Advanced Quantum Technologies

ADVANCED QUANTUM TECHNOLOGIES
www.advquantumtech.com

RESEARCH ARTICLE OPEN ACCESS

Training Continuously-Coupled Reconfigurable Photonic Chips with Quantum Machine Learning

Denis Stanev[1] | Nicolò Spagnolo[2] | Fabio Sciarrino[2]

[1]Gran Sasso Science Institute, L'Aquila, Italy | [2]Dipartimento di Fisica, Sapienza Università di Roma, Roma, Italy

Correspondence: Fabio Sciarrino (fabio.sciarrino@uniroma1.it)

Received: 16 December 2025 | **Revised:** 11 February 2026 | **Accepted:** 31 March 2026

Keywords: integrated photonics | quantum information | quantum machine learning

ABSTRACT

Integrated photonic technologies have recently shown significant advances, enabling the possibility to implement reconfigurable interferometers with increasing size. One of the main tasks to fully exploit the capabilities of reconfigurable integrated interferometers is the possibility to precisely program their operation to perform a desired target unitary. While recipes are known for circuit layouts based on a cascade of beam-splitter and phase-shifter operations, a methodology applicable for reconfigurable continuously coupled waveguide arrays is currently missing. Here, we devise a machine learning-based approach for this task, using a black box methodology that does not rely on precise a priori modeling of the circuit's internal architectures. We verify the effectiveness and the robustness of this approach via numerical simulations on different continuously coupled waveguide layouts, either with planar or 3D structures. The proposed method makes use of a limited number of single- and two-photon measurements, making it suitable for optical quantum information processing. The obtained results open the perspective of employing this methodology as an effective tool to program the operation of integrated interferometers designed via different architectures.

1 | Introduction

Over the last decade, quantum devices have shown a rapid growth and evolution, with some having already claimed the first instances of quantum advantage [1–8]. This technological development has been accompanied by the adoption of different architectures and technologies for the fabrication of quantum devices, each with its own set of advantages and drawbacks. Among the different experimental platforms, photonic systems have been shown to be a relevant approach to scale-up to large dimensional quantum systems [5, 6]. The technological development has also been accompanied to the theoretical definition of different encodings and protocols for quantum information processing, specifically suitable for photonic platforms [9–20]. Some of the more common approaches to process quantum information with photons involve using linear optical components, such as beamsplitters and phase shifters, as a building block for complex interferometric networks. Several recipes are already known to decompose an arbitrary target unitary into a sequence of elementary components [21, 22], with the number of elements scaling typically as $O(m^2)$, being m the number of optical modes. As an effective platform to implement multi-mode interferometer with this architecture, recent experiments have shown the realization of progressively larger devices with integrated photonics [23].

A different approach to implement multi-mode interferometric networks with integrated photonics involves using an architecture inspired by continuous-time quantum walks. More specifically, a set of continuously-coupled waveguides permits to implement a linear unitary dynamics, where the effective transformation depends on the mutual distance between the

modes and on the interaction length [24]. Furthermore, recent technological advances enabled the integration of reconfiguration capabilities via either thermal or elecro-optic phase shifters [25–27]. This architecture allows for compact designs, which enable coupling of several modes with a reduced circuit length, thus potentially reducing the impact of losses. Such a property have been further enhanced with recent demonstration of continuously-coupled multi-mode interferometers with a 3D structure, which enables improved connectivity between modes with respect to planar designs [25, 28–32]. However, there are still challenges in the adoption of this architecture to obtain fully-programmable and controllable devices, given that there is currently no known general algorithm that links the circuit parameters to the implementation of specific target unitaries.

Recently, machine learning has seen a surge in uses in all types of fields [33–37], and has proven to be a useful tool when used in conjunction with quantum devices [38–45]. The ability of machine learning techniques to work even on systems that are not well characterized, using a black box approach, makes them suitable for scenario in which a complete characterization by standard means might be challenging and expensive. Hence, these methods may prove to be applicable to the aforementioned task of programming multi-mode interferometers.

In this work, we devise a machine learning based approach to program reconfigurable integrated interferometers based on continuously-coupled waveguides. The advantages of this approach relies on the lack of needing the full reconstruction of the unitary matrices for a comprehensive set of system parameters, and on employing a limited number of single- and two-photon measurements. We then showcase that the method can be effective for different circuit layouts and programmabilities, including some of the recently demonstrated architectures based on 3D geometries [25].

This work is organized as follows. In Section 2, we discuss the theoretical framework for continuously-coupled integrated interferometers. Then in Section 3, we describe the methods at the basis of the devised machine-learning approach acting as local optimization. In Section 4, we discuss the results of numerical simulations of circuits with different layouts used to test the effectiveness of the method. In Section 5, we then discuss possible approaches to implement the initial step of the optimization, complementing the local optimization approach. Finally in Section 6, we provide some concluding remarks.

2 | Implementing Linear Interferometers via Continuously Coupled Waveguides

As briefly discussed in the introduction, implementation of linear optical unitary transformations can be performed by following two different approaches. As a first approach, a generic $m \times m$ transformation between optical modes can be implemented via a structure composed of discrete 2×2 cells, where each cell is composed by a beam-splitter of arbitrary transmittivity and a phase shifter in one of the modes (see Figure 1a). Different layouts [21, 22] have been identified leading to implementation of arbitrary unitary transformations, requiring $O(m^2)$ elementary cells arranged in $O(m)$ layers, each accompanied by a related algorithm to deterministically find the circuit parameters to implement a chosen overall transformation. This approach has the advantage of having a specific recipe for its programming, and is accompanied by a well defined scaling with losses as $O(\eta^m)$, where η is the transmission per element.

As a second approach, integrated devices composed of continuously coupled waveguides do not rely on the composition of multiple discrete optical units to achieve a final target unitary. Conversely, coupling between the modes is obtained by having the set of optical waveguides arranged so that their distance allows for mutual coupling for the complete circuit length (see Figure 1b). This can be described by an Hamiltonian $\hat{\mathcal{H}}$ between the mode operators $\{\hat{a}_i, \hat{a}_i^\dagger\}$ of the form:

$$\hat{\mathcal{H}} = \sum_{i=1}^{m} \beta_i \hat{a}_i^\dagger \hat{a}_i + \sum_{i,j\neq i}^{m} C_{i,j} \hat{a}_i^\dagger \hat{a}_j \tag{1}$$

where the coefficients β_i are the propagation constants of the modes, while the off-diagonal elements $C_{i,j}$ are the coupling coefficients between different modes. In the case of a chip with m modes with a planar structure and first-neighbor connectivity, the coupling coefficients are non-zero only if $j = \{i-1, i+1\}$. The effective unitary operator $\hat{U}(z)$ induced by a circuit of length z can be obtained from the Hamiltonian (1). It is important to note that the value of the parameters of the Hamiltonian can change over the length of the circuit, based on its structure. In that case, one needs to divide the circuit in k blocks of length Δz, with $z = k\Delta z$, each described by an Hamiltonian $\hat{\mathcal{H}}_l$ and a corresponding unitary operator $\hat{U}_l(\Delta z)$. The overall unitary operator is obtained $\hat{U}(z) = \hat{U}_m(\Delta z) \cdots \hat{U}_1(\Delta z)$. Considering the form of $\hat{\mathcal{H}}$, the transformation induced by this class of systems is linear between the modes, and is thus described by a unitary matrix $U_{i,j}$ that defines the input–output relations for the mode operators $\hat{b}_i = \sum_j U_{i,j} \hat{a}_i$ which are obtained from $\hat{b}_i = \hat{U}(z) \hat{a}_i \hat{U}^\dagger(z)$.

The off-diagonal $C_{i,j}$ elements of the Hamiltonian are given by the structure of the chip, since they depend on the distance between the waveguides, and are thus typically fixed in the fabrication process. Conversely, the diagonal elements β_i can be actively modified, through the use of the thermo-optic [25] or of the electro-optic [26, 27] effect. Modifying the propagation constants of the modes using thermo-optic effects is usually achieved by placing resistors on the chip surface that, when given a certain input current, change by heating the propagation constants β_i of the affected modes.

Albeit some recipes [46] based on stochastic quantum walks [47] to generate Haar-randomness from continuously-coupled waveguide arrays are known, the challenge is the lack of a known procedure to translate the coefficients of a desired unitary matrix U, of elements $U_{i,j}$ in a set of parameters to program a given interferometer. In this work, we intend to tackle this issue by using machine learning techniques to train continuously-coupled arrays with thermo-optic tuning to achieve specific unitaries. We test our methods on different waveguide structures, both planar and 3D. Furthermore, we will also consider a scenario where thermo-optic tuning does not allow to control only the propagation constant β_i of a single given mode due to the presence of thermal cross-talks. This will be accompanied also by an

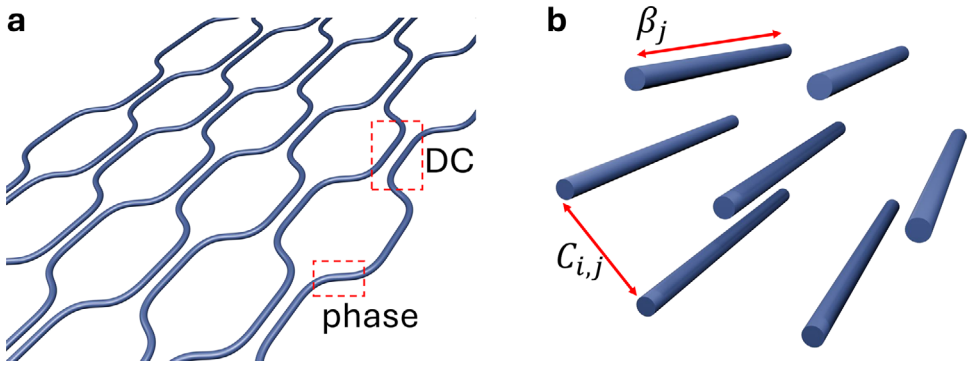

FIGURE 1 | Implementation of linear optical unitary transformation with integrated photonics. (a) Schematics of unitary decomposition via a discrete approach. The unitary matrix is obtained by a set of 2×2 elementary cells, composed of a directional coupler (DC) with arbitrary transmittivity and a phase shift, arranged in a suitable layout. (b) Interferometer composed by a network of continuously-coupled waveguides. The relevant parameters describing the evolution are represented by the propagation constants $\{\beta_i\}$, and by the coupling coefficients $\{C_{i,j}\}$.

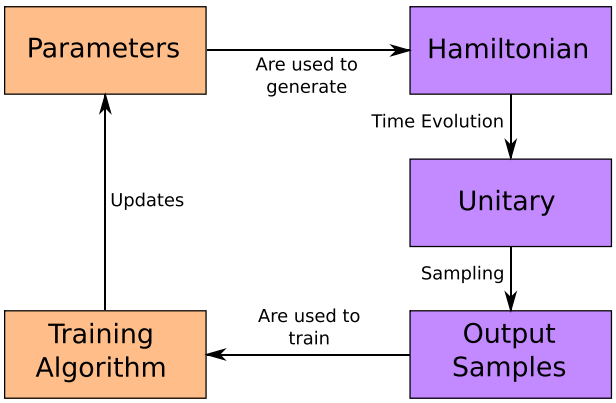

FIGURE 2 | Schematic of the training cycle. Logical sequence of operations to perform the training process in the algorithm. In purple, we highlight the parts that are performed on the quantum hardware, while in orange the parts performed on classical hardware.

analysis of the impact on the performance caused by performing the training process with a finite number of measurements.

3 | Methods

Our approach to program reconfigurable interferometers based on continuously-coupled waveguide arrays was based on a machine learning approach, where the set of parameters to be applied to obtain a given unitary transformation was determined through a training phase. A schematic of the training cycle can be seen in Figure 2. More specifically, the cycle embeds a sequence composed of (i) choosing a set of system parameters, (ii) applying them to the reconfigurable interferometer, (iii) measuring a finite sample at its output after feeding the circuit with a chosen set of input states, and (iv) determining through the training algorithm the set to be applied to the circuit in the subsequent step. The training phase stopped after a given halting condition was met.

As shown in Figure 2, this cycle involved two main blocks depending on the hardware employed at the corresponding state. A part of the process is performed on the quantum hardware (A), namely setting the parameters and measuring the samples, and a part of the process is performed on a classical hardware (B), namely the training algorithm that decides the set of parameters to be inserted at the next step. In our case, part (A) was performed by testing our approach on numerical simulations of the quantum hardware. The latter is performed by exploiting the Python libraries Qutip [48] and Perceval [49]. Conversely, in part (B) the training code is mostly ad-hoc implemented for this analysis. We describe below all the relative steps carried out in the performed analysis.

3.1 | Simulation of Continuously-Coupled Waveguide Arrays

In order to address the capability of the algorithm to find a chosen target unitary for different device architectures, we simulate various sizes and designs of optical continuously-coupled waveguide arrays. More specifically, we first started by considering integrated interferometers with a planar structure, such as those reported in [24, 26, 27], with varying number of optical modes. Then we also considered integrated interferometers with a 3D structure [28–32], more precisely a triangular structure reproducing the one of Ref. [25]. We show in Figure 3a–b the cross-section of the two different geometric architectures simulated in the present study.

A crucial aspect of the simulation process is how we introduce the active reconfigurability in the device through thermo-optic resistors. These elements inserted modifications in the index of refraction of the underlying waveguide(s), which depend on the amount of dissipated power in the resistor, due to the thermo-optic effect. Hence, thermo-optic resistors can be in principle used to modify the propagation constants β_i in Equation (1). However, one needs to consider that in general, due to the heat propagation process, thermal cross-talks can be present. This lead to inducing modifications of $\beta_{j\neq i}$ even if the resistor is placed on top of waveguide i. This is particularly relevant for 3D geometries, where the heat propagation progress can affect not only the waveguide placed right below the resistors, but also those buried deeper below in the structure. To take into account this relevant aspect for actual implementations, we progressively

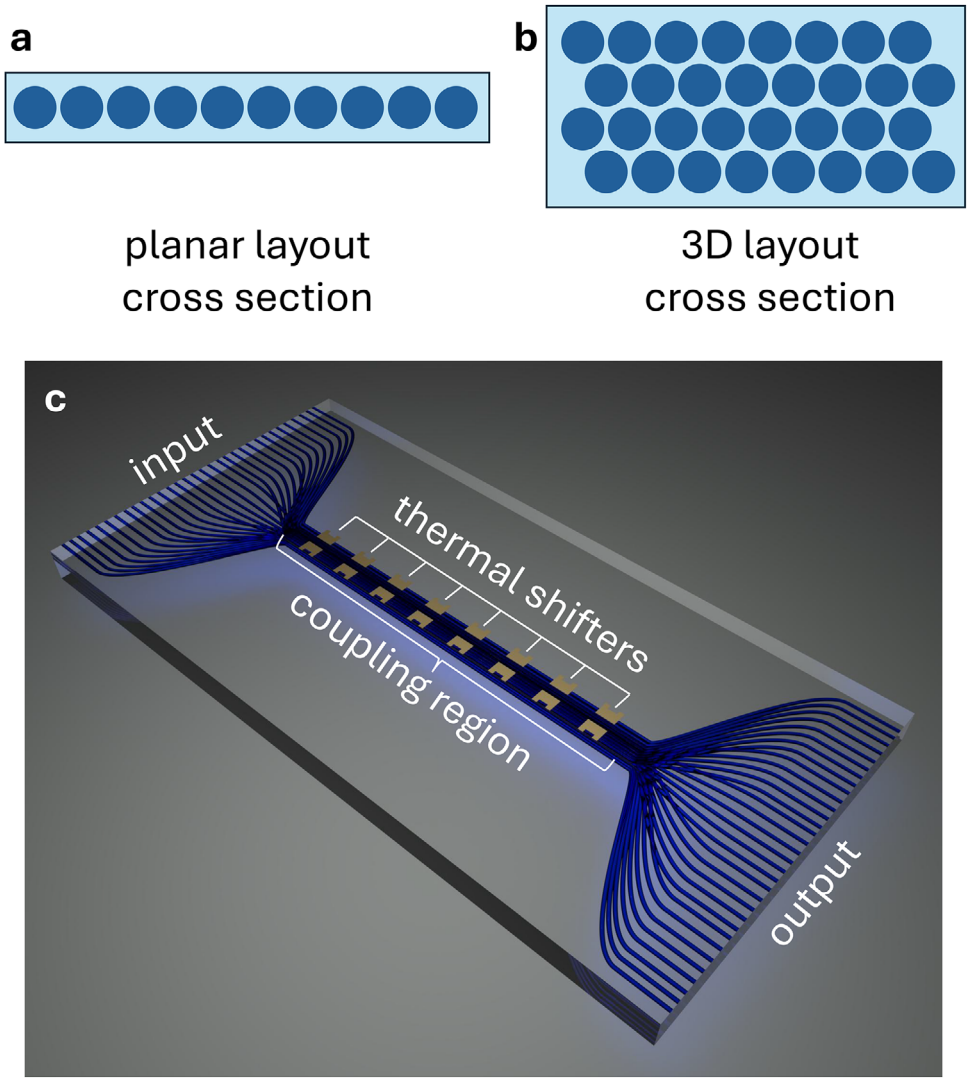

FIGURE 3 | Structure of the simulated architectures. (a,b) Cross-section of the different interferometer layout, where circle represents a mode. (a) Planar design, in this case representing the scenario with ten modes. (b) 3D structure with triangular mesh with 32 modes. (c) Visual representation of the architecture for 3D interferometers with multiple segments. The thermal resistors are placed on top of the device surface, and control the propagation constant of the underlying modes.

performed a more physically-detailed simulation of the process. More specifically, we first started by considering the case in which we have individual control over all of the modes. Then, in the case of the 3D structure, we also considered the case where tuning the current in a single resistor does not address only the propagation constant of a single mode, but affects multiple modes simultaneously (see also Ref. [25]). Finally, we modified the architecture of the simulated interferometers taking into account more complex geometries for the resistors [25], namely the scenario where the devices is performed via multiple connected segments which can be controlled independently (see Figure 3c). More details can be found in Appendix A.

The numerical simulation process of the interferometer evolution is performed with the following approach. First, we generate the Hamiltonian, representing the transformation performed by a cross-section of the chip, with a chosen set of parameters. Then, we calculate the unitary matrix U resulting from a chosen length z for the chip. Then we use the obtained unitary matrix to simulate a set of output events, with a specified number of samples, resulting from injecting the circuits with the set of input states used for the training. For the training process, we employ single- and two-photon input states. More specifically, the complete set of single-photon input states is probed, equal to the number of modes, following the consideration that this set provides information on the moduli of the unitary matrix elements ($|U_{i,j}|$). For the two-photon input states, we use a limited set of input configurations to also obtain information on the complex phases of the transformation, where the chosen set depends on the structure of the chip and on the training method.

3.2 | Training

We now describe the training procedure for our method. A schematic description of the used procedure is reported in Algorithm 1. The aim is to program the circuit to perform a specific target unitary U^t, defined initially before the starting

ALGORITHM 1 | Training of the continuously coupled optical chip

Input: Chip and training settings, initial value of trainable parameters $\{\beta_i^{(0)}\} = \{\beta_1^{(0)}, \beta_2^{(0)}, ..., \beta_n^{(0)}\}$

Result: Trained parameters $\{\beta_i^{(f)}\}$, where f indicates the last epoch of the training.

$l = 0$;
f = number of epochs ;
while $l < f$ **do**
  $\{\beta_i^{(l+1)}\} = \{\beta_i^{(l)}\}$;
  Choose index $i0$ of the parameter to modify at this iteration ;
  Choose subset of available two-photon input pairs to use for this epoch ;
  Compute $l_{prev} = L(\{\beta_i^{(l)}\})$;
  Compute $l_{up} = L(\{\beta_i^{(l)}\} + \delta_{i0})$;
  Compute $l_{down} = L(\{\beta_i^{(l)}\} - \delta_{i0})$;
  /* Choose in which direction to shift $\beta_{i0}^{(l)}$ to minimize the loss. The exact value of δ^{shift} depends on the training settings. */
  if $(l_{up} < l_{prev})$ *and* $(l_{down} < l_{prev})$ **then**
    if $(l_{down} < l_{up})$ **then**
      $\beta_{i0}^{(l+1)} = \beta_{i0}^{(l)} - \delta^{shift}$;
    else
      $\beta_{i0}^{(l+1)} = \beta_{i0}^{(l)} + \delta^{shift}$;
    end
  else
    if $(l_{up} < l_{prev})$ **then**
      $\beta_{i0}^{(l+1)} = \beta_{i0}^{(l)} + \delta^{shift}$;
    else
      if $(l_{down} < l_{prev})$ **then**
        $\beta_{i0}^{(l+1)} = \beta_{i0}^{(l)} - \delta^{shift}$;
      else
        /* Do not shift the parameter if both directions led to higher loss than its current position */
        $\beta_{i0}^{(l+1)} = \beta_{i0}^{(l)}$;
      end
    end
  end
  $l = l + 1$;
end
return $\{\beta_i^{(f)}\}$;

point. In principle, for the training to be possible in an exact form, there should exist a set of parameters $\{\beta_i^t, C_{i,j}^t\}$, which correspond to the unitary U^t according to the Hamiltonian description of Equation (1). In the experimental platform, the circuit can be programmed via a set of control knobs, provided by the currents applied to the resistors placed on the surface of the interferometer. As discussed previously, application of a current I_k in the k-th resistor modifies the propagation constants $\{\beta_i\}$ via the thermo-optic effect, according to heat propagation mechanisms. In general, this means that changing the current I_k on the resistor placed on top of waveguide i does not imply changing only a single β_i in the presence of thermal cross-talks. Conversely, the coupling parameters $\{C_{i,j}^t\}$ are unchanged by thermo-optic phase shifters, and thus we will consider them fixed in all performed numerical simulations.

Our training method works by training directly the parameters of the quantum system. In fact, the only classical part used in the approach is the optimizer, which takes the output distributions from the quantum systems, and decides how to update the parameters based on a cost function. These updated parameters are then provided to the quantum system for the successive epoch of training, repeating the process until a predetermined number of epochs has passed. This means that, when working with real quantum hardware, there is no need to simulate the behavior of the quantum system classically to perform the training process. This allows in principle to apply our training method to devices of arbitrary size.

The proposed training method is then tailored to allow for training directly on-chip, both by making use of a limited number of single- and two-photon measurements, and by ensuring that the techniques used can be applied directly on an actual device. This also means that we avoid using in the training process techniques that would be possible only in a simulated environment, such as backpropagation, to make our training method suitable for implementation on real hardware. Crucially, our training method treats the optical chip as a black box, and thus does not require extensive characterization and a precise a-priori modeling of the circuit internal architectures. This approach also allows the devised method to work for chips with various layouts, without requiring modifications to the training process. We also implemented several optimizations, such as reducing the number of input pairs that are used in each training epoch, to accelerate the training process.

We now give a short overview of the training process, which will be described below in more detail. We start with a desired target unitary, and a set of starting parameters, which will be used as the starting point for the training. We then choose a set of single- and two-photon input states, and collect N_s samples for each chosen input state, from which we obtain a set of output distributions. These output distributions are then used to evaluate a chosen loss function $L(\{\beta_i\})$ between the measured samples and the expected target probabilities. After that, we choose one of the trainable parameters, shift its value, and evaluate the loss at the new position. This allows us to use an adapted version of Finite Differences Stochastic Approximation (FDSA) [50, 51] to determine how to modify the chosen parameter to reduce the loss value. We then repeat this process a chosen number of times, obtaining as output the parameters to be set on the device to perform our desired transformation. We will now proceed to show in more detail how each of these steps works, including the methods used to increase the efficiency of the training process.

The starting point of the implemented approach is thus an initial set of currents $\{I_k^{(0)}\}$. In the numerical simulation of the method performed here to test its functioning, this corresponds to a set of initial propagation constants $\{\beta_i^{(0)}\}$, while as said the coupling parameters are fixed to $\{C_{i,j}^t\}$. The values of $\{\beta_i^{(0)}\}$ ($\{I_k^{(0)}\}$ in an actual experiment) reflect the initial knowledge on the system before the training algorithm is run. Specifically, the starting parameters $\{\beta_i^{(0)}\}$ are fixed by taking the true parameter values $\{\beta_i^t\}$, and applying a set of random shifts $\{\delta\beta_i\}$, whose value depend on the size and mesh of the circuit and on the amount of initial knowledge on the circuit operation.

The training process then starts by choosing a set of single- and two-photon input states, and collect N_s samples for each input state. A relevant aspect of the method is the criteria used to choose the set of states. As discussed above, in all tested scenarios the full set of single-photon inputs is employed. For the two-photon states, we considered different rules depending on the circuit geometry. In the case of the planar design, we use all of the input pairs of first neighbors $(i, i+1)$, plus the input pair $(1, m)$ composed by the first and last mode. This corresponds to an overall amount of $M_1 = m$ single-photon and $M_2 = m$ two-photon input states. In the case of the triangular mesh, we first considered the set of all first neighbor states, which is larger than m due to the higher connectivity of the design. For instance, this corresponds to $M_2 = 73$ distinct mode pairs for a $m = 32$ mode device. As a second trial, we tested the performance of the training when reducing the size of this set. This can be seen as similar to the approach used by Stochastic Gradient Descent [52], where only a subset of the dataset is used to approximate the gradient at each step. While in each epoch the training uses only a subset of all the input pairs, if this subset is of an appropriate size it is still informative on the complete evolution. This corresponds to an approximately correct identification of the parameter shift. Also, since the elements of the subset change at each epoch, over the length of the whole training all the input pairs are used to train the model, ensuring that the trained model is not biased toward a specific subset of input pairs. This leads to a significant speed up in the training process, while maintaining a similar final fidelity. We will see in the results section how using different numbers of pairs per epoch influences the training process.

The exploited set of output distributions used at each epoch is then composed by (i) estimates $\{\tilde{p}_{i;j}\}$ of the M_1 single-photon probabilities from input i to output j, obtained as $\tilde{p}_{i;j} = \tilde{N}_{i;j}/N_s$, where $\tilde{N}_{i;j}$ is the number of events obtained for the given configuration, and (ii) estimates $\{\tilde{p}_{i,k;j,l}\}$ of the M_2 two-photon probabilities from input pair (i,k) to output pair (j,l), also in this case obtained as $\tilde{p}_{i,k;j,l} = \tilde{N}_{i,k;j,k}/N_s$, where $\tilde{N}_{i,k;j,l}$ is the number of events obtained for the given configuration. In an experimental implementation, each output distribution is collected by directly preparing the chosen states and measuring output samples of the chosen size N_s for each input. In the numerical simulation performed here to test the algorithm operation, the estimates $\{\tilde{p}_{i;j}\}$ and $\{\tilde{p}_{i,k;j,l}\}$ of the single- and two-photon probabilities are obtained by sampling N_s events from the simulated output probabilities $\{p^{(l)}_{i;j}\}$ and $\{p^{(l)}_{i,k;j,l}\}$. The latter are evaluated via the permanent formula from the unitary matrix $U^{(l)}$ corresponding to $\{\beta^{(l)}_i, C^t_{i,j}\}$, where $\{\beta^{(l)}_i\}$ are the propagation constants at step l, as $p^{(l)}_{i;j} = |U_{j,i}|^2$ and $p^{(l)}_{i,k;j,l} = |U_{j,i}U_{k,l} + U_{j,l}U_{k,i}|^2$. Note that explicit knowledge of $\{\beta^{(l)}_i\}$ and $\{C^t_{i,j}\}$ is needed to perform the numerical simulation, while this is not necessary in an actual experiment since only the currents $\{I_k\}$ are modified, and the training dataset is estimated from the measurements.

The loss function $L(\{\beta_i\})$ of the algorithm is chosen to be the Mean Absolute Error (MAE) between the measured samples and the expected probabilities, calculated as:

$$L(\{\beta_i\}) = \frac{1}{M_1}\sum_{i,j}^{M_1} |\tilde{p}_{i;j} - p^t_{i;j}| + \frac{1}{M_2}\sum_{i,k,j,l}^{M_2} |\tilde{p}_{i,k;j,l} - p^t_{i,k;j,l}| \quad (2)$$

where the sums are extended over the measured input-output combinations, and the target probabilities are calculated directly from U^t. Note that using this loss function, rather than comparing the fidelity $F = |\mathrm{Tr}[(U^t)^\dagger U^{(l)}]|/m$ between the target unitary U^t and the actual unitary $U^{(l)}$, can be less expensive in terms of measurement overheads since estimating F requires performing the tomography of the implemented unitary at each step to reconstruct also the phase contributions [53–56]. While we do compute the fidelity F during the simulations to prove that the training is performing well, this information is not used to train the circuit, and is thus not needed when performing this training method on actual quantum hardware. An analysis on the loss landscape with respect to individual parameters is discussed in Appendix B.

One significant challenge present in training this model is that backpropagation, or even just backpropagation-like scaling, is generally not possible, at least without using multiple copies of a state [57]. Thus, we need to use a different method to estimate the gradient required by the training process. In the anayzed scenario, we found out that an adapted version of Finite Differences Stochastic Approximation (FDSA) [50, 51] suffices for our scope. For each epoch l, the starting point is a set of parameters $\{\beta^{(l)}_i\}$ (or, for an actual experiment, the currents $\{I^{(l)}_k\}$). The algorithm then chooses a parameter specified by a randomly-sampled index i_0, and requires to estimate $L(\{\beta_i\})$ in three different cases, namely for (1) $\{\beta^{(l)}_i\}$, (2) by shifting $\beta^{(l)}_{i_0}$ of a quantity $+\delta$ and (3) by shifting $\beta^{(l)}_{i_0}$ of a quantity $-\delta$, where δ is the learning rate that needs to be adapted during the algorithm. Knowledge of these values for the loss permits to estimate the correct direction to shift $\beta^{(l)}_{i_0}$ and define the set of parameters $\{\beta^{(l+1)}_i\}$ at epoch $l+1$. This approach can be also further adapted to involve in some epochs the estimate of $L(\{\beta_i\})$ in only 2 points. Thus, an epoch requires 2 or 3 full evaluation of the training dataset. In Appendix C, we discuss in more details this adapted version of FDSA, and also discuss prospects of applying other approaches such as Simultaneous Perturbation Stochastic Approximation (SPSA) [58].

During the numerical simulation, we included also some refinements to the discussed training algorithm to match the physical implementation. For instance, we included an upper and lower bound to the propagation constants, which reflect the finite range of operation of thermo-optic resistors. Furthermore, as discussed above, for the triangular mesh we included the effect that shifting a parameter β_i produces a cross-talk effect that modifies also neighbors $\beta_{j\neq i}$. In our simulation, we included this effect in a way that mimics heat propagation mechanisms (see Appendix A). Finally, we also tested the performance of the algorithm when the target unitary is not necessarily achievable by the device, namely that the thermo-optic shifters do not allow to program the set of parameters $\{\beta^t_i, C^t_{i,j}\}$ for the exact implementation of U^t. This was done by slightly changing the values of the coupling parameters $\{C^t_{i,j}\}$, and then using throughout the algorithm those modified parameters during the training.

4 | Results

In this section, we discuss the results of the different simulations performed to test the method described in the previous sections.

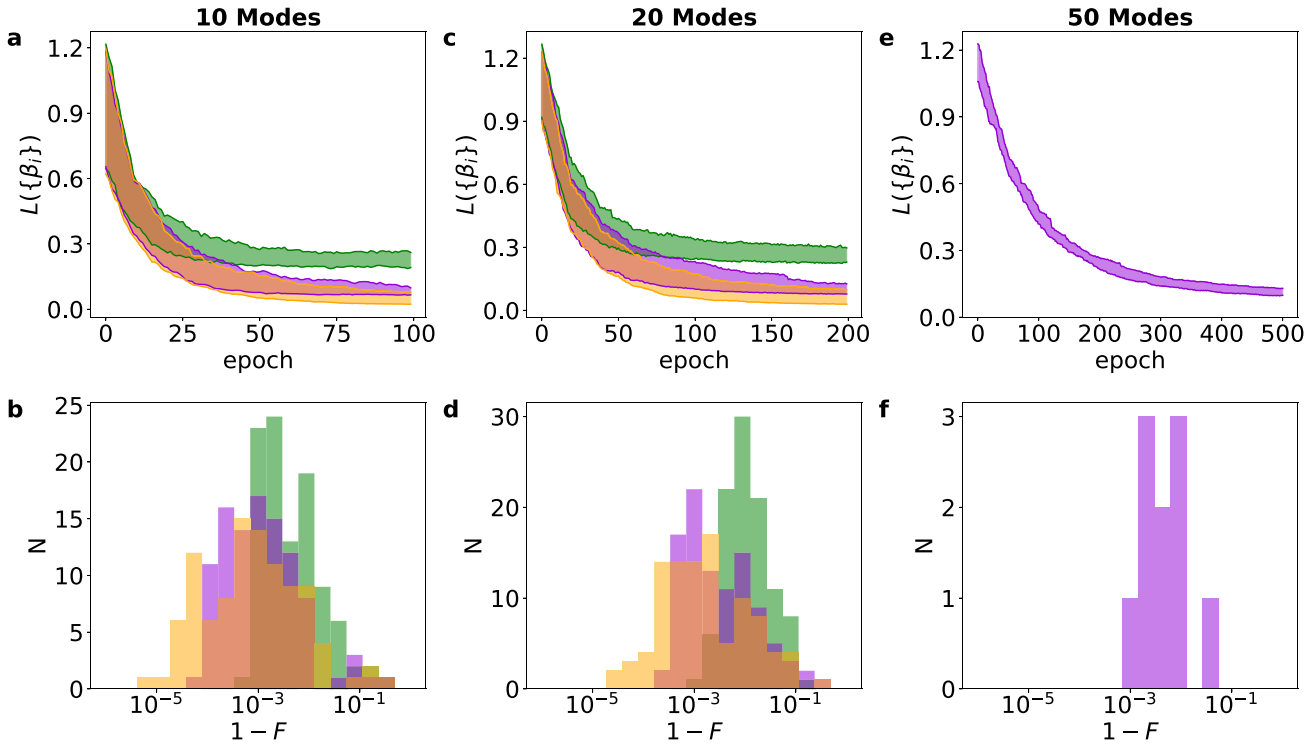

FIGURE 4 | Numerical simulations for the planar layout. (a) Loss and (b) infidelity for the 10-mode planar layout, with initial fidelity $F_{\text{in}} \sim 0.7$. Green data sets: $N_s = 10^3$. Purple data sets: $N_s = 10^4$. Orange data sets: $N_s = 10^5$. (c) Loss and (d) infidelity for the 20-mode planar layout, with initial fidelity $F_{\text{in}} \sim 0.7$. Green data sets: $N_s = 2 \times 10^3$. Purple data sets: $N_s = 2 \times 10^4$. Orange data sets: $N_s = 2 \times 10^5$. (e) Loss and (f) infidelity for the 50-mode planar layout, with initial fidelity $F_{\text{in}} \sim 0.7$. Purple data sets: $N_s = 5 \times 10^4$. For the 10 and 20 modes cases, 100 simulations were done for each case, while for the 50 mode case only ten simulations were performed. For the loss, the shaded zone represents the area between the 10th and 90th percentile of the runs. The histogram x axis is in logarithmic scale.

We simulated the training process for different systems, namely interferometers ranging from 10 to 50 modes with a planar structure, and interferometers with 32 modes using a 3D triangular mesh. In our analysis, we included in the simulations the effect of the sample size, by varying the number of samples N_s for each state in the range $N_s \in [10^3, 2 \times 10^5]$. The training has been shown for all these cases to highlight its capability to work with different structures and limitations, and thus show the flexibility of this method. As figures of merit, we report the values of the loss function $L(\{\beta_i\})$, which is effectively estimated during the training process, and of the infidelity $1 - F$, being F the fidelity between the target unitary and the trained one as defined above. The latter parameter is not used in the training process due to the need to perform expensive unitary tomography at each step, while it expresses how close the trained unitary is to the target unitary. We also show that the training algorithm works properly since, although the fidelity F is not computed during the process, minimization of the loss function corresponds to simultaneous minimization of $1 - F$.

4.1 | Planar Layout

First, we start by considering interferometers with a planar geometry, assuming that the propagation constant of each mode can be controlled directly via phase shifters placed on top of the surface above the waveguide. We have started by testing our method via numerical simulations for ten-mode interferometers, investigating the role of the sample size statistics per input state. In Figure 4a,b, we show specifically different trainings for several unitaries, by varying the number of samples N_s. Each run was performed by having a starting fidelity of $F_{\text{in}} \sim 0.7$, achieved in our simulation with a random shift of $\delta\beta_i \in [-0.1, +0.1]\,\text{mm}^{-1}$ for the propagation constants. For an experimental implementation, this corresponds to the need of having a starting knowledge of the circuit operation corresponding to that value of the fidelity. The final average fidelity over 100 runs is reported in Table 1. From this analysis, we observe that for this system size, an average fidelity > 0.99 can be achieve already with $N_s = 10^3$, while considering larger values can lead to further improving the average values of the programming fidelity. Ultimately, the choice of N_s has to take into account collecting a larger number of samples N_s increases the time required for each epoch of the training, while at a certain point increasing its value N_s stops yielding significant gains in the final fidelity of the trained optical chip.

We have then performed the same numerical simulation for the 20-mode scenario, using the same amount of initial shift $\delta\beta_i$ in the propagation constants, which corresponds to a starting fidelity in the range $F_{\text{in}} \sim 0.4 - 0.5$. We observed a similar behavior to the case with ten modes in terms of scaling with N_s, although for this size and starting point, we found runs where the training got stuck in a local minimum. However, we can also notice that the lower fidelity reflects in a higher loss. This analysis is shown for the ten mode case in Figure 5, and analogous results are obtained

TABLE 1 | Summary table of the output fidelity for the simulations with the planar layout. Average ($\overline{F}$) and minimum ($F_{\min}$) fidelity for the planar layout, for different number of modes and values of N_s. All results are averaged over 100 different runs, except for the 50 mode case, for which only *ten* different runs were performed. The starting fidelity for the training was of $F_{\text{in}} \sim 0.7$. For all results, except for the 50 mode one, a couple of the runs were discarded, based on a high loss at the end of training, such as the one shown in Figure 5.

m	N_s	$\overline{F}$	$F_{\min}$
10	10^3	0.9915	0.9249
10	10^4	0.9955	0.9073
10	10^5	0.9966	0.9009
20	2×10^3	0.9824	0.9008
20	2×10^4	0.9921	0.8664
20	2×10^5	0.9939	0.9113
50	5×10^4	0.9919	0.9637

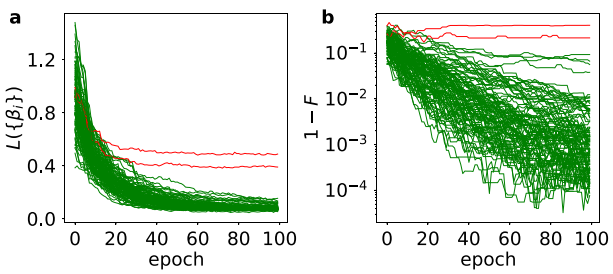

FIGURE 5 | Comparison between loss and fidelity for the ten-mode planar layout. (a) Loss and (b) infidelity for the ten-mode planar layout, with $N_s = 1 \times 10^4$. Each line corresponds to a different run. In the plots, we observe the presence of a few runs characterized by a value of the loss above a certain threshold, marked in red. These runs are also characterized by a low fidelity value. This highlights that runs characterized by low value of the fidelity can be identified from the loss values achieved during the training process, thus enabling the adoption of different techniques to avoid the evolution getting stuck in a local minimum.

in all considered system sizes. Hence, it is possible to recognize early during the process if the algorithm gets stuck in a local minimum. As a result, this permits the use of various techniques to correct the training, either stopping the training early and restarting, or shifting the parameters of a sufficient quantity to move outside the local minimum. We have performed additional analyses by fixing the starting fidelity to $F_{\text{in}} \sim 0.7$ for all circuit sizes. More specifically, we have then performed some numerical simulations in this condition for 20 and 50 modes, whose results are shown in Figure 4 and in Table 1.

These analyses show that the method is effective in reaching the target unitary even when scaling up to larger instances, although requiring increasing resources (N_s, number of epochs) with the system size. Furthermore, we observe that the chosen loss function is appropriate for the training process, since a reduction of the loss corresponds in parallel to an improvement in the fidelity.

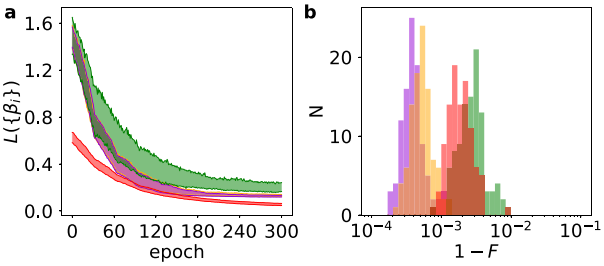

FIGURE 6 | Training process for 32-mode, 3D triangular layout interferometers, for different number of two-photon input combinations per epoch. (a) Loss and (b) infidelity for the training process. Input pairs for each epoch are chosen randomly from the list of first-neighbor pairs. For each tested number of input combinations per epoch, we report the results for 100 runs. Purple data sets: scenario with all 73 input pairs. Orange data sets: scenario with ten input pairs. Green data sets: scenario with one input pair. Red data sets: scenario by using only single-photon combinations. For the loss, the shaded zone represents the area between the 10th and 90th percentile of the runs. The histogram x axis is in logarithmic scale.

TABLE 2 | Summary table of the output fidelity for the simulations with the triangular layout with direct control. Average ($\overline{F}$) and minimum ($F_{\min}$) fidelity for the triangular layout with direct control, for different numbers of input pairs used per epoch. All results are averaged over 100 different instances, with a starting fidelity of $F \sim 0.7$.

# input pairs	$\overline{F}$	$F_{\min}$
73(all)	0.9996	0.9988
10	0.9994	0.9986
1	0.9969	0.9914
0	0.9980	0.9911

4.2 | 3D Layout with Direct Control on the Propagation Constants

We then proceeded to test numerically our method on interferometers with 3D triangular layout having $m = 32$ modes, and assuming direct control on the single propagation constants without cross-talks. As a first step, we performed some runs with $F_{\text{in}} \sim 0.7$ starting point. As discussed previously, in this scenario, we also tested the possibility to reduce the overhead required for the training procedure. More specifically, we tested how the method performed by reducing the number of two-photon input states combinations. Initially, we tested the scenario where, for each epoch, we used all 73 two-photon combinations corresponding to first-neighbor pairs, using $N_s = 3 \times 10^4$ samples per input, and performing the training process for 300 epochs. The results are shown in Figure 6. We observe that the training procedure provides promising results, reaching convergence on average after 200 epochs. We then proceeded to perform the training process by reducing the number of input two-photon pairs for each epoch. This is performed, as previously discussed, by randomly picking at each epoch a subset of the first-neighbor input combinations. The results over 100 runs are shown in Figure 6 and in Table 2. We find that, even with a limited number of ten input pairs per epoch, the training procedure

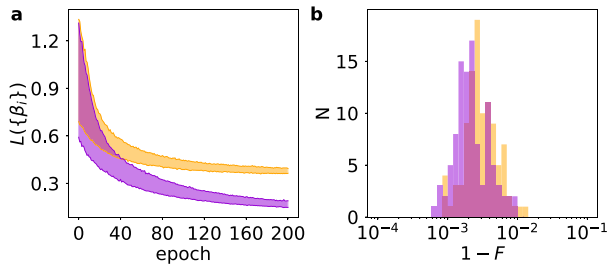

FIGURE 7 | Training process for 32-mode, 3D triangular layout interferometers, with multiple phase control. (a) Loss and (b) infidelity reached in the training process. The runs were done using only ten two-photon input configuration per epoch, chosen randomly from the set of first-neighbor mode pairs. Orange data sets: $N_s = 3 \times 10^3$ per input. Purple data sets: $N_s = 3 \times 10^4$ shots per input. For each tested configuration of number of samples N_s, we report the results for 100 runs. For the loss, the shaded zone represents the area between the 10th and 90th percentile of the runs. The histogram x axis is in logarithmic scale.

TABLE 3 | Summary table of the output fidelity for the simulations with the triangular layout with multiple phase control. Average ($\overline{F}$) and minimum ($F_{\min}$) fidelity for the triangular layout with multiple phase control, for different values of N_s. All results are averaged over 100 different instances, with starting propagation constants $\beta_i^{(0)}$ chosen randomly.

N_s	$\overline{F}$	$F_{\min}$
3×10^3	0.9966	0.9871
3×10^4	0.9974	0.9902

provides close performances to the case where all 73 combinations are used. However, it is worth noting that a limited number of two-photon combinations is required, since reducing to 1 or 0 two-photon pairs per epoch leads to significantly worse performances.

4.3 | 3D Layout with Multiple Phase Control

As the next step, we ran simulations by using the model, described above, to consider a scenario similar to the one of Ref. [25]. More specifically, we separate the chip in multiple sections, we place resistors on top of the device surface, and introduce the effect of thermal cross-talks due to heat propagation mechanisms. We performed 100 trainings with different starting parameters, starting from a completely random set of propagation constants β_i. Each training was carried out for 200 epochs. The loss and fidelity per epoch of the trainings are shown in Figure 7, while the obtained values of the fidelities are reported in Table 3. In this case, we observe that, with this kind of model, the training process allows us to obtain convergence regardless of the choice of starting propagation constants β_i, assuming that the target unitary is within the set achievable with the device structure. One of the reasons can be found in the presence of chip structure, where likely multiple parameter configurations can lead to the same target unitary. This potentially helps the training algorithm to find a path that leads to the desired output when starting from different position.

4.4 | 3D Layouts with Shifts in the Estimation of the Coupling Parameters

To complement this analysis taking into account the presence of small errors in the coupling estimations, we also did some simulations inserting shifts in the estimation of the coupling parameters $C_{i,j}$ as explained in the previous section (more details on this simulation procedure can be found in Appendix D). We did the tests for the ten mode planar layout, and for the 32 mode triangular layout, in the latter case considering both direct control of the propagation constants and multiple phase control. The results for the triangular layouts are shown in Figure 8 and in Tables 4 and 5. More specifically, we compare the performances in this scenario with the one assuming perfect knowledge on the coupling constants. We observe that, even with a moderate amount of error in the coupling constant knowledge, the training procedure still reaches high fidelity values, close to the ones achieved with $\{\beta_i^t\}$. In the case of the 32 mode with direct control of the propagation constants, we can notice that the trained parameters $\{\beta_i^f\}$ perform even slightly better than the target parameters $\{\beta_i^t\}$, when using the shifted coupling coefficients $\{C_{i,j}^s\}$. This indicates that, for this error regime, the training process within the tested layout was capable to compensate for characterization errors and find a unitary close to the target.

5 | Identifying the Optimization Starting Point

The variational method to train the integrated interferometers described above has been tested by considering a starting point which, in most of the cases, requires an initial knowledge on the circuit programming. One question that then arises is how one can find such a starting position. In some cases, where the optical chip can reproduce a limited range of unitaries, a random starting point can be in principle used since any programmed unitary will have a non-negligible fidelity with the target. However, when the interferometer spans a significant portion of the unitary space, it is necessary to develop additional methods that can be employed to find a suitable starting position.

Here, we consider the analysis of two specific approaches. The first one is based on training a classical machine learning model to obtain the starting parameters, by giving it as input the target unitary. The second approach relies on using the variational method to progressively search for a sequence of unitaries, starting from a random one, where the unitary at each step $k+1$ is progressively closer to the target one. A schematic view of the working principles for two approaches is shown in Figure 9. We tested these techniques on the 10 and 20 mode planar layouts. Both methods are shown to have their benefits and challenges, that we discuss below in this section.

5.1 | Classical Machine Learning Approach

We start by describing the classical machine learning method. In this approach, one constructs a neural network (NN) that takes as input a unitary, and gives as output a set of parameters that is as close as possible to those used to generate said unitary. As said, this method is implemented to provide the starting point for the variational local optimization algorithm described above.

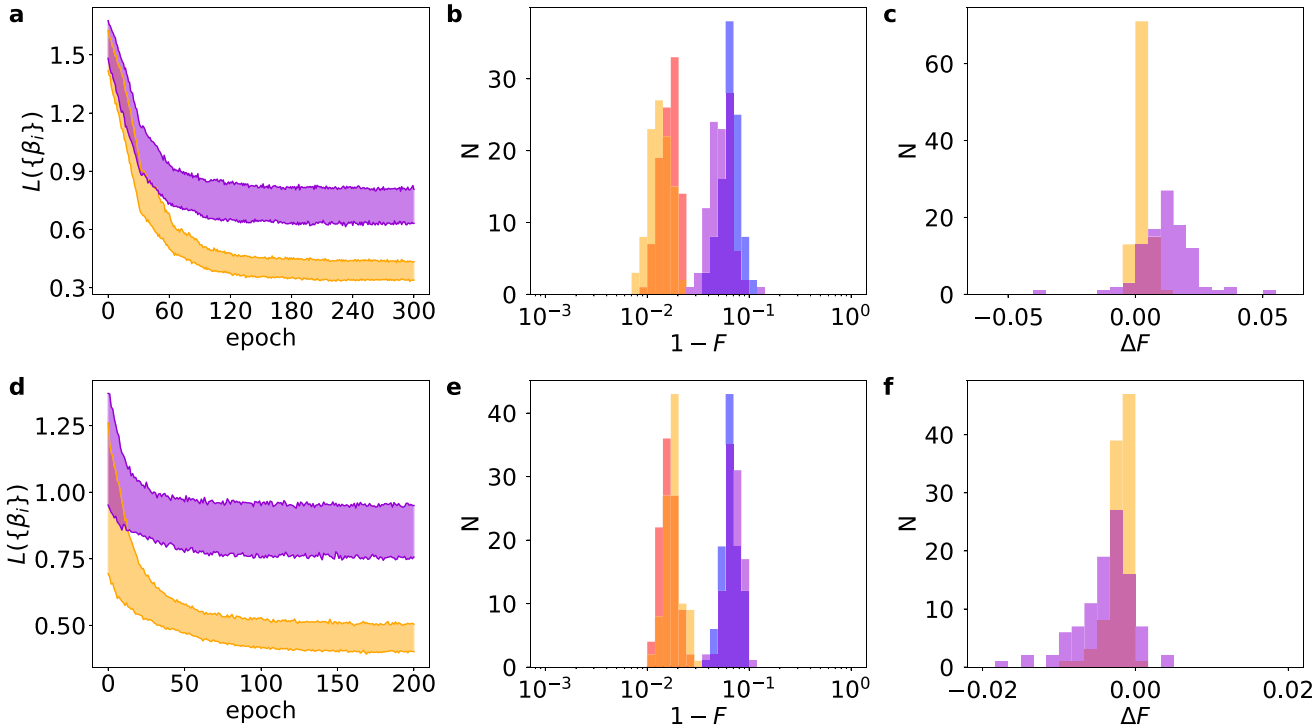

FIGURE 8 | Training process for 32-mode, 3D triangular layout interferometers, with different off-diagonal errors. Top row (a, b, c): simulations for the 3D layout with direct control. Bottom row (d, e, f): simulations for the 3D layout with multiple phase control. Orange data sets: $\delta C_{i,j} = 0.01$. Purple data sets: $\delta C_{i,j} = 0.02$. (a) and (d) Loss. For the training, the target unitary is calculated with the correct coupling coefficients $\{C^t_{i,j}\}$ and propagation constants $\{\beta^t_i\}$. Panels (b) and (e) compare the fidelity F_{base} reached when using the target parameters $\{\beta^t_i\}$ (in red and blue) with the fidelity F_{noise} reached when using the final trained parameters $\{\beta^f_i\}$ (in orange and purple), in the situation where the shifted coupling coefficients $\{C^s_{i,j}\}$ are used. The x axis is in logarithmic scale. Panels (c) and (f) compare the difference in fidelity $\Delta F = F_{\text{noise}} - F_{\text{base}}$. 100 runs were performed for each configuration. For more information, refer to Appendix D.

TABLE 4 | Summary table of the output fidelity for the simulations with the triangular layout with direct control in the case of imperfections in the estimation of $C_{i,j}$. Average fidelity using the target ($\overline{F}_{\text{base}}$) and trained ($\overline{F}_{\text{noise}}$) parameters while using the shifted values of $C_{i,j}$, as well as their difference $\Delta F = F_{\text{noise}} - F_{\text{base}}$, for different values of the shift $\delta C_{i,j}$. All results are averaged over 100 different instances, with a starting fidelity of $F_{\text{in}} \sim 0.7$. For more information, refer to App. D.

$\delta C_{i,j}$	$\overline{F}_{\text{base}}$	$\overline{F}_{\text{noise}}$	$\overline{\Delta F}$
0.01	0.9835	0.9863	0.0028
0.02	0.9335	0.9456	0.0121

TABLE 5 | Summary table of the output fidelity for the simulations with the triangular layout with multiple phase control in the case of imperfections in the estimation of $C_{i,j}$. Average fidelity using the target ($\overline{F}_{\text{base}}$) and trained ($\overline{F}_{\text{noise}}$) parameters while using the shifted values of $C_{i,j}$, as well as their difference $\Delta F = F_{\text{noise}} - F_{\text{base}}$, for different values of the shift $\delta C_{i,j}$. All results are averaged over 100 different instances, with starting propagation constants $\beta^{(0)}_i$ chosen randomly. For more information, refer to Appendix D.

$\delta C_{i,j}$	$\overline{F}_{\text{base}}$	$\overline{F}_{\text{noise}}$	$\overline{\Delta F}$
0.01	0.9836	0.9816	−0.0020
0.02	0.9332	0.9294	−0.0038

For this reason, for this initial step we consider the training to be successful if the fidelity between the input unitary and the one corresponding to the network output parameter satisfies $F > 0.69$. This bound is chosen since this fidelity threshold is found to be sufficient to provide a reasonable starting point for the local optimization.

In all of the trainings of the classical neural network, the input set is obtained by generating a single set of coupling coefficients $\{C_{i,j}\}$, which are common for all generated unitaries for that specific training set, and then selecting a random set of propagation constants $\{\beta_i\}$, different for each generated unitary. The input unitary is then computed from this set of parameters.

Unless otherwise stated, the train-test split was 0.8, meaning that 80% of the input unitaries were used for training, and 20% for testing. Further details on the structure and training of the neural network are reported in Appendix E.

As a first analysis, we applied this approach to the ten-mode scenario with planar layout. In particular, we tested the performances as a function of the number of unitaries shown to the NN during the training process. The results of this analysis are reported in Table 6. The performances of the trained network are tested by retrieving the circuit parameters for a distinct set of 1000 unitaries, employed as the test set and common to all

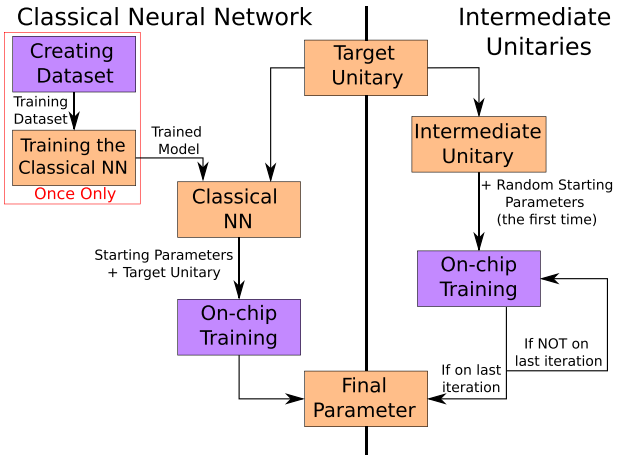

FIGURE 9 | Schematic of the two different methods to define the starting point for our training algorithm. Logical sequence of operations to implement the two methods described in the main text. In purple we highlight the parts that are performed on the optical chip, while in orange are shown the parts performed on classical hardware.

TABLE 6 | Summary table of the success rate of the trained neural network model for the ten-mode planar layout. Success rate and average fidelity of the trained neural network model for the ten mode planar layout with respect to the number and type of unitaries used during training. The values are obtained by testing on 1000 random unitaries, which are the same for all configurations tested.

# unitaries in the training set	Success rate	$\overline{F}$
100	8.9%	0.4049
1000	66.7%	0.7246
10000	98.3%	0.9350
10000 (noisy)	96.7%	0.9060

trainings. We then calculated the success rate, defined as the frequency where a fidelity $F > 0.69$ between the target unitary and the one obtained using the network output. We observed that, for a training process with 10000 unitaries, a success rate of 98% is obtained on the test set. Then, we selected a subset of 100 unitaries, and used the output of the network as a starting point for the local optimization algorithm described in the previous section. We observe that 97% of these unitaries reach a fidelity $F > 0.98$ after the complete training process.

As a further test for this classical neural network method, we also tested its resilience to noise. This was performed by using in the training set noisy unitaries, thus reflecting the scenario where a finite number of measurements is performed to characterize the unitaries of the training set. Robustness to this kind of noisy also suggests that less measurements need to be performed for the unitary characterization in this initial stage, thus lowering the resource requirements. Noisy unitaries are simulated by starting from the set of 10000 noiseless unitaries used previously, and applying to each unitary U_i in the set a random noise term obtained as:

$$U^{\text{noise}} = e^{i\epsilon H} \tag{3}$$

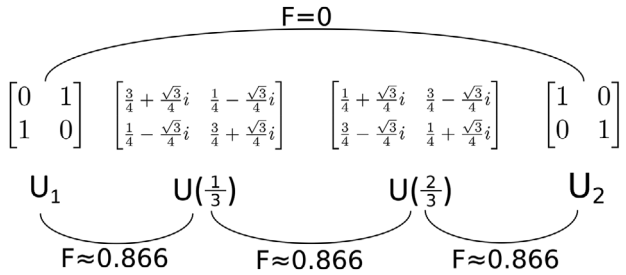

FIGURE 10 | Example of the intermediate unitaries method applied on a 2x2 unitary matrix. The fidelity F between the starting unitary U_1 and the target one U_2 is small, thus leading to reduced performance of the training algorithm. This can be circumvented by generating intermediate unitaries, that permits us to divide the training process in steps with targets that have a sufficient value of the fidelity for the method to be effective.

where ϵ is a parameter quantifying the noise strength, and H is a random Hermitian matrix. The noisy unitary is found as $U_i^{\text{noisy}} = U_i U^{\text{noise}}$. In our case, the noise parameter was set to $\epsilon = 0.1$, corresponding to an average fidelity $\overline{F} \sim 0.95$ between the noisy and noiseless unitaries. After training the network with the noisy unitary, we used a test sample with noiseless unitaries to check its performance, and obtained a success rate of 96%, which shows only a limited decrease in performance of the trained model (see Table 6).

We have then repeated a similar analysis for the 20-mode planar layout, corresponding to an increased size system. In this case, a training set composed of 40000 input unitaries leads to a success rate of $\sim 75\%$, while increasing the size of the training set to 400000 provides a success rate of $\sim 98\%$.

5.2 | Training Through Intermediate Unitaries

As a second approach to obtain the starting point for the local optimization, we considered a different approach based on dividing the optimization in multiple steps. The idea is to perform a progressive training through multiple steps having as intermediate targets a set of unitaries, which are progressively closer to the target one, chosen according to an appropriate metric. This set of intermediate unitaries can be generated by using the geodesic [59]. More specifically, let us call U_1 and U_2 the starting and target unitaries, respectively. Then, we can define:

$$\Gamma = U_1^\dagger U_2 \tag{4}$$

and obtain:

$$U(t) = U_1 e^{t \log \Gamma} \tag{5}$$

The parameter t identifies the position of a given unitary $U(t)$ in the sequence, with extremal points the unitaries U_1 for $t = 0$ and U_2 for $t = 1$, and between them for values of t between 0 and 1. A schematic view of this approach applied on a 2x2 unitary is shown in Figure 10.

For the intermediate unitaries method, in the case of the ten mode planar layout we used 3 steps, that is, 2 intermediate unitaries

before final target unitary. We found that, using the same 100 sample set that we used to test the classical machine learning method, only around 45% of the trainings resulted in a final fidelity $F > 0.98$. The reason for this behavior can be found in that the intermediate unitaries can end up being outside of the range of unitaries that the actual device can reach, given the specific architecture. This would lead to the training process for that step failing, and as a result impacting the whole training procedure. We argue that the efficacy of this method can improve as larger sets of unitaries can be obtained within the circuit architecture. Considering this issue, in the case of the 20 mode layout the performances of the approach worsen, as even using 5 steps the training failed to converge in the majority of the cases.

5.3 | Comparison of the Two Approaches

Both methods presented have their advantages and disadvantages. In the case of the method, which uses classical machine learning to obtain the starting parameters, one of the main disadvantages is that it requires an overhead of starting unitaries associated to the respective parameters to perform the training. This means that, on a physical device, a measurement overhead to perform the on-chip static characterization of the unitaries is present. However, once trained the neural network is capable of providing a reliable set of starting parameters in short computational time. On the other hand, the intermediate unitaries method does not require significant overheads. However, this approach requires a longer training process, since it effectively requires training the chip multiple times with different targets. Furthermore, the intermediate unitaries has a significant failure rate for those architectures where the set of unitaries achievable by the device is limited.

6 | Discussion

In this work, we have presented a machine learning-based approach to program multi-mode integrated interferometers based on continuously coupled waveguides. One of the main features of this method is the capability of performing the desired task without requiring extensive characterization of the integrated device. Furthermore, this approach removes the need for performing an accurate modeling of the circuit operations, since the interferometers are treated as black boxes. This makes this technique flexible and capable to be used with circuits of different layouts. We have shown via numerical simulations the effectiveness of this approach for different circuit layouts and its robustness to experimental errors. The results of the simulations support the capability to program integrated devices to implement a desired unitary transformation via the developed approach, provided that the target unitary can be reached by tuning the circuit parameters. Finally, to complement this approach, we have also presented possible methods to find the starting point of the optimization process.

Using the method described above, an estimate of the time required for the training process is of approximately ∼10 h for the 20-mode planar layout, and ∼15 h for the 32-mode 3D configuration, assuming a rate of 10^5 twofold coincidences per minute, using $N_s = 10^4$ and using 10 two-photon input states at each epoch. This has to be compared with the significantly higher cost required to perform the calibration with other approaches involving the reconstruction of the unitary transformations for several parameter values. It must also be noted that this time estimate is a baseline, as further improvements to the training process can be made, such as using a smaller number of shots per input N_s during the beginning of the training, and then increasing the value of N_s as training progresses, which can lead to additional speedups.

A currently open problem in quantum machine learning is how to efficiently train large quantum circuits. Our method provides a step forward in that direction, providing several optimizations to reduce the time required to train the circuits, leading to a training time that is expected to scale polynomially. Room for improvement can be found for instance in the presence of a reduced success rate when the starting fidelity is below a certain threshold, which we were able to partially mitigate with the use of classical machine learning. Thus, future research directions can be found in devising optimizations of the method to more efficiently train the circuits when their size increases. These improvements include finding ways to optimize multiple parameters simultaneously and implementing solutions to support the training process when starting from an initial set of parameters that is far from the target parameters.

The obtained results show that the devised method could be useful in a scenario where an accurate modeling of the devices is not available, while it is also applicable in all cases where a comprehensive characterization via standard means requires an exceedingly amount of resources. Hence, this approach is expected to be used as an effective and viable tool to program and operate reconfigurable integrated interferometers with various layouts.

Acknowledgements

This work was supported by the European Union via project QLASS ("Quantum Glass-based Photonic Integrated Circuits" - Grant Agreement No. 101135876) and by the ERC Advanced Grant QU-BOSS (QUantum advantage via non-linear BOSon Sampling, Grant Agreement No. 884676). D.S. acknowledges Thales Alenia Space Italia for supporting the PhD fellowship.

Conflicts of Interest

The authors declare no conflicts of interest.

Data Availability Statement

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

References

1. F. Arute, K. Arya, R. Babbush, et al., "Quantum Supremacy Using a Programmable Superconducting Processor," *Nature* 574 (2019): 505–510, https://doi.org/10.1038/s41586-019-1666-5.

2. H.-S. Zhong, H. Wang, Y.-H. Deng, et al., "Quantum Computational Advantage Using Photons," *Science* 370 (2020): 1460, https://doi.org/10.1126/science.abe8770.

3. H.-S. Zhong, Y.-H. Deng, J. Qin, et al., "Phase-Programmable Gaussian Boson Sampling Using Stimulated Squeezed Light," *Physical Review Letters* 127 (2021): 180502, https://doi.org/10.1103/PhysRevLett.127.180502.

4. Y. Wu, W.-S. Bao, S. Cao, et al., "Strong Quantum Computational Advantage Using a Superconducting Quantum Processor," *Physical Review Letters* 127 (2021): 180501, https://doi.org/10.1103/PhysRevLett.127.180501.

5. L. S. Madsen, F. Laudenbach, M. Falamarzi-Askarani, et al., "Quantum Computational Advantage with a Programmable Photonic Processor," *Nature* 606 (2022): 75, https://doi.org/10.1038/s41586-022-04725-x.

6. Y.-H. Deng, Y.-C. Gu, H.-L. Liu, et al., "Gaussian Boson Sampling with Pseudo-Photon-Number-Resolving Detectors and Quantum Computational Advantage," *Physical Review Letters* 131 (2023): 150601, https://doi.org/10.1103/PhysRevLett.131.150601.

7. Y. Kim, A. Eddins, S. Anand, et al., "Evidence for the Utility of Quantum Computing Before Fault Tolerance," *Nature* 618 (2023): 500, https://doi.org/10.1038/s41586-023-06096-3.

8. R. Acharya, D. A. Abanin, L. Aghababaie-Beni, et al., "Quantum Error Correction Below the Surface Code Threshold," *Nature* 638 (2024): 920–926, https://doi.org/10.1038/s41586-024-08449-y.

9. J. Wang, F. Sciarrino, A. Laing, and M. G. Thompson, "Integrated Photonic Quantum Technologies," *Nature Photonics* 14 (2019): 273–284, https://doi.org/10.1038/s41566-019-0532-1.

10. P. P. Rohde, "Simple Scheme for Universal Linear-Optics Quantum Computing with Constant Experimental Complexity Using Fiber Loops," *Physical Review A* 91 (2015): 012306, https://doi.org/10.1103/PhysRevA.91.012306.

11. B. Bartlett, A. Dutt, and S. Fan, "Deterministic Photonic Quantum Computation in a Synthetic Time Dimension," *Optica* 8 (2021): 1515, https://doi.org/10.1364/OPTICA.424258.

12. P. C. Humphreys, B. J. Metcalf, J. B. Spring, et al., "Linear Optical Quantum Computing in a Single Spatial Mode," *Physical Review Letters* 111 (2013): 150501, https://doi.org/10.1103/PhysRevLett.111.150501.

13. L. Carosini, V. Oddi, F. Giorgino, et al., "Programmable Multiphoton Quantum Interference in a Single Spatial Mode," *Science Advances* 10 (2024): eadj0993, https://doi.org/10.1126/sciadv.adj0993.

14. F. Flamini, N. Spagnolo, and F. Sciarrino, "Photonic Quantum Information Processing: A Review," *Reports on Progress in Physics* 82 (2018): 016001, https://doi.org/10.1088/1361-6633/aad5b2.

15. K. Fukui and S. Takeda, "Building a Large-Scale Quantum Computer with Continuous-Variable Optical Technologies," *Journal of Physics B: Atomic, Molecular and Optical Physics* 55 (2022): 012001, https://doi.org/10.1088/1361-6455/ac489c.

16. D. Gottesman, A. Kitaev, and J. Preskill, "Encoding a Qubit in an Oscillator," *Physical Review A* 64 (2001): 012310, https://doi.org/10.1103/PhysRevA.64.012310.

17. W.-Q. Liu and H.-R. Wei, "Linear Optical Universal Quantum Gates with Higher Success Probabilities," *Advanced Quantum Technologies* 6 (2023): 2300009, https://doi.org/10.1002/qute.202300009.

18. S. Bartolucci, P. Birchall, H. Bombín, et al., "Fusion-Based Quantum Computation," *Nature Communications* 14 (2023): 912, https://doi.org/10.1038/s41467-023-36493-1.

19. Y. Zhan, H. Zhang, R. Erbanni, et al., "Loop Quantum Photonic Chip for Coherent Multi-Time-Step Evolution," arXiv:2411.11307 [quant-ph] (2024), https://arxiv.org/abs/2411.11307.

20. S. Paesani, Y. Ding, R. Santagati, et al., "Generation and Sampling of Quantum States of Light in a Silicon Chip," *Nature Physics* 15 (2019): 925–929, https://doi.org/10.1038/s41567-019-0567-8.

21. M. Reck, A. Zeilinger, H. J. Bernstein, and P. Bertani, "Experimental Realization of Any Discrete Unitary Operator," *Physical Review Letters* 73 (1994): 58, https://doi.org/10.1103/PhysRevLett.73.58.

22. W. R. Clements, P. C. Humphreys, B. J. Metcalf, W. S. Kolthammer, and I. A. Walmsley, "Optimal Design for Universal Multiport Interferometers," *Optica* 3 (2016): 1460, https://doi.org/10.1364/OPTICA.3.001460.

23. T. Giordani, F. Hoch, G. Carvacho, N. Spagnolo, and F. Sciarrino, "Integrated Photonics in Quantum Technologies," *Rivista del Nuovo Cimento* 46 (2023): 71, https://doi.org/10.1007/s40766-023-00040-x.

24. A. Peruzzo, M. Lobino, J. C. F. Matthews, et al., "Quantum Walks of Correlated Photons," *Science* 329 (2010): 1500, https://doi.org/10.1126/science.1193.

25. F. Hoch, S. Piacentini, T. Giordani, et al., "Reconfigurable Continuously-Coupled 3D Photonic Circuit for Boson Sampling Experiments," *npj Quantum Information* 8 (2022): 35, https://doi.org/10.1038/s41534-022-00568-6.

26. Y. Yang, R. J. Chapman, B. Haylock, et al., "Programmable High-Dimensional Hamiltonian in a Photonic Waveguide Array," *Nature Communications* 15 (2024): 50, https://doi.org/10.1038/s41467-023-44185-z.

27. Y. Yang, R. J. Chapman, A. Youssry, et al., "Programmable Quantum Circuits in a Large-Scale Photonic Waveguide Array," *npj Quantum Information* 11 (2025): 19, https://doi.org/10.1038/s41534-024-00934-6.

28. W.-H. Zhou, X.-W. Wang, R.-J. Ren, et al., "Multi-Particle Quantum Walks on 3D Integrated Photonic Chip," *Light: Science & Applications* 13 (2024): 296, https://doi.org/10.1038/s41377-024-01627-7.

29. Z.-Q. Jiao, J. Gao, W.-H. Zhou, et al., "Two-Dimensional Quantum Walks of Correlated Photons," *Optica* 8 (2021): 1129, https://doi.org/10.1364/OPTICA.425879.

30. K. Poulios, R. Keil, D. Fry, et al., "Quantum Walks of Correlated Photon Pairs in Two-Dimensional Waveguide Arrays," *Physical Review Letters* 112 (2014): 143604, https://doi.org/10.1103/PhysRevLett.112.143604.

31. D. N. Biggerstaff, R. Heilmann, A. A. Zecevik, et al., "Enhancing Coherent Transport in a Photonic Network Using Controllable Decoherence," *Nature Communications* 7 (2016): 11282, https://doi.org/10.1038/ncomms11282.

32. F. Caruso, A. Crespi, A. G. Ciriolo, F. Sciarrino, and R. Osellame, "Fast Escape of a Quantum Walker from an Integrated Photonic Maze," *Nature Communications* 7 (2016): 11682, https://doi.org/10.1038/ncomms11682.

33. K. Albertsson, P. Altoe, D. Anderson, et al., "Machine Learning in High Energy Physics Community White Paper," *Journal of Physics: Conference Series* 1085 (2018): 022008, https://doi.org/10.1088/1742-6596/1085/2/022008.

34. G. Litjens, T. Kooi, B. E. Bejnordi, et al., "A Survey on Deep Learning in Medical Image Analysis," *Medical Image Analysis* 42 (2017): 60–88, https://doi.org/10.1016/j.media.2017.07.005.

35. A. Vaswani, N. Shazeer, N. Parmar, et al., "Attention Is All You Need," in *Advances in Neural Information Processing Systems*, eds. I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, vol. 30 (Curran Associates, Inc., 2017), https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

36. DeepSeek-AI: X. Bi, D. Chen, G. Chen, et al., "DeepSeek LLM: Scaling Open-Source Language Models with Long-Termism," arXiv:2401.02954 [cs.CL] (2024), https://arxiv.org/abs/2401.02954.

37. A. I. Khan and S. Al-Habsi, "Machine Learning in Computer Vision," *Procedia Computer Science* 167 (2020): 1444, https://doi.org/10.1016/j.procs.2020.03.355.

38. D. Stanev, N. Spagnolo, and F. Sciarrino, "Deterministic Optimal Quantum Cloning via a Quantum-Optical Neural Network," *Physical Review Research* 5 (2023): 013139, https://doi.org/10.1103/PhysRevResearch.5.013139.

39. G. R. Steinbrecher, J. P. Olson, D. Englund, and J. Carolan, "Quantum Optical Neural Networks," *npj Quantum Information* 5 (2019): 60, https://doi.org/10.1038/s41534-019-0174-7.

40. Y. Zhang and Q. Ni, "Recent Advances in Quantum Machine Learning," *Quantum Engineering* 2 (2020): e34, https://doi.org/10.1002/que2.34.

41. S. Bordoni, D. Stanev, T. Santantonio, and S. Giagu, "Long-Lived Particles Anomaly Detection with Parametrized Quantum Circuits," *Particles* 6 (2023): 297–311, https://doi.org/10.3390/particles6010016.

42. B. Bartlett and S. Fan, "Universal Programmable Photonic Architecture for Quantum Information Processing," *Physical Review A* 101 (2020): 042319, https://doi.org/10.1103/PhysRevA.101.042319.

43. A. Suprano, D. Zia, L. Innocenti, et al., "Photonic Quantum Extreme Learning Machine," in *Quantum 2.0 Conference and Exhibition* (Optica Publishing Group, 2024), QW4A.2, https://doi.org/10.1364/QUANTUM.2024.QW4A.2.

44. F. Hoch, E. Caruccio, G. Rodari, et al., "Quantum Machine Learning with Adaptive Boson Sampling via Post-Selection," *Nature Communications* 16 (2025): 902, https://doi.org/10.1038/s41467-025-55877-z.

45. K. H. Nielsen, Y. Wang, E. Deacon, et al., "Programmable Nonlinear Quantum Photonic Circuits," arXiv:2405.17941 [quant-ph] (2024), https://arxiv.org/abs/2405.17941.

46. H. Tang, L. Banchi, T.-Y. Wang, et al., "Generating Haar-Uniform Randomness Using Stochastic Quantum Walks on a Photonic Chip," *Physical Review Letters* 128 (2022): 050503, https://doi.org/10.1103/PhysRevLett.128.050503.

47. J. Whitfield, C. A. Rodriguez-Rosario, and A. Aspuru-Guzik, "Quantum Stochastic Walks: A Generalization of Classical Random Walks and Quantum Walks," *Physical Review A* 81 (2010): 022323, https://doi.org/10.1103/PhysRevA.81.022323.

48. N. Lambert, E. Giguère, P. Menczel, et al., "QuTiP 5: The Quantum Toolbox in Python," arXiv:2412.04705 [quant-ph] (2024), https://arxiv.org/abs/2412.04705.

49. N. Heurtel, A. Fyrillas, G. D. Gliniasty, et al., "Perceval: A Software Platform for Discrete-Variable Photonic Quantum Computing," *Quantum* 7 (2023): 931, https://doi.org/10.22331/q-2023-02-21-931.

50. H. Robbins and S. Monro, "A Stochastic Approximation Method," *Annals of Mathematical Statistics* 22 (1951): 400–407, https://doi.org/10.1214/aoms/1177729586.

51. J. Kiefer and J. Wolfowitz, "Stochastic Estimation of the Maximum of a Regression Function," *Annals of Mathematical Statistics* 23 (1952): 462–466, https://doi.org/10.1214/aoms/1177729392.

52. S.-I. Amari, "Backpropagation and Stochastic Gradient Descent Method," *Neurocomputing* 5 (1993): 185–196, https://doi.org/10.1016/0925-2312(93)90006-o.

53. A. Laing and J. L. O'Brien, "Super-Stable Tomography of Any Linear Optical Device," arXiv:1208.2868 [quant-ph] (2012), https://arxiv.org/abs/1208.2868.

54. M. Tillmann, C. Schmidt, and P. Walther, "On Unitary Reconstruction of Linear Optical Networks," *Journal of Optics* 18 (2016): 114002, https://doi.org/10.1088/2040-8978/18/11/114002.

55. N. Spagnolo, E. Maiorino, C. Vitelli, et al., "Learning an Unknown Transformation via a Genetic Approach," *Scientific Reports* 7 (2017): 14316, https://doi.org/10.1038/s41598-017-14680-7.

56. F. Hoch, T. Giordani, N. Spagnolo, A. Crespi, R. Osellame, and F. Sciarrino, "Characterization of Multimode Linear Optical Networks," *Advanced Photonics Nexus* 2 (2023): 016007, https://doi.org/10.1117/1.APN.2.1.016007.

57. A. Abbas, R. King, H.-Y. Huang, et al., "On Quantum Backpropagation, Information Reuse, and Cheating Measurement Collapse," arXiv:2305.13362 [quant-ph] (2023), https://arxiv.org/abs/2305.13362.

58. M. Chau and M. C. Fu, "An Overview of Stochastic Approximation," in *Handbook of Simulation Optimization*, ed. M. C. Fu (Springer, New York, NY, 2015), 149–178, https://doi.org/10.1007/978-1-4939-1384-8_6.

59. D. Lewis, R. Wiersema, J. Carrasquilla, and S. Bose, "Geodesic Algorithm for Unitary Gate Design with Time-Independent Hamiltonians," arXiv:2401.05973 [quant-ph] (2025), https://arxiv.org/abs/2401.05973.

60. C. Lee, H. Hasegawa, and S. Gao, "Complex-Valued Neural Networks: A Comprehensive Survey," *IEEE/CAA Journal of Automatica Sinica* 9 (2022): 1406, https://doi.org/10.1109/JAS.2022.105743.

61. D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs)," arXiv:1511.07289 [cs.LG] (2016), https://arxiv.org/abs/1511.07289.

62. D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv:1412.6980 [cs.LG] (2017), https://arxiv.org/abs/1412.6980.

63. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research* 15 (2014): 1929–1958, https://doi.org/10.5555/2627435.2670313.

64. S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, "Don't Decay the Learning Rate, Increase the Batch Size," arXiv:1711.00489 [cs.LG] (2017), https://arxiv.org/abs/1711.00489.

Appendix A: Simulating the Continuously Coupled Chip

In this appendix, we provide more details on the methodology used to simulate the evolution in multi-mode interferometers based on continuously-coupled waveguides.

General structure of the simulation process. – As discussed in the main text, the overall structure to simulate the multi-mode interferometer is composed of different steps. The starting point is the definition of the circuit structure and thus of the Hamiltonian, by specifying the propagation constants β_i and the coupling coefficients $C_{i,j}$ between the modes, and of the interaction length z. Then, by using the Qutip [48] library we retrieve the unitary matrix U corresponding to the overall evolution of the interferometer. Such a unitary is then used to simulate samples of N_s events obtained at the output of the interferometer by using the Perceval [49] library. Regarding the definition of the Hamiltonian and of the corresponding unitary matrix, they are defined by the number of modes, by the internal structure of the circuit and by the interaction length.

After we generate the starting parameters of the simulated chips, we take a copy of the tunable parameters $\{\beta_i^t\}$, and shift their values randomly from a uniform distribution. More specifically, the size of this shift depends on the size and mesh of the circuit, and on the required starting fidelity. For instance, in the case of the ten-mode planar layout and of the 32-mode 3D layout, we use a random shift between ± 0.1 mm^{-1} on each parameter. With this process we obtain the starting point for the trainable parameters $\{\beta_i^{(0)}\}$. This process is iterated until the starting fidelity is obtained in the required range.

We detail below the different cases analyzed in the main text.

Integrated interferometers with planar layout. – For the circuits with planar layout, the length of the circuit has been set as being $2m$ mm, with m being the number of modes. For the propagation constants, we generate a number of parameters equal to the number of modes, each one corresponding to a propagation constant β_i, and we generate them by sampling from a random uniform distribution between 0.7 and 1.3 mm^{-1}. Then we generate a set of coupling coefficients, with a number of parameters defined by the design and size of the chip, generating them from a random distribution between 0.1 and 0.3 mm^{-1}. This was chosen according to the values reported in recent implementations [25], where the coupling coefficients were chosen around 0.2 mm^{-1}.

3D interferometers with direct control. – For the circuits with the 3D layout we set the length of the circuit to be 36 mm, following the specification

of the 3D circuits implemented in Ref. [25]. The main differences when compared to the planar layout are that the Hamiltonian has a different structure, and that the coupling constants correspond to a different structure for the neighbor modes. Thus, the simulation process has been adapted to take into account these two differences with respect to the planar layout.

3D interferometers with multiple control. – For the circuits with the 3D layout we set the length of the circuit to be 36 mm, following the specification of the 3D circuits implemented in Ref. [25]. The simulation process of this layout is modified with respect the other cases to reproduce the behaviour of this kind of architecture.

As already stated in the main text, the first modification requiring considering a layout with multiple connected segments, which can be controlled independently. Then, we changed the way in which the parameters are controlled, no longer being able to control the propagation constants individually, but instead including the effect of thermal cross-talks. The latter effect correspond to having multiple propagation constants affected simultaneously by changing the current in a single resistor, which is due to heat propagation mechanism in the glass.

To reproduce the structure of the interferometers with this layout, we divide the simulated chip into 18 separate segments, each of length 2 mm. In eight of these segments, thermal phase shifters are inserted to modulate the propagation constants. More specifically, we alternate segments without active control and segments with phase shifters. For each segment where active control is available, we train only 2 parameters, which are associated with the currents I_k of the two available resistors. Each of these parameters then influences the propagation constants $\{\beta_i\}$. In our specific case, we start with a fixed base value for the propagation constants, chosen to be 0.7 mm^{-1}. Then, for each propagation constant we add an amount equal to $\log(d_{x_i})x_i$, where x_i is the value of the parameter, and d_{x_i} is the lattice distance between the mode and the resistor associated with such parameter. We implement this as a fixed multiplicative mesh, so that it can be altered to suit different chip structures. Regarding the value of x_i, we choose its value to be restricted between 0 and 0.6 mm^{-1}. In such a way, the final value of the propagation constants of the modes closest to the resistors is between 0.7 and 1.3 mm^{-1}, thus reproducing the range used for the previously considered architectures.

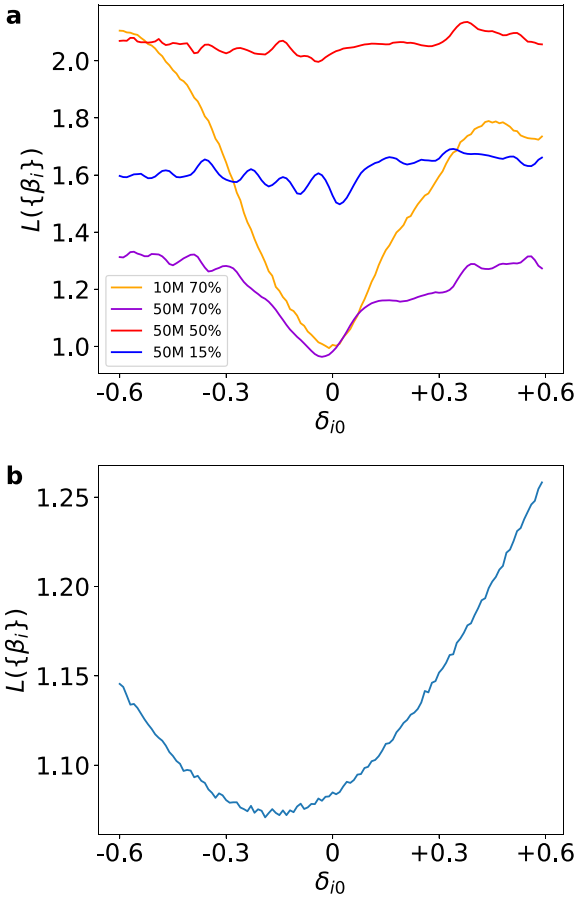

FIGURE B1 | Loss landscape when changing a single parameter β_{i0} in the training process. Loss change as a function of the parameter shift with respect to the correct value. (a) Loss landscape for the planar structure. Orange curve: $m = 10$ and $F_{in} \sim 0.7$. Purple curve: $m = 50$ and $F_{in} \sim 0.7$. Blue curve: $m = 50$ and $F_{in} \sim 0.5$. Red curve: $m = 50$ and $F_{in} \sim 0.15$. (b) Loss landscape for the 32-mode 3D layout according to Ref. [25].

Appendix B: Loss Landscape

To get some additional insights on the training process, we analyzed the loss landscape when tuning only a single parameter.

This analysis is performed by choosing a single propagation constant, identified by a given index k. Then, we set $\beta_k^{(0)}$ and β_k^t in the target and trainable set to the same value, placed in the middle of the range of possible values. This means that such parameter is set to 1.0 mm^{-1} in the case with direct control, and 0.3 mm^{-1} in the case with multiple control. We then evaluate the loss by tuning parameter $\beta_k^{(0)}$ in the range ± 0.6 mm^{-1} with respect to the central value, equal to twice the range used in the training process. This analysis provide an intuition of how the loss varies by changing a single parameter, as performed in a single step of the training process, by showing the emergence of eventual local or global minima.

As shown in Figure B1a for specific instances of interferometers with planar layout, we observe two different behaviors depending from the starting point of the optimization. More specifically, when the starting point has an initial fidelity $F_{\text{in}} \sim 0.7$, we observe the emergence of a global minimum of the loss when the parameter is equal to the target β_k^t. Conversely, for lower starting fidelities the loss landscape shows the presence of different local minima, and thus the training process is characterized by higher failure probability. This highlights the need of choosing appropriately the starting fidelity for the training process, as discussed in Section 4.

In Figure B1b, we report a similar analysis for the 3D interferometers with multiple phase control. In this case, the loss landscape of the single parameter has its global minima at a different value than the target β_k^t. This is likely due to the feature that, with this layout, multiple configurations can generate the same unitary. This also provides an intuition on the possibility of starting the training of interferometers with this layout from a random set of parameters, since it is likely that from any starting position, there is a close configuration that leads to the desired unitary. We can also notice that the curve of the loss shows one well defined minimum, thus helping the training process.

Appendix C: Details on the Approximation of the Gradient and on the Training

In this subsection, we will give some additional details on our modified FDSA and overall training method.

The choice of the parameter to be trained is performed in the following way. The parameters are organized in an ordered list, and we proceed changing the parameter at each epoch in sequence. Then, once the full set of parameters have been used, the list is randomly permuted. Then we

proceed in sequence changing the parameter following the shuffled list. This appraoch is then iterated until the training reaches convergence. This method, while retaining a degree of randomness in the choice, ensures at the same time that no parameter is left untouched for too long.

The training is performed then by shifting the value of the chosen parameter by a certain amount δ, and by evaluating how the loss changes with the shifted value. The choice of δ is performed as a compromise between two different trends. More specifically, reducing the value of δ permits to obtain a better approximation of the gradient, while at the same time it increases vulnerability to sampling noise as well as other forms of noise. The value of δ changes based on the size and on the interferometer layout. In our case, we tested various values and found out the one that simultaneously minimizes both the loss and the probability of the training getting stuck in a local minimum. Furthermore, the used value of δ is reduced as the loss decreases. Adapting this values permits to obtain faster convergence rate around the global minima at the beginning of the training. As an added advantage, we found that this approach reduces the risk of getting stuck in a narrow local minima during the training process.

Given a specific choice of δ, the training step involves shifting the value by $\pm\delta$, comparing the obtained losses with the one obtained when the parameter is left unchanged, and then moving in the direction where the loss is lower. If neither direction has lower loss, we do not change the value of that parameter. This takes three total evaluations per epoch, instead of the usual two needed by FDSA, while reducing the risk of obtaining an increase in the loss during that step of the process.

Once the correct direction for the parameter change has been identified, one then needs to choose the effective strength of the parameter shift to move to the next training step. To this end, we tried two approaches. The first method involves adding a change in the parameter in the direction where the loss decreases, by an amount proportional to the approximated gradient. The second approach corresponds to using the value of $\pm\delta$, already evaluated previously to identify the direction, corresponding to the lower value of the loss. This second method also allows the use of only two evaluations per epoch, instead of three, since we already know the loss of the unaltered value from the previous epoch. In general, we found out that the first method leads to a larger reduction rate of the loss when far away from the target point. Conversely, the second method permits to converge to a lower loss value when the parameter is close to the target, and removed the risk of increasing the loss from one epoch to the next. Given that we start from $F_{\text{in}} \sim 0.7$, we privileged the use of the second method. Further optimization could be possible by using the first method for the initial part of training, and then switching to the second method for the second part of training. Considering the involved parameter range, we expect that in the investigated scenarios this would lead to a marginal improvement in the algorithm efficiency.

Finally, the complete training process consists of multiple epochs that follow the procedure described above. The specific number of epochs changes depending on the structure and on the number of modes of the optical interferometer.

As a further direction for improvement to be used for interferometers with a large number of parameters and/or a lower starting fidelity, a possible approach involves performing the initial part of the training with SPSA, to obtain a first approximation of the target, and then use our modified FDSA to perform the second part fo the optimization process.

Appendix D: Details on the Simulation with Shifts in the Coupling Parameters

In this subsection, we will provide a more technical explanation on the fidelities used for the trainings with shifts in the estimation of the coupling coefficients $\{C_{i,j}\}$.

In the absence of shifts in the estimation of the coupling coefficients, we start from a set of propagation constants $\{\beta_i^t\}$, and a set of coupling coefficients $\{C_{i,j}^t\}$. We then create $\{\beta_i^{(0)}\}$ by taking the values of $\{\beta_i^t\}$ and shifting each of them by a random amount. The strength of the shift depends on the target starting fidelity ($F_{\text{in}} \sim 0.7$) and on the structure of the circuit. We then train those propagation constants $\{\beta_i^{(0)}\}$ as explained in the main text, using as target outputs those generated by simulating the circuit using $\{\beta_i^t\}$ as the propagation constants. In such a way we obtain the trained version, which we call $\{\beta_i^f\}$.

When inserting the shifts in the coupling coefficients, we generate a new set of coupling coefficients called $\{C_{i,j}^s\}$, which can be obtained by applying a small shift $\delta C_{i,j}$ to $\{C_{i,j}^t\}$. In this situation, the parameters $\{C_{i,j}^t\}$ can be considered to represent the knowledge on the coupling coefficients of the device, obtained either via pre-calibration or from the circuit fabrication, while $\{C_{i,j}^s\}$ are the true coupling coefficients of the device.

For notation on the fidelity comparisons, we will use: $F(\{\beta_i\}_1, \{C_{i,j}\}_1, \{\beta_i\}_2, \{C_{i,j}\}_2)$ to indicate the fidelity between two unitaries, where $\{\beta_i\}_k$ and $\{C_{i,j}\}_k$ indicate the propagation constants and the coupling coefficients of the k-th unitary.

As a first step, we get the fidelity $F_{\text{base}} = F(\{\beta_i^t\}, \{C_{i,j}^s\}, \{\beta_i^t\}, \{C_{i,j}^t\})$ to compare the distance from the target transformation induced by the shift. Then, we use $\{\beta_i^t\}$ with $\{C_{i,j}^t\}$ to get the target output values. We then get $\{\beta_i^{(0)}\}$ as previously describe to obtain the $F_{\text{in}} \sim 0.7$ starting point, and perform the training using those target values. In the training loop, we use $\{\beta_i^{(0)}\}$ for the propagation constants, while we use $\{C_{i,j}^s\}$ for the coupling coefficients. At the end of the training process we obtain $\{\beta_i^f\}$. From these parameters, we can calculate the fidelity $F_{\text{noise}} = F(\{\beta_i^f\}, \{C_{i,j}^s\}, \{\beta_i^t\}, \{C_{i,j}^t\})$ between the final trained circuit and the target obtained from the initial knowledge on the coupling constants.

Appendix E: Classical Machine Learning Model

In this subsection, we will give some additional details on the structure and training of the classical neural network we used in Section 5 to find the starting parameters for a given target unitary.

Since the values of the unitary are complex, we first need to split every complex value into two real ones for simplicity. This choice is performed considering that, although complex-valued neural networks have been defined [60], they do present some unique challenges. Such kind of neural networks are still partially unexplored, and thus fewer libraries optimized for their training are available. For our network, we use a multilayer perceptron with dense layers. If m is the number of modes of the interferometer, we get that the input layer of the network has size $2m^2$. The output layer, on the other hand, has the same size as the number of trainable parameters of the chosen optical interferometer. Conversely, the number and size of the hidden layers varies with the size of the interferometer. For instance, with the ten mode chip we use four hidden layers, with width going from 256 to 64 nodes per layer. In the case of the 20 mode chip, we add at the beginning an additional layer with 1024 nodes.

The inputs parameters of the network are normalized. Conversely, we use a linear transformation to map the output to the range of values [0,1], and use a sigmoid activation function on the last layer. Then, the output values are reverted back in the original range. For all the other layers, we use the Exponential Linear Units (ELU) activation function [61]. The used loss function is the Mean Squared Error MSE $= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$, being $\hat{y}_i$ the output parameters of the network, and y_i the target parameters, corresponding to those of the input unitary. For the optimizer, we use Adam [62].

We use dropout [63] to minimize the risk of overfitting, and improve stability of the trained model. As a further refinement, during the first part of training we increased the batch size instead of decreasing the learning rate. This approach leads to a similar behavior in training, but allows for greater parallelism, and thus shorter training times [64]. Then, once the batch size is big enough, we start to decrease the learning rate.