



HAL
open science

Photonic quantum generative adversarial networks for classical data

Tigran Sedrakyan, Alexia Salavrakos

► **To cite this version:**

Tigran Sedrakyan, Alexia Salavrakos. Photonic quantum generative adversarial networks for classical data. *Optica Quantum*, 2024, 2 (6), pp.458–467. 10.1364/OPTICAQ.530346 . hal-04828031v1

HAL Id: hal-04828031

<https://hal.science/hal-04828031v1>

Submitted on 17 Dec 2024 (v1), last revised 17 Dec 2024 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.



Distributed under a Creative Commons Attribution 4.0 International License

Photonic quantum generative adversarial networks for classical data

Tigran Sedrakyan^{1,2} and Alexia Salavrakos¹

¹*Quandela, 7 Rue Léonard de Vinci, 91300 Massy, France*

²*Sorbonne Université CNRS, LIP6, F-75005 Paris, France*

When Generative Adversarial Networks (GANs) first emerged, they marked a breakthrough in the field of classical machine learning. Researchers have since designed quantum versions of the algorithm, both for the generation of classical and quantum data, but most work so far has focused on qubit-based architectures. In this article, we focus on photonic quantum computing and present a quantum GAN based on linear optical circuits and Fock-space encoding for the generation of classical data. We explore the trainability and the performance of the model in a proof-of-concept image generation scenario. We then conduct an experiment where we train our quantum GAN on Quandela’s photonic quantum processor *Ascella*.

I. INTRODUCTION

Photonic quantum hardware represents one of the most promising paths for the realization of a quantum computer. There is a strong potential for scaling in the number of qubits [1], and various computation models and architectures have been designed for photonics [2, 3]. Moreover, the possibility of demonstrating a near-term computational advantage in tasks such as boson sampling [4] makes it a noteworthy candidate for Noisy Intermediate-Scale Quantum (NISQ) technology [5]. Even though building photonic circuits with large numbers of single photons and optical modes is challenging given current technology, the rate of hardware advancement in the field is very encouraging [6].

Nevertheless, little effort has been dedicated so far to exploiting photonic-native architectures for machine learning tasks, i.e. architectures where the components are single photon sources, photon detectors, linear optical circuits with beam-splitters and phase-shifters, and where the problem is encoded in the Fock space. Some of the existing works concentrate on discriminative learning [7–9], but generative learning models remain mostly understudied [10, 11], despite the great potential shown by classical generative models in recent years. While Fock-space based models do not consist of traditional qubits, they do exhibit quantum properties which could be harnessed in machine learning tasks.

Our work sets out to explore photonic quantum computing for the generation of classical data. In particular, we propose a quantum generative adversarial network (QGAN) where the generator network is a variational quantum circuit [12]. We train our model on the MNIST [13] dataset of handwritten digits in reduced dimension, using a patch-based image generation approach, in both ideal and noisy settings. Additionally, we run the full training procedure as a physical experiment on Quandela’s quantum processing unit *Ascella* introduced in [9]. Our work is a proof-of-concept demonstration that photonic quantum adversarial models can be trained to generate classical data.

II. BACKGROUND

Generative models are designed to produce previously unseen data that follow certain patterns. Their training consists in feeding the model some target training samples that are representative of the desired outcome, and optimizing the model so that its outputs grow closer to these target samples. This corresponds to learning the underlying distribution from which the training samples are drawn, i.e. the data generating distribution. Generative models have a long-standing history, however, only recent advances in deep neural networks enabled the creation of deep generative models (DGMs), such as GANs, as well as variational autoencoders (VAEs), autoregressive and diffusion models [14–17]. Among other factors, these advances were made possible due to training techniques and properties of deep neural networks [18].

A. Classical GANs

In this work, we focus on GANs as they are realizable on a relatively small scale and can be trained efficiently. First proposed by [14], GANs marked an important milestone in the field of generative learning models. The primary concept of adversarial learning consists of two competing deep neural networks - the generator, often denoted as G , and the discriminator D . The generator accomplishes the task of data generation, by transforming the noise $\mathbf{z} \sim p_z(\mathbf{z})$ sampled from the latent space \mathcal{Z} (also known as noise prior) into a fake data sample. Then, both generator and discriminator networks compete against each other in an adversarial zero-sum game, where the generator is trying to produce fake samples close to the real target samples drawn from the data generating distribution $\mathbf{x} \sim p_{data}(\mathbf{x})$, and the discriminator is trying to classify the real samples from the fake ones. The process is repeated iteratively until the generator starts producing realistic results, which may correspond to a Nash equilibrium being reached for the zero-sum game [19].

The learning process tries to maximize the loss value across the discriminator parameters, so that the discrim-

inator is able to distinguish between the fake and real data. At the same time, it tries to minimize the loss over the generator parameters, so as to generate more realistic samples and confuse the discriminator. Mathematically, this is equivalent to a min-max optimization of a loss function $L(D, G)$ defined on the discriminator and generator models:

$$\min_G \max_D L(D, G), \quad (1)$$

where:

$$L(D, G) = \mathbb{E}_{\mathbf{x} \sim p_{data}(\mathbf{x})} [\log(D(\mathbf{x}))] + \mathbb{E}_{\mathbf{z} \sim p_z(\mathbf{z})} [\log(1 - D(G(\mathbf{z})))] \quad (2)$$

This can be expanded into the problem of maximizing two separate loss functions, the generator loss L_G and the discriminator loss L_D :

$$\max_{\theta_G} L_G \text{ and } \max_{\theta_D} L_D, \quad (3)$$

where:

$$L_G = \frac{1}{n} \sum_{i=1}^n \log(D(G(\mathbf{z}_i))) \quad (4)$$

$$L_D = \frac{1}{n} \sum_{i=1}^n [\log(D(\mathbf{x}_i)) + \log(1 - D(G(\mathbf{z}_i)))] ,$$

with n the number of training samples from the dataset, and \mathbf{x}_i and \mathbf{z}_i respectively the i -th real and noise samples.

In practice, a batch of noise samples from the latent space is supplied to the generator. It produces a batch of results, which are then used for the discriminator. Simultaneously, the discriminator is supplied a batch of samples from the training dataset. The loss is therefore computed by averaging over the batch, which has the added advantage of stabilizing the learning process. For every learning iteration, optimization steps are performed first for the discriminator, and then for the generator, using gradient descent (or ascent), where the gradient is computed using backpropagation. The number of optimization steps k for the discriminator depends on the specific use case. The full training algorithm is shown Appendix A.

B. Quantum GANs

Quantum Generative Adversarial Networks (QGANs) were introduced in [20, 21] as a quantum alternative to GANs. Rather than a single architecture, they consist of a set of concepts where at least one, if not both of the components - the generator and the discriminator - possess a certain degree of quantum capabilities.

Moreover, not only the networks, but also the data or type of problem, can be quantum or classical, as is discussed in details in [20]. Additionally, the latent space

can also be generated by a quantum source. References [22] and [23] consider the alternative case where only the latent space of an otherwise fully classical GAN is generated quantumly, in qubit-based and photonic-native scenarios respectively. In [21], an early example for the generation of quantum data, i.e. the generation of a quantum state, is presented. The problem is defined so that in practice, the generator learns how to implement a CNOT quantum gate. A photonic example for the generation of quantum data was studied recently in [24]. For the generation of classical data with a QGAN, qubit-based models were developed in various works [25–28], and a model where the generator is based on the combination of a linear optical circuit and a neural network was very recently introduced in [29].

Here, we focus on the latter task of generating classical data, with photonic quantum circuits as a resource. In our scenario, the generator is a fully quantum variational circuit, while the discriminator is a fully classical discriminative neural network. The communication between both networks happens by obtaining classical samples from the output measurements of the generator, and feeding them to the discriminator, along with the target data. The rest of the training progresses as for classical GANs: the discriminator is optimized first, followed by the generator, iteratively.

C. Image generation and patch-based approach

The specific task we consider for our QGAN is to generate images. We use a patch-based approach in order to exploit only a small number of photons and modes, and thus make our scheme easier to execute experimentally. The patch-based approach involves generating parts of larger images from separate quantum generators, which we call sub-generators, and combining or stacking the outputs together to obtain the full image. Such an approach has previously been shown to work on the 8×8 MNIST dataset in [26].

We focus on this same dataset in our work. It is a downsampled version of the popular 28×28 MNIST dataset of handwritten digits, which was originally used for small-scale classical generative models. It consists of a collection of digit images ranging from 0 to 9, each image being of size 8×8 . The dataset contains approximately 560 entries for each digit, with 5621 datapoints in total. Each datapoint consists of 64 pixels, and each pixel has continuous intensity value in the interval $[0, 1]$, with 0 being fully black pixels and 1 being fully white ones. Some examples, sampled randomly, are shown in Figure 1. While the downscaling causes lower quality, the digits still closely resemble actual handwritten numbers and provide enough diversity for the tasks discussed in this work.



Figure 1. Randomly sampled entries from the 8×8 MNIST dataset. Each row corresponds to a separate digit, sorted in increasing order 0-9.

III. A PHOTONIC QGAN

We present our proposal for a photonic QGAN in Figure 2, where the quantum generator is implemented using linear optical variational quantum circuits. Having a photonic-native model means that the circuit ansatz consists of optical modes with parametrized phase shifters and beam-splitters, which is reflected in Figure 4. In this framework, we consider as the input and output states of the circuit the Fock states of n photons in m modes, as in [7]. We denote an input Fock state as $|\vec{n}_{in}\rangle = |n_1^{in}, \dots, n_m^{in}\rangle$, where n_i^{in} indicates the number of photons in mode i . Naturally, $\sum_i n_i^{in} = n$. Likewise, we can write an output Fock state as $|\vec{n}_{out}\rangle = |n_1^{out}, \dots, n_m^{out}\rangle$: they are detected as arrangements of photons in the output modes, which we denote $s = (n_1^{out}, \dots, n_m^{out})$. If there is no photon loss, the n_i^{out} sum to n as well.

An obvious way to design the quantum generator is to consider that one output state corresponds to one data sample, and to define a mapping between the Fock states and the space of the training data. As a simple example, let us imagine that we want to generate integers between 0 and 100, according to a certain target data distribution. We can then choose the number of modes m and the number of photons n such that there are at least 100 possible output Fock states, and map each output state to an integer. In this scenario, one run of the quantum circuit produces one sample. This approach, which we could call sample-based, is used for instance in related work on photonic quantum circuit Born machines [30].

However, while we are limited to small scale devices, such as the 12 modes and 6 photons of the *Ascella* processor [9], using this approach also means that we are restricted in the type of datasets that we can consider. Let us suppose that we aim to generate 8×8 MNIST digits, and that we use the patch-based approach as mentioned

in the previous section with four patches, so that each circuit must generate images of 16 pixels at a time. If each pixel only had two intensity values (black or white), the dimension of the resulting space would be 2^{16} . This is already beyond what we could implement on *Ascella*, and even more so if we considered the actual range of pixel intensities of the digit images. For this reason, and for the purpose of this work, we propose an alternative new mapping in the next section.

A. A distribution-based mapping

In this approach, we compute the probability distribution on the output Fock states by performing several thousands of measurement shots at the end of the generator circuit. This discrete output distribution is then mapped to a discrete distribution on integers. If needed, binning may be performed so that several output Fock states correspond to the same integer. The index of a bin, i.e. the integer, corresponds to the location of a pixel on the image, while the probability of the bin corresponds to the pixel intensity, as shown in Figure 3. This allows us to obtain continuous pixel intensity values. To cover their full range, the probability values are renormalized to the interval $[0, 1]$ using min-max normalization.

It is important to note that the number of possible output states of the generator does not always match the number of pixels necessary for the image or the patch, so the output distribution of each sub-generator may be trimmed equally on each tail, under the assumption that tails of distributions do not carry much information.

When transforming Fock states to integers, while we can apply an arbitrary mapping scheme, it would intuitively make sense if photon arrangements physically close to each other in the device would correspond to integers that are close as well. We thus consider outputs with the most number of photons in the rightmost modes as closer to 0 in their integer mapping. Moreover, the larger the number of photons in the rightmost mode the smaller is the mapped integer. Correspondingly, states with a larger number of photons in the leftmost modes are considered further away from 0. Naturally, the number of available integers corresponds to the number of distinguishable states.

This approach is most efficient when photon number resolving (PNR) detectors are available. One of the main advantages photon number resolution provides is that it allows us to observe a larger amount of output states for given values of m and n . However, PNR detectors remain difficult to design with current technology. With threshold detectors, the only accessible values are binary - 0 (no photon) or 1 (click, i.e. presence of photons). Learning is of course still possible, but the mapping differs since several states with photon bunching are indistinguishable from each other. A sample mapping for 3 modes and 3 photons is shown in Table I.

This mapping assumes ideal conditions without pho-

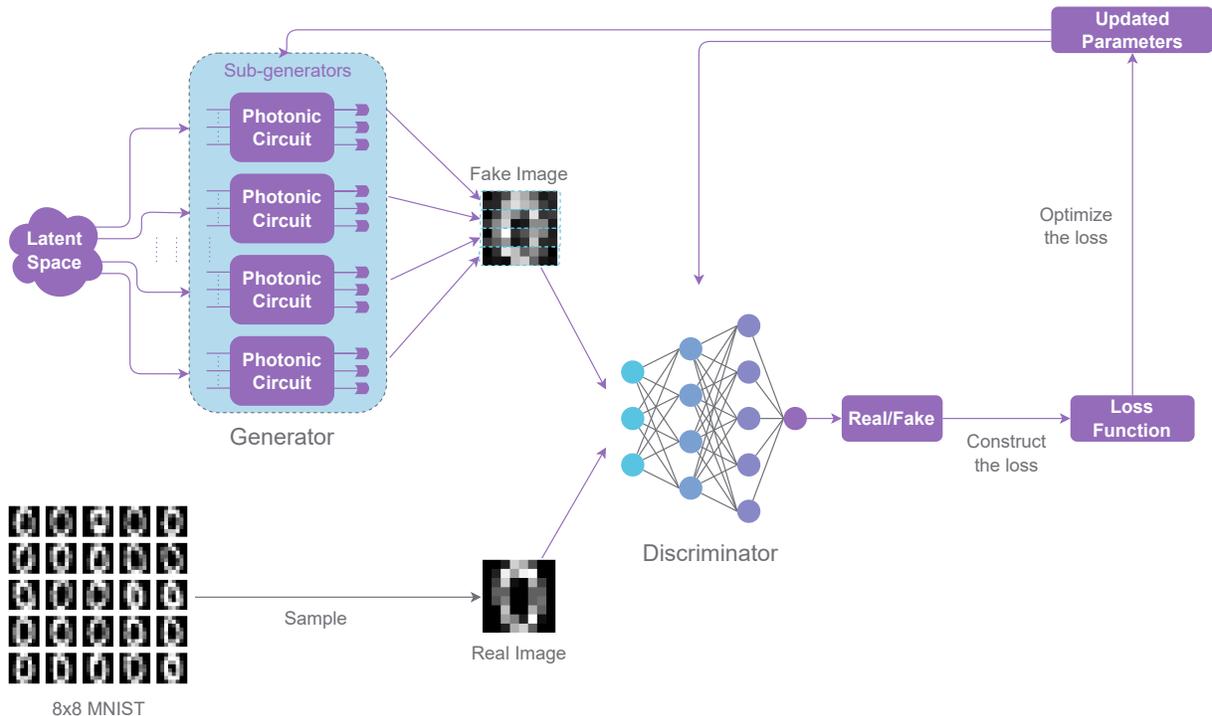


Figure 2. Proposal for image generation with a photonic QGAN. Noise from the latent space is fed into each sub-generator of the patch-based approach. These are variational photonic quantum circuits detailed in Figure 5. The output distribution of the sub-generators is mapped to image pixels, which are then recombined together to form a complete image, following the patch-based approach. The fake images are provided to the discriminator, along with the real images. The discriminator is a classical neural network, and classifies the image as real or fake. Based on these results, the loss is constructed as per Equation (4). After optimizing the loss, the parameters of the generator and the discriminator are updated.



Figure 3. Distribution-based mapping. Each output Fock state observed as arrangement $s = (n_1^{out}, \dots, n_m^{out})$ is mapped to a pixel number in the image, and the associated estimated probability is mapped to the intensity of the pixel.

ton loss. In noisy settings, the mapping does not need to be updated if PNR is available, since lossy states can be properly detected and filtered out of the final distribution with postselection. However, with threshold detectors, photon loss introduces an ambiguity in the output distribution and another mapping is necessary. In practice, lossless states with photon bunching cannot be distinguished from lossy states and they are both discarded in postselection. For the case of 3 photons with 3 modes, the number of output states reduces to one - $|1, 1, 1\rangle$. In such a situation, m or n must be increased to recover the necessary amount of integers for the image size.

Integer	PNR	No PNR (state)	No PNR (pattern)
0	$ 0, 0, 3\rangle$	$ 0, 0, 3\rangle$	$ 0, 0, \text{click}\rangle$
1	$ 0, 1, 2\rangle$	$ 0, 3, 0\rangle$	$ 0, \text{click}, 0\rangle$
2	$ 0, 2, 1\rangle$	$ 0, 2, 1\rangle, 0, 1, 2\rangle$	$ 0, \text{click}, \text{click}\rangle$
3	$ 0, 3, 0\rangle$	$ 3, 0, 0\rangle$	$ \text{click}, 0, 0\rangle$
4	$ 1, 0, 2\rangle$	$ 2, 0, 1\rangle, 1, 0, 2\rangle$	$ \text{click}, 0, \text{click}\rangle$
5	$ 1, 1, 1\rangle$	$ 2, 1, 0\rangle, 1, 2, 0\rangle$	$ \text{click}, \text{click}, 0\rangle$
6	$ 1, 2, 0\rangle$	$ 1, 1, 1\rangle$	$ \text{click}, \text{click}, \text{click}\rangle$
7	$ 2, 0, 1\rangle$	-	-
8	$ 2, 1, 0\rangle$	-	-
9	$ 3, 0, 0\rangle$	-	-

Table I. Fock state to integer mapping table for a noiseless setup with 3 modes and 3 photons. The last column clarifies the detection pattern without PNR.

B. The ansatz

In our patch-based approach, images are generated in horizontal patches by separate sub-generators and eventually stacked vertically to form the full image. Each sub-generator corresponds to a linear optical quantum circuit. When designing the ansatz, we considered setups with

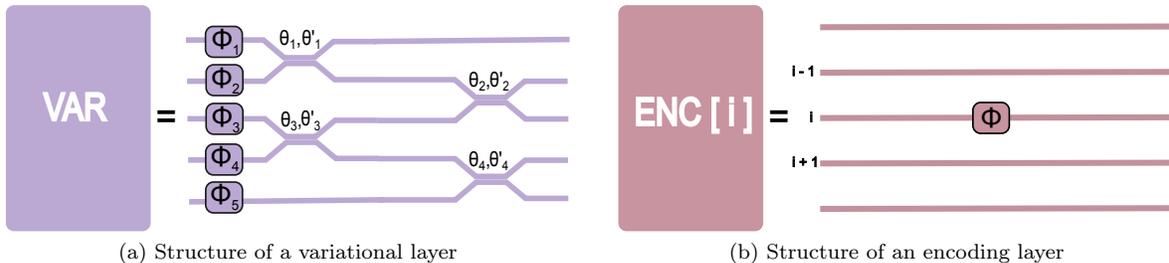


Figure 4. Structures of the variational layers and encoding or noise layers. Phase shifters are depicted by squares and beam splitters by crossing between the modes. Parameters of the variational layers are trainable and parameters of the encoding layers are sampled from the latent space.

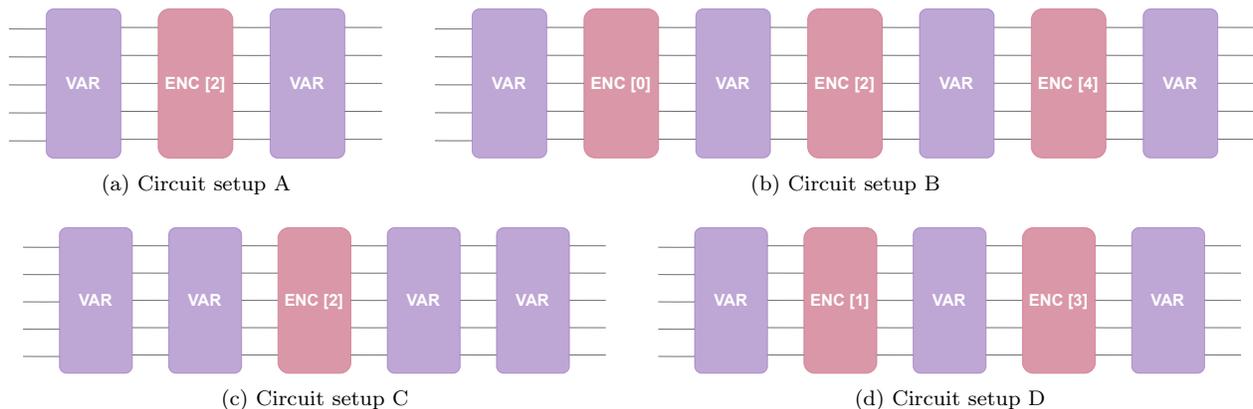


Figure 5. Sub-generator circuit structures used for photonic QGAN training. Layer structures are shown in Figure 4.

two quantum sub-generators, where each sub-generator generates patches of 32 pixels, as well as setups with four quantum sub-generators, where the patches contain 16 pixels.

In a given setup, all the sub-generators of the generator have the same structure. This structure consists of variational layers, and encoding or noise layers, as described in Figure 4. The variational layers contain the parameters that are optimized during the training of the model and their structure is inspired from [31, 32]. The encoding or noise layers are used to introduce noise z into the model (here sampled from a normal distribution). These encoding-reuploading layers consist of phase-shifters. They ensure that the resulting distribution over pixel intensities is different for each input noise sample, and that the model can thus generate a variety of data points, as well as generalize better. In general, noise-reuploading adds to the non-linearity of the input-output mapping, improving the diversity in the generated images and encouraging the model to learn patterns rather than memorizing them [33, 34].

We explored several structures for the sub-generator circuits. This structure can be adjusted by alternating the number and the arrangement of the variational and

encoding layers. We display in Figure 5 the circuits that we found to be fairly efficient at solving our image generation task. The smallest circuit only has one encoding layer, while the largest one has three. The number of modes may vary compared to the circuits displayed in Figures 4 and 5, but the general layer structure is preserved. It is important to note that considering the empiric nature of the findings, the circuit configurations are not guaranteed to be optimal, but are rather a heuristic combination of an educated guess and non-exhaustive search.

C. Training and optimization

The training of photonic QGANs progresses similarly to the regular GANs, as in Algorithm 1. The classical discriminator is trained first for one step using backpropagation-enabled stochastic gradient ascent to maximize L_D , after which the quantum generator is trained for several steps of Simultaneous Perturbation Stochastic Approximation (SPSA) [35] iterations, to optimize L_G . All the parameters are updated, and this process is repeated until the maximum number of training

epochs is reached.

SPSA is an optimization technique based on a stochastic approximation of the gradient. Due to the fact that this approximation is an almost unbiased estimator of the gradient, convergence of the method is guaranteed under reasonably general conditions. The primary advantage of the SPSA algorithm lies in the amount of circuit evaluations necessary for approximating the gradient and its robustness to noise, including the noise induced by quantum sources [36, 37]. SPSA requires only 2 evaluations, which allows cutting back on costly reconfiguration of linear optical gates and considerably reduces the duration of both simulations and experiments. Indeed, the number of training steps of the models scale only constantly (rather than linearly) with the number of parameters.

We initialize the parameters for SPSA in a way which allows the initial generator pseudo-gradients to be large enough for a successful kick-start to the optimization. In order to achieve this, parameters are initialized randomly, the initial gradients are computed and if the values are too small a reinitialization is performed. This parameter reinitialization is repeated until the starting pseudo-gradient values are in a desired range. Initialization performed in this way does not guarantee convergence, but in most cases allows the generator enough starting optimization momentum to be able to compete with the more exact gradient calculation of the classical discriminator, thus enabling a balanced training.

D. Numerical experiments: ideal simulations

We perform our numerical experiments using Qandela’s software package *Perceval* [38], designed for linear optical quantum circuits. We include an abstract pseudocode in Appendix A, and our Python code can be found in a companion Github repository [39].

Ideal conditions for simulations assume a perfect single photon source, ideal components, no photon loss, perfect detectors and the absence of sampling errors. First, we focus on the generation of digit "0" for the design of the model and the optimization of its hyperparameters. We then further study the best model for the generation of other digits, and for noisy simulations in the next sections. We include all details about the model optimization and hyperparameter search in Appendix B.

We present our results in Figure 6. The loss evolution plots describe how the values of the generator loss L_G and the discriminator loss L_D progress with the training epochs. For each configuration, we run 10 training instances, and we display the average over these runs as a bold line, and the standard deviation as a shaded area. We observe that even averaged over 10 runs, loss values have small constant fluctuations throughout the training, which is a strong characteristic of adversarial training. When the L_D loss increases, the L_G loss decreases, and vice versa. In the image evolution plots, we see that the models start with noisy outputs and that most of them

produce quite realistic "0"s by the end of the training (iteration 1500). The generated image plot shows several samples from the trained model, after the last epoch has been completed.

It appears that there is no specific loss value where the training can be stopped. However, good models generally reach an equilibrium where L_G and L_D losses start to fluctuate around the same moving average, without increasing or decreasing further. This is clearly observed in Figure 6c where both configurations converge to an equilibrium after around 1000 iterations. This equilibrium is centered around value close to 0.69, which corresponds to $\log 2$. Taking a look at loss from Equation (4), it becomes clear that $L_G = -\log 2$, when $D(G(z)) = 1/2$. That is, when the discriminator prediction about whether the fake image is real is as good as a random guess. Despite the fact this equilibrium is reached around iteration 1000, we note that further training does improve the results. Therefore, cutting off the training once the losses reach the value of $\log 2$ is not necessarily a desirable strategy. Rather, it makes sense to inspect the generated results using qualitative or quantitative metrics in order to be able to cutoff the training in due time. Quantitative assessment metrics are not discussed here, since the small scale of generated images allows for direct inspection of the results, as has been the case in many quantum generative learning works [26].

We refer the reader to Appendix B for further analysis of the results and for hypotheses as to why some configurations perform better than others. Overall, the models from Figure 6c behaved best and we select them for further simulations.

In particular, we apply the model from the left hand side of Figure 6c to the generation of other digits. We present results for some digits in Figure 7, while the rest of the digits can be found in the Github repository [39]. An additional 500 iterations were employed here in order to properly assess the convergence for different digits. When comparing to target data, we see that the model performs fairly well for most of the digits, and each sampled digit has recognizable contours. Importantly, the model either converges or is bound to do so for all the digits, which is indicated by a narrowing of the standard deviation for the L_G loss. While some digits may require more training iterations to reach an equilibrium state, we can assert that the model starts producing realistic results around iteration 1500.

Note that, for simplicity, we restricted our model to generating one digit at a time (unlike classical GANs trained on MNIST). This is a similar approach to previous QGAN proposals such as [26], but our model could be extended to a conditional QGAN in future work. It would then be able to generate all the digits depending on the input label, which could be supplied as a model input or encoded through an additional layer.

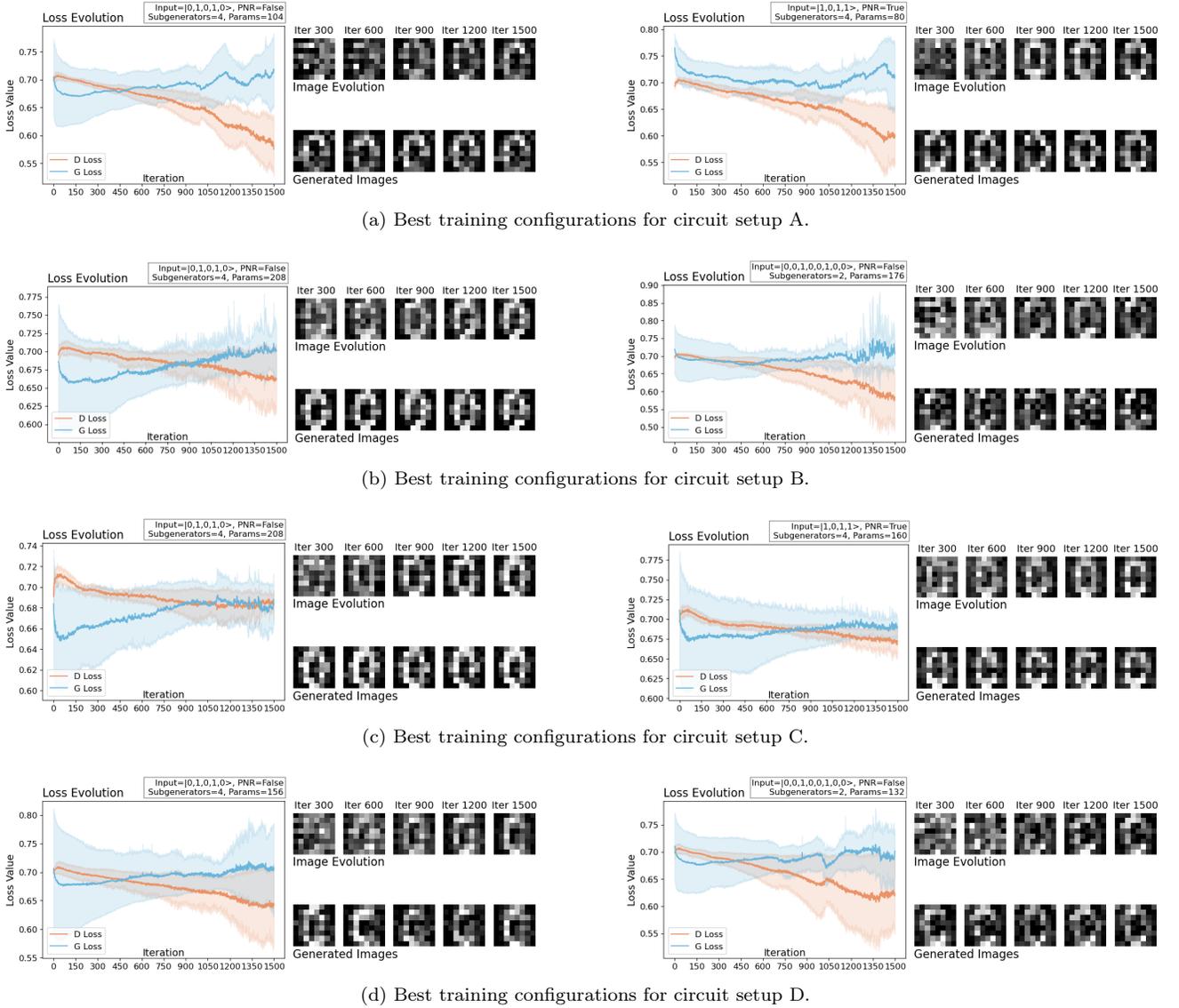


Figure 6. Best training results for different circuit setups. Each figure contains a plot with the evolution of the loss function, the evolution of the generated images as well as the final generated images of the trained model, and as a small infobox detailing the model hyperparameters. Figures on the left and right correspond to different sets of hyperparameters.

E. Noisy simulations

We now use the same model for noisy simulations. The *Perceval* package allows us to specify various parameters such as the emission probability of the source, the photon loss regime, and photon distinguishability. We set indistinguishability and photon loss to 0.92 each, so as to closely mimic the actual conditions on the *Ascella* processor as it was presented in [9].

In addition to the imperfect source and losses throughout the circuit, noisy simulations also introduce sampling error. For ideal simulations the distribution of the output with exact probabilities is directly available in *Perceval*. However, in this case the distribution is obtained by

collecting 10^5 measurement shots and postselecting lossy outputs. A discrete distribution obtained in such a way gets closer to the exact distribution with an increasing number of measurement shots. However, in an experimental setting, collecting a high number of shots requires precious time, and 10^5 shots were found to be an optimal compromise for the accuracy/training time trade-off.

As discussed in Section III A, the lack of PNR detectors combined with photon loss shrink the size of the output space. This requires us to change some hyperparameters and we display two strategies in Figure 8.

Clearly, the added noise considerably slows down the convergence. While the noiseless version reaches an equilibrium around iteration 900 on average, for the noisy ver-

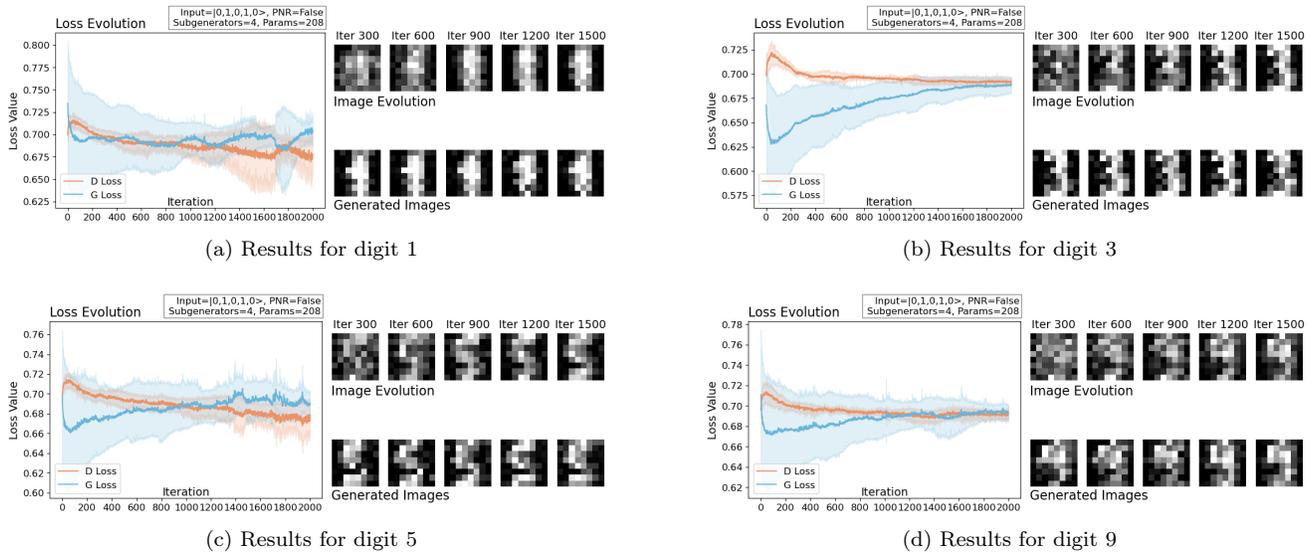


Figure 7. Training results for different digits.

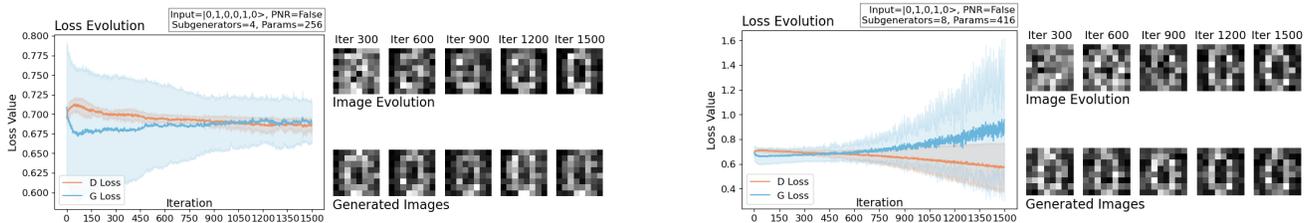


Figure 8. Training results for noisy simulations. The model on the left has a higher number of modes, and the model on the right contains more sub-generators, which compensates for a smaller output space in the lossy case.

sion on the right of Figure 8, even 1500 iterations are not enough. This version, which has more sub-generators, shows a diverging learning trend. This might be caused by the increased number of parameters, making it harder to train this model under the predefined learning rate restrictions (see Appendix B). Nevertheless, for the model on the left which has less sub-generators but a higher number of modes, one can see how the standard deviation of the generator decreases throughout the training, indicating that it is likely to eventually converge. Moreover, the generated results are satisfactory when applying a manual check. Results comprehensible to the human eye are available by iteration 600 for the ideal version, while the noisy version requires more than 1000.

Overall, training in the presence of noise and sampling errors comparable to those of a real quantum processor is still viable, albeit slower. Techniques for quantum error mitigation [40] might improve the results of noisy simulations, and could be explored in future work.

F. Physical experiment

Based on the insights gathered from our simulations, we run an experiment on Quandela’s processor *Ascella*, with our best performing model. The corresponding quantum circuit is of a reasonable depth, making it compatible with *Ascella*’s chip, which is naturally reused for each sub-generator. In order to make the duration of the experiment tractable, we decrease the total number of training iterations to 1000, with 3 SPSA steps for the generator at each iteration, which results in a total of 3000 SPSA steps. The results for one training instance are displayed in Figure 9.

While the training is slow, following the trend of noisy simulations, the results are promising. Importantly, we observe that the model is learning, with the loss functions behaving as they should, tending towards the equilibrium value, and results improving throughout the training. A point that would require improvements in future experiments is that the generated images do not present a lot of diversity. There are only minor differences between the "0"s which might be improved by adding more en-

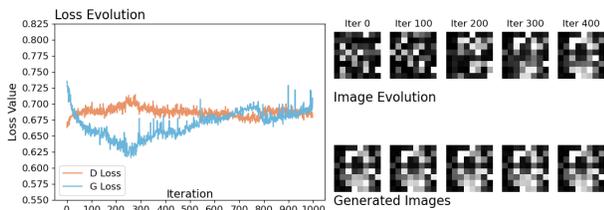


Figure 9. Training results for the experiment on the *Ascella* quantum processor.

coding phase-shifters into the ansatz. Nevertheless, the generated "0"s are of fairly good quality, with characteristic contours of the digit, demonstrating the experimental feasibility of photonic QGANs.

IV. DISCUSSION AND FUTURE WORK

In this work, we showed that photonic quantum circuits can work as a key component in a generative learning pipeline to produce images. This is in contrast to previous work where linear optical circuits were used for smaller-scale tasks such as latent space generation, without going as far as using a photonic generator. To the best of our knowledge, our experiment is the first demonstration of a photonic GAN with a fully quantum generator for classical data. Additionally, with the aim of transparency and collaboration within the quantum machine learning community, we make our code available online at [39].

Most of the QGAN literature concentrates on smaller scale models for generation of lower resolution images, such as 3×3 bars-and-stripes. However, the task of generating larger images is tractable with current hardware if the right approach is used – such as our distribution-based mapping. This highlights the importance of looking for alternative strategies when working with near-term quantum hardware. Using our mapping along with noise reuploading gave us greater flexibility, and in combination with patch-based learning the model could generate higher resolutions images.

We note that SPSA-based optimization seems to work surprisingly well on variational linear optical circuits. Especially in the noisy experimental setup, SPSA allowed for stable learning at a constant cost, albeit requiring more iterations to converge. Fine-tuning the optimiza-

tion and the model in general was an important aspect of the work. Future work may concentrate on improving the convergence rates of optimization techniques for photonic platforms, which can be used directly to improve the performance of our model. Indeed, only few works so far have focused on exploring methods like parameter-shift rules [41] in the context of quantum photonics [42]. As mentioned in the text, other improvements to this work include the extension of the model to a conditional QGAN, as well as the integration of error mitigation techniques for sources of noise such as photon loss and distinguishability [43].

Overall, our QGAN implementation is quite flexible and modular, so the model can be further explored for generation of other types classical data, by fine-tuning parameters and introducing some changes to the circuit structure. For instance, a generator may consist of sub-generators with different structures, depending on the task at hand. We could also consider a quantum discriminator and training data based on a quantum source, for the generation of quantum states, for instance. Additionally, the source of the latent space could also be changed to be quantum, by using another boson-sampling-based circuit as a source of noise, as in [23].

As is often the case in quantum machine learning, it is not always clear if a quantum model will perform better than a classical one and on which datasets that would be the case [44], although interesting alternative benchmarks such as the energy consumption of quantum versus classical computers are being explored [45]. Tasks based on quantum data clearly require a quantum model – for example if the generator learns how to implement a quantum gate or generate specific quantum states. For classical data on the other hand, a deeper investigation is required, likely related to the identification of inductive biases of quantum models. In our case in particular, identifying inductive biases of photonic quantum circuits may help us understand where photonic QGANs may be most useful.

ACKNOWLEDGEMENTS

We would like to thank Pierre-Emmanuel Emeriau for fruitful discussions and feedback on the manuscript. We acknowledge funding from the Plan France 2030 through the project OQuLus ANR-23-PETQ-0013.

[1] J. Bourassa, R. Alexander, M. Vasmer, A. Patil, I. Tzitrin, T. Matsuura, D. Su, B. Baragiola, S. Guha, G. Dauphinais, K. Sabapathy, N. Menicucci, and I. Dhand, *Quantum* **5**, 392 (2021).
 [2] R. Raussendorf and H. J. Briegel, *Phys. Rev. Lett.* **86**, 5188 (2001).

[3] S. Bartolucci, P. Birchall, and H. B. et al., *Nat. Commun.* **14** (2023), 10.1038/s41467-023-36493-1.
 [4] S. Aaronson and A. Arkhipov, in *Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing*, STOC '11 (Association for Computing Machinery, New York, NY, USA, 2011) p. 333–342.

- [5] J. Preskill, *Quantum* **2**, 79 (2018).
- [6] S. I. Davis, A. Mueller, R. Valivarthi, N. Lauk, L. Narvaez, B. Korzh, A. D. Beyer, O. Cerri, M. Colangelo, K. K. Berggren, M. D. Shaw, S. Xie, N. Sinclair, and M. Spiropulu, *Phys. Rev. Appl.* **18**, 064007 (2022).
- [7] B. Y. Gan, D. Leykam, and D. G. Angelakis, *EPJ Quantum Technology* **9**, 16 (2022).
- [8] K. Bartkiewicz, C. Gneiting, A. Černoč, K. Jiráková, K. Lemr, and F. Nori, *Scientific Reports* **10** (2020), [10.1038/s41598-020-68911-5](https://doi.org/10.1038/s41598-020-68911-5).
- [9] N. Maring, A. Fyrrillas, M. Pont, E. Ivanov, P. Stepanov, N. Margaria, W. Hease, A. Pishchagin, T. H. Au, S. Boissier, E. Bertasi, A. Baert, M. Valdivia, M. Billard, O. Acar, A. Brioussel, R. Mezher, S. C. Wein, A. Salavrakos, P. Sinnott, D. A. Fioretto, P.-E. Emeriau, N. Belabas, S. Mansfield, P. Senellart, J. Senellart, and N. Somaschi, *Nature Photonics* (2024), <https://doi.org/10.1038/s41566-024-01403-4>.
- [10] J. Tian, X. Sun, Y. Du, S. Zhao, Q. Liu, K. Zhang, W. Yi, W. Huang, C. Wang, X. Wu, M. Hsieh, T. Liu, W. Yang, and D. Tao, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **45**, 12321 (2023).
- [11] A. Sehwat, “Interferometric neural networks,” (2023), [arXiv:2310.16742 \[quant-ph\]](https://arxiv.org/abs/2310.16742).
- [12] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio, and P. J. Coles, *Nature Reviews Physics* **3**, 625 (2021).
- [13] L. Deng, *IEEE Signal Processing Magazine* **29**, 141 (2012).
- [14] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, *Commun. ACM* **63**, 139–144 (2020).
- [15] D. P. Kingma and M. Welling, in *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, edited by Y. Bengio and Y. LeCun (2014).
- [16] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, in *Advances in Neural Information Processing Systems*, Vol. 33, edited by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin (Curran Associates, Inc., 2020) pp. 1877–1901.
- [17] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, in *Proceedings of the 32nd International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 37, edited by F. Bach and D. Blei (PMLR, Lille, France, 2015) pp. 2256–2265.
- [18] L. Ruthotto and E. Haber, *GAMM-Mitteilungen* **44**, e202100008 (2021).
- [19] F. Farnia and A. Ozdaglar, in *Proceedings of the 37th International Conference on Machine Learning*, Proceedings of Machine Learning Research, Vol. 119, edited by H. D. III and A. Singh (PMLR, 2020) pp. 3029–3039.
- [20] S. Lloyd and C. Weedbrook, *Physical Review Letters* **121** (2018), [10.1103/physrevlett.121.040502](https://doi.org/10.1103/physrevlett.121.040502).
- [21] P.-L. Dallaire-Demers and N. Killoran, *Physical Review A* **98** (2018), [10.1103/physreva.98.012324](https://doi.org/10.1103/physreva.98.012324).
- [22] M. S. Rudolph, N. B. Toussaint, A. Katabarwa, S. Johri, B. Peropadre, and A. Perdomo-Ortiz, *Phys. Rev. X* **12**, 031010 (2022).
- [23] H. Wallner and W. R. Clements, in *ICLR 2023 Workshop on Physics for Machine Learning* (2023).
- [24] Y. Wang, S. Xue, Y. Wang, Y. Liu, J. Ding, W. Shi, D. Wang, Y. Liu, X. Fu, G. Huang, A. Huang, M. Deng, and J. Wu, *Optics Letters* **48**, 5197 (2023).
- [25] C. Zoufal, A. Lucchi, and S. Woerner, *npj Quantum Information* **5** (2019), [10.1038/s41534-019-0223-2](https://doi.org/10.1038/s41534-019-0223-2).
- [26] H.-L. Huang, Y. Du, M. Gong, Y. Zhao, Y. Wu, C. Wang, S. Li, F. Liang, J. Lin, Y. Xu, R. Yang, T. Liu, M.-H. Hsieh, H. Deng, H. Rong, C.-Z. Peng, C.-Y. Lu, Y.-A. Chen, D. Tao, X. Zhu, and J.-W. Pan, *Physical Review Applied* **16** (2021), [10.1103/physrevapplied.16.024051](https://doi.org/10.1103/physrevapplied.16.024051).
- [27] H. Situ, Z. He, Y. Wang, L. Li, and S. Zheng, *Information Sciences* **538**, 193–208 (2020).
- [28] J. Romero and A. Aspuru-Guzik, *Advanced Quantum Technologies* **4**, 2000003 (2021).
- [29] H. Ma, L. Ye, F. Ruan, Z. Zhao, M. Li, Y. Wang, and J. Yang, (2024), [arXiv:2404.05921 \[quant-ph\]](https://arxiv.org/abs/2404.05921).
- [30] A. Salavrakos, T. Sedrakyan, J. Mills, and R. Mezher, (2024), [arXiv:2405.02277 \[quant-ph\]](https://arxiv.org/abs/2405.02277).
- [31] S. Shankar and D. Towsley, in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases. Springer* (2022).
- [32] J. Shi, Y. Tang, Y. Lu, Y. Feng, R. Shi, and S. Zhang, *IEEE Transactions on Knowledge and Data Engineering* **35**, 1965 (2023).
- [33] S. Chaudhary, P. Huembeli, I. MacCormack, T. L. Patti, J. Kossaifi, and A. Galda, *Quantum Science and Technology* **8**, 035002 (2023).
- [34] A. Pérez-Salinas, A. Cervera-Lierta, E. Gil-Fuster, and J. I. Latorre, *Quantum* **4**, 226 (2020).
- [35] J. Spall, *IEEE Transactions on Aerospace and Electronic Systems* **34**, 817 (1998).
- [36] M. Wiedmann, M. Hölle, M. Periyasamy, N. Meyer, C. Ufrecht, D. D. Scherer, A. Plinge, and C. Mutschler, (2023), [arXiv:2305.00224 \[quant-ph\]](https://arxiv.org/abs/2305.00224).
- [37] F. Sauvage and F. Mintert, *PRX Quantum* **1** (2020), [10.1103/PRXQuantum.1.020322](https://doi.org/10.1103/PRXQuantum.1.020322).
- [38] N. Heurtel, A. Fyrrillas, G. d. Gliniasty, R. Le Bihan, S. Malherbe, M. Pailhas, E. Bertasi, B. Bourdoncle, P.-E. Emeriau, R. Mezher, L. Music, N. Belabas, B. Valiron, P. Senellart, S. Mansfield, and J. Senellart, *Quantum* **7**, 931 (2023).
- [39] T. Sedrakyan, “<https://github.com/Quandela/photon-qgan>,” (2024).
- [40] S. Endo, Z. Cai, S. C. Benjamin, and X. Yuan, *Journal of the Physical Society of Japan* **90**, 032001 (2021).
- [41] M. Schuld, V. Bergholm, C. Gogolin, J. Izaac, and N. Killoran, *Physical Review A* **99** (2019), [10.1103/physreva.99.032331](https://doi.org/10.1103/physreva.99.032331).
- [42] G. de Felice and C. Cortlett, (2024), [arXiv:2401.07997 \[quant-ph\]](https://arxiv.org/abs/2401.07997).
- [43] S. Endo, S. C. Benjamin, and Y. Li, *Physical Review X* **8** (2018), [10.1103/physrevx.8.031027](https://doi.org/10.1103/physrevx.8.031027).
- [44] J. Bowles, S. Ahmed, and M. Schuld, (2024), [arXiv:2403.07059 \[quant-ph\]](https://arxiv.org/abs/2403.07059).
- [45] A. Auffèves, *PRX Quantum* **3**, 020101 (2022).

Appendix A: Algorithms

Algorithm 1: GAN Training

Data: Neural networks G and D , data generating distribution $p_{data}(\mathbf{x})$
Result: G capable of sampling from $p_{data}(\mathbf{x})$
for *number of training iterations* **do**
 for k *steps* **do**
 Sample a batch of noise samples $\mathbf{z} \sim p_z(\mathbf{z})$
 Sample a batch of real data $\mathbf{x} \sim p_{data}(\mathbf{x})$
 Generate fake data batch $G(\mathbf{z})$ Get predictions for fake and real data batches: $D(G(\mathbf{z}))$ and $D(\mathbf{x})$
 Calculate and maximize L_D according to 4 Update parameters of D
 end
 Sample a batch of noise samples $\mathbf{z} \sim p_z(\mathbf{z})$
 Get predictions for the fake data batch $D(G(\mathbf{z}))$
 Calculate and maximize L_G according to 4
 Update parameters of G
end
return G

Algorithm 2: Patch-based image generation using photonic quantum circuits

Data: Sub-generator count c , sub-generator parameters $params$
 Noise batch z , input Fock state $|*\rangle \psi$, measurement shot count m
Result: A batch of fake images

Initialize an empty batch of fake images $fakebatch$
Initialize c sub-generator circuits according to $params$
foreach *noise sample* z_i *in* z **do**
 Initialize an empty fake image $fakeimage$
 foreach *sub-generator* g **do**
 Encode z_i into g through noise reuploading phase-shifter layers
 Run the circuit g with input $|*\rangle \psi$, performing m measurement shots
 Build the discrete distribution of the output Fock states
 Map the distribution to an integer distribution.
 Construct the patch $g(z)$ by renormalizing the integer distribution to the interval $[0, 1]$.
 Add $g(z)$ to $fakeimage$
 end
 Add $fakeimage$ to $fakebatch$
end
return $fakebatch$

Appendix B: Model details

In section III D, we optimize the model in ideal conditions before proceeding with noisy simulations and the experiment. This optimization is done through hyperparameter search, by testing and choosing the models that showed the best potential and performance. Batch size is one of the fixed hyperparameters over our search. All generators are supplied 4 noise samples and therefore produce a batch of 4 results over which averaging is done for loss evaluation. This batch size was found to be optimal in terms of computation time and implications on the stability of the training. Another training hyperparameter which is fixed across all models is the learning rate, for both the generator and the discriminator. For the discriminator, we chose a constant learning rate of 0.002. For the generator, the adaptive learning rate of SPSA is employed, with the initial rate dependent on the initial gradient values. However, the number of optimization steps per each discriminator step is fixed to be 7. A total of 10500 SPSA iterations are used, with 7 steps corresponding to a single step of gradient ascent of the discriminator. This means that the total optimization routine has $10500/7 = 1500$ steps.

Four hyperparameters were picked to be variable: the number of sub-generators, their circuit structure as shown in Figure 5, their input state, and the PNR capabilities of the detectors. We do a grid search on these configurations, and

we present the most interesting results in Figures 6a - 6d from the main text. The full list of the results is presented in the companion Github repository [39] for curious readers.

From our results, we observe that deeper circuits do not necessarily mean improved results. While there is certainly an improvement when increasing the number of layers from 3 to 5, this trend does not uphold for deeper circuits. As a matter of fact, the deepest circuit from the Figure 6b performs worse than the shallower analogs. This might be attributed to the difficulties associated with the training of more complex quantum circuits.

We can also see how the number of noise encoding layers affects the diversity of the generated results. Since the encoding layers are the interface for supplying the classical noise, increasing their amount means increasing the classical noise on the circuits. This has an adverse effect, however, wherein the quality of the results deteriorates with the number of encoding layers. The transition between Figures 6c and 6d indicates a noticeable decrease in image quality with the introduction of an additional encoding layer, in otherwise identical setups. Luckily, even one encoding layer provides enough diversity in the images. The results generated by such models with one encoding layer (Figures 6a and 6c) may initially seem less diverse, because they preserve the same shape for all generated samples. But results are indeed diverse, as can be seen through the placements of intensity value accents in generated samples. Most importantly, simple checks show that the generated results do not replicate original training samples, but rather generate previously unseen "0"s, which means that these models generalize well.

Finally, we note that our models benefit from the availability of PNR detectors. In otherwise identical setups in Figure 6a, PNR detectors make it possible to generate similar results by exploiting fewer modes and more photons, as expected. On the other hand, if the number of modes and photons is increased while PNR is made available, this could also potentially open up some prospects for larger-scale models and working with higher resolution images.