[1] and B. Shi[1]

[1] Eindhoven University of Technology, Dept. of Electrical Engineering, De Groene Loper 19, 5612 AP Eindhoven, The Netherlands

Photonics has the potential to be faster and more energy efficient than conventional electronics in parallel computing. We investigate the feasibility of training neural networks using photonics, in terms of speed and energy consumption. Two different training configurations are created: The first configuration is implemented on an FPGA, the second configuration replaces the matrix multiplications of the first configuration with their photonic equivalent. Both configurations are exploited to train a neural network to function as a XOR operation to get an accuracy of 100 %. An average loss of 42·10-5 for 8-bit decimal precision is acquired on the FPGA configuration. An average loss of 39·10-5 for the same bit precision is measured on the hybrid FPGA-photonics configuration. This suggests that both implementations can bring to a similar performance. However, the hybrid FPGA-photonics training with an averaged power of 7.044 W and an average speed of 440ns per iteration shows an improvement of 1 order of magnitude in energy efficiency. Therefore, training neural networks by exploiting also the photonics domain is feasible and is expected to bring improvements in performance.

Introduction

Recently photonics for neural networks have been developing as a good alternative to electronics. It has been proven that photonics can be used to mimic the brain even better in the field of artificial neural networks than their electrical counterparts [1], [2]. Previously, we have also demonstrated how to implement deep neural networks in photonics through a semiconductor-optical-amplifier based photonic integrated cross-connect [3]. However, the training of the neural networks is mostly performed on conventional electronics and afterwards the weights and biases are mapped to the photonic neural network [4], [5].

In this paper, the research to photonic neural networks will be extended to understand if it is feasible to train neural networks in the photonic domain. For this, two hardware configurations will be implemented. One with conventional electronics and one that is a hybrid alternative between conventional electronics and photonics. The neural network that the two hardware configurations will represent are explained with equations. Then results are shown and discussed. Finally, a conclusion is drawn.

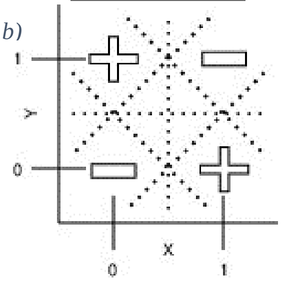
Methods

To understand the feasibility of training neural networks using photonics, a simple neural network is designed and trained on two different configurations of hardware. The first configuration is a neural network on conventional electronics, specifically a field-programmable gate array (FPGA). This configuration is taken as a reference. The configuration is implemented on a HTG-830 FPGA board using Verilog. The second configuration is a neural network that partly consists of the previous implementation on the FPGA, however the matrix multiplication part of the network is replaced with their photonic counterparts. These configurations can be used to compare the impact of the photonic implementations on the neural network.

a)

X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

b)



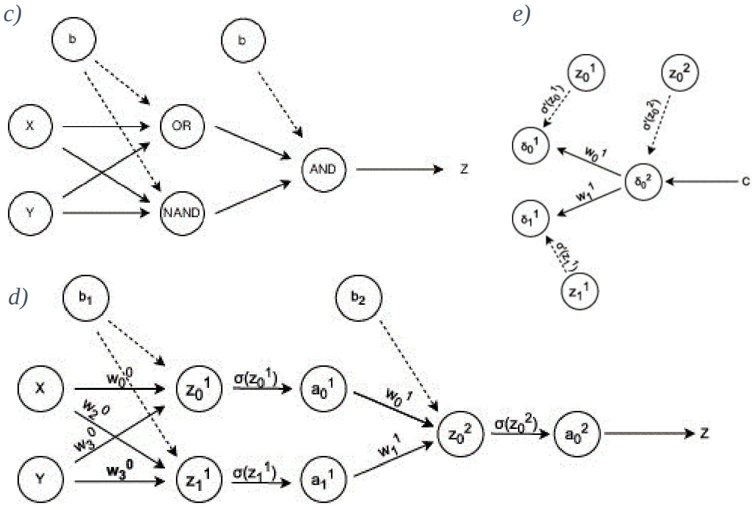


Figure 1 a) The truth table for the XOR operation. b) The two-dimensional representation with decision regions for the XOR operation. c) Schematic overview of the neural network suitable for training the XOR operation d) Schematic overview of the forward propagation of the neural network. e) Schematic overview of the backward propagation of the neural network.

The neural network will be designed to function as an XOR (exclusive OR) operation. Although it is a simple operation, it can prove that the neural network is also able to handle operations that are not linearly separable [6]. The truth table corresponding to the XOR operation can be seen in Fig. 1a where X and Y are the inputs to the neural network and Z will be the output. In Fig. 1b a two-dimensional representation of the XOR operation is illustrated with possible decision regions. The goal of the training of the neural network is to decide what the final decision regions will be. The neural network with the minimum amount of nodes and biases required to train the XOR operation is shown in Fig. 1c. It consists of 2 biases, 2 input nodes, 2 hidden layer nodes and 1 output node. This is also the neural network that is implemented in this work. Fig. 1c also indicates possible operations for successful training, including OR, AND and NAND operations.

The forward propagation of the neural network is built with the following procedure, as presented in Fig. 1d. At first, the inputs are connected to the network. Then from the inputs to the output the relation between the nodes of the different layers is calculated by first determining the weighted input for each layer as: $z^l = w^l a^{l-1} + b^l$ where l is the layer to be calculated, z^n is the weighted input at layer n, w^n is the matrix of the weights of layer n, a^n is the matrix of the activations of layer n and b^n is the matrix of the biases in layer n [7]. The layer outputs are calculated as: $a^l = \sigma(z^l)$ where σ is the activation function. The activation function is there to normalize the weighted input and will be implemented with the sigmoid function.

The backward propagation of the neural network is built with the following procedure, as presented in Fig. 1e. First, the cost is calculated with the difference between the expected value and the output value of the neural network using the mean squared error. This is defined as: $C = \frac{1}{N}\sum_j \left(\bar{e}_j - a_j^L\right)^2$, where C is the cost, N is the amount of output neurons, L is the output layer, a_j^n is the activation at layer n of neuron j and $\bar{e}_j$ is the expected value of output layer neuron j. Then the error at the output is calculated as: $\partial^L = \nabla_a C \odot$

$\sigma'(z^L)$ where ∂^n is the matrix of the errors at layer n. The ∇_a operator is in this case the gradient of C with respecting to a^L defined as a vector whose components are partial derivatives $\frac{\partial C}{\partial a_j^L}$ for every j. The $\odot$ operator is the Hadamard product. The errors of previous layers can be calculated as: $\partial^l = ((w^{l+1})^T \partial^{l+1}) \odot \sigma'(z^L)$ and the rate of change of the cost with respect to the bias is then: $\frac{\partial C}{\partial b^l} = \partial^l$. The rate of change of the cost with respect to the weights is then: $\frac{\partial C}{\partial w^l} = a^{l-1}\partial^l$. At last the weights and biases are updated as: $w^l = w^l - lr\frac{\partial C}{\partial w^l}$ and $b^l = b^l - lr\frac{\partial C}{\partial b^l}$ where lr is the learning rate.
For the Hybrid FPGA-photonics configuration a photonic neuron is proposed as depicted in Fig. 2a. The photonic neuron replaces the neuron calculations in both the forward and backward propagations. The photonic neuron consists of two transceivers, two VOA's (Variable Optical Attenuator's), a MUX (Multiplexer) and an electronic thresholder and is controlled by the FPGA. The transceivers are the inputs and output, the VOA is used as the weights, the MUX is the addition, and the electronic thresholder the bias.
To measure the performance of the networks the accuracy and loss are used. The accuracy is used to determine the percentage of correct predictions. The accuracy of the neural network is calculated as: $accuracy = \frac{No.\ correct\ predictions}{No.\ predictions} \times 100\%$. The loss is used to determine the difference between the expected value and the activation level at the output. The loss of the neural network is defined as: $loss = \frac{1}{N}\sum_j |\bar{e}_j - a_j^L|$.

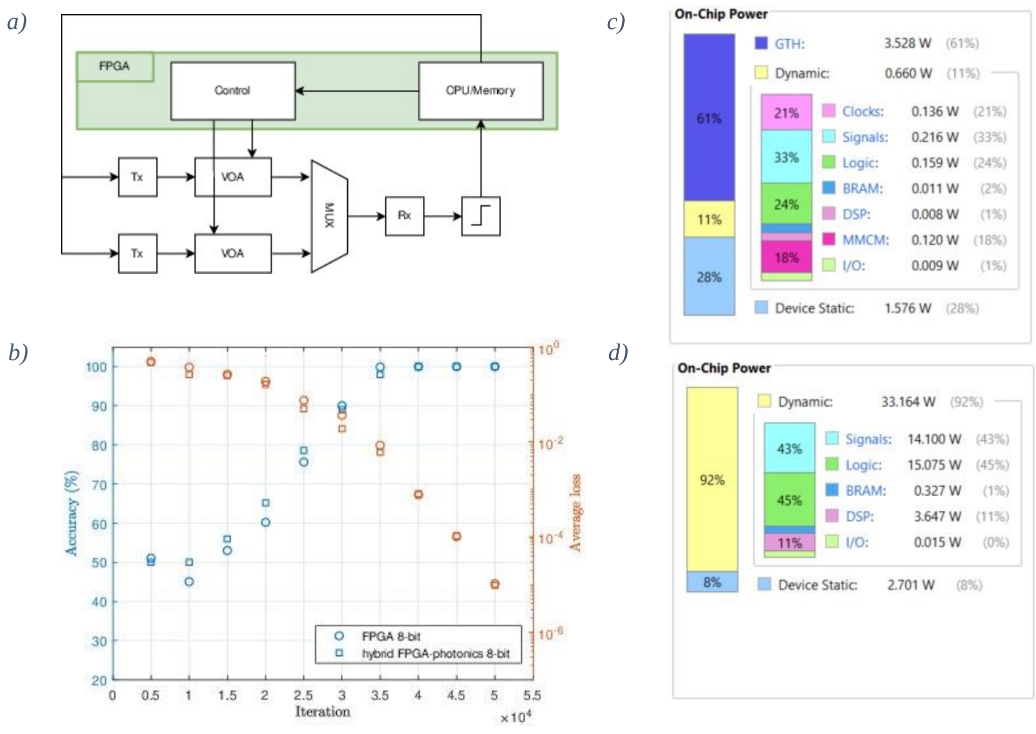


Figure 2 a) Possible implementation of a neuron with photonic components. b) The average loss and accuracy of the testing data on the FPGA configuration and the hybrid FPGA-photonics configuration neural network implementation per iteration for 8-bit decimal data precision. c) The average on-chip power on the FPGA board for training the neural network on the hybrid FPGA-photonics configuration. d) The average on-chip power on the FPGA board for training the neural network on the FPGA configuration.

Results

The average loss and accuracy for the FPGA and hybrid FPGA-photonic configuration is presented in Fig. 2b. In Fig. 2b the trend for the loss is that it keeps declining for more iterations. The trend for the accuracy is that is keeps improving for more iterations. Both FPGA only and hybrid FPGA-photonic neural networks converge to 100% accuracy after 3500 iterations of training And the loss of FPGA network is decreasing to $42 \cdot 10^{-5}$ while the loss of hybrid FPGA-photonic network is decreasing to $39 \cdot 10^{-5}$. For the hybrid FPGA-photonic configuration the power is presented in Fig. 2c and the average power is 7.044W. For the FPGA configuration the power is presented in Fig. 2d and the average power is 35.865W. For the FPGA configuration, the time per iteration is 918 ns and for the hybrid FPGA-photonics configuration, the time per iteration is 440ns.

Conclusion and Discussion

As can be seen in Fig. 2b both configuration have around the same loss and accuracy. Both are able to solve the XOR operation. The main difference between the neural networks are for the power and time per iteration. For the FPGA configuration, the time per iteration is 918 ns and for the hybrid FPGA-photonics configuration, the time per iteration is 440ns. Thus, the hybrid FPGA-photonics configuration has a time advantage of around 478ns. The average power of the FPGA configuration is also higher. The FPGA configuration has an average power of 35.865 W while the hybrid FPGA-photonics configuration has an average power of 7.044 W. This suggest that an improvement of 1 order of magnitude in energy efficiency is achieved via with photonics assisted approach. The power difference is due to the fact that the FPGA configuration has a lot of dynamic power that is mostly converted for the hybrid FPGA-photonics configuration to power for the transceiver and VOA that use less power. This shows that it is feasible to train neural networks in the photonic domain.

References

[1] B. Bai, H. Shu, X. Wang en W. Zou, „Towards silicon photonic neural networks for artificial intelligence," *Science China Information Sciences,* vol. 63, 2020.

[2] N. Passalis, G. Mourgias-Alexandris, N. Pleros en A. Tefas, „Initializing photonic feed-forward neural networks using auxiliary tasks," *Neural Networks,* vol. 129, pp. 103-108, 2020.

[3] B. Shi, N. Calabretta and R. Stabile, *Deep neural network through an InP SOA-based photonic integrated cross-connect,* 2020, 2020.

[4] Y. Shen, N. C. Harris, S. Skirlo, M. Prabhu, T. Baehr-Jones, M. Hochberg, X. Sun, S. Zhao, H. Larochelle, D. Englund en M. Soljačić, „Deep learning with coherent nanophotonic circuits," *Nature Photonics,* vol. 11, pp. 441-446, 2017.

[5] T. Zhang, J. Wang, Y. Dan, Y. Lanqiu, J. Dai, X. Han, X. Sun en K. Xu, „Efficient training and design of photonic neural network through neuroevolution.," *Optics express,* vol. 27, pp. 37150-37163, 2019.

[6] R. Beale and T. Jackson, Neural computing : an introduction, Bristol: Hilger, 1990.

[7] M. A. Nielsen, Neural Networks and Deep Learning, Determination Press, 2015.