LASER & PHOTONICS REVIEWS
www.lpr-journal.org

# Physics-Aware Analytic-Gradient Training of Photonic Neural Networks

*Yuancheng Zhan, Hui Zhang,* Hexiang Lin, Lip Ket Chin, Hong Cai, Muhammad Faeyz Karim, Daniel Puiu Poenar, Xudong Jiang, Man-Wai Mak, Leong Chuan Kwek, and Ai Qun Liu**

**Photonic neural networks (PNNs) have emerged as promising alternatives to traditional electronic neural networks. However, the training of PNNs, especially the chip implementation of analytic gradient descent algorithms that are recognized as highly efficient in traditional practice, remains a major challenge because physical systems are not differentiable. Although training methods such as gradient-free and numerical gradient methods are proposed, they suffer from excessive measurements and limited scalability. State-of-the-art in situ training method is also cost-challenged, requiring expensive in-line monitors and frequent optical I/O switching. Here, a physics-aware analytic-gradient training (PAGT) method is proposed that calculates the analytic gradient in a divide-and-conquer strategy, overcoming the difficulty induced by chip non-differentiability in the training of PNNs. Multiple training cases, especially a generative adversarial network, are implemented on-chip, achieving a significant reduction in time consumption (from 31 h to 62 min) and a fourfold reduction in energy consumption, compared to the in situ method. The results provide low-cost, practical, and accelerated solutions for training hybrid photonic-digital electronic neural networks.**

## 1. Introduction

Photonic Neural Networks[1–5] exploit the high connectivity and parallelism of optics for efficient information processing with reduced energy and latency consumption. However, training PNNs networks[6,7] is challenging due to the resource-intensive and time-consuming nature of gradient computation. Consequently, in the early stage, many PNN implementations[8–12] only support inference computation with weights obtained via off-chip training, resulting in low prediction accuracy and susceptibility to noise.

On-chip (online) PNN training can improve prediction accuracy, and it is classified into two main categories: gradient-free and numerical-gradient-based. Gradient-free methods employ non-gradient search algorithms, such as genetic algorithms,[13,14] biologically inspired method[15] and particle swarm optimization[16] to obtain optimal solutions. While these approaches yield high accuracy in simple classification tasks, their computation complexity increases significantly with the number of trainable parameters, restricting their scalability for complicated machine-learning tasks. Numerical gradient methods, on the other hand, measure the gradients through finite

Y. Zhan, H. Zhang, H. Lin, D. P. Poenar, L. C. Kwek, A. Q. Liu
Quantum Science and Engineering Centre (QSec)
Nanyang Technological University
Singapore 639798, Singapore
E-mail: zh0012ui@e.ntu.edu.sg; eaqliu@ntu.edu.sg

H. Zhang, A. Q. Liu
Institute of Quantum Technology (IQT)
The Hong Kong Polytechnic University
Hong Kong 999077, Hong Kong

L. K. Chin
Department of Electrical Engineering
City University of Hong Kong
Hong Kong 999077, Hong Kong

H. Cai
Institute of Microelectronics
A*STAR
Singapore 138634, Singapore

M. F. Karim, X. Jiang
School of Electrical & Electronic Engineering
Nanyang Technological University
Singapore 639798, Singapore

M.-W. Mak
Department of Electronic and Information Engineering
The Hong Kong Polytechnic University
Hong Kong 999077, Hong Kong

L. C. Kwek
Centre for Quantum Technologies
National University of Singapore
Singapore 117543, Singapore

The ORCID identification number(s) for the author(s) of this article can be found under https://doi.org/10.1002/lpor.202300445



**DOI: 10.1002/lpor.202300445**

difference,[17–20] causing prohibitive complexity and susceptibility to noise.

Therefore, analytic gradient methods[21–24] have long been desired for the training of photonic neural networks for their fast convergence and low measurement complexity but they are hindered by the non-differentiable properties[25] of photonic chips. An in situ training method[26,27] was proposed and recently demonstrated in a $4 \times 4$ PNN chip, but it requires extensive digital-to-analog conversion, additional infra-red (IR) cameras as the monitors on each Mach-Zehnder interferometer (MZI), and the frequent switching of bidirectional optical I/O between forward- and backward-propagating signals. When the chip scales to practical size, the above requirements will greatly increase the energy consumption and lengthen the time for camera imaging and I/O switching.

In this paper, we propose a physics-aware analytic-gradient training (PAGT) method and demonstrate it on an $8 \times 8$ PNN chip. The PAGT method uses the PNN, which is a hybrid photonic-digital electronic neural network in our hardware setting, to perform the forward pass and a pre-established digital neural network (DNN) to calculate the analytic gradient as the backward pass. The DNN model is obtained by learning the unique physical transformations of the PNN. The forward pass performed by the PNN eases the burden of requiring the differential DNN model to be exceptionally accurate and inherently mitigates the unique noise processes and imperfections. The differentiable DNN model is only utilized in the backward pass to supplement parts of the training loop that the physical system cannot perform.

From the on-chip demonstration, our PAGT method takes approximately 62 min (52 min for the DNN and 10 mins for the PNN) for the whole procedure, in contrast to the 31 h duration required by the in situ method[27] (mostly caused by the camera imaging). Specifically, once the DNN is obtained, it can be applied to any task, thus the DNN training is a one-time consumption. The energy consumption of PAGT method is 62 J, which is significantly lower than 280 J for the in situ method.[27] From the analysis of the scalability, our PAGT retains its advantages on energy and time consumption for most of the current PNNs. Besides, in some special network architectures that require the derivatives of hybrid photonic-digital models,[28,29] unifying the non-differentiable photonic part and the derivatives of the differential digital part into analytic derivatives will greatly accelerate the training speed and reduce training costs. Using a differentiable DNN to represent the non-differentiable PNN enables our method applicable to hybrid networks like Generative Adversarial Networks (GAN),[30] while most other methods are not unified. Our approach demonstrates fast and efficient photonic neural network training and exhibits broad potential in complicated network architectures and cascaded PNNs.

## 2. Framework

We design a hybrid photonic-digital electronic neural network chip (**Figure 1**a) incorporated with the PAGT method (Figure 1b). The training algorithm comprises two training phases: the differentiable DNN training and then the on-chip PNN training. The aim of DNN training is to build a differentiable model to characterize the unique physical transformation of the PNN. This DNN model is used to calculate the analytical gradient and perform the backward propagation, in lieu of the photonic chip. Then, the PNN is used to perform the forward propagation and updates the free parameters according to the analytic gradients acquired from the DNN. The details of both training phases are as follows.

### 2.1. DNN Training

The structure of differentiable DNN is shown in Figure 1c. The training goal is to attain output consistency between the PNN $f_p$ and the DNN $f_d$ for arbitrary inputs, so as to mimic the PNN accurately. The PNN is denoted by $\boldsymbol{\gamma}_\mathrm{p} = f_d(\boldsymbol{\varphi}, \boldsymbol{\theta}_p)$, containing a data-uploading circuit controlled by parameters $\boldsymbol{\varphi}$ and a variational circuit controlled by parameters $\boldsymbol{\theta}_p$. The differentiable DNN is defined as $\boldsymbol{\gamma}_\mathrm{d} = f_d(\{\boldsymbol{\varphi}, \boldsymbol{\theta}_p\}, \boldsymbol{\theta}_d)$, where we amalgamate the $\boldsymbol{\varphi}$ and $\boldsymbol{\theta}_p$ as the DNN input dataset, and $\boldsymbol{\theta}_d$ represents the training parameters in DNN.

In the beginning, we sample the $\boldsymbol{\varphi}$ and $\boldsymbol{\theta}_p$ from a random dataset. Randomization is employed to enhance the generalization capability of the DNN being trained. Then, we feed them into the PNN implemented by the photonic chip. The $\boldsymbol{\gamma}_\mathrm{p}$ is obtained from the photodetector measurements of the chip. The sampled dataset and measurements are then labeled as inputs and labels, respectively, and used to train the differentiable DNN. In this work, we explore two classical neural network structures as differentiable DNNs: Convolutional Neural Network (CNN) and Multi-Layer Perceptron (MLP). The loss function $L$ is defined as the kullback-leibler divergence loss (KLDiv) between the DNN outputs and the PNN outputs, $L(\boldsymbol{\gamma}_\mathrm{d} \| \boldsymbol{\gamma}_\mathrm{p}) = \boldsymbol{\gamma}_\mathrm{p} \cdot \ln \frac{\boldsymbol{\gamma}_\mathrm{p}}{\boldsymbol{\gamma}_\mathrm{d}}$. The DNN is trained on a digital computer and employed as a constant $\boldsymbol{\theta}_d$ pre-trained model in the PNN training.

### 2.2. PNN Training

The training aims to produce the desired output for a given input in any given task using both PNN and DNN, where PNN handles complex forward computations and backpropagation gradients are computed via a pre-trained DNN.

During the forward propagation stage, the dataset is initially encoded into a discrete form, denoted as $\boldsymbol{\varphi}^0$. Simultaneously, a random set is generated to serve as the initial training parameters $\boldsymbol{\theta}_p$ for the PNN, and the pre-trained DNN is incorporated into the system. Assuming we have a PNN with $K$ layers, the PNN expression in the $l$-th layer is defined as

$$\boldsymbol{\gamma}_\mathrm{p}^l = \boldsymbol{\varphi}^{l+1} = f_p^l(\boldsymbol{\varphi}^l, \boldsymbol{\theta}_p^l),\ l = 0, 1, \ldots, K-1 \tag{1}$$

and the corresponding DNN for each layer is defined as

$$\boldsymbol{\gamma}_\mathrm{d}^l = f_d^l(\{\boldsymbol{\varphi}^l, \boldsymbol{\theta}_p^l\}, \boldsymbol{\theta}_d^l) = f_d^l(\boldsymbol{\varphi}^l, \boldsymbol{\theta}_p^l),\ l = 0, 1, \ldots, K-1 \tag{2}$$

In the above equation, $\boldsymbol{\theta}_d$ are not required in training as they're fixed values. The output of the last layer of PNN, $\boldsymbol{\gamma}_p^{K-1}$, is imported into the digital computer with photodetectors.

During the backward propagation stage, the analytic gradient of each parameter is obtained by calculating the derivative of the loss function. Denoting the real gradients in the PNN as $g_{\theta_p}^p$, we

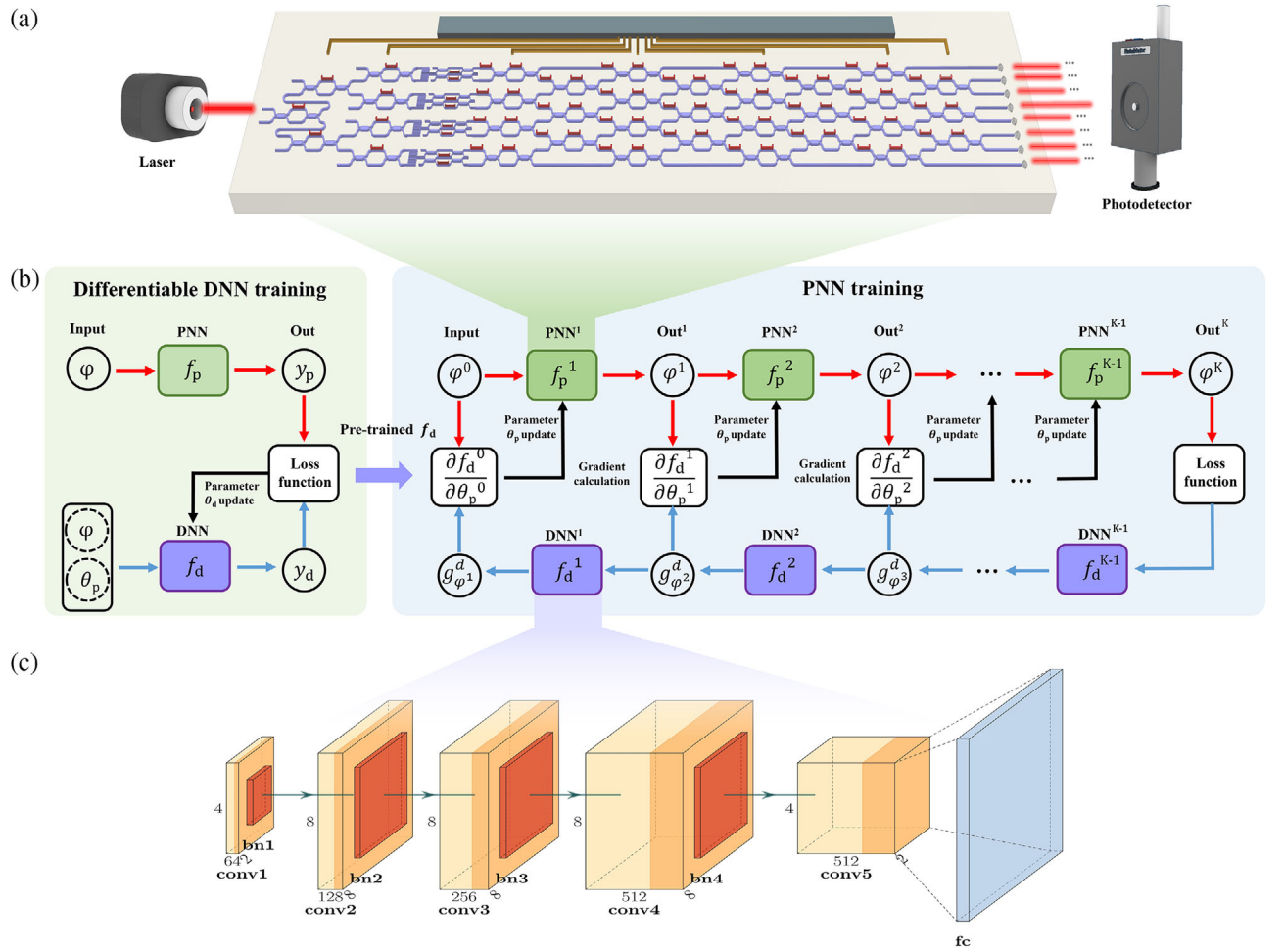


**Figure 1.** Hybrid Photonic-Digital Electronic Neural Network with PAGT. a) The 8-mode photonic neural network chip $f_p$, comprising a laser generating light, a data-uploading circuit for encoding light into input states, a variational circuit for performing photonic calculations, and eight photodetectors for measuring output photons. b) The flowchart of the PAGT method, consisting of differentiable DNN training and PNN training. The PNN training section illustrates the training methodology for an *l*-layer PNN, while the DNN training section illustrates the pretraining of $f_d$ for each layer of the PNN. c) The differentiable digital neural network $f_d$, containing five convolutional layers (conv), five batch normalization layers (bn), and a fully-connected layer (fc).

propose using the $f_d$ to perform the derivative operation and substitute $g^p_{\theta_p}$ with $g^d_{\theta_p}$. Utilizing the backpropagation algorithm, the analytical gradient of the system can be expressed as

$$g^d_{\varphi^l} = \frac{\partial L}{\partial \boldsymbol{\varphi}^l} = g^d_{\varphi^{l+1}} \frac{\partial f^l_d}{\partial \boldsymbol{\varphi}^l} \tag{3a}$$

$$g^d_{\theta^l_p} = \frac{\partial L}{\partial \boldsymbol{\theta}^l_p} = g^d_{\varphi^{l+1}} \frac{\partial f^l_d}{\partial \boldsymbol{\theta}^l_p} \tag{3b}$$

where $g^d_{\varphi^l}$ are the gradients with respect to $\boldsymbol{\varphi}^l$ as the intermediate variables in the calculation. This equation indicates that we can determine the outermost gradient ($g^d_{\varphi^K}$) first, and then use mathematical induction to derive the subsequent inner gradient ($g^d_{\varphi^l}$ and $g^d_{\theta^l_p}$). We can obtain the $g^d_{\varphi^K}$ directly from KLDiv loss as $g^d_{\varphi^K} = \frac{\partial L}{\partial \varphi^K} = \frac{\partial L(\gamma_{\text{label}} \| \gamma^{K-1}_p)}{\partial \gamma^{K-1}_p}$, where $\gamma_{\text{label}}$ represent labels for the specific task. Furthermore, the intermediate gradients $\frac{\partial \gamma_d}{\partial \varphi}$ and $\frac{\partial \gamma_d}{\partial \theta_p}$ can be obtained from the Jacobi Matrix. The parameters are updated by the analytic gradients at a certain learning rate. The training process of the PNN involves iteratively performing forward propagation and backward propagation until the stopping criteria (i.e., specified maximum iterations or loss function values) are met.

The PAGT method of training PNN enables the update of all parameters through a single measurement with a measurement complexity independent of the number of parameters, so as to ensure the computational efficiency of complicated problems and cascaded PNNs. Furthermore, due to its gradient-based nature, it exhibits superior overall efficiency in finding optimal solutions compared to non-gradient algorithms. Our approach to training hybrid photon-digital systems represents the non-differentiable PNN with a differentiable function. This transformation effectively converts the gradient training problem into a purely digital system, thus simplifying the training process for hybrid systems.

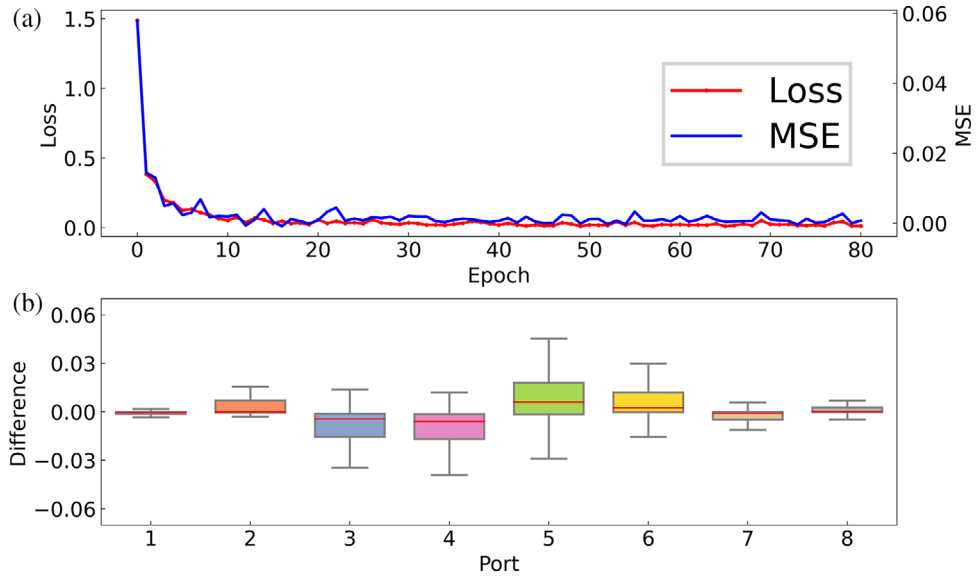


**Figure 2.** Results of training the differentiable DNN. Using identical input for both differentiable DNN and PNN, we measure PNN output as labels and DNN output as predictions. a) The average KLDiv loss and MSE between the DNN predictions and the PNN labels, which converges within 50 epochs. b) The statistical distribution of the differences between trained DNN and PNN, showing the goodness-of-fit of the trained model.

## 3. Results and Discussion

The proposed PAGT method was applied to the PNN with a convolution-based DNN. **Figure** 2a shows the average KLDiv loss and mean squared error (MSE) against the training iterations. Figure 2b shows the output discrepancy between trained the DNN and the PNN. This result demonstrates that the DNN achieves highly accurate simulations of PNNs in a short training time of about 52 min. The trained PNN was then employed for three practical tasks, i.e., the construction of unitary matrices, the classification of the Iris Flower dataset, and the training of a hybrid photonic-digital GAN architecture. The scalability of the PAGT method is discussed. A comparative analysis of our algorithm with other state-of-the-art methods is conducted in terms of performance and computational complexity at the end.

### 3.1. Construction of the Unitary Matrix

This task validates the accuracy of the constructed PNN and demonstrates its adaptability and efficiency without the need for MZI calibration or matrix decomposition in advance. The photonic circuit to construct the matrix uses the rectangular structure[31] of universal multiport interferometers.

The PNN training begins by generating a random set of $\boldsymbol{\theta}$ for the PNN and importing the pre-trained DNN model into the system. The matrix $U^6$ is used to represent the PNN $f_p$, where the superscript 6 indicates that the dimension of $U$ is 6 and $U^6$ is generated randomly during the experiment. $\boldsymbol{x}_{\text{in}}$ and $\boldsymbol{x}_{\text{out}}$ are defined as the input and output light distribution with $\boldsymbol{x}_{\text{out}} = U^6(\boldsymbol{\theta})\boldsymbol{x}_{\text{in}}$. To construct an accurate matrix, independent groups of bases were fed into the chip, and the corresponding outputs were measured. The different sets of inputs $x_{\text{in}}^i$ are grouped, where each $i$ represents light input from port $i$ (e.g., $x_{\text{in}}^2 = [0, 1, 0, 0, 0, 0, 0, ]$). Then, we measure the corresponding $x_{\text{out}}^i$ (which represents the $i_{th}$-row vectors of the target $U^6$) to form the output $\boldsymbol{x}_{\text{out}} = [x_{\text{out}}^1, \ldots, x_{\text{out}}^6]$.

The comparison between the chip-measured output distribution and the expected labels is shown in **Figure** 3a–c, which illustrates that the distribution generated by the PNN at the beginning of the training is completely random. While the PNN converges, all six $x_{\text{out}}^i$ converge to the target vector, demonstrating the capability of our method in constructing any desired unitary matrix. Figure 3d shows the average KLDiv and MSE against the training epoch. The PNN training converges within 40 epochs, with both KLDiv and MSE converging. This demonstrates the efficiency and effectiveness of the proposed PAGT method.

Compared to the traditional method of constructing a matrix on the chip, where the MZI is pre-calibrated and the $\boldsymbol{\theta}_p$ value of each MZI is calculated by a computer and transferred to the chip, our method eliminates the need for a computer to decompose the unit matrix to calculate $\boldsymbol{\theta}_p$ and also eliminates the errors arising from inaccurate MZI calibrations.

### 3.2. Iris Flower Classification

We evaluate the performance of our algorithm in classifying the Iris Flower dataset, a widely-used benchmark for low-dimensional neural networks. We use a subset of the iris dataset, comprising 100 samples of two types of flowers, the Setosa and the Versicolor. Each flower species has four characteristic features. The PNN architecture utilizes a 4-mode photonic chip with encoding and tunable parameters $\boldsymbol{\varphi}$ and $\boldsymbol{\theta}_p$, respectively. The DNN is pre-trained in advance. The forward propagation of the PNN is expressed as $\boldsymbol{y}_{\text{iris}} = f_p(\boldsymbol{\varphi}, \boldsymbol{\theta}_p)$. For data encoding, we normalize the iris data and map it to $\boldsymbol{\varphi}$, and utilize one-hot encoding to represent the iris labels. For data decoding, we apply the nearest-neighbor principle, whereby the output is classified as Setosa if most output photons come from the Setosa port ($|\boldsymbol{y}_{\text{iris}} - \boldsymbol{y}_{\text{setosa}}| < |\boldsymbol{y}_{\text{iris}} - \boldsymbol{y}_{\text{versi}}|$), and vice versa. To train the PNN to perform the classification task, we used the average gradient

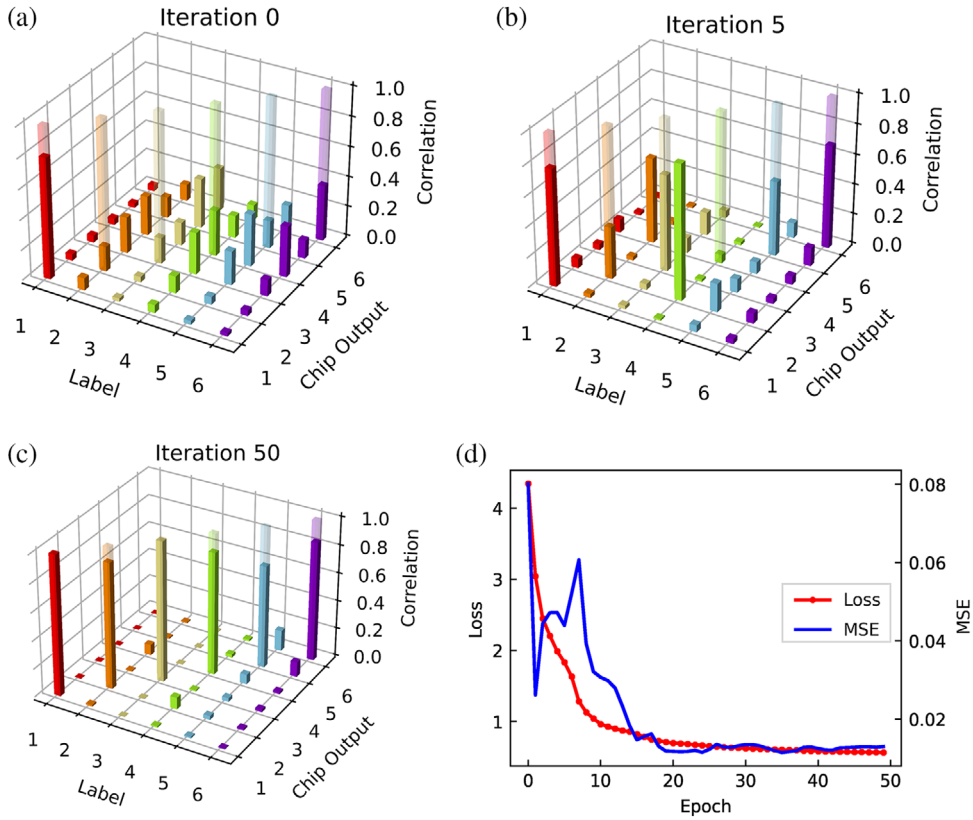


**Figure 3.** Results of Unitary Matrix Construction. a–c) The comparison between the chip-measured output distribution (solid bar) and the expected labels (transparent bar), at Iteration of 0, 5, and 50, respectively. d) The average KLDiv loss and MSE as a function of training epochs. The PNN training converges within approximately 40 epochs.

to update the network parameters. Our approach involves randomly selecting a batch of ten data points (the batch size for conventional approaches is one), calculating the average gradient, and performing a global update with the PAGT method. This approach helps to stabilize the training and ensures a smoother convergence toward the optimal solution.

**Figure** 4a displays the MSE of the training process, indicating that the PAGT method efficiently achieves convergence for the PNN within 35 epochs. Figure 4b presents the decision boundaries produced by the trained PNN on a 2D projection, which illustrates the effectiveness of our training strategy for data categorization. To assess different pre-trained differentiable DNNs on the PNN training, we selected three models: a CNN trained with sufficient random data (pagt-CNN), an MLP trained with smaller random data (pagt-MLP), and a CNN trained with the minimal dataset of the iris flower (pagt-irisdata), as shown in Figure 4c. Our results demonstrate that pre-trained CNN models performed best as a surrogate approximator in terms of accuracy and convergence speed when DNN trained on large amounts of data. However, for specific computation tasks, the general modeling of the chip is not always necessary. For example, for iris classification, using its dataset instead of a random sampling dataset will reduce the pre-training period of DNNs while maintaining the effectiveness of PNN training. Meanwhile, for simple tasks, training an MLP requires fewer samples, making it an economical choice for faster and more accurate training. These results provide greater flexibility in choosing an appropriate model for different task requirements. In Figure 4d, we compare the PAGT method with two representative on-chip training methods, the numerical gradient training[17] and gradient-free training,[2] under the same experimental conditions in our photonic platform. In terms of training accuracy, numerical gradient training achieves 80%, while gradient-free training improves the accuracy by 15%, and our proposed PAGT method shows a superior enhancement of 20%.

### 3.3. Hybrid Generative Adversarial Networks

We constructed a generative model using a hybrid GAN framework, as shown in **Figure** 5a. The objective of this study is to apply our PAGT to a hybrid photonic-digital system. Hybrid training necessitates the concurrent computation of on-chip PNN gradients and off-chip DNN gradients, a task that proves challenging for existing chip-only or off-chip training methodologies. Nevertheless, this issue can be addressed by utilizing analytic gradients through our PAGT method. Here, we used the GAN network as an example, which involves a generator network in generating distributions, and a discriminator network in distinguishing the validity of distributions.

In our experimental setup, we employed an 8-mode PNN $f_p(\boldsymbol{\varphi}, \boldsymbol{\theta}_p)$, trained with the PAGT method as the generator, and

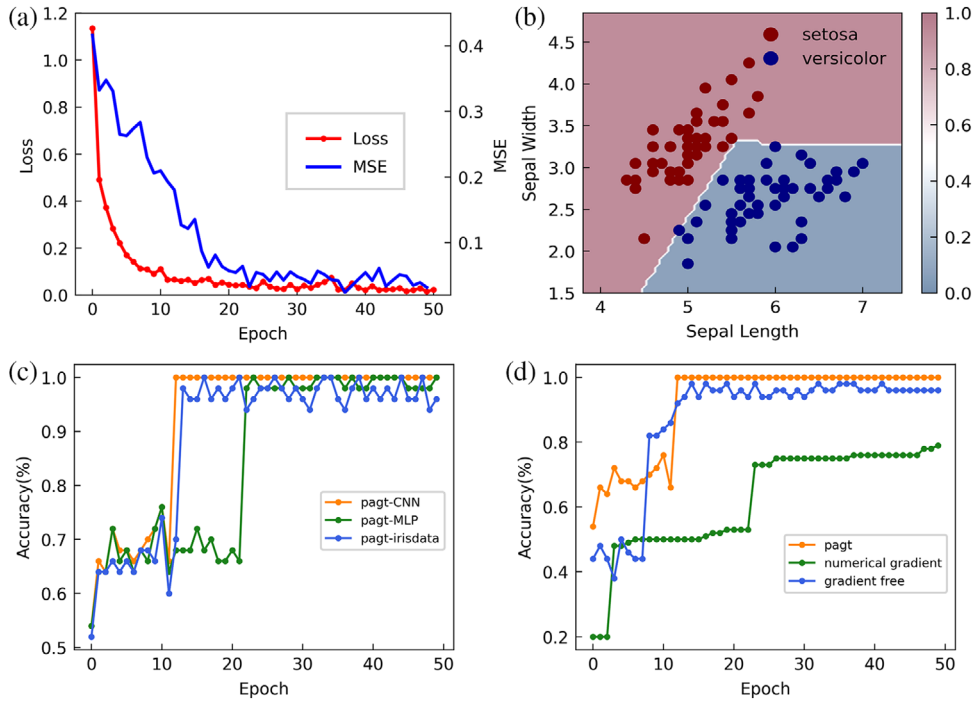


**Figure 4.** Results of Iris Flower Classification. a) The average KLDiv loss and MSE of the training process, demonstrating convergence within 30 epochs. b) Decision boundaries of the iris data, projecting the 4D data into the sepal width and sepal length dimensions, showcasing the effectiveness of PNN in accurately separating the data. Similar results can be obtained for other projections. c) Accuracy during the training process using different pre-trained DNN models. d) Accuracy during the training process for various training methods.

a four-layer CNN, denoted by $D_\tau$, trained with a traditional gradient descent algorithm as the discriminator, where $\boldsymbol{\theta}_p$ and $\boldsymbol{\tau}$ are trainable parameters generated randomly as initial. Light is incident to one input port of the chip, and generates eight-bin programmable light distribution at the output ports. The generator output $\boldsymbol{\gamma}_p$ and the real probability distribution $\boldsymbol{\gamma}$ are then fed into the discriminator as input. Using the Wasserstein GAN loss,[32] we can obtain the generator and discriminator losses, $L_G(\boldsymbol{\theta}_p, \boldsymbol{\tau})$ and $L_D(\boldsymbol{\theta}_p, \boldsymbol{\tau})$, respectively,

$$L_G(\boldsymbol{\theta}_p, \boldsymbol{\tau}) = \frac{1}{N}\sum_{i=1}^{N} D_\tau\left(f_p(\boldsymbol{\varphi}_i, \boldsymbol{\theta}_p)\right)$$
$$L_D(\boldsymbol{\theta}_p, \boldsymbol{\tau}) = \frac{1}{N}\sum_{i=1}^{N}\left[D_\tau(\boldsymbol{\gamma}_i) - D_\tau\left(f_p(\boldsymbol{\varphi}_i, \boldsymbol{\theta}_p)\right)\right] \quad (4)$$

where $i$ represents the $i$-th in the input dataset. Using Equation (3), we calculate the gradients of the generator with $L_G$ and $f_d$ and then update $\boldsymbol{\theta}_p$. We also update $\boldsymbol{\tau}$ with $L_D$ and $f_d$. These two steps are repeated until both networks converge. The advantages of the proposed PAGT training method for hybrid systems are demonstrated in this experiment. Specifically, for the generator (PNN), we directly obtain the gradient of it using our algorithm, while for the discriminator, we derive its loss function (e.g., Equation (4)) involving both generator and discriminator networks simultaneously and calculate the gradient. Our PAGT method addresses this process by deriving the analytic gradient expression of the generator and incorporating it into the discriminator gradient formula, allowing for simultaneous derivation in a digital computer. In contrast, gradient-free methods that cannot derive the gradient face difficulties in updating the discriminator. Numerical gradient and other analytical gradient methods, where the gradient is not an expression, require resetting the experimental setup to calculate the gradient once in each formula that contains the generator gradient, leading to a significant hindrance in training speed.

We used the PAGT method to generate three different distributions by PNN: Normal Distribution, Poisson Distribution, and Lognormal Distribution. The performance of the PNN is compared against the real distributions in Figure 5b–d. The accuracy of the PNN in generating the desired distributions is evident from the near equivalence of the probability values in the final epoch. The training process of the generator loss and discriminator loss is depicted in Figure 5e–g, while the training process of MSE is shown in Figure 5h. The loss value of $L_G$ and $L_D$ approach 0, which is in accordance with theoretical values, indicating good convergence of the PNN. Furthermore, the MSE reaches below -48 dB within 50 epochs, which demonstrates the fast convergence of our algorithm.

### 3.4. Scalability of the PAGT Method

As the PNN dimension increases, the PAGT method requires a substantial number of training samples to ensure that DNN can accurately approximate the underlying PNN. Here, we discuss the requirements imposed on the necessary volume of training

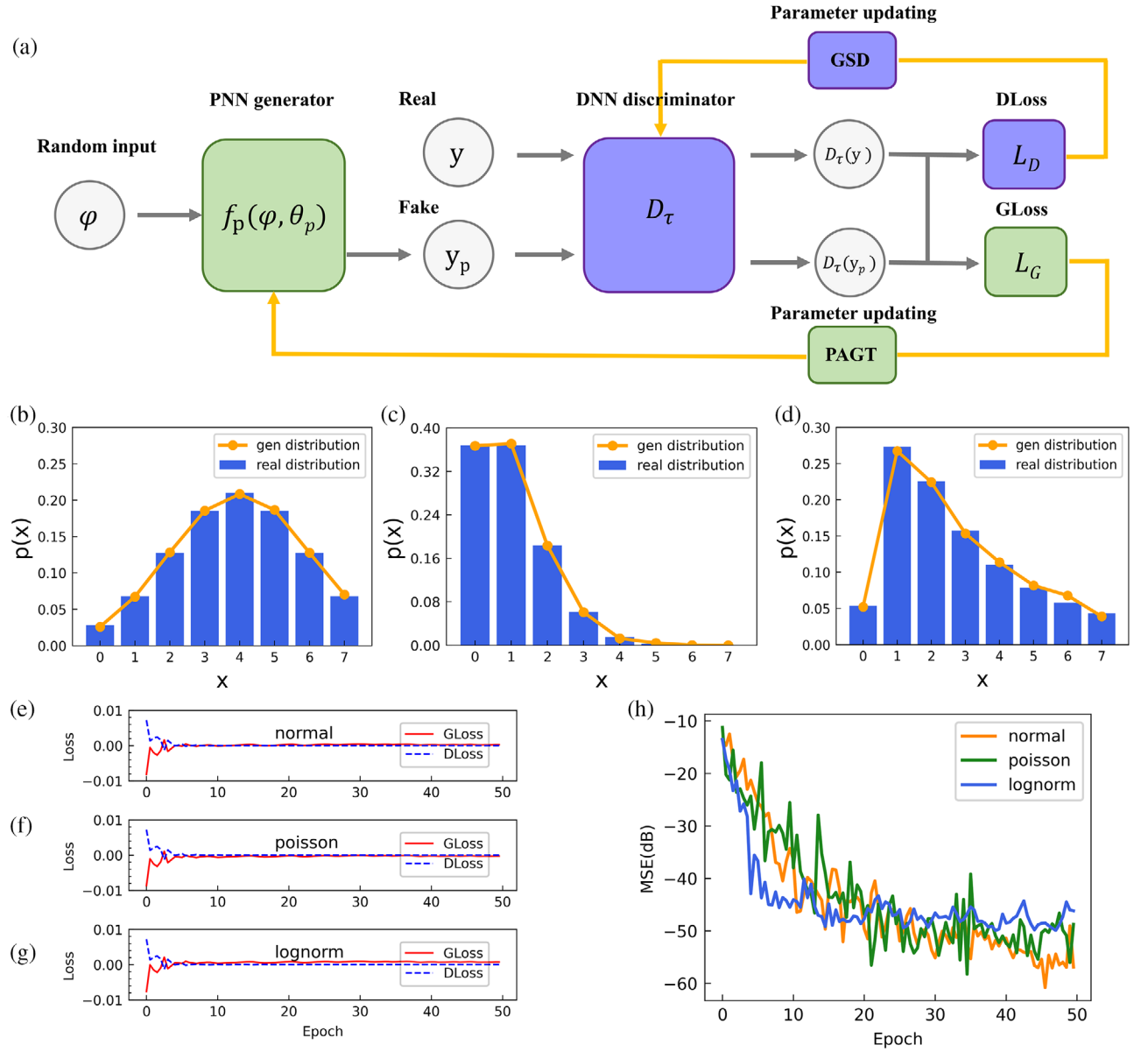


**Figure 5.** Results of Hybrid GAN. a) The flow chart of Hybrid GAN Training. b) A comparison between the PNN-generated distribution and the real distribution for Normal Distribution. c) The Poisson Distribution case. d) The Lognormal Distribution case. e–g) The generator loss ($L_G$) and discriminator loss ($L_D$) during training for Normal, Poisson, and Lognormal Distributions, respectively, showcasing convergence around 20 epochs. h) The MSE of three distributions during training, with all MSE values converging around 50 epochs and reaching below -48dB.

sample for a surrogate DNN when the PNN dimension increases, thereby identifying the optimal chip dimension to which the PAGT method is applicable. The Iris classification and MNIST classification tasks are used as examples to evaluate the PAGT method. Numerical experiments are conducted in the following three scenarios: 1) For an 8-mode PNN, we investigated the volume of training data required to obtain a surrogate DNN. The performance (i.e., accuracy and MSE) when training data volume varies from 0.2$k$ to 625$k$ are shown in **Figure** 6a,b. For an 8-mode PNN, 5$k$ samples are sufficient to train a surrogate DNN without suffering from underfitting issues, thus ensuring its capability to handle various complex tasks. 2) Given the training data volume, we investigate the maximum PNN dimension that can be surrogated. We studied PNN dimensions ranging from 4 to 24 modes, which correspond to 12 to 552 training parameters. As shown in Figure 6c, a data volume of 5$k$ samples is sufficient to train an 8-mode PNN (dotted line), while a volume of 15 million is required to train a 20-mode PNN (solid line). 3) Finally, we discuss the general trend of the required training data volume as the PNN dimension increases. As shown in Figure 6d, the required training data volume $D_{DNN}$ exhibits an exponential relationship with the PNN dimension and can be fitted by $D_{DNN} = a \times b^N$, where $a = 13.7$ and $b = 1.97$ obtained from fitting, and $N$ represents the PNN dimension. Our method demonstrates high

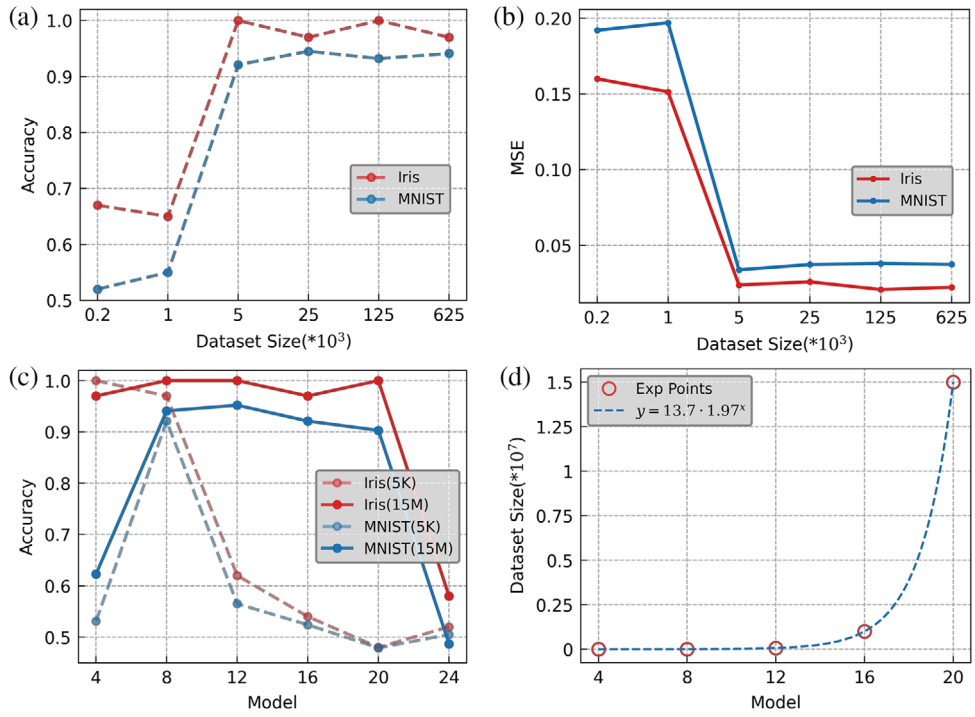


**Figure 6.** Scalability of the PAGT method. a,b) The accuracy and MSE of the PAGT method on an 8-mode PNN chip when the training data volume varies from 0.2*k* to 625*k*. c) The training accuracy of different PNN dimensions, when training data volumes of 5*k* and 15*M* samples are given, respectively. d) Relationship between PNN dimension and the required training data volume. The data points shown in the graph are the training data volumes needed to achieve a 90% accuracy on the MNIST dataset (excluding $N = 4$, which is on the Iris dataset).

efficiency for PNN dimensions below 20 modes, as the required training data volume can be controlled below 15 million, which remains advantageous over other methods (see the section "Comparison to Other Training Methods"). However, when N exceeds 20, the significantly increased data requirement leads to high energy consumption and latency and poses challenges to maintaining accuracy when the data requirement cannot be fulfilled. In summary, a training dataset with one million samples is adequate to support a PNN with up to 16 modes, which meets the requirements of the current largest-scale programmable photonic integrated neural networks.[33,34] Additionally, cascading PNNs provide a feasible approach to enhance PNN scalability. Our method offers a potential strategy for training cascading PNNs by managing complex parameter interactions and mitigating cumulative noise errors (see details in Note S7, Supporting Information).

### 3.5. Comparison to Other Training Methods

We present a comparative analysis of various on-chip training methods for photonic chips, including our proposed method, gradient-free method,[2] numerical gradient method,[17] and another analytic gradient method (in situ method).[27] The comparison results of most methods are based on our experimental platform conditions, except for in situ method, which requires some additional devices. We compare these methods based on two key performance metrics, energy consumption, which refers to the total energy used by the chip throughout the entire training process, is calculated by multiplying the energy used in a single training epoch by the total epochs, and latency, the time consumption of operations and computations. Consumptions of other devices (e.g., lasers, digital computers, etc.) are not compared here due to different experimental devices and configurations across different methods. We assume that each MZI on our chip consumes power $P = 4.8$ mW, the operation and effect time of the thermo-optic phase modulator for each MZI is $T_{MZI} = 1$ s, and the operation time of analog-to-digital conversion and updating parameters in the digital computer is $T_{DIG} = 2$ s. The time required for each optical computation, from the photon entering the chip to being detected by the photodetectors, can be ignored (<4 us).

Each layer of the photonic neural network with $N$ dimension requires $N^2$ MZIs. The gradient-free method requires $N^2 \times M \times P \times E$ energy consumption and $(M \times T_{\mathrm{MZI}} + T_{\mathrm{DIG}}) \times E$ time consumption, where $M$ is the number of groups of initial variables ($M = 1$ in other methods) and $E$ is the number of epochs needed for PNN convergence (assuming same for every method). The numerical gradient method requires $N^2 \times N^2 \times P \times E$ energy consumption and $(N^2 \times T_{\mathrm{MZI}} + T_{\mathrm{DIG}}) \times E$ time consumption since the gradient of each parameter requires $N^2$ operations to derive. The in situ method requires $(N^2 \times 3P + N \times 3P_{\mathrm{extra}}) \times E$ energy consumption, where 3 represents the number of measurements required for each epoch, $P_{\mathrm{extra}}$ represents the additional digital-to-analog converter and analog-to-digital converter. The in situ method requires 31 h for a task similar to ours, which includes time for the operation of MZIs, IR camera imaging, switching the bidirectional I/O, and digital readout and subtraction.

**Table 1.** Comparison to other training methods in energy and latency consumption. The settings in our experiment are $N = 8$, $M = 50$, $E \approx 200$, $P = 4.8$ mW, $P_{extra} = 20$ mW, $T_{MZI} = 1$ s, $T_{DIG} = 2$ s, $a = 13.7$, $b = 1.97$.

| | Gradient-free | Numerical gradient | In situ | Our PAGT | |
|---|---|---|---|---|---|
| | | | | PNN | DNN (one-time) |
| Energy | $N^2 \times M \times P \times E$ | $N^2 \times N^2 \times P \times E$ | $(N^2 \times 3P + N \times 3P_{extra}) \times E$ | $N^2 \times P \times E$ | $N^2 \times P \times a \times b^N$ |
| | (3072 J) | (3932 J) | (280 J) | (62 J) | (963 J) |
| Latency | $(M \times T_{MZI} + T_{DIG}) \times E$ | $(N^2 \times T_{MZI} + T_{DIG}) \times E$ | 31 h | $(T_{MZI} + T_{DIG}) \times E$ | $T_{MZI} \times a \times b^N$ |
| | (173 min) | (220 min) | 31 h | (10 min) | (52 min) |

The PAGT method can be separated into PNN training and one-time DNN training. The PNN training requires $N^2 \times P \times E$ in energy and $(T_{MZI} + T_{DIG}) \times E$ in time. The energy consumption and latency of various methods are compared in **Table 1** for PNN training. Our PAGT method outperforms the compared methods in terms of energy consumption and latency. Specifically, under the condition of the 8-mode chip, it reduces energy consumption by 4 to 63 times and time consumption by 17 to 186 times.

The consumption of DNN training is the only additional component of PAGT compared to others. It largely depends on the cost of collecting DNN training data, measurable in terms of $N^2 \times P \times a \times b^N$ energy consumption and $T_{MZI} \times a \times b^N$ time consumption, where $a$ and $b$ are from dataset volume $D_{DNN} = a \times b^N$. However, it is worth noting that the training of DNNs is a one-time consumption. In contrast to PNNs, which necessitate retraining for each distinct task, DNNs don't require the process. Although this method requires some initial cost, its scalability surpasses existing methods as the number of training tasks increases. This is because the training burden of the every-time PNN is outsourced to the one-time DNN training. This can be viewed as a larger intercept but a smaller slope when compared to other methods, as shown in Figure S2 (Supporting Information).

## 4. Conclusion

We demonstrate a significant advancement in PNN training by proposing a physics-aware analytic-gradient training algorithm for hybrid photonic-digital electronic neural networks. Low power consumption, efficient computation, fast convergence, and superior accuracy have been achieved compared to existing methods. The algorithm is validated using an 8-mode photonic neural network chip and exhibits great potential in various machine-learning tasks. In the construction of unitary matrices, the MSE between the reconstructed matrix and the original matrix was as low as 0.015, and the iris flower classification achieved an accuracy of 96.7%. Furthermore, our photonic GAN structure exhibited an impressive MSE below -48 dB for hybrid generating distributions. The analysis of PAGT scalability shows that PAGT meets the requirements of training most of the current PNN. Compared to other methods, our approach offers substantial improvements in both energy consumption and latency. Additionally, our method utilizes differentiable DNN to represent non-differentiable PNN, streamlining the training process and reducing costs in hybrid photonic-digital networks. These results demonstrate a promising path toward the development of highly efficient and accurate PNNs, with potential applications in optimization,[35] cryptography,[36] quantum machine learning,[37,38] quantum finance,[39–41] cascaded photonic neural network,[42] etc.

## Supporting Information

Supporting Information is available from the Wiley Online Library or from the author.

## Acknowledgements

This work is supported by the Singapore Ministry of Education Tier 3 grant (MOE2017-T3-1-001) and National Research Foundation grant (MOH-000926).

## Conflict of Interest

The authors declare no conflict of interest.

## Author Contributions

Y.C.Z. and H.Z. contributed equally to this work. Y.C.Z. and H.Z. jointly conceived the idea. Y.C.Z. and H.Z. designed the chip and built the experimental setup. H.C. and H.Z. fabricated the silicon photonic chip. Y.C.Z. performed the experiments. H.Z. and H.X.L. assisted with the set-up and experiment. X.D.J., M.W.M., D.P.P., and L.C.K. assisted with the theory. All authors contributed to the discussion of experimental results. H.Z., H.C., M.W.M., L.C.K., and A.Q.L. supervised and coordinated all the work. Y.C.Z., H.Z., and A.Q.L. wrote the manuscript with contributions from all co-authors.

## Data Availability Statement

The data that support the findings of this study are available from the corresponding author upon reasonable request.

## Keywords

on-chip training, optical computing, photonic integrated chip, photonic neural networks

Received: May 17, 2023
Revised: November 16, 2023
Published online: February 23, 2024

[1] D. Woods, T. J. Naughton, *Nat. Phys.* **2012**, *8*, 257.
[2] H. Zhang, M. Gu, X. Jiang, J. Thompson, H. Cai, S. Paesani, R. Santagati, A. Laing, Y. Zhang, M. Yung, M. H. Yung, Y. Z. Shi, F. K. Muhammad, G. Q. Lo, X. S. Luo, B. Dong, D. L. Kwong, L. C. Kwek, A. Q. Liu, *Nat. Commun.* **2021**, *12*, 457.
[3] T. Wang, S.-Y. Ma, L. G. Wright, T. Onodera, B. C. Richard, P. L. McMahon, *Nat. Commun.* **2022**, *13*, 123.
[4] T. Wang, M. M. Sohoni, L. G. Wright, M. M. Stein, S.-Y. Ma, T. Onodera, M. G. Anderson, P. L. McMahon, *Nat. Photonics* **2023**, *17*, 408.
[5] H. Zhou, J. Dong, J. Cheng, W. Dong, C. Huang, Y. Shen, Q. Zhang, M. Gu, C. Qian, H. Chen, Z. Ruan, X. Zhang, *Light: Sci. Appl.* **2022**, *11*, 30.
[6] S. M. Buckley, A. N. Tait, A. N. McCaughan, B. J. Shastri, *Nanophotonics* **2023**, *12*, 833.
[7] D. Brunner, M. C. Soriano, S. Fan, *Nanophotonics* **2023**, *12*, 773.
[8] R. Hamerly, L. Bernstein, A. Sludds, M. Soljačić, D. Englund, *Phys. Rev. X* **2019**, *9*, 021032.
[9] J. Gu, C. Feng, Z. Zhao, Z. Ying, R. T. Chen, D. Z. Pan, in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. *35*, MIT Press, Virtually **2021**, pp. 7583–7591.
[10] A. Cruz-Cabrera, M. Yang, G. Cui, E. Behrman, J. Steck, S. Skinner, *IEEE Trans. Neural Netw.* **2000**, *11*, 1450.
[11] H. Zhu, J. Zou, H. Zhang, Y. Shi, S. Luo, N. Wang, H. Cai, L. Wan, B. Wang, X. Jiang, J. Thompson, X. S. Luo, X. H. Zhou, L. M. Xiao, W. Huang, L. Patrick, M. Gu, L. C. Kwek, A. Q. Liu, *Nat. Commun.* **2022**, *13*, 1.
[12] T. Yan, R. Yang, Z. Zheng, X. Lin, H. Xiong, Q. Dai, *Sci. Adv.* **2022**, *8*, eabn7630.
[13] H. Zhang, J. W. Z. Lau, L. Wan, L. Shi, Y. Shi, H. Cai, X. Luo, G.-Q. Lo, C.-K. Lee, L. C. Kwek, A. Q. Liu, *Laser Photonics Rev.* **2023**, *17*, 2200698.
[14] H. Zhang, J. Thompson, M. Gu, X. D. Jiang, H. Cai, P. Y. Liu, Y. Shi, Y. Zhang, M. F. Karim, G. Q. Lo, X. Luo, B. Dong, L. C. Kwek, A. Q. Liu, *Acs Photonics* **2021**, *8*, 1662.
[15] M. Nakajima, K. Inoue, K. Tanaka, Y. Kuniyoshi, T. Hashimoto, K. Nakajima, *Nat. Commun.* **2022**, *13*, 7847.
[16] T. Zhang, J. Wang, Y. Dan, Y. Lanqiu, J. Dai, X. Han, X. Sun, K. Xu, *Opt. Express* **2019**, *27*, 37150.
[17] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, N. Killoran, *Phys. Rev. A* **2019**, *99*, 032331.
[18] L. De Marinis, M. Cococcioni, P. Castoldi, N. Andriolli, *IEEE Access* **2019**, *7*, 175827.
[19] Z. Zheng, Z. Duan, H. Chen, R. Yang, S. Gao, H. Zhang, H. Xiong, X. Lin, *Nat. Mach. Intell.* **2023**, *5*, 1119.
[20] S.-i. Yi, J. D. Kendall, R. S. Williams, S. Kumar, *Nat. Electron.* **2023**, *6*, 45.
[21] S.-i. Amari, *Neurocomputing* **1993**, *5*, 185.
[22] N. Ketkar, N. Ketkar, *Deep Learning with Python: A Hands-On Introduction*, Springer, Berlin, Heidelberg **2017**, pp. 113–132.
[23] T. Zhou, L. Fang, T. Yan, J. Wu, Y. Li, J. Fan, H. Wu, X. Lin, Q. Dai, *Photonics Res.* **2020**, *8*, 940.
[24] S. Xu, J. Wang, H. Shu, Z. Zhang, S. Yi, B. Bai, X. Wang, J. Liu, W. Zou, *Light: Sci. Appl.* **2021**, *10*, 221.
[25] C. Huang, V. J. Sorger, M. Miscuglio, M. Al-Qadasi, A. Mukherjee, L. Lampe, M. Nichols, A. N. Tait, T. Ferreira de Lima, B. A. Marquez, P. R. Prucnal, B. J. Shastri, *Adv. Phys.: X* **2022**, *7*, 1981155.
[26] T. W. Hughes, M. Minkov, Y. Shi, S. Fan, *Optica* **2018**, *5*, 864.
[27] S. Pai, Z. Sun, T. W. Hughes, T. Park, B. Bartlett, I. A. Williamson, M. Minkov, M. Milanizadeh, N. Abebe, F. Morichetti, Andrea Melloni, S. Fan, O. Solgaard, D. A. B. Miller, *Science* **2023**, *380*, 398.
[28] K. Shiflett, D. Wright, A. Karanth, A. Louri, in *2020 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, IEEE, Piscataway, NJ **2020**, pp. 474–487.
[29] Z. Liu, L. Raju, D. Zhu, W. Cai, *IEEE J. Emerg. Sel. Topics Circuits Syst.* **2020**, *10*, 126.
[30] C. Zoufal, A. Lucchi, S. Woerner, *npj Quantum Inf.* **2019**, *5*, 103.
[31] W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer, I. A. Walmsley, *Optica* **2016**, *3*, 1460.
[32] M. Arjovsky, S. Chintala, L. Bottou, in *International conference on machine learning*, PMLR, Sydney, Australia **2017**, pp. 214–223.
[33] C. Taballione, R. van der Meer, H. J. Snijders, P. Hooijschuur, J. P. Epping, M. de Goede, B. Kassenberg, P. Venderbosch, C. Toebes, H. van den Vlekkert, P. W. H. Pinkse, J. J. Renema, *Mater. Quantum Technol.* **2021**, *1*, 035002.
[34] J. Bao, Z. Fu, T. Pramanik, J. Mao, Y. Chi, Y. Cao, C. Zhai, Y. Mao, T. Dai, X. Chen, X. Jia, L. Zhao, Y. Zheng, B. Tang, Z. Li, J. Luo, W. Wang, Y. Yang, Y. Peng, D. Liu, D. Dai, Q. He, A. L. Muthali, L. K. Oxenløwe, C. Vigliar, S. Paesani, H. Hou, R. Santagati, J. W. Silverstone, A. Laing, et al., *Nat. Photonics* **2023**, *17*, 573.
[35] Z. Li, H. Zhang, B. T. T. Nguyen, S. Luo, P. Y. Liu, J. Zou, Y. Shi, H. Cai, Z. Yang, Y. Jin, Y. Hao, Y. Zhang, A.-Q. Liu, *Photonics Res.* **2021**, *9*, B38.
[36] N. Gisin, G. Ribordy, W. Tittel, H. Zbinden, *Rev. Mod. Phys.* **2002**, *74*, 145.
[37] H. Zhang, L. Wan, T. Haug, W.-K. Mok, S. Paesani, Y. Shi, H. Cai, L. K. Chin, M. F. Karim, L. Xiao, X. Luo, F. Gao, B. Dong, S. Assad, M. S. Kim, A. Laing, L. C. Kwek, A. Q. Liu, *Sci. Adv.* **2022**, *8*, eabn9783.
[38] Y. Zhan, H. Zhang, H. Cai, D. Poenar, L. Kwek, A. Liu, in *CLEO: Science and Innovations*, Optica Publishing Group, Washington, DC **2023**, pp. JTh2A–22.
[39] Y. Zhan, H. Zhang, L. Wan, M. Karim, H. Cai, L. Kwek, A. Liu, in *Novel Optical Materials and Applications*, Optica Publishing Group, Washington, DC **2022**, pp. JTh4A–4.
[40] N. Schetakis, D. Aghamalyan, P. Griffin, M. Boguslavsky, *Sci. Rep.* **2022**, *12*, 1.
[41] H. Zhang, L. Wan, S. Ramos-Calderer, Y. Zhan, W.-K. Mok, H. Cai, F. Gao, X. Luo, G.-Q. Lo, L. C. Kwek, J. I. Latorre, A. Q. Liu, *Photonics Res.* **2023**, *11*, 1703.
[42] X. Guo, S. Xiang, Y. Zhang, Z. Song, Y. Han, B. Gu, D. Zheng, X. Chen, Y. Shi, Y. Hao, *J. Lightwave Technol.* **2023**, *41*, 6533.